

Создание и запуск чат-бота

Конспект занятия 1

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

1. Введение в работу чат-ботов

1.1. Что такое чат-боты

Чат-бот — это программа, которая общается с людьми в мессенджерах или на сайтах.

- **Назначение**

Боты отвечают на вопросы, собирают данные от пользователя, предоставляют информацию, помогают оформлять заказы и даже развлекают в формате мини-игр или викторин, а также помогают в навигации.

- **Преимущества**

- Автоматизация и экономия времени — бот самостоятельно, без участия человека, отвечает пользователям.
- Масштабируемость — бот может одновременно общаться с большим количеством людей, он всегда на связи и отвечает на вопросы быстрее, чем человек.
- Возможность интеграции с другими сервисами: можно подключать базы данных SQL (SQLite или MySQL), ИИ (например, GigaChat), а также работать с API — программами, которые настраивают разные сервисы (например, погоды).

1.2. Где и как используются чат-боты

- **Мессенджеры**

Чат-боты быстро общаются с пользователями, отвечают на типовые вопросы и помогают снизить нагрузку на сотрудников техподдержки.

- **Сайты и сервисы**

Встроенные чат-боты консультируют клиентов, автоматически принимают заказы и отвечают на часто задаваемые вопросы.

- **Образование**

Боты рассылают учебные материалы, проводят тесты и напоминают о предстоящих уроках.

- **Медицина**

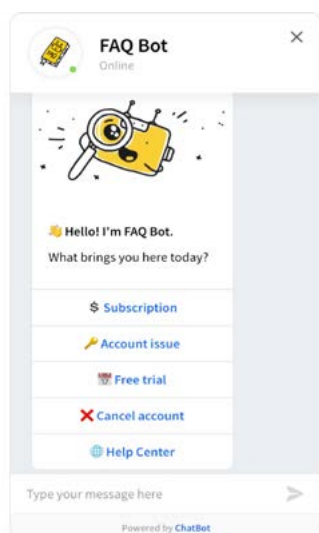
Чат-боты проводят первичный опрос пациентов, помогают записаться к врачу и уточнить расписание приёма.

- **Торговля**

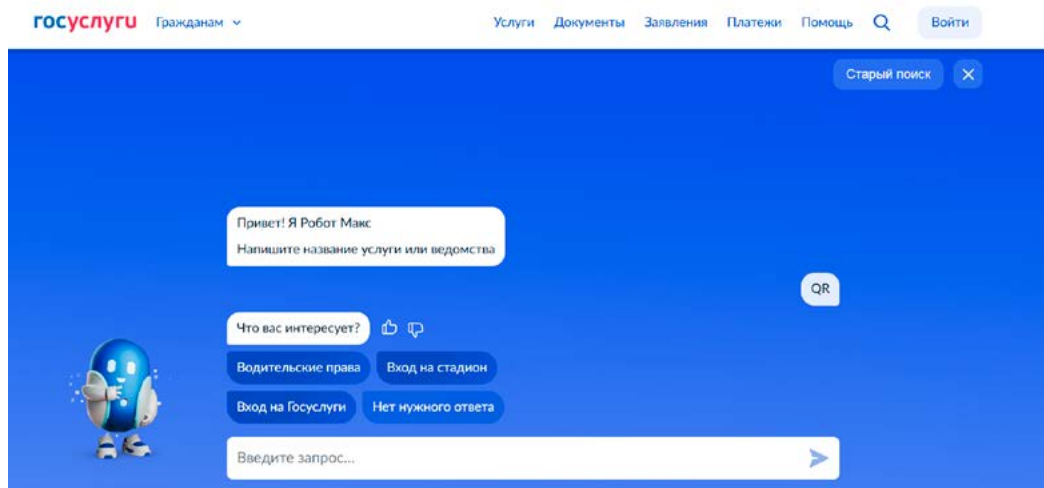
Боты помогают выбрать товары, оформляют заказы, уведомляют клиентов о статусе доставки.

1.3. Примеры ботов

- **Службы поддержки:** бот для FAQ (часто задаваемых вопросов), техподдержка провайдеров и интернет-магазинов.



- **Боты-навигаторы:** помогают находить нужные функции на сложных платформах.



1.4. Связь с искусственным интеллектом

- **NLP (Natural Language Processing):** технология, которая помогает ботам понимать естественный язык и «осмысленно» реагировать.
- **Машинное обучение:** боты могут обучаться на реальных диалогах, анализировать успешные/неудачные ответы и совершенствовать логику.
- **Продвинутые боты:** способны вести сложный диалог, предлагают ответы, глубже связанные с контекстом, подстраиваются под стиль общения пользователя и используют генеративные алгоритмы.

2. Подготовка рабочего окружения

Для создания Telegram-ботов мы будем использовать язык программирования Python. Для этого нам понадобится установить его и специальные библиотеки.

2.1. Установка Python

Проверка версии

Откройте терминал (командную строку) и введите:

```
bash
```

```
python --version
```

1. Если версия Python ниже 3.9 или Python не установлен, нужно перейти на [официальный сайт](#) и загрузить актуальную версию.
2. Установка Python:
 - Windows: запустите инсталлятор и в процессе установки убедитесь, что стоит галочка Add Python to PATH.
 - MacOS: чаще всего Python уже установлен, но, если нужно, также используйте установщик с сайта.
 - Linux: проверьте через пакетный менеджер (apt, dnf, pacman и др.).

2.2. Установка библиотеки python-telegram-bot

Откройте терминал и выполните команду:

```
bash
```

```
pip install python-telegram-bot==20.0
```

```
pip install nest_asyncio
```

pip — позволяет устанавливать библиотеки в Python.

Зачем это нужно?

Библиотека **python-telegram-bot** предоставляет удобный интерфейс для работы с [Telegram Bot API](#) (интернет-сервисы для взаимодействия с серверами —

удалёнными компьютерами). Она берёт на себя всю рутинную часть: приём и отправку сообщений, регистрацию команд и т. д.

2.3. Создание бота с помощью BotFather

1. Войдите в Telegram и найдите бота BotFather. Это официальный бот для управления другими ботами.
2. Наберите `/start`, а затем `/newbot`.
3. Придумайте для бота:
 - Имя (Name) — то, как он будет отображаться в контактах.
 - Юзернейм (Username) — то, как он будет зарегистрирован в системе. Юзернейм должен оканчиваться на `bot` (регистр не важен). Например, `MyFirstBot`.
4. Получите уникальный токен вида `1234567:ABC-....`. Никому не показывайте этот токен, потому что он даёт полный доступ к боту.

Подробное объяснение вы найдёте в [инструкции](#).

3. Создание первого бота

Теперь, когда у нас есть всё необходимое (Python, библиотека, токен), можно начать писать код нашего первого Telegram-бота.

3.1. Основы библиотеки `python-telegram-bot (v.20+)`

Это список классов, которые понадобятся в разработке. Мы импортируем их в начале скриптов.

- **`ApplicationBuilder`** — современный способ инициализировать бота.
- **`CommandHandler`** — «слушает» команды типа `/start`, `/help` и т. д.
- **`MessageHandler`** — «слушает» обычные сообщения без команды. Может реагировать на приветствие или любые другие тексты.

3.2. Структура проекта

Обычно удобнее хранить бота в отдельной папке:

```
my_telegram_bot/  
├── bot.py  
└── requirements.txt
```

- `bot.py` — основной файл с кодом бота.
- `requirements.txt` — список зависимостей (библиотек), которые нужны для проекта. Туда можно добавить строчку `python-telegram-bot`.

3.3. Пример кода (python-telegram-bot v.20+)

Подробное объяснение вы найдёте в [мини-учебнике](#).

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

# Включаем базовое логирование, чтобы видеть, что происходит
logging.basicConfig(
    format='%(asctime)s - %(name)s - %(levelname)s - %(message)s',
    level=logging.INFO
)

# Токен, полученный от BotFather (замените на свой)
BOT_TOKEN = "YOUR_BOT_TOKEN_HERE"

# Функция-обработчик команды /start
async def start_command(update: Update, context: ContextTypes.
DEFAULT_TYPE):
    # update.message содержит информацию о входящем сообщении
    # reply_text позволяет отправить ответ пользователю
    await update.message.reply_text("Привет! Я твой первый бот
на python-telegram-bot. Чем могу помочь?")

async def main():
    # Создаём приложение (бота) с вашим токеном
    app = ApplicationBuilder().token(BOT_TOKEN).build()

    # Регистрируем обработчик команды /start
    app.add_handler(CommandHandler("start", start_command))

    # Запускаем бота: bot будет постоянно «спрашивать» у
Telegram, есть ли новые сообщения
    await app.run_polling()

if __name__ == "__main__":
    import asyncio
    asyncio.run(main())
```

Этот код создаёт простого бота для Telegram, который отвечает на команду `/start`.

Код можно скопировать и демонстрировать на экране во время занятия.

Пояснения к коду

- `logging.basicConfig(...)` — настройка логов. Если что-то пойдёт не так (например, в случае если вы не сможете подключиться к боту), в терминале будут выводиться сообщения.
- `BOT_TOKEN = "YOUR_BOT_TOKEN_HERE"` — замените на реальный токен, выданный BotFather.
- `start_command()` — асинхронная функция (ключевое слово `async`), которая отвечает на команду `/start`.
- `await app.run_polling()` — бот в режиме *polling* каждые несколько секунд проверяет новые сообщения от пользователей в Telegram и отправляет ответы.

3.4. Запуск бота

1. Убедитесь, что вы находитесь в папке `my_telegram_bot` (где хранится `bot.py`).

В терминале запустите бота (он будет работать, пока открыт и запущен терминал):

```
bash
```

```
python bot.py
```

2. Если всё в порядке, в терминале вы увидите служебные лог-сообщения (INFO и т. д.). В случае если программа не запустится, вы увидите сообщение об ошибке.

Откройте Telegram, найдите своего бота по юзернейму. Введите `/start`. Бот должен ответить:

```
Привет! Я твой первый бот на python-telegram-bot. Чем могу помочь?
```

На этом этапе у вас уже должен работать простой бот.

3.5. Возможные трудности и их решение

1. **BotFather не выдаёт токен**

Убедитесь, что правильно ввели команды в BotFather и придумали корректный юзернейм (уникальный, оканчивающийся на `bot`).

Сохраните токен, он понадобится на следующих занятиях.

2. **Не установлена библиотека `python-telegram-bot`**

Проверьте `pip show python-telegram-bot` или `pip list`. Если библиотека не отображается, значит, надо переустановить: `pip install python-telegram-bot`.

3. **Ошибка при запуске бота**

Проверьте версию Python (должна быть 3.9+).

Убедитесь, что правильно скопировали токен: без лишних символов и пробелов.

4. Часто возникающие вопросы и ответы на них

1. Почему возникает ошибка `ModuleNotFoundError: No module named 'telegram'`?

Библиотека не установлена. Выполните `pip install python-telegram-bot`.

2. Бот не отвечает на сообщения

Используйте правильную версию библиотеки (v.20+ может отличаться синтаксисом от более ранних). Проверьте, что вы зарегистрировали обработчики (CommandHandler, MessageHandler) и что `app.run_polling()` действительно запущен.

3. Бот не реагирует, в логах пусто

- Проверьте токен (точность копирования).
- Убедитесь, что бот запущен в активном терминале и не завершается с ошибкой.
- Если произошла ошибка, следуйте инструкциям ошибки.

4. Как запустить бота 24/7?

Нужно использовать удалённый сервер (Cloud.ru, Yandex Cloud, TimeWeb) или оставлять ваш локальный компьютер постоянно включённым. Это выходит за рамки базового урока, но на практике часто бота разворачивают на хостинге.

Если у вас установлены две версии Python (python и python3), каждая из них имеет свой `pip` (`pip` для python и `pip3` для python3), что может приводить к путанице при установке пакетов. Чтобы избежать ошибок, всегда проверяйте версию интерпретатора командой `python --version` или `python3 --version` и используйте соответствующий `pip` для установки модулей. Это гарантирует, что пакеты будут установлены именно в ту версию Python, которая вам нужна.

5. Итоги занятия

- **Основы.** Поняли, что такое чат-бот, для чего нужен, где применяется.
- **Настройка среды.** Установили Python и библиотеку `python-telegram-bot`.
- **Первый бот.** Создали простого Telegram-бота, который реагирует на `/start`.

6. Домашнее задание (опционально)

Задание. Добавьте в бота команду `/help`, чтобы он выдавал список доступных команд и краткую справку по ним.

Пример

```
async def help_command(update: Update, context: ContextTypes.  
    DEFAULT_TYPE):  
    await update.message.reply_text("Доступные команды:  
    \n/start — запустить бота\n/help — помощь")
```

Не забудьте зарегистрировать новую команду `/help`:

```
app.add_handler(CommandHandler("help", help_command))
```

Дополнительное задание. Напишите обработчик (error handler), который будет «ловить» все возникающие ошибки и выводить их в лог.

7. Дополнительно: ссылки на полезные ресурсы

Официальная документация

- [Документация python-telegram-bot \(v.20+\)](#).
- [Telegram Bot API](#).

Примеры и статьи

- [GitHub: репозиторий python-telegram-bot](#).
- [Примеры мини-проектов ботов, которые можно найти в открытых репозиториях](#).

Заключение

Поздравляем! Теперь у вас есть Telegram-бот, который умеет отвечать на команду `/start`. Продолжайте экспериментировать и добавлять новые функции.

На следующих занятиях вы узнаете, как добавлять интерактивные кнопки, работать с различными типами сообщений (изображения, аудио, геолокация) и подключать базы данных для хранения информации.