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
Ахтамбаев Л.Н.

Проверил:
Канев А.И.

2021 г.

Оглавление

Постановка задачи:

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

Текст программы:

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.delivery import register_handlers_delivery
from app.handlers.cars import register_handlers_car
from app.handlers.common import register_handlers_common

logger = logging.getLogger(__name__)


async def set_commands(bot: Bot):
    commands = [
        BotCommand(command="/delivery", description="Выбор доставки"),
        BotCommand(command="/car", description="Заказ автомобиля"),
        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())

    register_handlers_delivery(dp)
    register_handlers_car(dp)
    register_handlers_common(dp, config.tg_bot.admin_id)
```

```
    # 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())
```

Файл requirements.py

```
aiogram==2.14.3
```

Файл bot.ini

```
[tg_bot]
token = 5085140066:AAGeVj0s9eFLGwn02nE90z0uo2vft6xyAro
admin_id = 484964911

# Не забудьте скопировать этот файл под именем bot.ini
```

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 delivery.py

```
from aiogram import Dispatcher, types
from aiogram.dispatcher import FSMContext
from aiogram.dispatcher.filters.state import State, StatesGroup

available_delivery_names = ["доставка курьером", "самовывоз"]
available_delivery_payment = ["наличными", "картой"]


class OrderDelivery(StatesGroup):
    waiting_for_delivery_name = State()
    waiting_for_delivery_payment = State()


async def delivery_start(message: types.Message):
    keyboard = types.ReplyKeyboardMarkup(resize_keyboard=True)
    for name in available_delivery_names:
        keyboard.add(name)
    await message.answer("Выберите способ получения заказанного автомобиля:",
reply_markup=keyboard)
    await OrderDelivery.waiting_for_delivery_name.set()


async def delivery_chosen(message: types.Message, state: FSMContext):
    if message.text.lower() not in available_delivery_names:
        await message.answer("Выберите способ доставки, используя кнопки
ниже.")
        return
    await state.update_data(chosen_delivery=message.text.lower())

    keyboard = types.ReplyKeyboardMarkup(resize_keyboard=True)
    for payment in available_delivery_payment:
        keyboard.add(payment)
    # для простых шагов можно не указывать название состояния, обходясь
next()
    await OrderDelivery.next()
    await message.answer("Теперь выберите способ оплаты:",
reply_markup=keyboard)


async def delivery_size_chosen(message: types.Message, state: FSMContext):
    if message.text.lower() not in available_delivery_payment:
        await message.answer("Выберите способ оплаты, используя кнопки
ниже.")
        return
    user_data = await state.get_data()
    await message.answer(f"Ваш способ доставки -
{user_data['chosen_delivery']}. Оплата {message.text.lower()}.\n"
                         f"Оформите заказ: /car",
reply_markup=types.ReplyKeyboardRemove())
    await state.finish()


def register_handlers_delivery(dp: Dispatcher):
    dp.register_message_handler(delivery_start, commands="delivery",
state="*")
    dp.register_message_handler(delivery_chosen,
state=OrderDelivery.waiting_for_delivery_name)
    dp.register_message_handler(delivery_size_chosen,
state=OrderDelivery.waiting_for_delivery_payment)
```

Файл cars.py

```
from aiogram import Dispatcher, types
from aiogram.dispatcher import FSMContext
from aiogram.dispatcher.filters.state import State, StatesGroup


available_cars_names = ["спорткар", "внедорожник", "минивэн"]
available_cars_equipment = ["минимальная", "стандартная", "максимальная"]


class OrderCar(StatesGroup):
    waiting_for_car_name = State()
    waiting_for_car_equipment = State()


async def car_start(message: types.Message):
    keyboard = types.ReplyKeyboardMarkup(resize_keyboard=True)
    for name in available_cars_names:
        keyboard.add(name)
    await message.answer("Выберите тип автомобиля:", reply_markup=keyboard)
    await OrderCar.waiting_for_car_name.set()


# Обратите внимание: есть второй аргумент
async def car_chosen(message: types.Message, state: FSMContext):
    if message.text.lower() not in available_cars_names:
        await message.answer("Выберите тип автомобиля, используя кнопки
ниже.")
        return
    await state.update_data(chosen_car=message.text.lower())

    keyboard = types.ReplyKeyboardMarkup(resize_keyboard=True)
    for equipment in available_cars_equipment:
        keyboard.add(equipment)
    # Для простых шагов можно не указывать название состояния, обходясь
next()
    await OrderCar.next()
    await message.answer("Выберите комплектацию:", reply_markup=keyboard)


async def car_eqp__chosen(message: types.Message, state: FSMContext):
    if message.text.lower() not in available_cars_equipment:
        await message.answer("Выберите комплектацию, используя кнопки ниже.")
        return
    user_data = await state.get_data()
    await message.answer(f"Вы заказали {user_data['chosen_car']}.
Комплектация - {message.text.lower()}\n"
                         f"Заполните форму получения заказа - /delivery",
reply_markup=types.ReplyKeyboardRemove())
    await state.finish()


def register_handlers_car(dp: Dispatcher):
    dp.register_message_handler(car_start, commands="car", state="*")
    dp.register_message_handler(car_chosen,
state=OrderCar.waiting_for_car_name)
    dp.register_message_handler(car_eqp__chosen,
state=OrderCar.waiting_for_car_equipment)
```

Файл common.py

```
from aiogram import Dispatcher, types
from aiogram.dispatcher import FSMContext
from aiogram.dispatcher.filters import Text, IDFilter


async def cmd_start(message: types.Message, state: FSMContext):
    await state.finish()
    await message.answer(
        "Выберите, с чего хотите начать: Заказа автомобия (/car) или выбор
способа доставки (/delivery).",
        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())


async def admin_command(message: types.Message):
    await message.answer("Поздравляю! Эта команда доступна только
администратору бота.")


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="*")
    dp.register_message_handler(admin_command, IDFilter(user_id=admin_id),
commands="Это команда доступна только админу")
```

Экранные формы с примерами выполнения программы: