**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 №6

# «Разработка бота на основе конечного автомата для Telegram с использованием языка Python»

Вариант 8

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-35Б |  |  |
| Костерин А.С. |  |  |
| Подпись и дата: |  | Подпись и дата: |

Москва, 2024 г

**Постановка задачи:**

**Цель лабораторной работы:** изучение разработки ботов в Telegram.

### Задание:

1. Разработайте бота для Telegram. Бот должен реализовывать конечный автомат из трех состояний.

**Выполнение:**

Файл – bot.py

import os

from telegram import Update, InlineKeyboardButton, InlineKeyboardMarkup

from telegram.ext import ApplicationBuilder, CommandHandler, CallbackQueryHandler, ConversationHandler

from dotenv import load\_dotenv

load\_dotenv()

TOKEN = os.getenv('TELEGRAM\_BOT\_TOKEN')

STATE1, STATE2, STATE3 = range(3)

async def start(update: Update, context):

    keyboard = [

        [InlineKeyboardButton("Перейти в состояние 1", callback\_data=str(STATE1))],

        [InlineKeyboardButton("Перейти в состояние 2", callback\_data=str(STATE2))],

        [InlineKeyboardButton("Перейти в состояние 3", callback\_data=str(STATE3))],

    ]

    reply\_markup = InlineKeyboardMarkup(keyboard)

    await update.message.reply\_text('Выберите состояние:', reply\_markup=reply\_markup)

    return STATE1

async def state1(update: Update, context):

    query = update.callback\_query

    await query.answer()

    keyboard = [

        [InlineKeyboardButton("Перейти в состояние 2", callback\_data=str(STATE2))],

        [InlineKeyboardButton("Перейти в состояние 3", callback\_data=str(STATE3))],

    ]

    reply\_markup = InlineKeyboardMarkup(keyboard)

    await query.edit\_message\_text(text="Вы в состоянии 1. Выберите следующее состояние:", reply\_markup=reply\_markup)

    return STATE1

async def state2(update: Update, context):

    query = update.callback\_query

    await query.answer()

    keyboard = [

        [InlineKeyboardButton("Перейти в состояние 1", callback\_data=str(STATE1))],

        [InlineKeyboardButton("Перейти в состояние 3", callback\_data=str(STATE3))],

    ]

    reply\_markup = InlineKeyboardMarkup(keyboard)

    await query.edit\_message\_text(text="Вы в состоянии 2. Выберите следующее состояние:", reply\_markup=reply\_markup)

    return STATE2

async def state3(update: Update, context):

    query = update.callback\_query

    await query.answer()

    keyboard = [

        [InlineKeyboardButton("Перейти в состояние 1", callback\_data=str(STATE1))],

        [InlineKeyboardButton("Перейти в состояние 2", callback\_data=str(STATE2))],

    ]

    reply\_markup = InlineKeyboardMarkup(keyboard)

    await query.edit\_message\_text(text="Вы в состоянии 3. Выберите следующее состояние:", reply\_markup=reply\_markup)

    return STATE3

async def cancel(update: Update, context):

    await update.message.reply\_text("Диалог завершен.")

    return ConversationHandler.END

def main():

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

    conv\_handler = ConversationHandler(

        entry\_points=[CommandHandler('start', start)],

        states={

            STATE1: [CallbackQueryHandler(state1)],

            STATE2: [CallbackQueryHandler(state2)],

            STATE3: [CallbackQueryHandler(state3)],

        },

        fallbacks=[CommandHandler('cancel', cancel)]

    )

    application.add\_handler(conv\_handler)

    application.run\_polling()

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

    main()

Пример выполнения:



