**Московский государственный технический**

**университет им. Н.Э. Баумана**

**Факультет ИУ**

**Кафедра ИУ5**

**Курс «Основы информатики»**

**Отчет лабораторной работе №5**

**lab\_bot1**

Выполнил студент группы ИУ5-33Б:

Османов З. Ш.

Подпись и дата:

Проверил преподаватель каф.:

Гапанюк Ю. Е.

Подпись и дата:

Москва, 2024 г

**Описание задания**

**Задание:**

1. Разработайте простого бота для Telegram. Бот должен использовать функциональность создания кнопок.

**Текст программы:**

import logging

import httpx

from telegram import Update

from telegram.ext import (

    ApplicationBuilder,

    CommandHandler,

    MessageHandler,

    filters,

    ContextTypes,

)

logging.basicConfig(

    format='%(asctime)s - %(name)s - %(levelname)s - %(message)s',

    level=logging.INFO

)

logger = logging.getLogger(\_\_name\_\_)

TOKEN = '\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*'

async def translate\_text(text: str, target\_lang: str) -> str:

    url = "https://translate.googleapis.com/translate\_a/single"

    params = {

        "client": "gtx",

        "sl": "auto",

        "tl": target\_lang,

        "dt": "t",

        "q": text,

    }

    async with httpx.AsyncClient() as client:

        response = await client.get(url, params=params)

        response.raise\_for\_status()

        print(response)

        translated\_data = response.json()

        print(translated\_data)

        translations = [translation[0] for translation in translated\_data[0]]

        return ''.join(translations)

async def start(update: Update, context: ContextTypes.DEFAULT\_TYPE) -> None:

    user = update.effective\_user

    await update.message.reply\_markdown\_v2(

        fr'Привет {user.mention\_markdown\_v2()}\! Добро пожаловать в Osmanov бот\.'

        '\nЯ могу переводить текст и не только\.\n'

        'Для получения списка команд отправте сообщение /help'

    )

async def help\_command(update: Update, context: ContextTypes.DEFAULT\_TYPE) -> None:

    await update.message.reply\_text(

        'при отправке сообщения боту, он переведет его на английский\n\n'

        'Вот список доступных команд:\n'

        '/start - Начать взаимодействие с ботом\n'

        '/help - Показать это сообщение\n'

        '/transl <текст> - Бот переведет на английский сообщение\n'

        '/reverse <текст> - Бот перевернёт ваше сообщение\n'

        '/upper <текст> - Бот преобразует текст в верхний регистр\n'

        '/lower <текст> - Бот преобразует текст в нижний регистр'

    )

async def transl(update: Update, context: ContextTypes.DEFAULT\_TYPE) -> None:

    text = update.message.text

    try:

        translated\_text = await translate\_text(text, 'en')

        await update.message.reply\_text(translated\_text)

    except httpx.HTTPStatusError as e:

        logger.error(f"Ошибка HTTP при переводе: {e}")

        await update.message.reply\_text("Не удалось перевести текст. Попробуйте позже.")

async def reverse(update: Update, context: ContextTypes.DEFAULT\_TYPE) -> None:

    args = context.args

    if not args:

        await update.message.reply\_text('Предоставьте текст для переворота. Например: /reverse Привет')

        return

    reversed\_text = ' '.join(args)[::-1]

    await update.message.reply\_text(reversed\_text)

async def upper(update: Update, context: ContextTypes.DEFAULT\_TYPE) -> None:

    args = context.args

    if not args:

        await update.message.reply\_text('Предоставьте текст для преобразования. Например: /upper Привет')

        return

    upper\_text = ' '.join(args).upper()

    await update.message.reply\_text(upper\_text)

async def lower(update: Update, context: ContextTypes.DEFAULT\_TYPE) -> None:

    args = context.args

    if not args:

        await update.message.reply\_text('Предоставьте текст для преобразования. Например: /lower Привет')

        return

    lower\_text = ' '.join(args).lower()

    await update.message.reply\_text(lower\_text)

async def error\_handler(update: object, context: ContextTypes.DEFAULT\_TYPE) -> None:

    logger.error(msg="Exception while handling an update:", exc\_info=context.error)

    if isinstance(update, Update) and update.effective\_message:

        await update.effective\_message.reply\_text('Произошла ошибка. Пожалуйста, попробуйте позже.')

def main() -> None:

    application = ApplicationBuilder().token(TOKEN).build()

    application.add\_handler(CommandHandler("start", start))

    application.add\_handler(CommandHandler("help", help\_command))

    application.add\_handler(CommandHandler("reverse", reverse))

    application.add\_handler(CommandHandler("upper", upper))

    application.add\_handler(CommandHandler("lower", lower))

    application.add\_handler(MessageHandler(filters.TEXT & ~filters.COMMAND, transl))

    application.add\_error\_handler(error\_handler)

    application.run\_polling()

if \_\_name\_\_ == '\_\_main\_\_':

    main()

**Выполнение программы**