Отчет о проделанной работе  
Тема СРСП: Телеграм Бот с тремя кнопками.

Участники:  
Исламберді Ә.  
Лукьянов Е.  
Серик Т.  
Скрипченко В.

Техническое задание:

1. Асинхронный подход к коду: использовать библиотеку aiogram 3.х
2. Логирование. Использование библиотеки logging
3. Использование нативного виртуального окружение venv и requirements.txt для развертывания бота.
4. Использование переменных среды при помощи библиотеки dotenv.
5. Приветствие пользователя по нику в Телеграм при команде /start
6. Реализовать клавиатуру с командами:

* Help – вывод списка команд бота
* Team\_members – вывод разработчиков бота
* Documentation – отправка данного docx файла
* Presentation – отправка презентации pptx формата о возможностях бота

(!) использовать FSInputFile для отправки файлов

Желательно:

1. Совместная разработка через систему git. (задействовано)
2. Публикация кода в частный репозиторий GitHub (выполнено)
3. Расширить функционал бота (задействовано)
   1. Реализована фича с отправкой анекдотов при приветствие

Архитектура проекта:  
**assets/** #директория с файлами tech\_doc.docx, presentation.pptx  
**Modules/** #директория для организации хранения модулей.   
**Modules/getAneсdote.py** /#парсит анекдот с сайта или берет локально при неудаче   
**.env.example** # экземпляр файла с переменными среды  
**.gitignore** # для работы с GitHub  
**README.md** # приветствующий файл на странице репозитория  
**config.py** # файл конфигурации. Подгружает переменные и управляет логикой  
**keyboard.py** # файл с описанием клавиатуры бота  
**main.py** # главный исполняемый файл с логикой бота  
**requirements.txt** # список зависимостей проекта

Распределение задач:  
Лукьянов Е. – верстка всего проекта. Объединение его частей в единый проект. Сопровождение кода.  
Серик Т. – работа с логикой команд бота: отправка файлов, сообщений. Изучение документации и реализаций функционала в коде.  
Исламберді Ә. – рефакторинг скрипта **getAneсdote.py**, обеспечение его отказоустойчивости.  
Скрипченко В. – написание документации, создание презентации, оформления бота.

Команда тестирования:  
Команда разработки.

**Процесс разработки:**  
В файле **main.py** импортируются библиотеки и модули необходимые для дальнейшей работы бота:

**main.py** | Автор – Серик Т.

import asyncio # для асинхронного кода

import logging # для логгирования

from aiogram.filters import Command # фильтр обработки команд

from aiogram.enums import ParseMode, ChatType # перечисления режима парсинга и типов чатов

from aiogram.client.default import DefaultBotProperties # базовая установка глобальных настроек бота

from aiogram import (

    F, # "магические фильтры". удобны для фильтрации событий

    Bot, # сущность для взаимодействия с Telegram API

    html, # модуль форматирования сообщений в html

    Dispatcher, # компонент для обработки и перенаправления событий

)

from aiogram.types import (

    Message, # класс типа сообщений

    FSInputFile, # класс для отправки файлов из файловой системы

    BotCommand, # сущность команды бота из меню команд

)

import config # импортируется конфиг

from keyboard import main\_kb # импортируется разметка клавиатуры

logger = logging.getLogger(\_\_name\_\_) # создается логгер и настраивается

logger.setLevel(logging.DEBUG if config.DEBUG else logging.INFO)

dp = Dispatcher() # создается диспатчер и экземпляр бота

bot = Bot(

    token=config.TOKEN,

    default=DefaultBotProperties(

        parse\_mode=ParseMode.HTML

    )

)

async def main() -> None:

    """

    Main function to start the bot

    """

    logging.info("Starting bot")

    dp.message.filter(F.chat.type == ChatType.PRIVATE) # фильтруем диспатчер по приватным сообщениям

    dp.callback\_query.filter(F.message.chat.type == ChatType.PRIVATE) # фильтруем реагирование на callback запросы (нажатие кнопок) только из приватных чатов

    await bot.delete\_webhook(drop\_pending\_updates=True)  
 await dp.start\_polling(bot)

if \_\_name\_\_ == "\_\_main\_\_":

    asyncio.run(main())

Реализуется файл-конфига **config.py**:

**config.py** | Автор – Лукьянов Е.

import os # необходимо для создания переменной среды

from dotenv import load\_dotenv # необходимо для удобной выгрузки переменных среды

if not load\_dotenv(): # подгружаем переменные среды из .env-файла

    raise FileNotFoundError('.env file does not exist') # вызываем ошибку при его отсутствие

TOKEN = os.environ.get('BOT\_TOKEN') # задаем значение переменной TOKEN

if not TOKEN:

    raise ValueError("BOT\_TOKEN is not set in the environment variables") # вызываем ошибку при его отсутствие

DEBUG = os.environ.get('DEBUG', 'False').lower() in ('true', '1', 'yes', 'on') # создаем кортеж синонимов для значения переменной debug.

TEAM\_MEMBERS\_STR = """

Team:

- Lukyanov E. - @queue\_qq

- Serik T. - @tam\_serik

- Skripchenko V. - @Sunny\_Vls

- Islamberdi A. - @Y191900

"""

Таким образом мы сохраняем секретный ключ в безопасности, храня его в игнорируемом гитом файле - .env

Спонтанно реализуются экземпляр **.env-**файла - **.env.example** и **.gitignore** файл:

**.env.example**

BOT\_TOKEN = "1-9:a-Z"

Этот файл хранит секретный ключ полученный из телеграм-бота @BotFather.

**.gitignore**

\_\_pycache\_\_/

.env

.venv/

.idea/

.vscode/

Хранит игнорируемые файлы при работе с гит

Исламберди А. создает разметку клавиатуры в файле **keyboard.py**, закладывая иерархию кнопок. Так же он оформляет их.

**keyboard.py** |Исламберді Ә.

from aiogram.types import (

    KeyboardButton,

    ReplyKeyboardMarkup,

)

# REPLY\_KEYBOARD

main\_kb = ReplyKeyboardMarkup(

    keyboard = [

        [

            KeyboardButton(text="/team\_members")

        ],

        [

            KeyboardButton(text="/documentation"),

           KeyboardButton(text="/presentation"),

        ],

        [

            KeyboardButton(text="/help")

        ]

    ],

    resize\_keyboard=True,

    input\_field\_placeholder="Choose one option",

)

Лукьянов Е. реализует функцию для регистрации команд в меню бота

**main.py** | Автор – Лукьянов Е.

async def set\_commands(service: Bot):

    """

    Регистрация команд бота в меню команд Telegram

    """

    commands = [

        BotCommand(command="start", description="Start the bot"),

        BotCommand(command="help", description="Help"),

        BotCommand(command="team\_members", description="Show members of development team"),

        BotCommand(command="documentation", description="Show documentation"),

        BotCommand(command="presentation", description="Show presentation"),

    ]

    await service.set\_my\_commands(commands)

    logger.info("Команды бота успешно зарегистрированы")

В **main()** добавляется вызов функции **await set\_commands(bot)**

Совместными усилиями Серика Т. и Лукьянова Е. были реализованы функции обработки команд.

**main.py** | Авторы – Лукьянов Е., Серик Т.

@dp.message(Command("start"))

async def command\_start\_handler(message: Message) -> None:

    """

    Handler for the `/start` command

    """

    await message.reply(

        text=f"Hello, {html.bold(message.from\_user.full\_name)}!",

        reply\_markup=main\_kb

    )

@dp.message(Command("team\_members"))

async def command\_of\_devers(message: Message):

    """

    Handler for the `/team\_members` command

    """

    await message.reply(text=config.TEAM\_MEMBERS\_STR)

@dp.message(Command("help"))

async def cmd\_help(message: Message):

    """

    Handler for the `/help` command

    """

    await message.reply(

        text="Вот список команд:\n/start - начать работу\n/help - помощь"

    )

@dp.message(Command("documentation"))

async def cmd\_doc(message: Message):

    """

    Handler for the `/documentation` command

    """

    await message.reply\_document(

        document=FSInputFile(path="assets/tech\_doc.docx", filename="техническая документация.docx")

    )

@dp.message(Command("presentation"))

async def cmd\_presentation(message: Message):

    """

    Handler for the `/presentation` command

    """

    await message.reply\_document(

        document=FSInputFile(path="assets/pressetation.pptx", filename="презентация.pptx")

    )

В ходе разработки отправки файлов столкнулись с проблемой отсутствия экземпляра кода, нами он не был найден. Изучение документации навело на мысль, что достаточно передать в метод **reply\_document** объект класса **FSInputFile**.

Сперва Лукьяновым Е. был написан асинхронный скрипт, а затем реализован модуль Исламберді Ә., для получения анекдотов.

**main.py** | Авторы – Лукьянов Е., Исламберді Ә.

import asyncio

import aiohttp

from bs4 import BeautifulSoup

import random

import logging

URL = 'https://nekdo.ru/random/'

HEADERS = {

    "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.77 Safari/537.36"

}

JOKES\_list = [

]

def findInSoup(soup: BeautifulSoup):

    element = soup.find('div', id=True, class\_=True)

    return element

async def getAnecdote() -> str:

    logging.info("Запрашиваю анекдот с сайта.")

    async with aiohttp.ClientSession(headers=HEADERS) as session:

        try:

            async with session.get(URL) as response:

                if response.status == 200:

                    logging.info("Успешно получен ответ от сервера.")

                    html = await response.text()

                    soup = BeautifulSoup(html, 'html.parser')

                    element = findInSoup(soup)

                    if element:

                        logging.info("Анекдот успешно найден на странице.")

                        return element.text.strip()

                logging.warning(f"Получен неудачный статус ответа: {response.status}. Использую локальный анекдот.")

                return random.choice(JOKES\_list)

        except Exception as e:

            logging.error(f"Ошибка при запросе: {e}. Использую локальный анекдот.")

            return random.choice(JOKES\_list)

if \_\_name\_\_ == "\_\_main\_\_":

    async def main():

        text = await getAnecdote()

        print(text)

    asyncio.run(main())

В конечном счете был определен файл зависимостей **requirements.txt**

**requirements.txt** | Автор – Исламберді Ә.

aiogram~=3.18.0

python-dotenv~=1.0.1

beautifulsoup4~=4.13.3

requests~=2.32.3

После всей выше проделанной работы Лукьянов Е. произвел финальный рефакторинг кода и проект был опубликован в публичный репозиторий https://github.com/ArtzSU/SRSP-bot

**О ходе разработки:**

В процессе разработки шло освоения git, Серит Т. и Лукьянов Е. коммитили свои этапы разработки в приватный репозиторий <https://github.com/ArtzSU/tgbotsrsp>.   
Мерж веток производил Лукьянов Е.  
Предложенные правки от коллаборатора Alex: exituser. Практикующий Python Software Engineer.

По итогам разработки весь код опубликован в публичный репозиторий <https://github.com/ArtzSU/SRSP-bot>

Результатом разработки стало освоение асинхронного подхода к разработке кода, а так же работа в команде. Использование переменных среды и использование git послужили хорошим опытом для всех участников.

Источники:  
Официальная документация aiogram – https://docs.aiogram.dev/en/v3.18.0/  
Stackoverflow - https://ru.stackoverflow.com