**ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ**

Навчально-науковий інститут Інформаційних технологій

Кафедра інженерії програмного забезпечення

# Пояснювальна записка

до бакалаврської роботи
на ступінь вищої освіти бакалавр

на тему: «**РОЗРОБКА TELEGRAM-БОТУ ДЛЯ ПЕРЕВІРКИ КІЛЬКОСТІ КАЛОРІЙ У ЇЖІ МОВОЮ PYTHON**»

Виконала: студент 4 курсу, групи ПД-42
спеціальності
121 Інженерія програмного забезпечення
(шифр і назва спеціальності)
Прокопець Дана Сергіївна
(прізвище та ініціали)
Керівник Гаманюк І.М.
(прізвище та ініціали)
Рецензент
(прізвище та ініціали)

Київ – 2023

**ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ**

**Навчально-науковий інститут Інформаційних технологій**

Кафедра Інженерії програмного забезпечення_____
Ступінь вищої освіти – «Бакалавр»_____
Напрям підготовки – 121 – «Інженерія програмного забезпечення»_____

**ЗАТВЕРДЖУЮ**
Завідувач кафедри
Інженерії програмного
забезпечення
_____О.В. Негоденко
“____” _____ 2023 року

# З А В Д А Н Н Я
## НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

**Прокопець Дана Сергіївна**

(прізвище, ім’я, по батькові)

1. Тема роботи: «Розробка Telegram-боту для перевірки кількості калорій у їжі мовою Python»
Керівник роботи ____Гаманюк Ігор Михайлович, старший викладач кафедри___,
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від 24 лютого 2023 р. №26.
2. Строк подання студентом роботи 01 червня 2023 року_____
3. Вхідні дані до роботи:
*Принципи функціонування Telegram-ботів;*_____
*Середовище для програмування мовою Python.*_____
*Матеріали переддипломної практики.*_____
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити).
*4.1. Принцип роботи бота в мережі Telegram.*_____
*4.2. Переваги IDE для професійної розробки PyCharm.*_____
*4.3. Дослідження команд та синтаксису при використанні мови Python.*_____
*4.4. Написання коду та тестування кінцевого продукту.*_____

5. Перелік графічного матеріалу (презентація)_____
5.1 Мета, об’єкта та предмет дослідження_____
5.2 Telegram-бот _____
5.3 IDE PyCharm. Можливості та переваги _____
5.4 Синтаксис та команди в Python_____
5.5 Кінцевий результат_____

5.6 Перспективи розвитку Telegram-ботів

6. Дата видачі завдання 24.02.2023

## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів бакалаврської роботи | Строк виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Постановка задачі | 27.03.2023 | |
| 2 | Підбір науково-технічної літератури | 03.04.2023 | |
| 3 | Дослідження джерел інформації | 10.04.2023 | |
| 4 | Вибір необхідної інформації | 17.04.2023 | |
| 5 | Підбір необхідного програмного забезпечення | 24.04.2023 | |
| 6 | Написання програмного коду | 01.05.2023 | |
| 7 | Тестування Telegram-боту | 08.05.2023 | |
| 8 | Формування висновків | 12.05.2023 | |
| 9 | Попередній захист роботи | 19.05.2023 | |
| 10 | Подання роботи в деканат | 25.05.2023 | |

Студент _____ Д.С. Прокопець
(підпис) (прізвище та ініціали)

Керівник роботи _____ І.М. Гаманюк
(підпис) (прізвище та ініціали)

# РЕФЕРАТ

Текстова частина бакалаврської роботи: 77 с., 50 рис., 3 табл., 2 дод., 19 джерел.

TELEGRAM-БОТ, ЧАТ-БОТ, МЕСЕНДЖЕР, СОЦІАЛЬНА МЕРЕЖА, КОМАНДА, PYCHARM, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, PYTHON

Об`єкт дослідження – процес перевірки кількості калорій у їжі.

Предмет дослідження – Telegram-бот для перевірки кількості калорій у їжі.

Мета роботи – автоматизація процесу розрахунку калорій в їжі шляхом впровадження застосунку для перевірки кількості калорій у їжі.

Методи дослідження – методи теорії інформації, синтез, тестування програмного забезпечення, обробка та аналіз отриманих результатів.

У роботі проведено аналіз ставлення суспільства до проблеми здорового харчування та підбору такого раціону, що буде найбільше пасувати конкретній людині. Задля цього виконано огляд існуючого програмного забезпечення для підбору найпростішого в експлуатації та освоєнні, щоб на базі нього створити необхідний продукт. Здійснено розробку автоматизованої системи (бота) на базі мережі Telegram, яка відповідає заданим в темі дипломної роботи вимогам.

Розроблено Telegram-бота мовою програмування Python на базі спеціалізованого програмного забезпечення PyCharm, який дозволяє отримувати та зберігати дані щодо кількості калорій в продукті харчування. Проведено аналіз отриманих результатів та визначено найкращий спосіб оформлення та використання Telegram-бота, а також визначено оптимальний шлях доповнення бази даних, якою наділений бот.

**ЗМІСТ**

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API (Application Programming Interface) - програмний інтерфейс, що дозволяє зв’язувати різні додатки і був створений для спрощення та прискорення розробки.

IDE (Integrated Development Environment) - інтегроване середовище розробки.

JavaScript – мова програмування, що найчастіше використовується в браузерах як мова сценаріїв для надання інтерактивності веб-сторінкам.

PEP8 (Python Enhancement Proposal) – це офіційне керівництво по стилю коду Python.

Pep8 – це модуль, який дозволяє перевіряти, чи відповідає код стандарту PEP8.

PyCharm – кросплатформенна IDE для мови програмування Python.

Python – мова програмування високого рівня зі строгою динамічною типізацією.

Telegram-бот – чат-бот в месенджері Telegram.

Месенджер – програма, веб-сервіс або мобільний додаток для миттєвого обміну повідомленнями.

Сегментація – технологія, яка дозволяє поділити адресний простір процесу на кілька сегментів.

# ВСТУП

Станом на січень 2023-го року месенджер Telegram налічує більш, ніж 700 млн. активних користувачів щомісяця по всій земній кулі, що дозволило йому піднятись на 9-й рядок в списку найпопулярніших месенджерів в світі, обігнавши китайський QQ та американський Snapchat. Далеко не в останню чергу на це позитивно вплинуло використання об’єкту дослідження даної дипломної роботи – Telegram-ботів. Велика кількість унікальних послуг в месенджері реалізується саме за допомогою них. Не дарма кажуть, що все геніальне – просто. Незважаючи на простоту з точки зору розробки і те, що дані боти розробляються з 2015-го року, вони надають якісний та зручний сервіс тим, хто ними користується. Тож тема лишається актуальною і донині.

*Метою даної роботи* є автоматизація процесу розрахунку калорій в їжі шляхом впровадження застосунку для перевірки кількості калорій у їжі.

*Об’єктом дослідження* є процес перевірки кількості калорій у їжі.

*Предмет дослідження* – Telegram-бот для перевірки кількості калорій у їжі.

*Методи дослідження.* У даній роботі теоретично і практично розглянуто алгоритм написання коду для функціонування Telegram-боту. Основну увагу приділено простоті та зручності використання системи задля поширення та популяризації її серед якомога більшої кількості людей. Докладно розглянуто програмне забезпечення, на базі якого написано Telegram-бот – середовище PyCharm.

*Суть розробки* – створення Telegram-боту з простим і водночас унікальним функціоналом, рекомендованим до використання широкому колу людей.

# 1 ДОСЛІДЖЕННЯ TELEGRAM-БОТІВ

## 1.1 Telegram-бот. Визначення

Боти - це спеціальні акаунти, які використовуються для автоматичної обробки та надсилання повідомлень. Користувачі взаємодіють з ботами, надсилаючи повідомлення як у звичайних, так і в групових чатах. Їхньою логікою керують HTTPS-запити до нашого бот API.

Телеграм-бот - це багатофункціональний автоматизований помічник, який може показувати підписникам певну інформацію та збирати її за запитом, за сценаріями, які заздалегідь запрограмовані розробником. Слово "бот" - це не що інше, як скорочення від всім відомого слова "робот". Таким чином, Telegram-бот - це, по суті, невеликий повноцінний додаток з усіма необхідними функціями та можливостями для виконання запрограмованих завдань, які не потребують безпосередньої участі користувача. Ці автоматично створені спеціальні акаунти дозволяють користувачам виконувати різні завдання за допомогою програми для обміну повідомленнями. Добре розроблений і продуманий бот може заощадити час фахівця, звільнивши його від нудних рутинних завдань і вивільнивши час для більш важливих справ.

З появою Telegram-ботів месенджер Telegram перестав бути просто файлообмінником для обміну фото- та відеоматеріалами, аудіозаписами, текстовими документами тощо. Компетентність Telegram-бота як самостійного чат-бота вражає: він може робити все, що без особливих зусиль може робити людина - відповідати на запитання, надсилати посилання на сайти/інтернет-ресурси і навіть створювати меми. Крім того, чат-бот може дізнаватися про останні новини, перевіряти курси валют, грати в ігри, перекладати слова або цілі тексти іншими мовами та багато іншого. На скріншоті нижче представлено бота, за допомогою якого людина може забронювати, замовити та оплатити квитки на залізницю.

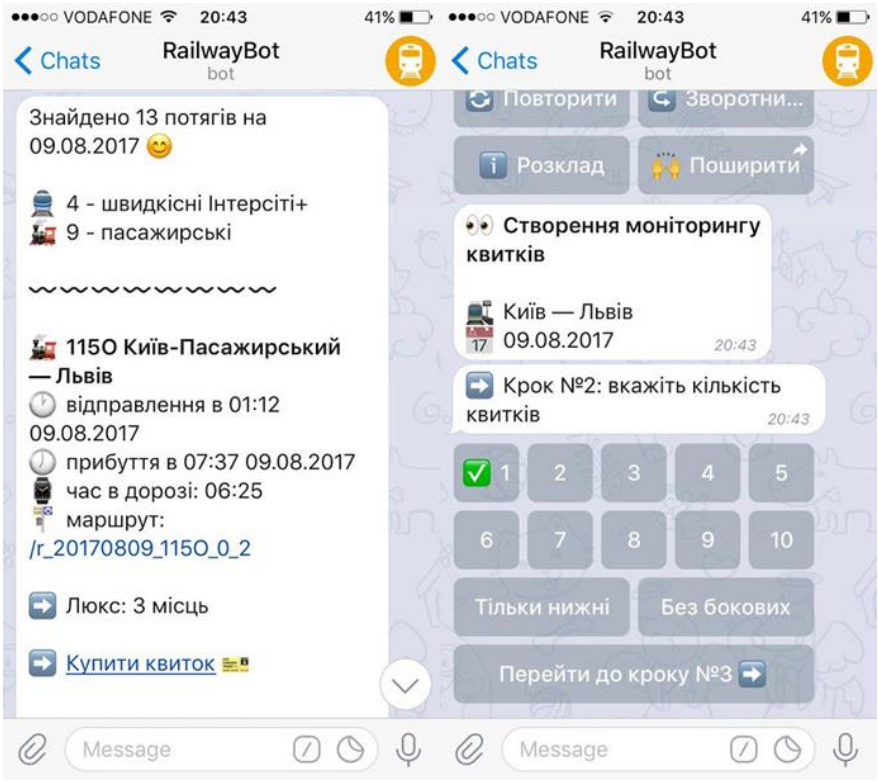



Рисунок 1.1 – Приклад Telegram-боту

## 1.2 Функціонал та особливості Telegram-ботів

Взаємодія між машиною та людиною починається одразу після натискання кнопки "Пуск" і будується на запитаннях та відповідях. Слід зазначити, що роль самого робота полягає не в тому, щоб ініціювати діалог, а лише в тому, щоб "віддзеркалювати" питання і запити користувача. Його завжди можна ідентифікувати за наявністю або відсутністю слова "бот" у назві акаунта. Більшість ботів працюють безперервно і контролюються особою, яка їх контролює, через програмне середовище, в якому був написаний код. Телеграм-боти пропонують користувачам необмежену кількість послуг. Існують боти для всіх типів програм для обміну повідомленнями; щоб знайти те, що вам потрібно, просто введіть потрібне запитання. Хочете дізнатися прогноз погоди? Хочете дізнатися поточний курс валют? Хочете дізнатися останні новини з вашого міста, країни чи світу? Не маєте з ким поговорити і хочете знайти співрозмовника? Шукаєте нову якісну музику? Ці та багато інших завдань можна легко вирішити за допомогою Telegram-роботів.

Боти в Telegram мають цілу низку можливостей:

1. Розваги. В Telegram-боті можна отримувати меми, фото та відео, жарти, GIF-ки. Також вони спрощують один з найскладніших моментів в житті людини – вибір фільму і музики.

2. Пошук та обмін файлами. Роботи дають дозвіл на здійснення збереження та відправлення файлів з різних джерел, знаходити електронні книги або потрібні торенти тощо.

3. Новини та інша важлива інформація. Бот в Telegram без проблем скине вам всі актуальні новини, відомості про погоду у вашому місті, курси валют тощо.

4. Утиліти та інструменти. Боти можуть виконувати переклад текстів, нагадувати про події та інше.

5. Інтеграція з іншими сервісами. Бот може керувати Smart-будинком, відправляти різного змісту повідомлення тощо.

6. Пошук місць. Не можете знайти театр, парк, кафе або мотель? Telegram-бот допоможе вирішити цю задачу.

7. Транзакції. Навіть на це здатен бот. З його допомогою можна і таксі викликати, і квитки замовити, і готель забронювати – будь-які фінансові операції.

І це лише мала часточка всіх опцій Telegram-ботів.

### 1.3 Види та принципи роботи Telegram-ботів

Принцип роботи ботів у соціальній мережі Telegram напрочуд простий. Зазвичай користувач створює запит або команду, яка надсилається програмі на серверах розробників. Анонімний сервер Telegram виступає посередником, обробляючи код і забезпечуючи зворотний зв'язок.

Алгоритм роботи з ботом можна сформулювати так:

- користувач дає команду;
- бот передає її на сервер;
- програма на сервері опрацьовує запит;
- сервер дає роботу відповідь;

- бот відображає її на екрані.

Цей цикл виконується постійно, коли віддає команду боту. Користувач при цьому взаємодіє із серверами за допомогою звичайного HTTPS-інтерфейсу, який за своєю природою є спрощеною версією API Telegram. Інші назви інтерфейсу – програмний каталог та бот-алгоритм. Сучасні роботи створюються за допомогою «головного» бота в Telegram - @BotFather, який полегшує процес розробки.

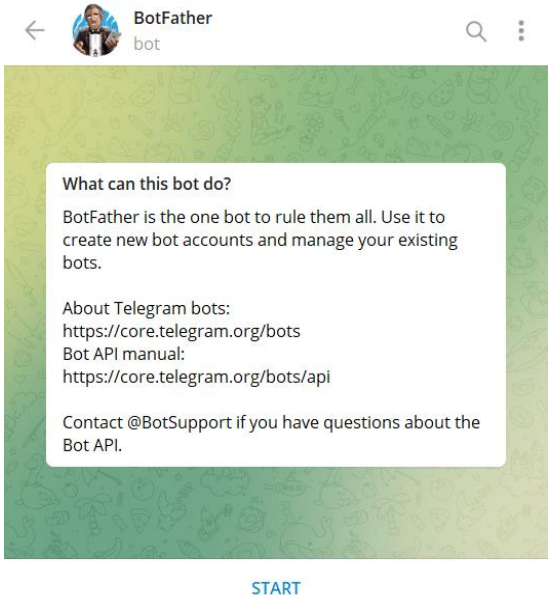



Рисунок 1.2 – BotFather, бот для створення нових ботів

У Telegram величезна кількість найрізноманітніших роботів. Інтернет-магазини, наприклад, використовують розумних помічників, щоб їхні клієнти мали змогу через Telegram оплачувати товари та спілкуватися з продавцями дистанційно. Контент-менеджери відстрочені публікації налаштовують. А кіномани за допомогою роботів сподіваються знайти цікавий фільм для перегляду з сім’єю або наодинці ввечері після роботи.

Існує декілька різновидів Telegram-ботів, а саме:

1. Чат-боти. «Класичний» вид ботів у мережі Telegram, вважається найпопулярнішим. Може взяти прості запити клієнтів на себе. Це допомагає в піковий годинник розвантажити операторів підтримки, щоб ті могли працювати з нестандартними та екстреними зверненнями. У цьому випадку алгоритми відповідають на запитання клієнтів, які найчастіше задаються. Наприклад, як

налаштувати чи підключити послугу, як оформити замовлення та доставку товарів. Цей тип чат-ботів здатний вирішувати масу різних проблем і завдань, цілодобово залишаючись на зв’язку з клієнтами.


Рисунок 1.3 – Приклад чат-боту від компанії «Київстар»

2. Ігрові боти. Створені спеціально для ігор та розваг.


Рисунок 1.4 – Приклад ігрового боту

3. Контентні боти та боти-нагадувалки. Функціонал дуже простий. Використовується, як стає зрозуміло з назви, контент-мейкерами, для публікації відкладених постів. В них завчасно завантажуються всі необхідні матеріали і ставиться таймер. Такі боти дуже рятують у вихідні дні та свята, коли робота це останнє, про що хочеться думати. Також це класний бот для тих людей, хто постійно щось забуває зробити – наприклад, вчасно випити пігулки.


Рисунок 1.5 – Приклад контентного боту

4. Боти-розшифрувальники. Як виявляється, необов’язково звертатися до спецслужб, щоб розшифрувати дані. Цей бот відображає інформацію про метадані, які надіслав користувач. Наприклад, якщо йому відправити відео, незабаром у відповідь прийде службова інформація, тайм-коди, текст. Бот також здатний перекладати голосове повідомлення текст. Така можливість є в Telegram Premium, хоча багато хто використовує безкоштовні боти. Наприклад, SaluteSpeech Bot переводить голосові повідомлення у текстові. Автори статей звертаються до нього, щоби розшифрувати інтерв’ю. Крім голосових, у бот також можна завантажувати відео та аудіо з диктофона або пересилати файли з інших чатів/каналів Telegram.


Рисунок 1.6 – Приклад боту-розшифрувальника

5. Боти-каталоги. Дають можливість в зручному форматі підібрати серіал, музику чи книги, при цьому попутно збагачуючи людину духовну та культурно.

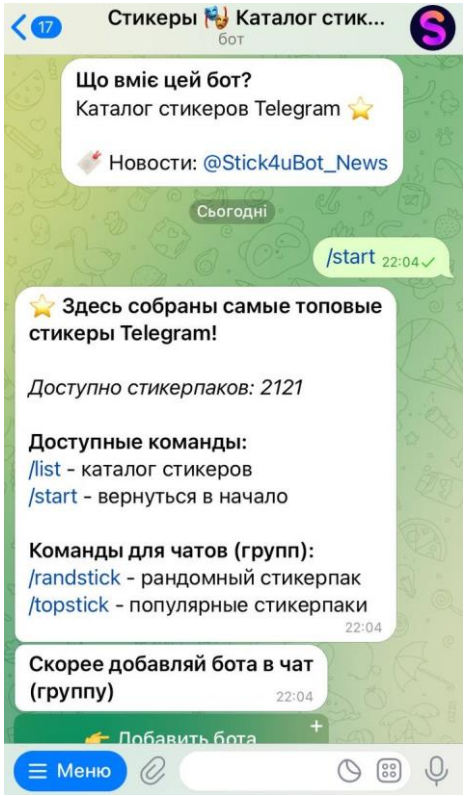



Рисунок 1.7 – Приклад боту-каталогу

6. Боти-інструменти. Допомагають користувачу здійснити певну операцію – наприклад, щось знайти або завантажити. Нижче наведено приклад такого боту.

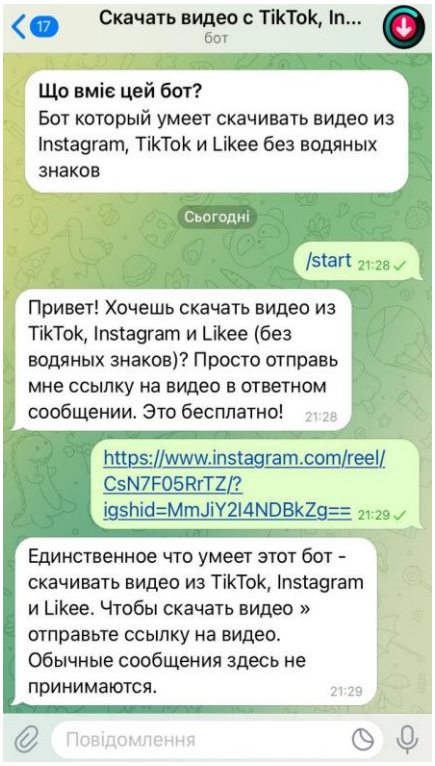



Рисунок 1.8 – Приклад боту-інструменту

7. Боти для оплати. Як правило, створюються власниками Інтернет-магазинів для надання можливості клієнтам безпечно оплачувати придбаний товар/послугу.


Рисунок 1.9 – Приклад бота для оплати

8. Боти-маркетологи. Вони містять інформацію, яка може бути корисною для блогерів. Один з таких роботів використовує такі фільтри, як тематика, ціна та статистика профілю, щоб знайти відповідні акаунти для реклами. Наприклад, на офіційному сайті робота можна вибрати блогерів, які спеціалізуються на подорожах в Україну, і оцінити їх. Профіль блогера містить типи реклами, статистику аудиторії, охоплення статей і постів, а також коментарі від інших рекламодавців. На зображенні нижче наведено приклад такого робота для пошуку користувачів каналу.


Рисунок 1.10 – Приклад бота-маркетолога

Деякі з цих категорій можна було б запросто об'єднати в одну – наприклад, боти-асистенти (інструменти, або боти-інформатори. Напевно, зараз абсолютно будь-яка компанія – маленька чи велика – як і будь-який підприємець, хотіли б мати хоча б одного свого власного телеграм-бота, який регулярно виконуватиме замість них однотипні дії, причому в автоматичному режимі.

### 1.3.1 Принципи роботи Telegram-ботів у діаграмах

Говорячи в цілому про загальний принцип роботи Telegram-боту, його можна представити у вигляді UML-діаграми.

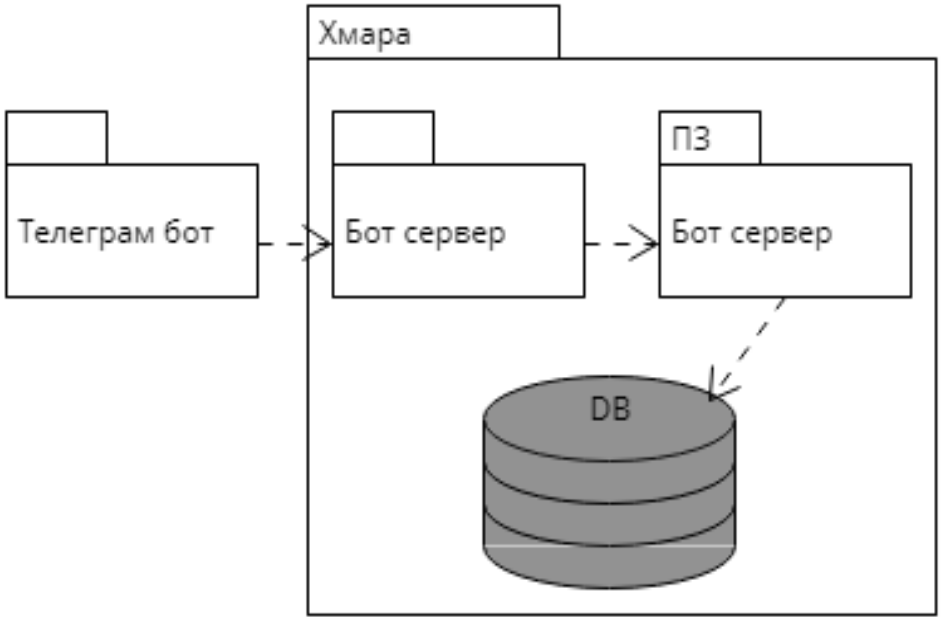



Рисунок 1.11 – Архітектура Telegram-бота

На даній діаграмі зображена архітектура Telegram-бота, який розробляється. На ньому показано взаємодію основних компонентів стандартного телеграм-бота:

1. Користувачі: Люди, які взаємодіють з телеграм-ботом, надсилаючи команди та отримуючи відповіді через платформу Телеграм.
2. Бот-сервер (хмара): Це серверна інфраструктура, яка забезпечує хостинг та розгортання телеграм-бота. Він отримує та маршрутизує повідомлення від користувачів та передає їх до бізнес-логіки.
3. Бізнес-логіка: Це серце телеграм-бота, де відбувається обробка команд, взаємодія з базою даних та надсилання відповідей користувачам. Тут виконується логіка, пов'язана з функціональністю бота.
4. База даних: Це місце, де зберігаються дані, пов'язані з телеграм-ботом, такі як користувацькі профілі, історія повідомлень, налаштування тощо. База даних забезпечує збереження та доступ до цих даних.

Ця діаграма демонструє загальну архітектуру телеграм-бота, де бот-сервер (хмара) забезпечує зв'язок між користувачами, бізнес-логікою та базою даних.

Також побудуємо для нашого бота діаграму предметної галузі.

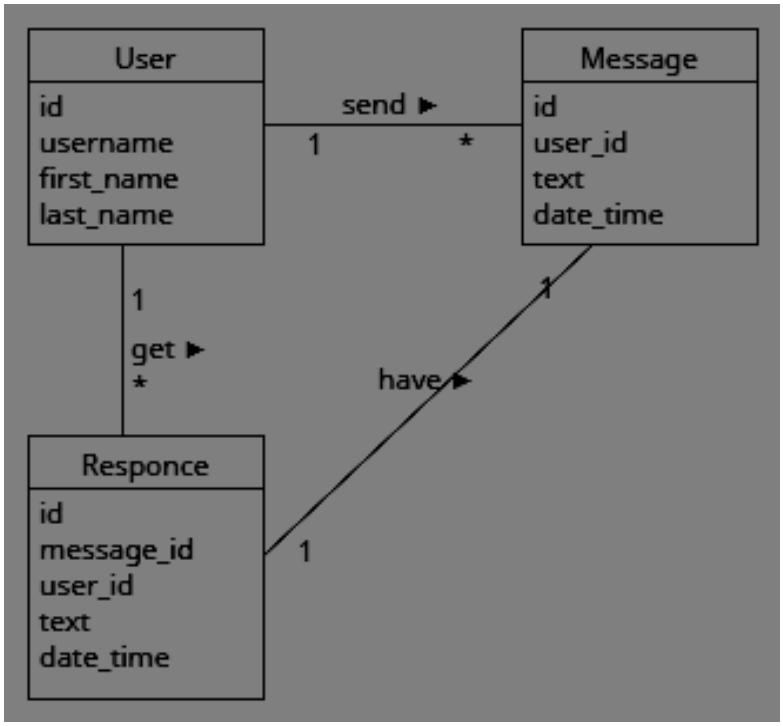



Рисунок 1.12 – Діаграма предметної галузі Telegram-боту

У цій діаграмі є три основні групи компонентів:

1. Користувач: Представляє користувачів телеграм-бота. Включає поля, такі як ідентифікатор користувача (id), ім'я користувача (username), ім'я та прізвище (first_name, last_name).

2. Повідомлення: Представляє повідомлення, які користувачі надсилають телеграм-боту. Має поля, такі як ідентифікатор повідомлення (id), ідентифікатор користувача, який надіслав повідомлення (user_id), текст повідомлення (text) та дата/час надходження (date/time).

3. Відповідь: Представляє відповіді, які телеграм-бот надсилає користувачам. Включає поля, такі як ідентифікатор відповіді (id), ідентифікатор повідомлення, на яке надсилається відповідь (message_id), ідентифікатор користувача, якому призначена відповідь (user_id), текст відповіді (text) та дата/час надсилання (date/time).

Ця діаграма предметної галузі відображає структуру даних, які використовуються в телеграм-боті. Користувачі надсилають повідомлення, які потім можуть бути оброблені ботом, і відповіді надсилаються користувачам у відповідь на їхні повідомлення.

І додатково побудуємо для нашого Telegram-боту діаграму пакетів.

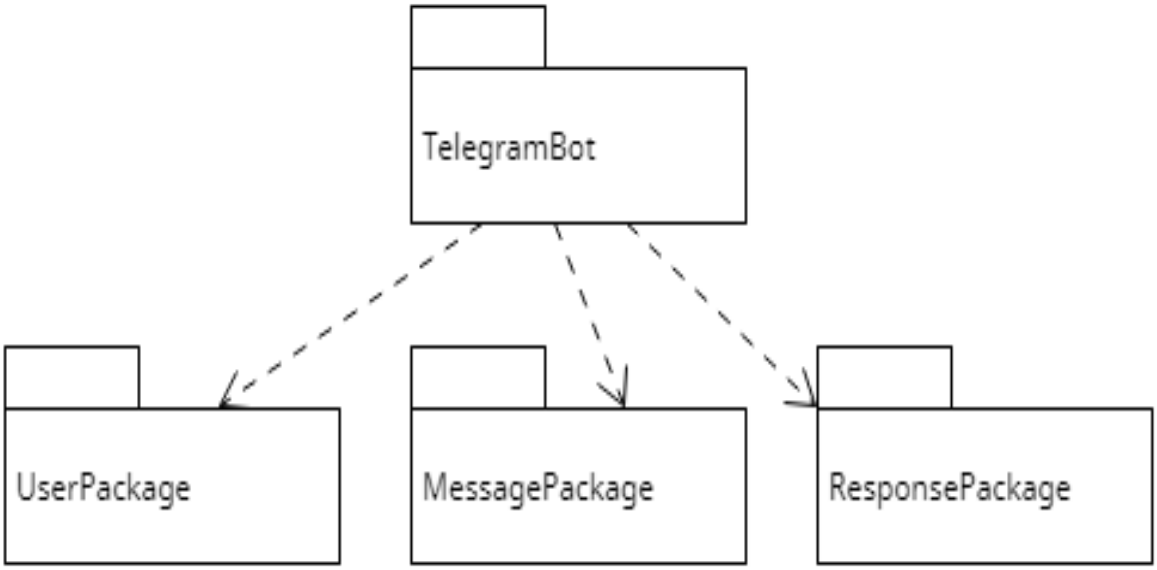



Рисунок 1.13 – Діаграма пакетів Telegram-боту

TelegramBot – це пакет, який містить всі компоненти, пов'язані з телеграм-ботом. Він містить три основних пакети:

1. UserPackage: Пакет, який містить класи і компоненти, пов'язані з користувачами телеграм-бота, такі як класи для представлення користувачів, обробки їхніх даних тощо.

2. MessagePackage: Пакет, який містить класи і компоненти, пов'язані з повідомленнями, надісланими користувачами. Це може включати класи для представлення повідомлень, обробки тексту, зберігання дати/часу тощо.

3. ResponsePackage: Пакет, який містить класи і компоненти, пов'язані ss відповідями, що були надіслані телеграм-ботом користувачам. Це може включати класи для представлення відповідей, форматування тексту, встановлення дати/часу тощо.

Ця діаграма пакетів показує структуру компонентів телеграм-бота, розташованих відповідними пакетами. Вона відображає групування класів і компонентів за їхньою функціональністю та допомагає зрозуміти, які частини бота пов'язані між собою.

Нижче побудуємо також діаграму артефактів.

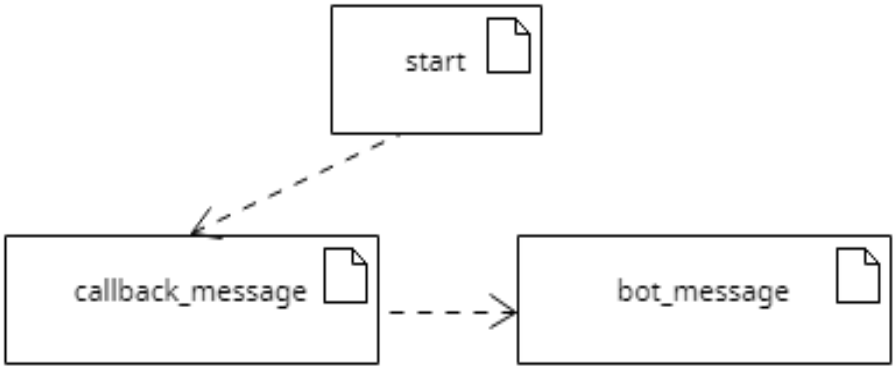



Рисунок 1.14 – Діаграма артефактів

Опис даної діаграми:

1. "start": Цей артефакт представляє початкову точку взаємодії користувача з ботом. Він може бути пов'язаний з кнопкою або командою, яку користувач натискає або вводить, щоб активувати бота. Артефакт "start" ініціює запуск бота та відкриває можливість користувачам розпочати процес перевірку кількості калорій в їжі.

2. "callback_message": Цей артефакт представляє повідомлення, яке користувач надсилає боту через взаємодію з інтерфейсом, наприклад, натисканням кнопок або введенням текстових команд. Він може містити інформацію про тип їжі, кількість порцій, назви продуктів або інші дані, необхідні для перевірки калорій. Цей артефакт є вхідною точкою для отримання даних від користувача.

3. "bot_message": Цей артефакт представляє повідомлення, яке бот надсилає користувачеві з результатами перевірки кількості калорій. Він може містити інформацію про загальну кількість калорій, рекомендації щодо дієти або будь-яку іншу інформацію, яка стосується харчування та здоров'я. Цей артефакт є вихідною точкою для надсилання результатів обробки даних користувачеві.

Ця діаграма артефактів відображає послідовність взаємодій між користувачем та ботом у процесі розрахунку кількості калорій в їжі. Вона показує, як користувач починає взаємодію з ботом, надсилає повідомлення з даними, а потім отримує відповідь з результатами розрахунку.

І також побудуємо для цього боту діаграму прецедентів, аби краще розібратись в деяких закономірностях в його роботі.

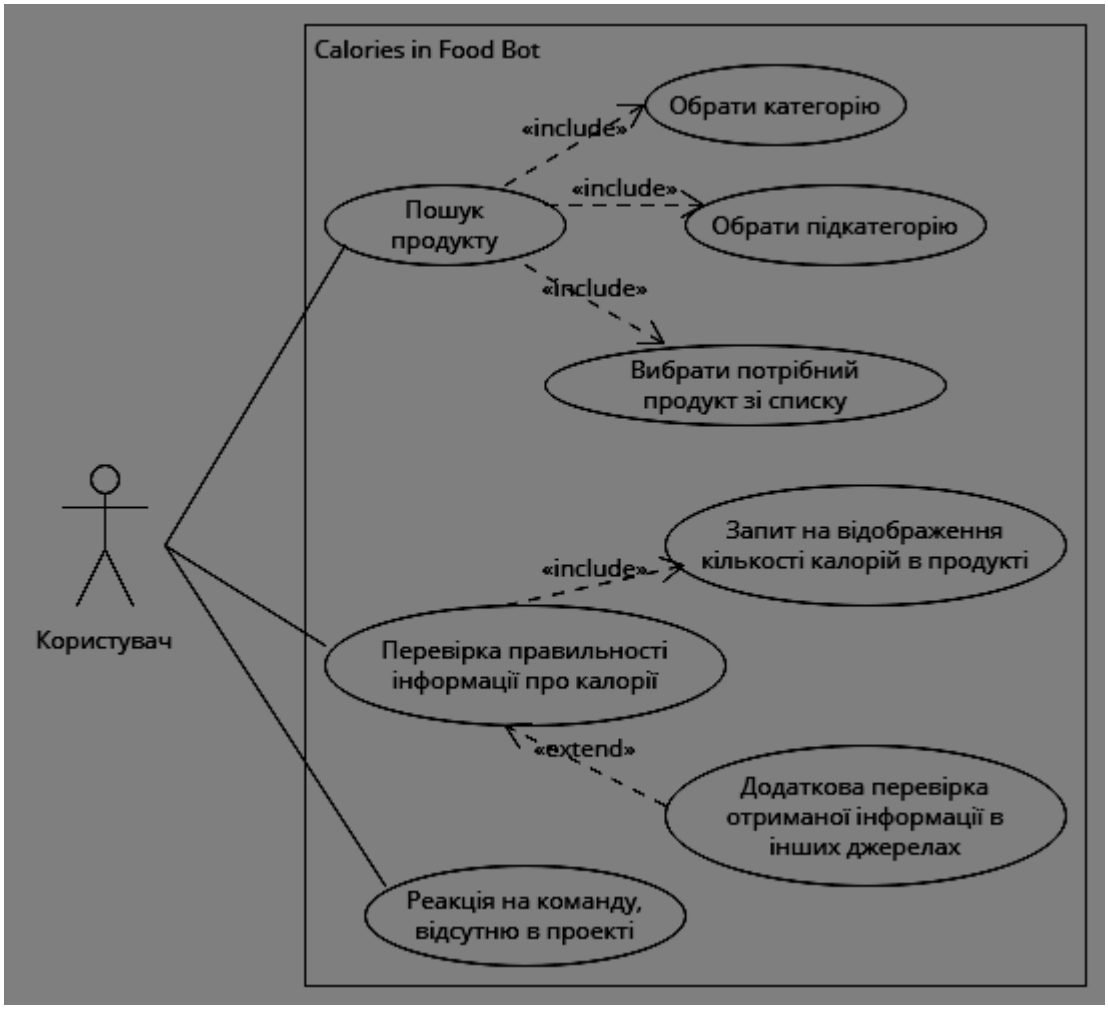



Рисунок 1.15 – Діаграма прецедентів

## 1.4 Сфери використання Telegram-ботів

Як було сказано вище, одними з найпопулярніших та найприбутковіших тематик контенту в Telegram є ті, що пов’язані із бізнесом. І дійсно, зараз майже кожен бізнес, що поважає себе, в тому чи іншому вигляді користується мережею Telegram. Тому першою і головною сферою застосування Telegram-ботів назвемо бізнес. Розглянемо деякі можливості ботів, котрі сприяють грамотному веденню бізнесу та збільшенню прибутку компанії або підприємця.

1. Безшовна екосистема. Telegram являє собою середовище із розширеним набором функцій, де гармонійно поєднуються і особисте, і робоче життя.

2. Економія часу та грошей. Вартість та час розробки Telegram-боту складає набагато менше, ніж сайту чи окремого повноцінного додатку.

3. Цілодобовий зв’язок з клієнтом. Як було вказано раніше, бот працює 24/7. При зверненні клієнта бот починає з ним спілкування, не будуючи нудний заскриптований діалог, а сповіщаючи про актуальні новини, акції та привітання, що може збільшити прибутковість компанії на 20-30% на рік.

4. Розширення рекламних можливостей. Інколи форма заявки на сайті розташована не в самому очевидному місці або збирає завелику кількість даних. Використання чат-боту в Telegram дозволить вирішити обидва фактори та зробити сегментації юзерів, що скористались ботом після перегляду реклами.

5. Сегментація та персоналізація. Той же чат-бот дасть можливість після короткого збору інформації розподілити клієнтів по категоріях, що дозволить в подальшому вийти з ними на більш високий рівень довіри.

6. Ефективна реферальна система. Багатьом людям набридла стандартна реферальна система, коли хтось, кого ти запрошуєш, отримує від тебе посилання, після якої на тебе чекає довга по часу та кількості кроків реєстрація. Telegram вирішує цю проблему, дозволяючи зробити всі необхідні дії за декілька кліків.

7. Велика кількість функцій. CRM, платіжна система, програма лояльності, вільні місця в залі – це і близько не повний список функцій, якими може бути наділений продуманий та якісний Telegram-бот.

Серед галузей бізнесу, де саме знадобляться Telegram-боти – Інтернет-магазини, офлайн магазини, служби доставки, організація масових заходів, служби таксі, beauty-сфера, фінансові організації (банки), готельно-ресторанний бізнес.

### 1.5 Питання безпеки Telegram-ботів

Усі дані месенджера знаходяться на віддалених серверах. Якщо припустити, що зловмисник випадково проник до дата-центру, він не матиме доступу до листування конкретної людини чи її персональних даних. Якщо пощастить, він дістанеться сервера, де зберігаються повідомлення, але тут воно закінчується, бо повідомлення шифруються в кілька шарів. Кожен рівень має свій ключ шифрування, що зберігається в іншому місці і захищений іншим ключем.

Бот – це програма, алгоритм якої призначений для автоматичного виконання закладених у нього дій. При створенні бота @Botfather ми отримуємо ідентифікатор – токен. Це послідовність символів, що має більшу довжину.

Наприклад: 532997345:ACGP_fmIO3mygOklaNp-WelQBJ4tGjaPvC8 .

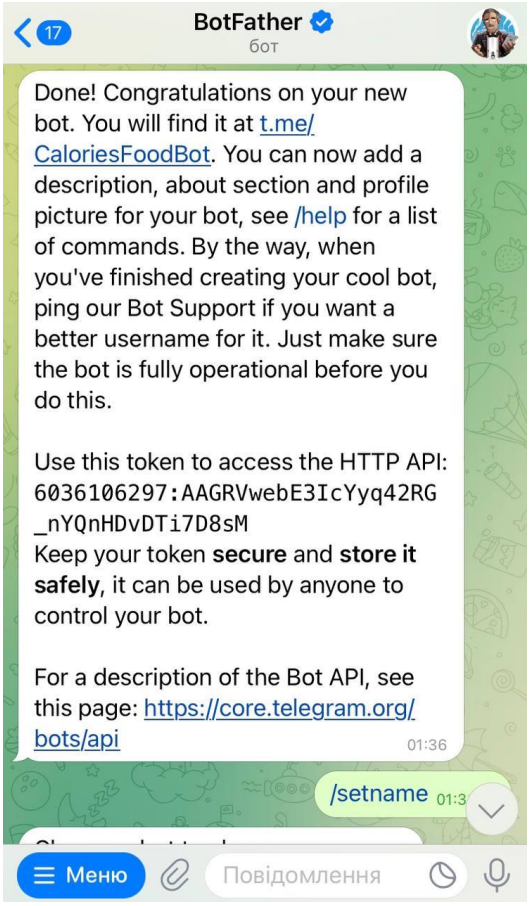



Рисунок 1.16 – Видача ботом BotFather індивідуального токену бота

Для злому робота потрібно знати цю послідовність, тому що всі запити до ресурсу містять його токен.

Наприклад: https://api.telegram.org/ bot532997345:ACGP_fmIO3mygOklaNp-WelQBJ4tGjaPvC8/sendMessage?chat_id=391911270&text=Hello .

Нерозумно намагатися вирішити цю проблему за допомогою координації. Такий підхід часто називають "грубою силою". Під цим терміном маються на увазі всі види злому, включно з програмами, додатками та Telegram-ботами. Це може бути будь-який інший метод. Це може зайняти кілька років. Тому злом токенів вважається неможливим; єдиним і найпоширенішим методом злому Telegram-ботів є викрадення акаунта власника за допомогою викраденого мобільного телефону.

Тому основний захист акаунтів і ботів Telegram безпосередньо пов'язаний з безпекою мобільного телефону як фізичного носія інформації.

Отже, стосовно безпеки можна зробити висновок: месенджер Telegram – максимально надійний. На стільки, що Павло Дуров декілька років тому пообіцяв особисто виплатити 300.000 доларів тому, кому вдасться зламати акаунт або Telegram-бот. До сих пір цю суму так ніхто і не отримав. І навряд чи найближчим часом хтось отримає.

## 1.6 Методи розробки Telegram-ботів

Створити власного бота для Telegram - непросте завдання. Так, вам не обов'язково бути програмістом, але тільки якщо ви можете створити простого помічника, який виконує прості команди і відповідає на очевидні запитання. Цього достатньо, щоб розважати користувачів і просувати свій канал.

Існує 2 способи створення Telegram-ботів: за допомогою середовища розробки та інструменту для ботів.

Спочатку розглянемо найпростіший спосіб створення ботів за допомогою інструменту для створення ботів. Він досить простий: список завдань, які повинен вирішувати бот, поступово складається блоками. Щоб створити бота, маркетологам потрібно встановити розклад, за яким бот буде надсилати повідомлення іншим користувачам.

Для тих, хто віддає перевагу персоналізації, є більш складний і цікавий варіант: оплатити послуги досвідченого програміста, який напише конкретного бота і додаток для виконання конкретних завдань. Як правило, це не цікавить маркетологів. Компілятори - це візуальні реалізації готового коду, написаного фахівцем. Їх легко знайти в будь-якій пошуковій системі.

Щоб запустити бота, спочатку потрібно зв’язатися з «батьком ботів» – знайти у Telegram бота Botfather. Назва повністю відповідає суті: через нього створюються нові боти, а він призначає їм власний номер - токен. Він потрібен, аби зв’язати бота,

який поки що нічого не вміє, із сайтом-конструктором. Потім потрібно прописати команди, які новоспечений бот повинен буде виконувати.

Знайшовши потрібного робота в пошуку, відкривайте його, щоб виконати такі дії:

1. Введіть /newbot, щоб створити бот.

2. Придумайте ім’я віртуального помічника. Підійде будь-яка комбінація із символів на латиниці, цифр, дефісів та підкреслень. Обов’язкова умова: назва має бути на «bot».

3. Встановіть аватарку для робота (не обов’язково).

4. Скопіюйте та збережіть особистий код, який ви отримаєте від @BotFather.

Він дозволить вам змінювати програму робота. Для цього можна використовувати сторонні програми. Наприклад, бот @Paquebot дає можливість інтегрувати у ваш віртуальний помічник кілька корисних функцій без знання тонкощів програмування. І це не єдиний бот у своєму роді, здатний розширити спектр можливостей створюваного вами робота.

Що стосується другого варіанту – із програмним середовищем – він вимагає певних навичок програміста. Цей спосіб і був застосований для виконання даної дипломної роботи, адже спектр завдань, які були поставлені перед виконанням дипломної роботи, вимагали їхнього виконання та вирішення саме на рівні програмного середовища. Детальніше про нього піде мова у наступному розділі.

## 1.7 Аналіз наявних Telegram-ботів для підрахунку енергетичної цінності продуктів

Серед Telegram-ботів, які вже існують на просторах даної платформи, в ході роботи над дипломним проектом було знайдено один схожий за задумкою, сферою використання та головною метою на той, що є предметом дослідження даної роботи. Це «Калькулятор калорій Ольги Базікало» - чат-бот в Telegram, який підраховує денну норму калорій, враховуючи введені фізіологічні дані користувача (стать, зріст, вага, вік, фізичні навантаження). Бот має мінімалістичний, простий у використанні інтерфейс.

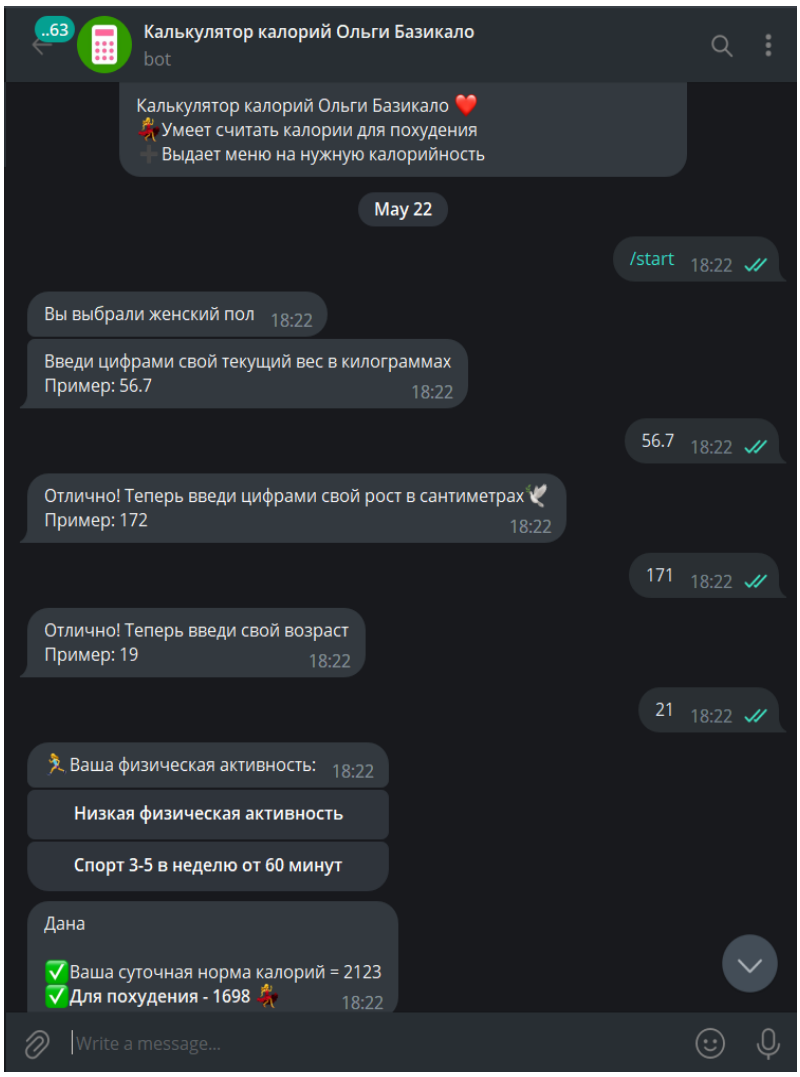



Рисунок 1.17 – Приклад роботи Telegram-боту “Калькулятор калорій Ольги Базікало”

Переваги “Калькулятор калорій Ольги Базікало”:

- простий у використанні інтерфейс;
- введення персональних даних користувача;
- підрахунок рекомендованих калорій споживання на день.

Недоліки даного чат-боту:

- немає підрахунку кількості калорій на 100г продукту;
- немає можливості повторного підрахунку калорій;
- немає інтерфейсу українською мовою.

Для наочності, порівняємо основні показники та характеристики нашого бота та у формі таблиці.

Таблиця 1.1 Порівняльний аналіз Telegram-ботів «Калькулятор калорій Ольги Базікало» та «Calories in Food Bot»

| Властивості | «Калькулятор калорій Ольги Базікало» | «Calories in Food Bot» |
|---|---|---|
| Середовище використання | Telegram | Telegram |
| Безкоштовний контент | - | + |
| Простий у використанні інтерфейс | + | + |
| Введення персональних даних користувача | + | - |
| Підрахунок рекомендованих калорій споживання на день | + | - |
| Підрахунок кількості калорій на 100г продукту | - | + |
| Можливість повторного підрахунку калорій | - | + |
| Інтерфейс українською мовою | - | + |

# 2 РОЗРОБКА TELEGRAM-БОТУ ДЛЯ ПЕРЕВІРКИ КІЛЬКОСТІ КАЛОРІЙ В ЇЖІ МОВОЮ PYTHON

## 2.1 Створення тіла боту

Розробка будь-якого чат-боту в Telegram, незалежно від способу – написання коду в програмному середовищі чи використання конструктору – розпочинається зі звернення до «спеціаліста» зі створення нових ботів – боту під назвою @BotFather.

Крок 1. Звернення до бота.

В пошуковому рядку в Telegram вводимо «BotFather» і обираємо перший варіант, який пропонує система. Цей бот був створений самим першим і його призначення – видача кожному новоутвореному боту так званого токена. Токен – це індивідуальний набір символів (букви, цифри, спеціальні знаки), який ідентифікує бота у внутрішній системі Telegram. За допомогою команди «/start» запускаємо бота.

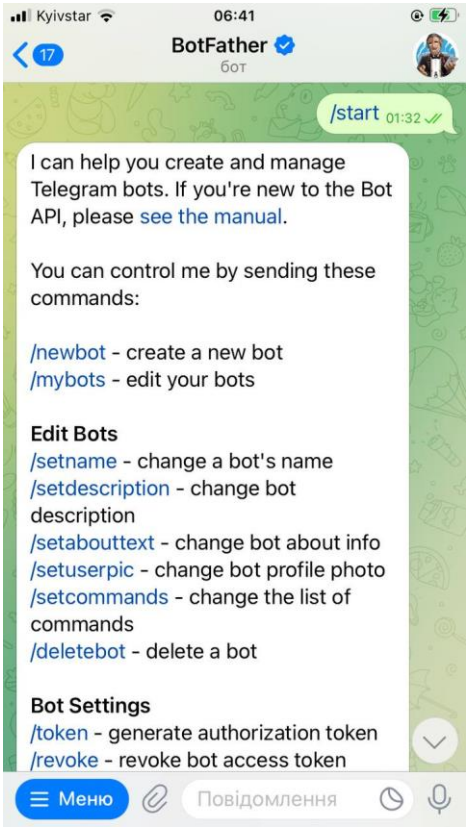



Рисунок 2.1 – Початок роботи з BotFather, ч.1

Як бачимо, після введення команди «/start» бот одразу вітає нас та детально інформує про те, які функції він вміє виконувати та який сервіс надає, а також знайомить із повним переліком команд, на які він реагує, для зручності поділивши їх на категорії, які називаються «Edit Bots» (надання боту назви, опису, списку команд для роботи), «Bot Settings» (налаштування бота), «Web Apps» (робота з веб-додатками у боті) та «Games» (інструменти для роботи з іграми).

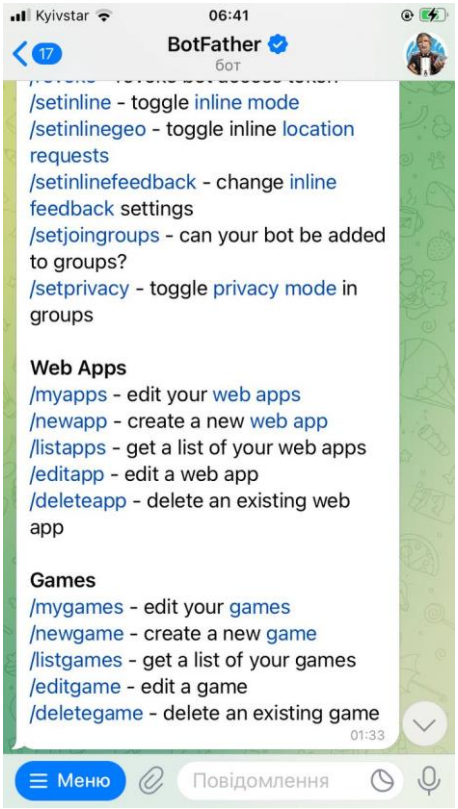



Рисунок 2.2 – Початок роботи з BotFather, ч.2

Крок 2. Створення нового боту, привласнення йому назви.

Після того, як бот з нами привітався, переходимо до створення нового боту. В цьому нам допоможе команда «/newbot». Забігаючи наперед, необхідно зазначити, що всі ті налаштування і параметри, які ми вказуємо в системі на момент створення Telegram-бота, в подальшому можуть змінюватись. Це виходить, що будь-який бот, який створювався для одних цілей, в майбутньому може бути з легкістю перепрограмований під зовсім інший тип діяльності.


Рисунок 2.3 – Створення нового боту

Далі BotFather, отримавши від нас відповідь на питання, як ми хочемо назвати бота, надає йому власний токен – індивідуальний набір символів, який є ідентифікатором боту в системі.

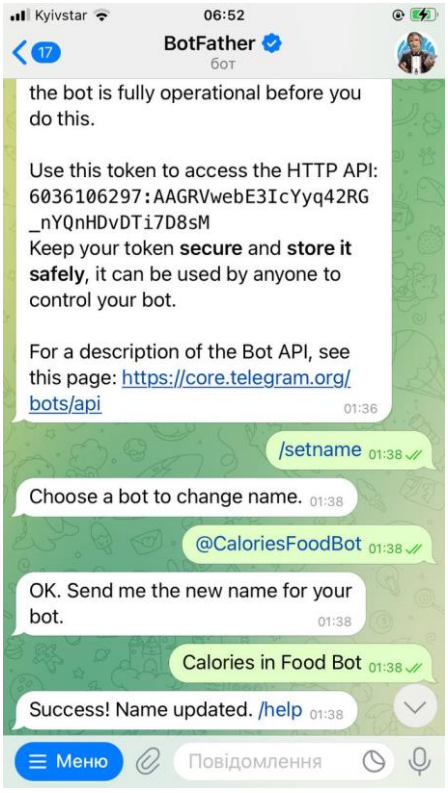



Рисунок 2.4 – Привласнення новому боту токена

Примітно те, що назву бота можна змінювати. Для цього існує команда «/setname».

Крок 3. Налаштування боту.

Для того, щоб зробити бот впізнаваним та цікавим для перегляду, він має бути привабливим візуально. Тому треба обрати для нього якусь картинку та скласти невеличкий та розширений опис, щоб новий користувач, який бачить бота, розумів, для чого він призначений. Для цих налаштувань мені знадобились 3 команди:

1. /setuserpic – встановлення користувацької картинки на аватарку боту.
2. /setdescription – короткий опис боту.
3. /setabouttext – більш розгорнутий опис боту.

Окрім цього, можна також застосувати команду «/setcommands», за допомогою якої користувач може налаштувати список команд, які виконуватиме бот та на які реагуватиме в разі надходження від людини того чи іншого спеціалізованого запиту.

Також в цьому блоці команд @BotFather дає можливість видалити бота шляхом виконання команди «/deletebot».

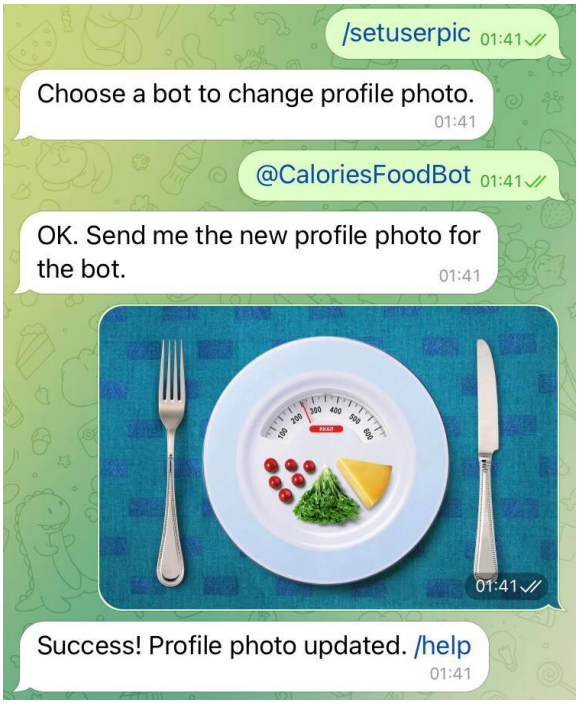



Рисунок 2.5 – Встановлення фото боту


Рисунок 2.6 – Створення опису боту

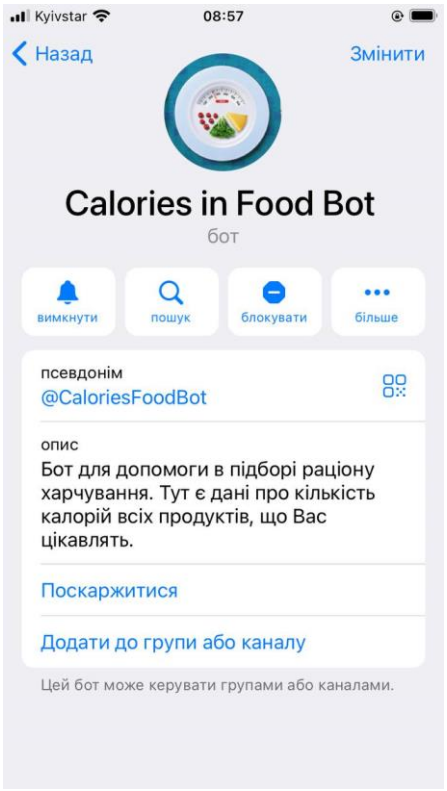



Рисунок 2.7 – Інформація про бот після налаштувань

Після цих кроків бот готовий до того, щоб починати його програмувати.

## 2.2 Програмне середовище PyCharm

Для написання боту була обрана мова програмування Python. На те є низка причин, які обґрунтовують такий вибір. По-перше, Python вважається однією з найлегших для вивчення мов для програмістів. Це обумовлено простим синтаксисом та загальною функціональністю. Пітон має свої особливості, як і будь-яка мова програмування. Головним підводним каменем, який може спіткати людину на початковому етапі освоєння – відступи. Розібравшись із їхньою логікою, подальша робота з ним є дуже легкою.

Друга причина – він містить в своїй екосистемі безліч доступних бібліотек для ботів, причому мова йде про бібліотеки на основі штучного інтелекту, що є великою перевагою для програмістів. Він має багато готових рішень для створення ботів, це десь декілька десятків бібліотек для створення ботів різної складності. Іншими популярними мовами, на яких пишуть чат-ботів, зокрема, є PHP, C#, C++, Java та JavaScript. Але пальму першості впевнено займає Python і це цілком виправдано.

Важливою обмовкою є сфера застосування бота. Якщо це чат-бот для месенджера, як в нашому випадку, то Python пасує більш, ніж ідеально. Але якщо це браузерна гра або комп’ютерна гра, тоді краще підійдуть JavaScript та C#/C++, відповідно.

За основу було взято програмне середовище PyCharm, яке надає користувачу комплекс засобів та інструментів для написання коду мовою програмування Python. Він містить вбудований якісний відладчик – комп’ютерну програму для автоматизації процесу пошуку помилок в програмі. Даний продукт є безкоштовним і знаходиться під ліцензією Apache License. В програмі зручно працювати завдяки тому, що різні групи команд підсвічуються різним кольором, а сама система постійно сигналізує в разі, якщо в коді допущено помилку. Також відладчик може виконувати трасування, відслідковувати, змінювати або встановлювати значення змінних в процесі виконання коду, встановлювати і видаляти умови зупинки роботи і так далі.

Рисунок 2.8 – Логотип програми PyCharm

Для роботи в програмі з офіційного сайту було завантажено та встановлено додаток Python 3.11.3 для Windows 10 (x64).

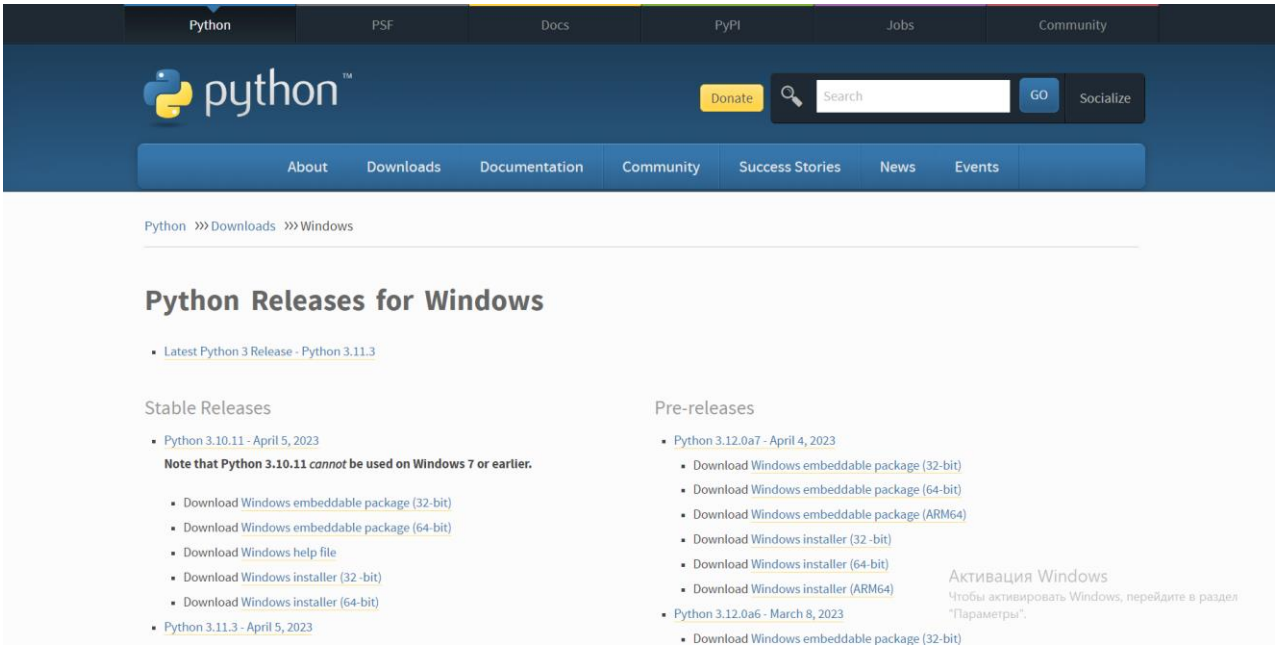



Рисунок 2.9 – Інтерфейс офіційного сайту Python

Програма PyCharm також була завантажена з офіційного сайту виробника – JetBrains.Com, версія – 2023.1.1. Була обрана безкоштовна версія, яка дозволяє займатись розробкою тільки на Python, в той час як платна версія дає можливість розроблять ще й на HTML, JavaScript та SQL. Безкоштовної версії вистачає більшості програмістів-початківців, а комерційна версія потрібна в основному компаніям та професіоналам, що працюють із великими проектами.

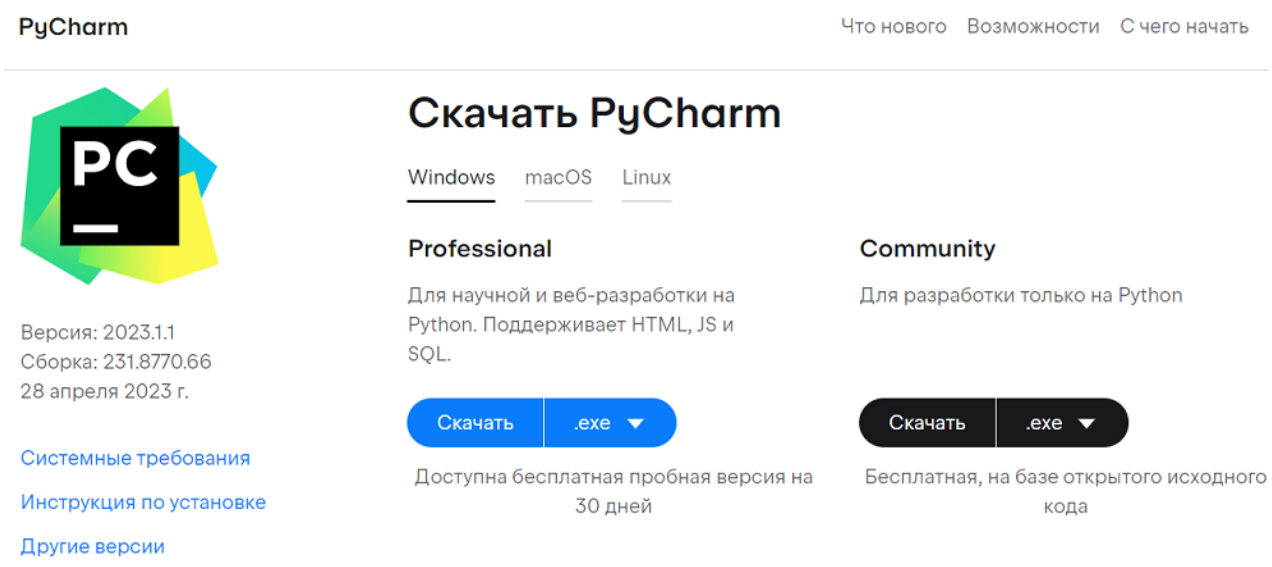



Рисунок 2.10 – Встановлення ПЗ PyCharm

Насправді, можна було розробляти цей код і в звичайному Блокноті, попередньо встановивши на ПК мову програмування Python, а потім виконувати код через консоль. Але це незручно, тому стояв вибір між редактором коду та інтегрованим середовищем розробки – IDE. Різниця полягає в тому, що:

- редактор коду – це проста програма, яка візуально нагадує текстовий редактор, адаптований під написання коду, що зберігає проекти в необхідному розширенні, підсвічує синтаксис і автоматично перевіряє відступи в коді;
- IDE – це більш різноплановий інструмент, де окрім редактор ще є власна консоль, інструменти для запуску, тестування відладки коду і через який можна створювати масштабні проекти та підключати систему контролю версій (Git).

Редактор може бути як самостійною програмою, так і частиною IDE. Він наділений такими властивостями:

1. Підсвітка синтаксису. Наприклад, службові слова матимуть один колір, коментарі – другий, класи – третій і т.д. Це допоможе вправніше орієнтуватись в коді та знаходити потрібні рядки.

2. Форматування та встановлення відступів. Хороші редактори, як і IDE, підтримують масштабне автоформатування. А відступи використовуються майже в кожні мові програмування, але в Python вони є обов’язковими та є частиною синтаксису.

3. Можливість запуску коду. Є не у всіх редакторах, реалізується шляхом натискання відповідної кнопки в самому редакторі.

4. Відладчик. Спеціальний інструмент або набір інструментів, які призначені для знаходження та виправлення помилок.

5. Кастомізація. Є можливість налаштувати майже кожен редактор під себе. Кольорова гама, шрифти, розташування вікон, коду, панелі під рукою, гарячі клавіші та багато чого іншого – все можна кастомізувати.

IDE, на відміну від редактора, має більше функцій та інструментів. Розглянемо окремі можливості, які надає IDE:

1. Повна синхронізація із системою контролю версій.

2. Інтерактивна консоль.

3. Візуальний редактор для швидкого створення проектів з блоків та окремих файлів.

4. Можливість встановлювати фреймворк або бібліотеку через інтерфейс програмного середовища.

5. Додаткові інструменти для конкретного напрямку IT.

6. Велика кількість можливостей для наочної відладки, аналізу коду, тестування, а також вирішення окремих задач.

7. Можливість працювати з декількома мовами одночасно.

Програмне середовище PyCharm було обране через низку суттєвих переваг над конкурентами, серед яких є як такі, що спеціалізуються виключно на Python (Spyder, IDLE, Thonny), так і ті, що є мультимовними IDE з підтримкою Python (Visual Studio, Eclipse). Розглянемо їх детальніше.

По-перше, середовище PyCharm по суті є повноцінним співавтором вашого коду, адже надає розумну допомогу в його написанні та редагування. Дане ПЗ знає все про ваш код. Він здатний миттєво перевіряти його на наявність помилок та швидко їх виправляти, надає можливість зручної навігації по проекту і навіть підказує, як зробити код більш правильним з точки зору синтаксису та розташування команд.

По-друге, ця IDE здатна підвищувати якість коду, контролюючи її шляхом регулярних перевірок PEP8, надання допомоги в тестуванні, інтелектуального рефакторингу та багатьох інших різнопланових перевірок.

По-третє, PyCharm надає чудову підтримку фрейморків для сучасних фрейморків веб-розробки (Django, Flask, Google App Engine, Pyramid, web2py) і містить в собі багато безкоштовних бібліотек. Також варто зазначити, що при купівлі платної комерційної версії програми PyCharm вона підтримує не тільки Python, а і ряд інших мов – JavaScript, SQL, TypeScript, HTML/CSS тощо.

Загалом, середовище чудово підтримує всі можливі доповнення та дає великий простір для дій. Немає потреби довго його налаштовувати, воно одразу адаптоване під Python. Однак є і недоліки. Наприклад, в порівнянні із середньостатистичним редактором, PyCharm завантажується і працює трохи повільніше. Та навряд чи це має значення, коли на іншій чаші терезів стільки переваг. Це середовище найчастіше радять програмістам-початківцям.

## 2.3 Написання програми в Python

При створенні нового проекту (назвемо його DanaFood) в нас на екрані з’являється стандартний приклад скрипта в Python.

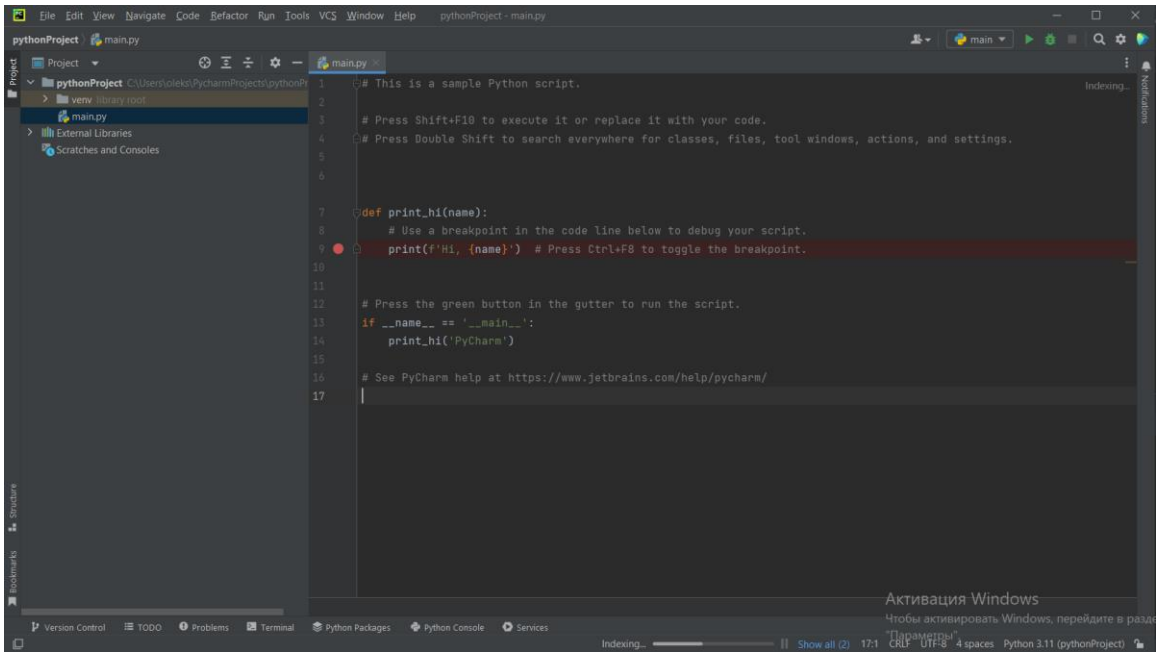



Рисунок 2.11 – Приклад скрипта Python

Для того, щоб почати працювати з ботом, одразу підключаємо до проекту бібліотеку «telebot».

```
1    import telebot
```

Рисунок 2.12 – Підключення бібліотеки telebot

З підключеної бібліотеки також імпортуємо об’єкт «types», який в подальшому дозволить нам створювати кнопки.

```
3    from telebot import types
```

Рисунок 2.13 – Підключення об’єкту types

Також за допомогою Терміналу імпортуємо ще одну бібліотеку для управління нашим ботом – PyTelegramBotAPI.

```
Terminal:  Local  +
Windows PowerShell
(C) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

Попробуйте новую кроссплатформенную оболочку PowerShell (https://aka.ms/pscore6)

(venv) PS C:\Users\oleks\PycharmProjects\DanaFood> pip install PyTelegramBotAPI
```

Рисунок 2.14 – Встановлення бібліотеки PyTelegramBotAPI

Далі ми записуємо той токен, який привласнив нашому боту BotFather.

```
5    bot = telebot.TeleBot('6036106297:AAGRVwebE3IcYyq42RG_nYQnHDvDTi7D8sM')
```

Рисунок 2.15 – Призначення токену

Наступним кроком ми прописуємо оператор, за допомогою якого ми будемо відслідковувати певні команди, а також створимо деяку функцію сервісом «def», яку ми назвемо так само, як називається команда (в нашому випадку це «start»), яка приймає певний параметр.

```
13   @bot.message_handler(commands=['start'])
14   def start(message):
```

Рисунок 2.16 – Оператор, що відслідковує команду /start

Отримавши від користувача запит на початок роботи з ботом, нам потрібно, щоб у відповідь він побачив якесь привітання від бота, а також клавішу, натиснувши на яку, юзер перейде до меню з різноманітними опціями. Про все по порядку.

1. Для виводу тексту у відповідь на /start, ми будемо використовувати команду «bot.send_message», а також змінну «mess», де буде прописаний текст, на який посилатиметься наша команда. В нашому випадку це буде привітальний текст із фразою «Привіт! Чим можу допомогти?».

2. Щоб вивести на екран ім’я та прізвище користувача, в тексті вводимо команди «message.from_user.first_name» та «message.from_user.last_name».

3. Аби додатково виділити їх шляхом підкреслення, скористаємося синтаксисом мови HTML, доступ до якої ми отримаємо, завантаживши відповідну бібліотеку (PyHTML) через ввід запиту у термінал (Alt+F12) або через налаштування самого проекту (File – Settings – Project – Python Interpreter – Install). Посиланням на html слугуватиме команда parse_mode, яка вказується в останньому рядку в дужках навпроти команди «bot.send_message».

4. Для виводу кнопок застосуємо змінну «markup», де буде записаний оператор «types», який ми імпортували раніше, та команда «InlineKeyboardMarkup», яка дозволить вивести нашу кнопку безпосередньо під текстом.

5. Щоб кнопка містила якийсь текст, трохи нижче використаємо змінну «item1», де запишемо назву нашої кнопки, а в кінці додамо команду «callback_data», яка знадобиться трохи згодом, і дамо їй також певне значення.

6. Для закріплення тексту за кнопкою, введемо в наш код команду «markup.add», яка додає елементи в наш markup.

7. В кінці команди «bot.send.message» додамо оператор «reply_markup», який не дасть забути про кнопку, яку ми описували в цьому блоці, і без проблем виведе її на екран користувача.

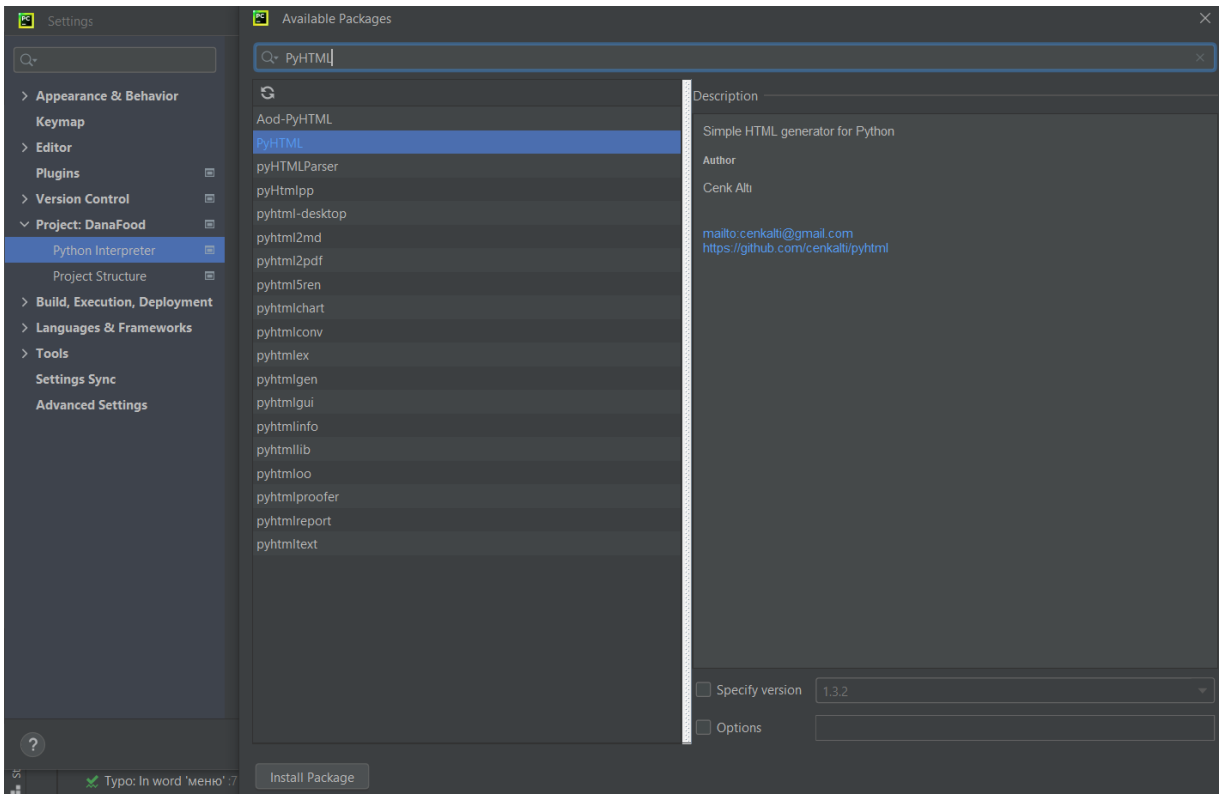



Рисунок 2.17 – Отримання доступу до PyHTML через налаштування проекту

```
@bot.message_handler(commands=['start'])
def start(message):
    mess = f'Привіт! Чим можу допомогти, <b>{message.from_user.first_name} <u>{message.from_user.last_name}</u></b>?'
    markup = types.InlineKeyboardMarkup()
    item1 = types.InlineKeyboardButton('Кількість ккал на 100г продукту', callback_data='next1')
    markup.add(item1)
    bot.send_message(message.chat.id, mess, parse_mode='html', reply_markup=markup)
```

Рисунок 2.18 – Команда, що містить інформацію для виводу і відповідь на /start

Тепер, після того, як ми прописали повністю команду в програмному середовищі, ввели значення кожної змінної і вказали в кінці команду «bot.send_message), наш бот може надавати повноцінну відповідь на користувацький запит, що ми і бачимо на скриншоті нижче. Фактично, перший крок вже зроблений і перша кнопка з’явилась у системі. В наступній частині цього розділу мова буде йти вже про створення повноцінного меню з великою кількістю вибору та альтернативних команд, кінцевим результатом натискання яких буде видача інформації по тому чи іншому продукту, скільки калорій міститься в 100 грамах конкретної їжі.


Рисунок 2.19 – Огляд першої прописаної в бота команди

## 2.4 Створення повноцінної програми в PyCharm

Після того, як наш бот почав реагувати на початок роботи з ним виводом тексту і клавіші «Кількість ккал на 100г продукту», настав час переходити до другої команди, яка міститиме в собі інформацію про всі кнопки, які ми будемо використовувати. Якщо ця кнопка в нас була задана командо «InlineKeyboardMarkup», то наступні ми будемо задавати командою «ReplyKeyboardMarkup». Різниця між цими двома командами в тому, що перша з них виводить клавішу під текстом, який надсилає бота, а друга – під полем для вводу повідомлення. Обидва варіанти можуть застосовуватись і будуть коректно відображатись на різних пристроях.

Також для цієї частини коду ми використаємо інший оператор, який називається «@bot.callback_query_handler», суть якого полягає в наданні інформації у відповідь на запит користувача. Цей оператор буде реагувати на

відгуки від користувача, тому команда, за якою він слідкуватиме, так і називається – callback. До того ж, реагуватиме він тільки на правильно подані користувачем запити, про що свідчить відповідний запис навпроти оператора (callback = True). Ця функція буде сталою, про що каже те, що значення деякої функції func = lambda. Некоректні команди бот ігноруватиме через неможливість розпізнати їх.

В меню, на яке посилатиметься кнопка, яку ми задали в попередній частині, входитиме 10 категорій продуктів, а саме: «Овочі», «Фрукти», «Молочні продукти», «Ковбасні вироби», «Хлібобулочні вироби», «Кондитерські вироби», «Безалкогольні напої», «Алкогольні напої», «Зернові культури» та «М'ясні вироби». Набір даних категорій був визначений як оптимальний для побудови цього Telegram-боту. В кожній з цих категорій будуть свої підгрупи. Для більш детального огляду коду за основу буде взята категорія «Овочі».

Спосіб задання змінних та клавіш буде той самий, що і в попередніх фрагментах коду. З нових функцій з'являється «markup.row». В ній ми вказуємо, які змінні (тобто, які кнопки) будуть розташовані в рядку. Даний оператор не є обов'язковим, проте він дозволяє конкретизувати розташування кнопок і уникнути можливих помилок програми. В рядку в нас буде розміщено по 2 клавіші.

```
@bot.callback_query_handler(func=lambda callback: True)
def callback_message(callback):
    if callback.data == 'next1':
        mess = f'Оберіть категорію:'
        markup = types.ReplyKeyboardMarkup(row_width=2)
        item1 = types.KeyboardButton('Овочі')
        item2 = types.KeyboardButton('Фрукти')
        markup.row(item1, item2)
        item3 = types.KeyboardButton('Молочні продукти')
        item4 = types.KeyboardButton('Ковбасні вироби')
        markup.row(item3, item4)
        item5 = types.KeyboardButton('Хлібобулочні вироби')
        item6 = types.KeyboardButton('Кондитерські вироби')
        markup.row(item5, item6)
        item7 = types.KeyboardButton('Безалкогольні напої')
        item8 = types.KeyboardButton('Алкогольні напої')
        markup.row(item7, item8)
        item9 = types.KeyboardButton('Зернові культури')
        item10 = types.KeyboardButton('M`ясні вироби')
        markup.row(item9, item10)
        markup.add()
        bot.send_message(callback.message.chat.id, mess, parse_mode='html', reply_markup=markup)
```

Рисунок 2.20 – Команда, що містить інформацію про категорії продуктів

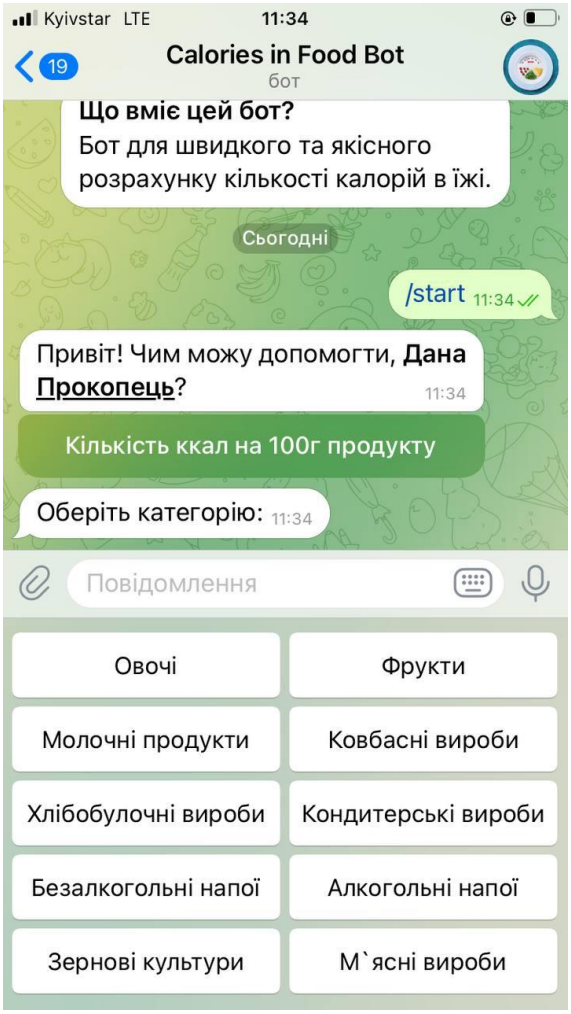



Рисунок 2.21 – Відображення головного меню всередині бота

В кожній з них буде міститись інформація ще про декілька категорій продуктів, в яких вже перебуватимуть назви конкретних товарів. Розглянемо фрагмент цієї частини коду на прикладі категорії «Овочі».

Тут ми знову повертаємось до оператора «@bot.message.handler». На цей раз він вже буде відслідковувати не команду /start, а текст, який обере користувач, натиснувши ту чи іншу клавішу, про що каже інформація в дужках біля оператора (content_types=[‘types’]). Таким чином, бот надаватиме інформацію по конкретній кнопці.

Після задання функції «def bot_message(message)», окремим рядком ми уточнюємо, що тип переписки, який ведеться із користувачем – приватний (if message.chat.type == ‘private’)

Зазначимо: при змінній markup з функцією «types.ReplyKeyboardMarkup», поруч в дужках ми будемо вказувати ширину рядка, тобто кількість кнопок, яка буде розташовуватись на одній лінії. Робитимемо ми це через команду «row_width», значення якої буде представлене у вигляді цифри.

```
# Категорії продуктів
@bot.message_handler(content_types=['text'])
def bot_message(message):
    if message.chat.type == 'private':
        if message.text == 'Овочі':
            markup = types.ReplyKeyboardMarkup(row_width=2)
            item1 = types.KeyboardButton('Бульбоплоди та коренеплоди')
            item2 = types.KeyboardButton('Капустяні')
            markup.row(item1, item2)
            item3 = types.KeyboardButton('Цибульні')
            item4 = types.KeyboardButton('Салатно-шпинатні та пагонові')
            markup.row(item3, item4)
            item5 = types.KeyboardButton('Пряносмакові')
            item6 = types.KeyboardButton('Томатні')
            markup.row(item5, item6)
            item7 = types.KeyboardButton('Гарбузові')
            back = types.KeyboardButton('Назад')
            markup.row(item7, back)
            markup.add()
            bot.send_message(message.chat.id, 'Овочі', reply_markup=markup)
```

Рисунок 2.22 – Фрагмент коду з даними про види певної категорії продуктів

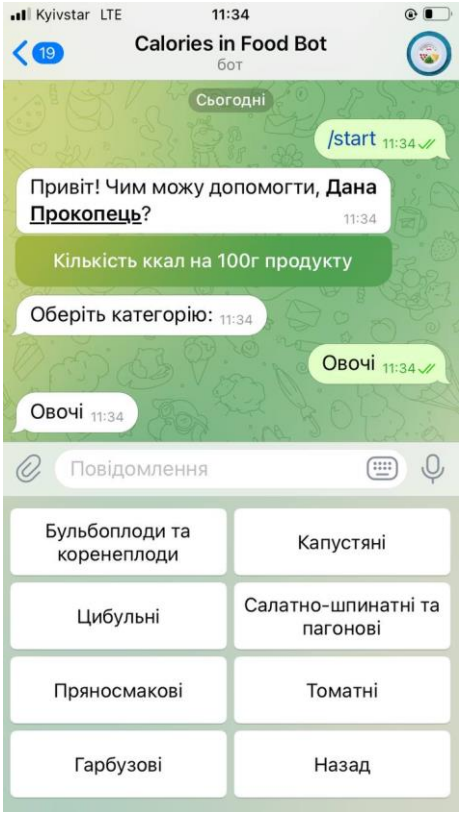



Рисунок 2.23 – Відображення меню всередині категорії «Овочі»

Таким же чином введемо дані по одній зі вказаних на попередньому скріншоті під категорій – «Бульбоплоди та коренеплоди». Та перевіримо, як бот відображатиме цю інформацію.

```
174    # Підгрупи категорії "Овочі"
175        if message.chat.type == 'private':
176            if message.text == 'Бульбоплоди та коренеплоди':
177                markup = types.ReplyKeyboardMarkup(row_width=2)
178                item1 = types.KeyboardButton('Картопля')
179                item2 = types.KeyboardButton('Морква')
180                markup.row(item1, item2)
181                item3 = types.KeyboardButton('Редиска')
182                item4 = types.KeyboardButton('Буряк')
183                markup.row(item3, item4)
184                item5 = types.KeyboardButton('Селера')
185                back = types.KeyboardButton('Назад до "Овочі"')
186                markup.row(item5, back)
187                markup.add()
188                bot.send_message(message.chat.id, 'Бульбоплоди та коренеплоди', reply_markup=markup)
```

Рисунок 2.24 – Список продуктів групи «Бульбоплоди та коренеплоди», категорії «Овочі»

Тепер переглянемо, як це виглядає в боті.


Рисунок 2.25 – Відображення меню всередині підгрупи «Бульбоплоди та коренеплоди», категорії «Овочі»

Далі нам потрібно вказати інформацію про калорії по кожному з представлених продуктів. Виводитись вони будуть в форматі звичайного текстового повідомлення шляхом команди «message.text».

```
# Калорії продуктів категорії "Овочі", підгрупа "Бульбоплоди та коренеплоди"
        if message.text == 'Картопля':
            mess = f'К-ть ккал на 100 г продукту "Картопля" = 82 ккал (варена), 90 ккал (печена), 142 ккал (смажена)'
            bot.send_message(message.chat.id, mess)
        elif message.text == 'Морква':
            mess = f'К-ть ккал на 100 г продукту "Морква" = 35 ккал (сира), 25 ккал (варена)'
            bot.send_message(message.chat.id, mess)
        elif message.text == 'Редиска':
            mess = f'К-ть ккал на 100 г продукту "Редиска" = 21 ккал'
            bot.send_message(message.chat.id, mess)
        elif message.text == 'Буряк':
            mess = f'К-ть ккал на 100 г продукту "Буряк" = 52 ккал (сирий), 45 ккал (варений)'
            bot.send_message(message.chat.id, mess)
        elif message.text == 'Селера':
            mess = f'К-ть ккал на 100 г продукту "Селера" = 44 ккал'
            bot.send_message(message.chat.id, mess)
```

Рисунок 2.26 – Фрагмент коду з інформацією про кількість калорій в продуктах підгрупи «Бульбоплоди та коренеплоди», ч.1

```
        elif message.text == 'Назад до "Овочі"':
            markup = types.ReplyKeyboardMarkup(row_width=2)
            item1 = types.KeyboardButton('Бульбоплоди та коренеплоди')
            item2 = types.KeyboardButton('Капустяні')
            markup.row(item1, item2)
            item3 = types.KeyboardButton('Цибульні')
            item4 = types.KeyboardButton('Салатно-шпинатні та пагонові')
            markup.row(item3, item4)
            item5 = types.KeyboardButton('Пряносмакові')
            item6 = types.KeyboardButton('Томатні')
            markup.row(item5, item6)
            item7 = types.KeyboardButton('Гарбузові')
            back = types.KeyboardButton('Назад')
            markup.row(item7, back)
            markup.add()
            bot.send_message(message.chat.id, 'Овочі', reply_markup=markup)
```

Рисунок 2.27 – Фрагмент коду з інформацією про кількість калорій в продуктах підгрупи «Бульбоплоди та коренеплоди», ч.2

На цьому алгоритм запиту/видачі інформації по кількості калорій в 100 г продукту закінчується. Як бачимо, для цього користувачу потрібно пройти наступний шлях: «/start» - «Кількість ккал на 100 г продукту» - «Категорія» - «Підгрупа» - «Назва продукту». Переглянемо, що вийшло.

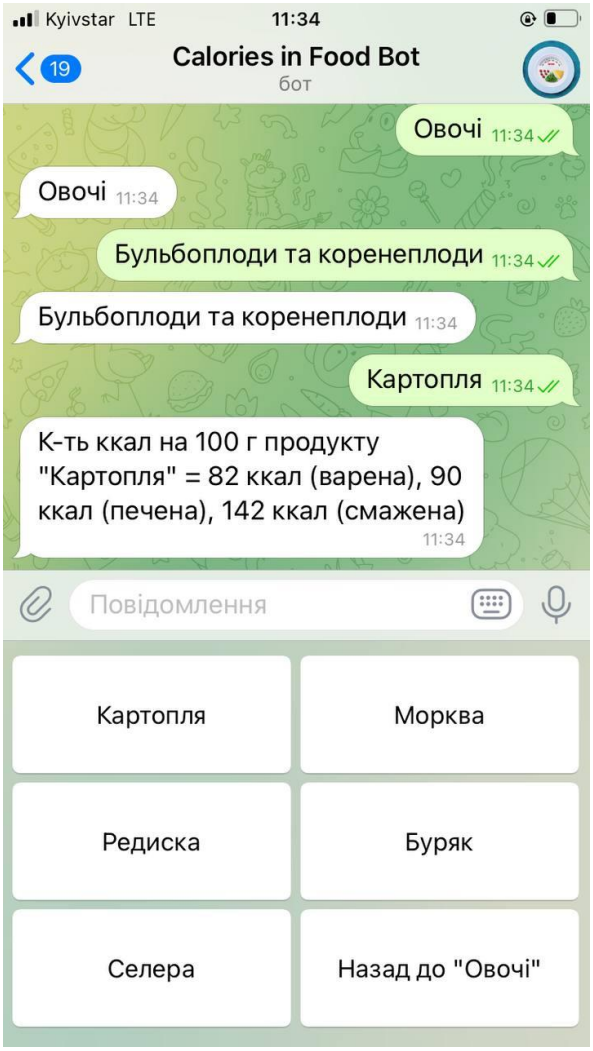



Рисунок 2.28 – Демонстрація роботи бота на прикладі продукту «Картопля»

Як бачимо, все працює чітко та без перебоїв. Бот безпомилково надає інформацію у відповідь  на конкретний запит від користувача. В залежності від типу продукту, в бот записана інформація щодо кількості калорій в по-різному приготованому продукті. В прикладі, продемонстрованому на Рис. 2.27, видно, що бот надав інформацію по енергетичній цінності (вмісту калорій) в картоплі, що приготована у трьох різних станах: у звареній картоплі, в запеченій картоплі та підсмаженій на пательні. У всіх трьох випадках кількість калорій абсолютно різна. На деякі продукти спосіб приготування майже не впливає, але не у випадку із картоплею.

Наостанок зазначимо, що задля нескінченної роботи нашого боту ми в кінці прописуємо команду «bot.polling(none_stop=True)».

```
1582    bot.polling(none_stop=True)
```

Рисунок 2.29 – Команда для неперервної роботи коду, що контролює роботу бота

Загальний обсяг програмного коду налічує 1581 рядок. Всі помилки та невідповідності були знайдені та виправлені ще в ході написання коду завдяки вбудованому в програму відладчику, який повідомляв про неправильні відступи, некоректний синтаксис, повторювані змінні в різних частинах коду, зайві елементи та неправильні посилання на ту чи іншу команду.

Алгоритм роботи боту зображено на діаграмі:

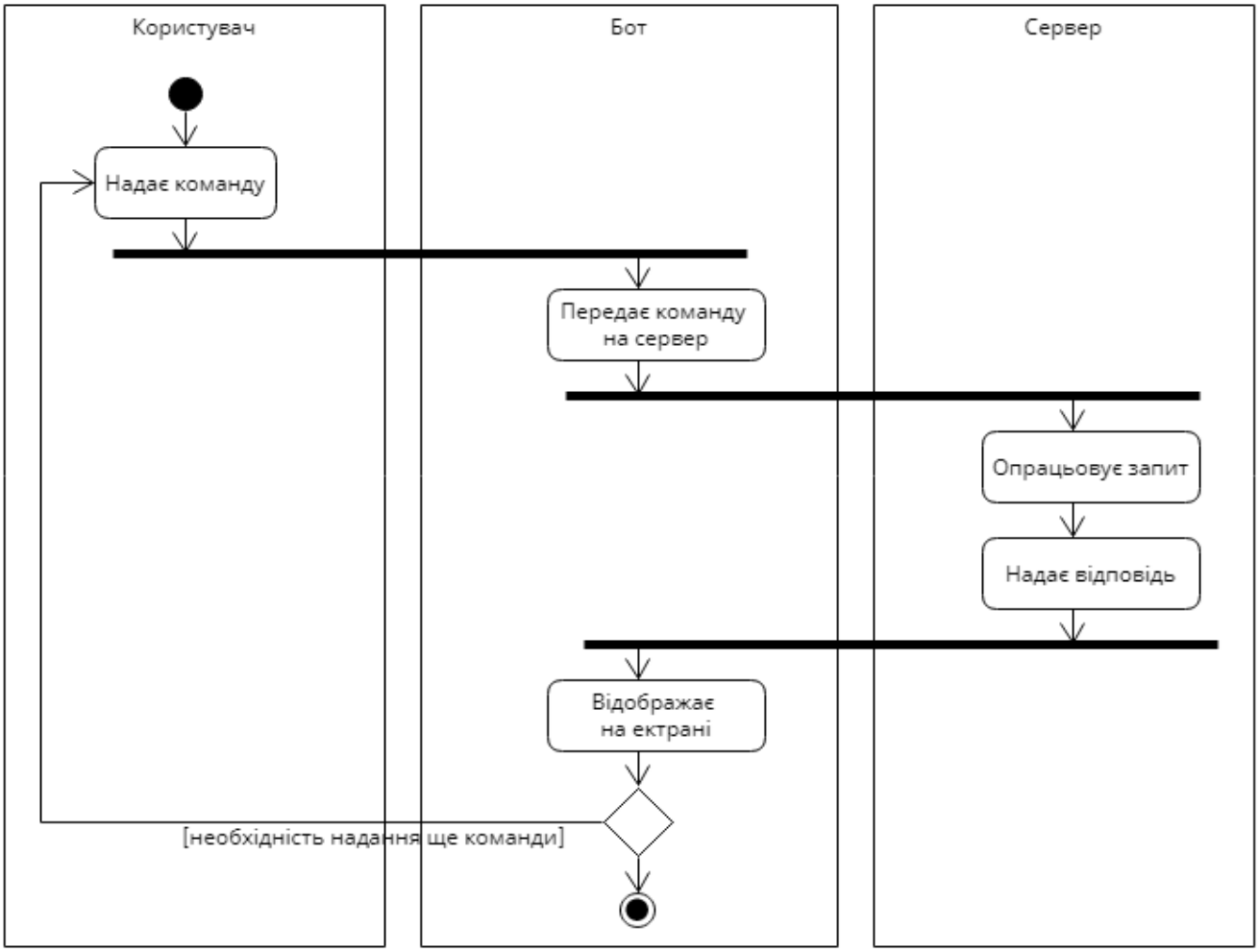



Рисунок 2.30 – Узагальнений алгоритм роботи Telegram-ботів

# 3 ТЕСТУВАННЯ ТА ОЦІНКА ПЕРСПЕКТИВ ОБ'ЄКТУ РОЗРОБКИ

## 3.1 Методи тестування чат-ботів

До заходів з тестування чат-боту, які проводяться перед його запуском, відносяться такі процеси як:

- загальне тестування;
- ручне тестування;
- пост-виробниче тестування;
- тестування домену;
- A/B-тест.

Розберемо детальніше кожен з них.

1. Загальне тестування – представляє з себе перевірку у форматі питання/відповідь. Ці питання орієнтовані на максимально легкі запитання, на які здатний відповісти навіть найпростіший бот. Суть даної перевірки – переконатись, що бот не вийде з ладу на першому етапі тесту і зможе підтримати не квапливу бесіду.

Рисунок 3.1 – Приклад листування з чат-ботом

2. Ручне тестування – тестування чат-боту для конкретного способу використання або для конкретного каналу. Щоб вручну протестувати бота, треба перевірити URL-код сторінки.


Рисунок 3.2 – URL-код сторінки Telegram-бота @CaloriesInFood

3. Пост-виробниче тестування – під час нього бота перевіряють на предмет того, як він відповідає на реальні питання користувачів. Оцінюється реакція бота і на її основі робляться висновки, чи треба вносити зміни в базу знань бота для розширювати діапазон ситуацій, в яких бот має дати клієнту більш точну відповідь.

4. Тестування домена – на цьому етапі аналізуються мова та вирази, щоб запевнитись, що чат-бот здатний відповідати на специфічні для домену запити. Неможливо охопити кожен конкретний тип запитань, що стосуються цієї конкретної області, але тести для конкретної предметної області мають поділятись на категорії, щоб гарантувати, що ключові категорії охоплюються автоматичним тестом.

5. A/B тести – порівняння двох версій одного й того самого продукту, щоб побачити, яка з них краще працює. Дане тестування знайшло широке застосування в різноманітних галузях маркетингу, однак для чат-ботів існує не так багато альтернатив A/B-тестуванню. Цей процес можна змоделювати у

вигляді двох окремих етапів тестування дизайну чат-боту. Перший визначає візуальні чинники боту – такі як дизайн, колір та розташування чат-боту на веб-сторінці. Другий – розмовні фактори з точки зору якості та продуктивності. Ці два фактори перевіряються, щоб забезпечити кращий користувацький досвід.

Необхідно також зазначити, що не кожен вид тестування підходить і взагалі можливий для того чи іншого боту. Мова йде про види тестування в цілому, аби краще розуміти, що в принципі перевіряється в ході тестування, чим вони відрізняються і який спосіб найліпше пасуватиме для того чи іншого месенджера, боту чи платформи.

### 3.2 Тестування Telegram-ботів

Що стосується конкретно Telegram-ботів, тут також присутня своя система тестування. Насправді, в мережі не надто багато матеріалів на тему того, як правильно тестувати подібних ботів.

Для того, щоб краще розуміти, що саме ми тестуємо, в аналітичних цілях наведемо перелік базових можливостей платформи Telegram.

1. Відправлення повідомлень з можливістю синхронізації з підключеними пристроями.
2. Наявність версії для десктопу.
3. Наявність нікнейму.
4. Можливість підключення декількох акаунтів.
5. Групові чати.
6. Параметри розширення групового чату.
7. Пошук співбесідника через символ @.
8. Розробка каналу.
9. Зберігання інформації в хмарному сховищі.
10. Безпека.
11. Наявність ботів.

Тепер перейдемо безпосередньо до характерних особливостей функціонування ботів всередині Telegram.

Таблиця 3.1 – Перелік характерних особливостей месенджеру Telegram

| **Особливість** | **Характеристика** |
|---|---|
| 1. Базова концепція | Інтеграція з обраними сервісами в межах платформи Telegram |
| 2. Номінальна реалізація | Bot API |
| 3. Атрибути ботів | Адреса, ім’я, фото, короткий опис |
| 4. Номінальні обмеження на створення | Відсутні |
| 5. Які види змісту підтримуються | Все, що підтримується на платформі Telegram (особисті дані, файли, опитування, місцезнаходження) |
| 6. Рівень комунікації в чатах | Так, за допомогою команд, клавіатур і текстових повідомлень |
| 7. Можливість монетизації | Так |
| 8. Додавання ботів до інших чатів | Так |
| 9. Дозвіл на бесіду | Так |
| 10. Розробка ботів | За допомогою боту @BotFather / Bot API |

З характеристиками також розібрались. Відзначимо, що є 2 типи ботів:

1. Чат-боти з фіксованими даними в базі даних. Їхня робота запрограмована на обмежений спектр сценаріїв поведінки (наприклад, кнопковий бот та бот-суфлер).

2. Чат-бот на базі штучного інтелекту з параметрами самонавчання та самооновлення інформації (наприклад, «розумний бот»).

Чат-боти з фіксованими даними, як вважається, надійніші в роботі, бо не мають напрацювань AI і, таким чином, не можуть вийти у програмістів з-під контролю. Чат-бот, наділений штучним інтелектом, є більш «чуйним», бо їхній інтелект дозволяє їм реагувати на те, що відбувається, по ситуації.

В першому розділі ми відзначали низку переваг, якими наділені боти в Telegram. Зараз же пропоную розглянути їхні недоліки:

- забагато функцій;
- часом нестандартні дії користувача можуть визвати збій в роботі;
- підходить не для всіх типів бізнесу.

Спочатку, ще до тестування, необхідно проаналізувати, з якою метою розробляється чат-бот, хто потенційна цільова аудиторія та які завдання він покликаний вирішити. Варто уточнити, які команди при взаємодії із чат-ботом можуть використовуватись у майбутньому. Напевно, список цих команд буде сильно відзначитися. Ще треба розуміти, під які країни розроблятиметься чат-бот і які іноземні мови підтримуватимуться. І можна позначити, на які платформи буде розрахований бот.

На початку процесу тестування обов'язково потрібно опрацювати базові позитивні сценарії, щоб розуміти проаналізувати, наскільки інтуїтивно зрозумілою є суть спільної роботи з чат-ботом. На цій стадії кожна дія користувача повинна знаходити фідбек з боку чат-бота.

За наявності різних локалізацій важливо також протестувати, чи є переклади для потенційних діалогів бота з іноземцями і чи правильно вони будуть побудовані. Необхідно враховувати характерні риси культури та традицій конкретної держави, граматичні та морфологічні особливості побудови словосполучень. Окрему увагу

слід приділити моментам, пов'язаним із реєстрацією, валідацією номерів телефонів та адрес, чи точно все відповідає формату країни, яка обрана.

І насамкінець потрібно переконатися, що однаково пройменовані об'єкти та елементи, які мають однакове позначення. У випадку, якщо обраний елемент планується видалити, обов'язково має бути видно повідомлення про підтвердження даної дії. Протестувати справжню відповідність завалідованому дизайну (макету) - добре, якщо графіка чат-бота витримана в одному стилі та кольоровій гамі для різних операційних платформ.

Зрозуміло, всі сценарії навряд чи вдасться протестувати, але важливо не забувати, що прямий обов'язок QA-інженера - перевірити бот на основі затвердженої тестової документації, виконати всі позитивні сценарії, обумовлені з клієнтом, і провести дослідницькі тести. Аналогічним перевірці веб-програми має бути і підхід до тестування чат-ботів. Втім, свої нюанси існують у будь-якій галузі, як і схожі моменти.

Проведемо для нашого Telegram-боту мануальне тестування з метою виявити помилки при виконанні тих чи інших команд.

Табл. 3.2 – Мануальне тестування боту @Calories_in_Food_Bot

| **Сценарій** | **Кроки виконання** | **Очікуваний результат** | **Фактичний результат** | **Пройдено?** |
|---|---|---|---|---|
| 1. Пошук продукту | 1) Обрати категорію | Перехід до нового меню | Відкрилось нове меню | Так |
| | 2) Обрати підкатегорію | Відображення переліку конкретних продуктів | З’явився список продуктів, по яких можна дізнатись ккал | Так |
| | 3) Вибрати потрібний продукт зі списку | Відображення кількості ккал на 100 г продукту | Бот видає дані по вмісту ккал у 100 г обраного продукту | Так |

| | | | | |
|---|---|---|---|---|
| 2. Перевірка правильності інформації про калорії | 1) Запит на відображення кількості калорій в продукті | Отримання числового значення кількості калорій, що містяться в 100 г продукту | Бот виводить текстове повідомлення з вказаною кількістю калорій | Так |
| | 2) Додаткова перевірка отриманої інформації в інших джерелах | Збіг даних, наданих ботом, з тими, що містяться на офіційному ресурсі | Дані від бота і на деяких сайтах незначно відрізняються | Ні |
| 3. Реакція на команду, відсутню в проекті | 1) Ввести в поле «Повідомлення» будь-який текст | Бот ніяк не буде на текст реагувати | Бот не відповідає на текстове повідомлення | Так |
| 4.Повернення до попереднього меню | 1) Обрати підкатегорію продуктів | Перехід до нового меню | Відкрилось нове меню | Так |
| | 2) Всередині неї натиснути кнопку «Назад» | Повернення до попередньої сторінки | Відкрилась сторінка, яка була до цього | Так |

Після проведення загального та ручного тестування нашого Telegram-боту, можна підвести деякі підсумки:

- чат-бот працює справно і коректно відображає інформацію у відповідь на запити користувача;
- час очікування фідбеку від боту – мінімальний, адже збережена лінійна структура написання коду;
- бот реагує виключно на кнопки, на текстові повідомлення реакції немає;

- бот має гарні перспективи для подальшого розвитку завдяки можливості постійно доповнювати базу даних, якою він володіє, та надавати більш якісний сервіс;
- бот зручний у використанні і не містить зайвих кнопок та функцій, тобто не є мультифункціональною, що дає можливість користувачу не відволікатись від того питання, з яким звернувся до нашого бота;
- всі недоліки, які були виявлені в ході тестування (непрацюючі клавіші, посилання на неіснуючі меню), були виправлені завдяки невичерпним можливостям правильно підібраного програмного середовища PyCharm, яке дозволило значно зекономити час, самостійно відшукавши та допомігши виправити помилки різного рівня складності;
- єдиним недоліком можна вважати розбіжність на 1-2 ккал в даних по деяких продуктах порівняно з інформацією на інших джерелах, що обумовлено порівнянням їх незалежно від сорту, виробника та інших харчових особливостей.

### 3.3 Перспективи Telegram-ботів

Не підлягає сумніву той тезис, що ми живемо в століття високих технологій, де все дуже стрімко розвивається, діджиталізується і вимагає від суспільства постійної адаптації. Навряд чи років 10-15 назад, коли тільки почали набувати масового поширення соціальні мережі, багато хто замислювався про те, що існуватиме подібна платформа, що вона буде такою багатофункціональною і що багато завдань виконуватиме та багато сервісів надаватиме не жива людина, а роботизована система, яку сьогодні ми знаємо як чат-бот.

За статистичними даними, місячна кількість активних користувачів Telegram зросла з 500 млн людей у квітні 2022-го року до 700 млн людей на момент січня 2023-го року. Аудиторія платформи зростає більше, ніж на 40% кожен рік, починаючи з 2013 року. Щомісячно за її допомогою спілкуються, листуються та відправляють файли 8,8% населення всієї земної кулі. За неповний 2023-й рік додаток в Google Play вже встигли скачати більше, ніж 1 млрд разів,

що дозволила месенджеру зайняти 2-ге місце в топі найбільш популярних месенджерів по встановленню в нинішньому році.

Зважаючи на те, як стрімко зростає кількість користувачів на платформі Telegram, і те, який широкий спектр задач можуть вирішувати чат-боти в ньому, не виникає сумнівів щодо того, що популярність та значимість ботів у Telegram з часом ставатиме тільки більше. Головним чином чат-боти приноситимуть користь сфері бізнесу, яка активно інтегрує усіх можливих ботів в свої робочі механізми.

Загалом, перспективи можна сформулювати та виділити в наступні пункти:

- економія часу роботи людей;
- автоматизація та оптимізація процесів;
- розширення спектру послуг та сервісів;
- розвиток сфер бізнесу, маркетингу, освіти, логістики, журналістики;
- зменшення навантаження на сайти;
- прогрес штучного інтелекту.

Дійсно, подібні боти значно зменшать кількість часу, яку робітники витрачають на виконання рутинних повторюваних дій та процесів. Завдяки Telegram-ботам більшість з них стануть автоматизованими, що допоможе значно оптимізувати ланцюг роботи більшості фахівців та покращити якість і збільшити обсяг виконуваної роботи загалом.

В першому розділі було розглянуто 8 видів ботів. Але навряд чи розробники планують на цьому зупинятись. Зважаючи на те, як стрімко розвивається технологія і як прогресує штучний інтелект, з кожним роком кількість напрямків та послуг, які надаватиме чат-бот, зростатиме, що позитивно впливатиме на підприємтсва і на соціум. Ті боти, що вже існують, активно впливають на сфери бізнесу, маркетингу, менеджменту, освіти, логістики та багато інших.

Також зазначимо, що боти, які прив’язані до якихось сайтів, допомагають зняти значну частку навантаження на них. Як приклад, можна навести вміння ботів з нейромережею генерувати заявки клієнтів через надання користі при зверненні. Такий підхід дозволяє набагато дешевше, а інколи й зовсім безкоштовно, приймати

та обробляти вхідні заявки клієнтів. Одна невелика компанія зізнавалась, що завдяки цій здатності Telegram-бота вона змогла за 2 тижні абсолютно безкоштовно залучити 1000 нових користувачів.

Рисунок 3.3 – Штучний інтелект, якби був людиною

Також Telegram-боти можуть використовуватись в маркетингових цілях – наприклад, для створення персоналізованих пропозицій чи надання клієнтам порад та рекомендацій стосовно тих чи інших продуктів та послуг. А ще чат-боти можна використовувати для відстеження настрою клієнтів, аби отримати важливі дані про їхні побажання, вподобання та клієнтські звички – як хороші, так і не дуже. Ще можна виокремити здатність ботів формувати цілі канали на основі своїх підписників, а також робити спеціалізовані розсилки, котрі можна проводити прямо всередині інструменту, підігріваючи інтерес зі сторони користувачів.

Не дивлячись на те, що боти на базі месенджеру Telegram створюються вже більше, ніж 8 років, ця ніша є достатньо свіжою та не сильно дослідженою, що дозволяє практично кожному охочому пробувати та намагатись з допомогою ботів у Telegram досягати поставлених цілей, розкручувати бізнес (не важливо, великий, середній чи малий), шукати та заохочувати нових клієнтів тощо.

Якщо говорити конкретно про Telegram-бот, що розроблявся в рамках даної дипломної роботи освітнього рівня бакалавр, то він був створений як щось, що в

подальшому принесе значну користь людям. Так, це не пов’язано з моєю спеціальністю чи роботою, але погодьтесь: з кожним роком все більше людей долучаються до занять спортом, ведення активного способу життя, правильного та здорового харчування. Я переконана в тому, що багатьом з нас час від часу стає цікаво не тільки те, на скільки корисний той чи інший продукт, який ми вживаємо, але й те, на скільки він калорійний. І мова йде не тільки про солодощі чи алкоголь, але навіть про овочі, фрукти та м’ясні вироби.

Та чи комусь хочеться витрачати час, щоб нишпорити по просторах Інтернету з метою дізнатись інформацію про 1-2 продукти? Набагато зручніше та більш функціонально, коли все знаходиться в одному місці, в одній базі даних та знань. Саме для оптимізації цих пошуків і був написаний цей бот. І це ще далеко не остаточна його версія. Для дипломної роботи в базу були завантажені далеко не всі продукти, а лише основні з кожної категорії та підкатегорії, щоб можна було збагнути сам принцип роботи Telegram-бота і оцінити задумку.

Зокрема, спочатку була ідея створити друге меню, в якому б користувач мав можливість дізнатися кількість калорій не в окремому продукті, а в цілій страві. Але це виявилось не дуже зручно, бо якщо брати за основу ресторанне меню, то там не всі страви ми їмо регулярно. А по-домашньому всі готують по-різному. В подальшому можна ввести в програмний код боту калькулятор, який рахуватиме кількість калорій в продукті на конкретну кількість грам, а також сумарну кількість калорій на прийом їжі. Так, він буде брати до уваги вагу кожного продукту у страві, розраховувати кількість калорій конкретно в ньому, а потім складатиме значення «ккал» усіх позицій і буде видавати кінцевий результат.

Чат-боти вважаються одним з найбільш перспективних сегментів ПЗ зі штучним інтелектом, частка якого з кожним роком зростає. Бізнес використовує штучний інтелект не просто в якості альтернативи операторам підтримки, але також для глибокого аналізу поведінки, звичок та вподобань користувачів. Попит на чат-боти буде постійно зростати, а зниження вартості їхньої розробки стимулюватиме активний зріст даної індустрії.

# ВИСНОВКИ

1. В дипломній роботі в повному обсязі розглянуті основні функціональні можливості Telegram-боту, що використовується в багатьох сферах і є багатофункціональним постачальником різноманітних сервісів та послуг. Завдяки сучасним технологіям представники сфер бізнесу, маркетингу, логістики, обслуговування клієнтів та багатьох інших економлять купу часу та енергії, залишаючи роботизованій системі чорнову роботу. Telegram на рівні з багатьма іншими популярними месенджерами – зокрема, WhatsApp та Viber - робить вагомий крок в майбутнє, де подібні процеси по всьому світу будуть автоматизовані. Під час виконання дипломної роботи, був розроблений новий Telegram-бот, який покликаний допомагати людям розраховувати кількість калорій в їжі.

2. За допомогою програмного середовища PyCharm, що використовує для розробки мову програмування Python, а також за допомогою Telegram-бота BotFather, що слугує для створення нових чат-ботів на платформі, було з нуля створено систему, яка видає на запит користувача інформацію стосовно енергетичної цінності того чи іншого харчового продукту. Функціональні можливості PyCharm дозволили максимально зручно написати програмний код, детально прописати послідовність команд та операторів, які відповідають за розміщення та видачу даних щодо їжі. Вбудований в систему PyCharm відладчик допоміг швидко усунути помилки, що могли завадити побудові коду, а наявність в програмі у вільному доступі багатьох бібліотек, що спеціалізуються на роботі з чат-ботами, дозволила швидко перейти до запису коду.

3. Для розробки ефективного Telegram-боту, було опрацьовано необхідні теоретичні матеріали, обрано відповідні інструментальні засоби, викладено теоретичні основи та суть досліджуваної проблеми, визначені основні характеристики, можливості та особливості чат-ботів на платформі Telegram.

4. Проаналізовано предметну галузь. Складено діаграму предметної галузі.

5. Здійснено моделювання функціональності, складено діаграму прецедентів.

# ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. GeekBrains. ИТ-образование [Електронний ресурс] – Режим доступу: https://gb.ru/blog/chto-takoe-telegram/ (дата звернення 17.04.2023). – Назва з екрану.
2. SendPulse. Єдина платформа для маркетингу та продажів [Електронний ресурс] – Режим доступу: https://sendpulse.com/ru/knowledge-base/chatbot/telegram/create-telegram-chatbot (дата звернення 18.04.2023). – Назва з екрану.
3. БлокЯПрактикума. Роботи увійшли в чат: які бувають Телеграм-боти и для чого вони потрібні [Електронний ресурс] – Режим доступу: https://practicum.yandex.ru/blog/telegram-boty-kak-rabotayut-i-kak-nastroit (дата звернення 18.04.2023). – Назва з екрану.
4. 1PS.RU. Феномен телеграм-ботів [Електронний ресурс] – Режим доступу: https://1ps.ru/blog/dirs/2021/fenomen-telegram-botov-chto-eto-zachem-nuzhnyi-primeryi (дата звернення 19.04.2023). - Назва з екрану.
5. ІНКЛІЄНТ. Статистика Telegram в 2023 році [Електронний ресурс] – Режим доступу: https://inclient.ru/telegram-stats/ (дата звернення 10.05.2023). – Назва з екрану.
6. BlueScreen. Історія та еволюція чат-ботів [Електронний ресурс] – Режим доступу: https://bluescreen.kz/longread/10075/istoriia-i-evoliutsiia-chat-botov (дата звернення 21.04.2023). - Назва з екрану.
7. Python. Офіційний сайт компанії [Електронний ресурс] – Режим доступу: https://www.python.org/downloads/windows/ (дата звернення 25.04.2023). – Назва з екрану.
8. JetBrains. Офіційний сайт компанії [Електронний ресурс] – Режим доступу: https://www.jetbrains.com/pycharm/download/#section=windows (дата звернення 25.04.2023). – Назва з екрану.
9. Все про Telegram @tlgmzy [Електронний ресурс] – Режим доступу: https://telegramzy.ru/vzlom-bota/ (дата звернення 29.04.2023). – Назва з екрану.

10. EasyDoIt.ru [Електронний ресурс] – Режим доступу: https://www.easydoit.ru/telegram/kak-vzlamyvayut-telegram-botov/ (дата звернення 29.04.2023). – Назва з екрану.

11. Вікіпедія. Вільна енциклопедія [Електронний ресурс] – Режим доступу: https://ru.wikipedia.org/wiki/PyCharm (дата звернення 27.04.2023). – Назва з екрану.

12. Tlgrm.ru. Інформація для розробників [Електронний ресурс] – Режим доступу: https://tlgrm.ru/docs/bots (дата звернення 28.04.2023). – Назва з екрану.

13. Хабр. Все, про що має знати розробник Телеграм-ботів [Електронний ресурс] – Режим доступу: https://habr.com/ru/articles/543676/ (дата звернення 01.05.2023). – Назва з екрану.

14. CoderNet. Якою мовою програмування та яким чином пишуть ботів [Електронний ресурс] – Режим доступу: https://codernet.ru/articles/drugoe/na_kakom_yazyike_programmirovaniya_i_kakim_obrazom_pishut_botov/ (дата звернення 02.05.2023). – Назва з екрану.

15. Таблиця калорійності [Електронний ресурс] – Режим доступу: https://www.tablycjakalorijnosti.com.ua/tablytsya-yizhyi (дата звернення 26.04.2023). – Назва з екрану.

16. Studfile.net. Хлібобулочні вироби: класифікація, асортимент та оцінка якості [Електронний ресурс] – Режим доступу: https://studfile.net/preview/5436902/ (дата звернення 26.04.2023). - Назва з екрану.

17. Все для садівництва та городництва [Електронний ресурс] – Режим доступу: https://sad.agroc.com.ua/ovochi (дата звернення 26.04.2023). – Назва з екрану.

18. ChatCompose. Платформа чат-боту [Електронний ресурс] – Режим доступу: https://www.chatcompose.com/ru/ (дата звернення 04.05.2023). – Назва з екрану.

19. TestMatick. Quality is never too much [Електронний ресурс] – Режим доступу: https://testmatick.com/ru/chat-boty-viber-i-telegram-i-kak-pravilno-ih-testirovat/ (дата звернення 08.05.2023). – Назва з екрану.

# ДОДАТОК А

Основні фрагменти програмного коду Telegram-боту @CaloriesInFoodBot

```python
import telebot

from telebot import types

bot = telebot.TeleBot('6036106297:AAGRVwebE3IcYyq42RG_nYQnHDvDTi7D8sM')

# Клавіатура головного меню
# keyboard1 = types.ReplyKeyboardMarkup(True, True)
# item1 = types.KeyboardButton('Кількість ккал на 100г продукту')
# keyboard1.add(item1)


@bot.message_handler(commands=['start'])
def start(message):
    mess = f'Привіт! Чим можу допомогти, <b>{message.from_user.first_name} <u>{message.from_user.last_name}</u></b>?'
    markup = types.InlineKeyboardMarkup()
    item1 = types.InlineKeyboardButton('Кількість ккал на 100г продукту', callback_data='next1')
    markup.add(item1)
    bot.send_message(message.chat.id, mess, parse_mode='html', reply_markup=markup)


@bot.callback_query_handler(func=lambda callback: True)
def callback_message(callback):
    if callback.data == 'next1':
        mess = f'Оберіть категорію:'
        markup = types.ReplyKeyboardMarkup(row_width=2)
        item1 = types.KeyboardButton('Овочі')
        item2 = types.KeyboardButton('Фрукти')
        markup.row(item1, item2)
        item3 = types.KeyboardButton('Молочні продукти')
        item4 = types.KeyboardButton('Ковбасні вироби')
        markup.row(item3, item4)
        item5 = types.KeyboardButton('Хлібобулочні вироби')
        item6 = types.KeyboardButton('Кондитерські вироби')
        markup.row(item5, item6)
        item7 = types.KeyboardButton('Безалкогольні напої')
        item8 = types.KeyboardButton('Алкогольні напої')
        markup.row(item7, item8)
        item9 = types.KeyboardButton('Зернові культури')
        item10 = types.KeyboardButton('М`ясні вироби')
        markup.row(item9, item10)
        markup.add()
        bot.send_message(callback.message.chat.id, mess, parse_mode='html', reply_markup=markup)
# Категорії продуктів


@bot.message_handler(content_types=['text'])
def bot_message(message):
    if message.chat.type == 'private':
        if message.text == 'Овочі':
            markup = types.ReplyKeyboardMarkup(row_width=2)
            item1 = types.KeyboardButton('Бульбоплоди та коренеплоди')
            item2 = types.KeyboardButton('Капустяні')
            markup.row(item1, item2)
            item3 = types.KeyboardButton('Цибульні')
            item4 = types.KeyboardButton('Салатно-шпинатні та пагонові')
```

```
            markup.row(item3, item4)
            item5 = types.KeyboardButton('Пряносмакові')
            item6 = types.KeyboardButton('Томатні')
            markup.row(item5, item6)
            item7 = types.KeyboardButton('Гарбузові')
            back = types.KeyboardButton('Назад')
            markup.row(item7, back)
            markup.add()
            bot.send_message(message.chat.id, 'Овочі', reply_markup=markup)
        elif message.text == 'Фрукти':
            markup = types.ReplyKeyboardMarkup(row_width=3)
            item1 = types.KeyboardButton('Розоцвіті')
            item2 = types.KeyboardButton('Цитрусові')
            item3 = types.KeyboardButton('Кісточкові')
            markup.row(item1, item2, item3)
            item4 = types.KeyboardButton('Субтропічні')
            item5 = types.KeyboardButton('Тропічні')
            back = types.KeyboardButton('Назад')
            markup.row(item4, item5, back)
            markup.add()
            bot.send_message(message.chat.id, 'Фрукти', reply_markup=markup)
        elif message.text == 'Молочні продукти':
            markup = types.ReplyKeyboardMarkup(row_width=2)
            item1 = types.KeyboardButton('Молоко та вершки')
            item2 = types.KeyboardButton('Кисломолочні продукти')
            markup.row(item1, item2)
            item3 = types.KeyboardButton('Сири')
            item4 = types.KeyboardButton('Морозиво')
            markup.row(item3, item4)
            back = types.KeyboardButton('Назад')
            markup.row(back)
            markup.add()
            bot.send_message(message.chat.id, 'Молочні продукти', reply_markup=markup)
        elif message.text == 'Ковбасні вироби':
            markup = types.ReplyKeyboardMarkup(row_width=2)
            item1 = types.KeyboardButton('Варені')
            item2 = types.KeyboardButton('Напівкопчені')
            markup.row(item1, item2)
            item3 = types.KeyboardButton('Копчені')
            item4 = types.KeyboardButton('Інше')
            markup.row(item3, item4)
            back = types.KeyboardButton('Назад')
            markup.row(back)
            markup.add()
            bot.send_message(message.chat.id, 'Ковбасні вироби', reply_markup=markup)
        elif message.text == 'Хлібобулочні вироби':
            markup = types.ReplyKeyboardMarkup(row_width=2)
            item1 = types.KeyboardButton('Хліб')
            item2 = types.KeyboardButton('Булочні та здобні хлібобулочні')
            markup.row(item1, item2)
            item3 = types.KeyboardButton('Сухарні, бубличні')
            item4 = types.KeyboardButton('Хлібобулочні дієтичні')
            markup.row(item3, item4)
            item5 = types.KeyboardButton('Пироги, пиріжки, пончики')
```

```
            back = types.KeyboardButton('Назад')
            markup.row(item5, back)
            markup.add()
            bot.send_message(message.chat.id, 'Хлібобулочні вироби', reply_markup=markup)
        elif message.text == 'Кондитерські вироби':
            markup = types.ReplyKeyboardMarkup(row_width=2)
            item1 = types.KeyboardButton('Борошняні')
            item2 = types.KeyboardButton('Цукрові')
            markup.row(item1, item2)
            back = types.KeyboardButton('Назад')
            markup.row(back)
            markup.add()
            bot.send_message(message.chat.id, 'Кондитерські вироби', reply_markup=markup)
        elif message.text == 'Безалкогольні напої':
            markup = types.ReplyKeyboardMarkup(row_width=2)
            item1 = types.KeyboardButton('Мінеральні води')
            item2 = types.KeyboardButton('Соки, фреші')
            markup.row(item1, item2)
            item3 = types.KeyboardButton('Газовані напої')
            item4 = types.KeyboardButton('Чай, кава')
            markup.row(item3, item4)
            back = types.KeyboardButton('Назад')
            markup.row(back)
            markup.add()
            bot.send_message(message.chat.id, 'Безалкогольні напої', reply_markup=markup)
        elif message.text == 'Алкогольні напої':
            markup = types.ReplyKeyboardMarkup(row_width=2)
            item1 = types.KeyboardButton('Слабоалкогольні напої (1,5%-8%)')
            item2 = types.KeyboardButton('Алкогольні напої (8,1%-30%)')
            markup.row(item1, item2)
            item3 = types.KeyboardButton('Міцноалкогольні напої (від 30%)')
            back = types.KeyboardButton('Назад')
            markup.row(item3, back)
            markup.add()
            bot.send_message(message.chat.id, 'Алкогольні напої', reply_markup=markup)
        elif message.text == 'Зернові культури':
            markup = types.ReplyKeyboardMarkup(row_width=2)
            item1 = types.KeyboardButton('Злакові та гречані')
            item2 = types.KeyboardButton('Бобові')
            markup.row(item1, item2)
            back = types.KeyboardButton('Назад')
            markup.row(back)
            markup.add()
            bot.send_message(message.chat.id, 'Зернові культури', reply_markup=markup)
        elif message.text == 'М`ясні вироби':
            markup = types.ReplyKeyboardMarkup(row_width=2)
            item1 = types.KeyboardButton('Курятина')
            item2 = types.KeyboardButton('Баранина')
            markup.row(item1, item2)
            item3 = types.KeyboardButton('Свинина')
            item4 = types.KeyboardButton('Телятина')
            markup.row(item3, item4)
            item5 = types.KeyboardButton('Яловичина')
            back = types.KeyboardButton('Назад')
```

```
            markup.row(item5, back)
            markup.add()
            bot.send_message(message.chat.id, 'M`ясні вироби', reply_markup=markup)
# Підгрупи категорії "Овочі"
    if message.chat.type == 'private':
        if message.text == 'Бульбоплоди та коренеплоди':
            markup = types.ReplyKeyboardMarkup(row_width=2)
            item1 = types.KeyboardButton('Картопля')
            item2 = types.KeyboardButton('Морква')
            markup.row(item1, item2)
            item3 = types.KeyboardButton('Редиска')
            item4 = types.KeyboardButton('Буряк')
            markup.row(item3, item4)
            item5 = types.KeyboardButton('Селера')
            back = types.KeyboardButton('Назад до "Овочі"')
            markup.row(item5, back)
            markup.add()
            bot.send_message(message.chat.id, 'Бульбоплоди та коренеплоди', reply_markup=markup)
        elif message.text == 'Капустяні':
            markup = types.ReplyKeyboardMarkup(row_width=3)
            item1 = types.KeyboardButton('Капуста білокачанна')
            item2 = types.KeyboardButton('Червонокачанна')
            item3 = types.KeyboardButton('Кольорова')
            markup.row(item1, item2, item3)
            item4 = types.KeyboardButton('Брюссельська')
            item5 = types.KeyboardButton('Пекінська')
            back = types.KeyboardButton('Назад до "Овочі"')
            markup.row(item4, item5, back)
            markup.add()
            bot.send_message(message.chat.id, 'Капустяні', reply_markup=markup)
        elif message.text == 'Цибульні':
            markup = types.ReplyKeyboardMarkup(row_width=3)
            item1 = types.KeyboardButton('Цибуля ріпчаста')
            item2 = types.KeyboardButton('Цибуля зелена')
            item3 = types.KeyboardButton('Цибуля фіолетова')
            markup.row(item1, item2, item3)
            item4 = types.KeyboardButton('Часник')
            item5 = types.KeyboardButton('Цибуля-порей')
            back = types.KeyboardButton('Назад до "Овочі"')
            markup.row(item4, item5, back)
            markup.add()
            bot.send_message(message.chat.id, 'Цибульні', reply_markup=markup)
        elif message.text == 'Салатно-шпинатні та пагонові':
            markup = types.ReplyKeyboardMarkup(row_width=2)
            item1 = types.KeyboardButton('Салат зелений')
            item2 = types.KeyboardButton('Шпинат')
            markup.row(item1, item2)
            item3 = types.KeyboardButton('Щавель')
            item4 = types.KeyboardButton('Спаржа')
            markup.row(item3, item4)
            back = types.KeyboardButton('Назад до "Овочі"')
            markup.row(back)
            markup.add()
            bot.send_message(message.chat.id, 'Салатно-шпинатні та пагонові', reply_markup=markup)
        elif message.text == 'Пряносмакові':
```

```
            markup = types.ReplyKeyboardMarkup(row_width=2)
            item1 = types.KeyboardButton('Кріп')
            item2 = types.KeyboardButton('Петрушка')
            markup.row(item1, item2)
            item3 = types.KeyboardButton('Базилік')
            item4 = types.KeyboardButton('Хрін')
            markup.row(item3, item4)
            back = types.KeyboardButton('Назад до "Овочі"')
            markup.row(back)
            markup.add()
            bot.send_message(message.chat.id, 'Пряносмакові', reply_markup=markup)
        elif message.text == 'Томатні':
            markup = types.ReplyKeyboardMarkup(row_width=2)
            item1 = types.KeyboardButton('Помідори')
            item2 = types.KeyboardButton('Баклажани')
            markup.row(item1, item2)
            item3 = types.KeyboardButton('Перець чілі стручковий')
            item4 = types.KeyboardButton('Перець червоний болгарський')
            markup.row(item3, item4)
            back = types.KeyboardButton('Назад до "Овочі"')
            markup.row(back)
            markup.add()
            bot.send_message(message.chat.id, 'Томатні', reply_markup=markup)
        elif message.text == 'Гарбузові':
            mess = f'Гарбузові'
            markup = types.ReplyKeyboardMarkup(row_width=2)
            item1 = types.KeyboardButton('Кавун')
            item2 = types.KeyboardButton('Диня')
            markup.row(item1, item2)
            item3 = types.KeyboardButton('Огірок')
            item4 = types.KeyboardButton('Гарбуз')
            markup.row(item3, item4)
            item5 = types.KeyboardButton('Кабачок')
            item6 = types.KeyboardButton('Актор з фільму "Сутінки"')
            markup.row(item5, item6)
            back = types.KeyboardButton('Назад до "Овочі"')
            markup.row(back)
            markup.add()
            bot.send_message(message.chat.id, mess, parse_mode='html', reply_markup=markup)
        elif message.text == 'Назад':
            mess = f'Оберіть категорію:'
            markup = types.ReplyKeyboardMarkup(row_width=2)
            item1 = types.KeyboardButton('Овочі')
            item2 = types.KeyboardButton('Фрукти')
            markup.row(item1, item2)
            item3 = types.KeyboardButton('Молочні продукти')
            item4 = types.KeyboardButton('Ковбасні вироби')
            markup.row(item3, item4)
            item5 = types.KeyboardButton('Хлібобулочні вироби')
            item6 = types.KeyboardButton('Кондитерські вироби')
            markup.row(item5, item6)
            item7 = types.KeyboardButton('Безалкогольні напої')
            item8 = types.KeyboardButton('Алкогольні напої')
            markup.row(item7, item8)
            item9 = types.KeyboardButton('Зернові культури')
            item10 = types.KeyboardButton('М`ясні вироби')
            markup.row(item9, item10)
            markup.add()
            bot.send_message(message.chat.id, mess, parse_mode='html', reply_markup=markup)
```

```python
# Калорії продуктів категорії "Овочі", підгрупа "Бульбоплоди та коренеплоди"
        if message.text == 'Картопля':
            mess = f'К-ть ккал на 100 г продукту "Картопля" = 82 ккал (варена), 90 ккал (печена), 142 ккал (смажена)'
            bot.send_message(message.chat.id, mess)
        elif message.text == 'Морква':
            mess = f'К-ть ккал на 100 г продукту "Морква" = 35 ккал (сира), 25 ккал (варена)'
            bot.send_message(message.chat.id, mess)
        elif message.text == 'Редиска':
            mess = f'К-ть ккал на 100 г продукту "Редиска" = 21 ккал'
            bot.send_message(message.chat.id, mess)
        elif message.text == 'Буряк':
            mess = f'К-ть ккал на 100 г продукту "Буряк" = 52 ккал (сирий), 45 ккал (варений)'
            bot.send_message(message.chat.id, mess)
        elif message.text == 'Селера':
            mess = f'К-ть ккал на 100 г продукту "Селера" = 44 ккал'
            bot.send_message(message.chat.id, mess)
        elif message.text == 'Назад до "Овочі"':
            markup = types.ReplyKeyboardMarkup(row_width=2)
            item1 = types.KeyboardButton('Бульбоплоди та коренеплоди')
            item2 = types.KeyboardButton('Капустяні')
            markup.row(item1, item2)
            item3 = types.KeyboardButton('Цибульні')
            item4 = types.KeyboardButton('Салатно-шпинатні та пагонові')
            markup.row(item3, item4)
            item5 = types.KeyboardButton('Пряносмакові')
            item6 = types.KeyboardButton('Томатні')
            markup.row(item5, item6)
                item7 = types.KeyboardButton('Гарбузові')
                back = types.KeyboardButton('Назад')
                markup.row(item7, back)
                markup.add()
                bot.send_message(message.chat.id, 'Овочі', reply_markup=markup)
# Калорії продуктів категорії "М`ясні вироби", підгрупа "Телятина"
        if message.text == 'Яловичина відварена':
            mess = f'К-ть ккал на 100 г продукту "Яловичина відварена" = 254 ккал'
            bot.send_message(message.chat.id, mess)
        elif message.text == 'Яловичина тушкована, консерва':
            mess = f'К-ть ккал на 100 г продукту "Яловичина тушкована, консерва" = 220 ккал'
            bot.send_message(message.chat.id, mess)
        elif message.text == 'Яловичина, грудинка':
            mess = f'К-ть ккал на 100 г продукту "Яловичина, грудинка" = 217 ккал'
            bot.send_message(message.chat.id, mess)
        elif message.text == 'Яловичина, печінка':
            mess = f'К-ть ккал на 100 г продукту "Яловичина, печінка" = 157 ккал'
            bot.send_message(message.chat.id, mess)
        elif message.text == 'Яловичина, філейна вирізка':
            mess = f'К-ть ккал на 100 г продукту "Яловичина, філейна вирізка" = 113 ккал'
            bot.send_message(message.chat.id, mess)


bot.polling(none_stop=True)

```

**ДОДАТОК Б**

Презентаційні матеріали

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКА

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

**Розробка Telegram-боту для перевірки кількості калорій у їжі мовою Python**

Виконала студентка 4 курсу
групи ПД-42
Прокопець Дана Сергіївна
Керівник роботи
Старший викладач Гаманюк Ігор Михайлович

Київ 2023

# МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи**

  Автоматизація процесу розрахунку калорій у їжі шляхом впровадження застосунку для перевірки кількості калорій в їжі.

- **Об'єкт дослідження**

  Процес перевірки кількості калорій у їжі

- **Предмет дослідження**

  Telegram-бот для перевірки кількості калорій у їжі.

# СПИСОК ЗАДАЧ ДИПЛОМНОЇ РОБОТИ

1. Проаналізувати існуючі аналоги та виявити їх недоліки
2. Проаналізувати середовище розробки, що використано для розробки Telegram-боту
3. Проаналізувати технології розробки, що використовуються для розробки Telegram-ботів
4. Розробити вимоги для боту на базі результату аналізу існуючих Telegram-ботів
5. Спроектувати та розробити Telegram-бот на основі створених вимог



# ВИМОГИ ДО TELEGRAM-БОТУ

- Функціональні вимоги:

- можливість багаторазового звернення до боту;
- можливість пошуку харчового продукту.
- можливість підрахунку кількості калорій на 100г продукту
- можливість повторного підрахунку калорій

- Нефункціональні вимоги:

- простий у використанні інтерфейс;
- відсутність зайвих кнопок;
- мінімум команд та зрозумілий зв’язок між ними;
- створення інтерфейсу українською мовою;

# АНАЛІЗ АНАЛОГІВ

| Властивості | “Калькулятор калорій Ольги Базікало” | “Calories in Food Bot” |
|---|---|---|
| Середовище використання | Telegram | Telegram |
| Безкоштовний контент | – | + |
| Простий у використанні інтерфейс | + | + |
| Введення персональних даних користувача | + | – |
| Підрахунок рекомендованих калорій споживання на день | + | – |
| Підрахунок кількості калорій на 100г продукту | – | + |
| Можливість повторного підрахунку калорій | – | + |
| Інтерфейс українською мовою | – | + |

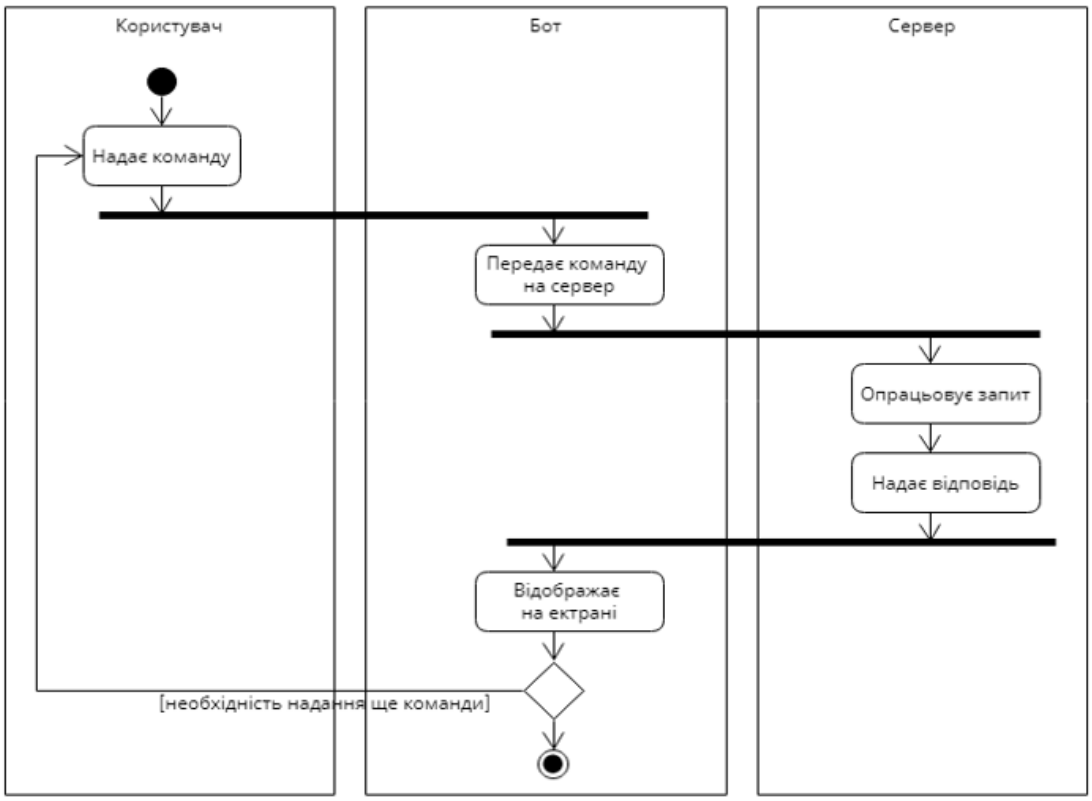


# ЗАСОБИ ТА СЕРЕДОВИЩЕ РОЗРОБКИ


Telegram

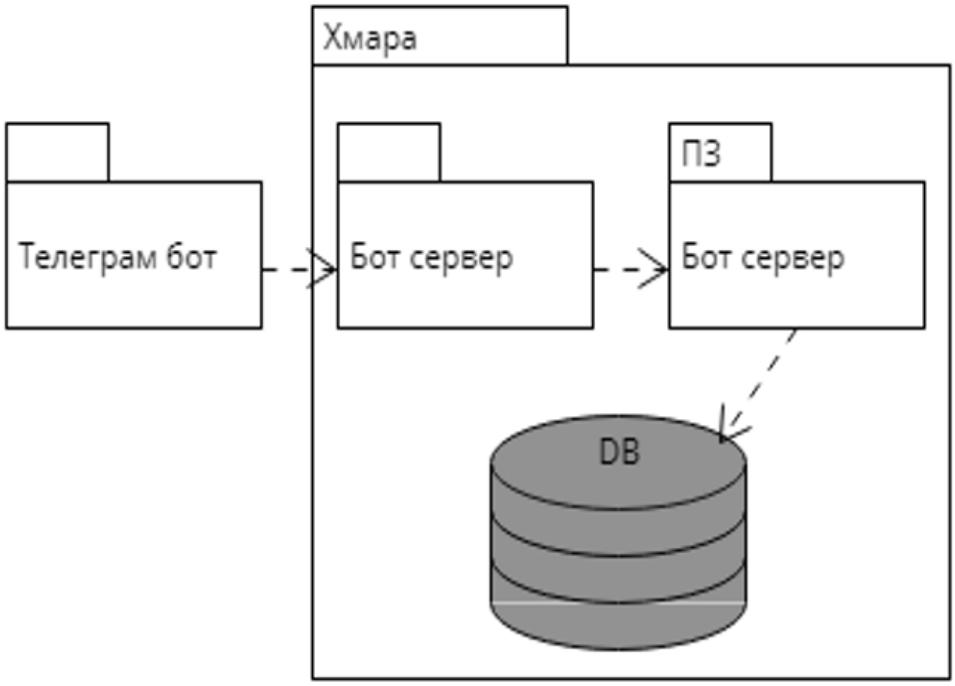

PyCharm

# УЗАГАЛЬНЕНИЙ АЛГОРИТМ РОБОТИ БОТА

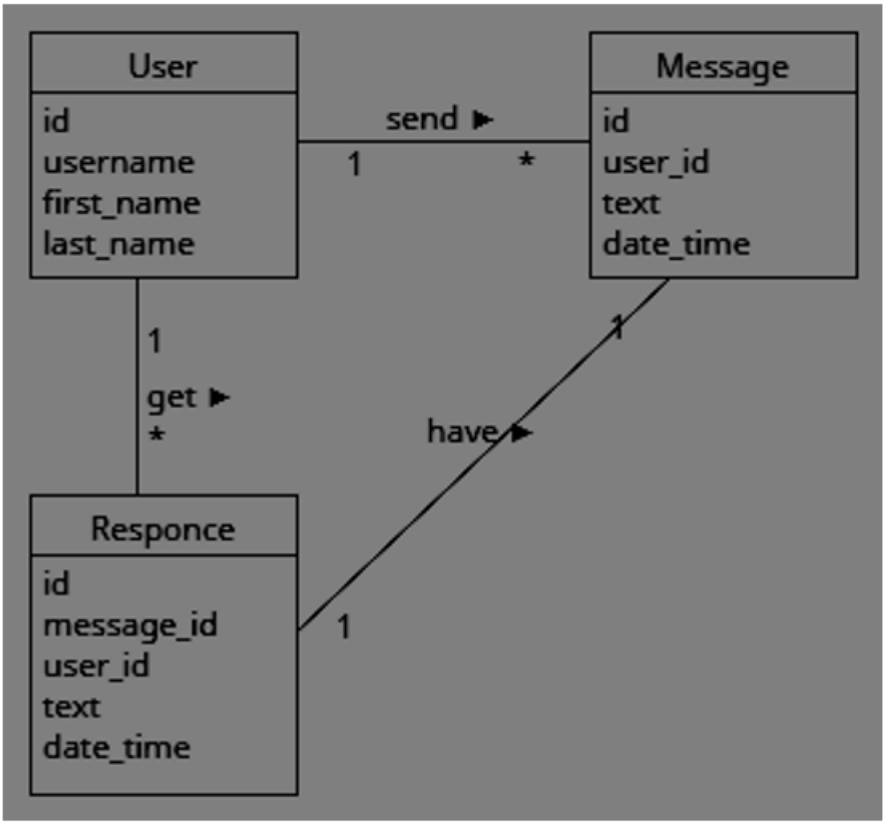





# АРХІТЕКТУРА TELEGRAM-БОТА

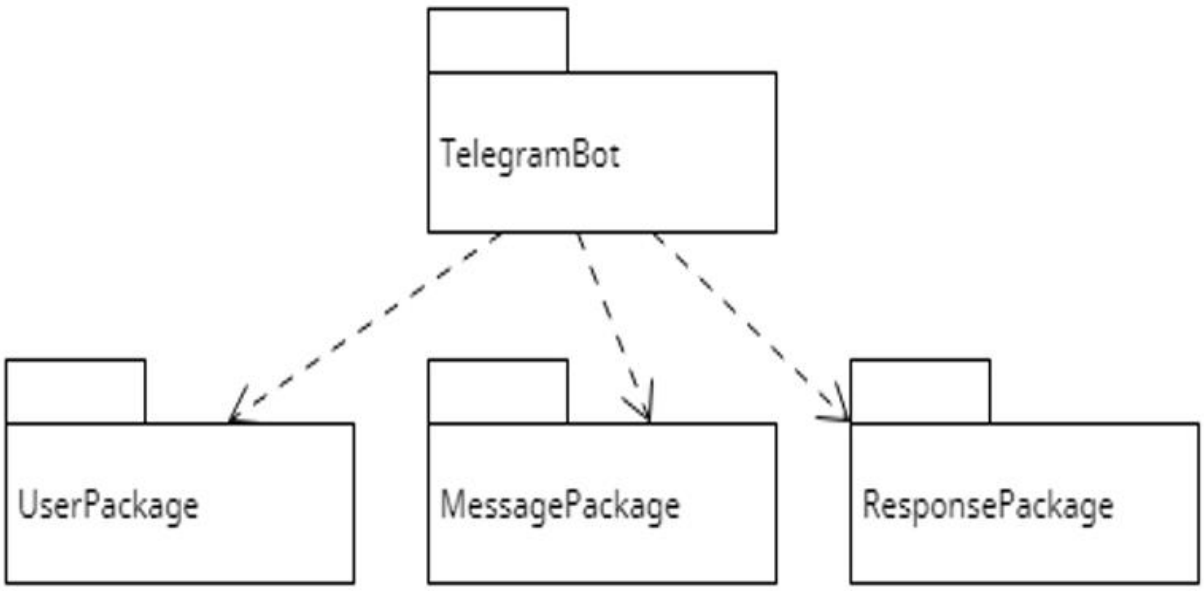


# ДІАГРАМА ПРЕДМЕТНОЇ ГАЛУЗІ

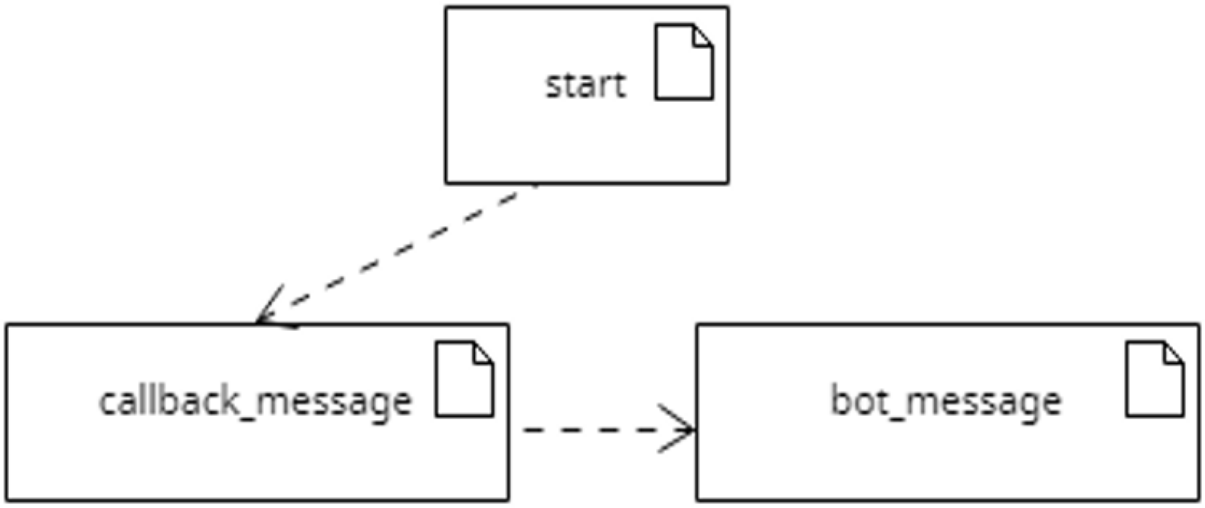





# ДІАГРАМА ПАКЕТІВ

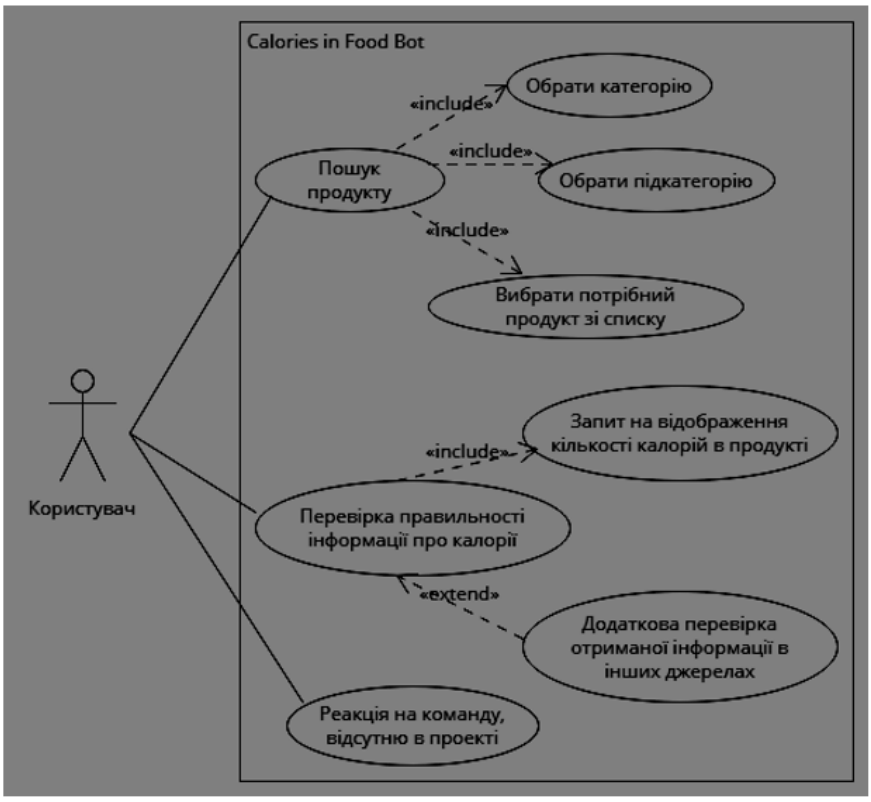


# ДІАГРАМА АРТЕФАКТІВ






# ДІАГРАМА ПРЕЦЕДЕНТІВ



# ІНТЕРФЕЙС TELEGRAM-БОТУ




# ВИСНОВКИ

1. Проаналізовано існуючі аналоги, їхні недоліки та переваги.
2. Проаналізовано програмне середовище, що використано для розробки Telegram-боту.
3. Складено діаграми предметної галузі, діаграми прецедентів
4. Проаналізовано технології розробки, що використовуються для розробки Telegram-ботів.
5. Розроблено бот відповідно до вимог на базі результатів аналізу існуючих Telegram-ботів.
6. Спроектовано та розроблено Telegram-бот на основі створених вимог.