Московский государственный технический университет им. Н.Э. Баумана

Отчет

Лабораторная работа № 7

ИСПОЛНИТЕЛЬ:

ФИО: Митрохина А.А.

Группа: ИБМ3-34Б

ПРЕПОДАВАТЕЛЬ:

Гапанюк Ю.Е.

Описание задания

Цель: Разработать Telegram-бота на Python, который управляет поведением пользователя, используя концепцию **конечного автомата (FSM)**. Бот должен последовательно переводить пользователя между тремя различными состояниями, запрашивая определенные данные на каждом этапе.

Состояния FSM:

1.

WAITING_FOR_NAME (Ожидание имени): Начальное состояние после /start. Бот просит пользователя ввести имя.

2.
3.

WAITING_FOR_AGE (Ожидание возраста): Переход после получения имени. Бот просит пользователя ввести возраст.

4.
5.

FINISHED (Завершено): Переход после получения возраста. Бот выводит собранную информацию и сбрасывает состояние.

6.

Инструмент: Библиотека pyTelegramBotAPI (telebot).

Текст программы

Для хранения состояний пользователей будем использовать простой словарь user_states.

1. Подготовка

Установка библиотеки:

Bash

pip install pyTelegramBotAPI

Получение токена: Замените 'ВАШ_ТОКЕН_БОТА' на ваш токен, полученный от BotFather.

2. Код программы (fsm_bot.py)

Python

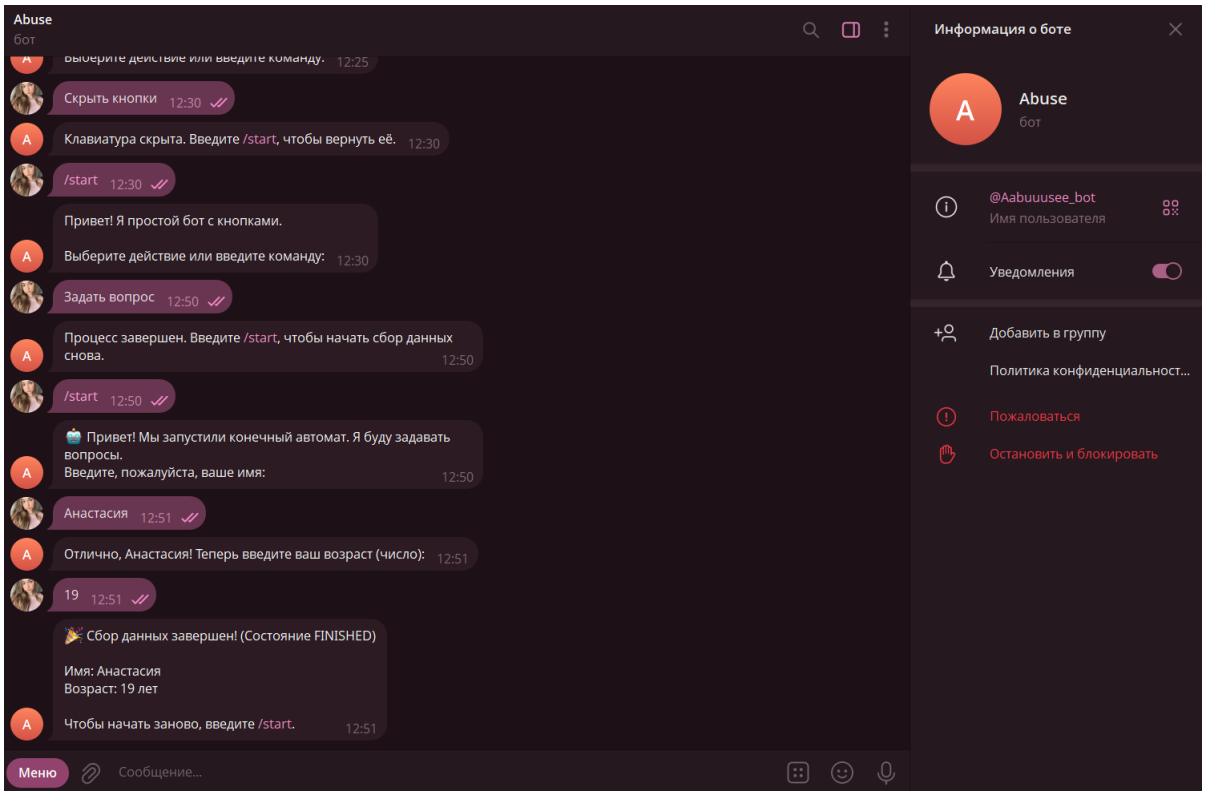
```
import telebot
from telebot import types
from telebot.handler_backends import State, StatesGroup  # Для структурирования состояний

# --- ВАШ ТОКЕН БОТА ---
TOKEN = '8370052481:AAGaeRJmP7tyrf5QSw-xxaBY8rEpZPbLvAU'

# Инициализация бота
bot = telebot.TeleBot(TOKEN)


# --- Менеджмент состояний (FSM) ---

# 1. Определяем класс для хранения состояний пользователя
class UserStates(StatesGroup):
    """Класс, описывающий три возможных состояния конечного автомата."""
    waiting_for_name = State()  # Состояние 1: Ожидание ввода имени
    waiting_for_age = State()  # Состояние 2: Ожидание ввода возраста
    finished = State()  # Состояние 3: Завершение сбора данных (и сброс)


# 2. Словарь для хранения данных и текущего состояния каждого пользователя
user_data = {}
user_state = {}  # Хранит ID чата и текущее состояние


# --- 1. Обработка команды /start ---
@bot.message_handler(commands=['start'])
def handle_start(message):
    chat_id = message.chat.id

    # Инициализация или сброс данных пользователя
    user_data[chat_id] = {}
```

```
    user_state[chat_id] = UserStates.waiting_for_name.name  # Установка начального состояния

    # Отправка первого запроса
    bot.send_message(
        chat_id,
        "□ Привет! Мы запустили конечный автомат. Я буду задавать вопросы.\n"
        "**Введите, пожалуйста, ваше имя:**",
        parse_mode='Markdown'
    )


# --- 2. Основная обработка текстовых сообщений (логика переходов) ---
@bot.message_handler(content_types=['text'])
def handle_text_input(message):
    chat_id = message.chat.id
    current_state = user_state.get(chat_id)

    # --- Состояние 1: WAITING_FOR_NAME ---
    if current_state == UserStates.waiting_for_name.name:
        name = message.text

        # Сохраняем имя и переходим в следующее состояние
        user_data[chat_id]['name'] = name
        user_state[chat_id] = UserStates.waiting_for_age.name

        # Запрашиваем следующее поле
        bot.send_message(
            chat_id,
            f"Отлично, {name}! Теперь **введите ваш возраст** (число):",
            parse_mode='Markdown'
        )

    # --- Состояние 2: WAITING_FOR_AGE ---
    elif current_state == UserStates.waiting_for_age.name:
        age_text = message.text

        # Проверка на корректность ввода (должно быть числом)
        if not age_text.isdigit():
            bot.send_message(chat_id, "❌Некорректный ввод. Возраст
```

```
должен быть числом. Попробуйте снова.")
            return  # Остаемся в текущем состоянии

        age = int(age_text)

        # Сохраняем возраст и переходим в финальное состояние
        user_data[chat_id]['age'] = age
        user_state[chat_id] = UserStates.finished.name

        # Вывод результатов
        show_results(chat_id)

    # --- Состояние 3: FINISHED или Неизвестное состояние ---
    elif current_state == UserStates.finished.name or current_state is None:
        # Предлагаем начать сначала
        bot.send_message(
            chat_id,
            "Процесс завершен. Введите /start, чтобы начать сбор данных
снова."
        )


# --- Функция вывода результатов и сброса FSM ---
def show_results(chat_id):
    data = user_data.get(chat_id, {})

    if data:
        response = (
            "□ **Сбор данных завершен! (Состояние FINISHED)**\n\n"
            f"Имя: {data.get('name')}\n"
            f"Возраст: {data.get('age')} лет\n\n"
            "Чтобы начать заново, введите /start."
        )
        bot.send_message(chat_id, response, parse_mode='Markdown')

    # Сброс состояния после завершения
    user_state[chat_id] = None


# --- Запуск бота ---
if __name__ == '__main__':
    print("□ FSM Бот запущен. Нажмите Ctrl+C для остановки.")
    try:
```

```
        # Устанавливаем команду help, чтобы избежать ошибок при
неизвестных командах
        bot.set_my_commands([
            types.BotCommand("/start", "Начать работу FSM"),
        ])
        # Запуск цикла прослушивания сообщений
        bot.infinity_polling()
    except Exception as e:
        print(f"Ошибка при запуске бота: {e}")
```

Экранные формы с примерами выполнения программы



1. Консоль (Запуск)

```
C:\Users\Admin\PycharmProjects\PythonProject3\.venv\Scripts\python.exe C:\Users\Admin\PycharmProjects\PythonProject3\main.py
🤖 FSM Бот запущен. Нажмите Ctrl+C для остановки.
```

Пример 1: Последовательный переход между состояниями

Пользователь проходит весь цикл FSM, корректно вводя данные.

Шаг	**Ввод пользователя**	**Состояние FSM**	**Ответ бота**
1.	/start	**WAITING_FOR_NAME**	Бот просит ввести имя.
2.	Алексей	**WAITING_FOR_AGE**	Бот просит ввести возраст.
3.	29	**FINISHED**	Бот выводит финальный результат.

Пример 2: Обработка некорректного ввода (остается в том же состоянии)

Пользователь вводит не число в состоянии WAITING_FOR_AGE.

Шаг	**Ввод пользователя**	**Состояние FSM**	**Ответ бота**
1.	/start	WAITING_FOR_NAME	Бот просит ввести имя.
2.	Мария	WAITING_FOR_AGE	Бот просит ввести возраст.
3.	Двадцать	**WAITING_FOR_AGE**	□ Некорректный ввод. Возраст должен быть числом. Попробуйте снова.
4.	35	FINISHED	Бот выводит финальный результат.

Вывод:

Разработанный Telegram-бот успешно реализует логику **конечного автомата** с тремя состояниями (WAITING_FOR_NAME, WAITING_FOR_AGE, FINISHED). Бот корректно управляет последовательностью ввода данных от пользователя и проверяет их тип, обеспечивая устойчивый переход между состояниями.