GeekBrains

**Разработка Телеграм-бота для агентства,**

**предоставляющего маркетинговые услуги**

IT-специалист:

Программист Python

Попова Е.А.

Москва

2024

Содержание

[Введение 3](#_Toc153488348)

[План 9](#_Toc153488349)

[Основная часть 10](#_Toc153488350)

[Подготовка виртуального окружения для разработки чат-бота 10](#_Toc153488351)

[Подготовка Telegram 19](#_Toc153488352)

[Разработка чат-бота 22](#_Toc153488353)

[Выводы 51](#_Toc153488354)

[Используемые материалы 53](#_Toc153488355)

**Введение**

В настоящее время чат-боты часто используются в бизнесе для консультации и предоставления информации клиентам. Для этих целей чаще всего используют чат-боты в социальных сетях или приложениях.

Виртуальный собеседник, программа-собеседник, чат-бот (англ. chatbot) — программа, которая выясняет потребности пользователей, а затем помогает  удовлетворить их.

Это алгоритм, с помощью которого происходит автоматическая отправка сообщений пользователю по заранее продуманному сценарию.

Чат-бот ведет коммуникацию от лица компании или бренда с  целью упростить онлайн-общение (предоставить актуальную информацию в  наиболее оперативные сроки), используется как альтернатива переписке с живым  оператором или звонку менеджеру компании.

Существуют разные компании-разработчики, предлагающие конструкторы чат-ботов, с доступным функционалом, в котором не сложно разобраться, чтобы начать использовать его для своих целей.

В простейшем случае чат-бот представляет собой наборы возможных вопросов/запросов пользователя и соответствующих им ответов, с помощью которых пользователь более подробно знакомится с компанией, ее услугами/товарами, опытом, отзывами.

Задачи чат-бота расположить пользователей к компании, подготовить их к приобретению товаров/услуг компании.

Чат-боты используются для:

* информирования клиентов:
* о новинках
* об эксклюзивных предложениях
* о предстоящих мероприятиях,
* об акциях, о промокодах
* об изменениях в расписании
* о назначенных встречах
* о записях на прием
* ознакомления с продуктовой линейкой
* выбора товара/услуги по персональным запросам пользователя
* оформления заказа
* подтверждения доставки
* предоставления ответов на часто задаваемые вопросы
* для предоставления дополнительной полезной для пользователей информации.

Чат-боты помогают повышать «доходимость» пользователей до мероприятий (семинаров, вебинаров и т.п.) путем создания дополнительных «касаний» с аудиторией.

С помощью чат-ботов можно оформлять заказы на товары/услуги уже известные пользователю.

Преимущества чат-ботов:

* настраиваются универсальные сценарии общения;
* открываемость сообщение в мессенджерах больше, чем e-mail-рассылок, sms-рассылок;
* многие сейчас привыкли общаться в мессенджерах и получать информацию/оформлять заказы также через мессенджеры, не хотят звонков;
* экономия времени;
* сокращение нагрузки на менеджеров по продажам;
* обработка бОльшего количества заявок/обращений, т.к. могут обрабатываться одновременно несколько заявок/обращений;
* повышение лояльности аудитории за счет быстрых ответов;
* сокращение потерь клиентов из-за невозможности ответить оперативно на вопросы;
* увеличение объема продаж;
* автоматические допродажи;
* тропинка возврата;
* соберет заявки/заказы для последующего подтверждения;
* экономит время сотрудников, отвечающих по телефону на однотипные вопросы, а в крупных компания – можно сократить расходы на персонал, задействовав его на более значимых операциях;
* не нужно мотивировать, не нужно подстраиваться под настроение;
* не забудет предложить дополнительные услуги/товары;
* позволяет познакомиться с пользователем и узнать его запросы/потребности;
* позволяет провести клиента по всей воронке продаж;
* позволяет предложить ту часть линейки продаж, которая подходит персонально пользователю в зависимости от его запросов, или всю линейку продаж;
* позволяет собирать отдельные группы клиентов по одинаковым запросам и отправлять только им отдельные предложения;
* обеспечивать круглосуточную поддержку клиентов;
* выгружает данные в Excel;
* позволяет загрузить данные в CRM систему для обработки и хранения информации;
* можно соединить с таргетированной рекламой;
* ответы сохраняются в системе;
* формирование базы клиентов, а также базы потенциальных клиентов – тех, кто сразу не решился вступить в диалог/что-то приобрести, можно продолжить с ним работу – отправлять им информацию, которая позволяет лучше узнать Вас/Ваш товар/услугу;
* все диалоги сохраняются – в любое время можно просмотреть

Виды чат-ботов:

* визитка
* ответы на часто задаваемые вопросы
* квиз, викторина, опрос, тест, калькулятор, розыгрыш
* сбор контактных данных
* вебинарная воронка
* марафоны с заданиями

Каналы, в которых используются чат-боты:

* Telegram
* WhatsApp
* Viber
* VK
* Instagram[[1]](#footnote-1)

Еще недавно в мире WhatsApp лидировал по всем фронтам, а сейчас Facebook1 Messenger не заменим в США и Канаде, а Telegram бьет рекорды популярности в России.

Статистика, ежегодно собираемая специалистами из SimilarWeb, наглядно отображает неравномерное распределение популярности мессенджеров в мире. Например, в Северной Америке аудитория почти не вылезает из Facebook1 Messenger, а в Китае почти не найти тех, кто не общается с друзьями и близкими в WeChat.

Telegram быстрее конкурентов обрастает новой аудиторией (прирост исчисляется миллионами пользователей).

Telegram – кроссплатформенная система обмена сообщениями с поддержкой VoIP[[2]](#footnote-2) и почти неограниченной коллекцией функций для взаимодействия с миром вокруг. Звонки и видеоконференции, обмен стикерами и фотографиями, загрузка аудиосообщений и файлов любого объема – мессенджер с легкостью подстраивается под запросы любой аудитории.

В ноябре 2023 года аудитория Telegram составляет 800 миллионов активных пользователей в месяц.

Преимущества Telegram :

* молниеносная скорость работы: Telegram мгновенно обрабатывает запросы и не мешает поддерживать общение в десяти чатах сразу;
* расширенные параметры безопасности, включающие и скрытый чат, и сквозное шифрование, и сообщения, стирающиеся по таймеру;
* полноценная синхронизация чатов: на любой платформе все сообщения, файлы и иная информация никуда не денется;
* не имеет ограничений по количеству подписчиков чат-бота.

Недостатки:

* технические неполадки в версии мессенджера для Android: например, периодически нельзя делиться ссылками или писать сообщения;
* некоторые обещанные функции в Telegram с недавних пор выдаются лишь в рамках платной подписки (от 309 рублей в месяц);
* действующие параметры безопасности по-прежнему пропускают спам-рассылку и рекламные сообщения.

Бот, разработанный в данной работе, будет приветствовать новых пользователей, предоставлять информацию о компании, о ее услугах, кейсах, ценах на предоставляемые услуги, контактах, присылать дополнительные материалы по запросу пользователя – презентация с кейсами компании, регистрировать заявки на обратный звонок.

**План**

* 1. Подготовка виртуального окружения для разработки чат-бота
  2. Подготовка Telegram – регистрация чат-бота
  3. Разработка схемы чат-бота
  4. Создание бота в соответствии со схемой

**Основная часть**

**Подготовка виртуального окружения для разработки чат-бота**

Работа выполняется в Visual Studio Code.

Visual Studio Code, также обычно называемый VS Code, является редактором исходного кода, созданным Microsoft на платформе Electron Framework для Windows, Linux и macOS. Функции включают поддержку отладки, подсветку синтаксиса, интеллектуальное завершение кода, фрагменты, рефакторинг кода и встроенный Git.

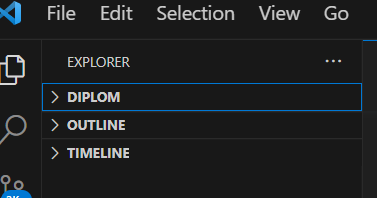
Git - это бесплатная распределенная система контроля версий с открытым исходным кодом, предназначенная для быстрой и эффективной работы со всеми проектами, от небольших до очень крупных.

Git - это консольная утилита, для отслеживания и ведения истории изменения файлов в проекте.

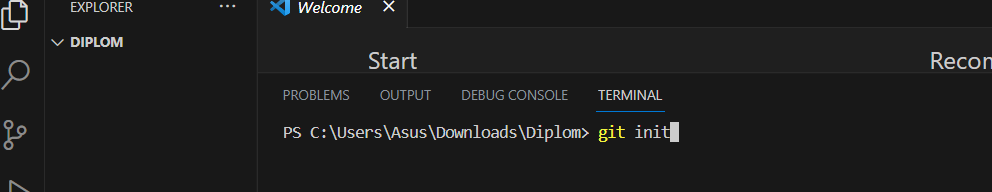
С помощью Git-a можно откатить проект до более старой версии, сравнивать, анализировать или сливать изменения в репозиторий.

Репозиторием называют хранилище кода и историю его изменений. Git работает локально, и все репозитории хранятся в определенных папках на жестком диске.

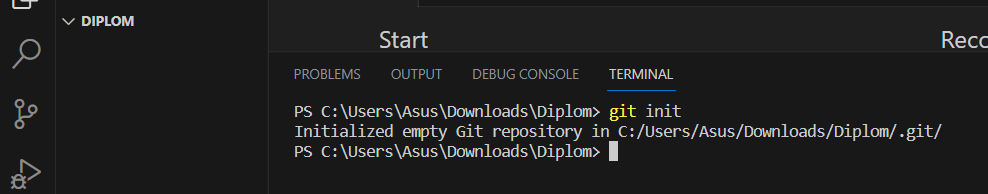
Создаем директорию, в которой будет создаваться и храниться проект и все, что необходимо для его работы

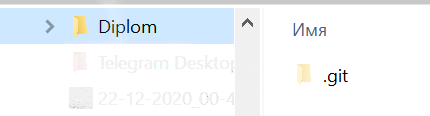


Создаем репозиторий Git – команда - git init.



В текущем рабочем каталоге создается подкаталог .git  со всеми необходимыми метаданными Git для нового репозитория. Метаданные включают подкаталоги для объектов, ссылок и файлов шаблонов.





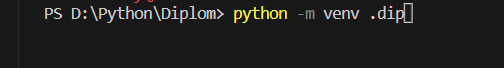
Создаем виртуальное окружение.

Виртуальное окружение - это отдельное пространство, которое используется для управления зависимостями и позволяет работать с разными версиями библиотек, т.е. со своим набором настроек. Это позволяет избежать конфликтов между разными проектами и их зависимостями, упрощает работу с проектом при передаче его для дальнейшей разработки/доработки/использования, т.к. не нужно новым разработчикам/пользователям не нужно переустанавливать у себя все используемые в проекте программы в тех же версиях – всё уже содержится в данном пакете. Также изменения внутри проекта не влияют на "эталонную" версию Python.

Команда: python -m venv .dip

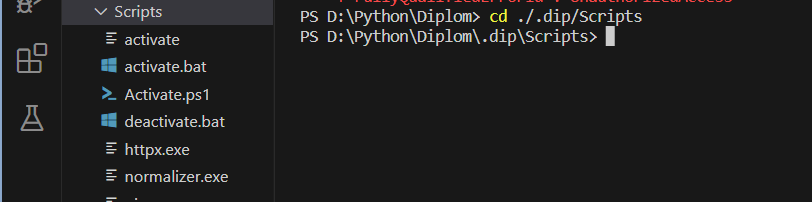
venv (Virtual Environment) — это модуль Python, который позволяет создавать изолированные виртуальные среды для Python-проектов.

.dip – папка, в которой будут храниться все библиотеки



Переходим в папку Scripts.

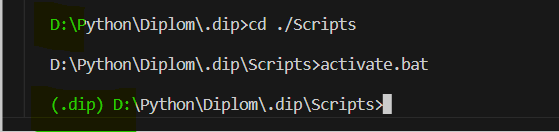
Для перемещения по папкам используется команда cd (change directory - смена директории). Если необходимо вернуться в директорию выше, используется команда - cd .. (с двоеточием).



Для работы в рамках виртуального окружения, его необходимо активировать. Для этого необходимо использовать команду:

acrivate.bat

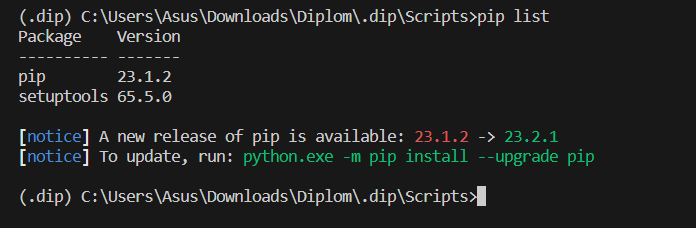
В начале строки должно быть указано название папки (.dip) – значит виртуальное окружение активировано:



Проверяем, что есть в виртуальном окружении:

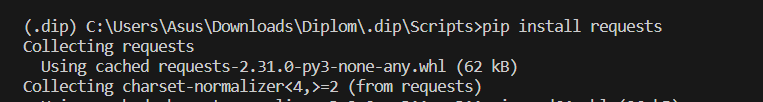
Команда: pip list

pip — система управления пакетами, которая используется для установки и управления программными пакетами, написанными на Python.



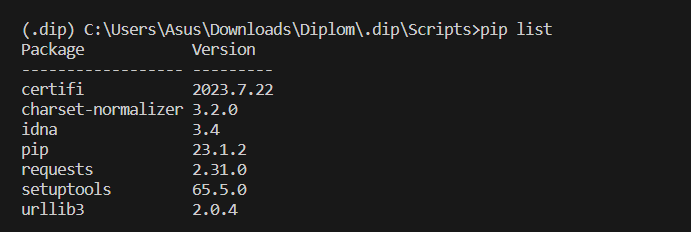
Устанавливаем библиотеку requests, которая создана для быстрой и простой работы с запросами:

pip install requests

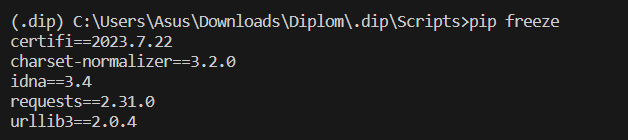


Проверяем, какие пакеты установлены сейчас:

pip list



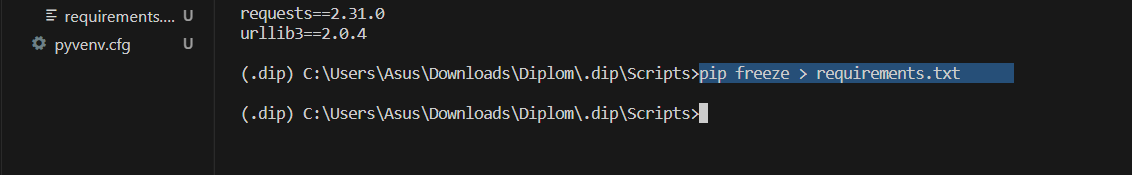
Просмотреть информацию о пакетах также можно с помощью команды: pip freeze



Команда  pip freeze  выводит все установленные в интерпретатор сторонние пакеты. В список попадают не только прямые зависимости, но и подзависимости — это поможет более точно воссоздать окружение по этому файлу.

Создаем файл, в котором будет сохраняться информация обо всех зависимостях, которые мы установили:

pip freeze > requirements.txt



requirements.txt — это список внешних зависимостей - список всех модулей и пакетов Python, которые нужны для полноценной работы нашей программы. Его использование позволяет легко отслеживать весь перечень нужных компонентов, избавляя пользователей от необходимости их ручного поиска и установки.

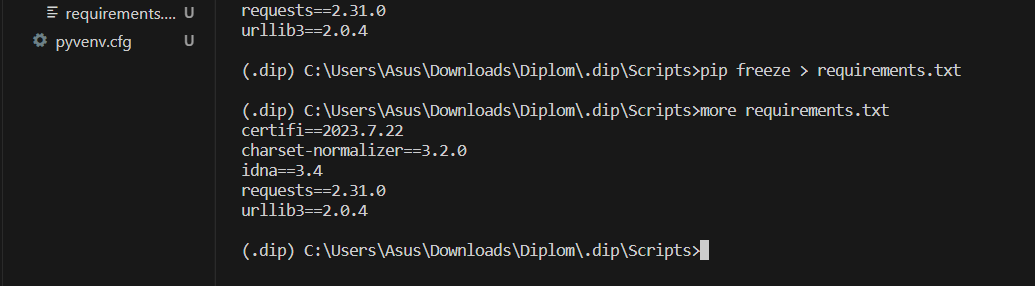
С помощью команды:

pip install –r requirement.txt

можно забрать из этого файла указанные в нем библиотеки.

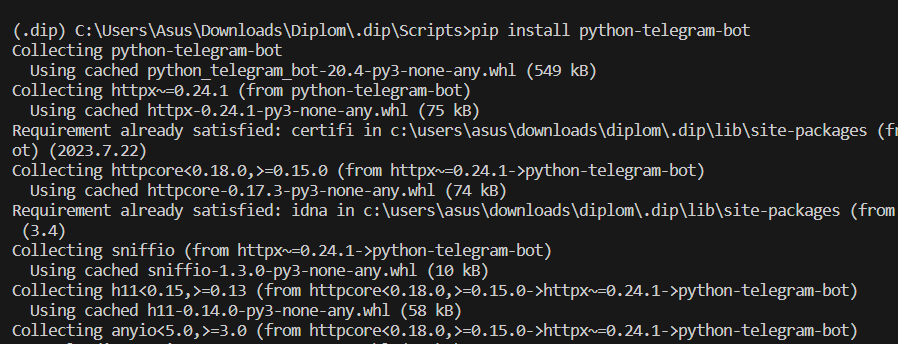
Просмотреть содержимое файла можно с помощью команды:

more requirements.txt



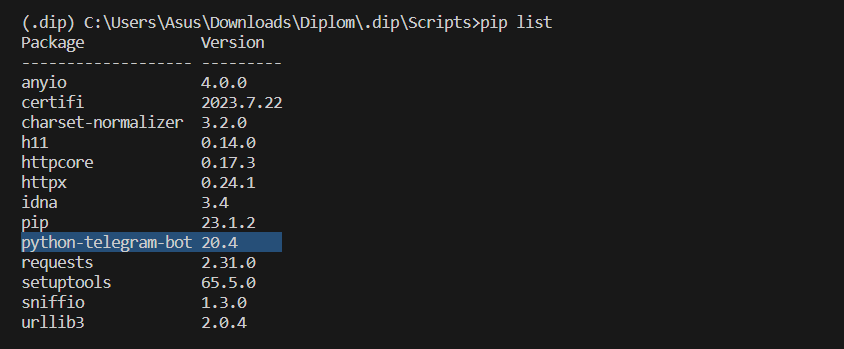
Для создания телеграмм-бота устанавливаем пакет python-telegram-bot:

pip install python-telegram-bot



Проверяем установку:

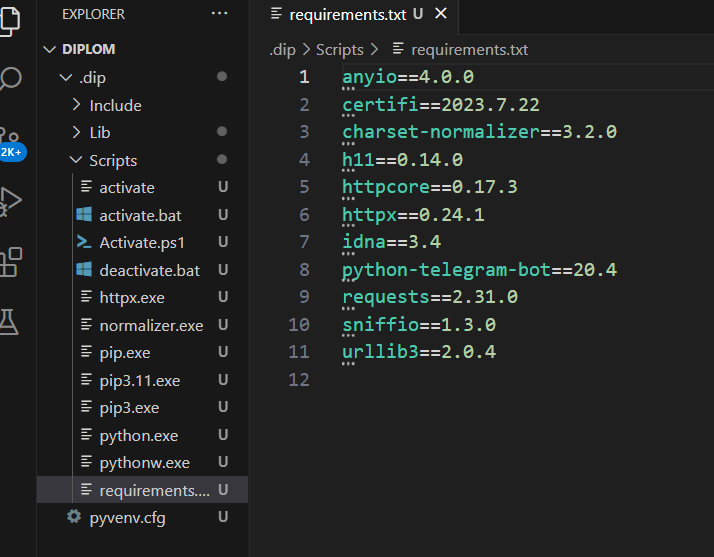
pip list



Снова набираем команду:

pip freeze > requirements.txt

и все новые зависимости сохраняются в файл requirements.txt автоматически.



[Пакет python-telegram-bot](https://docs-python.ru/packages/biblioteka-python-telegram-bot-python/) содержит ряд высокоуровневых классов, которые делают разработку ботов простой и понятной. Эти классы содержатся в модуле telegram.ext.

Пакет python-telegram-bot пытается использовать как можно меньше сторонних зависимостей. Однако для некоторых функций использование сторонней библиотеки более разумно, чем повторная реализация функциональности. Поскольку эти функции являются необязательными, соответствующие сторонние зависимости не устанавливаются по умолчанию. Это позволяет избежать ненужных конфликтов зависимостей для пользователей, которым не нужны дополнительные функции.

Высокоуровневый интерфейс пакета python-telegram-bot построен поверх [чистой реализации Telegram Bot API](https://docs-python.ru/packages/biblioteka-python-telegram-bot-python/modul-telegram-bot-raw/) и находится в подмодуле расширений telegram.ext. Он предоставляет простой в использовании интерфейс.

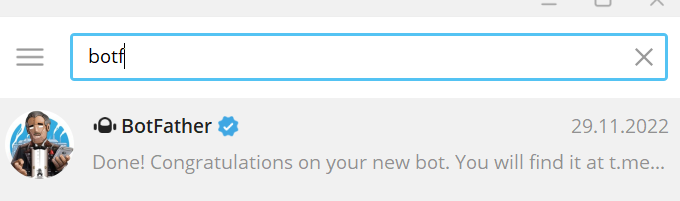
Для работы требуется сгенерировать токен доступа к API. Для этого необходимо пообщаться с [@BotFather](https://t.me/BotFahter) в Telegram(см.следующую главу).

Токен API, часто называемый токеном доступа, аутентификации или авторизации, представляет собой уникальный идентификатор, который предоставляет ограниченный доступ к определенным ресурсам и службам, предоставляемым через API приложения. Эти токены играют решающую роль в обеспечении безопасности и сохранении конфиденциальности пользователей в области веб-, мобильных и серверных приложений, особенно при доступе к данным или выполнении действий внутри приложения.

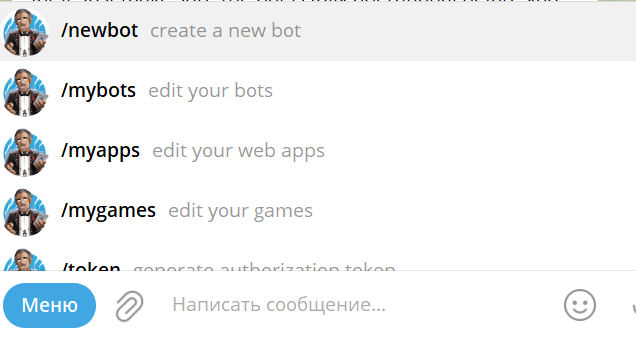
**Подготовка Telegram**

Необходимо создать сам Телеграм-бот:

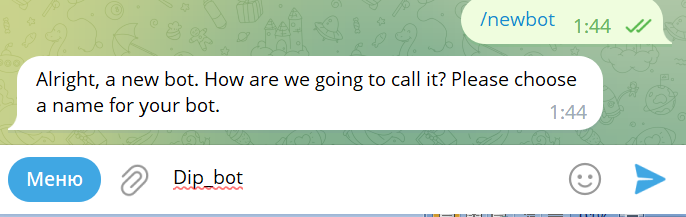
В Telegram необходимо найти пользователя - @BotFather



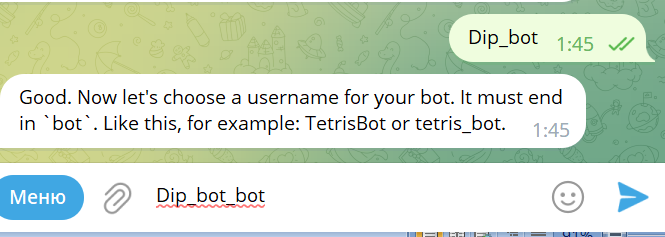
Нажимаем внизу кнопку Меню и выбираем /newbot



Указываем название бота:

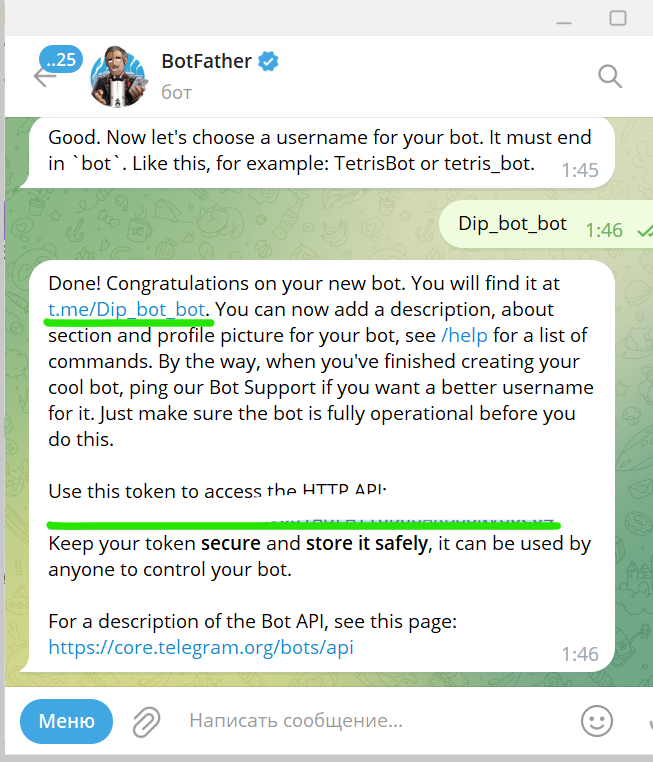


Необходимо указать username бота - публичное имя, благодаря которому пользователя можно найти в мессенджере через функцию глобального поиска. В конце выбранного имени обязательно должно быть указано слово bot:

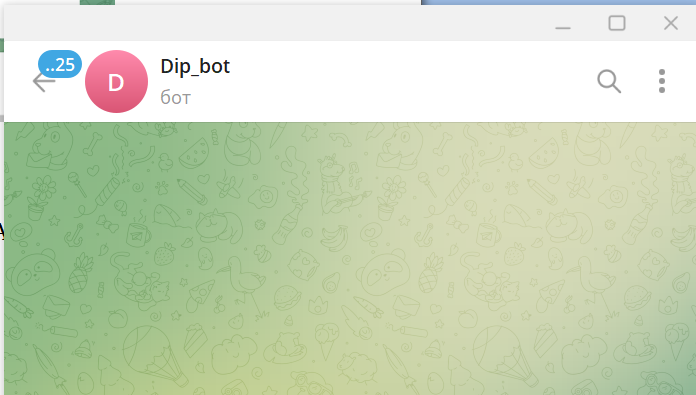


При удачной регистрации будет получено сообщение, что новый бот готов.

В сообщении также указана ссылка, по которой можно найти бот – t.me/Dip\_bot\_bot и здесь же указан token – его никому нельзя показывать/пересылать, т.к. это «ключ» к боту

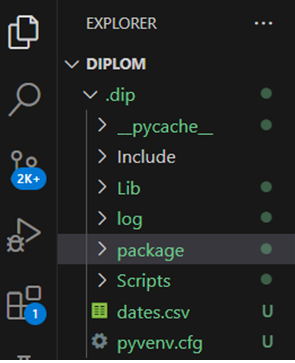


Вот и готовый бот:



**Разработка чат-бота**

В нашем виртуальном окружении создаем папку package – в ней будут храниться все файлы бота:



В папке package создаем файл с названием \_\_init\_\_.py

Написанные на Python сервисы обычно оформляют как пакеты. Для запуска используется корневой модуль пакета, т.е. запуск пакета как программы.

Каждая директория, которая указана в пути импорта, должна содержать файл \_\_init\_\_.py

Можно оставить файл \_\_init\_\_.py пустым. Можно указать в нем, что будет импортироваться при from модуль import \*.

Инициализация пакета - при первом импорте каталога, Python выполняет код в файле \_\_init\_\_.py этого каталога. Здесь можно открыть соединение с БД, создать файл с данными и тп. Т.е. в нем, с указанием

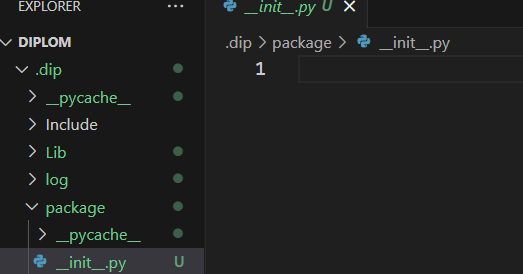
if \_\_name\_\_ == '\_\_main\_\_'

может быть указан разбор аргументов командной строки при запуске и запуск сервиса.

\_\_init\_\_.py - создает пространство имен для объектов модулей (особенно, если модулей в смысле файлов, в директории нет).

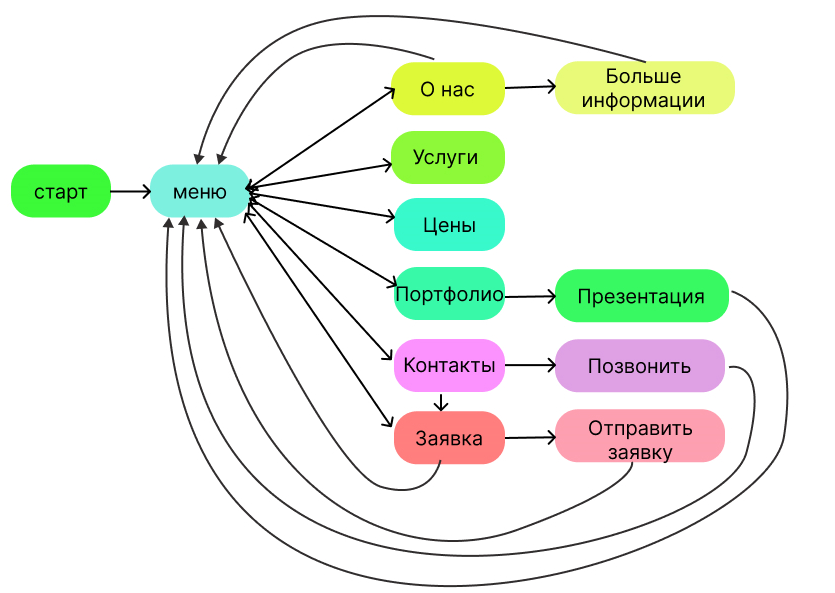
from .. \* - импортируется или все имена из каталога, либо, если в \_\_init\_\_.py определен список \_\_all\_\_, то только имена из этого списка.

Здесь файл оставляем пустым:



Все последующие файлы создаем в папке package.

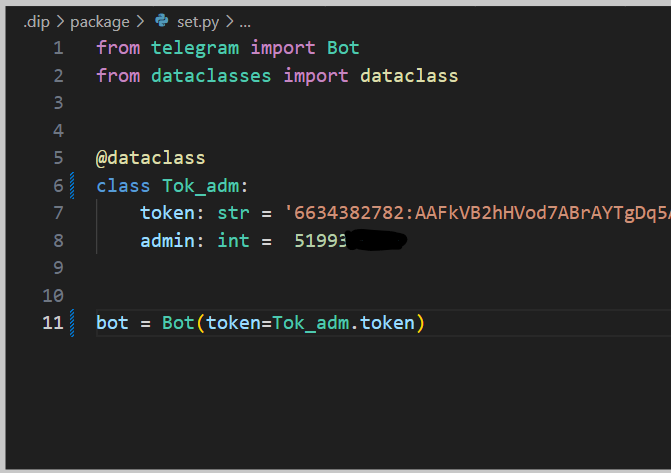
Схема чат-бота:



Создаем файл set.py

Зафиксируем в нем данные бота из Telegram, чтобы можно было с ним взаимодействовать с использованием токена, который сформировал @botfather

Создаем дата-класс Tok\_adm:



Импортируем из пакета telegram Bot и из dataclasses – dataclass.

admin – это личный идентификатор администратора бота в Telegram. Его можно использовать в дальнейшем для администрирования чат-бота. Id чат-бота можно получить в боте - https://t.me/getmyid\_bot.

Используется декоратор @dataclass – позволяет автоматически генерировать методы класса для работы с данными.

При применении этого декоратора к классу автоматически добавляется реализация методов - \_\_init\_\_- создает конструктор класса, принимающий значения для каждого атрибута, и \_\_repr\_\_ - создает строковое представление объекта, которое используется при вызове repr(). Позволяет использовать аннотации типов для объявления типов полей класса, что упрощает чтение кода.

Создаем файл view.py

В этом файле будут сохранены функции, содержащие тексты, которые будут показываться пользователям в Telegram при выборе ими необходимых им действий.

После запуска бота пользователь получит адресное приветствие с собственным именем. Для этого используется конструкция – {update.effective\_user.first\_name}.

Для переноса текста на следующую строку, чтобы текст легче читался чат-боте, в том числе можно использовать перенос на две строки, чтобы разграничить/выделить часть текста, используется конструкция:

\n\

\n

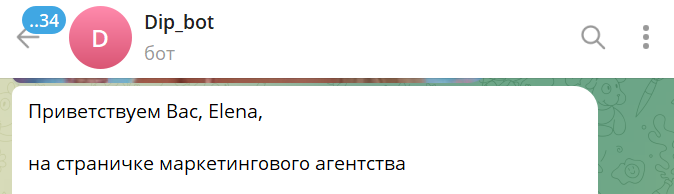
Модуль расширений telegram состоит из нескольких классов. Одним из наиболее важных является - Update.

Его импортируем в начале файла –

from telegram import Update

Класс Update постоянно слушает сервер Telegram, получает новые сообщения.

В функции start\_msg в тексте указываем ссылку - {update.effective\_user.first\_name}, с помощью которой в боте пользователь получит персональное приветствие по его имени (user.first\_name), которое он указал в настройках своего Telegram.



Ключевое слово в Python “def” используется для определения функции, оно помещается перед именем функции, которое предоставляется пользователем для создания пользовательской функции.

В Python функция - это логическая единица кода, содержащая последовательность инструкций с отступом под именем, заданным с использованием ключевого слова “def”.

Внутри каждой функции прописываем тексты, которые будут отправляться пользователю в чат-боте.

Полностью файл view.py:

from telegram import Update

def start\_msg(update: Update):

    return f'Приветствуем Вас, {update.effective\_user.first\_name},\n\

        \nна страничке маркетингового агентства\n\

        \n"Продаем всё, быстро и легко!"\n\

        \nНажмите кнопку "menu" для выбора необходимого действия\n\

        \n/menu'

def about\_msg():

    return 'Мы - маркетинговое агентство полного цикла по онлайн продвижению малого и среднего бизнеса в Интернете и соцсетях\n\

        \nМы работаем с 2017 года и каждый день увеличиваем прибыль своих клиентов за счет эффективных комплексных решений\n\

        \nУспешно реализовали более 150 проектов\n\

        \n/menu\n\

        \n/more\_information'

def information\_msg():

    return 'Мы настроены на долгосрочное сотрудничество и партнерские отношения. Наша команда вникает во все тонкости Вашего бизнеса и готовит персональное проработанное коммерческое предложение\n\

        \nУ нас есть разные варианты пакетов и отдельных услуг, которые подойдут под любой тип бизнеса\n\

        \nУ нас вы можете заказать комплекс услуг по продвижению бизнеса\n\

        \n/menu'

def services\_msg():

    return 'Наши услуги:\n\

        \nРазработка маркетинговой стратегии и ее внедрение\n\

        \nМаркетинговый аудит бизнеса\n\

        \nАутсорсинг маркетинга\n\

        \nРазработка сайтов\n\

        \nВедение социальных сетей\n\

        \nНастройка Яндекс Директ и Google Adwords, таргетированной рекламы\n\

        \n/menu'

def portfolio\_msg():

    return 'Вы можете ознакомиться с некоторыми нашими проектами здесь(кратко):\n\

        \nФинансовый проект "Бенфин" - Комплексное продвижение нового сервиса, разработка маркетинговой стратегии, подбор рекламных каналов и запуск рекламы\n\

        \nСеть студий женского фитнеса «GGym» - Комплексное продвижение бренда, разработка фирменного стиля, привлечение новых посетителей\n\

        \nПекарня «Хлеб с маслом» - разработка и внедрение комплексной стратегии для пекарни\n\

        \nПолная информация по этим и другим проектам представлена в презентации: \n\

        /get\_presentation\n\

        \n/menu'

def price\_msg():

    return 'Наши цены:\n\

        \nРазработка маркетинговой стратегии и ее внедрение - от 50 000 руб\n\

        \nМаркетинговый аудит бизнеса - от 30 000 руб\n\

        \nАутсорсинг маркетинга - от 30 000 руб\n\

        \nРазработка сайтов - от 15 000 руб\n\

        \nВедение социальных сетей - от 15 000 руб\n\

        \nНастройка Яндекс Директ и Google Adwords, таргетированной рекламы - от 15 000 руб\n\

        \n/menu'

def contacts\_msg():

    return 'Наши контакты:\n\

        \nтел.: + 7 495 550 15 05\n\

        \nадрес: Москва, Лесной туп., 25, оф.2\n\

        \nРежим работы: с 10-00 до 21-00\n\

        \nВы можете оставить заявку с Вашими контактными данными - мы свяжемся с Вами в указанное Вами время - обсудим интересующие Вас вопросы!\n\

        \nОставить заявку: /request\n\

        \nили позвонить нам: /call\n\

        \n/menu'

def call():

    return '+ 74955501505\n\

        \n/menu'

def send\_request():

    return 'Благодарим Вас за заявку! Мы свяжемся с Вами в ближайшее время!\n\

            \n/menu'

def request():

    return 'Нажмите, пожалуйста, кнопку send\_request,\n\

        \n/send\_request\n\

        \nчтобы автоматически отправить свои данные\n\

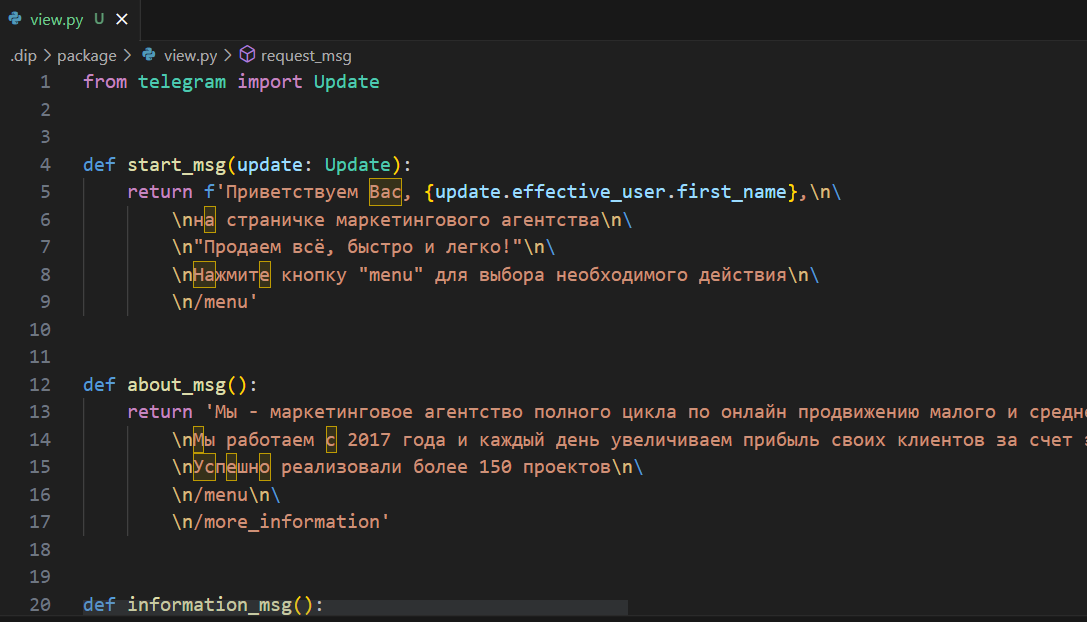
        \n(имя и ник в Telegram)\n\

        \nнашему менеджеру,чтобы он смог связаться с Вами,\n\

        \nлибо нажмите кнопку menu для возврата в основное меню\n\

        \n/menu'

В VS Code выглядит так:



Создаем в папке package новый файл – commands.py

В файле commands.py прописываем все команды с ссылками – меню в боте:

from telegram.ext import ApplicationBuilder, CommandHandler

import package.bot

import package.set

app = ApplicationBuilder().token(package.set.Tok\_adm.token).build()

app\_admin = ApplicationBuilder().token(package.set.Tok\_adm.admin).build()

app.add\_handler(CommandHandler("start", package.bot.start))

app.add\_handler(CommandHandler("menu", package.bot.menu))

app.add\_handler(CommandHandler("about\_us", package.bot.about))

app.add\_handler(CommandHandler("services", package.bot.services))

app.add\_handler(CommandHandler("portfolio", package.bot.portfolio))

app.add\_handler(CommandHandler("price", package.bot.price))

app.add\_handler(CommandHandler("contacts", package.bot.contacts))

app.add\_handler(CommandHandler("more\_information", package.bot.information))

app.add\_handler(CommandHandler("send\_request", package.bot.send\_request))

app.add\_handler(CommandHandler("request", package.bot.request))

app.add\_handler(CommandHandler("get\_presentation", package.bot.get\_presentation))

app.add\_handler(CommandHandler("call", package.bot.call))

app.run\_polling()

*ApplicationBuilder* – этот класс служит инициализатором (метод, который вызывается при создании нового объекта класса) для telegram.ext.Application с помощью так называемого шаблона builder. Чтобы создать telegram.ext.Application, сначала инициализируется экземпляр этого класса. Аргументы для создания приложения telegram.ext. затем добавляются путем последующего вызова методов конструктора. Наконец, приложение telegram.ext.создается путем вызова функции build():

app = ApplicationBuilder().token(package.set.Tok\_adm.token).build()

Некоторые аргументы являются взаимоисключающими. Например, после вызова функции token() нельзя установить пользовательского бота с помощью функции bot() и наоборот.

 CommandHandler - rласс для обработки команд Telegram.

Command - это сообщения Telegram, которые начинаются со знака – « / », за которыми необязательно следует @ и имя бота и/или какой-либо дополнительный текст. Обработчик добавит список в CallbackContext с именем CallbackContext.args. Он будет содержать список строк, который представляет собой текст, следующий за командой, разделенный на одиночные или последовательные пробельные символы.

Порядок и приоритет учитываются. Будет использоваться 0 или 1 обработчик для каждой группы. Завершите обработку обновления с помощью telegram.ext.ApplicationHandlerStop.

add\_handler – обработчик-экземпляр подкласса telegram.ext.BaseHandler. Все обработчики организованы в группы с числовым значением. Группа по умолчанию равна 0. Все группы будут оценены для обработки обновления, но будет использоваться только 0 или 1 обработчик для каждой группы. Если telegram.ext.ApplicationHandlerStop вызывается из одного из обработчиков, никакие другие обработчики (независимо от группы) вызываться не будут.

Приоритет/порядок обработчиков определяется следующим образом:

Приоритет группы (меньший номер группы == более высокий приоритет)

Будет использоваться первый обработчик в группе, который может обработать обновление. Другие обработчики из группы использоваться не будут. Порядок, в котором обработчики были добавлены в группу, определяет приоритет.

Каждый обработчик является экземпляром подкласса класса telegram.ext.Handler. Пакет python-telegram-bot предоставляет [классы обработчиков](https://docs-python.ru/packages/biblioteka-python-telegram-bot-python/tipy-obrabotchikov/) почти на все стандартные случаи, но если нужно что-то конкретное, то можно создать собственный обработчик, наследуясь от класса Handler.

Для того, чтобы использовать функции из других файлов, необходимо их импортировать:

import package.bot

import package.set

В папке package создаем файл bot.py.

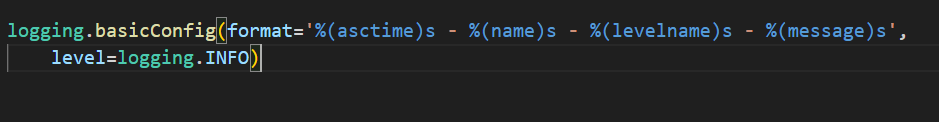
В этом файле прописываем функции, вызываемые при нажатии той или иной кнопки меню в боте, а также указываем для каждой функции фотографии, гифки, презентации, которые должны демонстрироваться пользователю при нажатии интересующего его пункта меню.

Также здесь прописываем функцию логирования (своего рода журнал) для понимания, что и когда происходило с ботом.

Используется встроенная библиотека logging.

Инициализируем метод basicConfig. В метод передается два параметра – level и – format.





level – определяет уровень выводимой информации (DEBUG, INFO, WARNING, ERROR, CRITICAL ).

DEBUG - самый низкий уровень логирования, предназначенный для отладочных сообщений, для вывода диагностической информации о приложении.

INFO - этот уровень предназначен для вывода данных о фрагментах кода, работающих так, как ожидается.

WARNING - этот уровень логирования предусматривает вывод предупреждений, он применяется для записи сведений о событиях, на которые программист обычно обращает внимание. Такие события вполне могут привести к проблемам при работе приложения. Если явно не задать уровень логирования — по умолчанию используется именно warning.

ERROR - этот уровень логирования предусматривает вывод сведений об ошибках — о том, что часть приложения работает не так как ожидается, о том, что программа не смогла правильно выполниться.

CRITICAL - этот уровень используется для вывода сведений об очень серьёзных ошибках, наличие которых угрожает нормальному функционированию всего приложения. Если не исправить такую ошибку — это может привести к тому, что приложение прекратит работу.

format - определяет формат выводимой информации:

- %(asctime)s- заменяется на текущее время в формате гггг-мм-дд чч:мм:сс

- %(levelname)s – уровень важности лога

- %(name)s – имя логгера, который записывает сообщение

- %(filename)s – имя файла, в котором была вызвана функция логирования

- %(funcName)s – имя функции, в которой была вызвана функция логирования

- %(lineno)d – номер строи, в которой была вызвана функция логирования

- %(message)s – текст сообщения, переданного для записи в лог

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

Также создаем отдельную функцию, в которой указаываем, в какой файл какие данные должны сохраняться при выборе пользователем определенных команд, и затем вызывать ее в других функциях, действия пользователя которых необходимо фиксировать.

При определенных действиях информация о их совершении будет сохраняться в файле ‘dates.csv’.

* Открываем файл, в который будем записывать информацию о событии, с помощью встроенной функции open():

в кавычках указывается название файла с расширением - ‘dates.csv’,

* Затем в кавычках указывается режим открытия файла.

По умолчанию используется режим чтения (r), если другое не указано.

В зависимости от поставленной задачи можно использовать другие режимы открытия файла:

| Режим | Описание |
| --- | --- |
| r | Только для чтения. |
| w | Только для записи. Создаст новый файл, если не найдет с указанным именем. |
| rb | Только для чтения (бинарный). |
| wb | Только для записи (бинарный). Создаст новый файл, если не найдет с указанным именем. |
| r+ | Для чтения и записи. |
| rb+ | Для чтения и записи (бинарный). |
| w+ | Для чтения и записи. Создаст новый файл для записи, если не найдет с указанным именем. |
| wb+ | Для чтения и записи (бинарный). Создаст новый файл для записи, если не найдет с указанным именем. |
| a | Откроет для добавления нового содержимого. Создаст новый файл для записи, если не найдет с указанным именем. |
| a+ | Откроет для добавления нового содержимого. Создаст новый файл для чтения записи, если не найдет с указанным именем. |
| ab | Откроет для добавления нового содержимого (бинарный). Создаст новый файл для записи, если не найдет с указанным именем. |
| ab+ | Откроет для добавления нового содержимого (бинарный). Создаст новый файл для чтения записи, если не найдет с указанным именем. |

В нашем случае используем режим:

‘a’ – файл откроется для добавления нового содержимого, при этом создается новый файл с указанным ранее названием для записи, если не найдется с таким именем.

* Далее указывается кодировка – ‘utf-8’

UTF-8 - распространённый стандарт кодирования [символов](https://ru.wikipedia.org/wiki/%D0%9D%D0%B0%D0%B1%D0%BE%D1%80_%D1%81%D0%B8%D0%BC%D0%B2%D0%BE%D0%BB%D0%BE%D0%B2), позволяющий более компактно хранить и передавать символы [Юникода](https://ru.wikipedia.org/wiki/%D0%AE%D0%BD%D0%B8%D0%BA%D0%BE%D0%B4), используя переменное количество байт (от 1 до 4), и обеспечивающий полную обратную совместимость с 7-битной кодировкой [ASCII](https://ru.wikipedia.org/wiki/ASCII) (стандарт кодирования знаков латинского алфавита, цифр, некоторых специальных знаков и управляющих последовательностей, принятый в 1963 году [Американской ассоциацией стандартов](https://ru.wikipedia.org/wiki/%D0%90%D0%BC%D0%B5%D1%80%D0%B8%D0%BA%D0%B0%D0%BD%D1%81%D0%BA%D0%B8%D0%B9_%D0%BD%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%B8%D0%BD%D1%81%D1%82%D0%B8%D1%82%D1%83%D1%82_%D1%81%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82%D0%BE%D0%B2) как основной способ представления текстовых данных в ЭВМ).

UTF-8, по сравнению с [UTF-16](https://ru.wikipedia.org/wiki/UTF-16), наибольший выигрыш в компактности даёт для текстов на [латинице](https://ru.wikipedia.org/wiki/%D0%9B%D0%B0%D1%82%D0%B8%D0%BD%D0%B8%D1%86%D0%B0), поскольку латинские буквы без [диакритических знаков](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%B0%D0%BA%D1%80%D0%B8%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B5_%D0%B7%D0%BD%D0%B0%D0%BA%D0%B8), цифры и наиболее распространённые знаки препинания кодируются в UTF-8 лишь одним байтом, и коды этих символов соответствуют их кодам в [ASCII](https://ru.wikipedia.org/wiki/ASCII).

Общий вид строчки записи:

file = open('dates.csv', 'a', encoding = 'utf-8')

Указываем, какие именно данные записываем в файл:

* имя пользователя - {update.effective\_user.first\_name},
* ник пользователя в Telegram - {update.effective\_user.name},
* действие, которое выполнил пользователь (какую кнопку нажал) -{update.message.text},
* дата и время выполнения пользователем зафиксированного действия - {datetime.now()}.

Необходимо импортировать функцию datetime:

from datetime import datetime

общий вид строчки записи данных в файл:

file.write(f'{update.effective\_user.first\_name}, {update.effective\_user.name}, {update.message.text}, {datetime.now()}\n')

После открытия файла в Python его нужно закрыть. Таким образом освобождаются ресурсы и убирается «мусор». Python автоматически закрывает файл, когда объект присваивается другому файлу.

После закрытия этот файл нельзя будет использовать до тех пор, пока заново его не открыть.

file.close()

Общий вид функции работы с файлом:

async def loggg(update: Update, context: ContextTypes.DEFAULT\_TYPE):

    file = open('dates.csv', 'a', encoding = 'utf-8')

    file.write(f'{update.effective\_user.first\_name}, {update.effective\_user.name}, {update.message.text}, {datetime.now()}\n')

    file.close()

Используем функцию записи данных пользователя в файл при нажатии кнопки start в чат-боте – добавляем в функцию start вызов функции loggg:

async def start(update: Update, context: ContextTypes.DEFAULT\_TYPE):

    await loggg(update, context)

    await update.message.reply\_animation(animation='https://usagif.com/wp-content/uploads/gify/barbie-animated-usagif-22.gif')

    await update.message.reply\_text(package.view.start\_msg(update))

update - это объект связанный с экземпляром Update, который присылает и отправляет все сообщения. Через него можно получить доступ к экземпляру telegram.Bot() как update.bot.

context - это объект, связанный с контекстом обработанного сообщения. Через него также можно получить доступ к экземпляру telegram.Bot() как context.bot.

Контекст  - context, при регистрации любого [обработчика сообщений](https://docs-python.ru/packages/biblioteka-python-telegram-bot-python/tipy-obrabotchikov/) передается в функцию обратного вызова этого обработчика (в нее так же передается Update). У контекста есть экземпляр класса telegram.Bot, он связан с конкретным сообщением, которое попало в эту функцию обратного вызова.

ContextTypes - класс для сбора настраиваемых типов интерфейса telegram.ext.CallbackContext.

В том же порядке создаем функцию для сохранения данных пользователя, который выбрал пункт меню – отправить заявку. Это необходимо для оперативного получения информации о поступлении новой заявки. Получаем новую функцию:

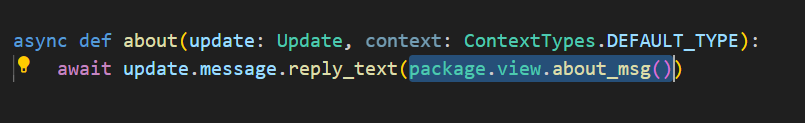
async def request\_file(update: Update, context: ContextTypes.DEFAULT\_TYPE):

    file = open('request.csv', 'a', encoding = 'utf-8')

    file.write(f'{update.effective\_user.first\_name}, {update.effective\_user.id}, {update.effective\_user.name}, {update.message.text}, {datetime.now()}\n')

    file.close()

Если выбранный пользователем пункт меню содержит только текст-пояснение, в функции указывается только ссылка на сообщение из файла view.py:



Также можно в боте показать гифку или фотографию, например, для приветствия. В этом случае необходимо добавить строчку с ссылкой на гифку или фотографию:

await update.message.reply\_animation(animation='https://usagif.com/wp-content/uploads/gify/barbie-animated-usagif-22.gif')

async def start(update: Update, context: ContextTypes.DEFAULT\_TYPE):

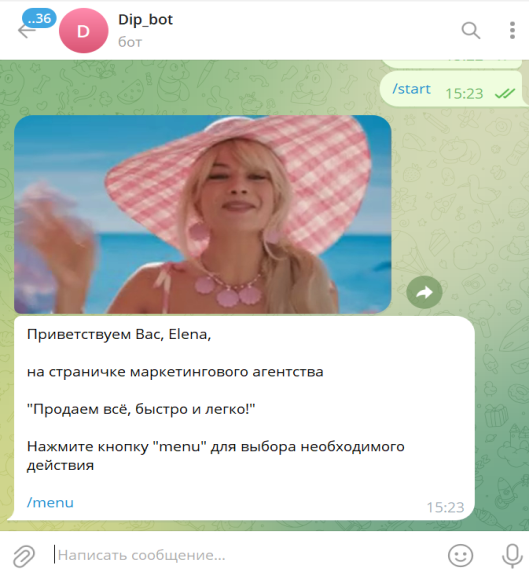
    await loggg(update, context)

    await update.message.reply\_animation(animation='https://usagif.com/wp-content/uploads/gify/barbie-animated-usagif-22.gif')

    await update.message.reply\_text(package.view.start\_msg(update))

При нажатии на кнопку start пользователь получит сообщение с приветственной гифкой и текстовым сообщением.

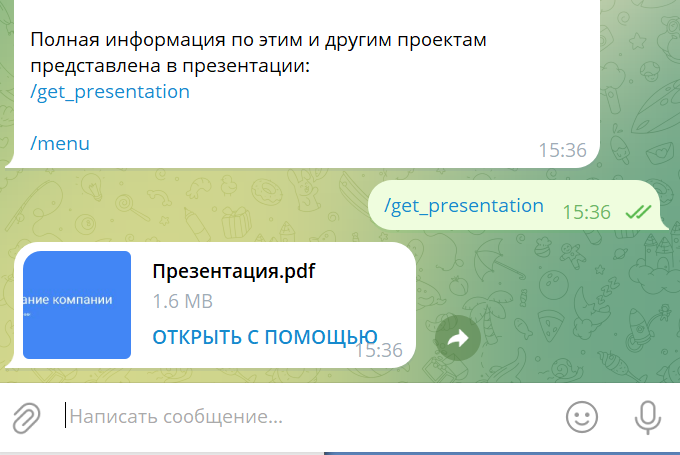
В боте это будет выглядеть так:



Также можно через бота отправить пользователю документ, например, презентацию, предварительно сохранив документ отдельным файлом:

async def get\_presentation(update: Update, context: ContextTypes.DEFAULT\_TYPE):

    await update.message.reply\_document(document=open('C:/Users/Asus/Desktop/Презентация.pdf', 'rb'))



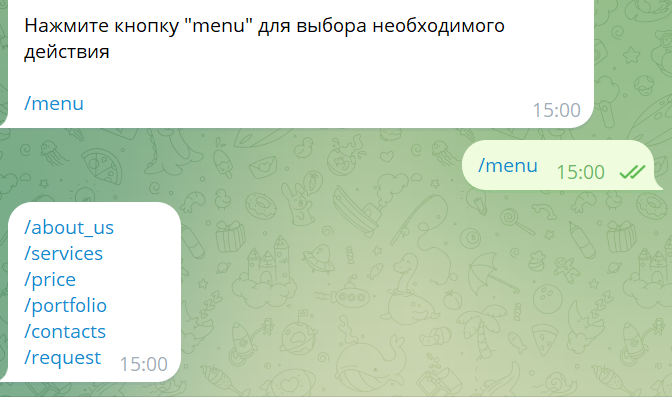
Полное меню чат-бота можно указать одной строкой –

f'/about\_us\n/services\n/price\n/portfolio\n/contacts\n/request'

async def menu(update: Update, context: ContextTypes.DEFAULT\_TYPE):

    await update.message.reply\_text(f'/about\_us\n/services\n/price\n/portfolio\n/contacts\n/request')

в Telegram это выглядит так:



Файл bot.py:

from telegram import Update

from telegram.ext import ContextTypes

from package.set import bot

import logging

import datetime

from datetime import datetime

import package.view

import package.set

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

async def loggg(update: Update, context: ContextTypes.DEFAULT\_TYPE):

    file = open('dates.csv', 'a', encoding = 'utf-8')

    file.write(f'{update.effective\_user.first\_name}, {update.effective\_user.name}, {update.message.text}, {datetime.now()}\n')

    file.close()

async def start(update: Update, context: ContextTypes.DEFAULT\_TYPE):

    await loggg(update, context)

    await update.message.reply\_animation(animation='https://usagif.com/wp-content/uploads/gify/barbie-animated-usagif-22.gif')

    await update.message.reply\_text(package.view.start\_msg(update))

async def about(update: Update, context: ContextTypes.DEFAULT\_TYPE):

    await update.message.reply\_text(package.view.about\_msg())

async def menu(update: Update, context: ContextTypes.DEFAULT\_TYPE):

    await update.message.reply\_text(f'/about\_us\n/services\n/price\n/portfolio\n/contacts\n/request')

async def information(update: Update, context: ContextTypes.DEFAULT\_TYPE):

    await update.message.reply\_text(package.view.information\_msg())

async def services(update: Update, context: ContextTypes.DEFAULT\_TYPE):

    await update.message.reply\_text(package.view.services\_msg())

async def portfolio(update: Update, context: ContextTypes.DEFAULT\_TYPE):

    await update.message.reply\_text(package.view.portfolio\_msg())

async def get\_presentation(update: Update, context: ContextTypes.DEFAULT\_TYPE):

    await update.message.reply\_document(document=open('C:/Users/Asus/Desktop/Презентация.pdf', 'rb'))

    await update.message.reply\_text(f'/menu')

async def price(update: Update, context: ContextTypes.DEFAULT\_TYPE):

    await update.message.reply\_text(package.view.price\_msg())

async def contacts(update: Update, context: ContextTypes.DEFAULT\_TYPE):

    await update.message.reply\_text(package.view.contacts\_msg())

async def send\_request(update: Update, context: ContextTypes.DEFAULT\_TYPE):

     await update.message.reply\_text(package.view.send\_request())

async def request\_file(update: Update, context: ContextTypes.DEFAULT\_TYPE):

    file = open('request.csv', 'a', encoding = 'utf-8')

    file.write(f'{update.effective\_user.first\_name}, {update.effective\_user.id}, {update.effective\_user.name}, {update.message.text}, {datetime.now()}\n')

    file.close()

async def request(update: Update, context: ContextTypes.DEFAULT\_TYPE):

    await update.message.reply\_text(package.view.request())

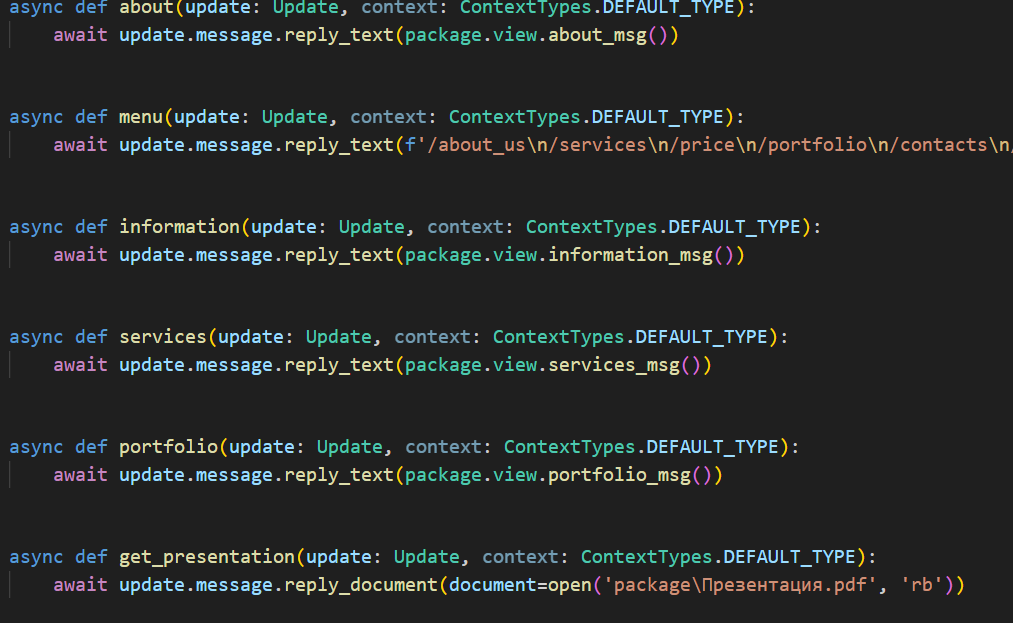
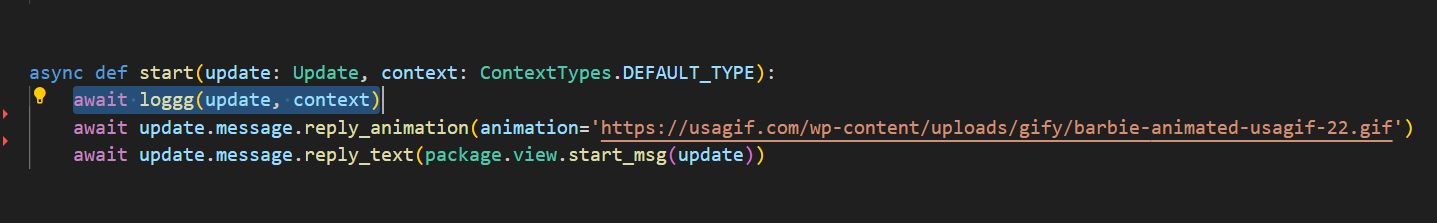
    await request\_file(update, context)

    await loggg(update, context)

async def call(update: Update, context: ContextTypes.DEFAULT\_TYPE):

    await update.message.reply\_text(package.view.call())

В VS Code выглядит так:

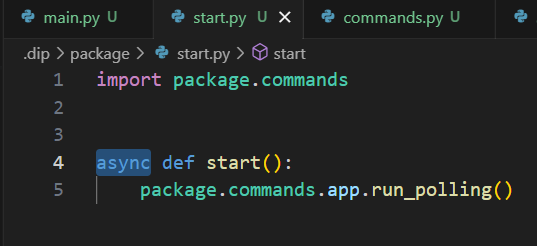


async – указывает, что эта функция может приостанавливать свое выполнение, чтобы выполнять другие задачи.

await используется только внутри асинхронной функции для вызова асинхронных операций, с операциями, выполнение которых требует времени. В связи с чем программа может продолжить выполнение других задач, а потом вернуться к исходной после завершения асинхронной операции.

Создаем файл start.py

Файл, в котором находится функция, запускающая работу бота:



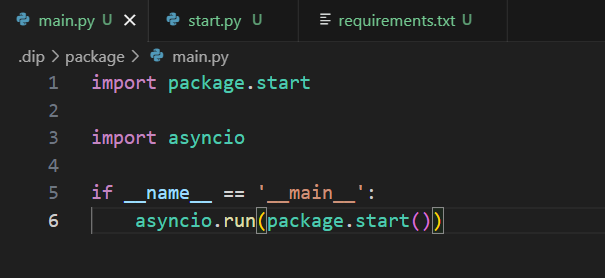
Вначале импортируем данные из файла commands, созданного в папке package – import package.commands.

Внутри функции start() указываем функцию run\_polling().

run\_polling() - метод, который заботится об инициализации и запуске приложения, опросе обновлений из Telegram с помощью telegram.ext.Updater.start\_polling() и плавном завершении работы приложения при выходе.

Создаем файл main.py

В этом файле будет запускаться функция, запускающая функцию работы чат-бота из файла start.py:



asyncio позволяет обрабатывать много задач одновременно, без блокировки потока исполнения, использует цикл событий, который обрабатывает различные задачи асинхронно и эффективно управляет ресурсами системы

Добавляем логическую проверку – if \_\_name\_\_ == ‘\_\_main\_\_’

В Python есть специальный прием, который позволяет указать, что какой-то код не должен выполняться при импорте: все строки, которые находятся в блоке if \_\_name\_\_ == '\_\_main\_\_' не выполняются при импорте.

В Python любая переменная, начинающаяся с двух символов нижнего подчеркивания (\_\_), является специальной. Это зарезервированные значения, которые выполняют определенную роль в программе.

\_\_main\_\_ указывает на область видимости, где будет выполняться код. Если запустить Python-файл прямо, то значением \_\_name\_\_ будет \_\_main\_\_. Если же его запустить в качестве модуля, то значением будет уже не \_\_main\_\_, а название модуля.

«Магическая» переменная \_\_name\_\_ содержит полное имя модуля. Это имя используется для уникальной идентификации модуля в системе импорта.

Это значит, что строка if \_\_name\_\_ == '\_\_main\_\_'  вернет True только в том случае, если программа будет запущена прямо.

Если же ссылаться на файл как на модуль, то содержимое конструкции if не будет выполнено.

Как правило, в блок if \_\_name\_\_ == '\_\_main\_\_' заносят все вызовы функций и вывод информации на стандартный поток вывода.

**Выводы**

В настоящее время чат-боты активно используются различными компаниями и физическими лицами, занимающимися продажей товаров и услуг в различных направлениях бизнеса. Они упрощают общение с клиентами и сокращают затраты на продажу товаров и услуг бизнеса, и упрощают процесс заказа товаров или услуг пользователями, а также экономят время пользователей (не нужно тратить время на дозвон, прослушивать не нужную информацию о дополнительных опциях и т.п.) и время сотрудников компаний на ответы на однотипные вопросы. Чат-боты активно используются государственными компаниями. Также чат-боты экономят трафик пользователей, т.к. трафик на общение в мессенджерах во многих тарифах предоставляется бесплатно, и в общем мессенджеры тратят меньше трафика, чем загрузка и изучение информации на сайтах.

Чат-боты довольно часто используются для проведения пользователя по разработанной бизнесом воронке продаж и перевода его в статус клиента с долгим LTV (lifetime value — это прибыль компании, полученная от одного клиента за всё время сотрудничества с ним. LTV отвечает на вопрос, сколько денег приносит клиент).

В данной работе реализован задуманный функционал в полном объеме. Разработанный чат-бот можно использовать для различных компаний и для личных целей, внеся небольшие правки в контактные данные и перечень предоставляемых услуг, расширив список однотипных вопросов, часто поступающих от пользователей.

Для компаний, занимающихся продажей товаров, услуг, доставкой чего-либо, которые не требуют дополнительной работы по переводу клиента из статуса потенциального в реального, которые имеют установленную цену, и пользователь может их просто выбрать и оплатить, можно доработать данный чат-бот, дополнив его возможностью выбора товаров/услуг из списка с указанием их стоимости, и переводом на страницу оплаты с оформлением и подтверждением заказа. Также можно добавить возможность записи на определенное время, например, для медицинских учреждений, для предпринимателей, предоставляющих услуги салонов красоты и т.п.

В целом чат-боты – полезный функционал для многих сфер жизни и деятельности.

**Используемые материалы**

Лекции и семинары GeekBrains (конспекты, презентации)

<https://docs.python-telegram-bot.org/>

https://appmaster.io/ru/glossary/token-api-ru

<https://proglib.io/>

<https://pythonru.com/>

https://vc.ru

<https://sergeysmirnovblog.ru>

https://wikipedia.org

https://www.geeksforgeeks.org/

<https://habr.com/ru>

(рабочий бот диплома - https://t.me/Dip\_bot\_bot)

1. принадлежит компании Meta, признана экстремистской организацией на территории РФ [↑](#footnote-ref-1)
2. Аббревиатура расшифровывается как Voice over Internet Protocol, в переводе – «голос по интернет-протоколу». Это метод с помощью которого выполняются и принимаются звонки через интернет. Кстати, эта технология не так нова. Телефония использует цифровые линии с конца 90-х годов. А VoIP – это экономичный способ обработки звонков без ограничений [↑](#footnote-ref-2)