A woman with short black hair is shown in profile, pointing her right index finger towards a glowing screen. The screen displays a grid of data points, and the entire scene is bathed in a blue and purple light. The text is overlaid on the right side of the image.

**БОТЫ**  
**на .Net**  
**информация и примеры**

# Боты

## Разновидности

Информационные боты

Боты обрабатывающие строго формализованные команды

Боты для автоматизации рабочих процессов  
(например, генерация задачи в TFS на основе письма об ошибке)

Поддерживающие естественный язык общения с  
клиентом/пользователем  
(Нейронные сети и другие технологии)

# Боты

## Web-приложение VS бот

В обоих случаях необходим middle-слой обработки запросов и сервер для его хоста.

### Web-приложение

Требуется затраты на разработку интерфейса, способного работать на разных браузерах и ОС. От разработчиков требуется высокая квалификация в части frontend-технологий.

Полный контроль внешнего вида приложения. Нестандартные кейсы

Полный контроль API

### Бот

Достаточно придумать названия команд. Всю визуальную часть и поддержку разных ОС и браузеров берёт на себя мессенджер.

Нет контроля внешнего интерфейса бота.

API может поменяться

# Бот для Telegram

## Способы реализации сервиса

Главный источник информации по созданию ботов

<https://core.telegram.org/bots>

Бот состоит из двух частей: интерфейса в Telegram и сервиса, обрабатывающий запросы.

Интерфейс в Telegram создаётся с помощью специального бота BotFather.

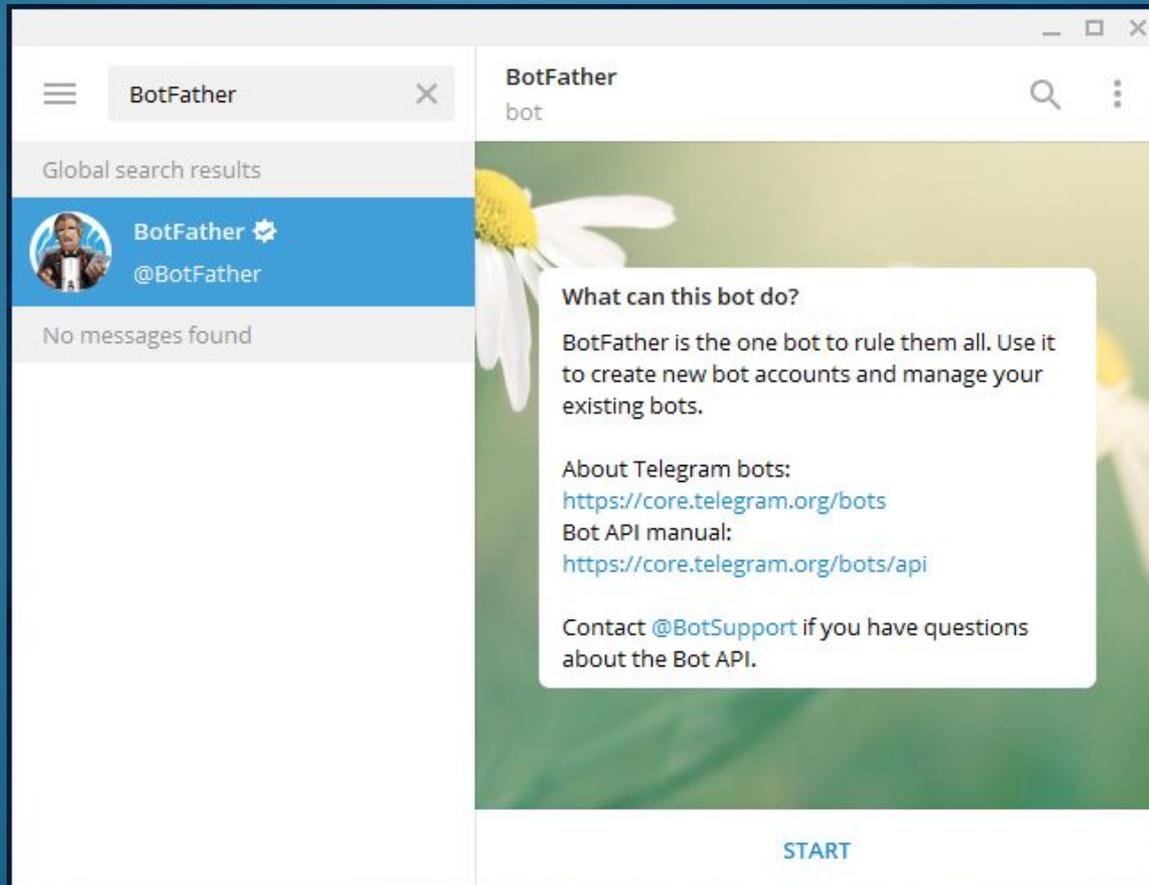
## Способы создания сервиса

1.  
Самописное SDK
2.  
Telegram.Bot готовое SDK на .Net (nuget-пакет Telegram.Bot)  
<https://github.com/TelegramBots/telegram.bot>
3.  
Bot Builder SDK for .NET (nuget-пакет Microsoft.Bot.Builder)

# Бот для Telegram BotFather

`\newbot` запуск создания бота

`\setcommands` изменение списка доступных команд бота



# Бот для Telegram

## Способы реализации сервиса

### getUpdates

При этом способе ваше приложение каждые 100мс (или реже) соединяется с сервером Telegram и опрашивает наличие изменений.

Минус подхода в том, что создается большая нагрузка на сервера Telegram.

Плюс в том, что он проще в реализации и тестировании, не нужно заморачиваться с SSL

### Webhook

В этом случае Telegram отправляет все изменения на указанный сервис, запущенный на определённом порту

Минус в том, что для тестирования необходим SSL-сертификат

Плюс в том, что бот, не нагружает Telegram, обрабатывая только реально произошедшие изменения

# Бот для Telegram

## Код getUpdates сервиса

```
class Program
{
    private static TelegramBotClient client;

    static void Main(string[] args)
    {
        const string token = "324323425:fad323A";
        client = new TelegramBotClient(token);
        client.OnMessage += BotOnMessageReceived;
        client.OnMessageEdited += BotOnMessageReceived;

        client.StartReceiving();
        Console.ReadLine();
        client.StopReceiving();
    }
}
```

# Бот для Telegram

## Код getUpdates сервиса

```
private async void BotOnMessageReceived(object sender, MessageEventArgs
messageEventArgs)
{
    var message = messageEventArgs.Message;
    if (message?.Type == MessageType.TextMessage)
    {
        await client.SendTextMessageAsync(message.Chat.Id, message.Text);
    }
}
```

# Бот для Telegram

## Код webhook сервиса

```
public sealed class BotService
{
    private static readonly Lazy<BotService> instanceHolder =
        new Lazy<BotService>(() => new BotService());
    private readonly TelegramBotClient client;

    private BotService()
    {
        const string token = "32423425:fadf343fRA";
        client = new TelegramBotClient(token);
        client.SetWebhookAsync("Сервер_сервиса").Wait();
    }

    public static BotService Instance { get { return instanceHolder.Value; } }

    public void Disconnect() { client.SetWebhookAsync().Wait(); }
}
```

# Бот для Telegram

## Код webhook сервиса

```
[Route("bot")]
public class BotController : Controller
{
    // POST bot/update
    [HttpPost]
    public async void Post([FromBody]Update update)
    {
        if (update == null) return;
        var message = update.Message;
        if (message?.Type == MessageType.TextMessage)
        {
            await BotService.Instance
                .SendTextMessageAsync(message.Chat.Id, message.Text);
        }
    }
}
```

# Инструменты проверки бота

## Ngrok

Ngrok (<https://ngrok.com/>) , генерируется https-адрес, который будет проксировать запросы на ваш сервис, запущенный на локальной машине.

Чтобы воспользоваться сервисом, нужно

- зарегистрироваться на сайте и получить персональный токен
- установить ngrok.exe и в командной строке ввести **ngrok authtoken ваштокен**
- после запуска сервиса, выполнить команду **ngrok http портсервиса**

Также если есть плагин к студии, то Ngrok можно запустить из меню Tools -> Start Ngrok tunnel

Студия должна быть при этом запущена в режиме администратора

```
ngrok by @inconshreveable
Session Status      online
Account             Maxim Walter (Plan: Free)
Version             2.2.8
Region              United States (us)
Web Interface        http://127.0.0.1:4040
Forwarding           http://2c94aea0.ngrok.io -> localhost:58141
Forwarding           https://2c94aea0.ngrok.io -> localhost:58141
Connections
  ttl    opn    rt1    rt5    p50    p90
   0     0     0.00  0.00  0.00  0.00
```

# Инструменты проверки бота

## Ngrok

Https-адрес, который вернёт Ngrok динамический, его каждый раз повторно необходимо указывать Telegram. Статический адрес – стоит денег (лучше оформить подписку в Azure)

У Ngrok есть админка, доступная по адресу <http://127.0.0.1:4040>

The screenshot displays the Ngrok web interface. At the top, there are tabs for 'ngrok' (with an 'online' indicator), 'Inspect', and 'Status'. Below the tabs, the 'All Requests' section shows a table of incoming requests. The first request is a GET to '/bot' with a 200 OK status and a response time of 14.01ms. The second request is also a GET to '/bot' with a 200 OK status and a response time of 25.02ms. The third request is a GET to '/favicon.ico' with a 404 Not Found status and a response time of 19.01ms. To the right of the request logs, there is a 'Connections' table showing 4 total connections and 0 open connections. Below that, the 'HTTP Requests' section shows a list of requests with their status codes: GET /bot (200 OK), GET /bot (200 OK), and GET /favicon.ico (404 Not Found).

All Requests			Connections	
GET /bot	200 OK	14.01ms	ttl	opn
GET /bot	200 OK	25.02ms	4	0
GET /favicon.ico	404 Not Found	19.01ms	HTTP Requests	
			GET /bot	200 OK
			GET /bot	200 OK
			GET /favicon.ico	404 Not Found

# Bot Framework

## Введение

Документация

<https://docs.microsoft.com/en-us/bot-framework/overview-introduction-bot-framework>

Регистрация ботов

<https://dev.botframework.com/bots/new>

## Каналы

Bing

Cortana

Email

Facebook

GroupMe

Kik

Skype

Skype for Business

Slack

SMS

Microsoft Teams

Telegram

## Недостатки

1. Нет поддержки .Net Core

<https://github.com/Microsoft/BotBuilder/issues/572>

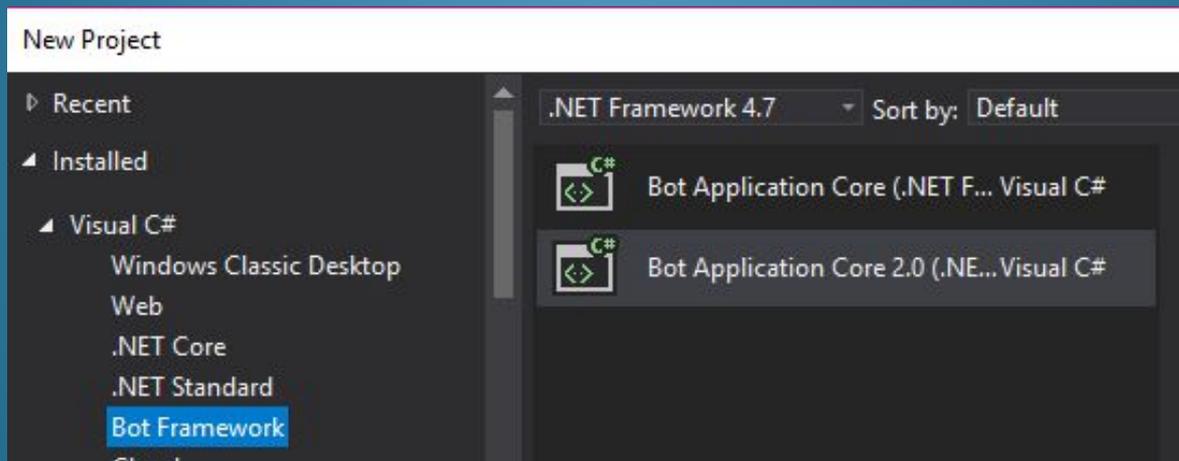
<https://designprincipia.com/microsoft-bot-framework-on-asp-net-core/>

2. Сервис размещается на серверах Microsoft

# Bot Framework SDK

Bot Builder SDK доступен для C# и для Node.js

Для .NET ставится через nuget-пакет Microsoft.Bot.Builder



# Bot Framework Emulator

<https://github.com/Microsoft/BotFramework-Emulator>

Bot Framework Channel Emulator

<http://localhost:51002/api/messages>

Microsoft App ID:

Microsoft App Password:     Locale:

**CONNECT**

Bot Framework Channel Emulator

<http://localhost:51002/api/messages>

Details

```
{
  "type": "message",
  "timestamp": "2017-10-25T20:11:49.105Z",
  "localTimestamp": "2017-10-25T23:11:49+03:00",
  "serviceUrl": "http://localhost:59037",
  "channelId": "emulator",
  "from": {
    "id": "63ja6j898717",
    "name": "Bot"
  },
  "conversation": {
    "id": "199ng9mhdg35"
  },
  "recipient": {
    "id": "default-user"
  },
  "membersAdded": [],
  "membersRemoved": [],
  "locale": "en-US",
  "text": "привет",
  "attachments": [],
  "entities": [],
  "replyToId": "5ga629jicina",
  "id": "k93i51ge492"
}
```

Log

Chat history:

- User: привет
- Bot: привет
- User: хороший бот
- Bot: хороший бот
- User: кофе будешь?
- Bot: кофе будешь? (at 11:13:12 PM)

Type your message...

# Bot Framework

## Код контроллера и Activity

```
[Route("api/[controller]")]
[BotAuthentication]
public class MessagesController : Controller
{
    [HttpPost]
    public async Task<HttpResponseMessage> Post([FromBody]Activity activity)
    {
        if (activity?.Type == ActivityTypes.Message)
        {
            await Conversation.SendAsync(activity, () => new Dialogs.RootDialog());
        }

        return new HttpResponseMessage(HttpStatusCode.OK);
    }
}
```

# Vot Framework

## Обработка сообщений (Dialog)

```
[Serializable]
public class RootDialog : IDialog<object>
{
    public Task StartAsync(IDialogContext context)
    {
        context.Wait(MessageReceivedAsync);
        return Task.CompletedTask;
    }

    private async Task MessageReceivedAsync(IDialogContext context,
IAwaitable<object> result)
    {
        var activity = await result as Activity;
        await context.PostAsync(activity.Text);
        context.Wait(MessageReceivedAsync);
    }
}
```

# Bot Framework Connector

Connector – API обеспечивающее связь между разными каналами

<https://docs.microsoft.com/en-us/bot-framework/dotnet/bot-builder-dotnet-connector>

```
[Route("api/[controller]")]
[BotAuthentication]
public class MessagesController : Controller
{
    [HttpPost]
    public async Task<HttpResponseMessage> Post([FromBody]Activity activity)
    {
        var connector = new ConnectorClient(new Uri(activity.ServiceUrl));
        var reply = activity.CreateReply(activity.Text);
        await connector.Conversations.ReplyToActivityAsync(reply);

        return new HttpResponseMessage(HttpStatusCode.OK);
    }
}
```

# Бот для Slack

<https://api.slack.com/apps>

<https://api.slack.com/apps/new>

## Create a Slack App ✕

 **Interested in the next generation of apps?**  
We're improving app development and distribution. Join the API Preview period for workspace tokens and the Permissions API.

**App Name**

Don't worry; you'll be able to change this later.

**Development Slack Workspace**

 ▼

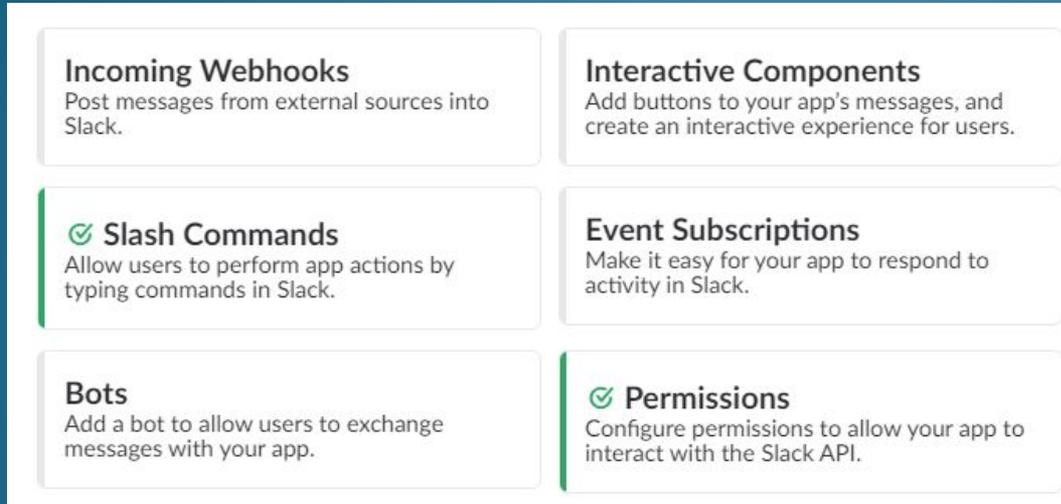
Your app belongs to this workspace—leaving this workspace will remove your ability to manage this app. Unfortunately, this can't be changed later.

By creating a Web API Application, you agree to the [Slack API Terms of Service](#).

# Бот для Slack

## Настройка бота

### Add features and functionality



### Install your app to your workspace

Установка приложения в ваше пространство.

### Manage distribution

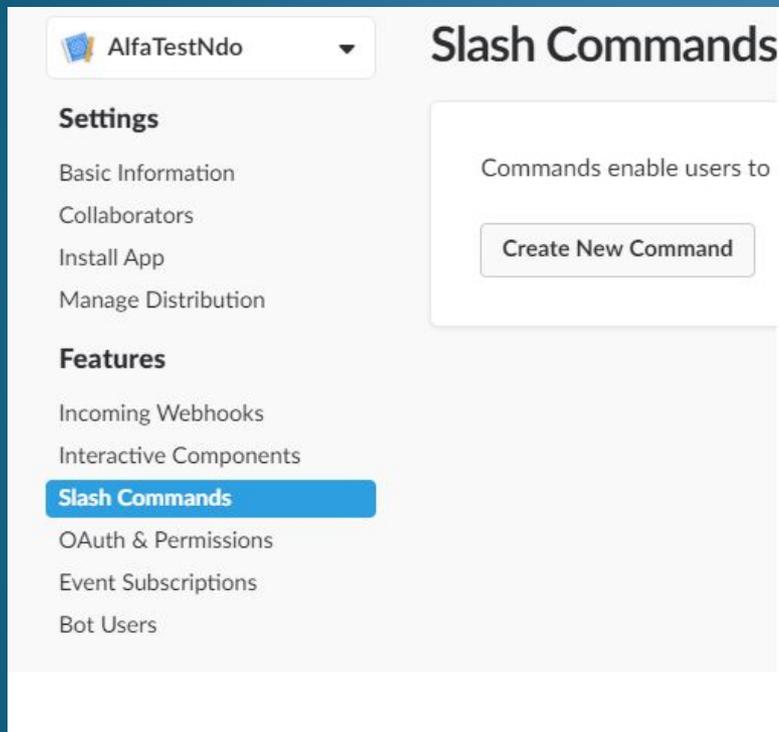
Позволяет открыть бота для других пространств.

### App Credentials

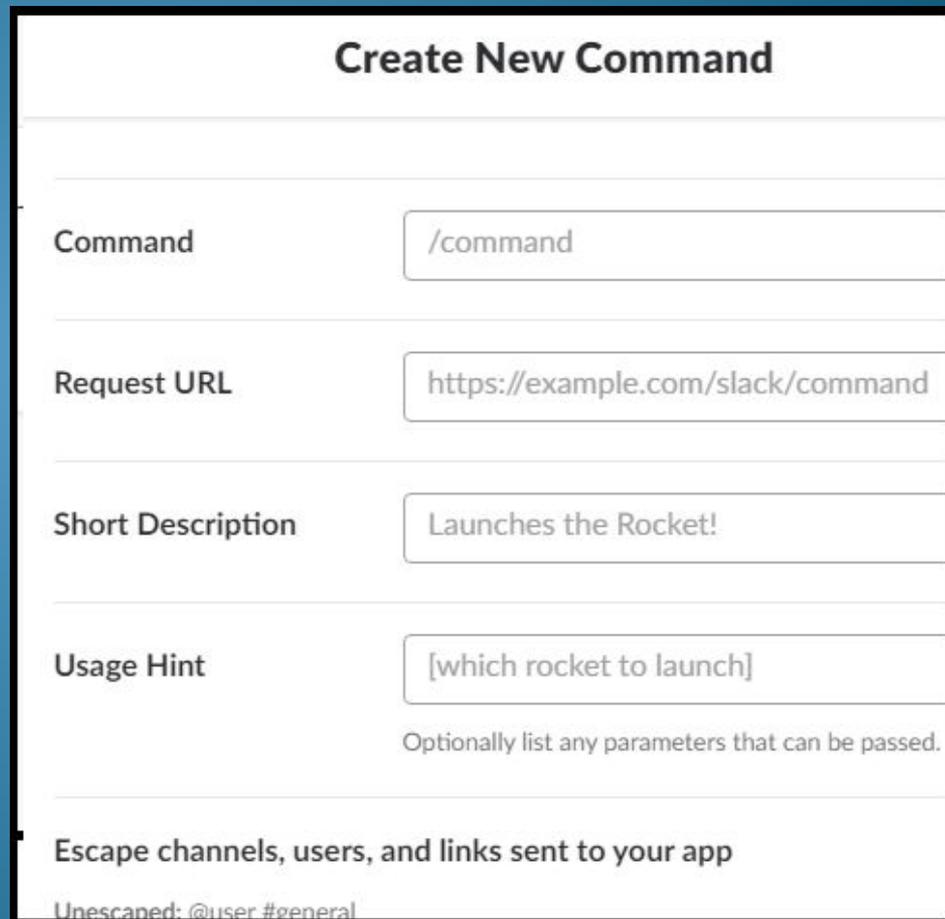
### Display Information

# Бот для Slack

## Настройка команд



The screenshot shows the Slack interface for the 'AlfaTestNdo' app. The left sidebar contains a 'Settings' section with options like 'Basic Information', 'Collaborators', 'Install App', and 'Manage Distribution'. Below that is a 'Features' section with 'Incoming Webhooks', 'Interactive Components', 'Slash Commands' (highlighted in blue), 'OAuth & Permissions', 'Event Subscriptions', and 'Bot Users'. The main content area is titled 'Slash Commands' and contains the text 'Commands enable users to' followed by a 'Create New Command' button.



The screenshot shows the 'Create New Command' form. It has the following fields and values:

- Command:** /command
- Request URL:** https://example.com/slack/command
- Short Description:** Launches the Rocket!
- Usage Hint:** [which rocket to launch]

Below the Usage Hint field, there is a note: 'Optionally list any parameters that can be passed.'

At the bottom, there is a section titled 'Escape channels, users, and links sent to your app' with the text 'Unescaped: @user #general'.

# Бот для Slack

## Типы ключей

**User tokens** - ключи пользователей, авторизованных через OAuth

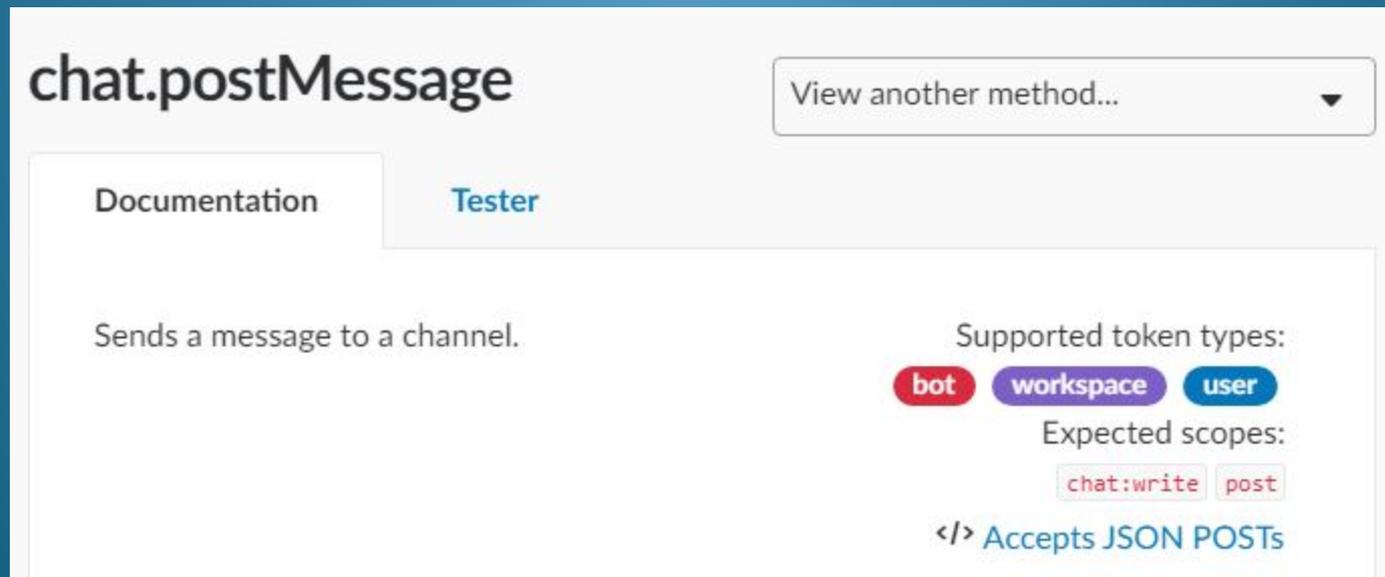
**Bot user tokens** – ключ, специального созданного пользователя для бота

**Workspace tokens** – ключ рабочей области

**Legacy tokens** – ни к чему не привязанный ключ (устаревший)

**Verification token** – ключ приложения

<https://api.slack.com/methods/chat.postMessage>



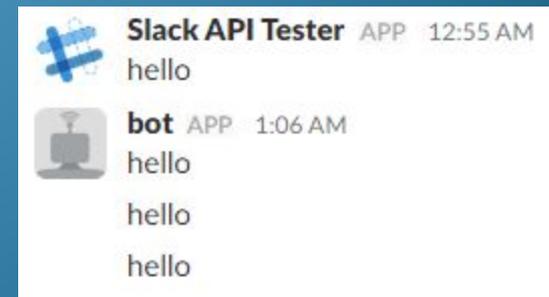
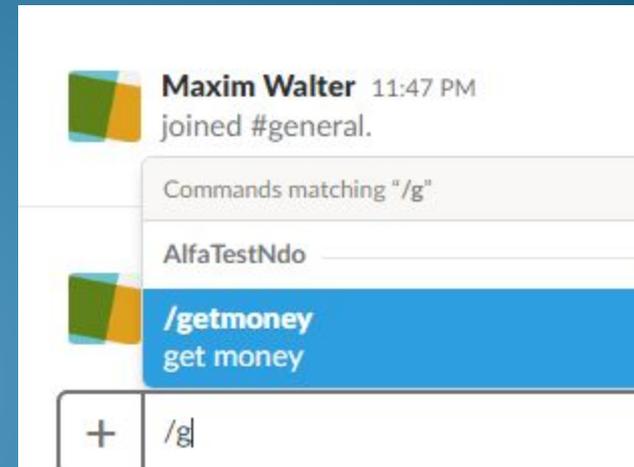
The screenshot shows the Slack API Tester interface for the `chat.postMessage` method. The title is `chat.postMessage` and there is a dropdown menu to "View another method...". Below the title, there are two tabs: "Documentation" and "Tester", with "Tester" being the active tab. The description of the method is "Sends a message to a channel." To the right of the description, there are two sections: "Supported token types:" with three buttons labeled "bot", "workspace", and "user"; and "Expected scopes:" with two buttons labeled "chat:write" and "post". At the bottom right, there is a note: "`</>` Accepts JSON POSTs".

Slack API Tester – специальный бот для тестирования API

# Бот для Slack

## Запрос из Slack

```
public class Message
{
    public string channel_id { get; set; }
    public string channel_name { get; set; }
    public string command { get; set; }
    public string response_url { get; set; }
    public string team_domain { get; set; }
    public string team_id { get; set; }
    public string text { get; set; }
    public string token { get; set; }
    public string trigger_id { get; set; }
    public string user_id { get; set; }
    public string user_name { get; set; }
}
```



# Бот для Slack Контроллер

<https://api.slack.com/community>

```
[Route("bot")]
public class BotController : Controller
{
    [HttpPost]
    public async void Post(Message message)
    {
        var uri = new Uri("https://slack.com/api/chat.postMessage?token="
            + token + "&channel=" + message.channel_id + "&text=hello");

        var httpClient = new HttpClient();
        await httpClient.GetAsync(uri).ConfigureAwait(false);
    }
}
```

# Бот для Facebook

Официальная инструкция

<https://developers.facebook.com/docs/messenger-platform/getting-started/quick-start>

Страница создания приложений

<https://developers.facebook.com/apps>

## Создайте новый ID приложения

Начните интегрировать Facebook в свое приложение или сайт

Отображаемое название

Эл. адрес для связи

Продолжая, вы соглашаетесь с Политикой платформы Facebook

Отмена

Создайте ID приложения

# Бот для Facebook

## Настройка Webhook

Новая подписка на Страницу

URL обратного вызова

Подтвердить маркер

Поля подписки

<input checked="" type="checkbox"/> messages	<input type="checkbox"/> messaging_postbacks	<input type="checkbox"/> messaging_optins
<input type="checkbox"/> message_deliveries	<input type="checkbox"/> message_reads	<input type="checkbox"/> messaging_payments
<input type="checkbox"/> messaging_pre_checkouts	<input type="checkbox"/> messaging_checkout_updates	<input type="checkbox"/> messaging_account_linking
<input type="checkbox"/> messaging_referrals	<input type="checkbox"/> message_echoes	<input type="checkbox"/> standby
<input type="checkbox"/> messaging_handovers	<input type="checkbox"/> messaging_policy_enforcement	

Подробнее

Страница

Маркер доступа Страницы

[Создать новую Страницу](#)

Select a page to subscribe your webhook to the page events  
Страницы, на которые подписано приложение: **Alfatestndopage**

# Бот для Facebook

## Настройка Webhook

```
[Route("bot")]
public class BotController : Controller
{
    [HttpGet]
    public string Verify()
    {
        var mode = Request.Query["hub.mode"].FirstOrDefault();
        var challenge = Request.Query["hub.challenge"].FirstOrDefault();
        var token = Request.Query["hub.verify_token"].FirstOrDefault();
        return challenge ?? string.Empty;
    }

    [HttpPost]
    public void Post([FromBody]string value)
    {
    }
}
```

# Бот для Facebook

## Входящее сообщение

```
{
  "object": "page",
  "entry": [
    {
      "id": "507370816302631",
      "time": 1509659789298,
      "messaging": [
        {
          "sender": {
            "id": "1187895957978601"
          },
          "recipient": {
            "id": "507370816302631"
          },
          "timestamp": 1509659769025,
          "message": {
            "mid": "mid.$cAAGXgDziYCdlsAFawVffrwNYg_C1",
            "seq": 269,
            "text": "hello"
          }
        }
      ]
    }
  ]
}
```

# Бот для Facebook

## Отправка ответа

```
[HttpPost]
public void Post([FromBody] Letter letter)
{
    var content = letter.entry[0].messaging[0];
    const string token = "yourtoken";
    var uri = new Uri("https://graph.facebook.com/v2.6/me/messages?access_token="
+ token);

    var request = (HttpWebRequest)WebRequest.Create(uri);
    request.ContentType = "application/json";
    request.Method = "POST";
    using (var requestWriter = new StreamWriter(request.GetRequestStream()))
    {
        requestWriter.Write($"@" {{recipient: {{ id: {content.sender.id}}},message: {{text:
""{content.message.text}"" }}}});
    }
    var response = (HttpWebResponse)request.GetResponse();
}
```

# ССЫЛКИ

Примеры использования API telegram.bot

<https://github.com/TelegramBots/telegram.bot.examples>

Пишем бота Telegram на C#

<http://aftamat4ik.ru/pishem-bota-telegram-na-c/>

Как легко написать бота для Telegram на C#

<https://habrahabr.ru/sandbox/103396/>

Microsoft Bot Framework (статья введение)

[http://ru.bmstu.wiki/Microsoft\\_Bot\\_Framework](http://ru.bmstu.wiki/Microsoft_Bot_Framework)

Цикл видео по Microsoft Bot Framework

<https://www.youtube.com/playlist?list=PLgE-CyaX1p3FE55OTRNH-kOb16zqeBZCo>

Microsoft Bot Framework на Linux под Node.JS

<https://habrahabr.ru/post/333824/>

Azure Bot Service (описание и цены на хостинг ботов)

<https://azure.microsoft.com/ru-ru/services/bot-service/>

# ССЫЛКИ

Разработка чат-бота для Facebook Messenger

<https://habrahabr.ru/post/281559/>

Facebook Chatbot in ASP.NET

<https://tutorials.botsfloor.com/facebook-chatbot-in-asp-net-2f9379a238b0>

Создание бота в контакте (для сообществ)

<https://vk.com/dev/bots>

[https://vk.com/dev/bots\\_docs](https://vk.com/dev/bots_docs)

Сводная страница информации по ботам в Viber

<https://habrahabr.ru/post/338970/>