МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
Факультет електроніки та інформаційних технологій

Кафедра комп'ютерних наук

Кваліфікаційна робота бакалавра

ТЕЛЕГРАМ-БОТ ДЛЯ РОЗРАХУНКУ ВАРТОСТІ І ОФОРМЛЕННЯ ДОСТАВКИ ТОВАРУ

Здобувач освіти гр. ІН-81 Євген МАЛЬЦЕВ

Науковий керівник,
кандидат фізико-математичних наук,
асистент кафедри комп’ютерних наук Ольга ШУТИЛЄВА

Завідувач кафедри
доктор технічних наук, професор Анатолій ДОВБИШ

СУМИ 2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютерних наук

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

ЗАВДАННЯ
до кваліфікаційної роботи

Студента 4-го курсу, групи ІН-81 спеціальності 122 - Комп'ютерні науки, дистанційної форми навчання Мальцева Є.Д.

Тема: Телеграм-бот для розрахунку вартості і оформлення доставки товару

Затверджена наказом по СумДУ
№_____ від _____2022 р.

Зміст пояснювальної записки: 1) літературний огляд за обраною тематикою роботи; 2) постановка завдання для розробки; 3) практична реалізація.

Дата видачі завдання «_____»_____ 2022 р.
Керівник роботи _____ Шутилєва О.В.
Завдання прийняв до виконання _____ Мальцев Є.Д.

РЕФЕРАТ

Записка: 26 стор., 4 рис., 1 додаток, 16 джерел.

Об’єкт дослідження – моніторинг іноземних ресурсів для придбання автомобілів.

Мета роботи – розробка платформи для організації процесу пошуку, придбання та доставки іноземних автомобілів з допомогою технологій чат-ботів для мессенджеру Telegram.

Методи дослідження – методи дослідження аудиторії та порівняння подібних сервісів.

Результати – розроблено програмне забезпечення для пошуку, придбання та доставки іноземних автомобілів. При цьому, для зручності та спрощення взаємодії з платформою розроблено API та чат-бот який його використовує. Розроблене програмне забезпечення реалізовано за допомогою мови програмування Python.

TELEGRAM BOT, PYTHON, ПРОДАЖ,
СЕРВІС, API

ЗМІСТ

ВСТУП

Актуальність роботи. Сьогодні майже у кожного є смартфон на якому встановлені мессенджери, які перевернули звичайне життя. Не можливо уявити день без користування ними, особливо в такий важкий час як сьогодні, не секрет, що у нас в Україні зараз тривають бойові дії, підчас яких кожен другий черпає новини через Telegram канали.

Користувачам вже ввійшло в звичку користуватися новітніми сервісами через чат ботів, які знаходяться в мессенджері. Кожен день з’являються нові боти, які спрощують життя людям, попит на які збільшується швидше ніж їх кількість.

Майже у кожній сфері є тисячі віртуальних помічників Telegram, які допомагають людям в повсякденному житті. Велика кількість компаній світу розробляють або вже мають власного бота. Адже це дієвий спосіб привернути більшу кількість споживачів до їх продукції. Я гадаю, що Telegram бот є дуже перспективним шляхом розвитку, як для великого бізнесу, так і для малого.

Метою роботи є розробка чат-боту для мессенджеру Telegram, як платформи для організації процесу пошуку, придбання та доставки іноземних автомобілів в Україну.

1 ІНФОРМАЦІЙНІЙ ОГЛЯД

1.1 Огляд проблемної області

В Україні прослідковується глобальна тенденція купівлі машин з інших країн світу, особливо під час війни і відсутності мита. Велика кількість людей не може самостійно придбати автомобіль, а компанії які може з цим допомогти беруть комісію, яка іноді перевищує 30% вартості товару.

Найбільша складність при покупці машини з Європи це пошук та розрахунок всіх витрат під час оформлення та доставки в Україну, саме тому задля оптимізації бізнесу вигідніше за все перекласти всю монотонну роботу на чат-бота. Людина може витратити години або, навіть, дні в той час як алгоритм зробить все за пару секунд.

1.2 Огляд подібних рішень

На даний момент є велика кількість подібних зарубіжних рішень в вигляді сайтів та додатків для телефонів. Частіше за все користувачі шукають власноруч автомобілі серед тисяч оголошень за допомогою фільтрів. Розглянемо найпопулярніші з них:

- markt.de
- 12gebrauchtwagen.de

markt.de, є популярним сайтом в Німеччині, для продажу товарів від людей. В нього є велика кількість фільтрів для пошуку автомобілів, а саме:

- Рік випуску
- Ціна
- Марка
- Модель
- Вид палива
- Регіон продажу

- Тип коробки передач

З недоліків він має відсутність:

- української мови
- сповіщення при появі оголошення яке відповідає фільтрам
- розрахунок вартості авто, якщо його будуть купувати з іншої країни.

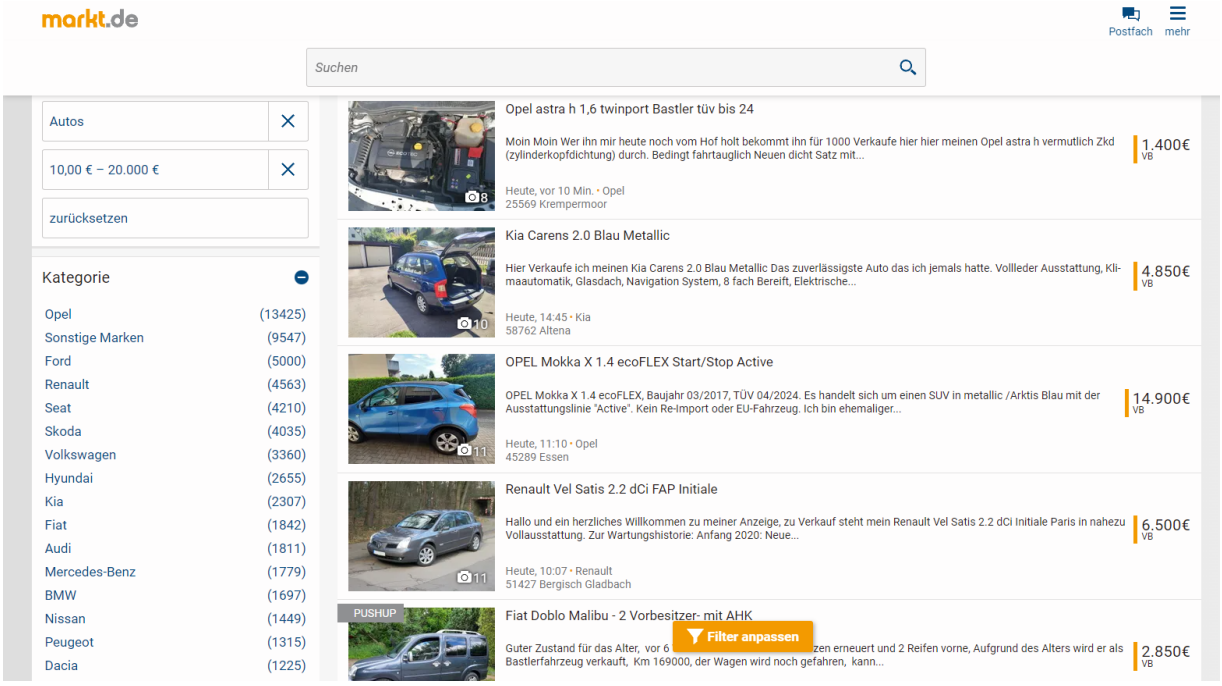

Рисунок 1.1 – Приклад ресурсу markt.de

У свою чергу 12gebrauchtwagen.de дещо відрізняється від markt.de тим, що цей сервіс використовується тільки для продажу автомобілів від офіційних представників компаній, які в свою чергу продають авто за дещо вищу ціну ніж звичайні люди, націлений даний сервіс більше на внутрішній ринок Німеччини.

До переваг сервісу можна віднести можливість подавати оголошення тільки автосалонам, що гарантує відсутність проблем з товаром. На сайті присутні фільтри для пошуку, такі як:

- Марка
- Модель
- Рік випуску

- Ціна
- Пробіг

З недоліків можна відмітить відсутність:

- Української мови
- Розрахунок доставки
- Пошук за місцем знаходження

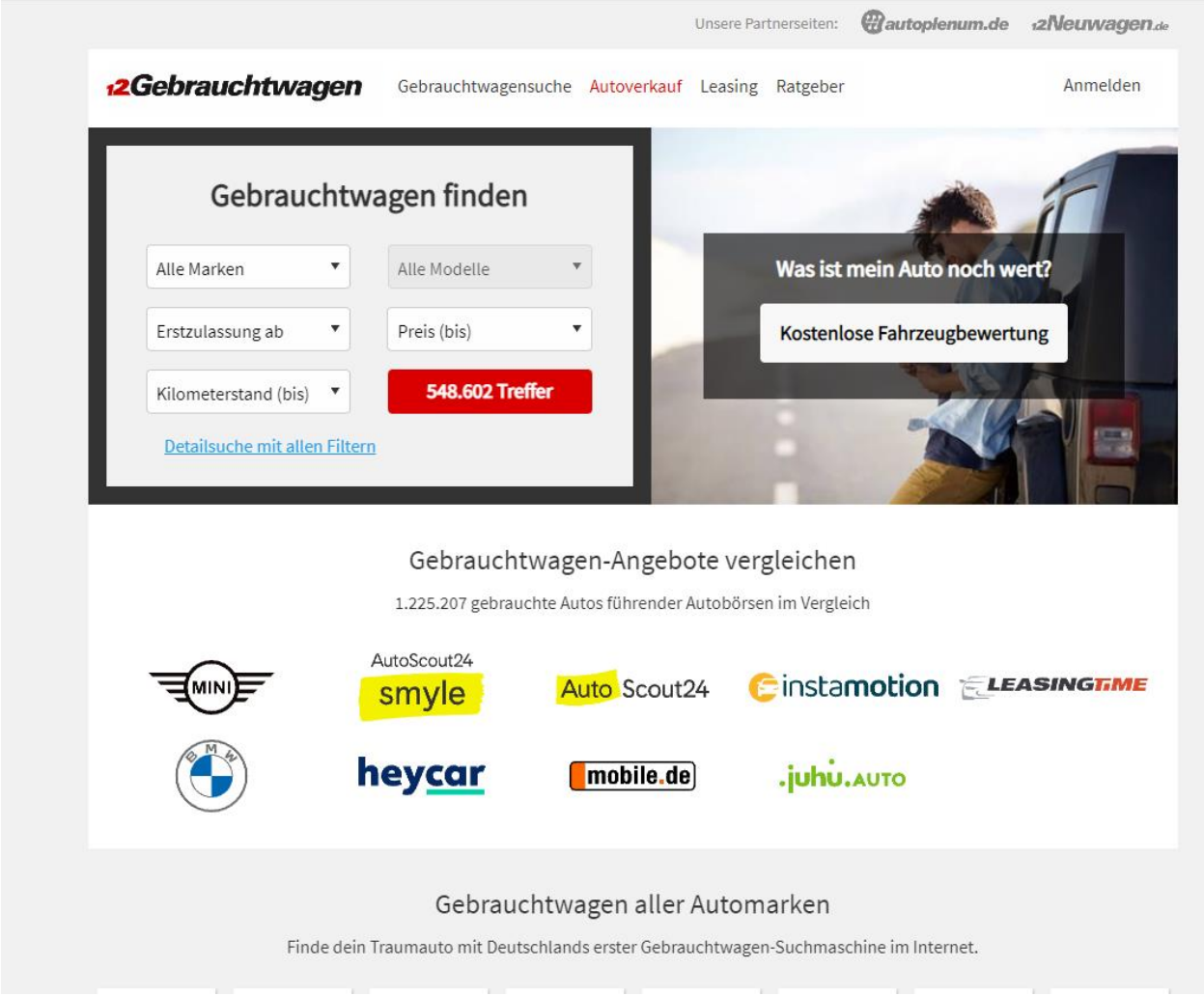

Рисунок 1.2 – Сторінка сервісу 12gebrauchtwagen.de

1.3 Актуальність створення чат-бота

У даний час дуже актуальною є проблема автомобілів, так як у нас в країні йде війна, багато людей втратили свої авто, держава відмінила обов’язкове мито. На даний час дуже важко знайти гарний варіант серед

оголошень, адже велика кількість людей постійно дивуйтеся оголошення авто на зарубіжних сайтах.

В таких ситуаціях треба автоматоматизування процесу, для збільшення продуктивності, наприклад чат-ботом, який буде власноруч робити рутинну роботу.

1.4 Постановка задачі

Розробити сервіс, який повинен шукати авто за фільтром, і відсилати знайдені результати через чат-бот.

Для цього потрібно виконати такі завдання:

1. Створити Telegram чат-бота, через який буде взаємодія з користувачем.
2. Створити базу даних, в якій будуть результати пошуку
3. Написати API, який буду шукати авто, добавляти їх в базу даних, відповідати на команди користувача та присилати йому оголошення.

2 ВИБІР МЕТОДУ РІШЕННЯ

2.1 Вибір програмної реалізації

Для розробки даного продукту ідеально підходить мова програмування PYTHON, адже зараз більшість подібних проектів написано саме на ній, вона має багату кількість бібліотек під роботу з Telegram, в разі виникнення труднощів можливо легко знайти приклади вирішення в інтернеті. Розробка застосунку займає менше часу ніж на аналогічних мовах, готовий код зразу можливо використовувати на любій платформі без значних змін.

Найкраще середовище для написання коду, на нашу думку, є PYCharm, в ньому є автоматичне завантаження бібліотек, які відсутні в базовому середовищі, автоматичний пошук помилок, зручне представлення написаного кода. PYCharm автоматично створює проект та проводить індексацію всіх файлів та папок в ньому, має зручне представлення структури проекту, підтримує більшість видів файлів які можуть знадобитися в ході написання продукту.

2.2 Вибір бібліотеки для взаємодії з Telegram

На даний час, є багато бібліотек для написання даного бота, я вирішив взяти за основу взаємодії з Telegram бібліотеку aiogram, вона схожа на pytelegrambotapi, але відрізняється більшою швидкістю взаємодії та, саме головне, вона має асинхронність, з якою відкривається більше можливостей та функцій бота.

Selenium, наступна по важливості, після aiogram, бібліотека. Вона дозволить нам відкрити в середені коду браузер, за допомогою якого ми зможемо шукати потрібну нам інформацію.

Бібліотеку logging необхідно використовувати для відображання інформації в консолі середовища, з її допомогою зможу швидше знаходити

помилки в роботі та бачити що виконується в реальному часі. Peewee, використаємо для зручного використання бази даних.

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ

3.1 Інформаційна модель

Даний бот повинен знайти автомобілі відповідаючи нашим критеріям оголошення автомобілів на декількох сайтах. Знайденні оголошення автоматично форматуватися в вигляд:

- Лінк
- Назва оголошення
- Ціна
-

Далі потрібно врахувати, що оголошення не тільки будуть додаватися, а і видалятися, для цього потрібно буде імплантувати функцію перевірки лінка оголошення, і якщо не буде існувати наступного видаляти з бази його.

Потрібно створити базу даних для всіх критеріїв пошуку авто під кожного користувача окремо підв’язавши їх під chat id, щоб запити були індивідуальні і кожен бачив лише оголошення які йому потрібні, з якої потім будуть братися критерії пошуку для додавання в базу знайдених оголошень, тим самим ми оптимізуємо пошук, якщо будуть однакові запити в різних користувачів, в базі не будуть дублюватися оголошення.

Після пошуку оголошень, потрібно реалізувати взаємодію с користувачем. Порівняння двох баз критерій та самих оголошень, щоб користувач міг отримати потрібні йому авто. Додаємо команди та кнопки курування ботом для зручності управління функціоналом. За допомогою яких користувач мав змогу отримати інструкцію для користування, можливість додавання та видалення фільтрів пошуку, перегляд вже знайдених оголошень.

3.2 Програмна реалізація

Першим кроком створимо самого чат-бота, допоможе с цим вбудований в Telegram інструмент BotFather. Водимо назву нашого бота та отримуємо наш токен с допомогою якого наш API буду ним керувати. Після цього створюємо пустий проект в PyCharm, середовище розробки автоматично створить всю потрібну структуру файлів та папок, а також ініціалізує всі необхідні класи та файли для запуску нашого коду. Наступним кроком підключаємо систему контроля версій Git. Що в свою чергу дозволить контролювати версію додатку та наддасть можливість відновити код на випадок його втрати або пошкодження.

Після виконання цих кроків створимо функцію яка в свою чергу згенерує нам базу даних, в якій ми будемо зберігати знайдені автомобілі та пошукові фрази користувачів:

```
class BaseModel(Model):
    class Meta:
        database=db



class car(BaseModel):
    title=CharField()
    price=TextField()
    url=TextField()


class SearchModel(BaseModel):
    title=CharField()
    chatid=CharField()

def init_db():
    db.create_tables([car, SearchModel])
```

Далі необхідно створити запити які будуть додавати в базу ключові пошукові фрази та знайдені авто, в запиті при добавлені автомобіля допишемо повідомлення для користувача, що знайдено авто за критеріями його пошуку:

```
async def process_search_model(message):
    search_exist = True
    try:
        search =
SearchModel.select().where(SearchModel.title==message.text).get()
        search.delete_instance()
        await message.answer('строка пошуку{} видалена'.format(message.text))
        return search_exist
    except DoesNotExist as de:
        search_exist = False

    if not search_exist:
       rec=SearchModel(title=message.text, chatid=message.chat.id)
       rec.save()
       await message.answer('строка пошуку {}
добавлена'.format(message.text))
    else:
       await message.answer('строка пошуку {} занята'.format(message.text))
    return search_exist

async def process_car_add(title,price,url,chat_id, bot):
    car_exist=True
    try:
       scar= car.select().where(car.title==title).get()
    except DoesNotExist as de:
        car_exist = False
    if not car_exist:
        rec=car(title=title,price=price, url=url)
        rec.save()
        prcint = price.replace('€', '')
        priceukr = (int(prcint.replace('.', '')) + deliver + muto + sert) *
kurs
        message_text = 'Знайдено автомобіль за пошуковою фразою {}\r\n Ціна в
Німеччині :{} \r\n Ціна в Україні(доставка {} € + мито {} € + сертифікація {}
€): {}грн\r\n  {}'.format(
            title, price, deliver, muto, sert, priceukr,
utils.markdown.hlink(title,url))

        await
bot.send_message(chat_id=chat_id,text=message_text,parse_mode=ParseMode.HTML)
    return car_exist
```

Після цих кроків нам необхідно зробити можливість пошуку ботом автомобілей, реалізуємо це за допомогою бібліотеки selenium, яка буде нам відкривати сторінку браузера. В якій ми будемо шукати наші оголошення, створимо клас:

```
class ParseCar:
    def __init__(self, url, bot=None):
        self.driver = webdriver.Chrome(executable_path=DRIVER_PATH)
        self.driver.minimize_window()
        self.url = url
        self.bot = bot

    def __del__(self):
        self.driver.close()

    async def parse(self):
        search_models = find_all_search()

        for page in range(1,20):
            print(self.url.format(page))
            self.driver.get(self.url.format(page))
            items = len(self.driver.find_elements_by_class_name("clsy-c-
result-list "))
            for item in range(items):
                scars = self.driver.find_elements_by_class_name("clsy-c-
result-list-item--wide")
                for car in scars:
                    car_title=car.accessible_name
                    prices=car.find_element_by_class_name("clsy-c-result-
list-item__price-amount")
                    price=prices.text
                    car_href = 'https://www.markt.de/essen' +
car.get_attribute('data-onclick-url')
                    for search_model in search_models:
                        if car_title.find(search_model.title) >= 0:
                            await  process_car_add(car_title,price, car_href,
search_model.chatid, self.bot)
```

Окрім цього в системі не вистачає обробки команд. Зробимо функцію яка буде вітати користувача після запуска чат-бота і в ній зробимо кнопки для додавання запиту або інструкції:

```
@dp.message_handler(commands=['start'])
async def cmd_start(message: types.Message):
    keyboard = types.ReplyKeyboardMarkup(resize_keyboard=True)
    btn1 = types.KeyboardButton("Інструкція(HELP)")
    btn2 = types.KeyboardButton("Додати запит")
    keyboard.add(btn1,btn2)
    await message.answer("Привіт! Я телеграмм бот, який допоможе тобі знайти
авто мрії.", reply_markup=keyboard)
```

Наступним кроком нам потрібно зробити відповідь на наші кнопки, першою зробимо інструкцію і добавим в неї можливість додавання запиту:

```
@dp.message_handler(Text(equals='Інструкція(HELP)'))
async def helps(message: types.Message):
    keyboard = types.ReplyKeyboardMarkup(resize_keyboard=True)
    btn2 = types.KeyboardButton("Додати запит")
    keyboard.add(btn2)
    await message.answer("Бот автоматично шукає авто по заданим параметрам та
якщо знаходе подібне відсилая його тобі\r\n "
                         "Щоб додати(англійською) або подивитися уже задані
запит натисни кнопку нижче, або напиши команду /search \r\n"
                         "Щоб видалити запит наберіть його повторно\r\n"
                         " Якщо хочеш подивитися список знайдених авто напиши
команду /list", reply_markup=keyboard)
```

Ми потребуємо можливості додавання та видалення запиту. Реалізуємо це через натискання кнопки та функції яку ми зробили раніше:

```
@dp.message_handler(Text(equals='Додати запит'))
async def search (message: types.Message):
    search_models = find_id_search(message.chat.id)
    for search_model in search_models:
        await message.answer(text=search_model.title)
    await message.reply("Ведіть пошукову фразу",
```

```
reply_markup=types.ReplyKeyboardRemove())
    pass
    @dp.message_handler()
    async def send_list(message: types.Message):
        await process_search_model(message)
```

Реалізуємо список знайдених перед цим автомобілів за допомогою команди /list та додамо команду /serch для виклику функції додавання запитів:

```
@dp.message_handler(commands='list')
async def send_listc(message: types.Message):
    search_models = find_id_search(message.chat.id)
    scars = find_all_cars()
    for car in scars:
        car_title = car.title
        for search_model in search_models:
            search_title = search_model.title
            if car_title.find(search_title) >= 0:
                prcint=car.price.replace('€','')
                priceukr=(int(prcint.replace('.',''))+deliver+muto+sert)*kurs
                message_text = 'Знайдено автомобіль за пошуковою фразою
{}\r\n Ціна в Німеччині :{} \r\n Ціна в Україні(доставка {} € + мито {} € +
сертифікація {} €): {}грн\r\n  {}'.format(search_title, car.price,
deliver,muto,sert ,priceukr, utils.markdown.hlink(car_title, car.url))
                await message.answer(text=message_text,
parse_mode=ParseMode.HTML)

@dp.message_handler(commands='search')
async def send_search(message: types.Message):
    search_models= find_id_search(message.chat.id)
    for search_model in search_models:
        await message.answer(text=search_model.title)

    @dp.message_handler()
    async def send_list(message: types.Message):
        await process_search_model(message)
```

Основна частина кода написана, так виглядає база даних яку ми створили:

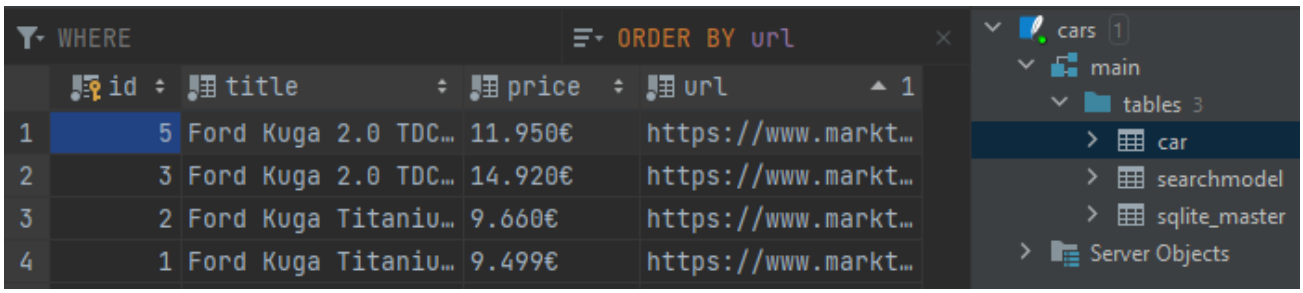

Рисунок 3.1 – Таблиця car

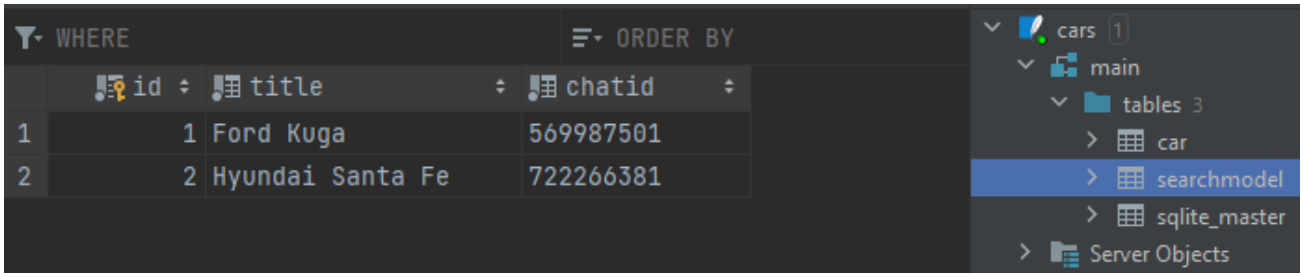

Рисунок 3.2 – Таблиця searchmodel

Приклад роботи Telegram чат-бота:

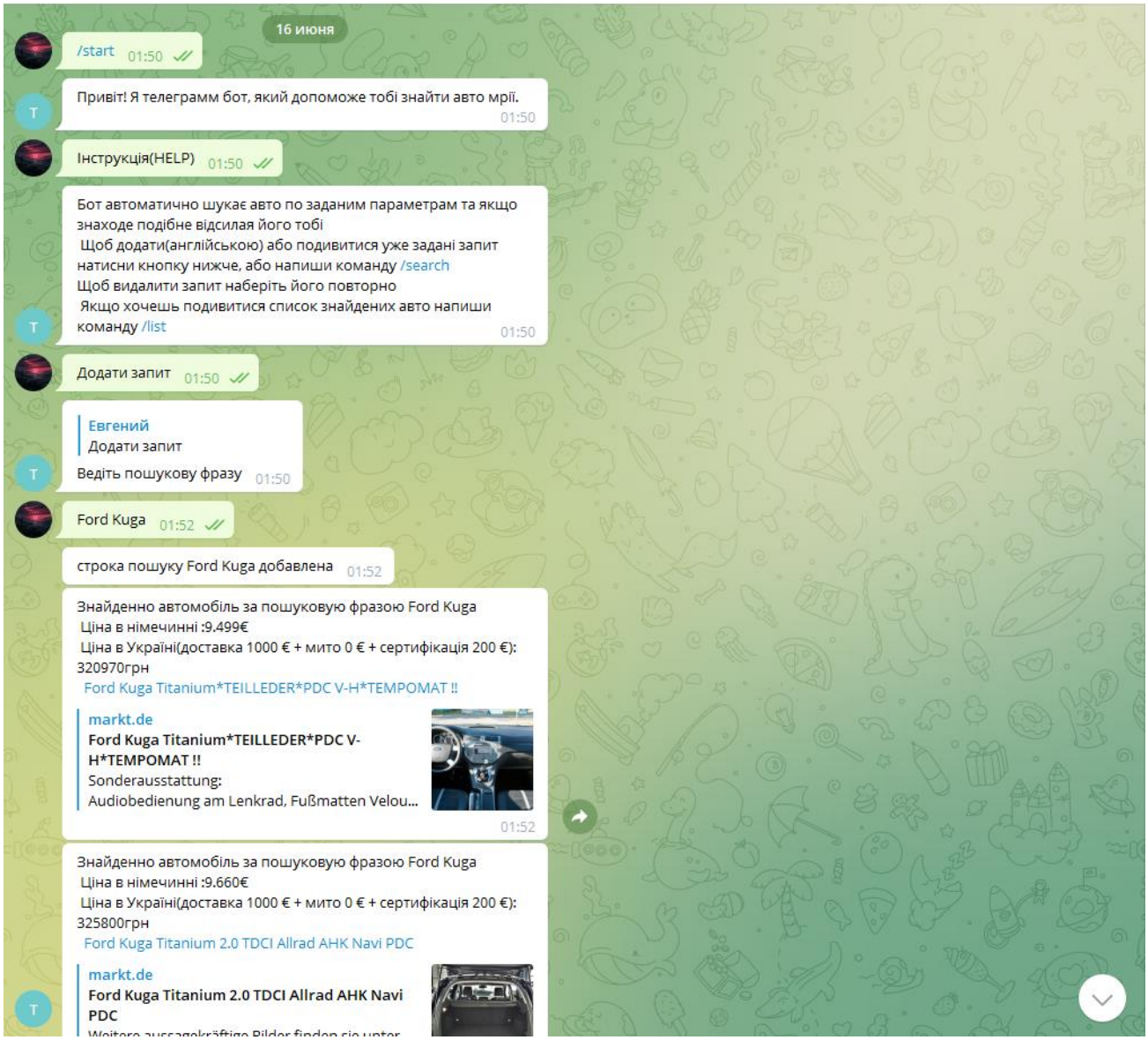

Рисунок 3.3— Приклад роботи

У ході розробки було створено працездатний чат-бот, у майбутньому проєкт буде масштабуватися.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було виконано наступне:

1) проведено огляд існуючих аналогів та обрано необхідні інструменти для розробки чат-боту;

1) було сформовано вимоги до функціоналу, який необхідно розробити;

2) розроблено концепцію розробки клієнтської та серверної частини додатку;

3) описано модель взаємодії з даними у додатку.

Результатом роботи є функціонуючий чат-бот, який виконує функції підбору автомобілів. Реалізовано підбір автомобіля за необхідними параметрами.

СПИСОК ЛІТЕРАТУРИ

1. **pyTelegramBotAPI**. Python Telegram Bot API [Електронний ресурс] – Режим доступу: https://github.com/eternnoir/pyTelegramBotAPI.
2. **AIOGram**. A pretty simple and fully asynchronous library for Telegram Bot API written with asyncio and aiohttp [Електронний ресурс] – Режим доступу: https://github.com/aiogram/aiogram.
3. Selenium with Python [Електронний ресурс] – Режим доступу: https://selenium-python.readthedocs.io/.
4. peewee — peewee 3.14.10 documentation [Електронний ресурс] – Режим доступу: http://docs.peewee-orm.com/en/latest/.
5. asyncio — Asynchronous I/O — Python 3.10.5 documentation [Електронний ресурс] – Режим доступу: https://docs.python.org/3/library/asyncio.html .
6. logging — Logging facility for Python [Електронний ресурс] – Режим доступу: https://docs.python.org/3/library/logging.html .

ДОДАТОК А

Лістинг основних частин коду

Файл main.py

```
import asyncio
import logging
from aiogram.dispatcher.filters import Text
from aiogram import Bot, Dispatcher, executor, types, utils
from aiogram.types import ParseMode
from parse import ParseCar
from config import TOKEN, URL, deliver, muto, kurs, sert
from db import process_search_model, init_db , find_id_search, find_all_cars
logging.basicConfig(level=logging.INFO)
bot = Bot(token=TOKEN,parse_mode=ParseMode.HTML)
dp = Dispatcher(bot)



@dp.message_handler(commands=['start'])
async def cmd_start(message: types.Message):
    keyboard = types.ReplyKeyboardMarkup(resize_keyboard=True)
    btn1 = types.KeyboardButton("Інструкція(HELP)")
    btn2 = types.KeyboardButton("Додати запит")
    keyboard.add(btn1,btn2)
    await message.answer("Привіт! Я телеграмм бот, який допоможе тобі знайти авто мрії.",
reply_markup=keyboard)

@dp.message_handler(Text(equals='Інструкція(HELP)'))
async def helps(message: types.Message):
    keyboard = types.ReplyKeyboardMarkup(resize_keyboard=True)
    btn2 = types.KeyboardButton("Додати запит")
    keyboard.add(btn2)
    await message.answer("Бот автоматично шукає авто по заданим параметрам та якщо знаходе
подібне відсилая його тобі\r\n "
                         "Щоб додати(англійською) або подивитися уже задані запит натисни кнопку нижче,
або напиши команду /search \r\n"
```

```
                "Щоб видалити запит наберіть його повторно\r\n"
                " Якщо хочешь подивитися список знайдених авто напиши команду /list",
reply_markup=keyboard)

@dp.message_handler(Text(equals='Додати запит'))
async def search (message: types.Message):
   search_models = find_id_search(message.chat.id)
   for search_model in search_models:
      await message.answer(text=search_model.title)
   await message.reply("Ведіть пошукову фразу", reply_markup=types.ReplyKeyboardRemove())
   pass
   @dp.message_handler()
   async def send_list(message: types.Message):
      await process_search_model(message)

@dp.message_handler(commands='help')
async def help(message: types.Message):
   await message.answer(
      "Бот автоматично шукає авто по заданим параметрам та якщо знаходе подібне відсилая його
тобі\r\n "
      "Щоб додати(англійською) або подивитися уже задані запит натисни кнопку нижче, або
напиши команду /search \r\n"
      "Щоб видалити запит наберіть його повторно\r\n"
      " Якщо хочешь подивитися список знайдених авто напиши команду /list")

@dp.message_handler(commands='list')
async def send_listc(message: types.Message):
   search_models = find_id_search(message.chat.id)
   scars = find_all_cars()
   for car in scars:
      car_title = car.title
      for search_model in search_models:
         search_title = search_model.title
         if car_title.find(search_title) >= 0:
            prcint=car.price.replace('€','')
            priceukr=(int(prcint.replace('.',''))+deliver+muto+sert)*kurs
```

```
            message_text = 'Знайденно автомобіль за пошуковую фразою {}\r\n Ціна в німечинні :{}
\r\n Ціна в Україні(доставка {} € + мито {} € + сертифікація {} €): {}грн\r\n  {}'.format(search_title,
car.price, deliver,muto,sert ,priceukr, utils.markdown.hlink(car_title, car.url))
            await message.answer(text=message_text, parse_mode=ParseMode.HTML)



@dp.message_handler(commands='search')
async def send_search(message: types.Message):
   search_models= find_id_search(message.chat.id)
   for search_model in search_models:
      await message.answer(text=search_model.title)

   @dp.message_handler()
   async def send_list(message: types.Message):
      await process_search_model(message)




async def scheduled(wait_for, parser):
   while True:
      await asyncio.sleep(wait_for)
      await parser.parse()

if __name__ == '__main__':
   init_db()
   parser = ParseCar(url=URL,bot=bot)
   loop = asyncio.get_event_loop()
   loop.create_task(scheduled(1000, parser))
   executor.start_polling(dp, skip_updates=True)
```

Файл db.py

```
from aiogram.types import ParseMode
from aiogram import utils
from peewee import *
```

```
from config import deliver, muto, kurs, sert
db = SqliteDatabase('cars.db')

class BaseModel(Model):
    class Meta:
        database=db



class car(BaseModel):
    title=CharField()
    price=TextField()
    url=TextField()


class SearchModel(BaseModel):
    title=CharField()
    chatid=CharField()


def find_all_cars():
    return car.select()

def find_id_search(chat_id):
    return SearchModel.select().where(SearchModel.chatid == chat_id)

def find_all_search():
    return SearchModel.select()
async def process_search_model(message):
    search_exist = True
    try:
        search = SearchModel.select().where(SearchModel.title==message.text).get()
        search.delete_instance()
        await message.answer('строка пошуку{} видалена'.format(message.text))
        return search_exist
    except DoesNotExist as de:
```

```
        search_exist = False

    if not search_exist:
        rec=SearchModel(title=message.text, chatid=message.chat.id)
        rec.save()
        await message.answer('строка пошуку {} добавлена'.format(message.text))
    else:
        await message.answer('строка пошуку {} зайнята'.format(message.text))
    return search_exist

async def process_car_add(title,price,url,chat_id, bot):
    car_exist=True
    try:
        scar= car.select().where(car.title==title).get()
    except DoesNotExist as de:
        car_exist = False
    if not car_exist:
        rec=car(title=title,price=price, url=url)
        rec.save()
        prcint = price.replace('€', '')
        priceukr = (int(prcint.replace('.', '')) + deliver + muto + sert) * kurs
        message_text = 'Знайденно автомобіль за пошуковую фразою {}\r\n Ціна в німечинні :{} \r\n
Ціна в Україні(доставка {} € + мито {} € + сертифікація {} €): {}грн\r\n  {}'.format(
            title, price, deliver, muto, sert, priceukr, utils.markdown.hlink(title,url))

        await bot.send_message(chat_id=chat_id,text=message_text,parse_mode=ParseMode.HTML)
    return car_exist
def init_db():
    db.create_tables([car, SearchModel])
Файл parse.py
from config import DRIVER_PATH, URL
from selenium import webdriver
from db import find_all_search, process_car_add


class ParseCar:
```

```
def __init__(self, url, bot=None):
    self.driver = webdriver.Chrome(executable_path=DRIVER_PATH)
    self.driver.minimize_window()
    self.url = url
    self.bot = bot

def __del__(self):
    self.driver.close()

async def parse(self):
    search_models = find_all_search()

    for page in range(1,20):
        print(self.url.format(page))
        self.driver.get(self.url.format(page))
        items = len(self.driver.find_elements_by_class_name("clsy-c-result-list "))
        for item in range(items):
            scars = self.driver.find_elements_by_class_name("clsy-c-result-list-item--wide")
            for car in scars:
                car_title=car.accessible_name
                prices=car.find_element_by_class_name("clsy-c-result-list-item__price-amount")
                price=prices.text
                car_href = 'https://www.markt.de/essen' + car.get_attribute('data-onclick-url')
                for search_model in search_models:
                    if car_title.find(search_model.title) >= 0:
                        await  process_car_add(car_title,price, car_href, search_model.chatid, self.bot)
```