УДК 004.624

Создание смс бота на базе Twilio

Кизянов Антон Олегович
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В данной статье описан процесс создания бота на базе платформы Twilio. Для создания используется язык программирования JavaScript и платформа Twilio, с доступом через API. Созданный бот может сам отправлять смс на нужные номера, но и отвечать на присланные ему смс.
Ключевые слова: NodeJS, Twilio, бот

Creation of SMS bot based on Twilio

Kizyanov Anton Olegovich
Sholom-Aleichem Priamursky State University
student

Abstract
This article describes the process of creating a bot based on the Twilio platform. For creation, the programming language JavaScript and the Twilio platform are used, with access through the API. The created bot can itself send SMS to the desired numbers, but also respond to SMS sent to it.
Keywords: NodeJS, Twilio, бот

В настоящее время клиенты хотят общаться с брендами, компаниями и организациями так же непринужденно, как и со своими друзьями, и ожидают немедленного ответа. Обеспечение такого уровня обслуживания для большинства организаций практически непрактично, если не с точки зрения логистики, без использования какой-либо формы автоматизации.

До недавнего времени ограничения автоматизированных технологий означали нарушение целостности и надежности работы, которая, как было доказано, способствует созданию лояльных клиентов. Создание центра обработки вызовов - дорогостоящее мероприятие, и тем не менее, чтобы обеспечить этот канал мгновенной связи с клиентами, большинство брендов и компаний предпочли это сделать, чтобы обеспечить мгновенный ответ.

С появлением API и фреймворков искусственного интеллекта (AI), обработки естественного языка (NLP), машинного обучения (ML) и анализа настроений полуавтоматические или полностью автоматизированные агенты, известные как боты, радикально меняют все, что знаем об общении с клиентами. инициировать революцию в способах взаимодействия с клиентами.

Поскольку все меньше людей используют свои телефоны для телефонных звонков, а вместо этого используют свои телефоны для чего-то еще, кроме разговоров, обмен сообщениями стал де-факто способом общения.

Многие владельцы смартфонов используют свои устройства для звонков, но большинство используют их для текстового общения (текстовые сообщения / SMS, обмен сообщениями или чат). В среднем взрослый тратит на переписку 23 часа в неделю. Кроме того, за всю жизнь средний миллениал потратит на переписку 12 лет.

Причина распространения текстовых сообщений как коммуникационной платформы заключается в том, что телефонные звонки прерываются, неудобны и неэффективны. Они не позволяют выполнять многозадачность: когда звоните со смартфона, его нельзя использовать ни для чего другого. Если раньше просто снимали трубку, чтобы решить проблему, то теперь начинаем с текстовых сообщений, а затем переходим к голосовым.

Еще одна важная причина для внедрения обмена сообщениями — это то, что клиенты требуют взаимодействия там, где они уже есть.

Приложения для обмена сообщениями и чата быстро набирают популярность по сравнению с SMS, особенно среди молодежи. В мире 6 из 10 лучших приложений — это приложения для обмена сообщениями, такие как Facebook Messenger, WhatsApp, Telegram и WeChat.

Основная причина такого увеличения использования приложений для обмена сообщениями заключается в том, что они не учитываются в ежемесячных лимитах SMS, а если подключены к Wi-Fi, они также не используют никаких данных. Кроме того, есть эмоциональная составляющая, которая улучшает общий разговор. Обмен сообщениями похож на разговор в реальном времени. знаете, когда друзья активны в приложении и даже когда они набирают ответ, что делает его захватывающим и очень интересным средством общения.

С учетом этого сценария создание ботов для обмена сообщениями, которые обеспечивают конструктивное взаимодействие с клиентами, дает передовое преимущество любому бизнесу за счет использования наиболее распространенных на сегодняшний день средств связи, а также нахождения там, где клиенты уже находятся, в своих приложениях для обмена сообщениями.

Цель исследования – создать бота отвечающего на смс по запросу.

Ранее этим вопросом интересовался А.В. Алексахин развивал тему «Программа для автоматизированной рассылки cmc-сообщений по всему миру» [1] в которой приведена программа предназначеная для автоматизированной рассылки СМС-сообщений по всему миру. Для отправки СМС-сообщений используются следующие провайдеры (каналы), исходя из международного телефонного кода получателя: Россия и страны СНГ - SMSSimple (targetSMS-резервный канал); США, Австралия, Франция, Англия, Япония - Twilio; Остальные страны - Sinch. ОС: Windows. А.Г.

Муталибова с темой «Создание вопросно-ответной системы для строительной компании» [2], а подробнее про проблему возможности круглосуточного первичного взаимодействия с клиентами предприятия, занимающегося строительной деятельностью; описана специфика предприятия и производимых им услуг; выделены наиболее оптимальные подходы и сервисы по созданию вопросно-ответной системы, и в последствие разработано дерево-решений для вопросно-ответной системы, которая также была создана в рамках данной работы. В.А. Кожевников, О.Ю. Сабинин, Ю.Е. Шац опубликовали статью «Современные мессенджеры в качестве помощника администратора базы данных» [3] описали вопрос использования современных мессенджеров и их возможностей администраторами баз данных. Описывается процесс создания бота, который позволяет администратору получать своевременные уведомления о проблемах и ошибках, произошедших с базой данных, а также получать статистику использования. Принципиальным отличием от других подобных продуктов является реализация для мессенджеров, которые в данный момент пользуются популярностью. Предусматривается описание разработки с использованием платформ Telegram, Facebook Messenger и Slack, на основе единой библиотеки.

В широком смысле этого определения бот — это программа, которая использует узкий искусственный интеллект для выполнения определенных задач вместо человека. Боты в определенной степени понимают язык, а не только команды. В конце концов, они могут научиться на своем взаимодействии, чтобы стать умнее и лучше.

Сегодня потребители во всем мире требуют обмена сообщениями как возможности обслуживания клиентов. Недостаточно иметь номер телефона службы поддержки, по которому клиент может позвонить, но становится почти обязательным, чтобы клиенты могли связаться с через какую-то платформу обмена сообщениями в реальном времени. Пользователи требуют быстрого взаимодействия и быстрых ответов.

Почти две трети потребителей, вероятно, будут иметь положительное восприятие организации, которая предлагает обмен сообщениями или чат как канал обслуживания. Тем не менее, к концу 2018 года примерно 40% центров обслуживания клиентов все еще будут упускать возможность произвести впечатление на своих клиентов. Это означает не только невозможность произвести впечатление на клиентов, но и потерю деловых возможностей. Клиенты, вероятно, будут более лояльными и останутся в тех организациях, которые способны взаимодействовать и взаимодействовать с ними более быстрыми и разумными способами.

Использование приложения для обмена личными сообщениями (например, Skype, Facebook Messenger, WhatsApp и т. д.) устраняет большинство препятствий, позволяя компаниям стать частью среды общения, которую пользователи уже знают и любят.

В приложениях для обмена сообщениями нет ни форм, ни загрузок, ни новых платформ. Клиент может использовать интерфейс, с которым он уже

знаком, для мгновенного взаимодействия с организацией. Пользователь может использовать естественный язык, чтобы купить билет, загрузить посадочный талон или задать вопрос. Более того, учитывая, что пользователь вряд ли перестанет использовать приложение для обмена сообщениями, организация может отслеживать обновления, опросы и другие уведомления через приложение для обмена сообщениями, которое пользователь уже знает и любит.

Наверняка будут случаи, когда боты могут столкнуться с ситуациями, требующими нюансов и аналитического мышления человека. Когда они это сделают, они могут перейти к агенту, передавая контекст, который они собрали во время взаимодействия, чтобы обеспечить бесперебойное обслуживание клиентов. В принципе, это должно быть полностью прозрачным для конечного пользователя.

Боты также могут:

- Разрешить клиентам совершать покупки, не выходя из приложения для обмена сообщениями.
- Предлагайте индивидуальные предложения по продукту.
- Связывайте пользователей с соответствующими веб-страницами, такими как обзоры продуктов клиентов.
- Инициируйте новые взаимодействия, чтобы повторно привлечь пользователей.
- Напоминания о корзине и истории успеха клиентов.

Смартфоны становятся все более важными в современном мире. Если сегодня человек потеряет телефон, то все, от электронной почты, календаря, обмена сообщениями, банковских операций и даже кошелька, каким-то образом связано с телефоном.

В сегодняшнем ярком, динамичном и всегда связанном обществе доступ к огромному количеству информации на кончиках пальцев через телефон может быть благословением, но это также может быть проклятием.

Сегодня занятым специалистам приходится иметь дело с сотнями электронных писем ежедневно или еженедельно, а также с множеством сообщений и уведомлений из социальных сетей, таких как Twitter и LinkedIn. Не отставать от такого огромного количества сообщений может быть непросто.

Но что, если телефоны действительно могли бы помочь облегчить часть этой информационной перегрузки, уведомляя о важных вещах или позволяя выполнять настраиваемые действия на основе SMS или голосовых команд?

До того, как социальные сети стали популярными, служба коротких сообщений (SMS) была наиболее распространенным способом обмена короткими сообщениями между людьми.

Согласно Википедии несмотря на то, что SMS по-прежнему сильны и растут, для обмена короткими сообщениями все чаще используются службы

обмена сообщениями в социальных сетях, такие как Facebook Messenger, WhatsApp, Skype и Viber, доступные на смартфонах.

Вообще говоря, решения с поддержкой SMS и голоса зависят от платформы и не могут быть настроены; однако есть платформа, которая была разработана с нуля для разработчиков, что позволяет любому, у кого есть навыки разработки, создавать собственные решения для обмена сообщениями и голоса. Добро пожаловать в Twilio! (https://www.twilio.com/).

Twilio — это API для обмена сообщениями, голоса, видео и аутентификации для каждого приложения. У него есть вспомогательные библиотеки или SDK на многих разных языках программирования, которые помогают разработчикам создавать приложения, которые могут использовать возможности голоса и обмена сообщениями.

Несмотря на это, SMS по-прежнему очень сильны и широко используются в корпоративных разработках для таких вещей, как маркетинг, автоматизация управления взаимоотношениями с клиентами (CRM), уведомления о предупреждениях в реальном времени и двухэтапная проверка личности пользователя.

Значение использования SMS в деловом мире невероятно важно, учитывая, что технология считается зрелой, широко распространенной, проверенной и надежной.

Доступ к сервисам Twilio осуществляется через HTTP (S) через RESTful API или вспомогательные библиотеки. Его услуги оплачиваются в зависимости от использования. Платформа основана на Amazon Web Services (AWS) для размещения инфраструктуры телефонной связи и обеспечения связи между HTTP и коммутируемой телефонной сетью общего пользования (PSTN) через свои API.

Twilio недавно расширила поддержку API для Facebook Messenger, что совпадает с введением компанией социальных сетей поддержки ботов на своей платформе Messenger.

Twilio предоставляет REST API, который позволяет разработчикам взаимодействовать со службами своей платформы, такими как SMS. Несмотря на то, что REST API - отличный способ взаимодействия со службами Twilio, существуют официальные вспомогательные библиотеки для наиболее распространенных на сегодняшний день языков программирования, таких как: PHP, ASP.NET (C #), Ruby, Python, Java, Salesforce (Apex) и, наконец, что не менее важно, Node.js.

Вспомогательную библиотеку Twilio Node.js можно получить по адресу https://www.twilio.com/docs/libraries/node. Для начала нужно установить Node.js, его можно найти по адресу https://nodejs.org, где на главной странице можете загрузить версию Node.js, соответствующую платформе, как на рисунке 1.

Рис. 1 Главная страница NodeJS

После выбора версии, нужно запустить установщик и проследовать базовый шагам установщика.

После установки Node.js необходимо установить вспомогательную библиотеку Twilio Node.js.

Для этого нужно создать в любом месте на компьютере папку для этого проекта, перейти к этой папке, затем открыть командную строку или оболочку и ввести эту команду:

```
npm init
```

Это создаст package.json файл, необходимый для проекта, с содержимым как на рисунке 2.

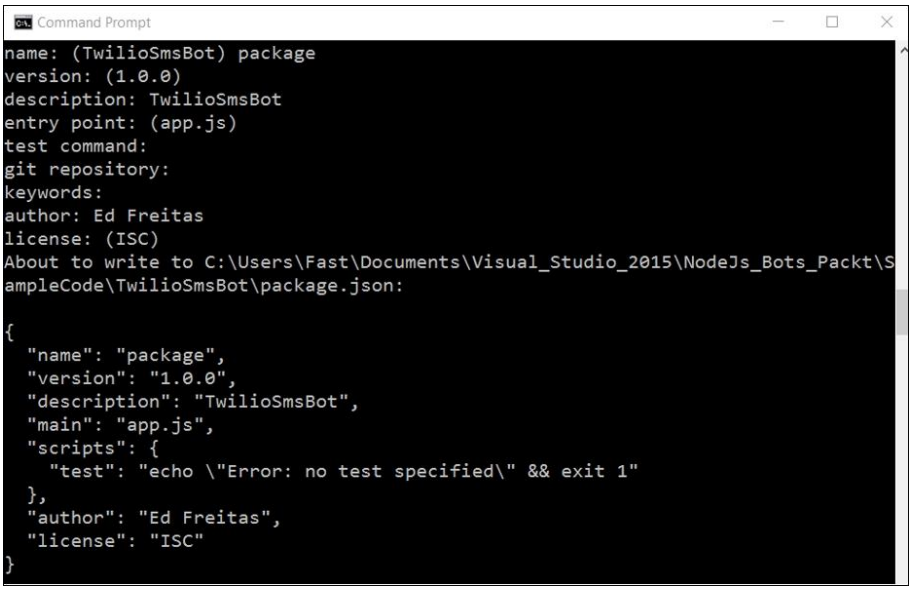

Рис. 2 Вывод команды

После создания package.json файла нужно ввести следующую команду:

```
npm install twilio --save
```

Это установит вспомогательную библиотеку Twilio Node.js и все ее зависимости и сохранит ссылку в package.json файле. Библиотека Twilio будет установлена в node_modules папку внутри папки, в которой находится package.json файл.

На данном этопе, можно начать писать код, но, сначала нужно зарегистрировать учетную запись Twilio и настроить Twilio, прежде чем отправлять первое SMS.

Чтобы иметь возможность отправлять SMS с помощью Twilio API и вспомогательной библиотеки Node.js, необходимо настроить учетную запись Twilio, а также приобрести одноразовый номер Twilio.

Twilio — это услуга с оплатой по мере использования, что означает, что необходимо создать учетную запись и предоставить данные своей кредитной карты, чтобы иметь достаточный кредит, который будет использоваться для оплаты каждого отправляемого SMS.

Также необходимо приобрести номер Twilio, который является обычным, но одноразовым номером телефона, который будет использоваться для отправки сообщений.

Номера Twilio доступны для многих стран. Они выглядят как любой другой действительный номер телефона, который можете придумать. Это настоящие телефонные номера, которыми можете распорядиться, когда они больше не нужны.

Чтобы создать учетную запись Twilio, нужно перейти на сайт https://www.twilio.com/. Затем нажать кнопку ПОДПИСАТЬСЯ.

Процесс регистрации довольно прост, его очень легко отслеживать и выполнять. Просто заполнить несколько обязательных полей, и все готово.

Когда на учетной записи Twilio появятся средства, нужно будет приобрести одноразовый номер телефона.

Нужно перейти по этому адресу https://www.twilio.com/user/billing, чтобы пополнить свой счет. Для этого нужно нажать на ссылку «Рекламные фонды», выделенную красным. Перед доступом к этому URL-адресу убедитесь, что вошли в Twilio, как на рисунке 3.

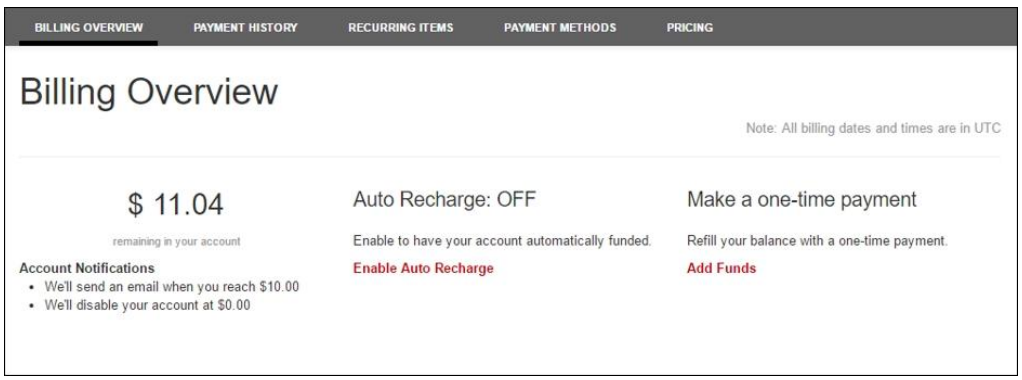

Рис. 3 Окно финансов в Twilio

Имея средства на счете, можно настроим номер Twilio. Это будет реальный номер телефона, который можно удалить в любой момент. можно выбрать, из какой страны и города будет принадлежать номер.

Затем нажать кнопку «Купить номер», как на рисунке 4.

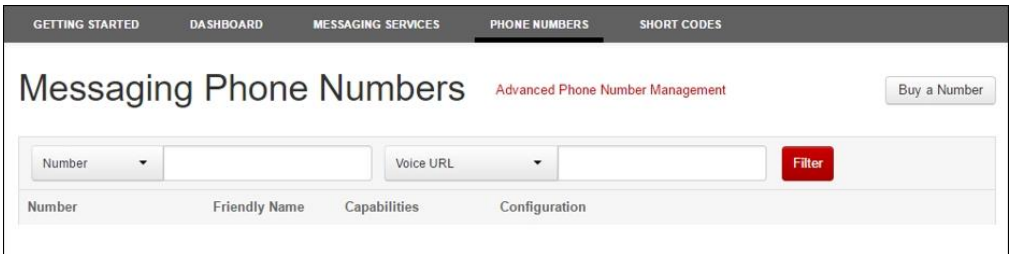

Рис. 4 Покупка номера

После нажатия кнопки «Купить номер», появится следующий всплывающий экран, как на рисунке 5.

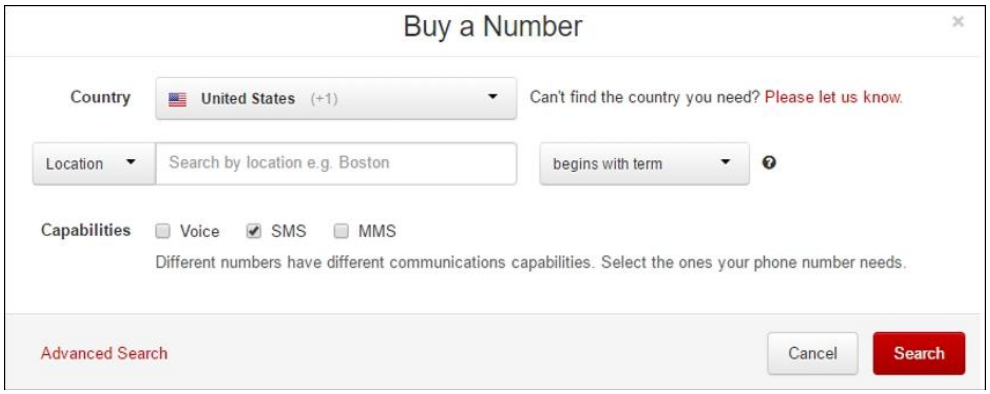

Рис. 5 Выбор страны телефона

На этом экране есть возможность выбрать, из какой страны получить номер, а также из какого географического местоположения.

Номер может быть использован для передачи голоса, SMS и даже MMS. На данный момент просто заинтересованы в том, чтобы поставить галочку напротив опции SMS.

После успешного приобретения номера Twilio, будет следующий экран, как на рисунке 6.

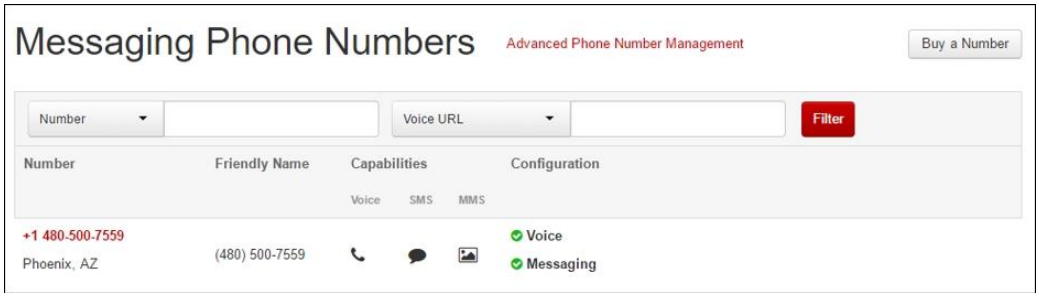

Рис. 6 Успешное создание номера

Чтобы начать писать код, нужно создать файл app.js, можно создать этот новый файл прямо из редактора.

Как только файл будет создан, нужно будет включить ссылку на библиотеку Twilio Node.js, которая была установлена через npm.

```
var twilio = require("twilio");
```

В мире Node.js это эквивалент импорта в Java или операторов using в C #. Отправка сообщений осуществляется с помощью кода ниже.

```
var accountSid = '<< your twilio account sid >>';
var authToken = '<< your twilio auth token >>';
```

Две переменные для хранения SID учетной записи Twilio и токена аутентификации. Оба значения можно получить, войдя в свою учетную запись Twilio и перейдя в консоль разработчика: https://www.twilio.com/console, как показано на рисунке 7.

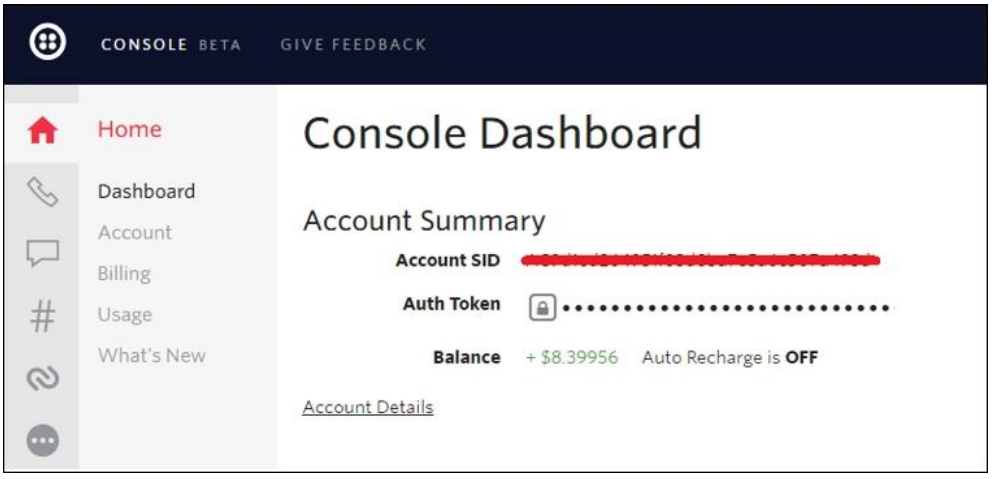

Рис. 7 Страница с ключами доступа

После того, как указали правильные значения для accountSid и authToken переменных, нужно создать экземпляр twilio.RestClient класса для того , чтобы иметь возможность отправить SMS:

```
var client = new twilio.RestClient(accountSid, authToken);
```

Создав экземпляр, можно отправить SMS с помощью Twilio:

```
client.messages.create({
    body: 'Greetings earthling, this is the TwilioSmsBot ;)',
    to: '+12345678901', from: '+12345678901'},
function(err, message) {console.log(message.sid);});
```

Обычно SMS отправляется путем вызова messages.create метода из clientэкземпляра Twilio .

Этот метод ожидает объект, который описывает свойства SMS, такие как body.to(номер получателя), from(номер отправителя), функция обратного вызова, которая описывает ошибку err(если ошибка действительно происходит), и содержимое отправленного сообщения message.

Это все, что требуется для отправки SMS с помощью Twilio. Ниже представлен полный код.

```
var twilio = require("node_modules/twilio/lib");
var accountSid = '<< your twilio account sid >>';
var authToken = '<< your twilio auth token >>';
var client = new twilio.RestClient(accountSid, authToken);
client.messages.create({
    body: 'Hello from Node',
    to: '+12345678901',
    from: '+12345678901'},
function(err, message) {
    console.log(message.sid);
});
```

Чтобы выполнить этот код, нужно запустить команду ниже.

```
node app.js
```

Это отправит SMS на указанный номер. можно увидеть message.sid отправленное SMS (которое было отправлено обратно как ответ от службы Twilio), посмотрев на командную строку, как на рисунке 8.

```
C:\Users\Fast\Documents\Visual_Studio_2015\NodeJs_Bots_Packt\SampleCode\TwilioSm
sBot>node app.js
SM6344d84ce7c84bfbbfe020146692cad3
```

Рис. 8 Ответ об успешной отправке SMS

Если номер назначения, по которому будете отправлять сообщения, является международным (не в США), необходимо включить определенные разрешения, чтобы Twilio могла выполнять это действие.

Эти разрешения можно проверить и настроить по этому URL-адресу: https://www.twilio.com/console/voice/settings/geo-permissions.

Для прослушивания входящих SMS необходимо настроить в приложении Node.js URL-адрес, который можно настроить в учетной записи Twilio как URL-адрес запроса. Этот URL-адрес запроса будет использоваться Twilio для отправки входящих сообщений на купленный номер Twilio в бот-приложение Node.js.

Чтобы сделать бота общедоступным, нужно разместить его на веб-сайтах Azure. Нужно создать конечную точку REST для приложения Node.js, которую можно использовать для прослушивания новых сообщений. Для этого нужно использовать платформу Express (http://expressjs.com/).

Express — это минимальная и гибкая платформа веб-приложений Node.js, которая обеспечивает надежный набор функций для веб-приложений и мобильных приложений. Он предоставляет тонкий слой основных функций веб-приложений, таких как маршрутизация и промежуточное ПО.

Сначала нужно установить платформу Express, выполнив эту команду из командной строки:

```
npm install express --save
```

Устанавливается удобная утилита Nodemon (http://nodemon.io/). Это позволяет вносить изменения в свой код и автоматически перезапускать приложение Node.js. можно установить Nodemon, выполнив эту команду из командной строки:

```
npm install nodemon --save
```

Вместо того, чтобы запускать приложение с узлом, теперь можно запустить его следующим образом:

```
nodemon app.js
```

Установив платформу Express, нужно расширить текущий код, чтобы создать конечную точку REST, которую затем можно использовать для подключения Twilio для отправки входящих сообщений:

```
var express = require('express');
var app = express();
app.get('/receive', function (req, res) {
  res.send('Hi, this is the TwilioBot listening endpoint!');});
app.listen(8080, function () {
  console.log('TwilioBot listening on port 8080.');
});
```

Прежде чем подключить бот-приложение к Twilio для обработки входящих сообщений, нужно подключим все инструменты, чтобы могли опубликовать приложение как оно есть на веб-сайтах Azure.

Нужно будет установить интерфейс командной строки (CLI) Azure (https://azure.microsoft.com/en-us/documentation/articles/xplat-cli-install/), чтобы отправить приложение в Azure. также необходимо будет зарегистрироваться в Azure, если нет учетной записи.

Все это можно сделать это, посетив: https://azure.microsoft.com.

После настройки учетной записи в Azure можно установить Azure CLI, используя соответствующий установщик для платформы или как пакет npm, следуя этим инструкциям.

Используя npm, Azure CLI можно установить следующим образом:

```
npm install azure-cli -g
```

После установки Azure CLI развернет приложение в Azure, чтобы убедиться, что все инструменты правильно подключены.

Для этого нужно запустить Azure CLI и login в Azure:

```
azure login
```

После выполнения команды должно появится следующее приветственное сообщение, в котором попросят включить сбор данных, как на рисунке 9.

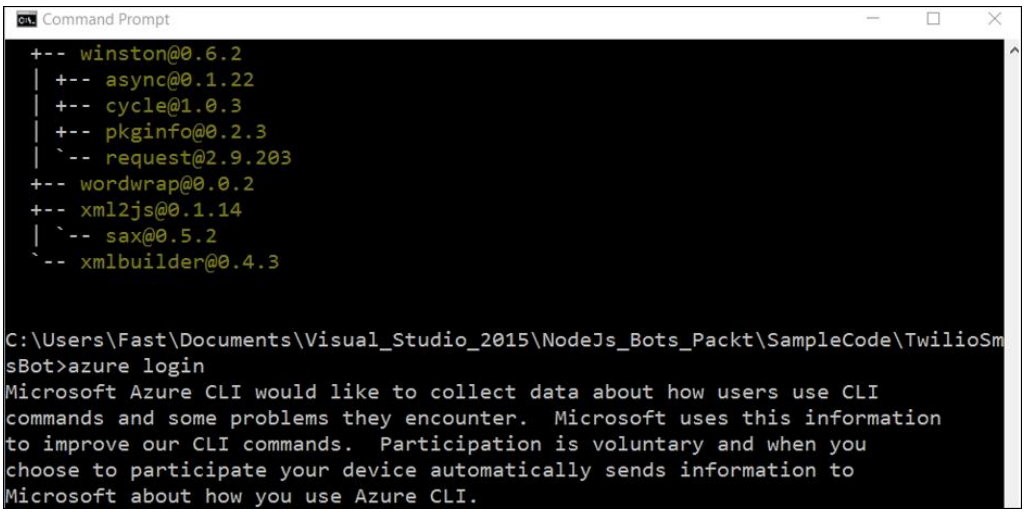

Рис. 9 Авторизация в Azure

Можете согласиться или нет, и это никак не влияет на разработку бот-приложения или использование Azure.

После того, как выбрали свой вариант, будет предложено ввести код, отображаемый в командной строке поэтому URL-адресу, http://aka.ms/devicelogin, а затем пройти аутентификацию с помощью своей учетной записи Microsoft, как показано ниже, на рисунке 10.

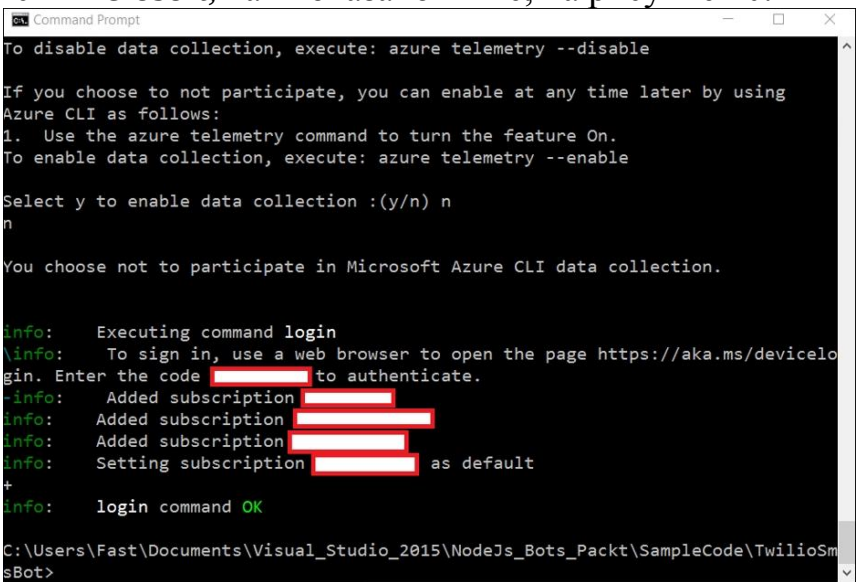

Рис. 10 Информация для связи с Azure

Теперь у Azure CLI все готово. Следующее, что нужно сделать, — это развернуть приложение в Azure с помощью интерфейса командной строки.

Следует выполните эту команду, чтобы создать веб-сайт в Azure. Нужно создайте ресурс приложения службы приложений в Azure с уникальным именем приложения с помощью следующей команды. URL-адрес веб-приложения будет http://<appname>. azurewebsites.net.

В этом случае именем приложения службы приложений в Azure, будет NodeBotSite (можете выбрать любое другое имя, если оно уже занято).

```
azure site create --git nodebotsite
```

Будет предложено выбрать регион Azure, в котором будет размещен сайт, как на рисунке 11.

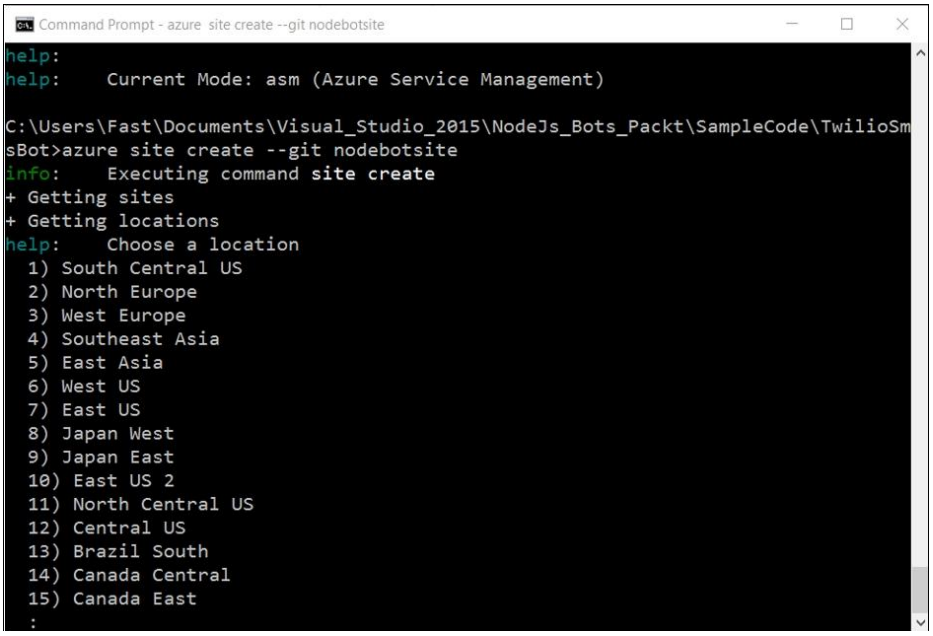

Рис. 11 Выбор региона

После выбора региона, Azure создаст сайт, и выдаст следующие сведения в командной строке, как на рисунке 12.

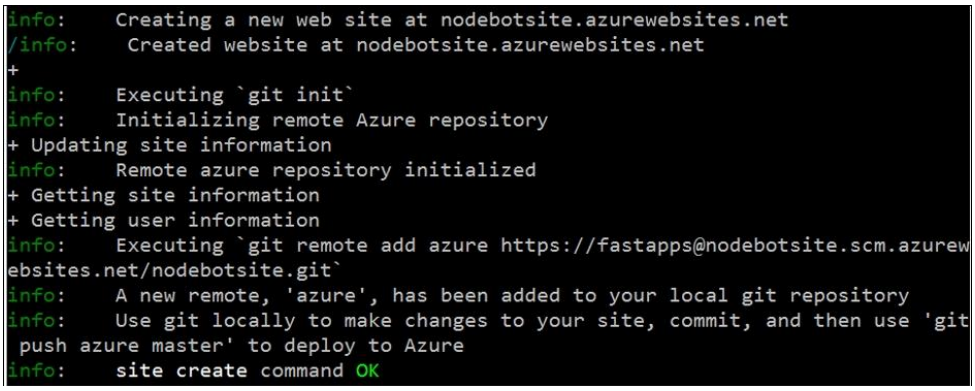

Рис. 12 Вывод информации по сайту

Следует изменить порт 8080 в приложении на process.env.port, как показано ниже:

```
app.listen(process.env.port, function () {
  console.log('Hi, this is the TwilioBot listening endpoint!');});
```

Nodemon, похоже, не слишком хорошо работает с Azure; поэтому, если оставите Nodemon как зависимость от package.json файла, можно столкнуться с проблемами при развертывании приложения в Azure. В свете этого лучше удалить из файла package.json зависимость, которая ссылается на Nodemon, перед развертыванием в Azure.

Сохранить изменения как в своих файлах, так package.json и в app.js файлах, а затем использовать команду git для развертывания приложения в Azure следующим образом:

```
git add .
git commit -m "TwilioNodeBot first commit"
git push azure master
```

После завершения команды git push приложение будет опубликовано в Azure и готово к использованию.

Чтобы просмотреть его, нужно открыть свой браузер и перейти на сайт, полученный этим URL: http://nodebotsite.azurewebsites.net/receive. Сообщение должно быть как на рисунке 13.

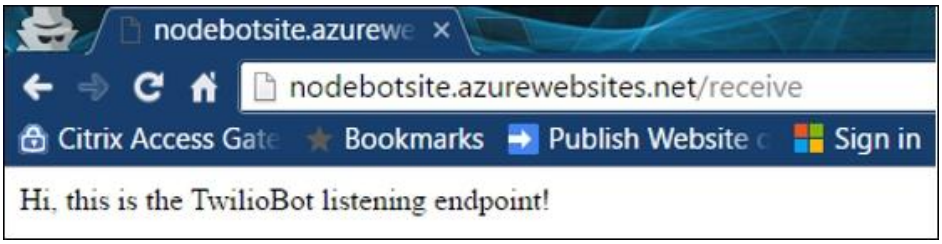

Рис. 13 Проверка бота через браузер

Для того, чтобы обновления веб - приложения Node.js работали на Azure, нужно запустить git add, git commit и git push как делали, когда развернули его первый.

Когда все подключено к Azure, следующим шагом будет настройка этого URL-адреса: https://www.twilio.com/console/phone-numbers/incoming, на панели инструментов голосового номера в учетной записи Twilio, а затем на приобретенный Twilio. URL-адрес номера Twilio и нажмите кнопку «Сохранить», как на рисунке 14.

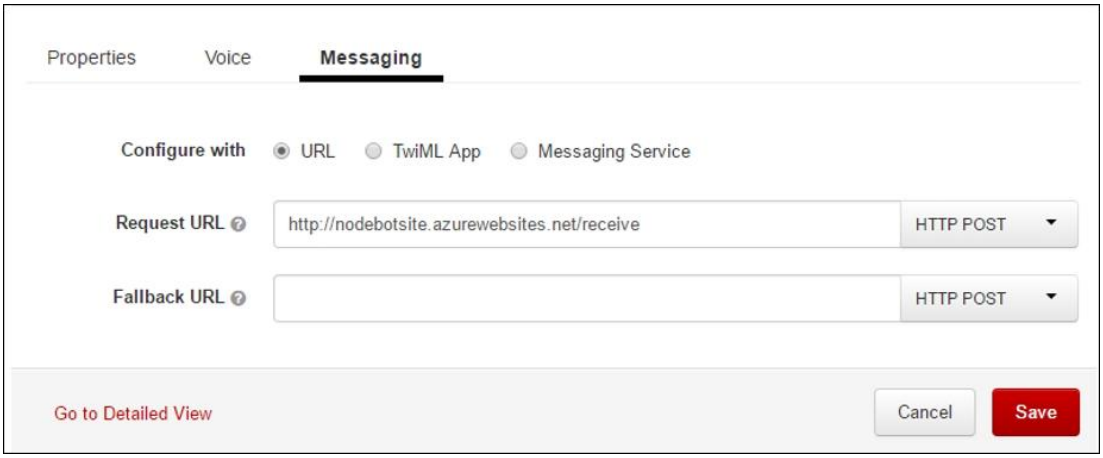

Рис. 14 Заполнения поля точки входа

Теперь Twilio привязан к приложению Azure, получает URL-адрес, который будет использоваться для получения входящего SMS.

После выполнения всех этих шагов по настройке можно сосредоточиться на добавлении логики приема для бот-приложения Twilio.

До сих пор был реализовн базовый шаблон для бот-приложения Twilio, а также выполнены все необходимые настройки, чтобы решение было подключено к Twilio, а также было легко развернуто в Azure.

Бота можно заставить отвечать на входящие сообщения. Для этого понадобится конечная точка POST в приложении Node/Express. Нужно рассмотреть следующий код:

```
app.post('/receive', function (req, res) {
  var twiml = new twilio.TwimlResponse();
  twiml.message('Hi, this is TwilioBot');
  res.writeHead(200, {'Content-Type': 'text/xml'});
  res.end(twiml.toString());
});
```

В нем можно заметить, что для ответа создается ответ TwiML и отправляется в качестве ответа POST / полученной конечной точки HTTP.

TwiML — это язык разметки XML, который представляет собой просто набор инструкций, которые можете использовать, чтобы сообщить Twilio, что делать, когда получает входящий звонок или SMS.

Twilio отправляет HTTP-запросы к приложению, как обычный веб-браузер. Включая параметры и значения в свои запросы, Twilio отправляет данные в приложение, с которыми можете действовать, прежде чем отвечать. Это то, что на самом деле делается с конечной точкой получения.

Twilio отправляет следующие параметры со своим запросом как параметры POST или параметры запроса URL, в зависимости от того, какой метод HTTP настроили.

При получении SMS или телефонного звонока на номер Twilio, Twilio получает URL-адрес, связанный с этим номером телефона, и выполняет HTTP-запрос на этот URL-адрес. Этот URL-адрес будет содержать XML-ответ с инструкциями по разметке, которые указывают, какие задачи Twilio необходимо выполнить. Некоторые из этих задач могут заключаться в записи разговора, воспроизведении сообщения, предложении вызывающему абоненту ввести несколько цифр и так далее.

В этом случае бот просто возвращает однострочное предложение. Короче говоря, предыдущий пример кода просто возвращает этот XML обратно в Twilio в качестве ответа, так что Twilio может фактически сгенерировать из него SMS-ответ и отправить его обратно на телефон отправителя:

```
<?xml version="1.0" encoding="UTF-8"?>
<Response>
<Say> Hi, this is TwilioBot.</Say>
</Response>
```

Очень просто указать Twilio выполнить определенное действие с помощью TwiML. Язык разметки состоит из выделенных синим цветом глаголов, которые представляют действия, которые Twilio выполнит.

Итак, бот начинает наконец обретать форму, Он может прослушивать сообщения и отправлять ответ. Но как бот знает, как действовать на определенные входные данные? Для этого ботам необходимо понимать параметры входящего сообщения и действовать в соответствии с ними.

Twilio отправляет несколько параметров со своим запросом в виде параметров POST или параметров запроса URL, в зависимости от того, какой метод HTTP был настроен на панели управления номера Twilio для входящего SMS.

Итак, чтобы написать некоторую логику, которая действует на ввод информации о характере полученных сообщений, необходимо проверить значение параметров, полученных как часть request.body объекта. Эти параметры будут иметь свойства этого объекта.

Так, например, если нужно узнать, с какого номера пришло сообщение, нужно будет сделать что-то вроде этого:

```
var from = req.body.From;
```

Фактическое полученное текстовое сообщение будет получено следующим образом:

```
var body = req.body.Body;
```

Зная, откуда исходит сообщение и фактическое содержимое, можно затем построить некоторую логику, чтобы внутренне дать боту некоторое представление о том, что делать с полученными входными данными и действовать в соответствии с ними.

Вывод

В этой статье заложили основы того, как бот будет развернут, настроен и настроен. также разобран простой шаблон.

Также изучили некоторые ключевые компоненты, которые позволят ему работать с точки зрения инфраструктуры, и то, как можно разместить его на Azure.

В дополнение к этому, были изучены основы Twilio как платформы обмена сообщениями и начало работы с ним.

Библиографический список

1. Алексахин А.В. Программа для автоматизированной рассылки cmc-сообщений по всему миру // Свидетельство о регистрации программы для ЭВМ RU 2019664708, 13.11.2019. Заявка № 2019619830 от 07.08.2019. URL: https://www.elibrary.ru/item.asp?id=41364636 (Дата обращения: 29.01.2021)
2. Муталибова А.Г. Создание вопросно-ответной системы для строительной компании // Точная наука. 2019. № 43. С. 31-38. URL: https://www.elibrary.ru/item.asp?id=38254018 (Дата обращения: 29.01.2021)

3. Кожевников В.А., Сабинин О.Ю., Шац Ю.Е. Современные мессенджеры в качестве помощника администратора базы данных // ScienceRise. 2018. Т. 6. С. 32-36. URL: https://www.elibrary.ru/item.asp?id=35780278 (Дата обращения: 29.01.2021)