**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 «Информатика и системы управления»**

**Кафедра ИУ5 «Системы обработки информации и управления»**

Лабораторная работа №6

по дисциплине «Базовые компоненты интернет-технологий»

Выполнил:

студент группы ИУ5-33Б

Некрасов С. А.

Проверил:

Канев А.И.

2021 г.

# **Задание:**

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

# **Текст программы:**

## **Файл common.py:**

from aiogram import Dispatcher, types  
from aiogram.dispatcher import FSMContext  
from aiogram.dispatcher.filters import Text  
  
  
async def cmd\_start(message: types.Message, state: FSMContext):  
 await state.finish()  
 await message.answer(  
 "Купить билет /ticket",  
 reply\_markup=types.ReplyKeyboardRemove()  
 )  
  
  
async def cmd\_cancel(message: types.Message, state: FSMContext):  
 await state.finish()  
 await message.answer("Действие отменено", reply\_markup=types.ReplyKeyboardRemove())  
  
  
  
def register\_handlers\_common(dp: Dispatcher, admin\_id: int):  
 dp.register\_message\_handler(cmd\_start, commands="start", state="\*")  
 dp.register\_message\_handler(cmd\_cancel, commands="cancel", state="\*")  
 dp.register\_message\_handler(cmd\_cancel, Text(equals="отмена", ignore\_case=True), state="\*")

## **Файл ticket.py:**

from aiogram import Dispatcher, types  
from aiogram.dispatcher import FSMContext  
from aiogram.dispatcher.filters.state import State, StatesGroup  
  
available\_city\_name = ["Анапа", "Сочи", "Ялта"]  
available\_day\_quantity = ["5", "7", "9"]  
available\_trip\_method = ["самолёт", "поезд", "яхта"]  
  
  
class OrderTicket(StatesGroup):  
 waiting\_for\_city\_name = State()  
 waiting\_for\_day\_quantity = State()  
 waiting\_for\_trip\_method = State()  
  
  
async def ticket\_start(message: types.Message):  
 keyboard = types.ReplyKeyboardMarkup(resize\_keyboard=True)  
 for name in available\_city\_name:  
 keyboard.add(name)  
 await message.answer("Выберите город:", reply\_markup=keyboard)  
 await OrderTicket.waiting\_for\_city\_name.set()  
  
  
  
async def ticket\_chosen(message: types.Message, state: FSMContext):  
 if message.text not in available\_city\_name:  
 await message.answer("В данный город нет билетов, выберите город, используя клавиатуру ниже")  
 return  
 await state.update\_data(chosen\_ticket=message.text)  
  
 keyboard = types.ReplyKeyboardMarkup(resize\_keyboard=True)  
 for quantity in available\_day\_quantity:  
 keyboard.add(quantity)  
 await OrderTicket.next()  
 await message.answer("Теперь выберите количество дней:", reply\_markup=keyboard)  
  
  
async def ticket\_quantity\_chosen(message: types.Message, state: FSMContext):  
 if message.text not in available\_day\_quantity:  
 await message.answer(  
 "На такое количество дней нет билетов, выберите количество дней, используя клавиатуру ниже")  
 return  
 await state.update\_data(chosen\_days=message.text)  
  
 keyboard = types.ReplyKeyboardMarkup(resize\_keyboard=True)  
 for method in available\_trip\_method:  
 keyboard.add(method)  
 await OrderTicket.next()  
 await message.answer("Теперь выберите способ поездки:", reply\_markup=keyboard)  
  
  
async def ticket\_method\_chosen(message: types.Message, state: FSMContext):  
 if message.text not in available\_trip\_method:  
 await message.answer("Такого способа поездки нет, выберите способ поездки, используя клавиатуру ниже")  
 return  
 user\_data = await state.get\_data()  
 await message.answer(f"Вы отдохнёте {user\_data['chosen\_days']} дней в городе {user\_data['chosen\_ticket']}. "  
 f"Ваш способ поездки: {message.text}\n",  
 reply\_markup=types.ReplyKeyboardRemove())  
 await state.finish()  
  
  
def register\_handlers\_ticket(dp: Dispatcher):  
 dp.register\_message\_handler(ticket\_start, commands="ticket", state="\*")  
 dp.register\_message\_handler(ticket\_chosen, state=OrderTicket.waiting\_for\_city\_name)  
 dp.register\_message\_handler(ticket\_quantity\_chosen, state=OrderTicket.waiting\_for\_day\_quantity)  
 dp.register\_message\_handler(ticket\_method\_chosen, state=OrderTicket.waiting\_for\_trip\_method)

## **Файл config.py:**

import configparser  
from dataclasses import dataclass  
  
  
@dataclass  
class TgBot:  
 token: str  
 admin\_id: int  
  
  
@dataclass  
class Config:  
 tg\_bot: TgBot  
  
  
def load\_config(path: str):  
 config = configparser.ConfigParser()  
 config.read(path)  
  
 tg\_bot = config["tg\_bot"]  
  
 return Config(  
 tg\_bot=TgBot(  
 token=tg\_bot["token"],  
 admin\_id=int(tg\_bot["admin\_id"])  
 )  
 )

## **Файл bot.ini:**

[tg\_bot]  
token = 5003047874:AAHmH7rfUJmXdg5vyrDnaddK1fUNSxBNbac  
admin\_id = 987654321

## **Файл bot.py:**

import asyncio  
import logging  
  
from aiogram import Bot, Dispatcher  
from aiogram.types import BotCommand  
from aiogram.contrib.fsm\_storage.memory import MemoryStorage  
  
from app.config import load\_config  
from app.handlers.ticket import register\_handlers\_ticket  
from app.handlers.common import register\_handlers\_common  
  
logger = logging.getLogger(\_\_name\_\_)  
  
  
async def set\_commands(bot: Bot):  
 commands = [  
 BotCommand(command="/ticket", description="Купить билет"),  
 BotCommand(command="/cancel", description="Отменить текущее действие")  
 ]  
 await bot.set\_my\_commands(commands)  
  
  
async def main():  
 # Настройка логирования в stdout  
 logging.basicConfig(  
 level=logging.INFO,  
 format="%(asctime)s - %(levelname)s - %(name)s - %(message)s",  
 )  
 logger.error("Starting bot")  
  
 # Парсинг файла конфигурации  
 config = load\_config("config/bot.ini")  
  
 # Объявление и инициализация объектов бота и диспетчера  
 bot = Bot(token=config.tg\_bot.token)  
 dp = Dispatcher(bot, storage=MemoryStorage())  
  
 # Регистрация хэндлеров  
 register\_handlers\_common(dp, config.tg\_bot.admin\_id)  
 register\_handlers\_ticket(dp)  
  
 # Установка команд бота  
 await set\_commands(bot)  
  
 # Запуск поллинга  
 # await dp.skip\_updates() # пропуск накопившихся апдейтов (необязательно)  
 await dp.start\_polling()  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 asyncio.run(main())

## **requirements**

aiogram==2.17.1

# **Экранные формы с примерами выполнения программы:**

