|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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 4](#_Toc91184837)

[Лабораторная работа №6 4](#_Toc91184838)

[ЛР.5 - код 5](#_Toc91184839)

[ЛР.5- результат: 7](#_Toc91184840)

[ЛР.6 – Код: 9](#_Toc91184841)

[ЛР.6 – результат 11](#_Toc91184842)

# 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:

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

# 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, c 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

# Изображение выглядит как текст Автоматически созданное описаниеИзображение выглядит как текст Автоматически созданное описаниеИзображение выглядит как текст Автоматически созданное описаниеИзображение выглядит как текст Автоматически созданное описание