УДК 004.624

Бот цитат для Slack

Кизянов Антон Олегович
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В данной статье описан процесс создания бота на NodeJS для Slack. Для создания используется NodeJS и API сервис Mashepe. Созданный бот позволяет получать цитаты через канал Slack через API сервис Mashepe.
Ключевые слова: Slack, Mashepe, бот

Quote Bot for Slack

Kizyanov Anton Olegovich
Sholom-Aleichem Priamursky State University
student

Abstract
This article describes the process of creating a NodeJS bot for Slack. For creation, NodeJS and the Mashepe API service are used. The created bot allows you to receive quotes through the Slack channel through the Mashepe API service.
Keywords: Slack, Mashepe, bot

Slack (https://slack.com/) – это известное приложение для обмена сообщениями в реальном времени и пакет для совместной работы для команд. Slack был создан с нуля, чтобы его было легко использовать. Он предлагает широкий набор API-интерфейсов, который позволяет разработчикам расширять его возможности, чтобы сделать его еще более полезным.

Одна из функций в Slack, — это Slackbot. Это дружелюбный бот, доступный в каждой команде Slack, чтобы помогать пользователям создавать свои профили и объяснять им, как работает Slack.

Slackbot дает возможность создавать своих собственных ботов, которые могут действовать как автоматические пользователи, которые могут реагировать на определенные события и помогать команде делать полезные вещи.

Цель исследования – создать бота для цитат на базе Slack.

Ранее этим вопросом интересовались В.В. Сорокин развивал тему «Использование платформы "slack" в образовании» [1] в которой описаны особенности и возможности платформы Slack, а также даны примеры использования данной платформы в образовательном процессе. В рамках Slack люди могут быстро и легко обмениваться информацией и общаться с

коллегами, прикреплять файлы для совместного использования. Частные каналы разработаны для мозгового штурма идей, координации между членами команды, имеется возможность обмениваться файлами, задавать вопросы, договариваться о деталях коллективного проекта/работы. А. Коробкин с темой «Slack – автоматизируем настройку сервисов» [2], а подробнее про работу по настройке сервера превращается в рутину, если этих серверов сотня. С помощью slack этот процесс можно автоматизировать быстро и без лишних затрат. Ю.Е. Шац, В.А. Кожевников опубликовали статью «Разработка библиотеки, предназначенной для создания ботов slack, telegram и facebook мессенджеров» [3] описали обзор и сравнительный анализ технических средств для разработки чат-бота, а именно рассмотрены общие понятия чат-бота и его преимущества, а также где применяются чат-боты, была представлена статистика и анализ языков программирования и выбран язык для разработки чат-бота, а также произведен выбор других средств для реализации проекта.

Бот будет давать цитаты в ответ на общий канал. Идея состоит в том, чтобы создать бота, который вдохновляет команду в повседневной деятельности, и цитаты, безусловно, отличный способ вдохновиться.

Первое, что нужно сделать, это настроить бота в Slack и зарегистрировать его, чтобы использовать Slack API.

Для этого нужно использовать Slack Real Time Messaging API (https://api.slack.com/rtm), который представляет собой API на основе WebSocket, который позволяет получать события в реальном времени и отправлять сообщения каналам, частным группам и пользователям.

API действительно хорошо построен, а документация проста в использовании. WebSockets не будет использоваться напрямую, а вместо этого будет использоваться модуль Node.js (https://www.npmjs.com/package/slackbots), который значительно упрощает разработку с использованием JavaScript.

Дальше нужно настроить расширения каналов и создать нового бота. Таким образом получить токен API, который необходим для аутентификации в Slack и начала работы.

Чтобы добавить бота в организацию Slack, необходимо зарегистрировать его по следующему URL-адресу: https://yourorganization.slack.com/services/new/bot.

Нужно будет изменить свою организацию на название компании или команды, которое использовали при регистрации своей учетной записи Slack. При открытии URL-адреса будет экран как на рисунке 1.

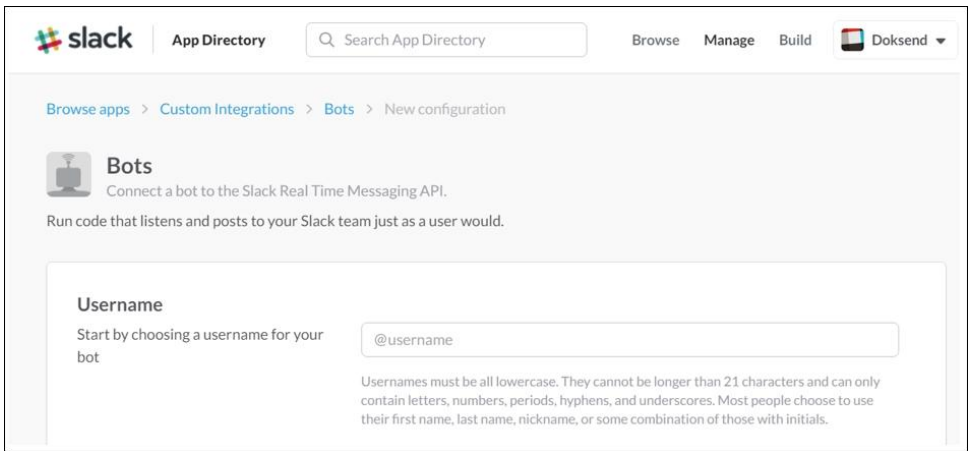


Рис. 1 Главная страница SlackBot

Именем бота будет «quotebot», также Slack требует, чтобы все имена ботов были написаны строчными буквами.

После того, как имя будет введено в поле «Имя пользователя», нужно нажать кнопку «Добавить интеграцию с ботом», как на рисунке 2.

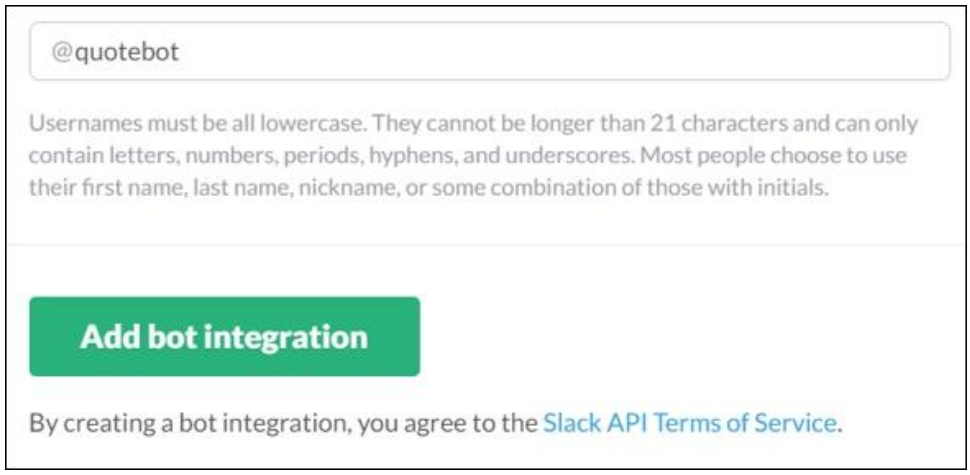


Рис. 2 Форма создания бота

Как только это будет сделано, будет представлен экран, на котором можно дополнительно настроить бота и добавить функции, такие как изображение или эмодзи. Он выглядит как на рисунке 3.

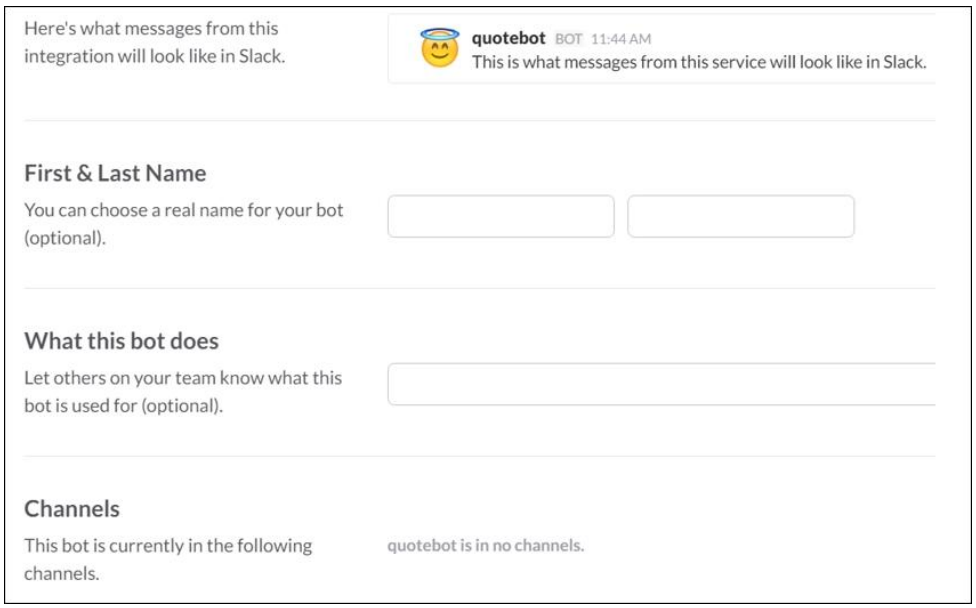


Рис. 3 Форма с дополнительной информацией

Доступны и другие параметры, которые можно настроить, но они не помещаются на одном снимке экрана.

В этой форме также есть API токен, который потребуется при работе, он выглядит как на рисунке 4.

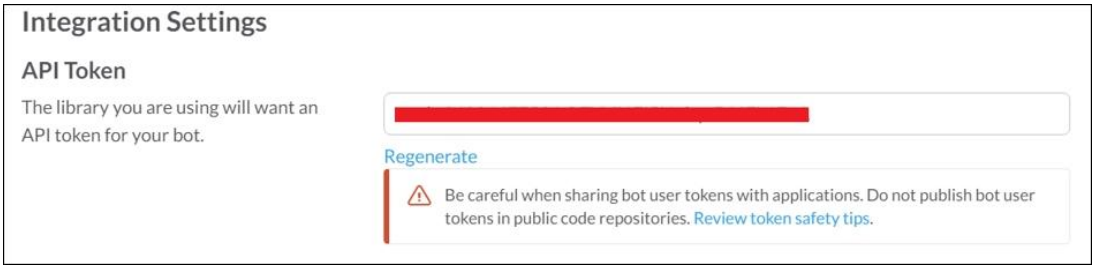


Рис. 4 API токен

Теперь нужно создать файл package.json с помощью команды ниже.

```
npm init
```

После консоль должна выглядеть как на рисунке 5.

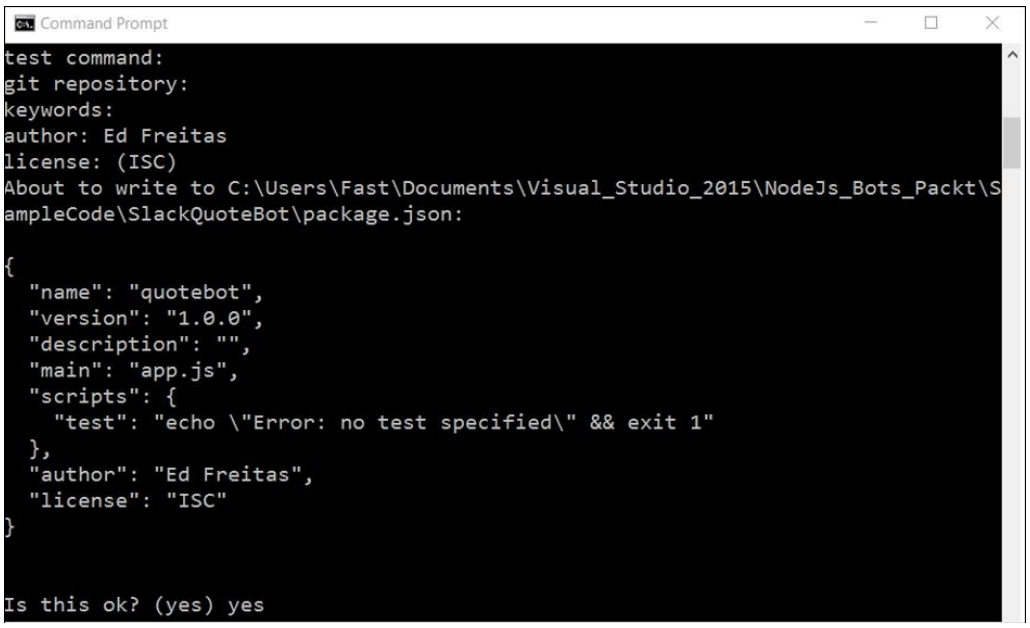


Рис. 5 Инициализация файла package.json

Бот должен иметь возможность получать цитату и отвечать в общий канал.

Есть замечательный сайт под названием They Said So (https://theysaidso.com/), который представляет собой сервис, предоставляющий услуги (QAAS). Цитаты от нескольких авторов можно получить с помощью простого в использовании REST API. Главная страница выглядит как на рисунке 6.

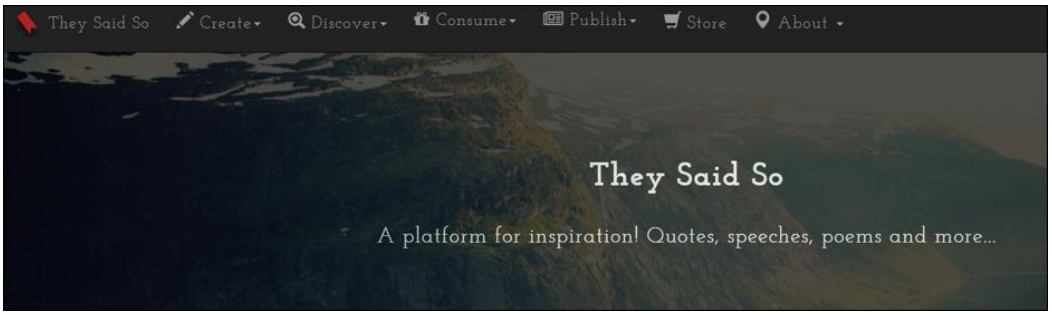


Рис. 6 Главная страница They said so

Поскольку понадобится доступ к этой службе с помощью REST, нужно установить клиентскую библиотеку REST для Node.js в приложение.

Чтобы установить эту библиотеку, нужно выполнить команду ниже.

```
npm install request --save
```

Теперь, когда это сделано, следующим шагом будет установка библиотеки с именем (https://www.npmjs.com/package/slackbots), которая будет действовать как уровень абстракции для работы с API обмена сообщениями в реальном времени Slack:slackbots

```
npm install slackbots --save
```

После этого package.json файл будет выглядеть как на рисунке 7.

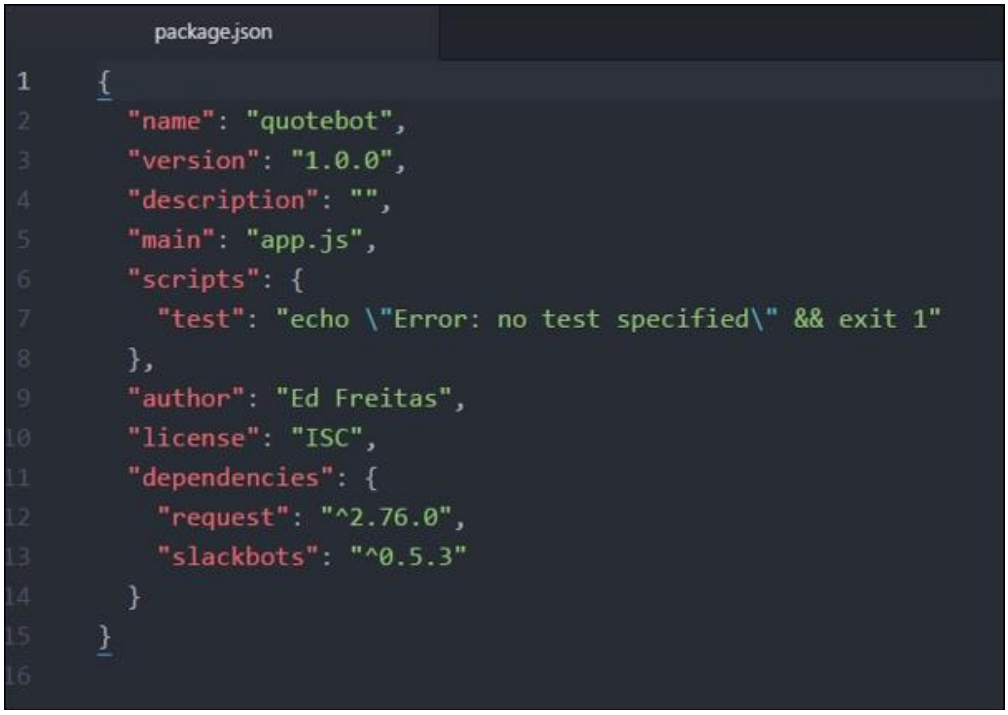


Рис. 7 Файл Package.json

Теперь, все настройки бота Node.js настроены, можно начать писать код.

Как упоминалось ранее, для взаимодействия с API обмена сообщениями в реальном времени Slack будем использовать библиотеку Node.js (пакет npm) под названием slackbots (https://www.npmjs.com/package/slackbots).

```
var Bot = require('slackbots');
var settings = {token: 'API TOKEN', name: 'quotebot’};
var bot = new Bot(settings);
bot.on('start', function() {
    bot.postMessageToChannel('channel-name', 'Hi channel.');
    bot.postMessageToUser('a-username', 'Hi user.');
    bot.postMessageToGroup('a-private-group', 'Hi private group.');
});
```

Перед тем, как запустить этот код, нужно заменить строки channel-name, a-username и a-private-groupc собственными значениями, взятыми из Slack. Также нужно будет заменить API TOKEN строку токеном, который был предоставлен при создании бота. Теперь код должен выглядеть примерно так:

```
var Bot = require('slackbots');
var settings = {token: 'xoxb-.........-R7VVJ1FI5Hzfcyt.........', name:
'quotebot'};
var bot = new Bot(settings);
bot.on('start', function() {
    bot.postMessageToChannel('general', 'Hi channel.');
    bot.postMessageToUser('radkiddo', 'Hi user.');
    bot.postMessageToGroup('tisdoksend', 'Hi private group.');
```

```
});
```

После замены этих значений можете запустить приложение из командной строки следующим образом.

```
Node app.js
```

Если войти в Slack и открыть страницу команды, можно увидеть бота, как на рисунке 8.

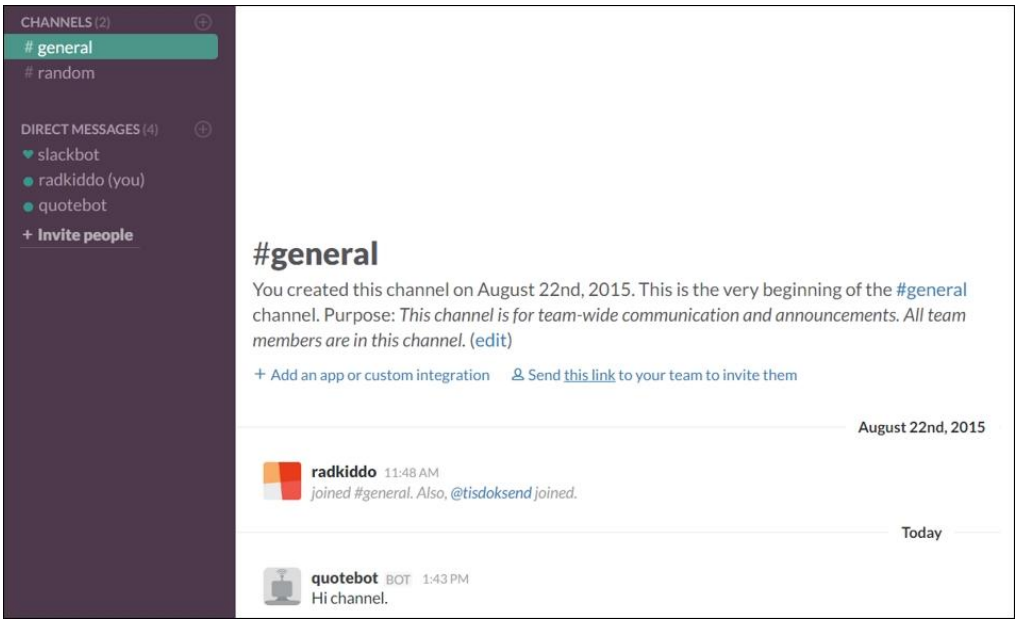


Рис. 8 Появление бота в Slack

Как видно из предыдущего кода, первое, что нужно сделать, это вызвать конструктор Slackbot. Оттуда можно создать новый Bot объект и добавить обратные вызовы к определенным событиям.

В этом коде используется start событие, которое запускается, когда бот успешно подключается к серверу Slack:

```
bot.on('start', function() {
    bot.postMessageToChannel('general', 'Hi channel.');
    bot.postMessageToUser('radkiddo', 'Hi user.');
    bot.postMessageToGroup('tisdoksend', 'Hi private group.');
});
```

Затем можно использовать методы, предлагаемые библиотекой, для публикации сообщения в канале с помощью postMessageToChannel метода, для пользователя в виде личного сообщения с использованием postMessageToUser или в частном групповом разговоре путем вызова postMessageToGroup.

Сервис They Said So имеет огромную коллекцию цитат в своей базе данных, а Quotes API - отличный и удобный способ доступа к этим данным. Чтобы использовать Quotes API, сначала необходимо зарегистрироваться в службе поэтому URL-адресу: https://theysaidso.com/register.

После регистрации, должно прийти электронное письмо с автоматическим подтверждением, которое будет выглядеть как на рисунке 9.

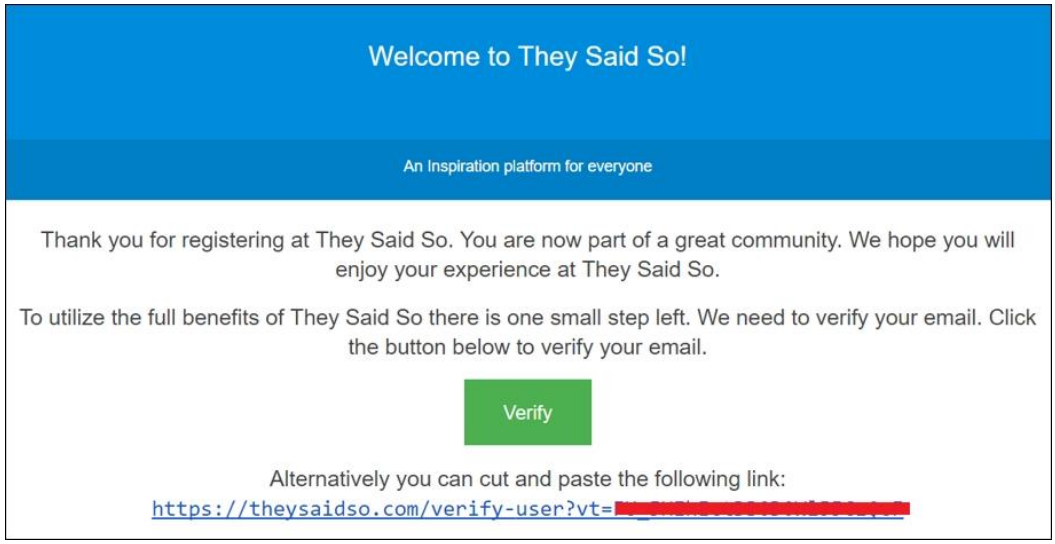


Рис. 9 Письмо с подтверждением

Здесь достаточно нажать на кнопку «Verify» кнопку, чтобы подтвердить свой недавно зарегистрированный счет и начать пользоваться услугой.

Как только это будет сделано, вскоре придёт следующее электронное письмо как на рисунке 10.

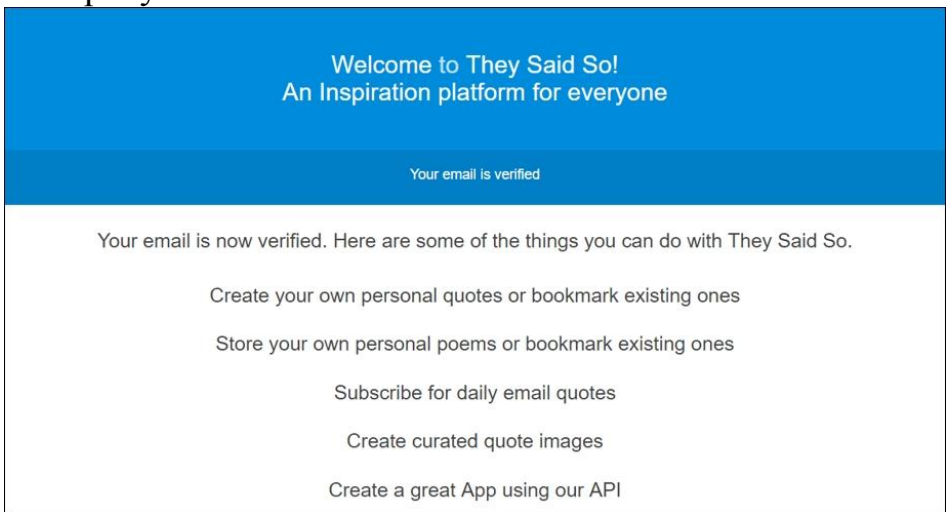


Рис. 10 Второе электронное письмо

Следующим шагом является подписка на Quotes API, чтобы начать его использовать. Это можно сделать, посетив следующий URL-адрес: https://theysaidso.com/api/#subscribe или щелкнув ссылку «Создать отличное приложение с помощью API» в полученном электронном письме.

Поскольку это демонстрационное приложение, лучше будет использовать API через Mashape (https://market.mashape.com/orthosie)

Mashape — это сервис, который помогает разработчикам предоставлять API и микросервисы. Многие сторонние API-интерфейсы предоставляются через Mashape или аналогичные службы.

При использовании Mashape, нужно нажать кнопку «Использовать API» в нижней части экрана, как на рисунке 11.

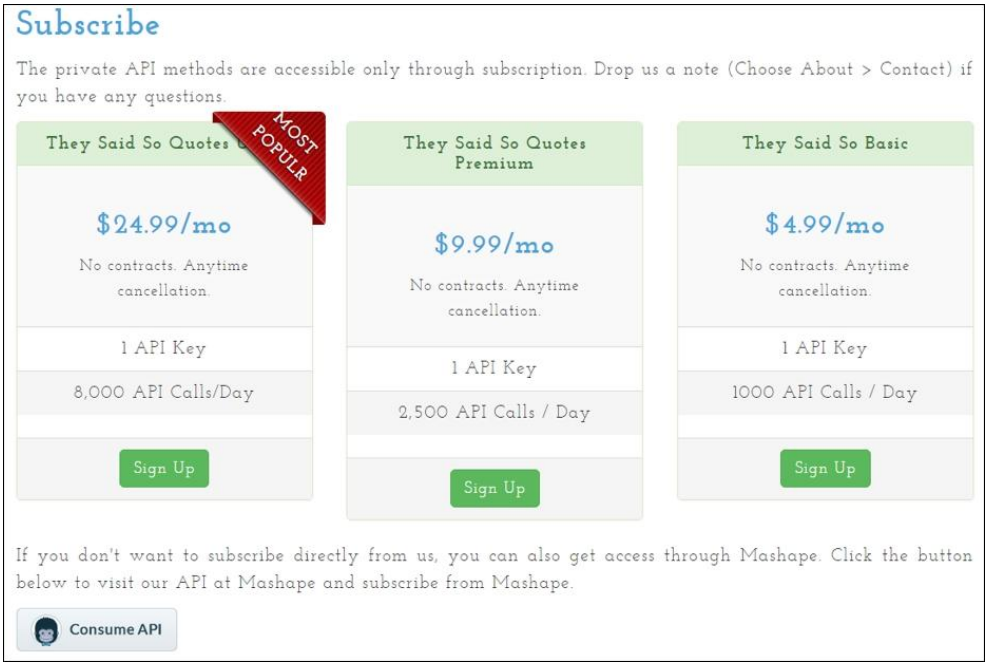


Рис. 11 Тарифные планы Mashape

При нажатии на кнопку, появится страница как на рисунке 12.

При нажатии на кнопку, появится следующий экран. Оказавшись там, нужно нажать на вкладку цены, как на рисунке 13.

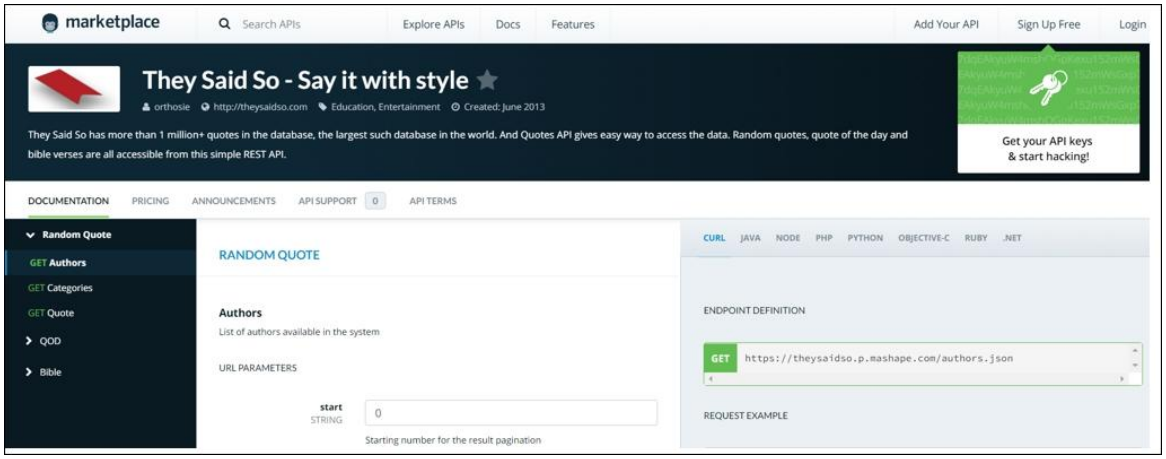


Рис. 12 Страница интеграции API

На вкладке цены под базовым планом нужно нажать кнопку подписаться.

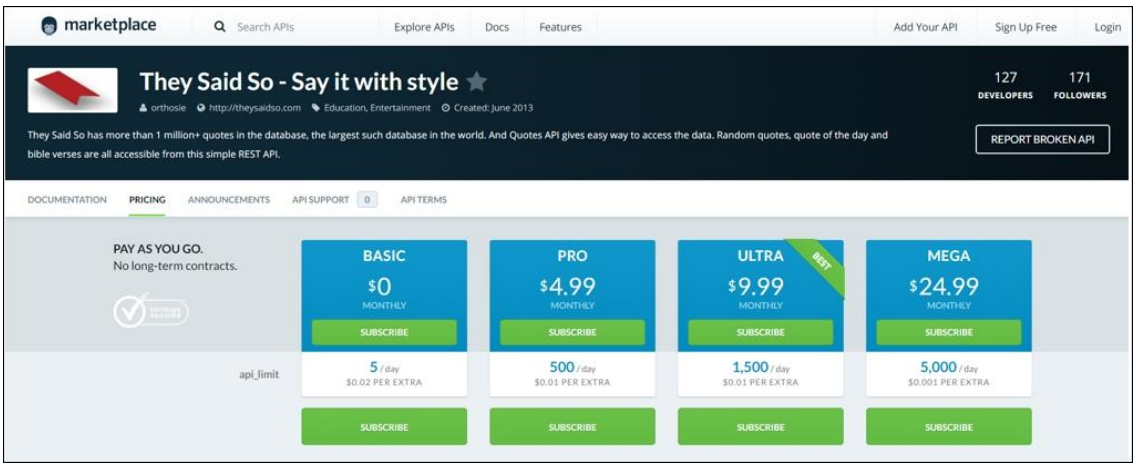


Рис. 13 Список планов подписки

При нажатии на кнопку подписаться, появится следующий экран как на рисунке 14.

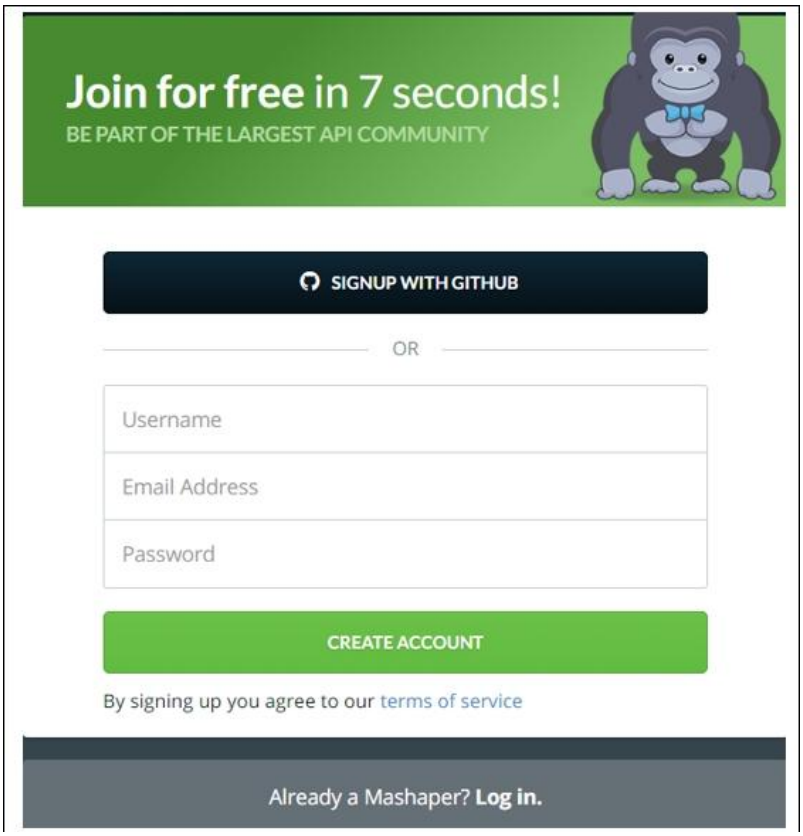


Рис. 14 Форма регистрации

Если учетная запись GitHub или Mashape, можно просто подписаться на API, войдя в службу с любой из этих учетных записей. В противном случае придется создать учетную запись на Mashape.

В любом случае процесс очень простой и понятный. Как только это будет сделано, можно приступать к изучению API и его использованию, как на рисунке 15.

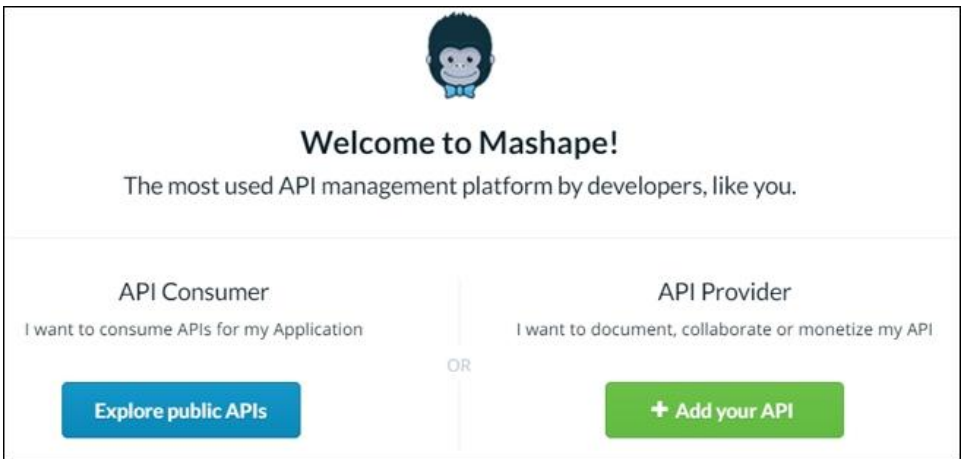


Рис. 15 Форма подключения к Mashape

Нужно нажать кнопку «Изучить общедоступные API-интерфейсы», а затем нажать кнопку «Назад» в своем браузере или перейдите по URL-адресу: https://market.mashape.com/orthosie/they-said-so-say-it-with-style/, чтобы начать знакомство с API.

Из API интересует раздел цитат. можем посмотреть на это, щелкнув ссылку GET Quote в левой части экрана, как на рисунке 16.

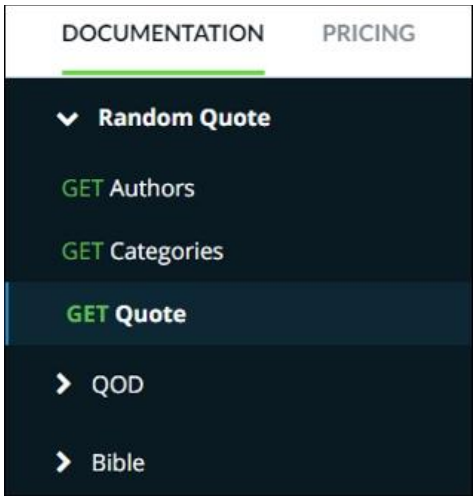


Рис. 16 Меню выбора API

Это перенесет на следующую страницу, где будет информация про вызов API, как на рисунке 17.

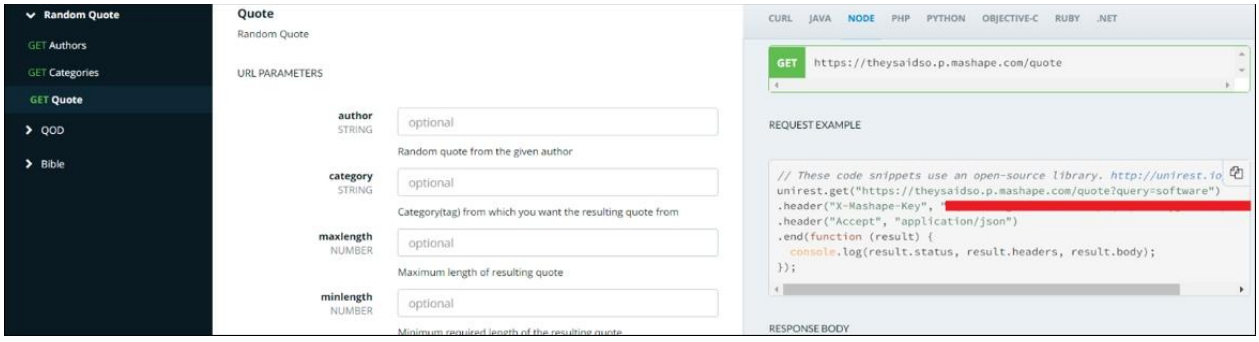


Рис. 17 Страница с примерами использования Mashape

Есть также несколько примеров на разных языках программирования, включая Node.js, который использует библиотеку Unirest (http://unirest.io/) для выполнения HTTP-запросов. В приложении вместо этого будет использоваться библиотека запросов (https://www.npmjs.com/package/request).

Во всех примерах кода, включая Node.js, упомянутый на странице документации GET Quote, ключ API Token передается в заголовке HTTP-запроса в качестве значения X-Mashape-Key параметра.

Нужно создать новый файл с именем, TestRequest.js со следующем содержимым.

```
var rq = require('request');
var token = 'Your They Said So API Key';
GetQuote = function() {
  var options = {
   url: 'https://theysaidso.p.mashape.com/quote?query=software',
   headers: {'User-Agent': 'request', 'X-Mashape-Key': token}};
  rq(options, function (error, response, body) {
    if (!error && response.statusCode == 200) {console.log(body);}})};
GetQuote();
```

Прежде чем запускать, нужно убедиться, что тарифный план стоит BASIC, который включает пять звонков в день. все равно придется ввести номер кредитной карты, по которой будет выставлен счет, если выполняете более пяти запросов в день.

Всегда можно отказаться от подписки поэтому URL-адресу: https://market.mashape.com/orthosie/they-said-so-say-it-with-style/pricing, щелкнув ссылку «Отказаться от подписки» под планом BASIC.

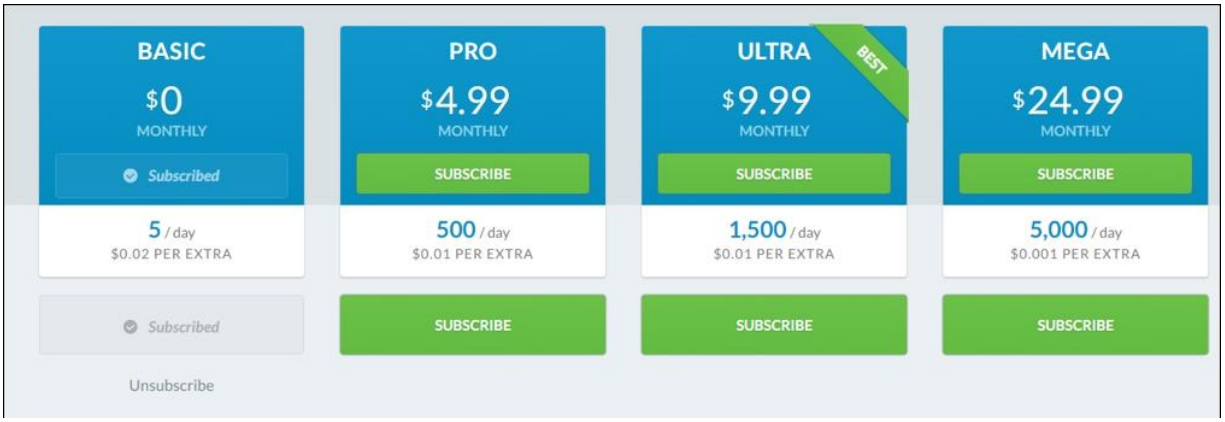


Рис. 18 Список тарифных планов

Теперь можно запустить этот сценарий из командной строки следующим образом:

```
Node TestRequest.js
```

Это дает следующий результат, как на рисунке 19.

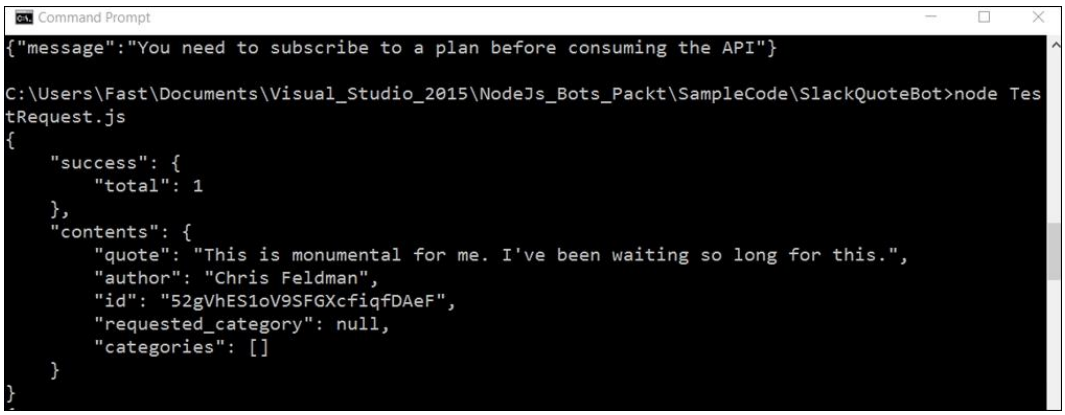


Рис. 19 Вывод результата запуска TestRequest

В предыдущем фрагменте кода использовалось стартовое событие. В дальнейшем также понадобится событие message, которое будет использоваться для перехвата входящего сообщения и на основе этого ответа.

Нужна функция, которая будет перехватывать каждое сообщение API в реальном времени, которое может прочитать бот. Сюда входят практически все сообщения чата в любом канале, где был установлен бот, а также личные сообщения, адресованные боту, или другие уведомления в режиме реального времени, такие как ввод пользователя в канал, редактирование или удаление сообщений, присоединение или уход пользователей. канал и так далее.

Сообщения API в реальном времени — это не просто сообщения чата, а любые события, происходящие в организации Slack.

В идеальном варианте, нужно чтобы бот фильтровал все эти события, чтобы обнаруживать общедоступные сообщения в каналах, в которых упоминается getquote или имя бота, а затем отреагировать на это сообщение, ответив случайной цитатой, полученной из имеющегося у API.

Код будет выглядеть следующим образом:

```
onMessage = function (msg) {
    if (isChatMsg(msg) &&
        !isFromQuoteBot(msg) &&
        isMentioningQuote(msg)) {
        replyWithRandomQuote(bot, msg);}};
```

onMessage Функция принимает msg объект в качестве параметра. Msg содержит всю информацию, которая описывает в реальное время события, полученное через Slack Real Time API.

Теперь нужен разбор на каждую вспомогательную функцию одну за другой.

```
isChatMsg = function (msg) {return msg.type === 'message';};
```

Эта функция проверяет, соответствует ли событие в реальном времени msg отправляемому пользователем. Создав первую вспомогательную функцию, давайте посмотрим на вторую:

```
isFromQuoteBot = function (msg) {return msg.username === 'quotebot';};
```

Эта вспомогательная функция позволяет увидеть, msg исходит ли сообщение от пользователя, который не является самим котировщиком.

И последнее, но не менее важное: последняя вспомогательная функция проверяет, содержат ли сообщения строку getquote. Без этой проверки потенциально могли бы получить бесконечный цикл цитат.

```
isMentioningQuote = function (msg) {
    return msg.text.toLowerCase().indexOf('getquote') > -1;};
```

После выполнения всех вспомогательных функций проверки метод ответа на случайную цитату будет выглядеть так.

```
replyWithRandomQuote = function (bot, oMsg) {
  var options = {
   url: 'https://theysaidso.p.mashape.com/quote?query=software',
   headers: {'User-Agent': 'request', 'X-Mashape-Key': token}};
  rq(options, function (error, response, body) {
    if (!error && response.statusCode == 200) {
      bot.postMessageToChannel(bot.channels[0].name, body);}})};
```

Передаем onMessage обратный вызов прослушивающему событию следующим образом.

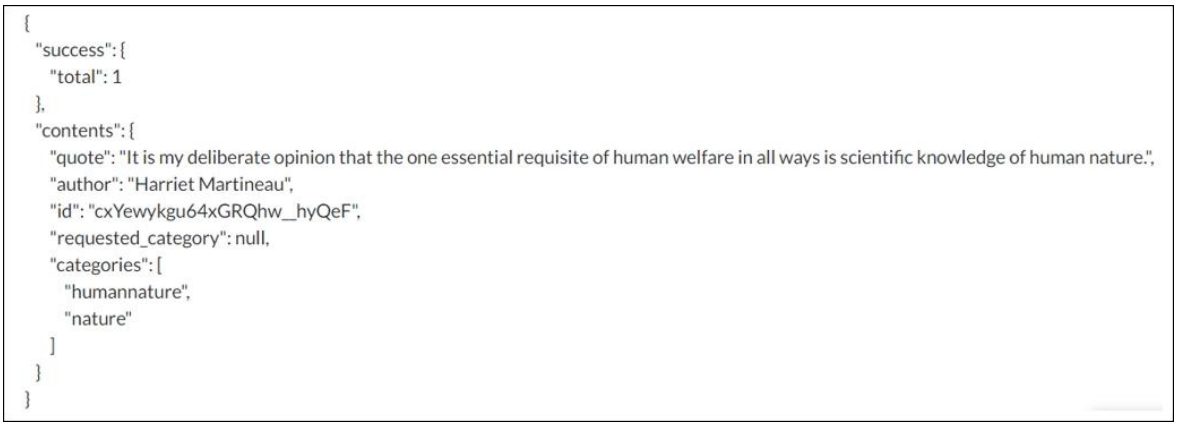
```
bot.on('message', onMessage);
```

Полный код выглядит так.

```
var Bot = require('slackbots');
var rq = require('request');
var token = ' YOUR MASHAPE API TOKEN ';
var settings = {token: 'YOUR SLACK API TOKEN', name: 'quotebot'};
var bot = new Bot(settings);
isChatMsg = function (msg) {return msg.type === 'message'; };
isFromQuoteBot = function (msg) {return msg.username === 'quotebot';};
isMentioningQuote = function (msg) {
    return msg.text.toLowerCase().indexOf('getquote') > -1;};
replyWithRandomQuote = function (bot, oMsg) {
  var options = {
    url: 'https://theysaidso.p.mashape.com/quote?query=software',
    headers: {'User-Agent': 'request', 'X-Mashape-Key': token }};
  rq(options, function (error, response, body) {
    if (!error && response.statusCode == 200) {
      bot.postMessageToChannel(bot.channels[0].name, body);}})};
bot.on('message', function (msg) {
    if (isChatMsg(msg) &&
        !isFromQuoteBot(msg) &&
        isMentioningQuote(msg)) {
        replyWithRandomQuote(bot, msg);}});
```

Чтобы увидеть это в действии, нужно написать цитату с текстом getquote в Slack.

Как только это произойдет, появится сообщение на канале #general, как на рисунке 20.



```
{
  "success": {
    "total": 1
  },
  "contents": {
    "quote": "It is my deliberate opinion that the one essential requisite of human welfare in all ways is scientific knowledge of human nature.",
    "author": "Harriet Martineau",
    "id": "cxYewykgu64xGRQhw__hyQeF",
    "requested_category": null,
    "categories": [
      "humannature",
      "nature"
    ]
  }
}
```


Рис. 20 Ответ на запрос цитаты

Это можно было бы дополнительно оптимизировать, чтобы в конечном итоге проанализировать ответ тела и просто вывести цитату и автора без каких-либо других деталей.

В дополнение к этому также можете добавить дополнительный код для обработки естественного языка, интерпретации большего количества команд, а также для ответа на разные каналы.

Возможности откровенно безграничны, и все, что нужно, — это время, воображение и преданность делу.

Вывод

Результатом статьи стал бот в Slack управляемый с помощью node.js и позволяющий забирать через API сервис Mashepe цитаты по запросу с канала в Slack. На данном примере можно разобраться в работе API сервисов и взаимодействию с ними через NodeJS.

Библиографический список

1. Сорокин В.В. Использование платформы "slack" в образовании // В сборнике: Инновации в здоровье нации. Сборник материалов V Всероссийской научно-практической конференции с международным участием. 2017. С. 536-539. URL: https://www.elibrary.ru/item.asp?id=34986223 (Дата обращения: 29.01.2021)
2. Коробкин А. Slack – автоматизируем настройку сервисов // Системный администратор. 2009. № 2 (75). С. 32-34. URL: https://www.elibrary.ru/item.asp?id=20406877 (Дата обращения: 29.01.2021)
3. Шац Ю.Е., Кожевников В.А. Разработка библиотеки, предназначенной для создания ботов slack, telegram и facebook мессенджеров // В сборнике: Информатика и кибернетика (ComCon-2017). Сборник докладов студенческой научной конференции Института компьютерных наук и технологий. 2017. С. 216-219. URL:

https://www.elibrary.ru/item.asp?id=35348218 (Дата обращения: 29.01.2021)