Дослідження методів розробки чат-ботів

Підготувала студентка 3 курсу КН Сидорова Єлізавєта

Науковий керівник Салата Кирило Володимирович

Вступ

Актуальність: цифрові технології, широкий спектр засобів, зручність й універсальність

Мета: дослідження теоретичної основи, її застосування під час створення власного чат-боту й аналіз роботи

Постановка задачі: дослідження та розробка застосунку

ТЕХНОЛОГІЇ, ЯКІ ЗАКЛАДЕНІ В ЧАТ-БОТ

Тип чат-боту:

На основі кнопкового меню

Використані методи:

Основи обробки природної мови

Мова програмування:

Python

Месенджер:

Telegram

Огляд методів та тестування

botToken.py

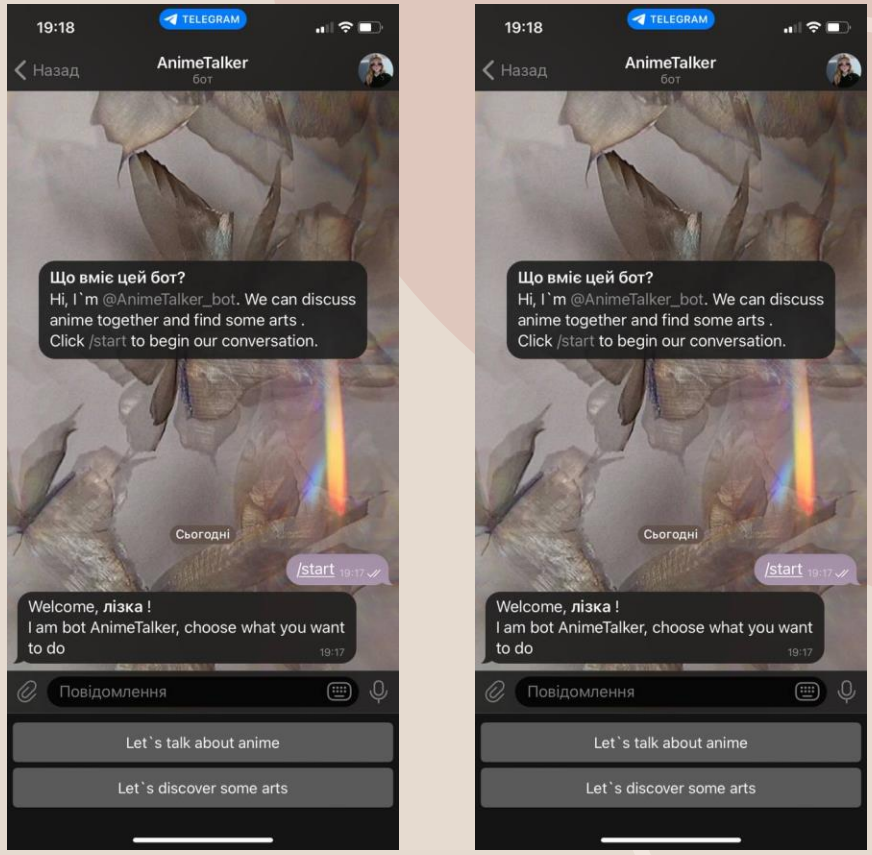
```python
import telebot
token = '5986581745:AAFjedvw-7_bW6bXxaKy9
bot = telebot.TeleBot(token)
```

Модуль botToken.py

main.py

```python
import urllib.error
from telebot import types
from analysingMessage import analyse
from fatic import talking
from botToken import bot
from checking import *
from animeInfo import choose_character, anime_info, for_request, make_request

user_info = {}


# command /start + menu bar
@bot.message_handler(commands=['start'])
def start(message):
    id_user = message.from_user.id
    if user_info.get(id_user) is None:
        user_info[id_user] = ['anime_link', 'characters', 'name_anime']
    markup = types.ReplyKeyboardMarkup(resize_keyboard=True, row_width=1, one_time_keyboard=True)
    button1 = types.KeyboardButton('Let`s talk about anime')
    button2 = types.KeyboardButton('Let`s discover some arts')
    markup.add(button1, button2)
    answer = f'Welcome, *{message.from_user.first_name}* ! \n' + talking("greeting")
    bot.send_message(message.chat.id, answer, reply_markup=markup, parse_mode='Markdown')

```

Модуль main.py(1)



Тестування початкового меню та команди /start

Огляд методів та тестування

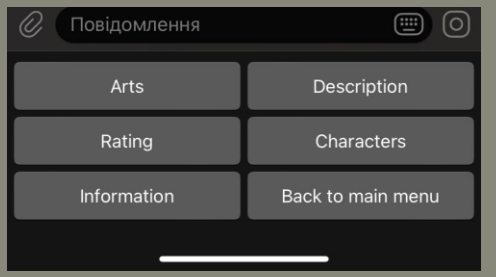
МОДУЛЬ MAIN.PY(2)

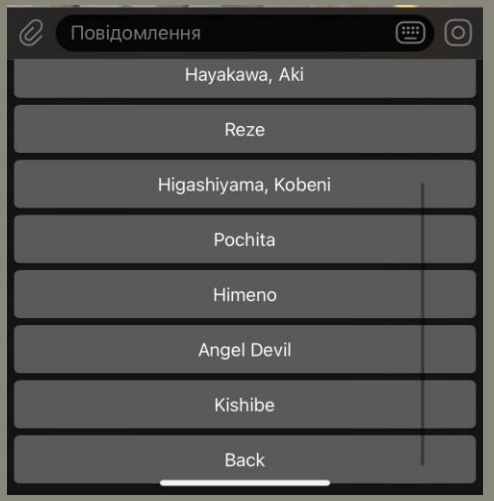
main.py

```
# text
@bot.message_handler(content_types=['text'])
def get_text(message):
    id_user = message.from_user.id
    if id_user not in user_info.keys():
        user_info[id_user] = ['anime_link', 'characters', 'name_anime']
        get_text(message)
    elif message.text == 'Let`s talk about anime':
        bot.send_message(message.chat.id, talking("talking"), parse_mode='html'
    elif message.text == "Let`s discover some arts" or message.text == "Arts":
        if check_anime(id_user, user_info):
            user_info[id_user][1] = choose_character(id_user, user_info)
            ch = (user_info[id_user])[1]
            markup = types.ReplyKeyboardMarkup(resize_keyboard=True, row_width=
            b = types.KeyboardButton('Back')
            for c in ch:
                button = types.KeyboardButton(c)
                markup.add(button)
            markup.add(b)
            bot.send_message(message.chat.id, talking("characters"), reply_mark
        else:
            bot.send_message(message.chat.id, talking("anime miss"), parse_mode
    elif message.text == "Description" or message.text == "Rating" or message.t
            message.text == "Information":
        if check_anime(id_user, user_info):
            text = anime_info(message.text.lower(), id_user, user_info)
            bot.send_message(message.chat.id, text, parse_mode='html')
        else:
            bot.send_message(message.chat.id, talking("anime miss"), parse_mode
    elif message.text == "Back to main menu":
        start(message)
    elif message.text == "Back":
        message.text = 'Yes, that`s my anime'
        get_text(message)
    elif message.text == "Yes, that`s my anime":
        text = 'Choose what do you want:'
        markup = types.ReplyKeyboardMarkup(resize_keyboard=True, row_width=2, c
        button1 = types.KeyboardButton('Description')
```

```
        button2 = types.KeyboardButton('Rating')
        button3 = types.KeyboardButton('Characters')
        button4 = types.KeyboardButton('Information')
        button5 = types.KeyboardButton('Back to main menu')
        button6 = types.KeyboardButton('Arts')
        markup.add(button6, button1, button2, button3, button4, button5)
        bot.send_message(message.chat.id, text, reply_markup=markup)
    elif message.text == "No":
        markup = types.ReplyKeyboardMarkup(resize_keyboard=True, row_width=1,
        button1 = types.KeyboardButton('Back to main menu')
        markup.add(button1)
        bot.send_message(message.chat.id, talking("problems"), reply_markup=m
    elif check_characters(id_user, message.text, user_info):
        bot.send_message(message.chat.id, talking("waiting"), parse_mode='htm
        url = make_request(message.text, '', 'character', id_user, user_info)
        try:
            bot.send_photo(message.chat.id, url)
        except urllib.error.HTTPError:
            url = make_request(message.text, '', 'character', id_user, user_i
            bot.send_photo(message.chat.id, url)
        bot.send_message(message.chat.id, "Pick another one or go to main men
    else:
        user_info[id_user][2] = 'name_anime'
        bot.send_message(message.chat.id, talking("waiting"), parse_mode='htm
        res = (for_request(analyse(message.text), 'anime', id_user, user_info
        if res == "language" or user_info[id_user][2] == 'name_anime':
            bot.send_message(message.chat.id, talking("language"), parse_mode
        else:
            text = 'Is it anime ' + res + '?'
            markup = types.ReplyKeyboardMarkup(resize_keyboard=True, row_widt
            button1 = types.KeyboardButton('Yes, that`s my anime')
            button2 = types.KeyboardButton('No')
            markup.add(button1, button2)
            bot.send_message(message.chat.id, text, reply_markup=markup, pars

bot.polling(none_stop=True)
```

ТЕСТУВАННЯ





Огляд методів та тестування

checking.py

```python
def check_anime(user_id, user_info):
    anime_link = (user_info[user_id])[0]
    if anime_link == 'anime_link':
        return False
    else:
        return True


def check_characters(user_id, name, user_info):
    characters = (user_info[user_id])[1]
    if characters == 'characters' or name not in characters:
        return False
    else:
        return True
```

fatic.py

```python
import random
import json

with open('answers.json') as file:
    data_answers = json.load(file)


def talking(topic):
    for tags in data_answers['answers']:
        topic2 = tags["tag"]
        if topic == topic2:
            answers = tags["message"]
            return random.choice(answers)
```

answers.json

```json
{"answers": [
        {"tag": "greeting",
         "message": ["I am bot AnimeTalker, what would you like to do?", "I am bot AnimeTalker, choose what you want to do",
           "I am AnimeTalker, so... Would you like to talk about the anime or discover some arts", "Nice to see you there, I am bot AnimeTalker, choose what you want to do",
           "Let`s begin with anime you want to talk about or discover arts of your favourite characters"]
        },
        {"tag": "talking",
         "message": ["It`s a great idea, so let`s start from your favourite anime!", "I am crazy about anime, my favourite is Nana, what about you?",
           "So... What anime do you want to discuss?", "What is the name of anime you want to talk about?", "Let`s begin with anime you want to talk about",
           "Tell me please the name of anime", "Let`s start with the name of your anime"]
        },
        {"tag": "characters",
         "message": ["Please choose your character", "Choose the character from the list",
           "Here are characters, please choose one from the list", "Here are the buttons with characters, choose the right one",
           "You need to choose at least one character from the given list"]
        },
        {"tag": "waiting",
         "message": ["Just wait... It will take some time", "Just wait... I am searching for it",
           "Please wait while I am searching", "Nice choice, please wait", "Please wait, I am searching"]
        },
        {"tag": "problems",
         "message": ["Sorry, your request is not correct to me, try again", "Just try again... I can`t work with your request.",
           "Sorry, I can`t find any information to your request. Please, can you say it in other words"]
        },
        {"tag": "language",
         "message": ["Sorry, your request is not correct to me, try to say it in English, please", "Just try again, but write in English.",
           "Sorry, I can`t find any information to your request. Please, can you say it in English, please"]
        },
        {"tag": "anime miss",
         "message": ["Sorry, you need to choose anime at first and then try again", "Just try again, but, please, choose your anime at first",
           "Sorry, you have not chosen any anime, write its name name and try again"]
        },
        {"tag": "info miss",
         "message": ["Sorry, this information is absent", "My apologizes, this information is absent",
           "Sorry, I tried to find this info, but it`s absent"]
        }
    ]
}
```

analysingMessage.py

```python
import re
from textblob import TextBlob


def analyse(message):
    i = len(re.findall(r'\w+', message))
    res = []
    while i > 0:
        blob = TextBlob(message)
        list_ngrams = blob.ngrams(i)
        res += list_ngrams
        i -= 1
    print(res)
    return res
```

```python
def make_request(request, name, typeR, user_id, user_info):
    if typeR == 'anime':

        url = 'https://myanimelist.net/anime.php?q=' + request + '&cat='
        try:
            html = urlopen(url).read()
        except:
            return "language"

        soup = BeautifulSoup(html, 'html.parser')
        res = soup.find_all('strong')
        for el in res:
            anime_name = el.string
            for link in soup.find_all("a", {"class": "hoverinfo_trigger f
                u = link.get('href')
                if u is not None and link.string == anime_name:
                    symbols = ":", "!", "'", "(", ")", ".", "/", "+", ";"
                    for s in symbols:
                        anime_name = anime_name.replace(s, "")
                        name = name.replace(s, "")
                    if name.lower() == anime_name.lower() or (el == res[0
                        (user_info[user_id])[0] = u
                        (user_info[user_id])[2] = anime_name
                        if name.lower() == anime_name.lower():
                            return "exact"
        return "maybe"
    elif typeR == 'character':

        name_anime = (user_info[user_id])[2]
        url = bing_image_urls(request + " " + name_anime, limit=1000)
        rand_url = random.choice(url)
        while rand_url:
            try:
                urlopen(rand_url)
                return rand_url
            except:
                rand_url = random.choice(url)
```

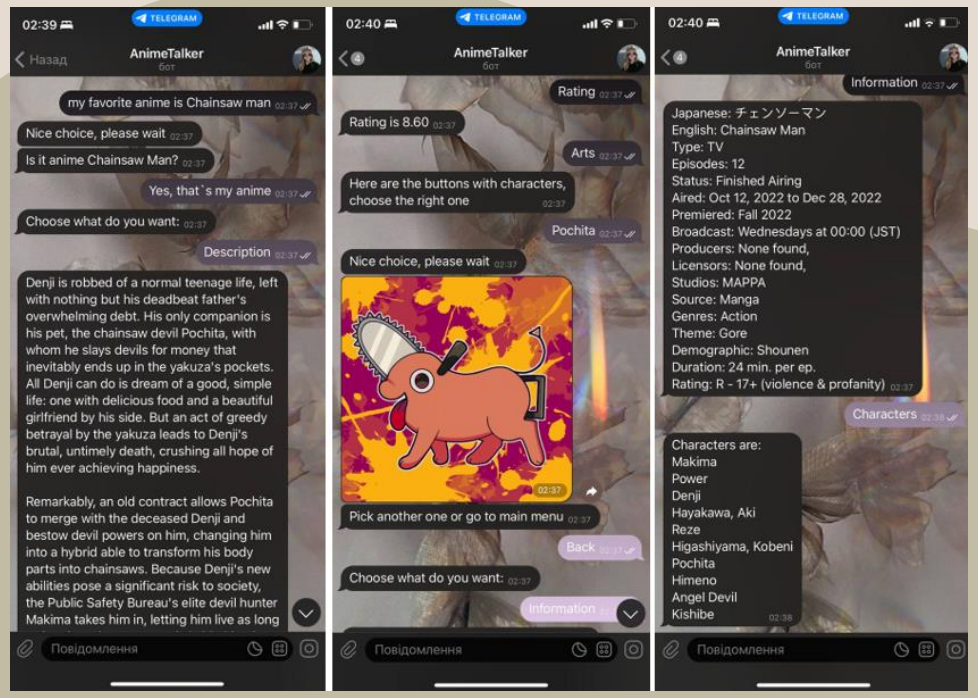
```python
def for_request(list1, typeR, user_id, user_info):
    request = ''
    i = 0
    while i < len(list1):

        words = list1[i]
        k = 0

        while k < len(words):
            if k != 0:
                if typeR == 'anime':
                    request += '%20'
                else:
                    request += '+'
            request += words[k]
            k += 1
        names = [''.join(w) for w in words]
        name = ''
        n = 0
        while n < len(names):
            if n != len(names) - 1:
                name += names[n] + ' '
            else:
                name += names[n]
            n += 1
        calling = (make_request(request, name, typeR, user_id, user_
        if calling == "exact":
            make_request(request, name, typeR, user_id, user_info)
            return (user_info[user_id])[2]
        else:
            request = ''
            i += 1
    return (user_info[user_id])[2]
```

```python
def anime_info(task, user_id, user_info):
    try:
        anime_link = (user_info[user_id])[0]
        html = urlopen(anime_link).read()
    except urllib.error.HTTPError:
        return 'error'

    soup = BeautifulSoup(html, 'html.parser')
    result = ''
    if task == 'description':

        res = soup.find("p", {"itemprop": "description"})
        for e in res:
            if e.string is not None:
                result += e.string
        if result == '':
            result = talking("info miss")
        return result

    elif task == 'rating':
        answer = ''
        res = soup.find_all("div", {"class": "score-label score-8"})
        if res:
            for e in res:
                answer = 'Rating is ' + ''.join(e.string.strip())
        else:
            answer = talking("info miss")
        return answer

    elif task == 'information':
        result = " "
        res = soup.find_all('h2')
        for e in res:
            if e.string == 'Alternative Titles':
                res2 = (e.find_all_next("div", {"class": "spaceit_pad"}))
                i = 0
                while i < len(res2):
                    a = res2[i]
                    t = a.find_next('span').string
                    if t == 'Score:':
                        return result
                    result += t
                    if i == 0:
                        t = ((e.find_next("div", {"class": "spaceit_pad"})).find_all(text=True, recursive=False)[1])
                        t = t.replace('\n', '')
                        result += ' ' + t.lstrip() + '\n'
                    else:
                        a = res2[i - 1]
                        t = ((a.find_next("div", {"class": "spaceit_pad"})).find_all(text=True, recursive=False)[1])
                        if t == '\n':
                            a = res2[i]
                            t = (a.find_next("a"))
                        m = t.string.replace('\n', '')
                        result += ' ' + m.lstrip() + '\n'
                    i += 1
        if result == '':
            return talking("info miss")

    elif task == 'characters':
        result = 'Characters are:' + '\n'
        res = soup.find("div", {"class": "detail-characters-list clearfix"})
        for e in res:
            res2 = (e.find_all("h3", {"class": "h3_characters_voice_actors"}))

            for c in res2:
                result += c.find_next('a').string + '\n'
        if result == '':
            result = talking("info miss")
        return result
```

Модуль animeInfo.py

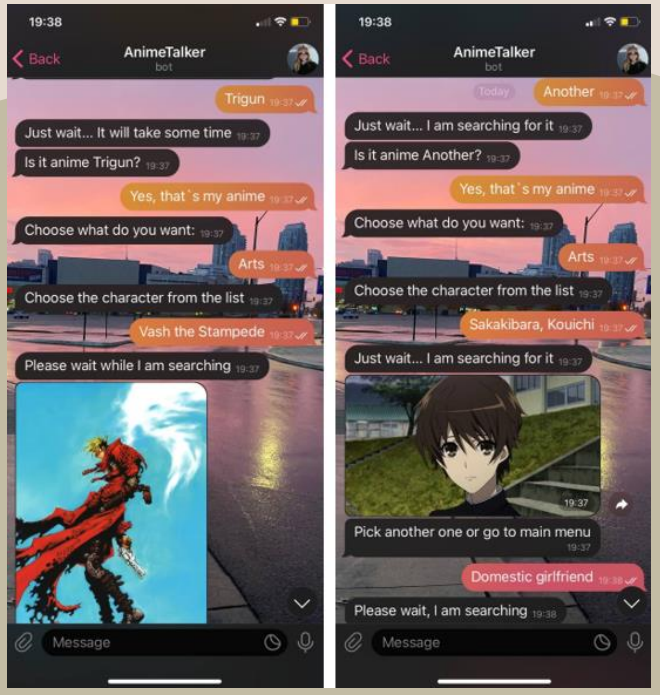
```python
def choose_character(id_user, user_info):
    characters = anime_info('characters', id_user, user_info)
    characterArr = characters.split('\n')
    if len(characterArr) > 0:
        del characterArr[0]
        del characterArr[len(characterArr) - 1]
    return characterArr
```

Кінцеве тестування



Перший користувач

Другий користувач



Третій користувач

Висновок

Завдяки цьому дослідженню вдалось розробити власного чат-бота для спрощення пошуку інформації в інтернеті, який можна й надалі вдосконалювати та впроваджувати на постійне онлайн використання.

Дякую за увагу!