

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ**

**ИНСТИТУТ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ**

КАФЕДРА АНАЛИЗА ДАННЫХ И ИССЛЕДОВАНИЯ ОПЕРАЦИЙ

Направление 38.03.05 «Бизнес-информатика (бакалавриат)

КУРСОВАЯ РАБОТА

**Разработка Telegram бота для изучения русского языка жестов и
дактиля**

Работа завершена: Обучающаяся гр. 09-701

«__»_____2020г. _____ / Э.Г.Якубова /

Работа допущена к защите:

Научный руководитель:

ассистент каф.теор.киб.

«__»_____2020 г. _____ / А.Ф.Хайруллин /

Заведующий кафедрой:

д.ф. – м.н.

«__»_____2020 г. _____ / М.Д. Миссаров /

Казань – 2020

Содержание

| | |
|--|----|
| Введение..... | 3 |
| Основная часть..... | 5 |
| 1. Основная информация о чат-бота Telegram..... | 5 |
| 2. Описание бота-помощника BotFather..... | 6 |
| 3. Язык программирования Python 3..... | 8 |
| 4. Среда разработки PyCharm..... | 9 |
| 5. Разработка бота с помощью pyTelegramBotAPI..... | 10 |
| 6. Размещение проекта на хостинге Heroku..... | 18 |
| Заключение..... | 21 |
| Литература..... | 22 |
| Приложение..... | 24 |

Введение

Общение является сложным процессом обмена информацией и развитием контактов между людьми или группами людей. Не найдется человека, который бы прожил всю жизнь без какой-то формы общения, будь то вербальное или невербальное, поскольку потребность в нем – одна из самых значимых человеческих потребностей.

Большинство людей используют речь для выражения собственных мыслей, передачи и приема знаний, влияния друг на друга. Иная же категория людей – глухие и слабослышащие, – для коммуникации используют жестовый язык. Вопреки популярному мнению, жестовые языки считаются вербальным видом общения, поскольку жестовый словарь, грамматика и другие лингвистические структуры подчиняются всем необходимым классификациям разговорных языков. Также многие слышащие люди принимают дактиль (пальцевый алфавит) за жестовый язык. Однако он передает лишь звуки речи (отдельные буквы) и чаще всего используется для имен собственных, вспомогательных слов и географических местоположений.

Слышащим людям нет необходимости в изучении жестового языка, но в жизни мы можем столкнуться с абсолютно разными личностями, и если возникает потребность в общении с людьми с нарушением слуха, то язык жестов может упростить взаимодействия, а также наладить связь. Однако стоит принимать во внимание, что нет общего мирового жестового языка и в каждой стране определенный жестовый язык со своей грамматикой и синтаксисом.

Так или иначе, несмотря на сопутствующие сложности, у каждого слышащего человека свои побуждающие мотивы для изучения языка жестов. Платформы для саморазвития имеют много разных форм. В эпоху современных технологий нет нужды записываться на тет-а-тет курсы по изучению языка, можно найти в Интернете любой сайт или приложения по собственному вкусу.

В России многие интернет-пользователи стали отдавать предпочтение ботам Telegram мессенджера, которые в последнее время только наращивают популярность, благодаря своей доступности и легкой навигации. Telegram сочетает в себе приложение для обмена сообщений, так же как и для получения определенной информации с помощью тех же самых специализированных ботов. Их функций и темы разнятся: можно посмотреть прогноз погоды, загрузить музыку, книги или, в моем случае, выучить язык.

Анализ сайтов и приложений показал, что эти платформы предлагают либо сразу большое количество жестов и правил, что может отпугнуть начинающего ученика, либо предлагают, например, только пальцевую азбуку. Более того, не было найдено ботов по данной теме, в связи с чем, данную работу можно считать актуальной. Нет необходимости самостоятельно искать информацию, бот предоставит нужные изображения как повседневных тем, так и так дактиля.

Целью данной курсовой работы является разработка Telegram бота для изучения общих повседневных выражений на языке жестов, а также дактиля для представление своего имени или иных имен собственных.

Для достижения поставленной цели мною были выделены нижеуказанные задачи:

1. Получение первоначальных знаний о языке жестов;
2. Проведение анализа аналогов приложений, сайтов, предлагающих курсы по изучению языков;
3. Изучение интерфейса для разработчиков ботов Telegram Bot API;
4. Разработка основного механизма на языке программирования Python 3.5 с помощью Telegram Bot API;
5. Тестирование и отладка разработанного приложения.

Основная часть

1. Основная информация о чат-ботах Telegram

Бот является сокращением от слова «робот». Это программа, которая автоматически выполняет действия в соответствии с заданными параметрами, командой или планом.

Боты в Telegram – это специальные аккаунты, у которых нет значка статуса (например, онлайн), вместо этого отличие от обычного пользователя заключается в том, что под названием пишется просто «бот». Пользоваться такими ботами легко и удобно, поскольку они отвечают на введенные команды пользователя, имитируя действия живого юзера.

Боты могут делиться на несколько подвидов:

- Чат-боты

Является обычным чатом, который старается «поддерживать» или имитировать реальный разговор на заданную пользователем тему.

- Боты-информаторы

Подобные боты являются своеобразной рассылкой, на которую подписываются пользователи. Например, с помощью них можно узнать новости, новые релизы или публикации.

- Боты-ассистенты

Разработанные онлайн-сервисами боты, и существующие как своеобразное дополнение к полноценной десктопной версии сайта

Так или иначе, нет четкого разделение на категории, потому что какие-то боты могут сочетать в себе, например, функционал чат-бота и бота-информатора. Это вполне исчерпывающе доказывает высокую популярность Telegram ботов.

2. Описание бота-помощника BotFather

Как было упомянуто ранее, для реализации моей идеи, заключающейся в разработке приложения для изучения русского языка жестов и дактиля, выбор пал на Telegram боты. Начать работу с ними легко, поскольку можно всего лишь ввести название бота в строку поиска и начать с ним общаться или найти ссылку на него из каталога, которая сразу перенаправит в Telegram.

Для начала работы необходимо добавить себе в контакт-лист бота-помощника BotFather. Он позволяет зарегистрировать неограниченное количество ботов.

Работа с этим ботом начинается с команды /newbot.

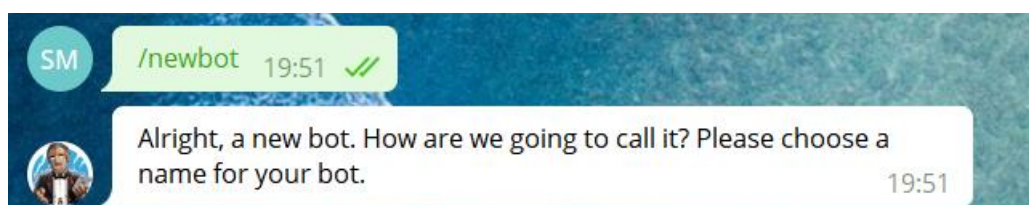


Рисунок 1. Создание бота с помощью BotFather

После чего он предлагает нам выбрать имя бота, которое будет отображаться в списке всех чатов, а также в контактах. Название бота можно в любой момент будет сменить.

Для упрощенного поиска бота назовем бот так:

RussianSignLanguage
бот

Рисунок 2. Название бота

Следующим этапом будет выбор юзернейма для бота. Обязательно, чтобы оно было уникальным и заканчивалось словом «bot».



Рисунок 3. Юзернейм бота

Если условия выполнены, то бот пришлет сообщением специальный токен, который необходим для работы с Bot API посредством http-протокола. Токен из соображений безопасности был скрыт, поскольку в ином случае любой человек сможет контролировать бота.

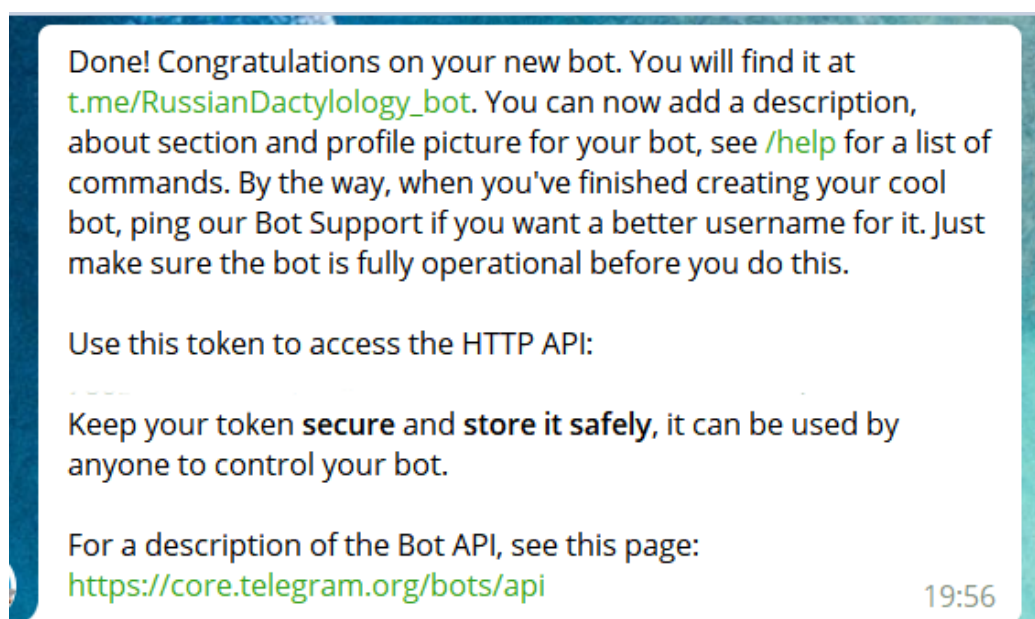


Рисунок 4. Получение токена бота

По окончании регистрации чат-бота Telegram, есть возможность применить различные команды для большей узнаваемости бота: установить описание, поменять картинку бота и так далее.

Edit Bots

`/setname` - change a bot's name
`/setdescription` - change bot description
`/setabouttext` - change bot about info
`/setuserpic` - change bot profile photo
`/setcommands` - change the list of commands
`/deletebot` - delete a bot

Bot Settings

`/token` - generate authorization token
`/revoke` - revoke bot access token
`/setinline` - toggle inline mode
`/setinlinegeo` - toggle inline location requests
`/setinlinefeedback` - change inline feedback settings
`/setjoingroups` - can your bot be added to groups?
`/setprivacy` - toggle privacy mode in groups

Games

`/mygames` - edit your games [beta]
`/newgame` - create a new game
`/listgames` - get a list of your games
`/editgame` - edit a game
`/deletegame` - delete an existing game

19:51

Рисунок 5. Команды для настройки бота

3. Язык программирования Python 3



Рисунок 6. Лого языка Python

Python 3 — язык программирования и мощный инструмент для создания программ. Является интерпретируемым, что позволяет упростить отладку программ, а также в нем используется динамическая типизация, то есть не нужно явно объявлять тип переменной.

Вместо того, чтобы встроить функциональную составляющую в ядро, создатели Python разработали его весьма расширяемым. Это сделало его

довольно популярным в плане добавления программируемых интерфейсов в существующие приложения.

У Python много готовых библиотек для решения конкретных типов задач. Ниже будут описаны несколько из них:

- NumPy

Библиотека, используемая для сложных математических вычислений.

- Pandas

Эта библиотека прекрасно подходит для анализа и работы с большими данными.

- Matplotlib

Массивная библиотека для построения графиков, корни которой растут из MATLAB.

Этот язык используется во многих сферах, но самыми главными направлениями, в которых он себя зарекомендовал являются веб-разработка (используются Python-фреймворки Django и Flask), машинное обучение, анализ данных (и их визуализация), автоматизация процессов.

4. Среда разработки PyCharm

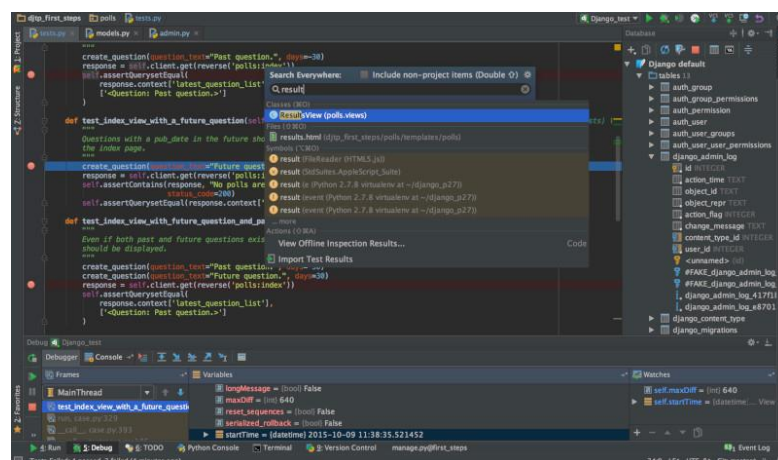


Рисунок 7. Интерфейс среды разработки PyCharm

PyCharm – это интегрированная кросс-платформенная среда разработки (IDE - Integrated development environment – комплекс программных средств, позволяющих вести разработку на определенном ЯП) для компьютерного программирования, в особенности на языке Питон. В нем есть все функции, необходимые для проведения анализа кода, отладки кода с графическим интерфейсом, подсветки ошибок, авто форматирования и многое другое. PyCharm предлагает API для разработчиков с помощью которого они смогут писать свои собственные плагины для расширения функционала PyCharm.

5. Разработка бота с помощью pyTelegramBotAPI

Так как в качестве языка программирования был выбран язык программирования Python 3.7.4 и среда разработки PyCharm, у нас есть возможность воспользоваться pyTelegramBotAPI. Для этого мы создаем проект и загружаем в него этот пакет. Следующим шагом будет импорт библиотеки telebot и определение переменной со своим токеном.

```
import telebot

from telebot import types

bot = telebot.TeleBot("786106722:AAGIQZSe-0owTMuvwBaGjUsOc7E2GROIW58")
```

После этого объявления нам необходимо зарегистрировать так называемые message handlers. Они определяют фильтры, которые текст сообщения должен обязательно пройти. Если сообщение прошло так называемый фильтр, то будет вызвана функция-декоратор, а входящее сообщение будет передано в качестве аргумента.

Следуя соглашению Telegram следует научить бота поступающей от пользователя команде /start.

```
@bot.message_handler(commands=["start"])

def start_message(message):

    bot.send_message(message.chat.id,
```

```

        'Привет, попробуем выучить русский язык жестов и дактиль.
Введи любую букву.' + '\n' + 'Напиши
'/help
'для
'большей
'информации ')

```

Напишем команду и посмотрим, что выведет нам бот:

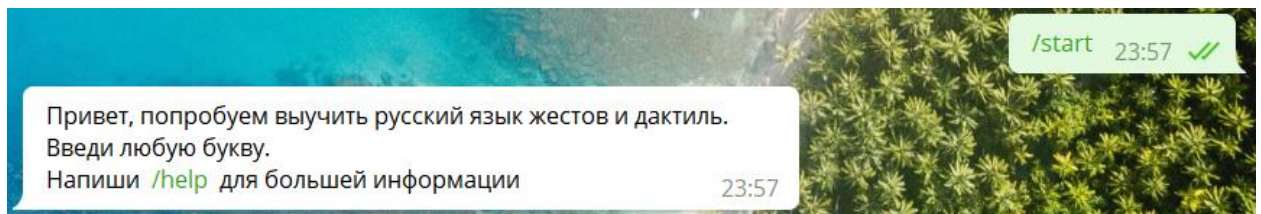


Рисунок 8. Результат работы команды /start

Как было заранее сказано, бот направлен на обучение пользователя языка жестов и дактиля. Если пользователь захочет выучить букву жестом, то ему всего лишь необходимо ввести эту букву алфавита и отправить сообщение боту. Бот в свою очередь пришлет картинку буквы с этим жестом. Регистр буквы в данном случае неважен.

```

@bot.message_handler(content_types=['text'])

def send_words(message):

    for key in alphabet:

        if message.text.lower() == key:

            bot.send_photo(message.chat.id, alphabet[key])

```

Отправка картинки регулируется методом `send_photo`. Есть несколько способов это сделать: можно загрузить картинку на сервера Telegram и получить уникальный идентификатор картинки, можно напрямую передать ссылку в качестве параметра или открыть файл на чтение в двоичном режиме и передать в качестве второго параметра. В нашем случае был выбран второй вариант по причине более удобной реализации.

В чате Telegram бота эта команда будет выглядеть таким образом:

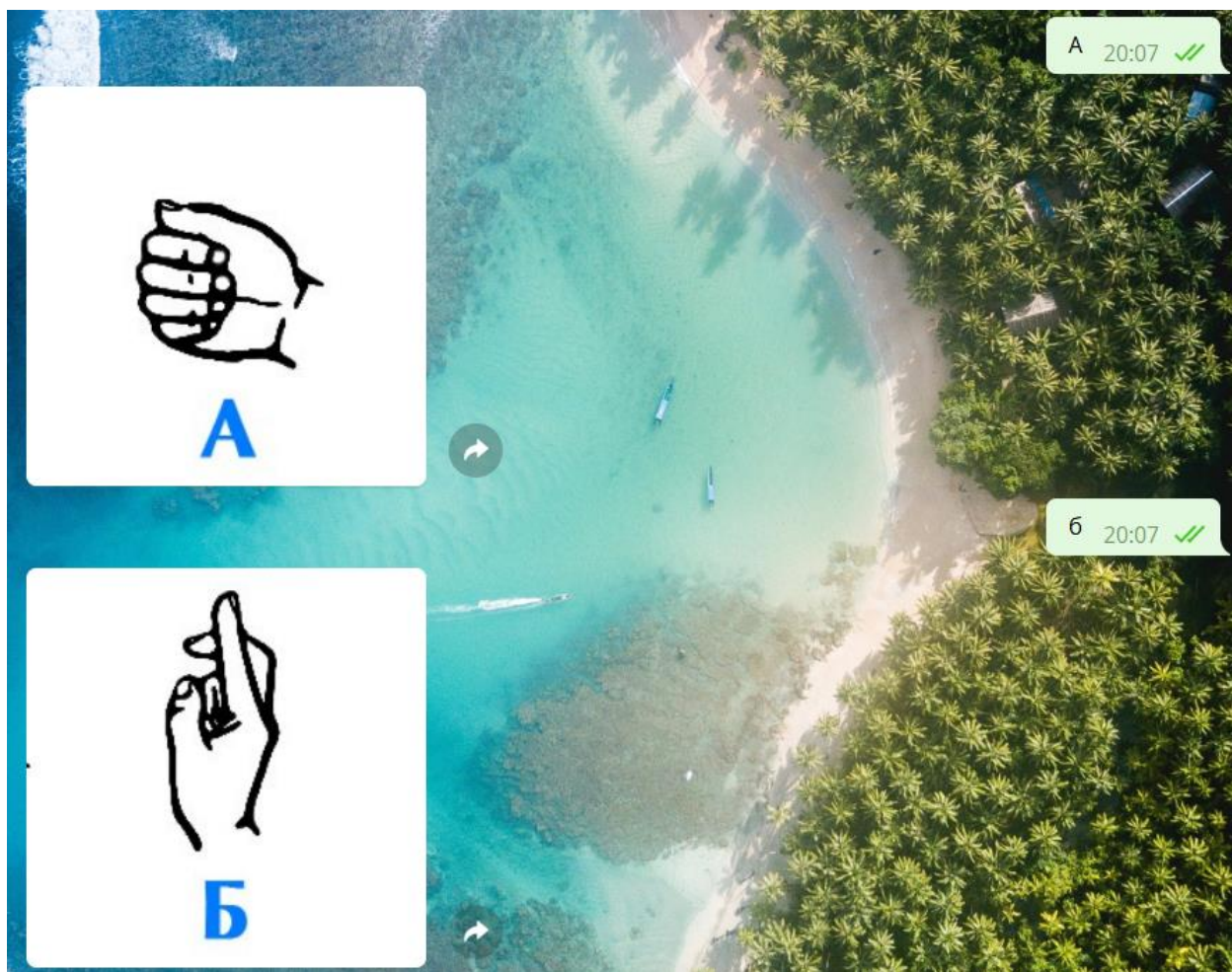


Рисунок 9. Жест буквы пальцевого алфавита

По соглашению Telegram также нельзя забывать о команде /help. Пользователи захотят увидеть список всевозможных команд, которые можно передать боту для обработки:

```
@bot.message_handler(commands=['help'])
def command_help(m):
    cid = m.chat.id
    help_text = "Доступные команды бота: \n"
    for key in commands:
        help_text += "/" + key + ": "
        help_text += commands[key] + "\n"
    bot.send_message(cid, help_text)
```

Метод `send_message` позволяет боту отправить текстовое сообщение. К качестве одного из параметров принимает `chat.id`, являющийся уникальным идентификатором для чата и пользователя чата. Вторым же параметром будет текст.

В чате Telegram бота эта команда будет выглядеть таким образом:

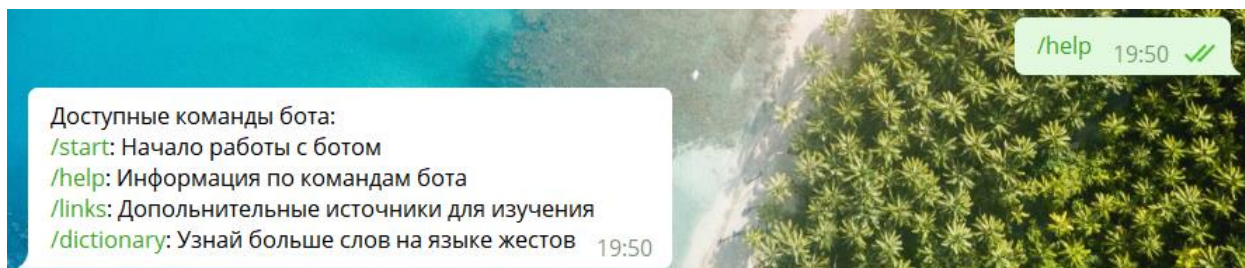


Рисунок 10. Результат работы команды `/help`

Данный бот не является полноценным в плане контента, поскольку предназначен лишь для получения базовых знаний. В связи с этим, если у пользователя появится желание дальше расширить свои познания, он может отправить команду `/links`. Она передаст пару надежных ресурсов для изучения языка жестов.

```
@bot.message_handler(commands=["links"])

def send_link(message):

    bot.send_message(message.chat.id,
        'https://www.memrise.com/course/1379220/russkii-zhestovyi-iazyk/')

    bot.send_message(message.chat.id, 'https://jestov.net')
```

В чате будет выглядеть подобным образом:

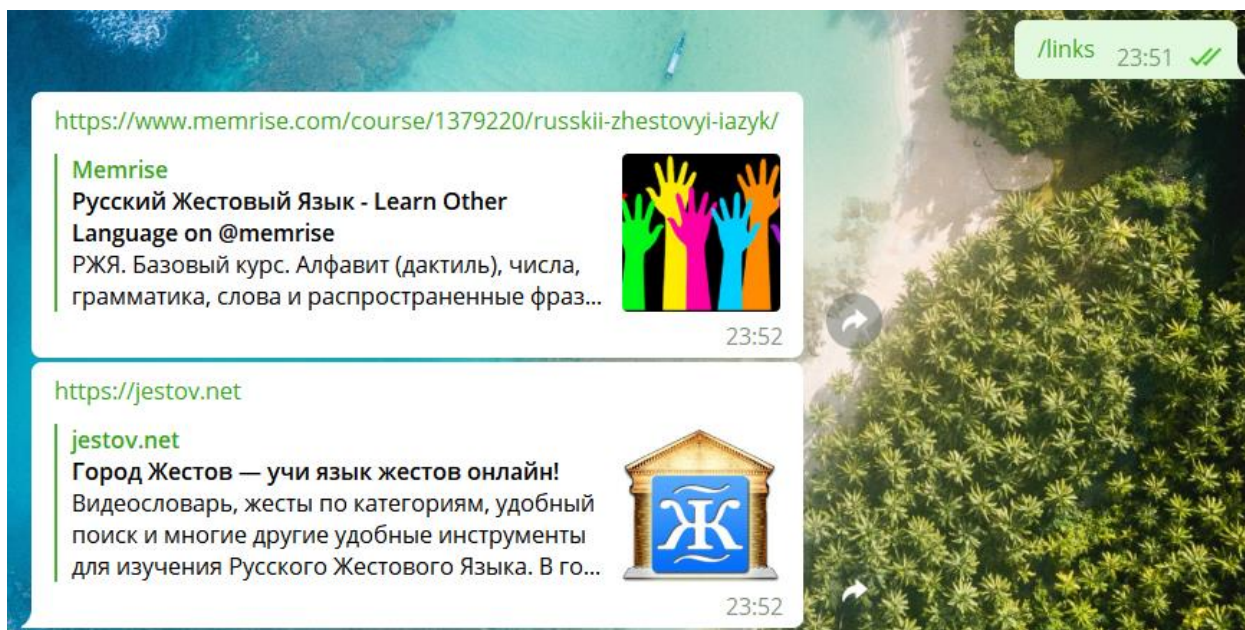


Рисунок 11. Результат работы команды /links

Для того, чтобы выучить слова определенной тематики на русском языке жестов была создана команда /dictionary. На данный момент реализовано четыре категорий: эмоции, общие фразы, идиомы и цвета. Чтобы пользователю был предоставлен наглядный выбор категории, с помощью InlineKeyboardMarkup была создана разметка клавиатуры. Далее мы рядом добавляем категорий, используя InlineKeyboardButton. Основные параметры при создании кнопки – text и callback_data: первый отвечает за текст на кнопке, второй – данные, которые будут переданы боту при выборе пользователем определенного варианта ответа.

```
@bot.message_handler(commands=["dictionary"])

def inline(message):

    keyboard = types.InlineKeyboardMarkup()

    keyboard.row(

        types.InlineKeyboardButton('Эмоции', callback_data='эмоции'),

        types.InlineKeyboardButton('Фразы', callback_data='фразы'),

        types.InlineKeyboardButton('Идиомы', callback_data='идиомы'),

        types.InlineKeyboardButton('Цвета', callback_data='цвета')

    )
```

```
bot.send_message(message.chat.id,
                  'Привет, ты можешь узнать больше слов. Выбери категорию!',
                  reply_markup=keyboard)
```

Запустим скрипт и напишем нашу команду:

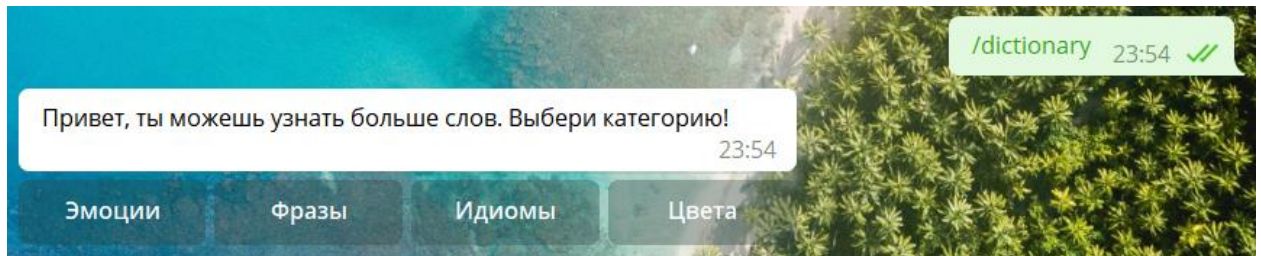


Рисунок 11. Результат работы команды /dictionary

Бот прислал варианты ответов. Но если мы нажмем на одну из кнопок, то ничего не произойдет, так как нам нужна обработка результатов. Идея будет заключаться в следующем: пользователь нажимает на нужную ему кнопку, бот возвращает список слов выбранной категории. В `call.data` будет передано значение, которое мы указывали при создании клавиатуры в параметре `callback_data`.

```
@bot.callback_query_handler(func=lambda call: True)
def revise(call):
    if call.data == "эмоции":
        key = ", ".join(list(emotions.keys()))
        bot.send_message(call.message.chat.id, key)
        bot.send_message(call.message.chat.id, "Напиши слово из выбранного списка!")

    elif call.data == "фразы":
        key = ", ".join(list(hello.keys()))
        bot.send_message(call.message.chat.id, key)
        bot.send_message(call.message.chat.id, "Напиши слово из выбранного списка!")

    elif call.data == "идиомы":
        key = ", ".join(list(idioms.keys()))
```

```

bot.send_message(call.message.chat.id, key)

bot.send_message(call.message.chat.id, "Напиши слово из выбранного
списка!")

elif call.data == "цвета":

    key = ", ".join(list(colors.keys()))

    bot.send_message(call.message.chat.id, key)

    bot.send_message(call.message.chat.id, "Напиши слово из выбранного
списка!")

```

Протестируем это с помощью нашего бота:

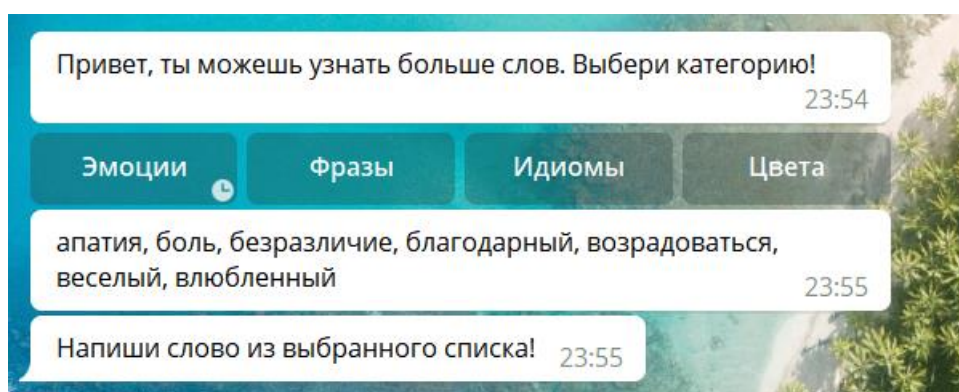


Рисунок 11. Отображение списка жестов категории «Эмоции»

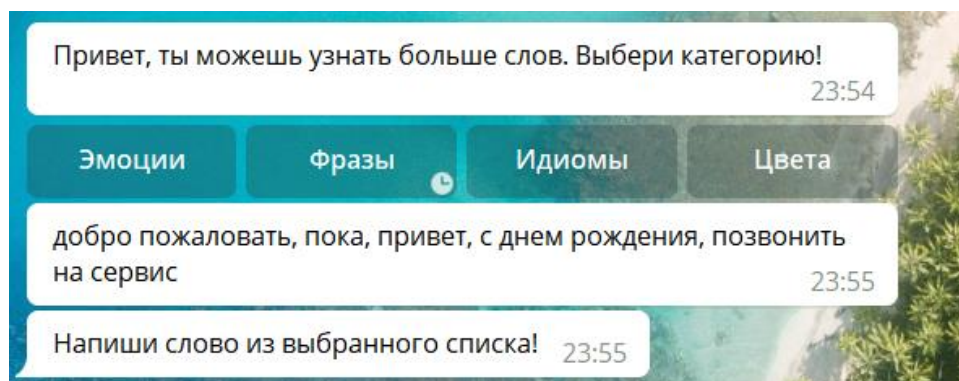


Рисунок 12. Отображение списка жестов категории «Фразы»

Все, что остается сделать пользователю – написать слово из выбранного списка. В этот раз только бот будет отправлять не статическую картинку, а gif-изображение, поскольку жестами изобразить целое слово намного сложнее, чем отдельную букву.

```
@bot.message_handler(content_types=['text'])
```



```
def send_words(message):
    for key in emotions:
        if message.text.lower() == key:
            bot.send_video(message.chat.id, emotions[key])

    for key in hello:
        if message.text.lower() == key:
            bot.send_video(message.chat.id, hello[key])

    for key in idioms:
        if message.text.lower() == key:
            bot.send_video(message.chat.id, idioms[key])

    for key in colors:
        if message.text.lower() == key:
            bot.send_video(message.chat.id, colors[key])
```

Алгоритм схож с методом `send_photo`, только передача gif-изображения регулируется методом `send_video`. Есть несколько способов это сделать: можно загрузить gif на сервера Telegram и получить ее уникальный идентификатор, можно напрямую передать ссылку в качестве параметра или с помощью встроенной функции `open` открыть файл на чтение в двоичном режиме, хранящееся на персональном компьютере. В нашем случае был выбран второй вариант по причине более удобной реализации.

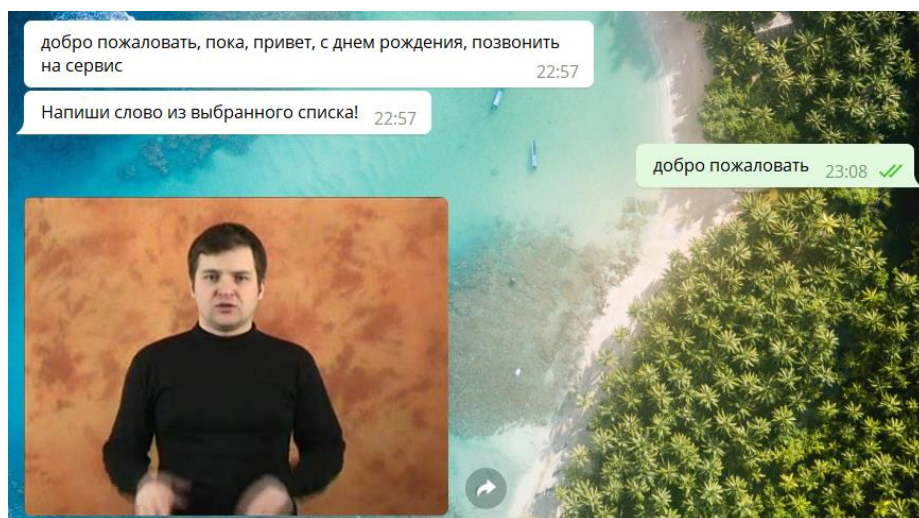


Рисунок 12. Отображение gif-изображения жеста

```
bot.polling(none_stop=True, interval=0)
```

Функция `polling` запускает т.н. Long Polling, а параметр `none_stop=True` говорит, что бот должен стараться не прекращать работу при возникновении каких-либо ошибок.

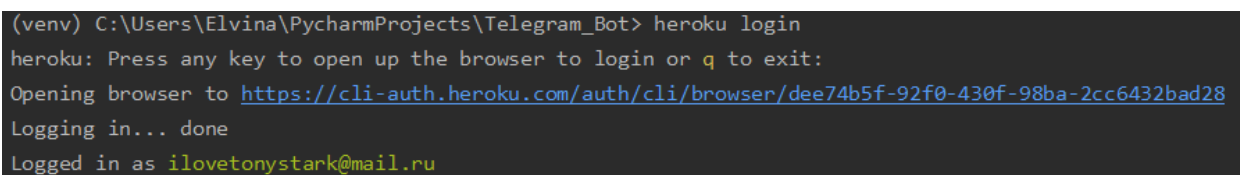
6. Размещение проекта на хостинге Heroku

Для бесперебойного доступа к нашему чат-боту, после его реализации необходимо развернуть его на удаленном сервере. В качестве хостинга была выбрана облачная PaaS-платформа, поддерживающая ряд языков программирования – Heroku.

Приложениям, которые запускаются на Heroku обычно присваиваются уникальные домены (например, "applicationname.herokuapp.com"), которые используются для отправки HTTP запросов к корректному контейнеру приложения, так называемые «dyno». Каждый dyno распределен по виртуальной сетке «dyno grid», состоящей из нескольких серверов. Heroku также имеет систему контроля версий Git.

Для деплоя нашего чат-бота необходимо скачать Heroku Command Line Interface (CLI) и Git. В терминал нашего проекта необходимо написать. Нас перекинет на сайт Heroku для авторизации.

heroku login



```
(venv) C:\Users\Elvina\PycharmProjects\Telegram_Bot> heroku login
heroku: Press any key to open up the browser to login or q to exit:
Opening browser to https://cli-auth.heroku.com/auth/cli/browser/dee74b5f-92f0-430f-98ba-2cc6432bad28
Logging in... done
Logged in as ilovetonystark@mail.ru
```

Рисунок 13. Успешная аутентификация через терминал

Следующим шагом будет создание двух файлов: `requirements.txt` и `Procfile`. Следует отметить, что у `Procfile` нет никакого расширения.

В `Procfile` напишем эту строку:

```
worker: python main.py
```

Это означает, что тип нашего dyno - worker будет работать с файлом main.py. При этом, будет работать как нам и надо – постоянно.

В файл requirements.txt пишем эту строку, которая является необходимым требованием для работы бота на сервере: pyTelegramBotAPI == 3.7.1

На этот раз в терминале прописываем Heroku create для создания проекта.

```
(venv) C:\Users\Elvina\PycharmProjects\Telegram_Bot> heroku create
Creating app... done, ● floating-ridge-35196
https://floating-ridge-35196.herokuapp.com/ | https://git.heroku.com/floating-ridge-35196.git
```

Рисунок 14. Создание приложения

Вот он появился в личном кабинете:

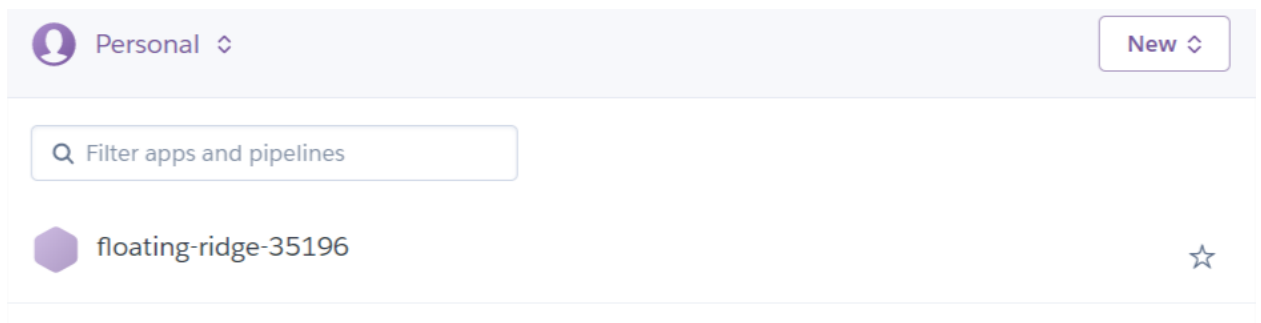


Рисунок 15. Личный кабинет на сайте Heroku

Далее, в терминале следует написать:

```
git add .
```

и сделать первый коммит:

```
git commit -am "make it better"
```

```
(venv) C:\Users\Elvina\PycharmProjects\Telegram_Bot>git add .

(venv) C:\Users\Elvina\PycharmProjects\Telegram_Bot>git commit -am "make it better"
[master 6ecc4c9] make it better
1 file changed, 12 insertions(+), 8 deletions(-)
```

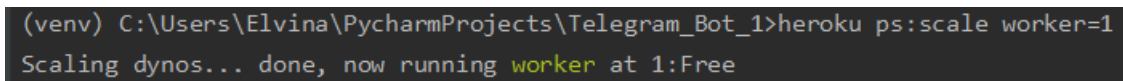
Рисунок 16. Первый коммит с помощью терминала

Так будет происходить деплой кода:

```
git push heroku master
```

Последним шагом будет эта строка, чтобы запустить worker dyno:

```
heroku ps:scale worker=1
```



```
(venv) C:\Users\Elvina\PycharmProjects\Telegram_Bot_1>heroku ps:scale worker=1
Scaling dynos... done, now running worker at 1:Free
```

Рисунок 17. Завершающий этап деплоя бота

За несколько шагов мы развернули наш бот на удаленном сервере, и теперь любой пользователь сможет получить стабильный доступ ко всем командам.

Заключение

В ходе курсовой работы был разработан полностью функционирующий Telegram чат-бот, разработанный с помощью программного интерфейса Bot API, помогающий людям получить базовое представление о русском языке жестов посредством картинок и gif-изображений.

Тестирование и отладка Telegram прошли успешно, все поставленные цели реализованы. В будущем библиотека жестов будет пополняться, чтобы прогресс пользователей был постоянным, а функционал станет более расширенным.

Литература

1. Доусон М. Програмируем на Python / М. Доусон – СПб.: Питер, 2019. – 416 с.
2. Как создать Telegram бота с помощью Python [Электронный ресурс] /. – Электрон. текстовые дан. – Режим доступа: <https://medium.com/nuances-of-programming/как-создать-telegram-бота-с-помощью-python-a80592e0ffe9>, свободный
3. Лутц М. Изучаем Python. Том 1 / М. Лутц – М.: Диалектика, 2019. – 832 с.
4. Официальный сайт Heroku. Облачная PaaS-платформа [Электронный ресурс] /. – Электрон. текстовые дан. – Режим доступа: <https://www.heroku.com/>, свободный
5. Пишем telegram-бота на python с помощью библиотеки telebot часть 1 [Электронный ресурс] /. – Электрон. текстовые дан. – Режим доступа: <https://habr.com/ru/post/448310/>, свободный
6. Пишем Telegram бота на Python + хостинг на Heroku [Электронный ресурс] /. – Электрон. текстовые дан. – Режим доступа: https://teletype.in/@cozy_codespace/Hk70-Ntl4, свободный
7. Простой Telegram-бот на Python за 30 минут [Электронный ресурс] /. – Электрон. текстовые дан. – Режим доступа: <https://habr.com/ru/post/442800/>, свободный
8. Словарь жестовых языков|SpreadTheSign [Электронный ресурс] /. – Электрон. текстовые дан. – Режим доступа: <https://www.spreadthesign.com/ru.ru/search/>, свободный
9. Создаем Telegram бота на Python часть-1 [Электронный ресурс] /. – Электрон. текстовые дан. – Режим доступа: <https://itvdn.com/ru/blog/article/telegrambot-1>, свободный
10. Толковый словарь специальных терминов на русском жестовом языке [Электронный ресурс] /. – Электрон. текстовые дан. – Режим доступа:

<http://82.137.156.202/safarov/jest/index.php?id=PLxZvL3C5rIhgYg8nrpeSERQBApPx2v8>, свободный

11. [eternnoir/pyTelegramBotAPI](https://github.com/eternnoir/pyTelegramBotAPI) / [GitGub](https://github.com) [Электронный ресурс] /. – Электрон.

текстовые дан. — Режим доступа:

<https://github.com/eternnoir/pyTelegramBotAPI>, свободный

Приложение

Листинг кода:

```
import telebot

from telebot import types

bot = telebot.TeleBot("786106722:AAGIQZSe-0owTMuvwBaGjUsOc7E2GROIW58")

commands = {

    'start': 'Начало работы с ботом',

    'help': 'Информация по командам бота',

    'links': 'Дополнительные источники для изучения',

    'dictionary': 'Узнай больше слов на языке жестов'

}

alphabet = {

    'a': 'https://abvgdee.ru/images/alfavit/gluhonemyh/a.png',

    'б': 'https://abvgdee.ru/images/alfavit/gluhonemyh/b.png',

    'в': 'https://abvgdee.ru/images/alfavit/gluhonemyh/v.png',

    'г': 'https://abvgdee.ru/images/alfavit/gluhonemyh/g.png',

    'д': 'https://abvgdee.ru/images/alfavit/gluhonemyh/d.png',

    'е': 'https://abvgdee.ru/images/alfavit/gluhonemyh/e.png',

    'ж': 'https://abvgdee.ru/images/alfavit/gluhonemyh/zh.png',

    'з': 'https://abvgdee.ru/images/alfavit/gluhonemyh/z.png',

    'и': 'https://abvgdee.ru/images/alfavit/gluhonemyh/i.png',

    'й': 'https://abvgdee.ru/images/alfavit/gluhonemyh/j.png',

    'к': 'https://abvgdee.ru/images/alfavit/gluhonemyh/k.png',

    'л': 'https://abvgdee.ru/images/alfavit/gluhonemyh/l.png',

    'м': 'https://abvgdee.ru/images/alfavit/gluhonemyh/m.png',
```



```

'н': 'https://abvgdee.ru/images/alfavit/gluhonemyh/n.png',
'o': 'https://abvgdee.ru/images/alfavit/gluhonemyh/o.png',
'п': 'https://abvgdee.ru/images/alfavit/gluhonemyh/p.png',
'р': 'https://abvgdee.ru/images/alfavit/gluhonemyh/r.png',
'с': 'https://abvgdee.ru/images/alfavit/gluhonemyh/s.png',
'т': 'https://abvgdee.ru/images/alfavit/gluhonemyh/t.png',
'у': 'https://abvgdee.ru/images/alfavit/gluhonemyh/u.png',
'ф': 'https://abvgdee.ru/images/alfavit/gluhonemyh/f.png',
'х': 'https://abvgdee.ru/images/alfavit/gluhonemyh/h.png',
'ц': 'https://abvgdee.ru/images/alfavit/gluhonemyh/ts.png',
'ч': 'https://abvgdee.ru/images/alfavit/gluhonemyh/ch.png',
'ш': 'https://abvgdee.ru/images/alfavit/gluhonemyh/sh.png',
'щ': 'https://abvgdee.ru/images/alfavit/gluhonemyh/shch.png',
'ъ': 'https://abvgdee.ru/images/alfavit/gluhonemyh/tvjordyj-znak.png',
'ы': 'https://abvgdee.ru/images/alfavit/gluhonemyh/y.png',
'ь': 'https://abvgdee.ru/images/alfavit/gluhonemyh/myagkij-znak.png',
'э': 'https://abvgdee.ru/images/alfavit/gluhonemyh/ee.png',
'ю': 'https://abvgdee.ru/images/alfavit/gluhonemyh/yu.png',
'я': 'https://abvgdee.ru/images/alfavit/gluhonemyh/ya.png',
}

```

```

emotions = { 'апатия':
'https://media.spreadthesign.com/video/mp4/12/26950.mp4',
        'боль': 'https://media.spreadthesign.com/video/mp4/12/4439.mp4',
        'безразличие':
'https://media.spreadthesign.com/video/mp4/12/296729.mp4',
        'благодарный':
'https://media.spreadthesign.com/video/mp4/12/43658.mp4',
        'возрадоваться':
'https://media.spreadthesign.com/video/mp4/12/296382.mp4',

```

```

        'веселый':
'https://media.spreadthesign.com/video/mp4/12/176715.mp4',

        'влюбленный':
'https://media.spreadthesign.com/video/mp4/12/295228.mp4',

    }

hello = { 'добро пожаловать':
'https://media.spreadthesign.com/video/mp4/12/18420.mp4',

        'пока': 'https://media.spreadthesign.com/video/mp4/12/100036.mp4',

        'привет': 'https://media.spreadthesign.com/video/mp4/12/17658.mp4',

        'с днем рождения':
'https://media.spreadthesign.com/video/mp4/12/295884.mp4',

        'позвонить на сервис':
'https://media.spreadthesign.com/video/mp4/12/134844.mp4',

    }

idioms = { 'слава богу':
'https://media.spreadthesign.com/video/mp4/12/321197.mp4',

        'счастливого пути':
'https://media.spreadthesign.com/video/mp4/12/227221.mp4',

        'держаться себя в руках':
'https://media.spreadthesign.com/video/mp4/12/133763.mp4',

        'будет сделано':
'https://media.spreadthesign.com/video/mp4/12/18542.mp4',

        'легко на помине':
'https://media.spreadthesign.com/video/mp4/12/315696.mp4'

    }

colors = { 'красный': 'https://media.spreadthesign.com/video/mp4/12/4495.mp4',

        'желтый': 'https://media.spreadthesign.com/video/mp4/12/5776.mp4',

        'зеленый': 'https://media.spreadthesign.com/video/mp4/12/5717.mp4',

```

```

        'темнота':
'https://media.spreadthesign.com/video/mp4/12/349236.mp4',

        'белый': 'https://media.spreadthesign.com/video/mp4/12/4425.mp4'

    }

@bot.message_handler(commands=["dictionary"])

def inline(message):

    keyboard = types.InlineKeyboardMarkup()

    keyboard.row(

        types.InlineKeyboardButton('Эмоции', callback_data='эмоции'),

        types.InlineKeyboardButton('Фразы', callback_data='фразы'),

        types.InlineKeyboardButton('Идиомы', callback_data='идиомы'),

        types.InlineKeyboardButton('Цвета', callback_data='цвета')

    )

    bot.send_message(message.chat.id,

                      'Привет, ты можешь узнать больше слов. Выбери категорию!',

                      reply_markup=keyboard)

@bot.callback_query_handler(func=lambda call: True)

def revise(call):

    if call.data == "эмоции":

        key = ", ".join(list(emotions.keys()))

        bot.send_message(call.message.chat.id, key)

        bot.send_message(call.message.chat.id, "Напиши слово из выбранного списка!")

    elif call.data == "фразы":

        key = ", ".join(list(hello.keys()))

        bot.send_message(call.message.chat.id, key)

        bot.send_message(call.message.chat.id, "Напиши слово из выбранного списка!")

    elif call.data == "идиомы":

        key = ", ".join(list(idioms.keys()))

```

```

        bot.send_message(call.message.chat.id, key)

        bot.send_message(call.message.chat.id, "Напиши слово из выбранного
списка!")

    elif call.data == "цвета":

        key = ", ".join(list(colors.keys()))

        bot.send_message(call.message.chat.id, key)

        bot.send_message(call.message.chat.id, "Напиши слово из выбранного
списка!")

@bot.message_handler(commands=['help'])

def command_help(m):

    cid = m.chat.id

    help_text = "Доступные команды бота: \n"

    for key in commands:

        help_text += "/" + key + ": "

        help_text += commands[key] + "\n"

    bot.send_message(cid, help_text)

@bot.message_handler(commands=["start"])

def start_message(message):

    bot.send_message(message.chat.id,

                      'Привет, попробуем выучить русский язык жестов и дактиль.
Введи любую букву. ' + '\n' + 'Напиши '
'/help '
'для '
'большой '
'информации ' )

@bot.message_handler(commands=["links"])

def send_link(message):

    bot.send_message(message.chat.id,
'https://www.memrise.com/course/1379220/russkii-zhestovyi-iazyk/')

    bot.send_message(message.chat.id, 'https://jestov.net')

@bot.message_handler(content_types=['text'])

def send_words(message):

```

```

for key in emotions:
    if message.text.lower() == key:
        bot.send_video(message.chat.id, emotions[key])

for key in alphabet:
    if message.text.lower() == key:
        bot.send_photo(message.chat.id, alphabet[key])

for key in hello:
    if message.text.lower() == key:
        bot.send_video(message.chat.id, hello[key])

for key in idioms:
    if message.text.lower() == key:
        bot.send_video(message.chat.id, idioms[key])

for key in colors:
    if message.text.lower() == key:
        bot.send_video(message.chat.id, colors[key])

bot.polling(none_stop=True, interval=0)

```