МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

Кваліфікаційна робота бакалавра

ІНФОРМАЦІЙНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ІНТЕРНЕТ-МАГАЗИНУ З ІНТЕГРАЦІЄЮ У СОЦІАЛЬНИХ МЕРЕЖАХ

Здобувач освіти гр. ІН – 82 Михайло ВОЗНЕСЕНСЬКИЙ

Науковий керівник,
кандидат фізико-математичних наук,
старший викладач Дмитро ВЕЛИКОДНИЙ

Завідувач кафедри
доктор технічних наук, професор Анатолій ДОВБИШ

СУМИ 2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

Затверджую_____
Зав. кафедрою Довбиш А.С.
"_____" _____2022 р.

ЗАВДАННЯ

до кваліфікаційної роботи

здобувача вищої освіти четвертого курсу, групи ІН-82 спеціальності «122 – Комп'ютерні науки» денної форми навчання Вознесенського Михайла Олександровича.

Тема: «ІНФОРМАЦІЙНЕ ПРОГРАМЕ ЗАБЕЗПЕЧЕННЯ ІНТЕРНЕТ-МАГАЗИНУ З ІНТЕГРАЦІЄЮ У СОЦІАЛЬНИХ МЕРЕЖАХ»

Затверджена наказом по СумДУ

№_____ від _____2022 р.

Зміст пояснювальної записки: 1) аналітичний огляд; 2) постановка завдання; 3) вибір методу рішення завдання; 4) програмна реалізація; 5) аналіз результатів роботи.

Дата видачі завдання «_____»_____ 2022 р.

Керівник роботи _____Дмитро ВЕЛИКОДНИЙ

Завдання прийняв до виконання _____Михайло ВОЗНЕСЕНСЬКИЙ

РЕФЕРАТ

Записка: 48 стор., 27 рис., 1 додаток, 15 джерел.

Об’єкт дослідження – процес розробки програмного забезпечення чат-боту.

Мета роботи – розробка Telegram боту для замовлення техніки.

Методи дослідження – методи обробки інформації, методи розробки програмного забезпечення.

Результати – розроблений Telegram бот для замовлення техніки.

ЗАМОВЛЕННЯ ТЕХНІКИ, ОБРОБКА ІНФОРМАЦІЇ, ЧАТ-БОТ, TELEGRAM, PYTHON, SQLITE, ASYNCIO/AWAIT

ЗМІСТ

ВСТУП

Усім нам відомо, як стрімко розвиваються комп’ютерні технології у наш час. З кожним роком компанії створюють все більше продуктів, що покращують і полегшують наше життя сьогодні. Зовсім не секрет, що кілька десятиліть тому ми не мали тих технологій, які зараз стали рутинними та аж ніяк не особливим, та які зараз набирають оберти у попиті. Це все – наше сьогодення. І для нас вже не проблема зв’язатися з кимось на великій відстані. Дуже стрімко з’являються нові функції, які швидко підхоплюють всі розробники, додають щось своє або удосконалюють вже існуючі.

Одним із таких продуктів є месенджер. На мою думку, месенджери – це найкращий спосіб спілкуватися з кимось на відстані, бо завдяки привабливому інтерфейсу та швидкій роботі користуватися ними дуже зручно. Але це не все, на що вони здатні. За допомогою месенджерів можна спілкуватися с бізнесом (за допомогою спеціальних ботів), створювати канали та постити там все, що заманеться. Також можна створювати чати на різні тематики, де буде можливість знайти друзів або просто людей, яким подобається все, що і Вам.

1. АНАЛІТИЧНИЙ ОГЛЯД

1.1 Що таке месенджери?

Месенджери – це програми для миттєвого обміну текстовими повідомленнями, аудіозаписами, фотографіями та іншими мультимедіа. Програми встановлюються на комп’ютер, смартфон, планшет та працюють через інтернет.

Першим месенджером у світі став EMISARI (Emergency Management Information Systems and Reference Index – Інформаційні системи управління надзвичайними ситуаціями та довідковий індекс). Його створив фізик та математик Мюррей Турофф для уряду США у 1971 році [1].

З допомогою EMISARI державні службовці могли швидко зв’язуватися друг з одним. Вони підключалися до мережі міжміськими телефонними лініями через телетайпи – електромеханічні друкарські машини. Так влада США могла швидко реагувати на кризові ситуації в країні але звичайні громадяни EMISARI не використовували.

Першим месенджером, доступним для цивільних, став Internet Relay Chat. Його розробив фінський програміст Яркко Ойкарінен у 1988 році. Програма була популярна в Європі та Північній Америці – у 2009 році на серверах спілкувалося понад 500 тисяч користувачів.

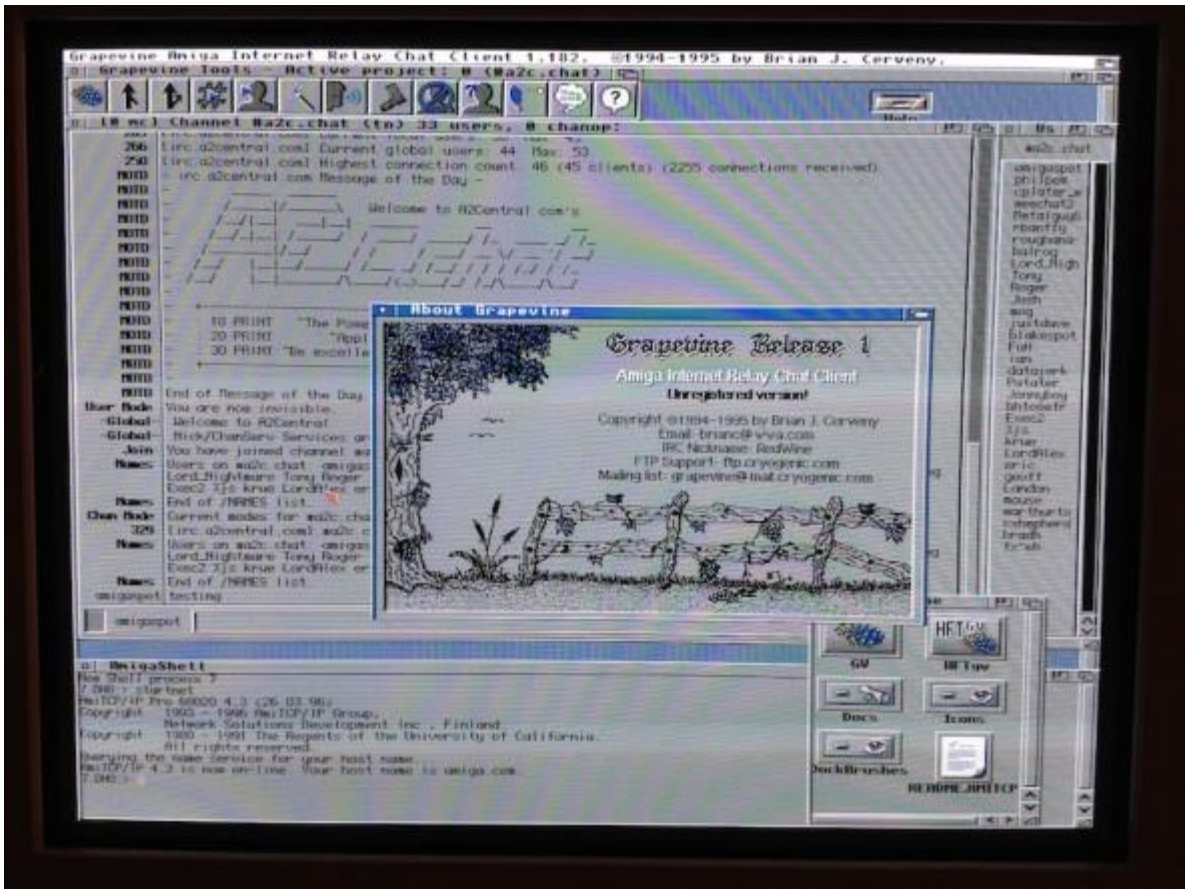

Рисунок 1.1 - інтерфейс Internet Relay Chat [1]

Я вважаю, що якщо потрібно швидко комусь написати або відповісти на повідомлення, то краще це робити за допомогою месенджерів. Так, соціальні мережі, як і месенджери, призначені для спілкування. Але вони надто навантажені – поєднують у собі чат, сервіс знайомств, стрічку новин та особистий блог в одному додатку. У свою чергу месенджери призначені для приватного спілкування між декількома людьми. Вони не перевантажені іншими функціями, при користуванні не є можливим відволіктися на новини або щось інше.

У 2020 році послуги для обміну повідомленнями стали на 20% популярнішими, ніж соцмережі За даними дослідження креативного агенства «ZAK», користувачі віддають перевагу месенджерам, оскільки це більш

закритий простір. 43% опитаних у віці від 16 до 30 років вважають, що у соціальних мережах «Instagram» та «Facebook» занадто багато людей: будь-який користувач може зайти на твою сторінку та побачити особисті фотографії, записи, а у месенджерах можна вибирати, якою інформацією ділитися і з ким [1].

1.2 Telegram

Telegram – це сучасний, безкоштовний, кросплатформний клауд-месенджер з функціями VoIP для смартфонів, планшетів та ПК, який дозволяє обмінюватися текстовими, голосовими та відеоповідомленнями, наліпками (стікерами), фотографіями та файлами багатьох форматів. Також має функції відео- і аудіодзвінків, організації відеоконференцій у групах і каналах.

Проект створив Павло Дуров, співзасновник соцмережі «ВКонтакте». За його словами, початкова ідея застосунку виникла в 2011 році, коли до його дверей приходили спецпризначенці [12]. Після цього, Павло Дуров відразу ж написав своєму братові Миколі. Тоді ж він усвідомив, що у нього немає безпечного способу комунікації з братом.

1.3 Технологія Telegram

Для месенджера був створений протокол MTProto, що передбачає використання декількох протоколів шифрування. Під час авторизації і аутентифікації використовуються алгоритми RSA-2048, DH-2048 для шифрування, під передачі повідомлень протоколу в мережу вони шифруються AES з ключем, відомим клієнту і серверу. Також застосовуються криптографічні хеш-алгоритми SHA-1 і MD5.

Однак, безпека під перехоплення повідомлень, що пересилаються з боку сервера Telegram, забезпечується лише в режимі «секретних» чатів, доступному з 8 жовтня 2013 року. Цей режим реалізує шифрування, при якому відправник і одержувач мають спільний лише для них ключ (end-to-end шифрування), із застосуванням алгоритму AES-256 у режимі IGE (англ. Infinite Garble Extension) для повідомлень, що пересилаються. На відміну від звичайного режиму, повідомлення в секретних чатах не розшифровуються сервером – історія листування зберігається лише на тих пристроях, на яких був створений чат.

При обміні файлами можна як відправити з пристрою, так і шукати файли в інтернеті, якщо використовується мобільна версія для IOS та Android. Програма використовує систему докачування файлів після обриву зв'язку. Єдина річ, яка мені не подобається стосовно файлів – розмір файлів, що передаються. Їх розмір обмежений до 2 ГБ Але, у зв'язку з останніми оновленнями (повідомлення Павла Дурова від 10 червня 2022 року), багато функцій та ліміти, в числі яких ліміт на розмір файлів, можна буде покращити або збільшити за підписку Telegram Premium.

Команда Telegram хоче, щоб додаток для обміну повідомленнями виділявся, пропонувавши швидкість і безпеку, а також покладався на зусилля спільноти [13].

1.4 Протокол MTProto

MTProto – криптографічний протокол, який використовується в системі обміні повідомленнями Telegram для шифрування листування користувачів. Протокол був розроблений Миколою Дуровим та іншими програмістами Telegram. Також на основі протоколу було створено MTProxy.

Шифрування повідомлень

На рис. 1.2 зображено зміст пакета для шифрування.

Length	Header	Random bits	Layer	seq_in	seq_out	Header	Random id	TTL	Message	Header	Padding
32 bit	32 bit	128 bit	32 bit	32 bit	32 bit	32 bit	64 bit	32 bit	Произвольная длина	32 bit	0-96 bit
Блок 1		Блок 2			Блок 3			Блок 4	Блоки	Блок N	

Рисунок 1.2 - Зміст пакета для шифрування [11]

- Length – довжина пакета без урахування padding.
- Header – кожен пакет складається з трьох заголовків, що містять інформацію про версію протоколу, тип прикладених медіафайлів.
- Random bits – 120 біт, згенерованих клієнтом, і 8 біт визначають довжину поля в байтах. Використовується як Salt для шифрування.
- Layer – позначає версію протоколу.
- seq_in – кількість надісланих повідомлень до творця чату.
- seq_out – кількість надісланих повідомлень від творця чату.
- Random id – довільне число, згенероване клієнтом, що відправляє, відправляється як текст.
- TTL – кількість секунд, протягом яких одержувач зможе бачити повідомлення перед його видаленням.

- Message – повідомлення, введене користувачем (довільної довжини).
- Padding – додається безпосередньо перед шифруванням.

На рис. 1.3 зображена схема шифрування:

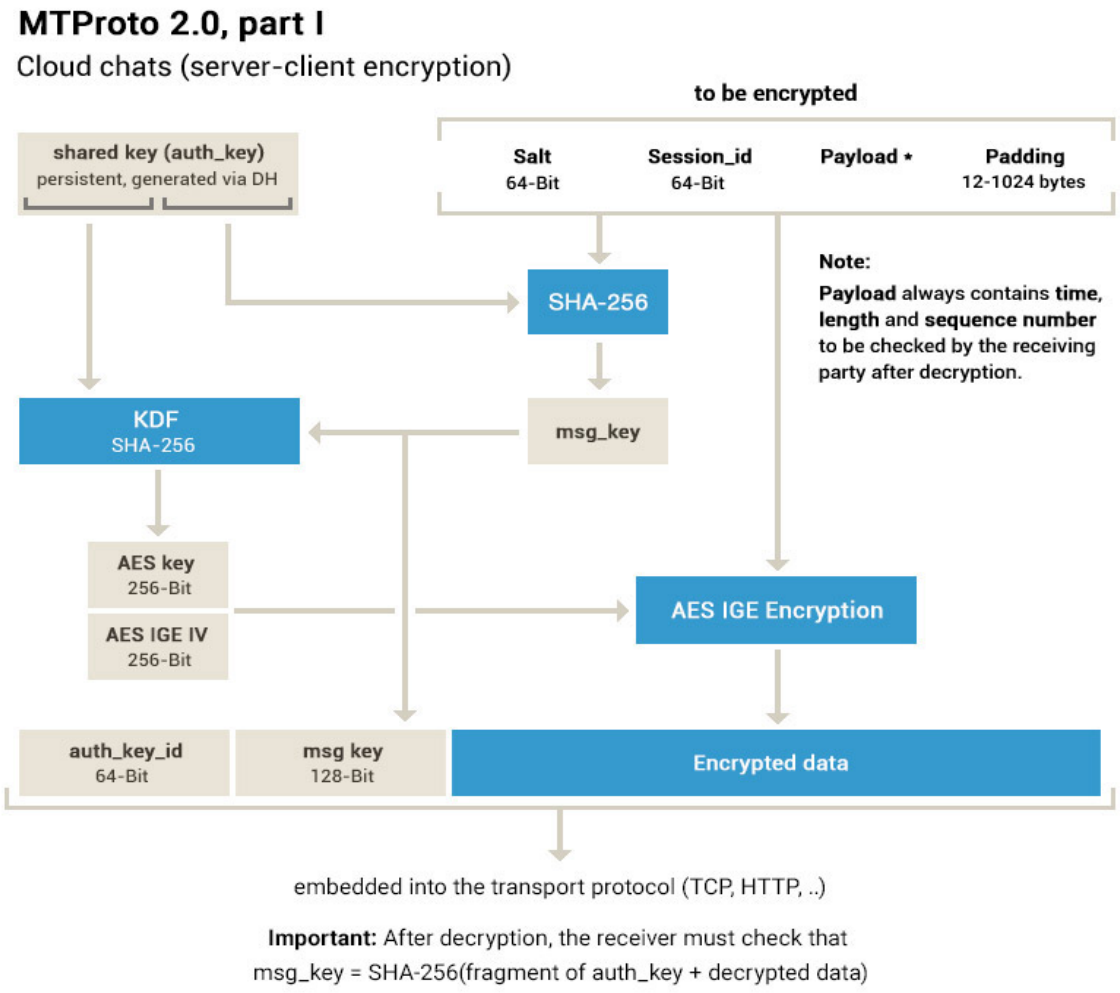

Рисунок 1.3 - Схема шифрування [14]

- auth_key – відкритий ключ шифрування, отриманий під час створення чату.

- Payload – пакет для шифрування.
- msg_key – молодші 128 біт SHA1-хеша пакету, що шифрується. Використовується для перевірки коректності під час розшифрування.
- Padding – 12-1024 біт. Додавання даних до інформації, націлене на підвищення криптостійкості.
- AES key та IV – параметри для шифрування алгоритмом AES у режимі IGE. Отримано за допомогою KDF.
- KDF (key derivation function) – функція формування AES key та IV на основі msg_key та auth_key.
- auth_key_id – молодші 64 біт SHA1-хеша відкритого ключа. У разі колії відкритий ключ буде згенеровано знову.
- Session_id – випадкове 64-бітове число, що генерується клієнтом з метою розрізнити окремі сеанси одного користувача.
- Server salt – випадкове 64-бітове число, що, окремо для кожної сесії, змінюється кожні 30 хвилин на запит сервера.

1.5 Можливості та функції Telegram

Коли я тільки дізнався про Telegram, то подумав що це простий месенджер. Так, ним приємно користуватися, чудовий дизайн, достатньо швидкий. Але з кожним оновленням мені подобався цей додаток все більше і більше. Подивимося, як я вважаю, на головні функції Telegram:

Обмін файлами

Підтримується обмін фотографіями, відеозаписами та файлами будь-якого типу.

Групові чати

Станом на 2018 рік є можливість організовувати чати до 100 тисяч учасників.

Коментарі

У каналах, прив’язаних до групи, під кожною публікацією є кнопка для зображення гілки відповідей на її автоматичну копію в цій групі – коментарів до публікації.

Дзвінки

Спочатку можливість зателефонувати була доступною для користувачів із західної Європи. Вона з’явилася і в тих, хто отримав виклик. Пізніше ця можливість стала доступною всім охочим. Виклики захищені шифруванням. Якість голосового виклику автоматично налаштовується залежно від якості зв’язку користувача. У 2020 році було додано відеодзвінки.

Архів чатів

Є можливість приховувати (архівувати) будь-які чати в розділі з назвою «Archived Chats». Користувач може розархівувати чат, повернувши його до основного списку чатів.

Збережене

Чат користувача із самим собою, призначений для зберігання користувачем окремих повідомлень. Дуже зручно коли необхідно зберегти якесь посилання або файл.

Боти

Це моя улюблена тема, коли йдеться про Telegram. У Telegram працює платформа чатботів. Боти можуть виконувати різноманітні завдання, такі як пошук в інтернеті чи держреєстрах, покупки, платежі, розваги, модерація груп тощо. Користувач може взаємодіяти з ботом за допомогою елементів інтерфейсу месенджера: надсилання повідомлень, натискання на команди та кнопки, використання інлайн-режиму. Telegram надає три способи взаємодії користувача з ботом:

- найпоширеніший спосіб – приватний чат;
- деякі боти можуть бути учасниками груп – у групах бот може модерувати повідомлення, бути ведучим гри, надсилати прогноз погоди та інше;
- інлайн-режим –користувач записує у поле для введення повідомлень запит, що починається з короткого імені бота, і далі користувач може вибрати й надіслати один з результатів.

Насправді функціонал бота обмежується лише фантазією програміста. Існують багато різновидів ботів: для спілкування с бізнесом, для слідкування за курсом валют, для отримання повідомлень про якийсь цікавий івент, для конвертації валют (рис. 1.4), для показу прогнозу погоди (рис. 1.5) та інше.

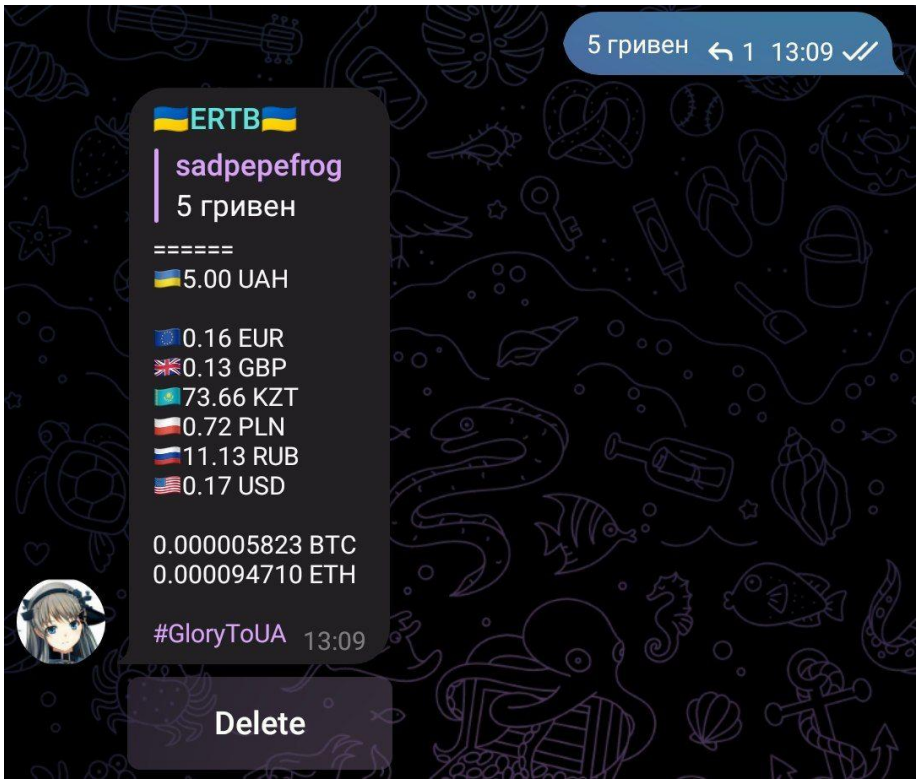

Рисунок 1.4 - Бот для конвертації валюти

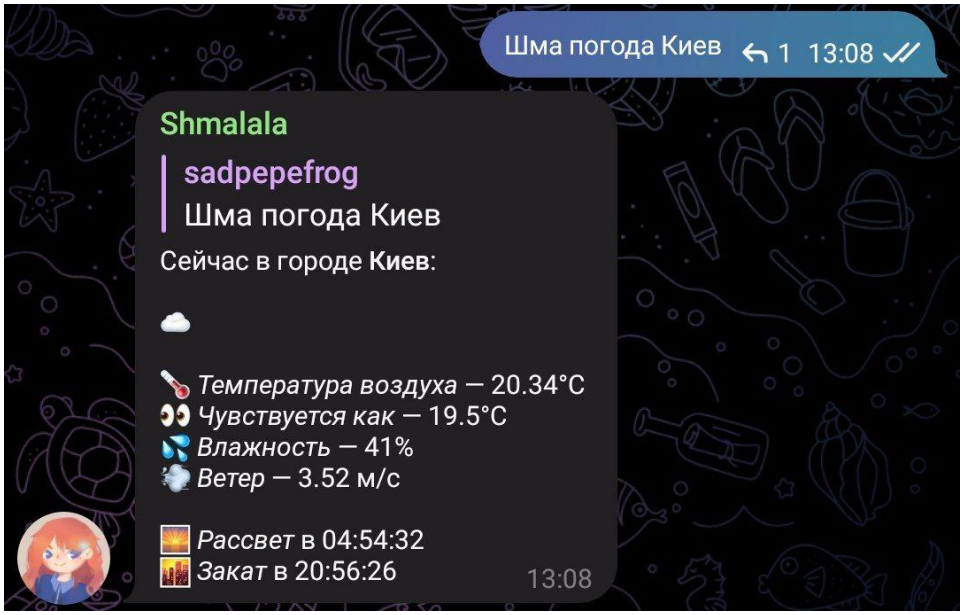

Рисунок 1.5 - Бот для відображення погоди у місті на даний момент

1.6 Вебскрапінг, парсинг та етапи парсингу

Вебскрапінг – це процес автоматичного збору інформації з Інтернету. Це трудомістка та рутинна робота, що забирає багато часу. Парсери здатні протягом доби перебрати більше частину веб-ресурсів у пошуках відповідної інформації.

Парсинг – етап скрапінгу, який «витягує» необхідну інформацію із завантажених даних. Парсинг буває різного виду:

- пошук та наповнення ресурсів текстовим та мультимедійним контентом;
- товари та ціни в інтернет-магазаних;
- дані з оголошень, розміщених на спеціальних ресурсах;
- пошук та збирання контактних даних користувачів;
- у рамках соціальних мереж (відгуки або коментарі);
- сайти, що спеціалізуються на публікації спортивних результатів.

Етапи парсингу:

1. Пошук даних – у програму-парсер завантажується вихідний HTML-код сторінки сайту. З цим кодом починаю працювати скрипт, який розбиває текст та виділяє необхідну інформацію.
2. Вилучення інформації – пошук даних відбувається завдяки набору знаків, що описують мету пошуку.
3. Збереження даних – після отримання інформація зберігається у вигляді таблиць або бази даних.

1.7 Синхронне програмування

Традиційно в програмуванні використовується синхронне програмування – інструкції виконуються послідовно за допомогою синхронних системних викликів, які повністю блокують потік виконання до завершення системної операції (наприклад, читання з диска). У високонавантажених системах найчастіше відбувається наступне – майже весь час програма чогось чекає: диска, базу даних, мережі, загалом якоїсь зовнішньої, незалежної від самої програми події. У малонавантажених системах можна вирішити створенням нового потоку для кожної блокуючої дії. Поки один потік спить, інший працює [5].

Але виникає питання – що робити, коли користувачів дуже багато? При створенні на кожного хоча б один потік – продуктивність такого сервера стрімко впаде через те, що контекст виконання потоку постійно змінюється. Кожен потік створює свій власний контекст виконання, включаючи пам’ять для стека, що має мінімальний розмір 4 КБ. Асинхронне програмування може вирішити цю проблему.

1.8 Асинхронне програмування

Асинхронне програмування – це концепція програмування, коли результат виконання функції доступний через деякий час у вигляді асинхронного (тобто порядок виконання йде не по порядку) виклику. Запуск тривалих операцій відбувається без очікування їх завершення та не блокує подальшого виконання програми [2] [4]. На рисунку 1.6 показано різницю між синхронним та асинхронними виконанням програми:

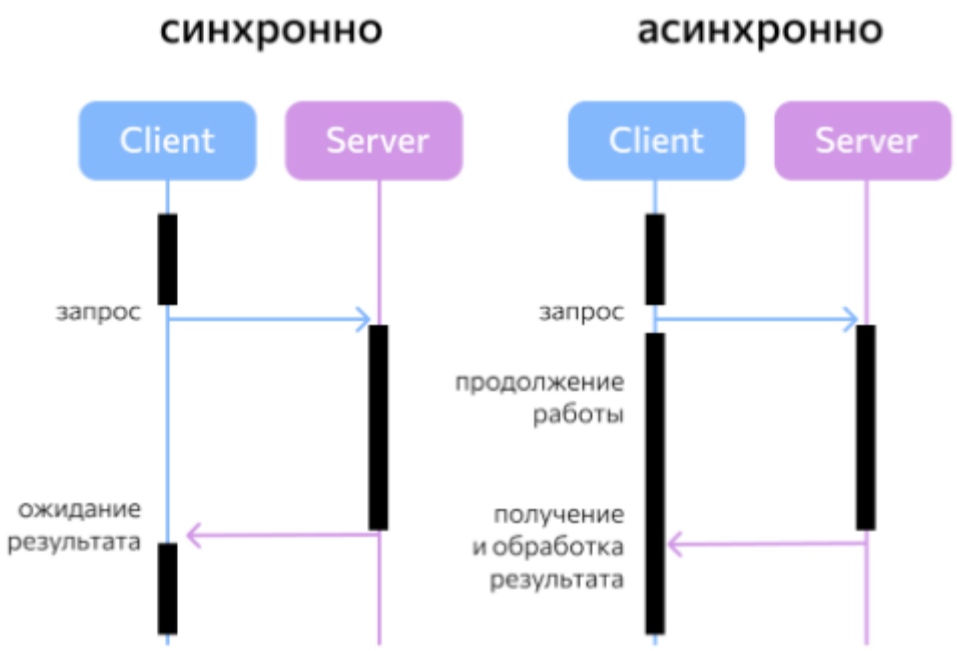

Рисунок 1.6 - Різниця між синхронним та асинхронним виконанням

Такий спосіб програмування став популярним завдяки масштабним додаткам. Використання принципів асинхронного програмування підвищує продуктивність та стабільність додатків, які відповідають наступним характеристикам [3]:

- безперервно виконують велику кількість задач;
- задачі виконують безліч операцій із введення та виведення інформації, змушуючи синхронну програму марнувати час на блокування;
- задачі переважно незалежні друг від друга, і необхідність обміну даними між операціями практично відсутня.

1.9 Постановка задачі

Месенджери зараз використовують майже всі, особливо Telegram. Оскільки інтернет-магазин може бути перевантаженим різними анімаціями та функціями, то чудовою альтернативою може стати бот, за допомогою якого з’явиться можливість зробити замовлення у кілька кліків. Тому у разі успішної реалізації буде можливим обирати товар та одразу переходити сторінку для його замовлення в інтернет магазині.

На основі проаналізованої інформації можна сформулювати етапи виконання практичної роботи:

1. Проаналізувати якою технікою люди частіше користуються у повсякденному житті.
2. Зібрати інформацію про цю техніку з інтернет-магазину.
3. На основі зібраної інформації створити базу даних.
4. За допомогою Telegram Bot API створити бота, через якого користувач зможе одразу перейти на сторінку обраного товару в інтернет-магазині.

2. МЕТОДИКА ВИРІШЕННЯ ПОСТАВЛЕНИХ ЗАДАЧ

2.1 Програмування мовою Python

Python – одна з найпопулярніших мов програмування. Це інтерпретована об’єктно-орієнтована мова програмування високого рівня з динамічною типізацією.

Популярність мови обумовлена її відносною простотою – працювати х нею може швидко розпочати навіть новачок. Звісно що він не зможе одразу писати високонавантажені проекти, але знайти рішення для завдання базового рівня цілком можливо [7].

Як і будь-яка мова, Python має переваги та недоліки [6]:

1. Переваги
   - Гнучкість

     На мою думку, це одна з двох основних переваг мови, оскільки завдяки своїй гнучкості мова набула популярності серед багатьох розробників.

   - Розширюваність

     Це друга основна перевага мови. Один із слоганів Python звучить як «Just Import!», що повністю пояснює, наскільки мову було розширено за останні роки. Існують бібліотеки та фреймворки під будь-який тип завдань та потреб

   - Простота синтаксису

Синтаксис – це саме те, через що багатьом людям подобається Python. І це не дивно – із синтаксису було прибрано все «зайве», код виглядає чистим та зрозумілим без зайвих дужок та виразів.

- Інтерпретованість

  Інтерпретатор Python існує для всіх популярних платформ і за замовчуванням входить до більшості дистрибутивів Linux.

- PEP

  PEP – це єдиний стандарт для написання коду, що робить код підтримуваним і читабельним навіть при переході від одного розробника до іншого.

- Open Source

  Код інтерпретатора Python є відкритим, що дозволяє будь-кому, хто зацікавлений у розвитку мови взяти участь у розробці та покращити її.

2. Недоліки

   - Продуктивність

     Більшість розробників сходяться на думці, що Python не настільки спритний, наскільки багатьом хотілося б. Це пов’язано з тим, що Python інтерпретується. Але навіть у порівнянні з іншими мовами, що інтерпретуються, Python помітно програє у продуктивності.

- Динамічна типізація

  Через динамічну типізацію Python споживає більше ресурсів, ніж міг би.

- Global Interpreter Lock або GIL (рис. 2.1)

  В Python використовується глобальне блокування інтерпретатора, що накладає деякі обмеження на потоки. Тобто не можна використовувати кілька процесів одночасно. На даний момент це є основною проблемою продуктивності, а також цим обумовлена погана реалізація багатопоточності. Код GIL не змінювався з першої версії мови [9].

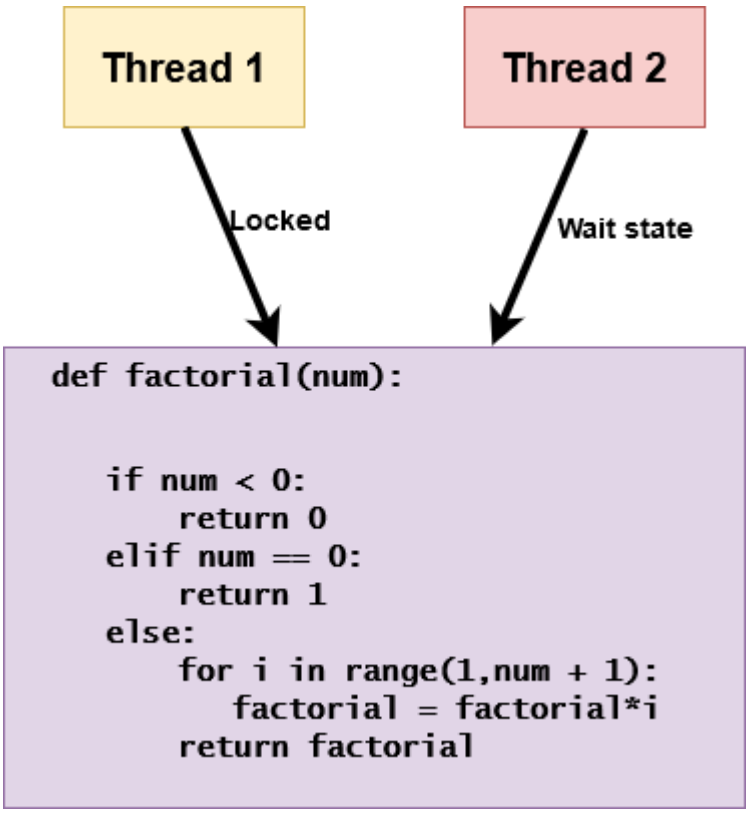

Рисунок 2.1 - GIL Python

2.2 Асинхронне програмування в Python

Спочатку в Python для вирішення задач асинхронного програмування використовувалися корутини, що базуються на генераторах (конструкція yield). Потім у Python 3.4 з'явився модуль asyncio, в якому реалізовані механізми асинхронного програмування. У Python 3.5 з'явилася конструкція async/await. Корутини та задачі – невід'ємна частина асинхронної розробки в Python.

Зазвичай корутина – це асинхронна (async) функція. Вона може бути об'єктом, повернутим з корутини-функції. Якщо при оголошенні функції хахначено, що вона є асинхронною, то викликати її можна з використанням ключового слова await (рис. 2.2) [8].

```
await say_something(to_say)
```

Рисунок 2.2 - Виклик асинхронної функції за допомогою ключового слова await

Така конструкція означає, що програма буде виконуватися доти, доки не зустріне await-вираз, після чого викличе функцію та призупинить своє виконання до тих пір, коли робота викликаної функції не завершиться. Після цього запуститися з'явиться і в іншим корутинах.

Призупинення виконання програми означає, що керування повертається циклу подій. Коли використовують модуль asyncio, цикл подій виконує

асинхронні задачі, виконує операції введення та виведення, а також виконує підпроцеси.

У свою чергу, задачі використовуються для запуску корутин. Вони дозволяють запускати корутини у циклі подій. Це спрощує управління виконанням кількох корутин. Розглянемо приклад з офіційної документації Python (рис. 2.3).

```
import asyncio
import time

async def say_after(delay, what):
    await asyncio.sleep(delay)
    print(what)

async def main():
    print(f"started at {time.strftime('%X')}")

    await say_after(1, 'hello')
    await say_after(2, 'world')

    print(f"finished at {time.strftime('%X')}")

asyncio.run(main())
```

Рисунок 2.3 - Приклад асинхронного коду на мові Python [10]

Даний приклад демонструє конкурентний запуск двох корутин. Для кожної з них використовуються задачі. В результаті час виконання всієї програми становить близько двох секунд (рис. 2.4). Якщо ж запускати їх послідовно, то час виконання буде відрізнятися на секунду, тобто становитиме близько трьох секунд (рис. 2.4).

```
started at 17:13:52
hello
world
finished at 17:13:55
```

Рисунок 2.4 - Результат виконання асинхронного коду [10]

2.3 Збір інформації за допомогою асинхронного програмування

За основу для збору інформації про техніку було взято один із найпопулярніших інтернет-магазинів України – ROZETKA. Цей магазин дуже чудовий, адже продається не тільки техніка, а й багато іншого: одяг, напої, аксесуари, запчастини і т. д.

Було вирішено брати такі товари (всіх брендів): телефони, телевізори, годинники, планшети, ноутбуки, монітори, комп'ютери (вже готові збірки) та навушники. Тобто все те, чим всі люди активно користуються у повсякденному житті.

Всі дані зібрані в словник – тип даних в Python, який має вигляд {key: value}, де key – унікальний ключ, який не може повторюватися більше одного разу, а value – будь-яке значення будь-якого типу даних (число, масив та ін.). Ключем в словнику буде бренд товару, його значенням - ще одним словник, в якому ключ – модель продукту, а значення – масив з ціни моделі та посилання на нього. Тобто словник матиме такий вигляд: {бренд: {модель продукту: [ціна, посилання]}}.

На рисунку 2.5 показано метод (деякі говорять функція) для парсингу однієї одиниці товару.

```
def get_product_info(soup, products, producer):
    if producer in ('lg', 'jbl', 'asus', 'hp', 'msi', 'artline', 'aoc'):
        producer = producer.upper()

    elif producer == 'benq':
        producer = 'BenQ'

    else:
        producer = producer.capitalize()

    if producer not in products:
        products[producer] = {}

    is_stored = soup.find_all('div', class_='goods-tile__availability goods-tile__availability--available ng-star-inserted')
    if is_stored:
        link = soup.find_all('a', class_='goods-tile__heading ng-star-inserted')
        model = soup.find_all('span', class_='goods-tile__title')
        price = soup.find_all('span', class_='goods-tile__price-value')

        min_len = min(len(model), len(price), len(is_stored), len(link))

        for idx in range(min_len):
            l = link[idx]['href']
            m = model[idx].text.strip()
            p = int(price[idx].text.strip().replace(u'\xa0', u''))
            s = is_stored[idx].text.strip()

            if producer in m and m not in products[producer] and s in ('Есть в наличии', 'Готов к отправке'):
                products[producer][m] = [p, l]
```

Рисунок 2.5 - Метод для парсингу однієї одиниці товару

Відбувається тут наступне:

1. Із сторінки збираємо інформацію про наявність продукту, посилання на нього, ціну та модель. Якщо ж товару немає в наявності, то він не враховується.
2. Створюється словник, в якому ім’я бренду – унікальний ключ, його значення – ще один словник, в якому унікальним ключем є модель продукту цього бренду, а значенням цього ключа – масив з ціни та посилання на нього.

Для кожної категорії товару був створений окремий модуль, в якому є два методи: перший – для збору інформації всіх товарів на першій сторінці, і другий – на всіх наступних сторінках (рис. 2.6).

```python
import asyncio
import aiohttp
from bs4 import BeautifulSoup as bs
from common_methods import get_product_info

async def get_first_phones(session, phones, producer):
    url = f'https://rozetka.com.ua/mobile-phones/c80003/producer={producer}/'

    async with session.get(url) as res:
        html = await res.text()
        soup = bs(html, 'lxml')

        get_product_info(soup, phones, producer)

async def get_other_phones(page, session, phones, producer):
    url = f'https://rozetka.com.ua/mobile-phones/c80003/page={page};producer={producer}/'

    async with session.get(url) as res:
        html = await res.text()
        soup = bs(html, 'lxml')

        get_product_info(soup, phones, producer)
```

Рисунок 2.6 - Два методи для збору інформації про товари

Функція get_first_phones – асинхронно збирає інформацію про телефони на першій сторінці, а функція get_other_phones – на всіх інших сторінках.

3. МЕТОДИКА ВИРІШЕННЯ ПОСТАВЛЕНИХ ЗАДАЧ

3.1 Розробка бази даних

Варто зазначити, що у цій роботі база даних використовується не зовсім за призначенням. Тут вона більше як простий текстовий файл, ніж як стандартні бази даних з рівнями доступу для користувачів і таблицями із залежностями. Особисто для мене це було зручніше, адже при записі даних у файл не було б меж, де і яка категорія продукту.

Для проектування бази даних була використана стандартна бібліотека sqlite3 в Python, так як вона дуже проста, зрозуміла та зручна у використанні [15]. Всього було створено сімнадцять таблиць:

- categories – первинним ключем є номер категорії за рахунком у списку (рис. 3.1);

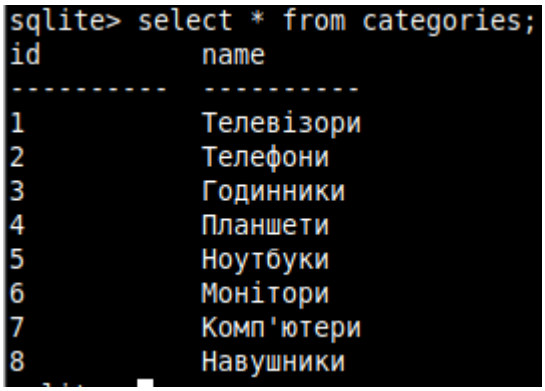

Рисунок 3.1 - Таблиця categories

- вісім таблиць з брендами для кожної категорії товару (наприклад, tv_brands – таблиця з брендами для телевізорів

(рис. 3.2)), в яких первинним ключем є номер бренду за рахунком у списку;

```
sqlite> select * from tv_brands;
id          brand_name
----------  ----------
1           LG
2           Philips
3           Samsung
4           Sony
5           Xiaomi
```

Рисунок 3.2 - Таблиця tv_brands

- вісім таблиць з моделями для кожного бренду товару (наприклад, tv_models – таблиця з моделями для кожного бренда (рис. 3.3)), в яких первинним ключем є номер моделі за рахунком у списку.

```
sqlite> select * from tv_models;
id          model                  price       link                                                brand
----------  ---------------------  ----------  --------------------------------------------------  ----------
1           Телевизор LG 43UP75006LF  14699       https://rozetka.com.ua/lg_43up75006lf/p292223868/  LG
2           Телевизор LG 50UP75006LF  16999       https://rozetka.com.ua/lg_50up75006lf/p292227863/  LG
3           Телевизор LG 43UP81006LA  18999       https://rozetka.com.ua/lg_43up81006la/p292230848/  LG
4           Телевизор LG 55UP75006LF  20499       https://rozetka.com.ua/lg_55up75006lf/p292227868/  LG
5           Телевизор LG OLED65C14LB  84999       https://rozetka.com.ua/lg_oled65c14lb/p292190978/  LG
6           Телевизор LG 43NANO756PA  20499       https://rozetka.com.ua/lg_43nano756pa/p292219168/  LG
7           Телевизор LG 55NANO816PA  29499       https://rozetka.com.ua/lg_55nano816pa/p292219358/  LG
8           Телевизор LG OLED55A16LA  47999       https://rozetka.com.ua/lg_oled55a16la/p292190813/  LG
9           Телевизор LG 50UP81006LA  21499       https://rozetka.com.ua/lg_50up81006la/p292231078/  LG
10          Телевизор LG 55NANO776PA  27499       https://rozetka.com.ua/lg_55nano776pa/p292219348/  LG
11          Телевизор LG 50NANO776PA  25499       https://rozetka.com.ua/lg_50nano776pa/p292219343/  LG
12          Телевизор LG 50NANO866PA  30499       https://rozetka.com.ua/lg_50nano866pa/p292219368/  LG
13          Телевизор LG 55NANO916PA  40999       https://rozetka.com.ua/lg_55nano916pa/p292220498/  LG
14          Телевизор LG OLED55G16LA  68999       https://rozetka.com.ua/lg_oled55g16la/p292196463/  LG
15          Телевизор LG OLED55C14LB  58999       https://rozetka.com.ua/lg_oled55c14lb/p292190973/  LG
16          Телевизор LG 55UP78006LB  23499       https://rozetka.com.ua/lg_55up78006lb/p292230118/  LG
17          Телевизор LG OLED65A16LA  68999       https://rozetka.com.ua/lg_oled65a16la/p292190818/  LG
18          Телевизор LG OLED48A16LA  40499       https://rozetka.com.ua/lg_oled48a16la/p292183903/  LG
19          Телевизор LG 75NANO916PA  59999       https://rozetka.com.ua/lg_75nano916pa/p292221658/  LG
20          Телевизор LG QNED MiniLE  82999       https://rozetka.com.ua/lg_65qned916pa/p315956098/  LG
21          Телевизор Philips 55PUS7  18999       https://rozetka.com.ua/philips_55pus7906_12/p3069  Philips
22          Телевизор Philips 55PUS7  15999       https://rozetka.com.ua/philips_55pus7506_12/p3149  Philips
23          Телевизор Philips 58PUS8  26999       https://rozetka.com.ua/philips_58pus8506_14/p3069  Philips
24          Телевизор Philips 48OLED  40999       https://rozetka.com.ua/philips_48oled806_12/p3266  Philips
```

Рисунок 3.3 - Таблиця tv_models

3.2 Розробка бота в додатку Telegram

Для початку створимо бота (рис. 3.4). Створення та подальші налаштування бота відбуваються в програмі Telegram. Для цього потрібно в пошуку знайти канал з назвою @BotFather, вписати команду «/newbot», відправити в чат і дотримуватися вказівок. Після успішного створення нового бота, в чат буде відправлено посилання на нього та його токен. Дуже важливо зберігати цей токен у безпеці, тому що за його допомогою здійснюється контроль бота.

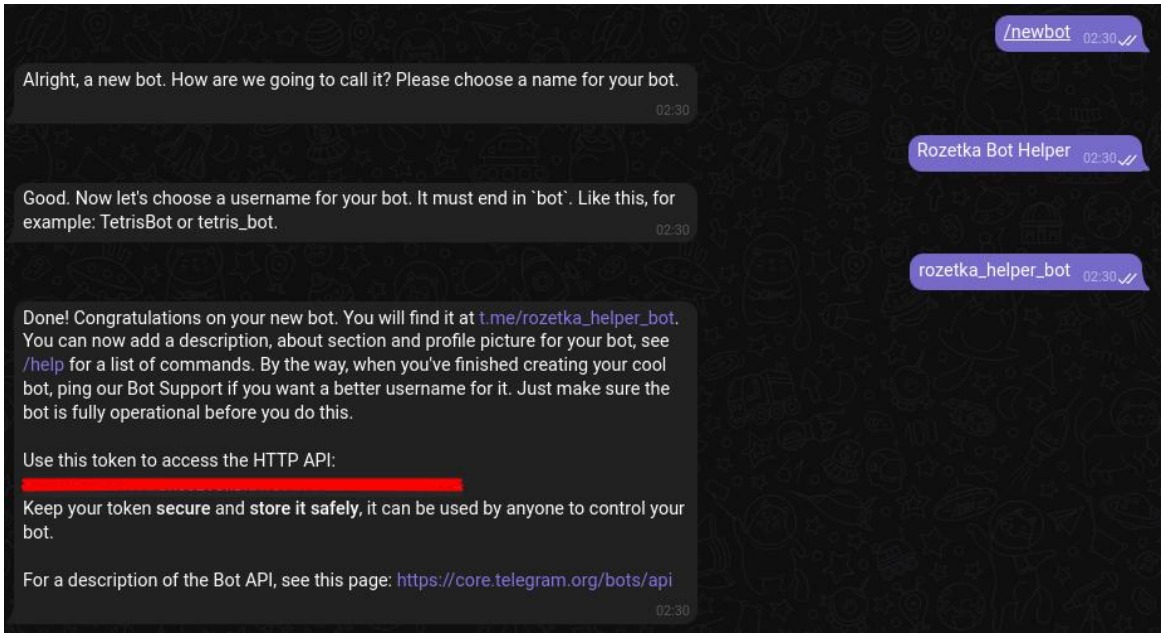

Рисунок 3.4 - Створення бота з назвою Rozetka Bot Helper

Бот повинен працювати так, як і інтернет-магазин: можна вибрати категорію, бренд, модель, перейти на сторінку товару, перейти на наступну сторінку. У цій реалізації вибір категорії, бренду або сторінки здійснюється за допомогою кнопок, які з’являтимуться користувачу в міру самого вибору.

На одній сторінці може відображатися до десяти товарів. Для відображення інших товарів було створено обробник сторінок. При натисканні на сторінку з номером 2 з’являються наступні десять товарів, при натисканні на сторінку з номером 3 – наступні десять товарів і т. д. Поточна сторінка виділена двома крапками з обох боків (рис. 3.5). Якщо ж товару якогось з брендів немає, то бот відправить про це повідомлення.

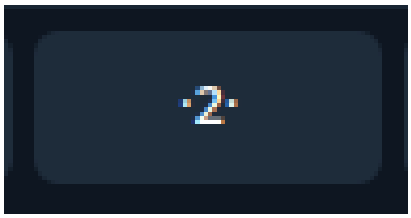

Рисунок 3.5 - Відображення поточної сторінки

Якщо якась із моделей товару сподобалася користувачу, то потрібно боту в чат відправити номер, під яким стоїть ця модель у списку, після чого бот відправить повідомлення з інформацією про ціну та кнопкою-посиланням, при натисканні на яку користувач переходить на сторінку цієї моделі в інтернет-магазині ROZETKA.

3.3 Тестування Telegram бота Rozetka Bot Helper

Починаємо роботу з ботом, для цього тиснемо на кнопку START. З’являється інформаційне повідомлення з невеликим описом, що це за бот (рис. 3.6).

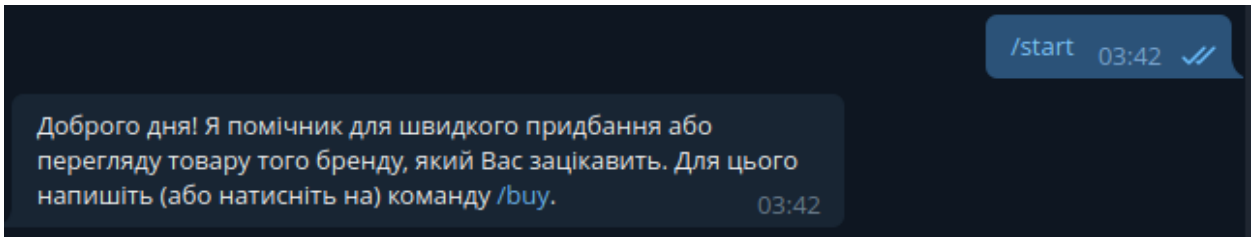

Рисунок 3.6- Початок роботи з Rozetka Bot Helper

Вписуємо команду «/buy» та відправляємо боту, після чого надходить повідомлення з вибором категорії (рис. 3.7).

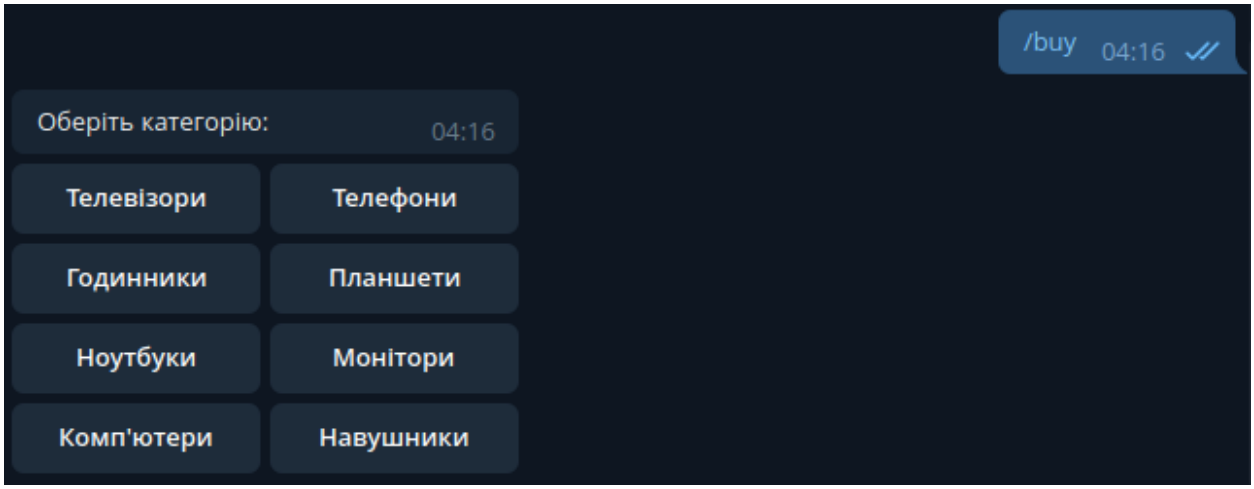

Рисунок 3.7 - Вибір категорії

Після вибору категорії, наприклад «Планшети», з’являється можливість обрати бренд (рис. 3.8):

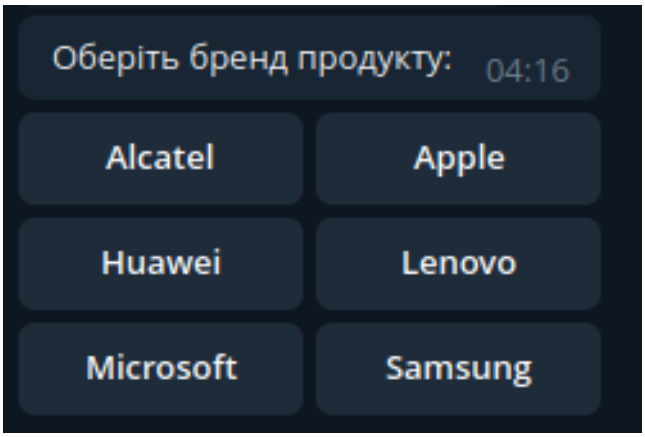

Рисунок 3.8 - Вибір бренду товару

- обираємо Apple – результатом бачимо повідомлення про те, що товарі цього бренду немає в наявності (рис. 3.9):

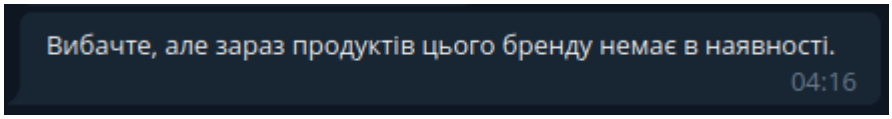

Рисунок 3.9 - Повідомлення про те, що планшетів від Apple немає в наявності

- обираємо Huawei – результатом бачимо список із декількох моделей (рис. 3.10);

Рисунок 3.10 - Список моделей планшетів від компанії Huawei

- обираємо Microsoft – результатом бачимо список із десяти моделей та сторінки, а отже, моделей ще більше (рис. 3.11).

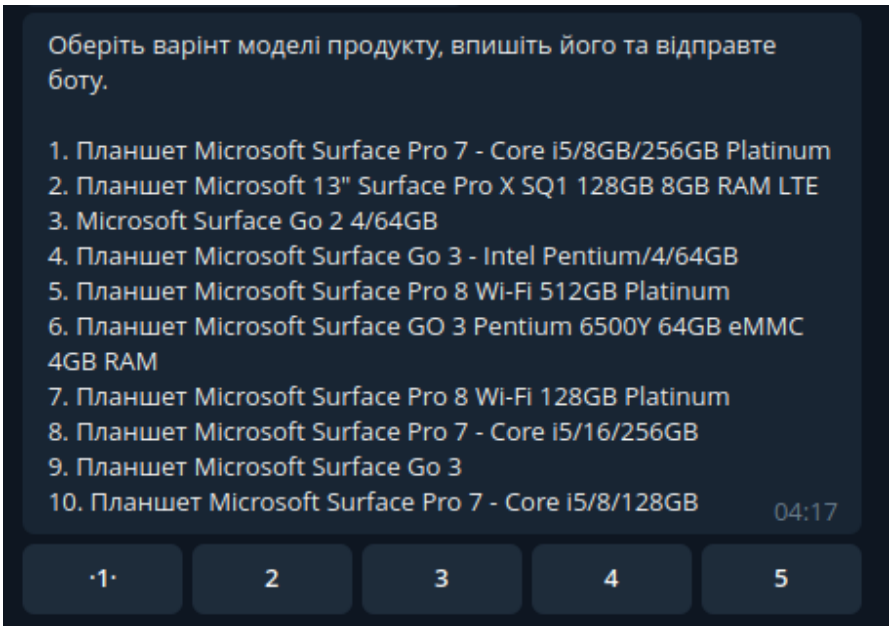

Рисунок 3.11 - Перші десять моделей планшетів від компанії Microsoft

Натискаємо кнопку 2 і переходимо на другу сторінку (рис. 3.12).

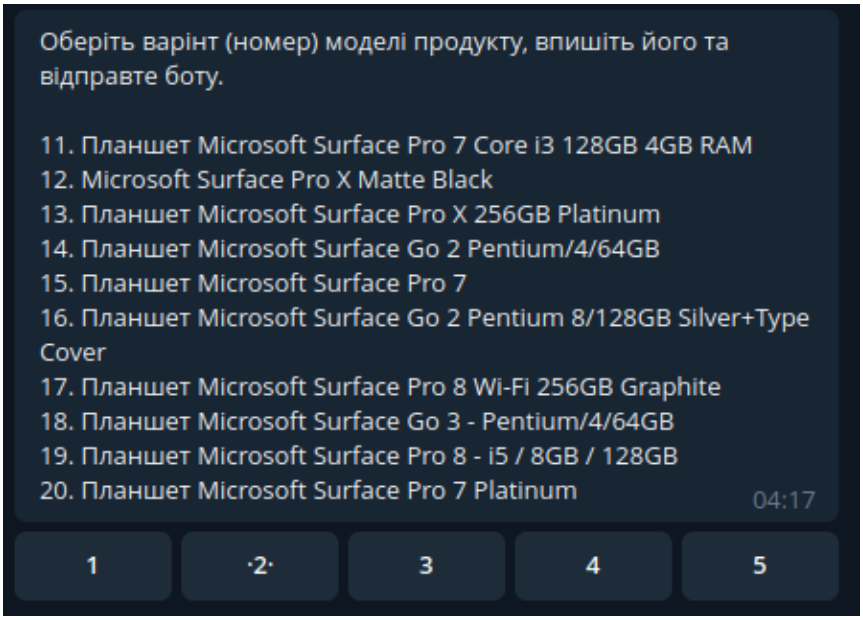

Рисунок 3.12 - Друга сторінка з моделями планшетів від компанії Microsoft

Наприклад, користувачу сподобався варіант під номером 14. Для отримання посилання на цю модель в інтернет-магазині необхідно вписати цифру 14 і відправити боту, в результаті чого бот надсилає інформаційне повідомлення про ціну обраної моделі та кнопку-посилання (рис. 3.13), при натисканні на яку з’являється вікно з вибором – відкрити сторінку, чи скасувати (рис. 3.14). Натискаємо «Open» та потрапляємо на сторінку інтернет-магазину з вибраним варіантом моделі (рис. 3.15).

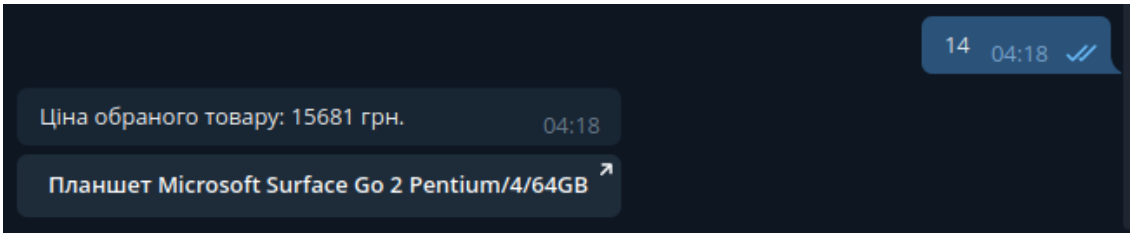

Рисунок 3.13 - Обираємо варіант моделі

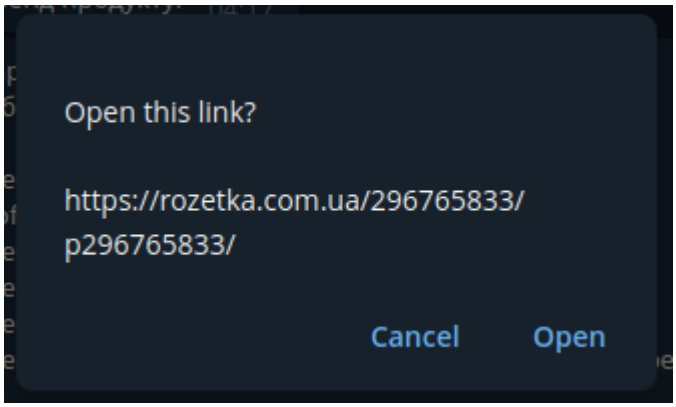

Рисунок 3.14 - Вікно з відкриттям посилання

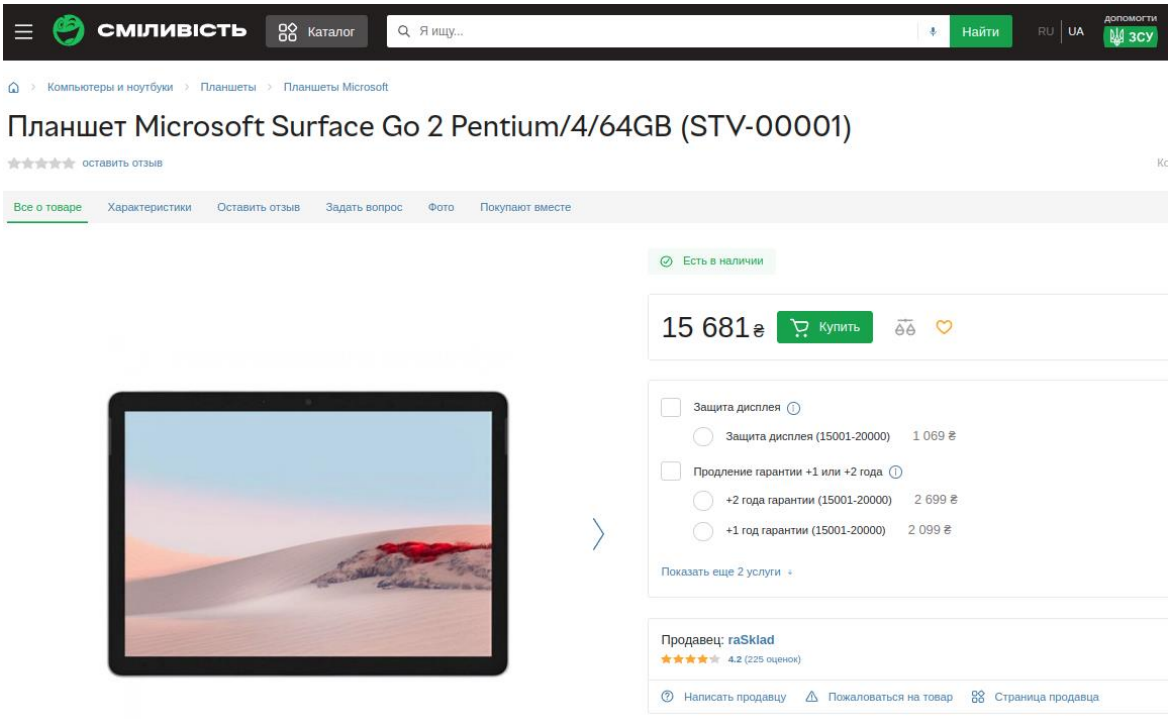

Рисунок 3.15 - Відкрита сторінка з моделлю, яку обрав користувач

Завдяки асинхронному програмуванню інформація про товари була зібрана набагато швидше, ніж якби був обраний варіант синхронного виконання роботи коду.

За допомогою Telegram Bot API був розроблений чат-бот, який працює так само, як і інтернет-магазин - користувач обирає категорію, бренд товару та модель.

ВИСНОВКИ

При виконанні кваліфікаційної роботи бакалавра був розглянутий додаток Telegram, а саме його функції та можливості. Telegram – чудовий месенджер для смартфонів або ПК, який дозволяє підтримувати зв’язок на великій відстані повідомленнями різних типів (текстові, голосові та відеоповідомлення) та дзвінками. Більша частина уваги була звернена на можливість створювати чат-боти, їх функції та застосування у повсякденному житті.

До найбільш популярних мов програмування належить мова Python. Вона популярна за свій великий список бібліотек, як стандартних, так і написаних розробниками з усього світу, та синтаксис. Навіть початківець зможе зрозуміти що написано в коді, адже сам код програми виглядає як «псевдокод». Також було досліджено концепцію асинхронного програмування та її різницю із синхронним програмуванням. Саме завдяки асинхронному програмуванню виконання програми може значно покращитись у часі.

Було сформовано вимоги до чат-бота, який має:

1. бути швидким і зручним у користуванні;
2. ефективно систематизувати замовлення товарів.

В рамках роботи був розроблений чат-бот Rozetka Bot Helper, за допомогою якого користувач може швидко обрати модель товару та перейти на його сторінку в інтернет-магазині для замовлення або огляду детальної характеристики.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Мессенджер: что такое мессенджер, обзор популярных, преимущества и перспективы [Електронний ресурс] – Режим доступу до ресурсу: https://trends.rbc.ru/trends/social/617a68a89a79476935d1f857

2. Что такое синхронный и асинхронный код? [Електронний ресурс] – Режим доступу до ресурсу: https://qna.habr.com/q/626141

3. Асинхронное программирование: концепция Deferred [Електронний ресурс] – Режим доступу до ресурсу: https://blog.skillfactory.ru/glossary/asinhronnoe

4. Асинхронное программирование: концепция, реализация, примеры [Електронний ресурс] – Режим доступу до ресурсу: https://proglib.io/p/asynchrony

5. Асинхронность в програмировании [Електронний ресурс] – Режим доступу до ресурсу: https://tproger.ru/articles/asynchronous-programming/

6. Python Programming Language [Електронний ресурс] – Режим доступу до ресурсу: https://www.geeksforgeeks.org/python-programming-language/

7. Head First Python. A Brain Friendly Guide by Paul Barry [Електронний ресурс] – Режим доступу до ресурсу: https://www.pdfdrive.com/head-first-python-a-brain-friendly-guide-e183836129.html

8. Асинхронное програмирование в Python: краткий обзор [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/company/ruvds/blog/475246/

9. Как устроен GIL в Python [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/post/84629/

10. Official Python Documentation: Coroutines and Tasks [Електронний ресурс] – Режим доступу до ресурсу: https://docs.python.org/3.7/library/asyncio-task.html

11. MTProto [Електронний ресурс] – Режим доступу: https://ru.wikipedia.org/wiki/MTProto

12. The New York Times: Once Celebrated in Russia, the Programmer Pavel Durov Chooses Exile [Електронний ресурс] – Режим доступу: https://www.nytimes.com/2014/12/03/technology/once-celebrated-in-russia-programmer-pavel-durov-chooses-exile.html?_r=0

13. TechCrunch: Meet Telegram, A Secure Messaging App From The Founders Of VK, Russia’s Largest Social Network [Електронний ресурс] – Режим доступу: https://techcrunch.com/2013/10/27/meet-telegram-a-secure-messaging-app-from-the-founders-of-vk-russias-largest-social-network/

14. Что там по MTProto в Telegram-то? [Електронний ресурс] – Режим доступу: https://habr.com/ru/post/590667/

15. sqlite3 – DB-API 2.0 interface for SQLite databases [Електронний ресурс] – Режим доступу: https://docs.python.org/3/library/sqlite3.html

ДОДАТОК А

Програмний код чат-бота Rozetka Bot Helper:

```
import datab
import asyncio
import config
import telebot
import products
from loguru import logger
from telegram_bot_pagination import InlineKeyboardPaginator

counter = 0
main_table = {}
product, brand = None, None
database_path = './datab/dbs'
bot = telebot.TeleBot(config.TOKEN)

products_info = asyncio.run(products.collect_info.collect_products_info())
table_names = ('tv', 'phone', 'watch', 'tablet', 'laptop', 'display', 'computer', 'headphones')

db = datab.sql.Database()
connection, cursor = db.create_db('rozetka', './datab/dbs')
db.create_categories_table(cursor, connection)
db.create_brands_table(products_info, table_names, cursor, connection)
db.create_models_table(products_info, table_names, cursor, connection)
```

```
cursor.execute('SELECT * FROM categories;')
categories = cursor.fetchall()

for item in table_names:
cursor.execute(f'SELECT * FROM {item}_brands;')
table = cursor.fetchall()
cursor.execute(f'SELECT * FROM {item}_models;')
models = cursor.fetchall()
if item not in main_table:
        main_table[item] = {}
for t in table:
        _, b = t
        if b not in main_table[item]:
                main_table[item][b] = {}
        for m in models:
                _, model_name, price, link, _ = m
                if b.lower() in model_name.lower():
                        if model_name not in main_table[item][b]:
                                main_table[item][b][model_name] = []
                        main_table[item][b][model_name].append(price)
                        main_table[item][b][model_name].append(link)

@bot.message_handler(commands=['start', 'help'])
def send_welcome(msg):
```

```
bot.send_message(msg.chat.id, 'Доброго дня! Я помічник для швидкого придбання або перегляду товару того бренду, який Вас зацікавить. Для цього напишіть (або натисніть на) команду /buy.')

@bot.message_handler(commands=['buy'])
def get_categories(msg):
buttons = []
keyboard = telebot.types.InlineKeyboardMarkup()
keyboard.row_width = 2

for i, category_tuple in enumerate(categories, start=0):
    _, category_name = category_tuple
    key_category = telebot.types.InlineKeyboardButton(text=f'{category_name}', callback_data=table_names[i])
    buttons.append(key_category)

for idx in range(len(buttons) - int(len(buttons)/2)):
    if idx != 0:
        idx += idx
    keyboard.row(buttons[idx], buttons[idx + 1])
bot.send_message(msg.chat.id, text='Оберіть категорію:\n', reply_markup=keyboard)

@bot.callback_query_handler(func=lambda call: call.data in main_table)
def get_brands(call: telebot.types.CallbackQuery):
keyboard = telebot.types.InlineKeyboardMarkup()
```

```
    keyboard.row_width = 2

    btns = []
    b_table = main_table[call.data]
    for b in main_table[call.data]:
        key_brand = telebot.types.InlineKeyboardButton(text=f'{b}',
callback_data=f'{call.data}_{b}')
        btns.append(key_brand)

    if len(btns) % 2 == 0:
        for idx in range(len(btns) - int(len(btns) / 2)):
            if idx != 0:
                idx += idx
            keyboard.row(btns[idx], btns[idx + 1])
    else:
        for idx in range(len(btns) - 1 - int((len(btns) - 1) / 2)):
            if idx != 0:
                idx += idx
            keyboard.row(btns[idx], btns[idx + 1])
        keyboard.add(btns[-1])
    bot.send_message(call.message.chat.id, text='Оберіть бренд продукту:\n',
reply_markup=keyboard)
    bot.edit_message_reply_markup(call.message.chat.id,
call.message.message_id)

    @bot.callback_query_handler(func=lambda call: call.data.split('_')[0] in
main_table and call.data.split('_')[1] in main_table[call.data.split('_')[0]])
```

```
def get_models(call):
    global counter
    global product
    global brand
    text = 'Оберіть варінт моделі продукту, впишіть його та відправте боту.\n\n'

    product, brand = call.data.split('_')[0], call.data.split('_')[1]
    if len(main_table[product][brand]) == 0:
        bot.send_message(call.message.chat.id, text='Вибачте, але зараз продуктів цього бренду немає в наявності.\n')

    elif 0 < len(main_table[product][brand]) <= 10:
        for i, model in enumerate(main_table[product][brand], start=1):
            text += f'{i}. {model}\n'
        bot.send_message(call.message.chat.id, text=text)

    else:
        for k, key in enumerate(main_table[product][brand], start=1):
            if k > 10 and str(k)[-1] == '1':
                counter = k
                break
            text += f'{k}. {key}\n'
        send_models_page(call.message, product, brand, text)
    bot.edit_message_reply_markup(call.message.chat.id, call.message.message_id)
```

```
@bot.callback_query_handler(func=lambda call: call.data.split('#')[0]=='model')
def get_next_models(call):
global counter
global product
global brand
text = 'Оберіть варінт (номер) моделі продукту, впишіть його та відправте боту.\n\n'
page = int(call.data.split('#')[1])

for k, key in enumerate(main_table[product][brand], start=1):
    if k > counter and str(k)[-1] == '1':
            counter = k
            break
    if k >= counter:
        text += f'{k}. {key}\n'
bot.delete_message(call.message.chat.id, call.message.message_id)
send_models_page(call.message, product, brand, text, page)

@bot.message_handler(content_types=['text'])
def get_model_info(msg):
global prodcut
global brand
price = 0
keyboard = telebot.types.InlineKeyboardMarkup()
msg_text = msg.text
if msg_text.isdigit() is False:
```

```
            bot.send_message(msg.chat.id, text=f'Будь ласка, введіть номер моделі або команду.')
        else:
            for i, item in enumerate(main_table[product][brand], start=1):
                if int(i) == int(msg_text):
                    price = main_table[product][brand][item][0]
                    link = main_table[product][brand][item][1]
                    key_link = telebot.types.InlineKeyboardButton(text=item, url=link)
                    keyboard.add(key_link)
            bot.send_message(msg.chat.id, text=f'Ціна обраного товару: {price} грн.', reply_markup=keyboard)

        def send_models_page(message, pr, br, text, page=1):
        page_len = (len(main_table[pr][br]) // 10) + 1
        paginator = InlineKeyboardPaginator(page_len, current_page=page, data_pattern='model#{page}')
        bot.send_message(message.chat.id, text=text, reply_markup=paginator.markup)

        if __name__ == '__main__':
        logger.info(f'\n\tNow u can interact with the bot')
        bot.polling(none_stop=True)
```