Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс
«**Парадигмы и конструкции языков программирования**»

Отчет по лабораторной работе №5
«Разработка простого бота для Telegram с использованием языка Python»

Выполнил:
студент группы ИУ5-31Б
Санников Н.А.

Проверил:
преподаватель каф. ИУ5

Москва, 2024 г.

Описание задания

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

Текст программы

#bot_v1.py

```
from telegram import Update, InlineKeyboardButton, InlineKeyboardMarkup
from telegram.ext import Application, CommandHandler, CallbackQueryHandler, ContextTypes

TOKEN = "токен удалён в целях безопасности"

# Функция, которая вызывается, когда пользователь вводит команду /start
async def start(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:
    # создаем список с одной кнопкой
    keyboard = [
        [InlineKeyboardButton("Нажми меня!", callback_data='button_pressed')]
    ]
    # оборачиваем кнопку в InlineKeyboardMarkup для отображения пользователю
    reply_markup = InlineKeyboardMarkup(keyboard)
    # отправляем сообщение с кнопкой пользователю
    await update.message.reply_text('Привет! Это мой первый бот с кнопками.',
reply_markup=reply_markup)

# функция, которая вызывается, когда пользователь нажимает на кнопку
async def button_handler(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:
    # получаем инфу о нажатии на кнопку
    query = update.callback_query
    # штука для тг, чтобы не было ошибок
    await query.answer()
    # меняем прошлое сообщение на нужный текст при нажатии на кнопку
    await query.edit_message_text(text="Вы нажали на кнопку!")

def main() -> None:
    # инициализация бота
    application = Application.builder().token(TOKEN).build()

    # Регистрируем обработчик команды /start
    application.add_handler(CommandHandler("start", start))
    # Регистрируем обработчик нажатия на кнопку
    application.add_handler(CallbackQueryHandler(button_handler))

    # Запускаем бота и начинаем polling (циклично докапываться до тг, есть ли новые
сообщения?)
    application.run_polling()

if __name__ == '__main__':
    main()
```

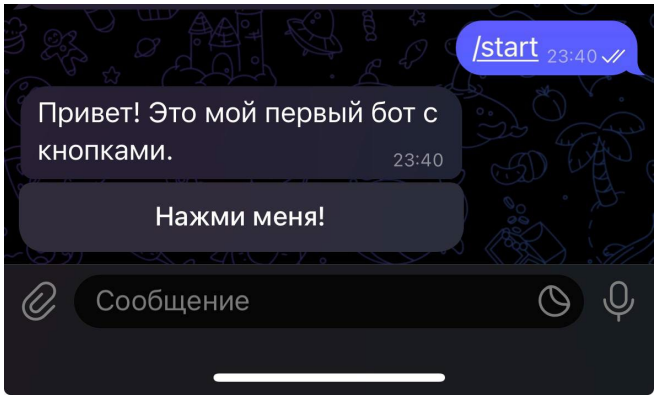
#bot_v2.py

```
from telegram import Update, InlineKeyboardButton, InlineKeyboardMarkup
from telegram.ext import Application, CommandHandler, CallbackQueryHandler, ContextTypes

TOKEN = "токен удалён в целях безопасности"

# Функция, которая вызывается, когда пользователь вводит команду /start
async def start(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:
    # создаем список с одной кнопкой 1_1
    keyboard = [
        [InlineKeyboardButton("Нажми меня!", callback_data='button_pressed')]
    ]
    # оборачиваем кнопку в InlineKeyboardMarkup для отображения пользователю
    reply_markup = InlineKeyboardMarkup(keyboard)
    # отправляем сообщение с кнопкой 1_1 пользователю
    await update.message.reply_text('Привет! Это мой первый бот с кнопками.',
reply_markup=reply_markup)

# функция, которая вызывается, когда пользователь нажимает на кнопку 1_1
async def button_handler(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:
    # получаем инфу о нажатии на кнопку 1_1
    query = update.callback_query
    # штука для тг, чтобы не было ошибок
    await query.answer()

    # создаем новый список кнопок
    new_keyboard = [
        [InlineKeyboardButton("Новая кнопка 1", callback_data='new_button_1')],
        [InlineKeyboardButton("Новая кнопка 2", callback_data='new_button_2')]
    ]
    # оборачиваем новый список в InlineKeyboardMarkup
    reply_markup = InlineKeyboardMarkup(new_keyboard)
    # заменяем сообщение на новое с другим списком кнопок
    await query.edit_message_text(text="Выберите новую кнопку:",
reply_markup=reply_markup)

# функция для обработки нажатия на кнопку 2_1
async def new_button_1_handler(update: Update, context: ContextTypes.DEFAULT_TYPE) ->
None:
    query = update.callback_query
    await query.answer()
    await query.edit_message_text(text="Вы нажали на Новую кнопку 1!")

# функция для обработки нажатия на кнопку 2_2
async def new_button_2_handler(update: Update, context: ContextTypes.DEFAULT_TYPE) ->
None:
    query = update.callback_query
    await query.answer()
    await query.edit_message_text(text="Вы нажали на Новую кнопку 2!")

def main() -> None:
    # инициализация бота
```

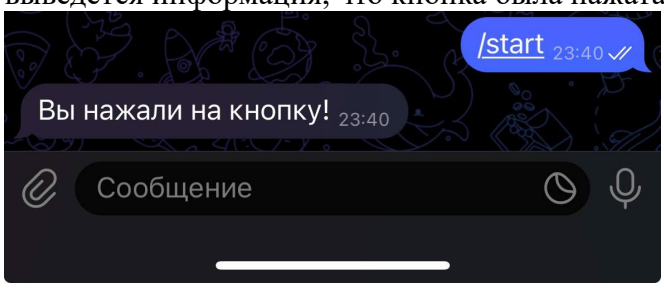
```
    application = Application.builder().token(TOKEN).build()

    # Регистрируем обработчик команды /start
    application.add_handler(CommandHandler("start", start))
    # Регистрируем обработчик нажатия на кнопку 1_1
    application.add_handler(CallbackQueryHandler(button_handler, pattern='button_pressed'))
    # Регистрируем обработчик нажатия на кнопку 2_1
    application.add_handler(CallbackQueryHandler(new_button_1_handler,
pattern='new_button_1'))
    # Регистрируем обработчик нажатия на кнопку 2_2
    application.add_handler(CallbackQueryHandler(new_button_2_handler,
pattern='new_button_2'))

    # Запускаем бота и начинаем polling (циклично докапываться до тг, есть ли новые
сообщения?)
    application.run_polling()

if __name__ == '__main__':
    main()
from telegram import Update, InlineKeyboardButton, InlineKeyboardMarkup
```

Экранные формы
с примерами выполнения программы

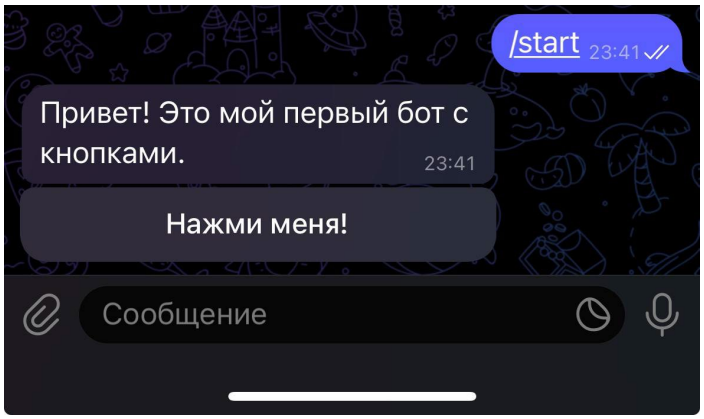
работа бота в режиме 1



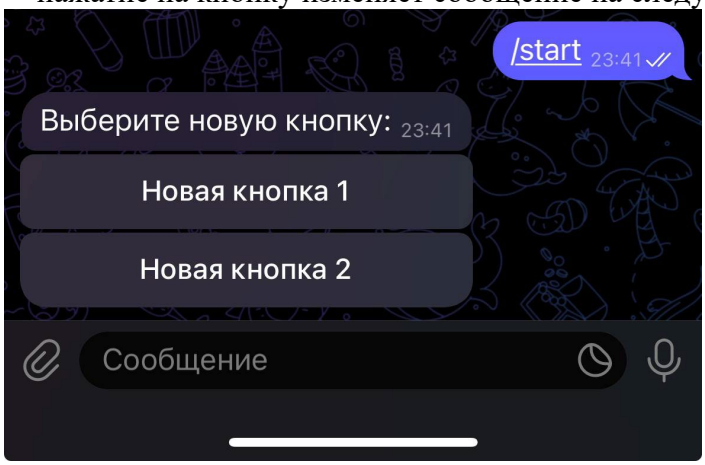
**нажатие на кнопку изменяет сообщение на следующее состояние, например здесь выведется информация, что кнопка была нажата:



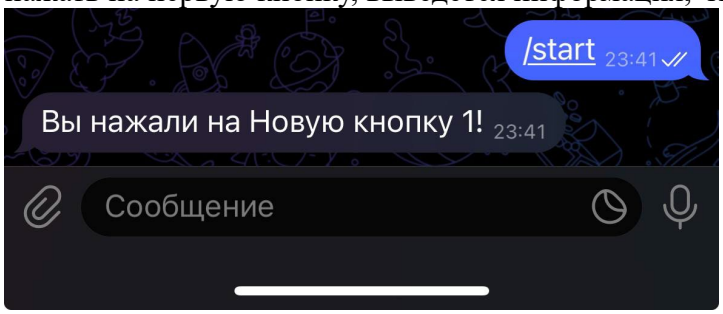
работа бота в режиме 2



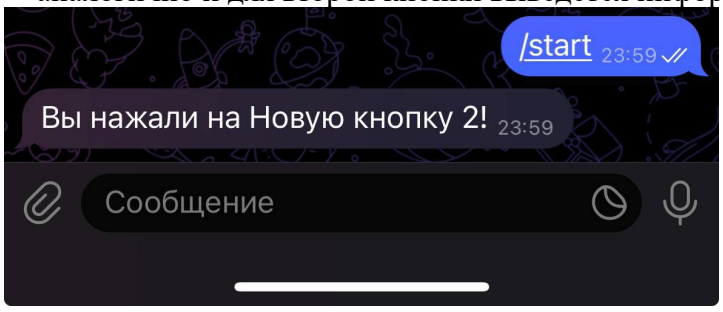
**нажатие на кнопку изменяет сообщение на следующие две кнопки:



**нажатие на любую кнопку изменяет сообщение на следующее состояние, например если нажать на первую кнопку, выведется информация, что новая кнопка 1 была нажата:



**аналогично и для второй кнопки выведется информация о нажатии новой кнопки 2:



Заключение

В заключение хочу уточнить, что бот, показанный выше, не имеет практической пользы на данном этапе разработки, а служит лишь «фундаментом», на котором можно реализовать необходимую функциональность.

В ходе выполнения данной лабораторной работы были приобретены базовые навыки разработки Telegram-ботов с использованием языка Python и библиотеки python-telegram-bot. Бот был успешно реализован с использованием кнопок, что позволяет пользователю взаимодействовать с ним через интерфейс. Также был продемонстрирован механизм обработки нажатий на кнопки и изменения сообщений в ответ на действия пользователя.

Разработка подобных ботов может быть полезна для создания различных автоматизированных решений, таких как опросы, навигация по информации, а также для интеграции с внешними системами. Полученные знания могут служить основой для создания более сложных ботов, предоставляющих дополнительные возможности и удобные интерфейсы для взаимодействия с пользователем.