

# Создание и запуск чат-бота

Методические рекомендации  
к занятию 2

*Сервисы Telegram, GigaChat и другие упоминаются только в образовательных целях и не являются рекламой.*

## Развитие бота: обработка команд, взаимодействие с пользователями и отправка стикеров в python-telegram-bot (90 мин.)

### Цели занятия

1. Научиться обрабатывать несколько различных команд (помимо `/start`): `/help`, `/info`, `/setname`, `/greet` и т. д.
2. Освоить работу с интерактивными элементами (Reply-клавиатуры и Inline-кнопки).
3. Научиться сохранять пользовательские данные в памяти (внутри словарей, в оперативной памяти) или базе данных.
4. Научиться узнавать `file_id` стикеров и отправлять их в сообщениях.

### Структура урока

1. Обзор предыдущего урока (10 мин.).
2. Обработка команд и интерактивность (20 мин.).
3. Работа с пользовательскими данными (25 мин.).
4. Отправка стикеров и получение их ID (15 мин.).
5. Интерактивная работа (15 мин.).
6. Обсуждение и вопросы (5 мин.).

**Общее время:** 90 мин.

## 1. Обзор предыдущего урока (10 мин.)

### Цели

- Вспомнить основные моменты из первого занятия (создание бота с помощью **BotFather**, настройка **python-telegram-bot**, базовая команда **/start**).
- Обсудить возникшие сложности и способы их преодоления.

### Шаги

#### 1. Обратная связь

- Попросите нескольких обучающихся коротко рассказать о том, как они добавили простейшего бота и команду **/start**.
- Узнайте, у кого возникали технические проблемы при установке или запуске и какие.
- Попросите обучающихся открыть свой код с прошлого занятия, так как он поможет им сегодня.

#### 2. Анонс новых тем

- Сообщите, что на этом занятии нужно будет расширить функциональность бота: обрабатывать несколько команд, выводить кнопки (Reply/Inline) и сохранять (или даже базово обрабатывать) данные пользователя.
- Дополнительно нужно будет поработать со стикерами в боте: научиться узнавать их идентификаторы и отправлять стикеры.

## 2. Обработка команд и интерактивность (20 мин.)

### Цели

- Научить обучающихся добавлять несколько команд в бота (**/help**, **/info**, **/setname**, **/greet**).
- Показать основы работы с Reply-клавиатурами и Inline-кнопками.

## 2.1. Обработка нескольких команд

Команды `/help` и `/info`

```
from telegram import Update
from telegram.ext import CommandHandler, ContextTypes

async def help_command(update: Update, context: ContextTypes.
    DEFAULT_TYPE):
    await update.message.reply_text("Я простой бот. Напиши
    /info, чтобы узнать больше.")

async def info_command(update: Update, context: ContextTypes.
    DEFAULT_TYPE):
    await update.message.reply_text("Я создан, чтобы помочь
    тебе. Попробуй написать что-нибудь!")
```

### 1. Регистрация обработчиков

В функции инициализации (например, `main()`) добавляем:

```
application.add_handler(CommandHandler("help", help_command))
application.add_handler(CommandHandler("info", info_command))
```

### 2. Проверка

Перезапустите бота в Telegram, отправьте `/help`, `/info` и убедитесь, что получаете ответы.

## 2.2. Интерактивные клавиатуры

Reply-клавиатура

```
from telegram import ReplyKeyboardMarkup, KeyboardButton

async def start_command(update: Update, context: ContextTypes.
    DEFAULT_TYPE):
    keyboard = [
        [KeyboardButton("Привет!"),
         KeyboardButton("Как дела?")]
    ]

    reply_markup = ReplyKeyboardMarkup(keyboard, resize_
    keyboard=True)
    await update.message.reply_text(
        "Привет! Нажми на кнопку:",
        reply_markup=reply_markup
    )
```

Здесь мы создаём две кнопки в разных строках («Привет!» и «Как дела?»).

## Inline-кнопки

```
from telegram import InlineKeyboardMarkup, InlineKeyboardButton

async def help_command(update: Update, context: ContextTypes.
    DEFAULT_TYPE):
    keyboard = [
        [InlineKeyboardButton("Перейти на сайт", url="https://
example.com")]
    ]
    reply_markup = InlineKeyboardMarkup(keyboard)
    await update.message.reply_text(
        "Нажми на кнопку для перехода на сайт:",
        reply_markup=reply_markup
    )
```

При нажатии на кнопку «Перейти на сайт» откроется ссылка.

## 3. Работа с пользовательскими данными (25 мин.)

### Цель

Показать, как «запомнить» имя пользователя и персонализировать обращение к нему.

### 3.1. Сохранение данных в памяти (словарь)

Словарь для хранения имени пользователя

```
user_data = {} # { user_id: name }
```

Команда `/setname <имя>`

```
async def setname_command(update: Update, context:
    ContextTypes.DEFAULT_TYPE):
    # /setname Имя
    args = context.args
    if args:
        name = " ".join(args)
        user_id = update.effective_user.id
        user_data[user_id] = name
        await update.message.reply_text(f"Отлично, {name}!
Теперь я тебя запомнил.")
    else:
        await update.message.reply_text("Пожалуйста, укажи своё
имя после команды /setname")
```

## Команда /greet

```
async def greet_command(update: Update, context: ContextTypes.  
DEFAULT_TYPE):  
    user_id = update.effective_user.id  
    if user_id in user_data:  
        name = user_data[user_id]  
        await update.message.reply_text(f"Привет, {name}! Чем я  
могу помочь?")  
    else:  
        await update.message.reply_text("Я не знаю твоего  
имени. Используй /setname <имя>.")
```

## Регистрация команд

```
application.  
add_handler(CommandHandler("setname", setname_command))  
application.  
add_handler(CommandHandler("greet", greet_command))
```

## 3.2. Сохранение в базе данных (необязательно)

- Если успеваете, покажите пример с SQLite.
- Объясните, что это позволяет хранить данные между перезапусками бота.

### SQLite для продвинутых

```
import sqlite3  
  
conn = sqlite3.connect('user_data.db')  
cur = conn.cursor()  
cur.execute("CREATE TABLE IF NOT EXISTS users (user_id INTEGER,  
name TEXT)")  
conn.commit()  
  
async def setname_command(update: Update, context:  
ContextTypes.DEFAULT_TYPE):  
    args = context.args  
    if args:  
        name = " ".join(args)  
        user_id = update.effective_user.id  
        cur.execute("INSERT INTO users (user_id, name) VALUES  
(?, ?)", (user_id, name))  
        conn.commit()  
        await update.message.reply_text(f"Привет, {name}!")  
    else:  
        await update.message.reply_text("Пожалуйста, укажи своё  
имя после /setname")
```

## Обсудите плюсы и минусы использования SQL

- Плюс: данные сохраняются между перезапусками бота.
- Минус: нужно следить за корректностью SQL-запросов и безопасностью (не допустить возможность [инъекций](#) в базу данных бота, которые могут привести к утечке информации).

## 4. Получение ID стикеров и их отправка (15 мин.)

### Цель

Научиться узнавать `file_id` стикеров и отправлять их в сообщениях.

### 4.1. Отправка стикеров

#### 1. Статический `file_id`

Когда вы уже знаете `file_id` стикера (например, заранее узнали его через тестовый бот), можете отправлять его так:

```
async def send_sticker_command(update: Update, context:
ContextTypes.DEFAULT_TYPE):
    sticker_file_id = "CAACAgIAAxkBAAET...<длинная строка>..."
    # Заранее сохранённый file_id
    await update.message.reply_sticker(sticker=sticker_file_id)
```

### Регистрация команды

```
application.
add_handler(CommandHandler("sendsticker", send_sticker_
command))
```

#### 2. Тест

Отправьте `/sendsticker` боту — он ответит вашим заготовленным стикером.

### 4.2. Получение `file_id` у присланного стикера

#### 1. Обработчик стикеров

Сделаем так, чтобы бот реагировал на любые стикеры от пользователя и присылал `file_id` в ответ:

```
from telegram.ext import MessageHandler, filters

async def sticker_handler(update: Update, context:
ContextTypes.DEFAULT_TYPE):
    sticker_obj = update.message.sticker
    if sticker_obj:
        sticker_id = sticker_obj.file_id
        await update.message.reply_text(f"Sticker ID:
{sticker_id}")
    else:
        await update.message.reply_text("Это не похоже
на стикер.")
```

### Регистрация

```
application.add_handler(MessageHandler(filters.STICKER,
sticker_handler))
```

## 2. Проверка

Отправьте боту любой стикер. Бот должен ответить строкой вида  
"CAACAqIAAxkBA..."

### Примечание

`file_id` может со временем меняться (особенно если стикер был перезагружен на серверах Telegram), но обычно работает достаточно долго.

## 5. Интерактивная работа (15 мин.)

### Цель

Дать возможность обучающимся самостоятельно объединить все новые навыки в одном боте.

### Задание

#### 1. Обязательная часть

- Добавить команды `/help`, `/info`, `/setname`, `/greet`, `/sendsticker`.
- Сделать Reply-клавиатуру, выводящую хотя бы одну кнопку.
- Если пользователь отправляет стикер — бот присылает `file_id`.
- Не забудьте попросить обучающихся сохранить прогресс за сегодня. Этот код понадобится и на следующем занятии.



## 2. Дополнительная часть

- Попробовать Inline-кнопку с `callback_data`.
- Организовать хранение имени пользователя в SQLite (для тех, кто уже знаком с SQL).

*Если осталось достаточно времени, предложите обучающимся протестировать ботов друг друга.*

### Поддержка

- Помогайте обучающимся при возникновении ошибок.
- Проверьте, у всех ли корректно работает отправка и приём стикеров (особенно если в группе используются разные версии Telegram).

## 6. Обсуждение и вопросы (5 мин.)

### Цели

- Подвести итоги, обсудить возникающие затруднения.
- Наметить пути развития бота.

### Шаги

#### 1. Демонстрации

Позвольте 1–2 обучающимся показать, как их бот отправляет стикер при команде `/sendsticker`, а также получает `file_id` при получении стикера от пользователя.

#### 2. Вопросы

Ответьте на вопросы обучающихся.

#### 3. Домашнее задание (опционально)

- **Задание 1.** Добавить несколько стикеров в ассортимент бота, использовать их с помощью разных команд.
- **Задание 2.** Создать более продвинутую Inline-кнопку, которая при нажатии отправляет в чат новый стикер.

## Итоги занятия

### 1. Что обучающиеся освоили

- Обработку нескольких команд (`/help`, `/info`, `/setname`, `/greet`, `/sendsticker`).
- Создание Reply-клавиатур и Inline-кнопок.

- Сохранение пользовательских данных (имя) в словаре или базе данных.
- Отправку стикеров и извлечение их `file_id` из полученных сообщений.

## 2. Практический результат

- Каждый обучающийся имеет бота, который умеет работать не только с текстом, но и со стикерами, а также может запоминать имя пользователя.
- У бота появились интерактивные элементы в виде кнопок, что делает взаимодействие более удобным и наглядным.

## 3. Перспективы

На следующем занятии можно попробовать подключать внешний API (например, погоду или курсы валют), расширять систему команд, обрабатывать callback от Inline-кнопок и создавать более сложные сценарии общения.