СОЗДАНИЕ TELEGRAM-БОТА С ПОМОЩЬЮ ЯЗЫКА ПРОГРАММИРОВАНИЯ PYTHON

Telegram давно перестал быть просто мессенджером – это целая платформа с огромным потенциалом, и боты играют в ней ключевую роль. Эти умные помощники способны автоматизировать множество задач, от предоставления новостей и прогноза погоды до управления умным домом и обработки заказов. Боты бывают самых разных типов: информационные, которые делятся свежими данными; интерактивные, предлагающие игры и опросы; полезные инструменты, упрощающие повседневные дела; и даже бизнес-боты, помогающие компаниям взаимодействовать с клиентами.

Сегодня мы создадим интерактивного бота для игры в «Камень-Ножницы-Бумага» и разберём шаг за шагом, как это сделать.

Инструментарий, который мы будем использовать при создании нашего бота, вы можете видеть на вижуал-борде ниже:

Перечисление инструментов, справа налево, верхний ряд: язык Python, основная библиотека для написания PyTelegramBotApi, текстовый редактор для кода Sublime Text

Перечисление инструментов, справа налево, нижний ряд: десктопное и мобильное приложение Telegram, бот @TheBotFather

Итак, приступим!

Для начала нам необходимо открыть десктопное или мобильное приложение Telegram. Мы в своей работе используем ПК, следовательно открываем Telegram desktop app.

Шаг 1

Рисунок 1

В строке поиска вводим юзернейм @BotFather (рисунок 1).

Этот бот – самый простой и интуитивно понятный способ для регистрации, настройки и управления вашими будущими ботами. Вам не понадобятся специальные навыки или знания программирования – BotFather сделает процесс максимально легким и доступным. С его помощью вы сможете зарегистрировать неограниченное количество ботов, что открывает простор для творчества и экспериментов. Главное условие – уникальность имени пользователя (username) для каждого бота. Это необходимо для корректной работы и идентификации вашего бота в Telegram.

В строке поиска Telegram введите `@BotFather`.

Вы увидите верифицированного бота с галочкой рядом с именем.

Нажмите на него, чтобы открыть чат.

Шаг 2

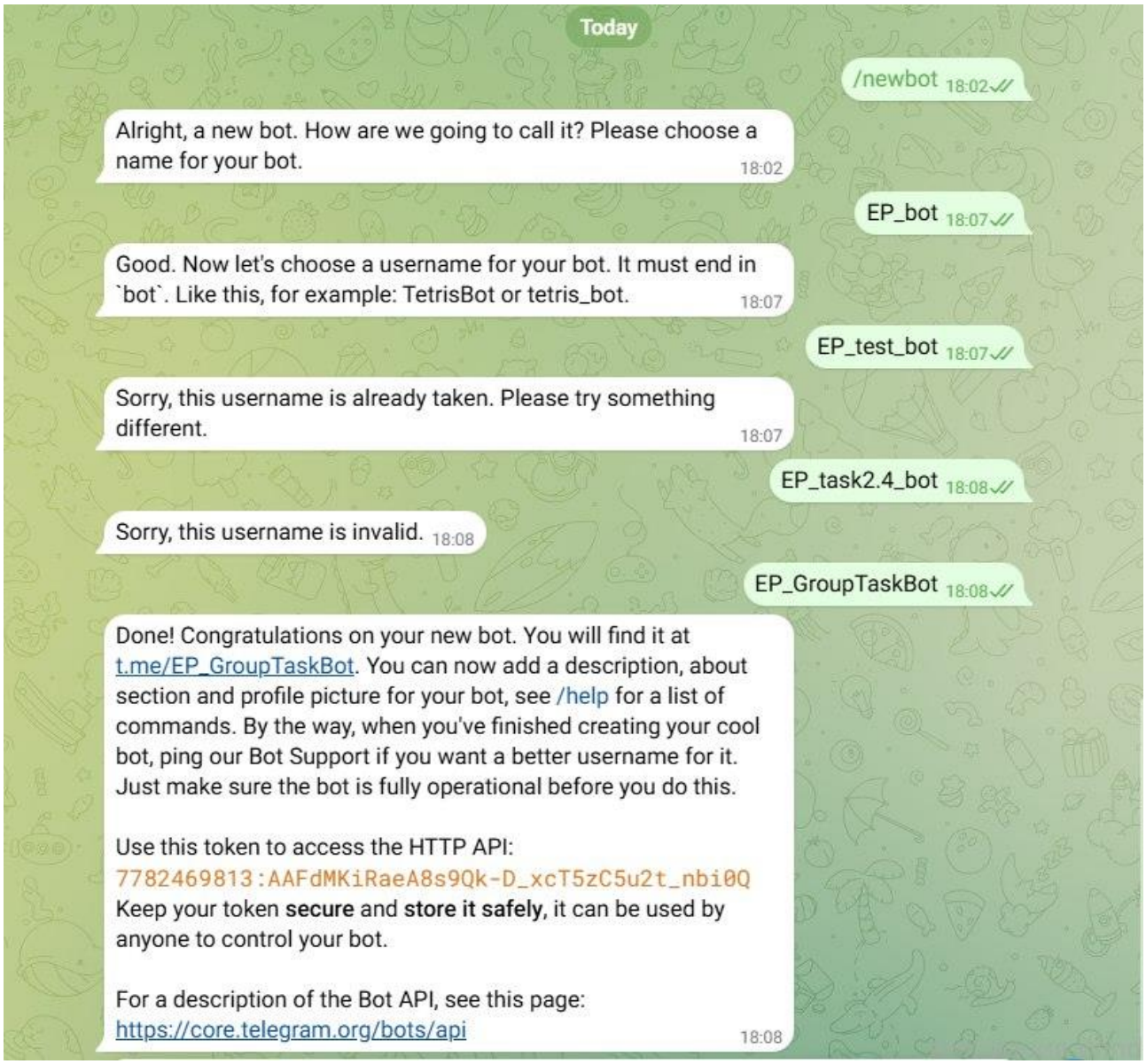


Рисунок 2

1. Запустите BotFather, нажав кнопку "Start" внизу экрана, если она есть. Если кнопки нет, просто введите команду «/start» в поле для ввода сообщения и отправьте ее. BotFather поприветствует вас и покажет список доступных команд.

2. Инициируйте создание нового бота. Из списка команд или просто введите и отправьте команду «/newbot». Это команда для BotFather, которая запускает процесс регистрации нового Telegram-бота.

3. Придумываем и задаем никнейм (username) для вашего бота.BotFather запросит у вас "a name for your bot". Это не имя бота, которое будут видеть пользователи, а именно ***никнейм (username).*** Username — это уникальный идентификатор бота в системе Telegram, который начинается с символа @ и используется для поиска бота и добавления его в группы.

Придумайте уникальный никнейм.

Никнейм должен быть:

- Уникальным: Такой никнейм еще не должен быть занят другим ботом в Telegram.
- Оканчиваться на "bot" или "_bot". Это обязательное требование Telegram для никнеймов ботов. Например: `MyTestBot`, `My_Amazing_Bot`, `UniversityHelperBot`.

Введите придуманный никнейм и отправьте его BotFather. Например, если вы придумали никнейм `MyFirstPracticalBot`, введите `MyFirstPracticalBot` и отправьте сообщение.

Обратите внимание на сообщение от BotFather.

Если никнейм доступен, BotFather сообщит об успехе и перейдет к следующему шагу.

Если никнейм уже занят, BotFather сообщит об этом и попросит вас придумать другой никнейм.

Попробуйте еще раз, изменив имя, пока не найдете уникальный вариант.

4. Задаем "имя" (name) для вашего бота.После успешного выбора никнейма, BotFather запросит: "Now let's choose a username for your bot. How are we going to call it?"

Вот здесь нужно задать ***имя (name)*** бота. Имя бота — это то, что пользователи будут видеть в списке чатов, в профиле бота и в уведомлениях. Оно

может быть любым, не обязательно уникальным и не должно заканчиваться на "bot".

Придумайте имя для вашего бота.Оно должно быть понятным и отражать назначение бота. Например: `Практический бот`, `Бот-помощник студента`, `Информационный бот`.

Введите придуманное имя и отправьте его BotFather.

5. Получаем токен для управления ботом. После ввода имени, BotFather поздравит вас с созданием нового бота! И самое главное – он пришлет вам токен (API token). Токен – это очень важный ключ доступа к вашему боту. Он выглядит как длинная строка символов, например: «123456789:ABCDEFG_HIJKLMNOPQRSTUVWXYZabcde».

Скопируйте и сохраните этот токен в надежном месте! Токен – это как пароль от вашего бота. Никому его не показывайте и не публикуйте в открытом доступе. Именно этот токен позволит вам и другим разработчикам (если вы будете работать в команде) управлять ботом, настраивать его функционал и подключать его к программному коду.

BotFather также предоставит вам ссылки и инструкции по дальнейшей настройке бота. Например, ссылки на документацию Telegram Bot API, команды для установки аватарки, описания и т.д. На данном этапе нам важен именно токен.

6. Подтверждение успешного создания. После получения токена, ваш бот зарегистрирован и готов к работе!

Вы можете найти его в Telegram по заданному ***никнейму (username, начинающемуся с @).***

Поздравлем, ваш первый Telegram-бот создан! Теперь вы можете перейти к следующему этапу – разработке функционала, используя полученный токен.

Шаг 3

Переходим в текстовый редактор для кода Sublime Text

Наш код

```
1  import telebot
2  from random import randint
```

Для начала импортируем библиотеку telebot для написания бота. Чтобы реализовать механизм игры импортируем функцию randint из модуля random.

```
5  API_TOKEN = "7782469813:AAFdMKiRaeA8s9Qk-D_xcT5zC5u2t_nbi0Q"
6
7
8  bot = telebot.TeleBot(API_TOKEN)
```

Записываем токен в отдельную переменную, чтобы его было удобнее использовать. Создем класс бота, используя токен

```
31 @bot.message_handler(commands=["start"])
32 def send_welcome(message):
33     bot.send_message(message.chat.id, """Я бот для "игры в камень, ножницы, бумагу".
34 Чтобы узнать, как играть со мной, введи команду /help""", reply_markup=markup)
35
36
37 @bot.message_handler(commands=["help"])
38 def send_welcome(message):
39     bot.send_message(message.chat.id, "Просто выбери то, чем будешь ходить, после этого я покажу, чем я сходил", reply_markup=markup)
```

Создаем функции для обработки команд /start и /help. С помощью декораторов указываем, что функция выполняется только при вводе вышеописанных команд. В самой функции отправляем сообщение, соответствующее введенной функции (/start приветствует, а /help описывает работу бота)

```
markup = telebot.types.ReplyKeyboardMarkup(row_width=2)

rock = telebot.types.KeyboardButton("\u270A")
paper = telebot.types.KeyboardButton("\u270C")
scissors = telebot.types.KeyboardButton("\u270B")

markup.add(rock, paper, scissors)
```

Создаем объект класса ReplyKeyboardMarkup, представляющий собой клавиатуру с определенными кнопками. В данном случае будут три кнопки с жестами "камень", "ножницы" и "бумага". Параметр row_width указывает, сколько кнопок будет в одной строке. Далее создаем эти кнопки и добавляем их на клавиатуру. Чтобы все работало, добавляем объект markup в качестве аргумента в параметр reply_markup при отправке сообщений

```
def beat(first, second):
    rock = "\u270A"
    paper =  "\u270B"
    scissors = "\u270C"

    if (first == rock and second == scissors) or (first == scissors and second == paper) or (first == paper and second == rock):
        return True

    return False
```

Напишем функцию, которая проверяет, выиграл ли первый второго (first, second). Если да, то возвращается True, если нет - False. Данная функция пригодится при описании основного функционала бота

```
@bot.message_handler(func=lambda message: message.text in ["\u270A", "\u270C", "\u270B"])
def reply(message):
    moves = ["\u270A", "\u270C", "\u270B"]
    bot_move = moves[randint(0, 2)]
    user_move = message.text
    bot.send_message(message.chat.id, bot_move, reply_markup=markup)

    if bot_move == user_move:
        bot.send_message(message.chat.id, "Ничья", reply_markup=markup)
    elif beat(user_move, bot_move):
        bot.send_message(message.chat.id, "Ты выиграл!", reply_markup=markup)
    elif not beat(user_move, bot_move):
        bot.send_message(message.chat.id, "Ты проиграл!", reply_markup=markup)
```

Создаем основную функцию reply. Декоратор выше указывает, что данная функция вызовется только в том случае, если передаваемое сообщение будет стикером жестов "камень", "ножницы" или "бумага". Далее бот выбирает случайный жест и с помощью условных операторов выясняется, кто победил

```
57    bot.infinity_polling()
```

Чтобы при запуске скрипта бот заработал, вызываем метод infinity_polling() нашего бота.

Важно! Бот будет работать до тех пор, пока его работа не будет остановлена принудительно (остановка работы, разрыв соединения, выключение ПК). Чтобы бот работал всегда, его необходимо разместить на хостинговый сервис. Так как это платная услуга, этот шаг мы пропустим. Перейдем к самой работе бота.

ПРОВЕРКА РАБОТОСПОСОБНОСТИ

Так как мы создавали бота на десктопе, проверим нашего бота на мобильной версии приложения.

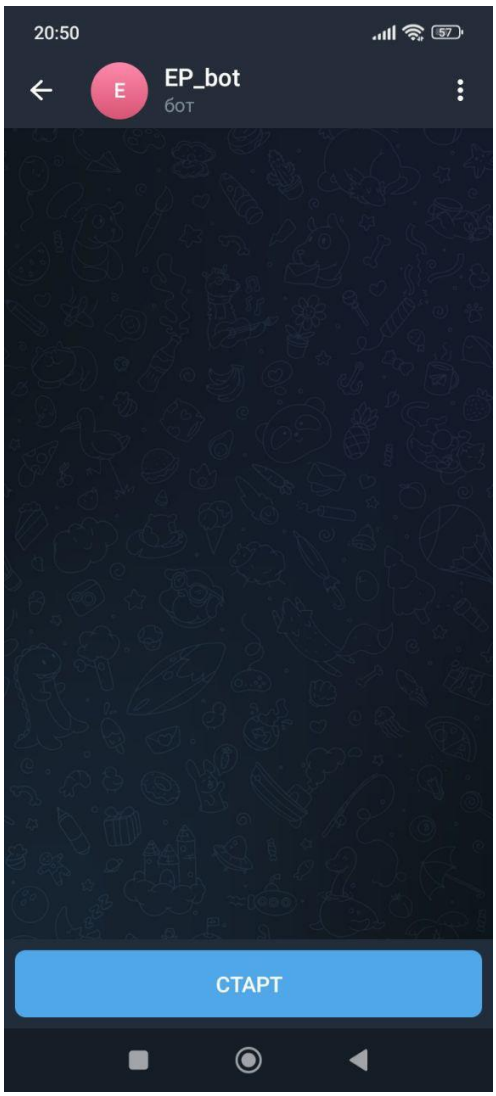


Рисунок 1

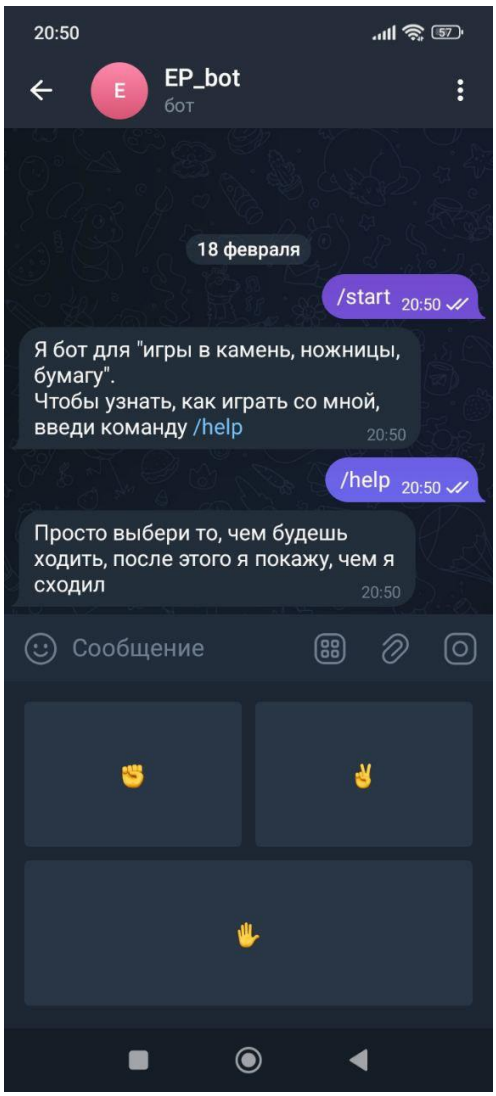
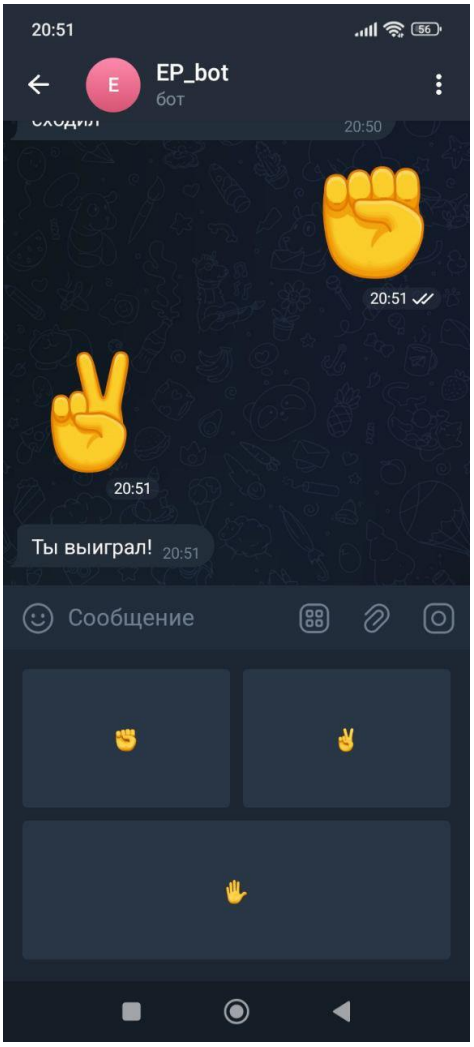


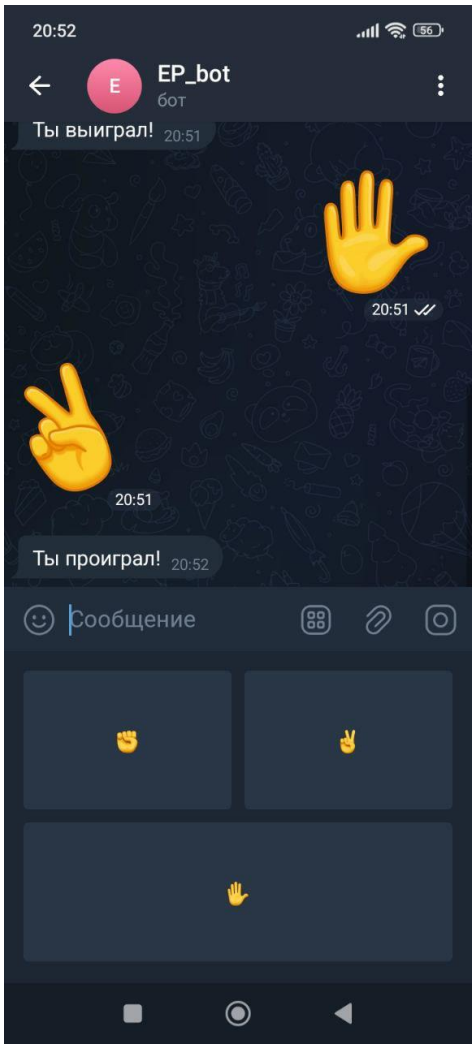
Рисунок 2

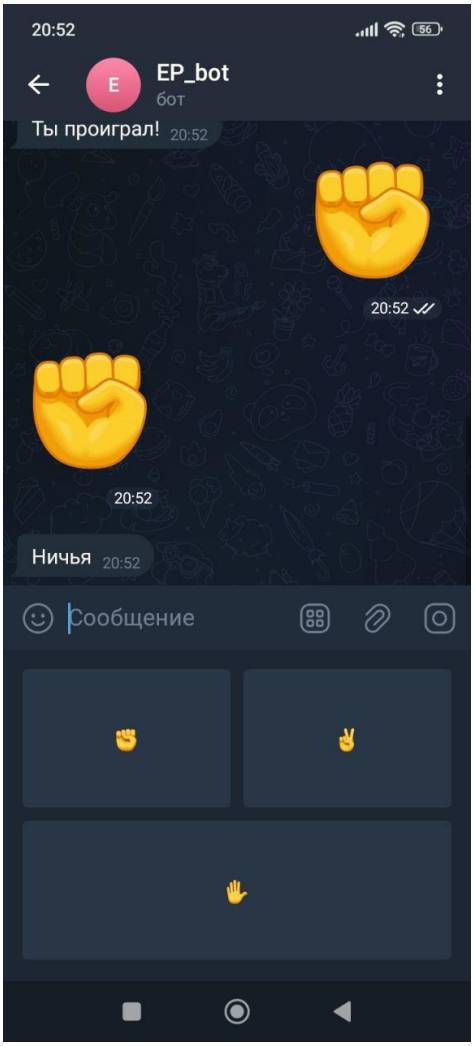
Находим нашего бота (рисунок 1). Найти бота можно по выбранному нами в начале юзернейму, а также напрямую перейдя из БотФазер в него.

Запустим его и проверим основные команды (/start, /help) (риуснок 2)

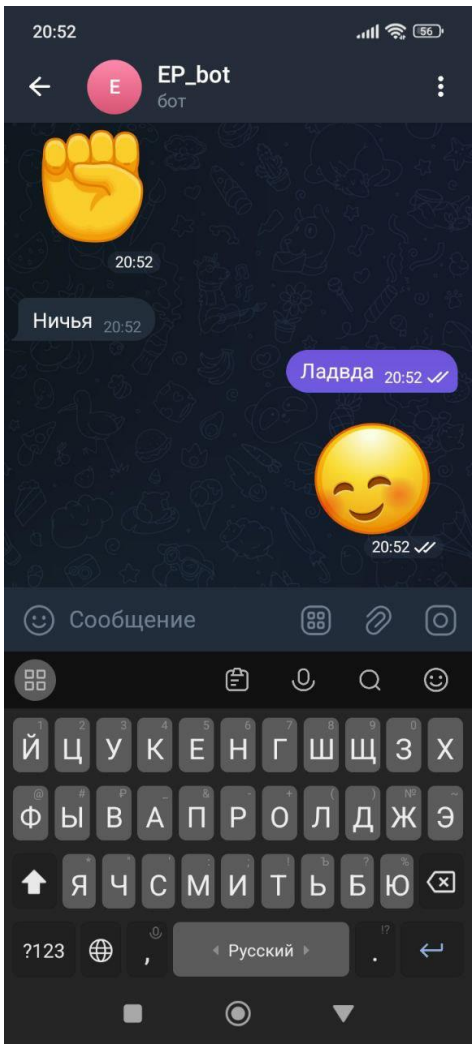
Попробуем сыграть!


20:51
EP_bot
бот
20:50
20:51
20:51
Ты выиграл! 20:51
Сообщение


20:52
EP_bot
бот
Ты выиграл! 20:51
20:51
20:51
Ты проиграл! 20:52
Сообщение


20:52
EP_bot
бот
Ты проиграл! 20:52
20:52
20:52
Ничья 20:52
Сообщение

Рисунки 1 — 3



При вводе текста или других эмодзи, наш бот не реагирует

Однако, если после этого ввод будет корректным, то всё так же работает

Таким образом, мы создали бота, ура!