



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»

ДИСЦИПЛИНА:

«БКИТ»

Лабораторная работа № 5–6

Студент Распашнов А.А. ИУ5Ц-54Б

(И.О. Фамилия) (Группа)

(Подпись, дата)

Преподаватель Гапанюк Ю.Е.

(И.О. Фамилия)

(Подпись, дата)

СОДЕРЖАНИЕ:

Лабораторная работа №5	3
Лабораторная работа №6	3
ЛР.5 - код.....	4
ЛР.5- результат:	6
ЛР.6 – Код:	8
ЛР.6 – результат.....	10

Лабораторная работа №5

Задание:

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

Лабораторная работа №6

Задание:

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

ЛР.5 - код

```
import logging
import time
import random
from sys import exit
from aiogram.types import *
from aiogram import Bot, Dispatcher, executor, types
from aiogram.contrib.middlewares.logging import LoggingMiddleware

TOKEN = '5012836949:AAHzd5kE4JMgdzc4CyDk0nRdmK5d258mvjw'
if not TOKEN:
    exit('Error: no token provided')

logging.basicConfig(level=logging.INFO,
                    format="%(asctime)s - %(levelname)s - %(name)s - %(message)s")

# Initialize bot and dispatcher
bot = Bot(token=TOKEN)
dp = Dispatcher(bot)
dp.middleware.setup(LoggingMiddleware())

@dp.message_handler(commands=['start'])
async def send_welcome(message: types.Message):
    """
    This handler will be called when user sends `/start` command
    """
    keyboard = types.ReplyKeyboardMarkup(resize_keyboard=True)
    keyboard.add('🕒 Узнать время')
    keyboard.add('🎲 Сыграть в игру')
    await message.answer("Привет, я тестовый бот реализующий функционал кнопок. \n"
                        "Я могу показать вам время и \"бросить монетку\". \n"
                        "Используйте кнопки снизу: ", reply_markup=keyboard)

@dp.message_handler(commands=['help'])
async def send_help(message: types.Message):
    """
    This handler will be called when user sends `/help` command
    :param message:
    :return:
    """
    await message.answer("Help here!")

@dp.message_handler(lambda message: message.text == '🕒 Узнать время')
async def show_time(message: types.Message):
    await message.answer(time.strftime('Сейчас %H:%M'))

@dp.message_handler(lambda message: message.text == '🎲 Сыграть в игру')
async def start_game(message: types.Message):
    keyboard = InlineKeyboardMarkup(row_width=2)

    keyboard.add(InlineKeyboardButton('Орёл', callback_data=f'Орёл'),
                  InlineKeyboardButton('Решка', callback_data=f'Решка'))
    await message.answer('Выберите сторону:', reply_markup=keyboard)
```

```

@dp.callback_query_handler(text='Орёл')
async def func5(call: types.CallbackQuery):
    # орёл == 0
    # решка == 1
    t = random.randint(0, 1)
    if t:
        await call.message.answer(f'Бот выбросил {"Решку" if t else
"Орла"}!\n'
                                f'Вы не угадали!')
    else:
        await call.message.answer(f'Бот выбросил {"Решку" if t else
"Орла"}!\n'
                                f'Вы угадали!')
    await call.answer()

@dp.callback_query_handler(text='Решка')
async def func6(call: types.CallbackQuery):
    # орёл == 0
    # решка == 1
    t = random.randint(0, 1)
    if t:
        await call.message.answer(f'Бот выбросил {"Решку" if t else
"Орла"}!\n'
                                f'Вы угадали!')
    else:
        await call.message.answer(f'Бот выбросил {"Решку" if t else
"Орла"}\n'
                                f'Вы не угадали!')
    await call.answer()

'''

@dp.callback_query_handler(text=[bond+'2' for bond in unique_names])
async def func4(call: types.CallbackQuery):
    await call.message.answer('Я вторая функция!')
    await call.answer()

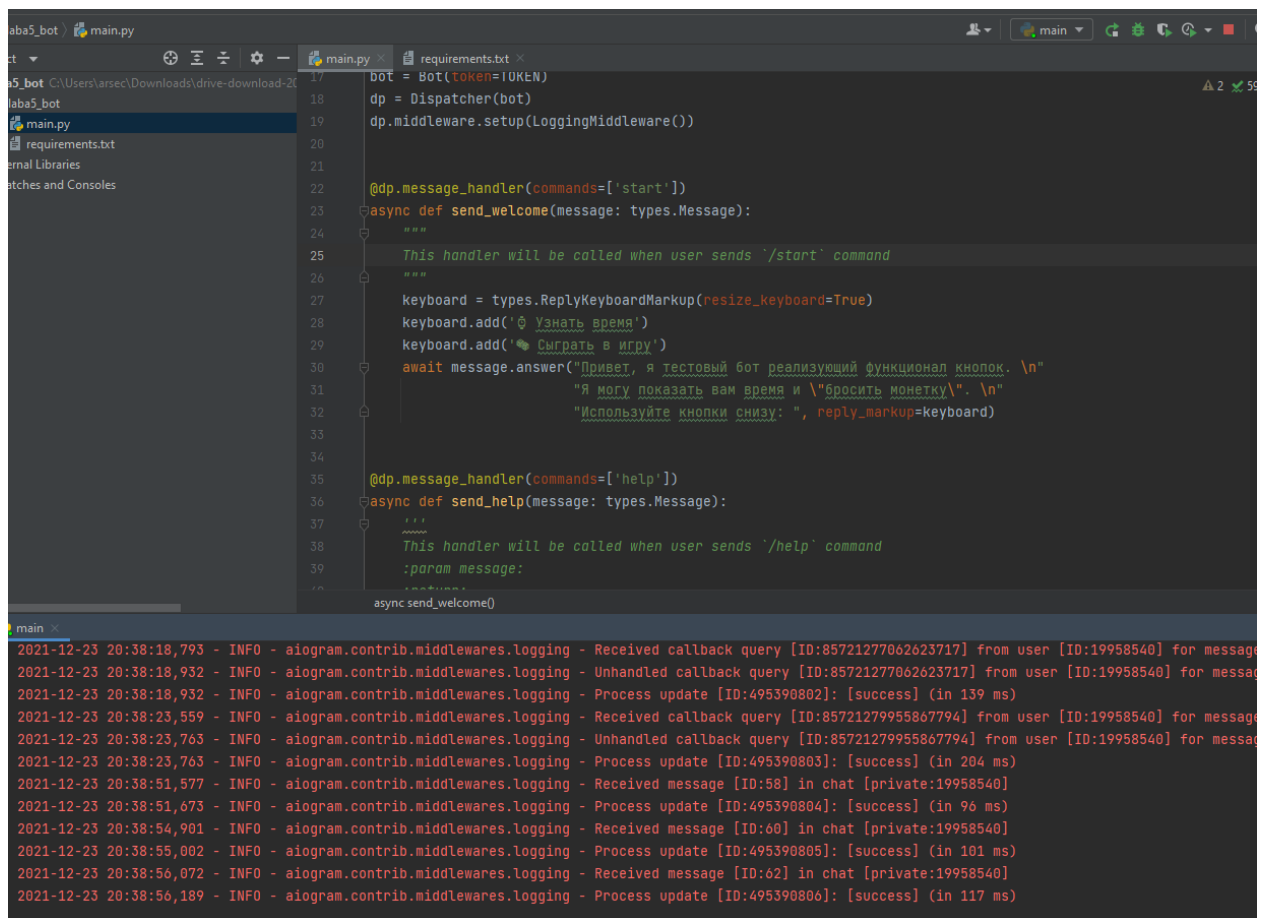
'''

@dp.message_handler()
async def nothing(message: types.Message):
    await message.answer('Я не знаю такой команды.')

if __name__ == '__main__':
    executor.start_polling(dp, skip_updates=True)

```

ЛР.5- результат:

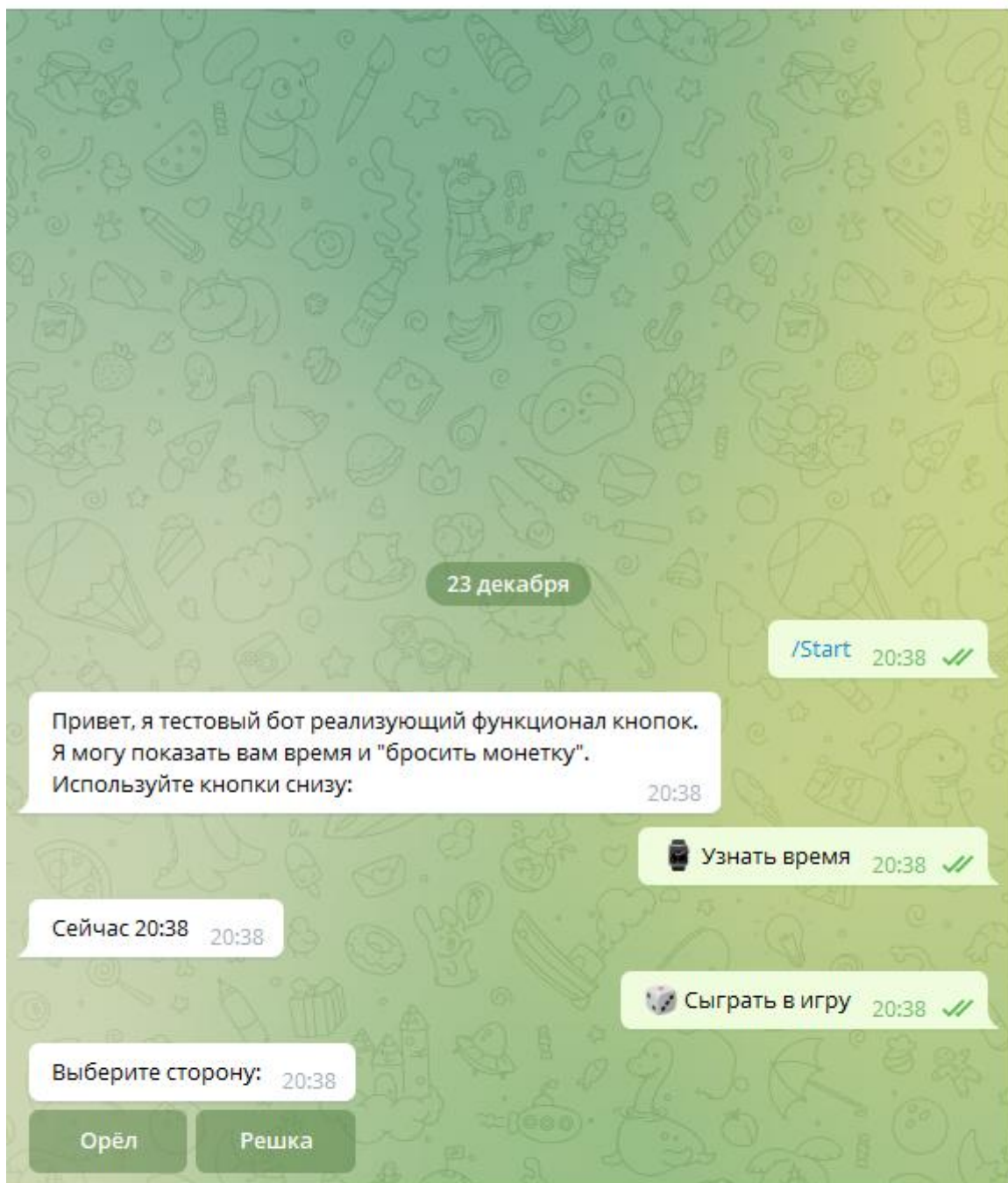


```
17 bot = Bot(token=token)
18 dp = Dispatcher(bot)
19 dp.middleware.setup(LoggingMiddleware())
20
21
22 @dp.message_handler(commands=['start'])
23 async def send_welcome(message: types.Message):
24     """
25     This handler will be called when user sends '/start' command
26     """
27     keyboard = types.ReplyKeyboardMarkup(resize_keyboard=True)
28     keyboard.add('🕒 Узнать время')
29     keyboard.add('🎮 Сыграть в игру')
30     await message.answer("Привет, я тестовый бот реализующий функционал кнопок. \n"
31                          "Я могу показать вам время и \"Бросить монетку\". \n"
32                          "Используйте кнопки снизу: ", reply_markup=keyboard)
33
34
35 @dp.message_handler(commands=['help'])
36 async def send_help(message: types.Message):
37     """
38     This handler will be called when user sends '/help' command
39     :param message:
40     """
41     await message.answer("Помощь")
42
43
44 if __name__ == '__main__':
45     loop = asyncio.get_event_loop()
46     loop.run_until_complete(dp.start())
```

```
2021-12-23 20:38:18,793 - INFO - aiogram.contrib.middlewares.logging - Received callback query [ID:85721277062623717] from user [ID:19958540] for message
2021-12-23 20:38:18,932 - INFO - aiogram.contrib.middlewares.logging - Unhandled callback query [ID:85721277062623717] from user [ID:19958540] for message
2021-12-23 20:38:18,932 - INFO - aiogram.contrib.middlewares.logging - Process update [ID:495390802]: [success] (in 139 ms)
2021-12-23 20:38:23,559 - INFO - aiogram.contrib.middlewares.logging - Received callback query [ID:85721279955867794] from user [ID:19958540] for message
2021-12-23 20:38:23,763 - INFO - aiogram.contrib.middlewares.logging - Unhandled callback query [ID:85721279955867794] from user [ID:19958540] for message
2021-12-23 20:38:23,763 - INFO - aiogram.contrib.middlewares.logging - Process update [ID:495390803]: [success] (in 204 ms)
2021-12-23 20:38:51,577 - INFO - aiogram.contrib.middlewares.logging - Received message [ID:58] in chat [private:19958540]
2021-12-23 20:38:51,673 - INFO - aiogram.contrib.middlewares.logging - Process update [ID:495390804]: [success] (in 96 ms)
2021-12-23 20:38:54,901 - INFO - aiogram.contrib.middlewares.logging - Received message [ID:60] in chat [private:19958540]
2021-12-23 20:38:55,002 - INFO - aiogram.contrib.middlewares.logging - Process update [ID:495390805]: [success] (in 101 ms)
2021-12-23 20:38:56,072 - INFO - aiogram.contrib.middlewares.logging - Received message [ID:62] in chat [private:19958540]
2021-12-23 20:38:56,189 - INFO - aiogram.contrib.middlewares.logging - Process update [ID:495390806]: [success] (in 117 ms)
```

(laba5) special-robot

бот



Написать сообщение...



Узнать время

Сыграть в игру

ЛР.6 – Код:

```
import asyncio
import logging
import types

from aiogram import Bot
from aiogram.contrib.fsm_storage.memory import MemoryStorage
from aiogram.types import BotCommand

from handlers.common import *
from handlers.food import *

class OrderFood(StatesGroup):
    waiting_for_burger_name = State()
    waiting_for_fries_name = State()
    waiting_for_drink_name = State()

async def food_start(message: types.Message):
    keyboard = types.ReplyKeyboardMarkup(resize_keyboard=True)
    for name in available_burger_names:
        keyboard.add(name)
    await message.answer("Выберите, с чем вы хотите бургер:",
reply_markup=keyboard)
    await OrderFood.waiting_for_burger_name.set()

async def food_chosen(message: types.Message, state: FSMContext):
    if message.text.lower() not in available_burger_names:
        await message.answer("Пожалуйста, выберите мясо в бургере, используя
клавиатуру ниже.")
        return
    await state.update_data(chosen_burger=message.text.lower())

    keyboard = types.ReplyKeyboardMarkup(resize_keyboard=True)
    for name in available_fries_names:
        keyboard.add(name)
    await OrderFood.next()
    await message.answer("Теперь выберите вид картошки:",
reply_markup=keyboard)

async def food_fries_chosen(message: types.Message, state: FSMContext):
    if message.text.lower() not in available_fries_names:
        await message.answer("Пожалуйста, выберите вид картошки, используя
клавиатуру ниже.")
        return
    await state.update_data(chosen_fries=message.text.lower())

    keyboard = types.ReplyKeyboardMarkup(resize_keyboard=True)
    for name in available_drink_names:
        keyboard.add(name)
    await OrderFood.next()
    await message.answer('Теперь выберите напиток:', reply_markup=keyboard)

async def food_drink_chosen(message: types.Message, state: FSMContext):
    if message.text.lower() not in available_drink_names:
        await message.answer("Пожалуйста, выберите напиток, используя
клавиатуру ниже.")
```



```

        return
    user_data = await state.get_data()
    await message.answer(f"Вы заказали бургер
{user_data['chosen_burger'] [2:]}, \n"
        f"картофель-{user_data['chosen_fries']} и напиток -
{message.text.lower()}. \n"
        f"Спасибо за ваш заказ! ",
reply_markup=types.ReplyKeyboardRemove())
    await state.finish()

def register_handlers_food(dp: Dispatcher):
    dp.register_message_handler(food_start, commands="food", state="*")
    dp.register_message_handler(food_chosen,
state=OrderFood.waiting_for_burger_name)
    dp.register_message_handler(food_fries_chosen,
state=OrderFood.waiting_for_fries_name)
    dp.register_message_handler(food_drink_chosen,
state=OrderFood.waiting_for_drink_name)

logger = logging.getLogger(__name__)

# Регистрация команд, отображаемых в интерфейсе Telegram
async def set_commands(bot: Bot):
    commands = [
        BotCommand(command="/food", description="Заказать блюда"),
        BotCommand(command="/cancel", description="Отменить текущее
действие")
    ]
    await bot.set_my_commands(commands)

async def main():
    logging.basicConfig(
        level=logging.INFO,
        format="%(asctime)s - %(levelname)s - %(name)s - %(message)s",
    )
    logger.error("Starting bot")

    TOKEN = '5093952542:AAEfnITGgt2MSjOUOlQapYGUYWqiJdY2QJY'

    # Объявление и инициализация объектов бота и диспетчера
    bot = Bot(token=TOKEN)
    dp = Dispatcher(bot, storage=MemoryStorage())

    # Регистрация хэндлеров
    register_handlers_common(dp)

    register_handlers_food(dp)

    # Установка команд бота
    await set_commands(bot)

    # Запуск поллинга
    await dp.skip_updates()
    await dp.start_polling()

if __name__ == '__main__':
    asyncio.run(main())

```

ЛР.6 – результат

