Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждениевысшего образования «Московский государственный технический университетимени Н.Э. Баумана (национальный исследовательский университет МГТУ им. Н.Э. Баумана)

Факультет «Информатика и системы управления» Кафедра «Системы обработки информации и управления»

Лабораторная работа №7

«Разработка бота на основе конечного автомата для Telegram с использованием языка Python»

по предмету

«Базовые компоненты интернет-технологий»

Выполнил:

студент группы № ИУ5-31Б

Михалёв Ярослав

Проверил:

Преподаватель кафедры ИУ-5

Гапанюк Юрий

2022 г.

Задание

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

Текст программы

config.py

```
BOT_TOKEN = ""
ADMIN_ID = ""
```

loader.py

```
import logging

from aiogram import Bot, Dispatcher, types
from aiogram.contrib.fsm_storage.memory import MemoryStorage

from config import BOT_TOKEN


bot = Bot(token=BOT_TOKEN, parse_mode=types.ParseMode.HTML)
storage = MemoryStorage()
dp = Dispatcher(bot, storage=storage)

logging.basicConfig(format=u'%(filename)s [LINE:%(lineno)d] #%(levelname)-8s [%(asctime)s]  %(message)s',
level=logging.INFO)
```

app.py

```
from aiogram import executor
from handlers import dp

from loader import bot, storage

from utils.notify_admins import on_startup_notify


async def on_startup(dp):
  await on_startup_notify(dp)

async def on_shutdown(dp):
  await bot.close()
  await storage.close()


if __name__ == '__main__':
  executor.start_polling(dp, on_startup=on_startup_notify, on_shutdown=on_shutdown)
```

handlers/operations.py

```python
import aiogram.utils.markdown as md
from aiogram import types
from aiogram.dispatcher import FSMContext
from aiogram.dispatcher.filters import Command
from aiogram.types import Message, ContentType

from loader import dp

from states.form import Form


@dp.message_handler(Command("start"))
async def show_menu(message: Message):
    await Form.name.set()

    await message.answer(text="Привет! Давай знакомится, как тебя зовут?")


@dp.message_handler(state=Form.name)
async def process_deposit_money(message: Message, state: FSMContext):
    async with state.proxy() as data:
        data['name'] = message.text

    await Form.next()

    await message.answer("Сколько тебе лет?")


@dp.message_handler(state=Form.age)
async def on_derive_money_pressed(message: Message, state: FSMContext):
    async with state.proxy() as data:
        data['age'] = message.text

    await Form.next()

    markup = types.ReplyKeyboardMarkup(resize_keyboard=True, selective=True)
    markup.add("Я парень", "Я девушка")

    await message.reply("Теперь определимся с полом", reply_markup=markup)


@dp.message_handler(state=Form.gender)
async def process_gender(message: types.Message, state: FSMContext):
    async with state.proxy() as data:
        data['gender'] = message.text

    types.ReplyKeyboardRemove()

    await Form.next()

    markup = types.ReplyKeyboardMarkup(resize_keyboard=True, selective=True)
    markup.add("Парни", "Девушки")

    await message.answer("Кто тебе интересен?", reply_markup=markup)


@dp.message_handler(state=Form.interestGender)
async def process_gender(message: types.Message, state: FSMContext):
    async with state.proxy() as data:
        data['interestGender'] = message.text

    markup = types.ReplyKeyboardRemove()
```

```
    await Form.next()

    await message.answer("Из какого ты города?", reply_markup=markup)


@dp.message_handler(state=Form.city)
async def process_gender(message: types.Message, state: FSMContext):
    async with state.proxy() as data:
        data['city'] = message.text

    await Form.next()

    await message.answer("Расскажите о себе")


@dp.message_handler(state=Form.description)
async def process_gender(message: types.Message, state: FSMContext):
    async with state.proxy() as data:
        data['description'] = message.text

    await Form.next()

    await message.answer("Теперь пришли своё фото")


@dp.message_handler(content_types=ContentType.PHOTO, state=Form.photo)
async def send_photo_file_id(message: types.Message, state: FSMContext):
    async with state.proxy() as data:
        data['photo_id'] = message.photo[-1].file_id

    await message.answer_photo(
        photo=data['photo_id'],
        caption=md.text(
            md.text('Так выглядит твоя анкета:'),
            md.text(""),
            md.text(data['name'] + ", " + data['age'] + ", " + data['city']),
            md.text(data['description']),
            sep='\n',
        )
    )

    await state.finish()
```

states/form.py

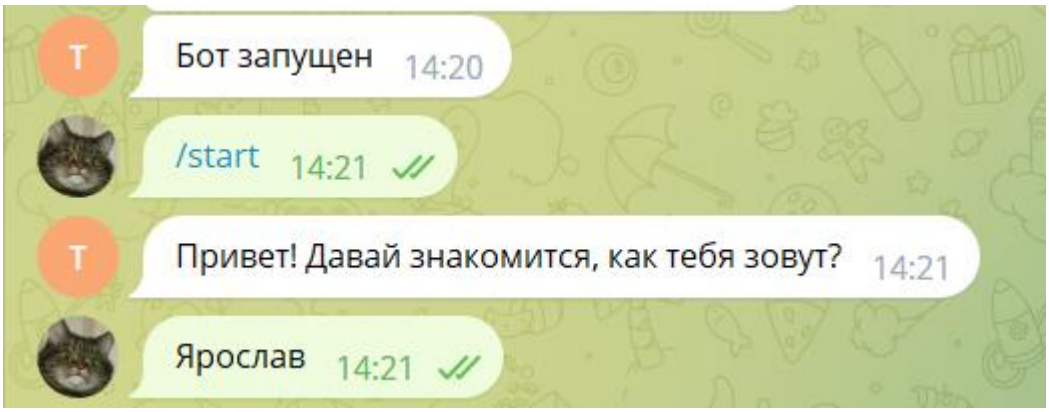
```
from aiogram.dispatcher.filters.state import State, StatesGroup

class Form(StatesGroup):
    name = State()
    age = State()
    gender = State()
    interestGender = State()
    city = State()
    description = State()
    photo = State()
```

utils/notify_admins.py

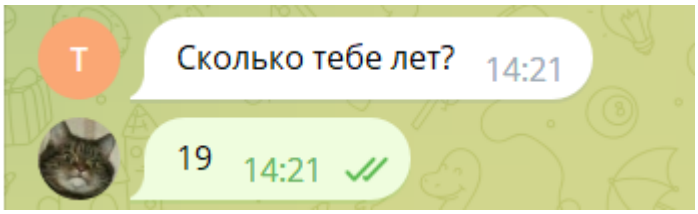
```python
import logging

from aiogram import Dispatcher

from config import ADMIN_ID


async def on_startup_notify(dp: Dispatcher):
  logging.info("Бот запущен")
  await dp.bot.send_message(chat_id=ADMIN_ID, text="Бот запущен")
```

Возможности бота

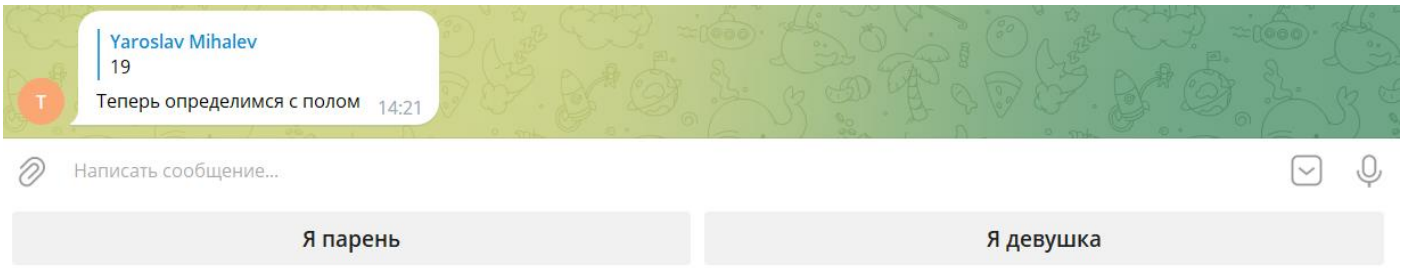
Запрос имени



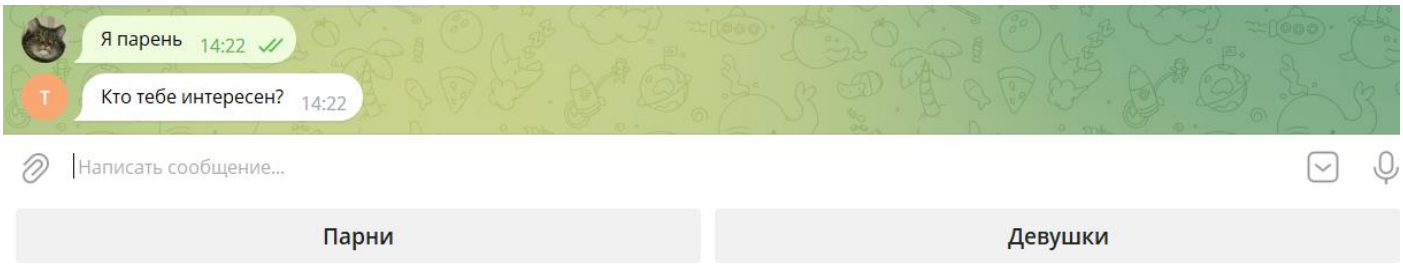
Запрос возраста



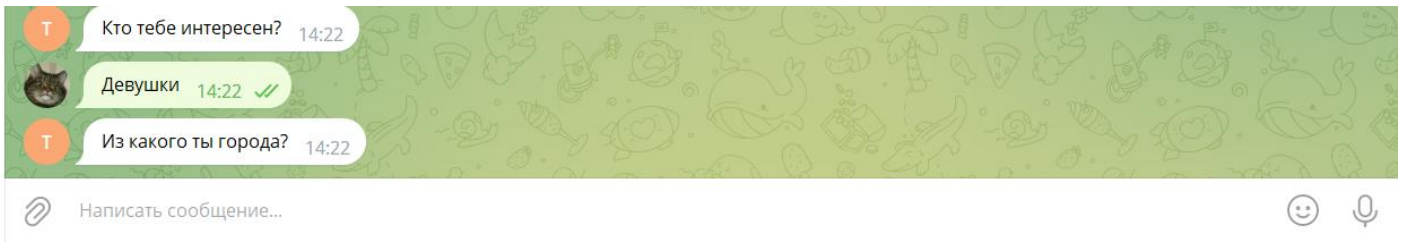
Запрос пола



Запрос пола партнера



Запрос города



Запрос информации о себе

Запрос фото

После получения всей необходимой информации бот высылает сообщение с заполненной анкетой пользователя