|  |  |
| --- | --- |
|  | Министерство науки и высшего образования Российской Федерации  Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»  (МГТУ им. Н.Э. Баумана) |
| ФАКУЛЬТЕТ «ИНЖЕНЕРНЫЙ БИЗНЕС И МЕНЕДЖМЕНТ»  КАФЕДРА «ПРОМЫШЛЕННАЯ ЛОГИСТИКА» (ИБМ-3)  Лабораторные работы  «Парадигмы и конструкции языков программирования»  38.03.05 «Бизнес-информатика» (бакалавриат)  Студент ИБМ3-34Б Сергеева М. А.  (Подпись, дата)    2024 г. | |

**Лабораторная работа № 7**

**Тема:** Разработка Telegram-бота с конечным автоматом из трех состояний

**Цель работы**

Разработать Telegram-бота с использованием конечного автомата (FSM) на языке программирования Python. Бот должен переходить между тремя состояниями в зависимости от ввода пользователя.

**Ход выполнения работы**

**1. Подготовка среды**

* Установлен Python (версия 3.8+).
* Установлены необходимые библиотеки:

pip install aiogram

* Получен токен Telegram-бота через @BotFather.

**2. Разработка бота**

* Создан Python-скрипт и подключена библиотека aiogram.
* Определена модель конечного автомата с тремя состояниями:
  1. **Ожидание ввода (waiting\_for\_input)** – бот ждет, когда пользователь введет текст.
  2. **Обработка данных (processing\_data)** – бот выполняет обработку введенного текста.
  3. **Возврат в начальное состояние** – бот завершает работу и предлагает ввести новое сообщение.
* Реализованы переходы между состояниями с помощью FSMContext.
* Добавлены обработчики команд /start и ввода текста.
* Реализована проверка, чтобы бот не ломался при отправке стикеров, фото и других типов сообщений.

**3. Тестирование бота**

* Проверены все состояния и переходы.
* Улучшен ответ бота для обработки неожиданных сообщений.

**Вывод**

В ходе работы был разработан Telegram-бот с конечным автоматом, который корректно обрабатывает пользовательский ввод. Реализованы три состояния, обеспечены плавные переходы между ними. Код соответствует требованиям задания и успешно протестирован.

**Приложение**

Код бота прилагается к отчету.

|  |
| --- |
| # ============================================================= # TELEGRAM BOT ver 2 # =============================================================  # token = ""  from aiogram import Bot, Dispatcher, types, F from aiogram.types import Message from aiogram.fsm.context import FSMContext from aiogram.fsm.state import State, StatesGroup from aiogram.fsm.storage.memory import MemoryStorage from aiogram.filters import Command import asyncio import logging  TOKEN = "" # здесь должен быть токен  # Настройка логирования logging.basicConfig(level=logging.INFO)  # Создаем бота и диспетчер bot = Bot(token=TOKEN) storage = MemoryStorage() dp = Dispatcher(storage=storage)   # Определяем конечный автомат с тремя состояниями class Form(StatesGroup):  waiting\_for\_input = State() # Ожидание ввода  processing\_data = State() # Обработка данных   # Обработчик команды /start @dp.message(Command("start")) async def start\_command(message: Message, state: FSMContext):  await state.set\_state(Form.waiting\_for\_input)  await message.answer("Привет! Введи любое слово, и я обработаю его!")   # Ожидание ввода от пользователя @dp.message(Form.waiting\_for\_input) async def get\_user\_input(message: Message, state: FSMContext):  user\_text = message.text  await state.update\_data(user\_text=user\_text)  await state.set\_state(Form.processing\_data)  await message.answer("Спасибо! Я обрабатываю твои данные...")   # Переход в следующее состояние  await process\_data(message, state)   # Обработка данных и возврат в начальное состояние async def process\_data(message: Message, state: FSMContext):  data = await state.get\_data()  user\_text = data.get("user\_text", "")  response = f"Ты ввел: {user\_text.upper()}"   await message.answer(response)  await state.clear() # Завершаем состояние  await message.answer("Можешь ввести новое слово или снова написать /start.")   # Реакция на ввод непонятного сообщения @dp.message() async def unknown\_message(message: Message):  await message.answer("Я не понимаю тебя. Напиши /start, чтобы начать.")  # Реакция на картинку, стикер или голосовое сообщение @dp.message(Form.waiting\_for\_input, F.text) async def get\_user\_input(message: Message, state: FSMContext):  user\_text = message.text  if not user\_text: # Проверяем, что это именно текст  await message.answer("Отправь мне текстовое сообщение.")  return   await state.update\_data(user\_text=user\_text)  await state.set\_state(Form.processing\_data)  await message.answer("Спасибо! Я обрабатываю твои данные...")   await process\_data(message, state)   # Функция запуска бота async def main():  await dp.start\_polling(bot)   if \_\_name\_\_ == "\_\_main\_\_":  asyncio.run(main()) |