

## Создание бота помощника в Skype

*Кизянов Антон Олегович*

*Приамурский государственный университет имени Шолом-Алейхема*

*Студент*

### **Аннотация**

В данной статье описан процесс создания бота помощника в среде Skype. Для создания используется аккаунт skype, среда NodeJS и BotFramework для создания ботов. Созданный макет позволяет частично заменить HR сотрудника, тем, что может сам предоставить информацию по отпуску или больничному каждому пользователю.

**Ключевые слова:** Skype, BotFramework, NodeJS, бот

## Creating an assistant bot in Skype

*Kizyanov Anton Olegovich*

*Sholom-Aleichem Priamursky State University*

*student*

### **Abstract**

This article describes the process of creating an assistant bot in the Skype environment. To create a skype account, the NodeJS environment and BotFramework are used to create bots. The created layout allows you to partially replace the HR of an employee by the fact that he himself can provide information on vacation or sick leave to each user.

**Keywords:** Skype, BotFramework, NodeJS, bot

Skype - отличное программное обеспечение и надежная платформа, которую используют миллионы людей во всем мире для совершения звонков, организации встреч и общения друг с другом. Он используется как для личного общения, так и для бизнеса.

Одна из замечательных особенностей Skype заключается в том, что он позволяет совершать бесплатные одноранговые VoIP-звонки с любым другим пользователем, у которого также есть учетная запись Skype. Он также позволяет звонить по телефонным номерам по очень низким ценам и даже бесплатно в некоторые места.

Кроме того, Skype также позволяет принимать входящие звонки на реальный номер телефона или переводить их в текстовые сообщения. Он также позволяет пересылку сообщений, конференцсвязь, групповой чат, передачу файлов, презентации на удаленном рабочем столе, просмотр и многие другие функции.

Skype можно использовать в качестве автоматизированного агента, который может помочь выполнить некоторую работу и автоматизировать некоторые бизнес-процессы.

Skype является частью Microsoft, и недавно на мероприятии разработчиков сборки была представлена платформа для создания интерактивных ботов с помощью Skype. Skype уже имеет набор классных и чрезвычайно полезных API-интерфейсов, которые позволяют разработчикам относительно легко взаимодействовать со службой и отлично подходят для всех видов приложений, связанных с голосовой связью и чатом. Однако у него нет API, который ориентирован исключительно на автоматизацию интерактивных сообщений, и именно здесь Bot Framework заполняет пробел.

В этой статье будет создан бот, который действует как виртуальный помощник по работе с персоналом (HR), который будет предоставлять информацию о днях отпуска, периодах уведомления и других HR запросов.

Цель исследования – создать бота помощника в Skype.

Ранее этим вопросом интересовались Р.К. Потапова, А.Н. Собакин, А.В. Маслов развивали тему «Возможность идентификации говорящего по голосу в системе интернет-телефонии skype» [1] в которой рассмотрен импульсный метод преобразования речи в применении к проблеме идентификации говорящего по голосу в системе интернет-телефонии Skype. С целью выявления индивидуальных особенностей функционирования голосового источника исследовались речевые сигналы, записанные в акустической студии, и те же сигналы, прошедшие канал передачи IP-телефонии Skype. С.Н. Прохорова с темой «Использование скайп-технологий в дистанционном обучении» [2], а подробнее про опыт использования скайп-технологий в организации дистанционного обучения на факультете филологии и коммуникации. С использованием методики включенного наблюдения и анкетирования предпринимается попытка установить условия эффективного использования данной технологии обучения и решить вопросы повышения качества образования студентов-филологов. С.В. Григорьева, Н.Г. Игошкина, О.Г. Васильева опубликовали статью «Организация мониторинга сервиса skype for business для оптимизации работы отдела поддержки сервисов» [3] описали разработку системы мониторинга сервиса Skype for Business для оптимизации работы отдела поддержки сервисов. Разработка системы проводилось с использованием программного обеспечения Microsoft Power BI. Разработанная система мониторинга способна выполнять следующие функции: возможность наблюдать за трендом использования сервиса; оперативное обновление данных в системе; наблюдение за самыми частыми ошибками; определение тренда ошибок; сбор сведений о качестве звонков.

Бот Skype, просто еще один контакт Skype, разница в том, что вместо разговора с другим человеком это автоматизированный процесс, который знает, как ответить на введенные данные. Боты могут делать множество

вещей, например получать новости, проверять погоду, получать фотографии или информацию с веб-сайтов, запускать игру, заказывать еду или такси.

Все, что можно превратить в сервис, можно превратить в автоматизированный разговор с помощью бота. С помощью Skype боты могут вести интерактивные беседы практически на любой платформе, в любое время и в любом месте.

Пользователи могут отправлять запросы ботов Skype, а бот может отправлять содержательные отзывы на основе полученного контента. Бот Skype также может участвовать в групповом разговоре и отправлять сведения всем участникам этой группы.

Технически бот Skype состоит в том, что он подключается к платформе ботов и слушает их, используя API ботов Skype напрямую или используя C # или Node.js SDK.

Когда пользователь отправляет сообщение боту Skype. Бот отправляет ответы обратно на платформу ботов, которая передает их пользователю. Веб-перехватчики (которые являются действительными общедоступными URL-адресами - конечными точками обмена сообщениями HTTP) обычно работают в облачной службе, такой как Microsoft Azure.

Веб-перехватчики вызываются запросами в формате JSON. Каждый объект JSON указывает на какое-то обновление и выглядит так.

```
[{"activity": "message",  
  "from": "awesomeskypebot",  
  "to": "28:2c967451-ee01-421f-92aa-1a80f9e163dc",  
  "time": "2016-03-30T09:50:01.123Z",  
  "id": "1443805282113",  
  "content": "Hello from a Bot!"}]
```

По сути, бот проходит несколько этапов. Изначально бот может быть добавлен для разработки ограниченному числу пользователей, что позволяет редактировать детали бота, а также позволяет предварительно просматривать такие функции, как групповой чат или звонки. В данной статье, будет создан на чате (текстовое взаимодействие), а не на звонках, но полезно знать, что эта функция также возможна. Вот этапы:

За этапом создания или редактирования следует этап проверки, который происходит непосредственно перед публикацией бота. После проверки невозможности редактировать атрибуты бота на портале (например, его имя и другие свойства). Важно отметить, что бот не может быть отправлен с использованием функций предварительного просмотра, таких как групповой чат или звонок.

После того, как отзыв о боте принят, он переходит в стадию публикации. На данный момент бот может быть добавлен любым количеством пользователей через ссылку или кнопку URL-адреса бота.

Наконец, вскоре после публикации бот отображается в каталоге ботов Skype.

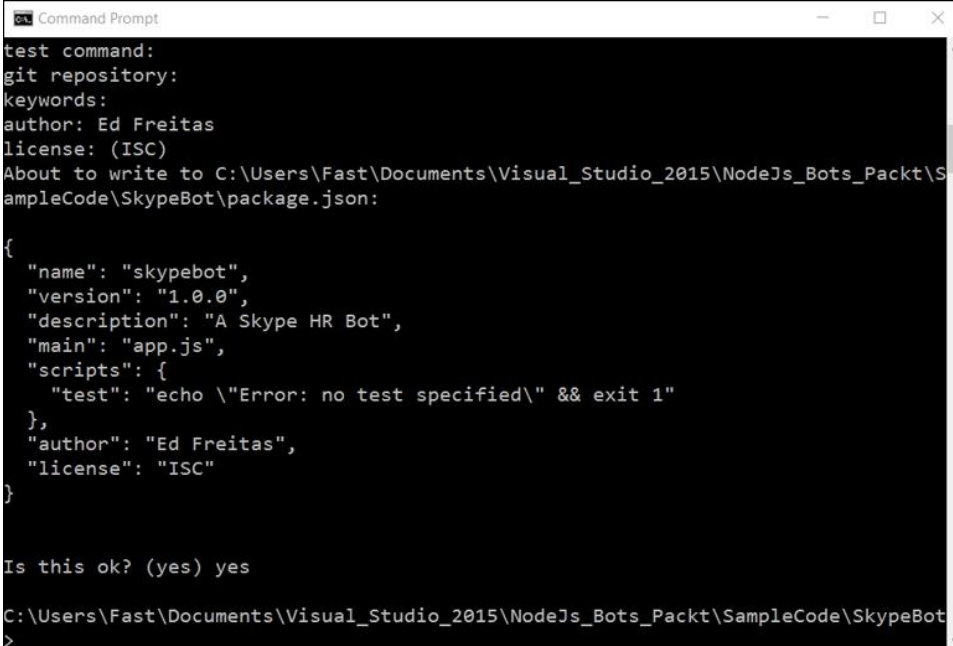
Сначала нужно создать папку на локальном диске из командной строки для хранения бота:

```
mkdir skypebot  
cd skypebot
```

Предполагается, что NodeJS и npm уже установлен в системе. Для создания файла package.json, который будет хранить зависимости и определения бота, нужно выполнить следующую команду.

```
npm init
```

При правильной инициализации в консоли будут следующие сообщения как на рисунке 1.



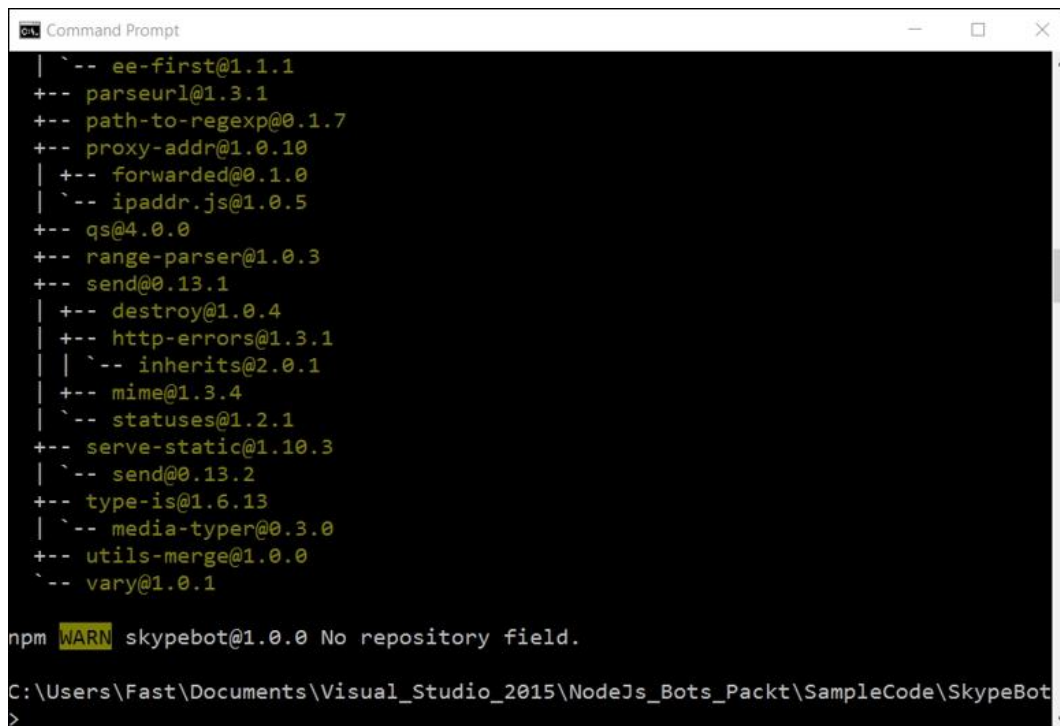
```
Command Prompt  
test command:  
git repository:  
keywords:  
author: Ed Freitas  
license: (ISC)  
About to write to C:\Users\Fast\Documents\Visual_Studio_2015\NodeJs_Bots_Packt\SampleCode\SkypeBot\package.json:  
  
{  
  "name": "skypebot",  
  "version": "1.0.0",  
  "description": "A Skype HR Bot",  
  "main": "app.js",  
  "scripts": {  
    "test": "echo \"Error: no test specified\" && exit 1"  
  },  
  "author": "Ed Freitas",  
  "license": "ISC"  
}  
  
Is this ok? (yes) yes  
C:\Users\Fast\Documents\Visual_Studio_2015\NodeJs_Bots_Packt\SampleCode\SkypeBot  
>
```

Рис. 1 Инициализация файла package.json

Будет использоваться Express в качестве структуры REST Node.js. Установить его можно командой ниже.

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

После установки Express, в консоли будет выглядеть как на рисунке 2.



```
Command Prompt
| `-- ee-first@1.1.1
+-- parseurl@1.3.1
+-- path-to-regexp@0.1.7
+-- proxy-addr@1.0.10
| +-- forwarded@0.1.0
| `-- ipaddr.js@1.0.5
+-- qs@4.0.0
+-- range-parser@1.0.3
+-- send@0.13.1
| +-- destroy@1.0.4
| +-- http-errors@1.3.1
| | `-- inherits@2.0.1
| +-- mime@1.3.4
| `-- statuses@1.2.1
+-- serve-static@1.10.3
| `-- send@0.13.2
+-- type-is@1.6.13
| `-- media-typer@0.3.0
+-- utils-merge@1.0.0
`-- vary@1.0.1

npm WARN skypebot@1.0.0 No repository field.

C:\Users\Fast\Documents\Visual_Studio_2015\NodeJs_Bots_Pack\SampleCode\SkypeBot
>
```

Рис. 2 Процесс установки Express

После нужно установить BotBuilder пакет, который соответствует библиотеке Microsoft Bot Framework Node.js.

Чтобы установить его, нужно запустить команду ниже.

```
npm install --save botbuilder
```

После всех установок, файл package.json должны выглядеть примерно как на рисунке 3.

=5.6.0' } }." data-bbox="198 571 865 853"/>

```
package.json — C:\Users\Fast\Documents\Visual_Studio_2015\NodeJs_Bots_Pack\SampleCode\SkypeBot\package.json — Atom
package.json
1 {
2   "name": "skypebot",
3   "version": "1.0.0",
4   "description": "A Skype HR Bot",
5   "main": "app.js",
6   "scripts": {
7     "test": "echo \\\"Error: no test specified\\\" && exit 1"
8   },
9   "author": "Ed Freitas",
10  "license": "ISC",
11  "dependencies": {
12    "express": "^4.13.4",
13    "skype-sdk": "file:skype-sdk.tar.gz"
14  },
15  "engines": {
16    "node": ">=5.6.0"
17  }
18 }
```

Рис. 3 Файл package.json

Дальше потребуется app.js файл, который будет точкой входа для бота.

Бот Skype app.js должен выглядеть как показано ниже.

```
var skype = require('botbuilder');
var express = require('express');
var app = express();
var botService = new skype.ChatConnector({appId: '', appPassword: ''});
var bot = new skype.UniversalBot(botService);
app.post('/api/messages', botService.listen());
bot.dialog('/', function (session) {
  if (session.message.text.toLowerCase().indexOf('hi') >= 0){
    session.send('Hi ' + session.message.user.name +
      ' thank you for your message: ' + session.message.text);
  } else{session.send('Sorry I don't understand you...'); }});
app.get('/', function (req, res) {res.send('SkypeBot listening...');});
app.listen(process.env.port, function () {console.log('SkypeBot listening...');});
```

После того, как все ссылки настроены, можно приступить к созданию botService объекта и подключить его к HTTP POST, размещенной на веб-сайтах Azure, на которую бот-служба Skype будет отправлять входящие сообщения, на которые бот будет отвечать.

Стоит обратить внимание, что botService объект требует APP\_ID и APP\_SECRET переменные, которые можно получить от Bot Framework, при прохождении регистрации на портале разработчиков.

botService Объект создается следующим образом:

```
var APP_ID = '';
var APP_SECRET = '';
var botService = new skype.ChatConnector({
  appId: APP_ID,
  appPassword: APP_SECRET
});
var bot = new skype.UniversalBot(botService);
```

С botService созданным объектом, он должен быть подключен так, что бот по Skype знает, где POST входящие запросы сообщений, так что они могут быть обработаны с помощью робота. Это достигается путем добавления этого к app.js, как показано ниже:

```
app.post('/api/messages', botService.listen())
```

Это в основном регистрирует botService объект в общедоступной /api/messages конечной точке HTTP, доступной через веб-сайт Azure, на котором будет работать этот Node.js. Наконец, приложение Node.js открывается путем прослушивания порта process.env.port следующим образом, добавляя это в app.js:

```
app.listen(process.env.port, function () {console.log('SkypeBot listening...');});
```

Чтобы это сработало, необходимо зарегистрировать бота на портале разработчиков Bot Framework. Для этого нужно войти в свою учетную запись Microsoft на странице <https://dev.botframework.com/>, как на рисунке 4.

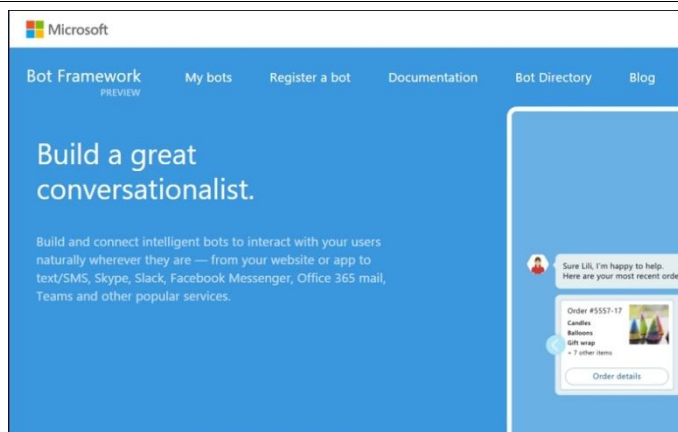


Рис. 4 Главная страница bot framework

Нажать на опцию «Зарегистрировать бота», чтобы создать и зарегистрировать своего скайп-бота, как на рисунке 5.

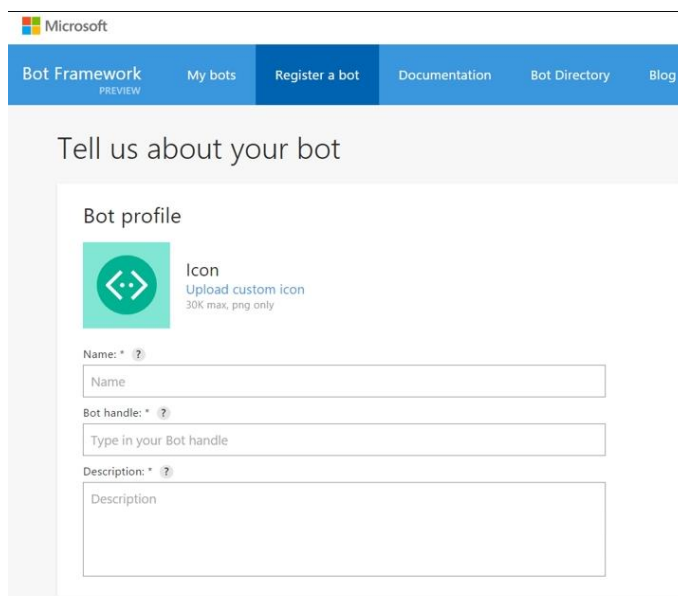


Рис. 5 Форма регистрации бота

Первое поле представляет имя бота, которое будет использоваться для идентификации бота в каталоге ботов. Он не может быть длиннее 35 символов.

Второе поле — это дескриптор бота, который будет использоваться как часть общедоступного URL-адреса бота. Он позволяет использовать только буквенно-цифровые символы и символы подчеркивания и не может быть изменен после регистрации.

Третье поле - описание бота. Первые 46 символов отображаются на карточке бота в каталоге бота, а остальная часть описания отображается под деталями бота.

В случае имени будет использоваться имя NodeJsPacktSkypeBot, однако можно использовать любое другое уникальное имя. Нужно прокрутить страницу вниз, чтобы дополнить информацию о боте.

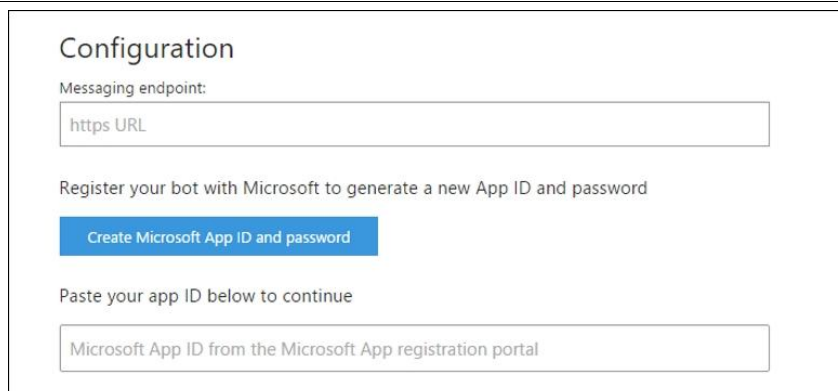


Рис. 6 Форма дополнительной информации о боте

Затем нужно добавить конечную точку обмена сообщениями и добавить идентификатор приложения для бота. Далее нажать кнопку «Создать идентификатор приложения Microsoft и пароль», чтобы получить требуемый идентификатор приложения, как на рисунке 7.

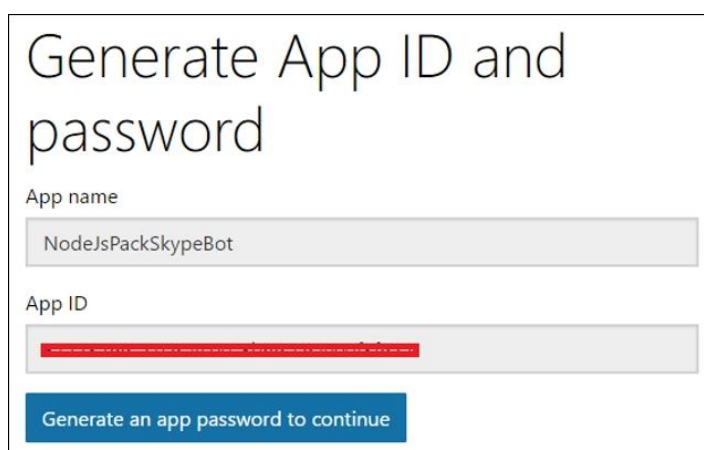


Рис. 7 Получение идентификатора приложения

Далее нужно создать пароль для приложения, для этого нужно нажать на кнопку «Создать пароль приложения», как на картинке 8.

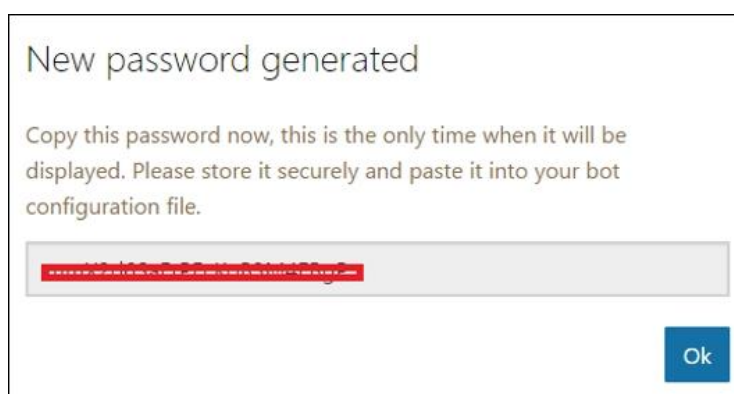
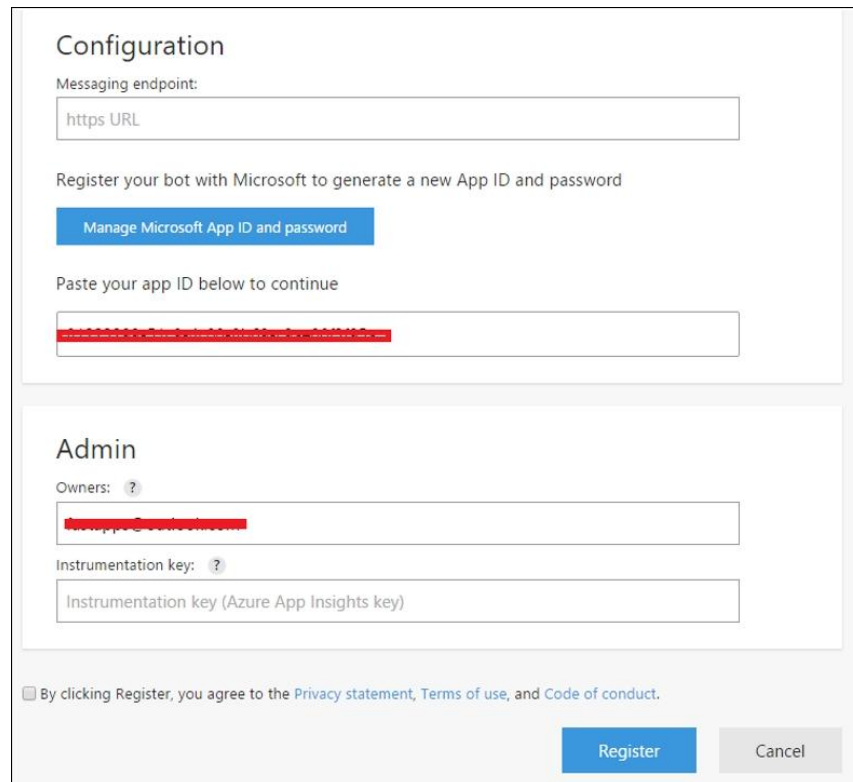


Рис. 8 Сгенерированный пароль



Сразу после этого нажать кнопку ОК, и вернуться к экрану «Сгенерировать идентификатор приложения и пароль». Оказавшись там, нажать кнопку «Готово» и вернуться к кнопке Bot Framework:

Как только это будет сделано, нужно вернуться на главный экран регистрации. Единственной недостающей информацией будет конечная точка обмена сообщениями и поля адреса электронной почты владельцев, как показано на рисунке 9.



The image shows a web form for bot registration, divided into two main sections: Configuration and Admin. The Configuration section includes a 'Messaging endpoint' field with the placeholder 'https URL', a button to 'Manage Microsoft App ID and password', and a 'Paste your app ID below to continue' field. The Admin section includes an 'Owners' field, an 'Instrumentation key' field with the placeholder 'Instrumentation key (Azure App Insights key)', and a checkbox for agreeing to terms. At the bottom right, there are 'Register' and 'Cancel' buttons.

Рис. 9 Заполненная форма регистрации бота

Теперь нужно создать учетную запись Azure и войти в него используя следующую команду.

```
azure login
```

Как только это будет сделано и предоставлены учетные данные, в консоль будет как на рисунке 10.

```
C:\Users\Fast\Documents\Visual_Studio_2015\NodeJs_Bots_Packt\SampleCode\SkypeBot
>azure login
info:    Executing command login
|info:    To sign in, use a web browser to open the page https://aka.ms/deviceclo
gin. Enter the code CZ3NECJ56 to authenticate.
|info:    Added subscription BizSpark
info:    Added subscription Pay-As-You-Go
+
info:    login command OK
C:\Users\Fast\Documents\Visual_Studio_2015\NodeJs_Bots_Packt\SampleCode\SkypeBot
>
```

Рис. 10 Вывод команды azure login

Теперь нужно перейти по URL-адресу, указанному в консоли. Затем введите предоставленный код. Как только это будет сделано, появится баннер как на рисунке 11.

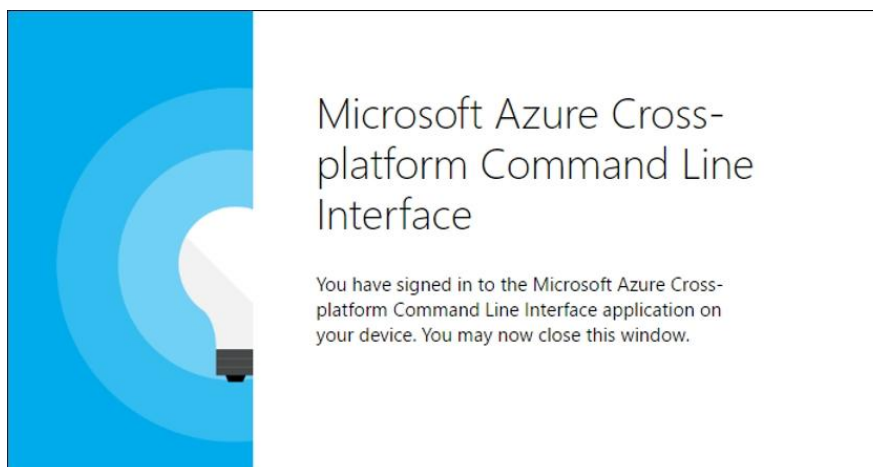


Рис. 11 Авторизация в Azure

Это означает успешную авторизацию в Azure с помощью командной строки.

Следующее, что нужно сделать, это создать службу веб-сайта Azure, на которой будет размещен код бота Skype; это можно сделать, выполнив эту команду из командной строки.

```
azure site create --git nodeskypehrbotsite
```

При выполнении этой команды будет предложено выбрать регион Azure, в котором хотите развернуть бота, как показано на рисунке 12.

```
>azure site create --git nodeskypehrbotsite
info: Executing command site create
+ Getting sites
+ Getting locations
help: Choose a location
1) South Central US
2) North Europe
3) West Europe
4) Southeast Asia
5) East Asia
6) West US
7) East US
8) Japan West
9) Japan East
10) East US 2
11) North Central US
12) Central US
13) Brazil South
14) Australia East
15) Australia Southeast
16) Canada Central
17) Canada East
18) West Central US
19) West US 2
20) UK West
21) UK South
:
```

Рис. 12 Выбор региона для бота

После этого можно увидеть следующую информацию, указывающую на то, что сайт был успешно создан, как на рисунке 13.

```
info: Creating a new web site at nodeskypehrbotsite.azurewebsites.net
info: Created website at nodeskypehrbotsite.azurewebsites.net
+
info: Initializing remote Azure repository
+ Updating site information
info: Remote azure repository initialized
+ Getting site information
+ Getting user information
info: Executing `git remote add azure https://fastapps@nodeskypehrbotsite.scn
azurewebsites.net/nodeskypehrbotsite.git`
info: A new remote, 'azure', has been added to your local git repository
info: Use git locally to make changes to your site, commit, and then use 'git
push azure master' to deploy to Azure
info: site create command OK
```

Рис. 13 Сообщение об успешном создании бота

Перед развертыванием кода бота в Azure сначала нужно войти на портал Azure со своей учетной записью и указать имя пользователя и пароль FTP развертывания, как на рисунке 14.

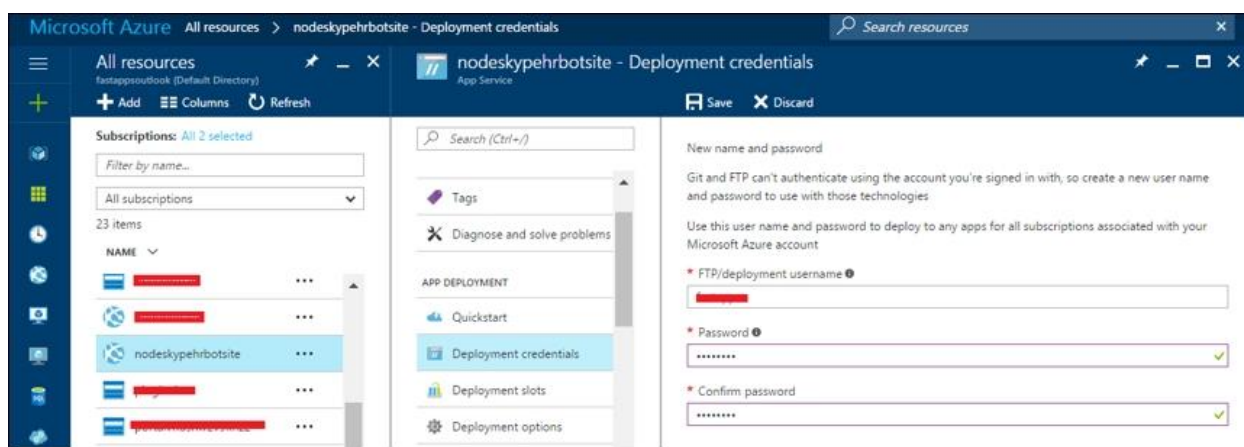


Рис. 14 Создание FTP доступа

Как только это будет сделано, следует подождать пару минут. Затем можно развернуть код на веб-сайтах Azure следующим образом:

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

После выполнения этих команд код бота будет развернут на веб-сайте Azure. Развернув сайт, наконец, можно получить общедоступный URL-адрес, который в случае будет таким <https://nodeskypehrbotsite.azurewebsites.net/api/messages>, учитывая, что в коде определили POST конечную точку, которую будет прослушивать бот.

Это URL-адрес, нужно указать в качестве конечной точки обмена сообщениями.

Затем можно вернуться к экрану веб-сайта регистрации бота, где недавно ввели идентификатор приложения бота, и ввести конечную точку обмена сообщениями, как показано на рисунке 15.

Configuration

Messaging endpoint:

https://nodeskypehrbotsite.azurewebsites.net/api/messages

Register your bot with Microsoft to generate a new App ID and password

Manage Microsoft App ID and password

Paste your app ID below to continue

fastapps@outlook.com

Instrumentation key: ?

Instrumentation key (Azure App Insights key)

By clicking Register, you agree to the [Privacy statement](#), [Terms of use](#), and [Code of conduct](#).

Register Cancel

Рис. 15 Ввод конечной точки приложения

Обязательно отметить опцию «Заявление о конфиденциальности, Условия использования и Кодекс поведения», затем нажать кнопку «Зарегистрироваться», как показано на рисунке 16.

By clicking Register, you agree to the [Privacy statement](#), [Terms of use](#), and [Code of conduct](#).

Register Cancel

Рис. 16 Кнопки регистрации

После должно появиться всплывающее диалоговое окно, в котором будет указано, что бот был создан, как показано на рисунке 17.

Bot created

OK

Рис. 17 Сообщение о создании бота

После нажатия на ОК, откроется окно с информацией как на рисунке 18.

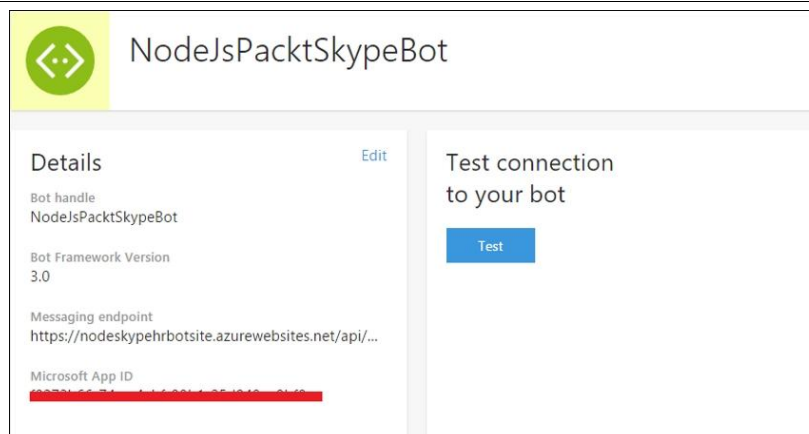


Рис. 18 Подробная информация о боте

Там можно быстро проверить соединение с ботом, нажав на кнопку «Проверить».

Если немного прокрутите вниз, можно найти каналы, которые включены по умолчанию. Один из них - Skype, как показано на рисунке 19.


Channels		Test link	Issues	Enabled	Published	
	Skype		0	Yes	<input type="checkbox"/> Off	Edit
	Web Chat		0	Yes	<input type="checkbox"/> Off	Edit

Рис. 19 Чаты по умолчанию

Нужно нажать кнопку «Добавить в Skype», чтобы добавить бота в список контактов Skype. Потом откроется новая вкладка или окно браузера, и будет представлен следующий экран как на рисунке 20.

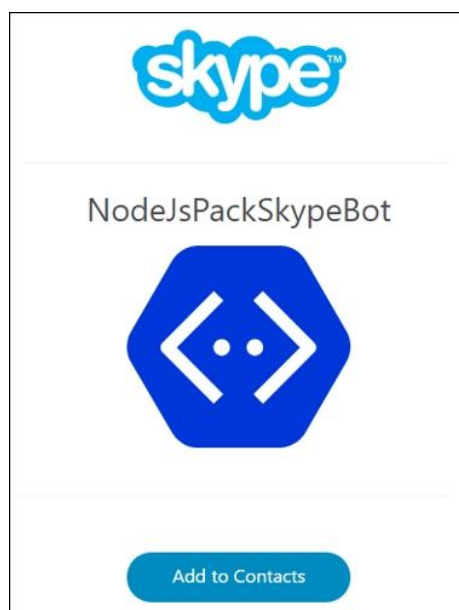


Рис. 20 Окно добавления бота

Чтобы добавить бота в Skype, нужно нажать кнопку «Добавить в контакты». Это запустит приложение Skype и добавит его в список контактов.

Если учетная запись Skype не совпадает с учетной записью Azure, будет предложено выйти из системы, а затем войти в нее с помощью своей учетной записи Skype, чтобы добавить бота в список контактов.

После всех настроек, нужно изменить код, чтобы добавить идентификатор бота, идентификатор приложения и пароль приложения. Идентификатор бота и идентификатор приложения, которые нужно будет добавить в код, совпадают, и они вводятся при регистрации бота.

Skype-бот app.js теперь должен выглядеть так:

```
var skype = require('botbuilder');
var express = require('express');
var app = express()
var botService = new skype.ChatConnector({
  appId: '<< Your Application Id >>',
  appPassword: '<< Your Application Password >>'});
var bot = new skype.UniversalBot(botService);
app.post('/api/messages', botService.listen());
bot.dialog('/', function (session) {
  if (session.message.text.toLowerCase().indexOf('hi') >= 0){
    session.send('Hi ' + session.message.user.name +
      ' thank you for your message: ' + session.message.text);
  } else{session.send('Sorry I don't understand you...'); }});
app.get('/', function (req, res) {res.send('SkypeBot listening...');});
app.listen(process.env.port, function () {console.log('SkypeBot listening...');});
```

Если опубликовать изменения на веб-сайтах Azure и добавить бота в список контактов Skype, используя URL-адрес, обозначенный меткой «Добавить в Skype», можно увидеть следующее, как на рисунке 21.

На данный момент бот ответит тем же сообщением, которое предоставили в качестве ответа, дополненного красивым благодарственным письмом, добавленным к исходному сообщению.

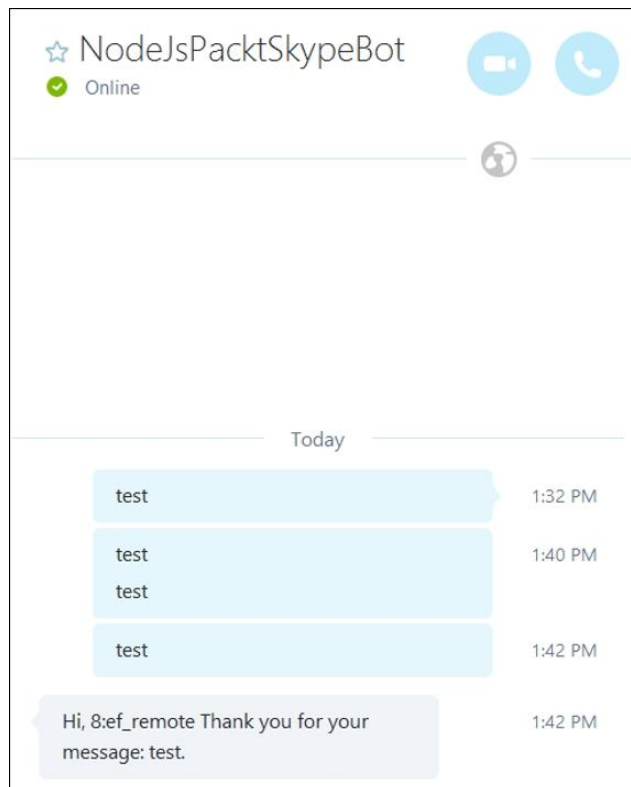


Рис. 21 Ответ бота на сообщения

В коде за ответ отвечает `bot.dialog` событие.

Это событие, запускается, когда Skype отправляет HTTP POST запрос, когда бот получает сообщение.

```
bot.dialog('/', function (session) {
  if (session.message.text.toLowerCase().indexOf('hi') >= 0){
    session.send('Hi ' + session.message.user.name +
      ' thank you for your message: ' + session.message.text);
  } else{session.send('Sorry I don't understand you...');}}
```

`Session` Объект содержит информацию, которая Skype переводит на приложение бота, который описывает данные сессии, которые были получены и от кого. Объект сеанса содержит такие свойства, как `message` и `text`.

До сих пор удалось создать и развернуть в Azure базового бота Skype, который, по сути, отвечает на любое отправленное сообщение с тем же текстом, что и полученный, с добавленным сообщением с благодарностью.

Теперь следует расширить бота Skype, чтобы создать базового агента отдела кадров (HR), способного отвечать на определенные запросы, например, проверять, сколько праздников осталось у человека, или запрашивать отпуск по болезни.

HR — это обширная область, охватывающая множество тем, и, очевидно, в автоматизированный HR-агент можно добавить гораздо больше логики. Однако, поскольку цель состоит в том, чтобы проиллюстрировать своего рода автоматическое общение, ограничимся простой обработкой запросов на отпуск и отпуск по болезни.

Поскольку уже используем Azure, будем использовать хранилище таблиц, чтобы определить некоторые данные и некоторые ответы, которые бот предоставит в зависимости от типа отправленного сообщения и типа запроса, предоставленного пользователем.

Служба хранилища таблиц использует табличный формат для хранения данных. Каждая запись представляет собой сущность, а столбцы представляют различные свойства этой сущности (поля в таблице).

У каждой сущности есть пара ключей (PartitionKey и RowKey) для ее уникальной идентификации. В нем также есть столбец с меткой времени, который служба хранилища таблиц использует, чтобы узнать, когда объект был обновлен в последний раз (это происходит автоматически, и значение метки времени не может быть перезаписано, это внутренне контролируется самой службой).

Чтобы начать работу с экземпляром хранилища в Microsoft Azure, необходимо войти на портал Azure с учетной записью Microsoft. После входа на портал Azure нужно просмотреть список служб Azure и выбрать учетные записи хранения (классические), как показано на рисунке 22.

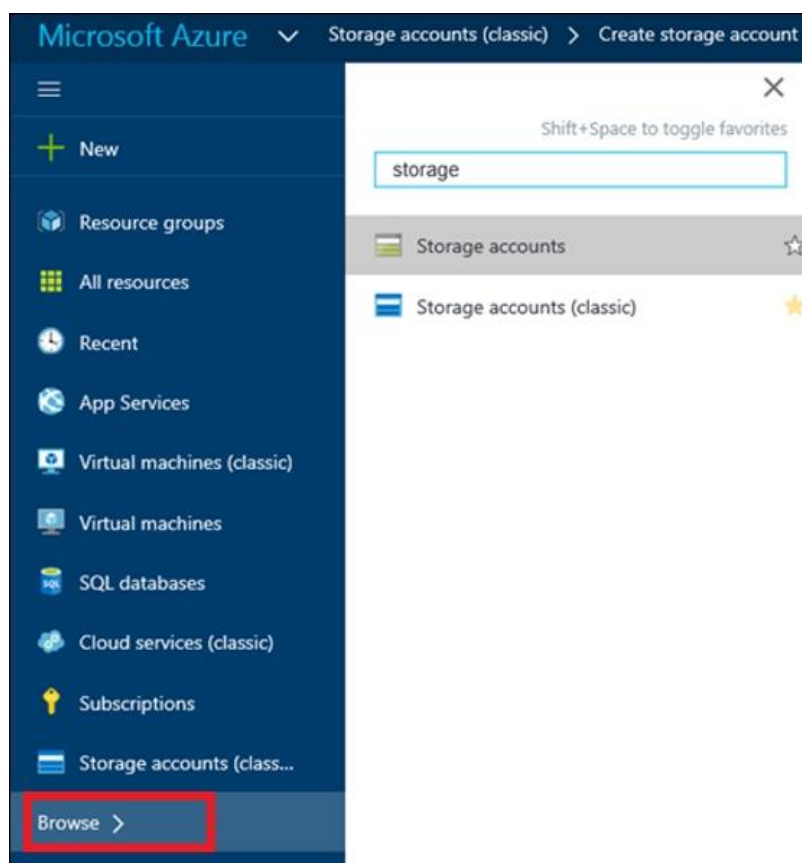


Рис. 22 Создание аккаунта под хранилище

После того, как выбрали Учетные записи хранения (классические), будет представлен следующий экран, на котором нужно добавить новую учетную запись хранения, как показано на рисунке 23.



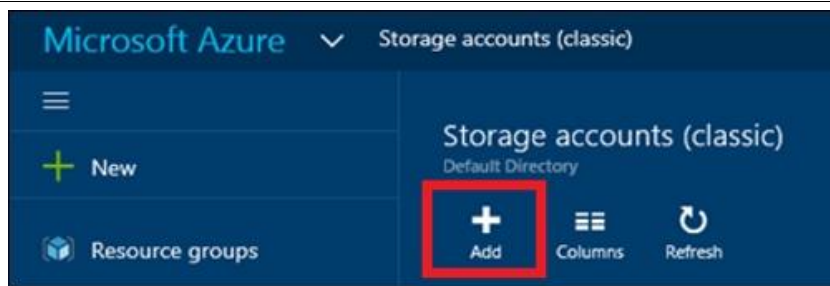


Рис. 23 Добавление хранилища

Чтобы добавить новые учетные записи хранения (классические), нужно нажать кнопку «Добавить».

The image shows the 'Create storage account' form in the Microsoft Azure portal. The form is titled 'Create storage account' and includes a note: 'The cost of your storage account depends on the usage and the options you choose below. Learn more'. The form has several fields and options: 'Name' (with a red square around it), 'Deployment model' (with 'Resource manager' and 'Classic' options), 'Performance' (with 'Standard' and 'Premium' options), 'Replication' (with a dropdown menu set to 'Locally-redundant storage (LRS)'), 'Subscription' (with a dropdown menu set to 'BizSpark'), 'Resource group' (with a dropdown menu set to '+ New'), 'New resource group name' (with an empty text box), 'Location' (with a dropdown menu set to 'East US'), and a 'Pin to dashboard' checkbox. At the bottom, there is a 'Create' button, which is highlighted with a red square.

Рис. 24 Форма создания аккаунта хранилища

После выбора имени нужно нажать кнопку «Создать». Сразу после этого Azure создаст учетную запись хранения. Понадобятся ключи доступа,

чтобы взаимодействовать со службой из кода или любого внешнего инструмента. Ключи можно найти, щелкнув настройку ключей, как показано на рисунке 25.

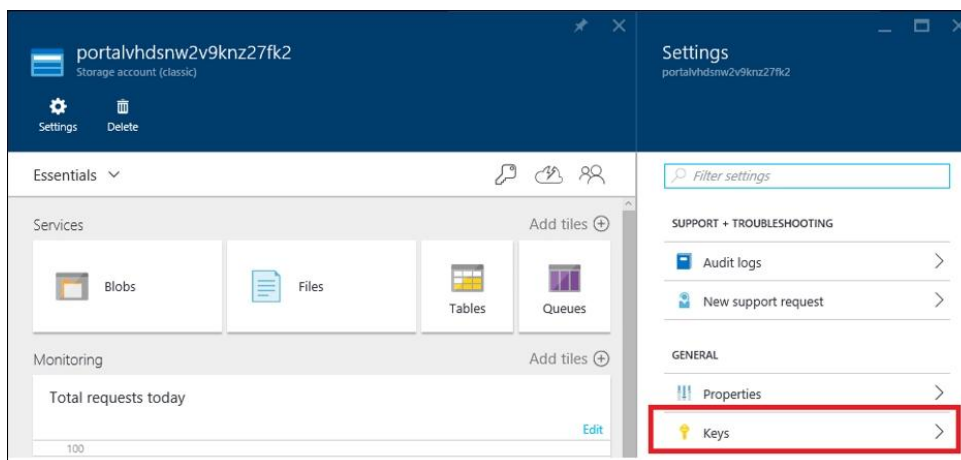


Рис. 25 Расположение вкладки с ключами

Теперь, когда учетная запись хранилища Azure готова, можно использовать очень удобный инструмент с открытым исходным кодом под названием Azure Storage Explorer (<https://azurestorageexplorer.codeplex.com>), который позволит легко подключиться к учетной записи хранилища и создавать, обновлять, удалить и просмотреть любые таблицы хранения и данные.

Мастер установки очень прост в использовании и не требует пояснений, для него требуется всего несколько щелчков мышью. Приложение Azure Storage Explorer работает только в Windows.

Чтобы запустить инструмент, нужно дважды щелкнуть ярлык Обзорателя хранилищ Azure. После запуска приложения необходимо подключить к нему свою учетную запись хранилища Azure. Это можно сделать, нажав кнопку «Добавить учетную запись», как на рисунке 26.

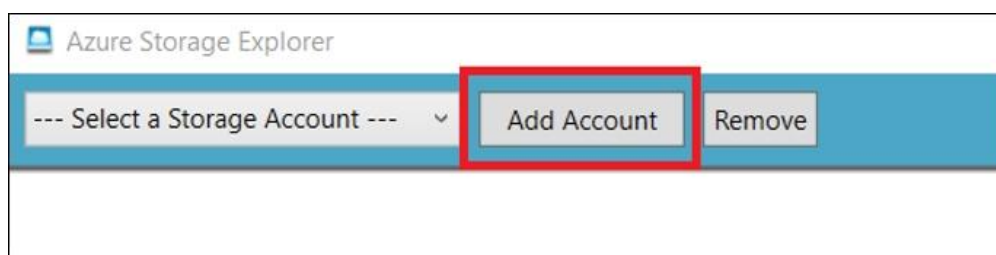


Рис. 26 Добавление аккаунта в Azure Storage Explorer

После нажатия на кнопку «Добавить учетную запись», будет предложено добавить имя и ключ учетной записи хранилища как на рисунке 27.

Рис. 27 Форма для добавления имени и ключей учетной записи

После ввода запрошенных данных нужно нажать кнопку «Проверить доступ», чтобы проверить, работает ли соединение. В случае успеха нажать кнопку «Сохранить».

Настроив учетную запись хранилища таблиц Azure и увидев, как использовать обозреватель хранилища для подключения к ней, следует создать таблицу с некоторыми данными, которые бот Skype будет использовать для интерпретации запросов и работы в качестве автоматического агента по кадрам.

Нужно создать таблицу с именем HolidaysHRBot, которая будет содержать как PartitionKey имя пользователя Skype, так и RowKey полное имя человека в форме FirstName-LastName. Также будет третье поле с именем DaysLeft, которое будет целым числом и будет представлять количество дней отпуска, которые человек может использовать.

Предположим, что оно DaysLeft начнется со значения 25. Сейчас нужно добавить еще одно поле, в котором будет указано количество использованных дней по болезни. Называться оно будет DaysSick и определено как целое число, которое изначально будет установлено в ноль.

Из всех полей нужно собрать таблицу в Azure Storage Explorer как на рисунке 28.

PartitionKey	RowKey	DaysLeft	DaysSick
Eduardo Freitas	Ed-Freitas	24	1
John Robinson	John-Robinson	25	0
John Smith	John-Smith	25	0

Рис. 28 Таблица в Azure Storage Explorer

Логика работы HR-агента заключается в том, что после того, как пользователь был проверен и был размещен запрос на отпуск или болезни, бот ответит с выбранной опцией.

Итак, определив эти основные правила, нужно разработать BotBrain метод, который может с этим позаботиться.

Чтобы начать работу, первое, что нужно сделать, — это добавить хранилище Azure в проект в виде пакета npm. Это можно сделать из командной строки следующим образом:

```
npm install azure-storage --save
```

Это обновит package.json файл, добавив ссылку на библиотеку хранилища Azure. После установки пакета добавим ссылку на него в код:

```
var azure = require('azure-storage');
```

Добавив ссылку, нужно создать tableSvc объект, который будем использовать для подключения и связи с HolidaysHRBot таблицей. Необходимо пройти AZURE\_ACCOUNT и ACCOUNT\_KEY, которые можно найти на портале Azure.

```
var tableSvc = azure.createTableService(AZURE_ACCOUNT, AZURE_KEY);
```

Поскольку данные в таблицу уже добавлены с помощью Storage Explorer, для получения данных на основе PartitionKey и RowKey нужно использовать retrieveEntity метод из tableSvc объекта. Вот как:

```
tableSvc.retrieveEntity('HolidaysHRBot', 'Eduardo Freitas', 'Ed-Freitas',  
function(error, result, response){if(!error){ }});
```

Теперь нужно создать небольшой метод проверки пользователя, который принимает первое сообщение от пользователя и проверяет в таблице Azure, существует ли пользователь на самом деле, и, если да, бот выполняет остальную часть процесса интерактивного обмена сообщениями. Это будет отправной точкой для BotBrain метода.

```
userVerification = function(session) {  
  session.send('Hey, let me verify your user id ' +  
    userId + ' (' + userName + '), bear with me...');  
  tableSvc.retrieveEntity(AZURE_TABLE, userId, userName, function  
    entityQueried(error, entity) {  
    if (!error) {  
      authenticated = true;  
      userEntity = entity;  
      session.send('I have verified your id, how can I help you?' +  
        ' Type a) for Holidays, b) for Sick Leave.');    }  
    else {  
      session.send('Could not find: ' + userName + ', please make sure you use  
        proper casing :));});});
```

В основном обернули retrieveEntity функцию в метод, в зависимости от того, какой результат был получен из AZURE\_TABLE, данное значение session.send отправляется обратно пользователю.

Если для пользователя существует соответствующая запись для указанных `userId` (`PartitionKey`) и `userName` (`RowKey`), то устанавливается состояние аутентификации, и сущность `retrieved`(запись) копируется в `userEntity` объект.

Теперь осталось связать все это, обрисовав полный код бота следующим образом:

```
var skype = require('botbuilder');
var express = require('express');
var azure = require('azure-storage');
var app = express();
var APP_ID = '<< Your App ID >>';
var APP_SECRET = '<< Your App Password >>';
var AZURE_ACCOUNT = '<< Your Azure Account ID >>';
var AZURE_KEY = '<< Your Azure Account Key >>';
var AZURE_TABLE = 'HolidaysHRBot';
var tableSvc = azure.createTableService(AZURE_ACCOUNT, AZURE_KEY);
var authenticated = false;
var holidays = false;
var sick = false;
var userId = '';
var userName = '';
var userEntity = undefined;
var botService = new skype.ChatConnector({
  appId: APP_ID, appPassword: APP_SECRET});
var bot = new skype.UniversalBot(botService);
app.post('/api/messages', botService.listen());
userVerification = function(session) {
  session.send('Hey, let me verify your user id ' + userId + '(' +
    userName + '), bear with me...');
  tableSvc.retrieveEntity(AZURE_TABLE, userId, userName,
    function entityQueried(error, entity) {
      if (!error) {
        authenticated = true; userEntity = entity;
        session.send('I have verified your id, how can I help you?' +
          ' Type a) for Holidays, b) for Sick Leave.');
        else {session.send('Could not find: ' + userName +
          ', please make sure you use proper casing :)}); }));};
cleanUserId = function(userId) {
  var posi = userId.indexOf(':');
  return (posi > 0) ? userId.substring(posi + 1) : userId;};
BotBrain = function(session) {
  var orig = session.message.text;
  var content = orig.toLowerCase();
  var from = session.message.user.name;
  if (authenticated) {
    if (content === 'a') {
      holidays = true;
      session.send('Please indicate how many vacation days' +
        ' you will be requesting, i.e.: 3');
    else if (content === 'b') {
      sick = true;
      session.send('Please indicate how many sick days' +
        ' you will be requesting, i.e.: 2'); }
    else if (content !== 'a' && content !== 'b') {
      if (holidays) {
        session.send(userName + '(' + userId + ')' +
          ', you have chosen to take ' + content + ' holiday(s). Session ended.');
        sick = false;
```

```

    authenticated = false; }
    else if (sick) {
        session.send(userName + '(' + userId + ')' +
            ', you have chosen to take ' + content +
            ' sick day(s). Session ended.');
```

```

        holidays = false; authenticated = false;}
    else if (!holidays && !sick) {
        session.send('I can only process vacation or sick leave requests.' +
            ' Please try again.')}}}
else {
    authenticated = false, holidays = false, sick = false;
    userId = '', userName = '', userEntity = undefined;
    if (content === 'hi') {
        session.send('Hello ' + cleanUserId(from) +
            ', I shall verify your identify...');
```

```

        session.send('Can you please provide your FirstName-LastName?' +
            ' (please use the - between them)');
```

```

    else if (content !== '') {
        userId = cleanUserId(from);
        userName = orig;
        if (userName.indexOf('-') > 1) {userVerification(session);}
        else {session.send('Hi, please provide your FirstName-LastName' +
            ' (please use the - between them) or say hi :)')}}}];
bot.dialog('/', function (session) {BotBrain(session)});
app.get('/', function (req, res) {res.send('SkypeBot listening...')});
app.listen(process.env.port, function () {console.log('SkypeBot listening...')});

```

Первое, что следует заметить, это то, что при bot.Dialog срабатывании события вызывается BotBrain функция:

```
bot.dialog('/', function (session) {BotBrain(session)});
```

Еще одна важная часть заключается в том, что нужно каким-то образом сохранять состояние, чтобы иметь возможность определять, на каком этапе разговора находится бот с пользователем.

Относительно простой способ сделать это - использовать переменные, которые сохраняют состояние разговора или применяют его к некоторым частям. Нужно будет знать, когда пользователь прошел аутентификацию, что в основном означает, что его идентификатор Skype и имя были проверены на соответствие данным, содержащимся в AZURE\_TABLE. В дополнение к этому, также должны держать userId, userName и userEntity для идентифицированного пользователя.

Также важно знать, отправил ли пользователь holidays запрос или запрос на sick отпуск. С помощью этих переменных можем очень просто сохранить состояние.

В идеале для нескольких пользователей, одновременно запрашивающих взаимодействие с ботом, состояние должно сохраняться индивидуально для каждого пользователя, вошедшего в систему или прошедшего аутентификацию. Как на следующем фрагменте кода:

```

var authenticated = false;
var holidays = false;
var sick = false;
var userId = '';
var userName = '';

```

```
var userEntity = undefined;
```

Рассмотрев проблему управления состоянием, следует сосредоточиться на внутреннем устройстве BotBrain функции. Она будет состоять из двух частей.

Первая - аутентифицирован ли пользователь уже (подтверждено существование пользователя на AZURE\_TABLE), а вторая - то, где пользователь еще не прошел аутентификацию, что соответствует начальному этапу разговора:

```
authenticated = false, holidays = false, sick = false;
userId = '', userName = '', userEntity = undefined;
if (content === 'hi') {
    session.send('Hello ' + cleanUserId(from) +
        ', I shall verify your identify...');
    session.send('Can you please your provide your FirstName-LastName?' +
        ' (please use the - between them)');}
else if (content !== '') {
    userId = cleanUserId(from); userName = orig;
    if (userName.indexOf('-') > 1) {
        userVerification(session);}
    else {
        session.send('Hi, please provide your FirstName-LastName' +
            ' (please use the - between them) or say hi :)');}}
```

Чтобы начать разговор, пользователь должен написать сообщение, содержащее слово hi. После этого бот отвечает и просит пользователя ввести свое имя в форме FirstName-LastName(следует использовать правильный регистр).

FirstName-LastName будет использоваться для запроса RowKey AZURE\_TABLE и проверки соответствия userId (Skype Id пользователя) записи, которая также содержит значение, указанное в FirstName-LastName. Это делается внутри userVerification функции.

После подтверждения личности пользователя authenticated устанавливается значение, true и поэтому бот может спросить, какое действие хочет выполнить пользователь.

```
if (content === 'a') {
    holidays = true;
    session.send('Please indicate how many vacation days' +
        ' you will be requesting, i.e.: 3');}
else if (content === 'b') {
    sick = true;
    session.send('Please indicate how many sick days' +
        ' you will be requesting, i.e.: 2');}
else if (content !== 'a' && content !== 'b') {
    if (holidays) {
        session.send(userName + '(' + userId + ')' +
            ', you have chosen to take ' + content +
            ' holiday(s). Session ended.');
```

```
holidays = false; authenticated = false;}  
else if (!holidays && !sick) {  
    session.send('I can only process vacation or sick leave requests.' +  
        ' Please try again.');
```

После того, как пользователь был проверен и состояние бота было аутентифицировано, бот запрашивает у пользователя выбор, хочет ли он запросить несколько дней отпуска или дней отпуска по болезни. Как только пользователь отвечает, каждое состояние затем сохраняется с использованием логических переменных, называемых `holidays` и `sick`, которые затем используются ботом для отправки ответа с вопросом, сколько дней пользователь хочет забронировать. Пример можно наблюдать на рисунке 29.

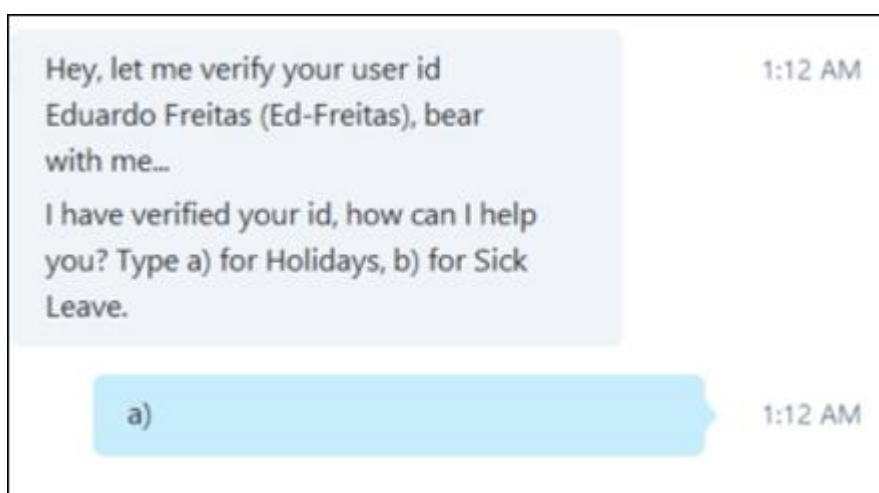


Рис. 29 Выбор варианту у бота

Когда пользователь указывает количество дней, бот отвечает, подтверждая запрос. Результат можно увидеть на рисунке 30.

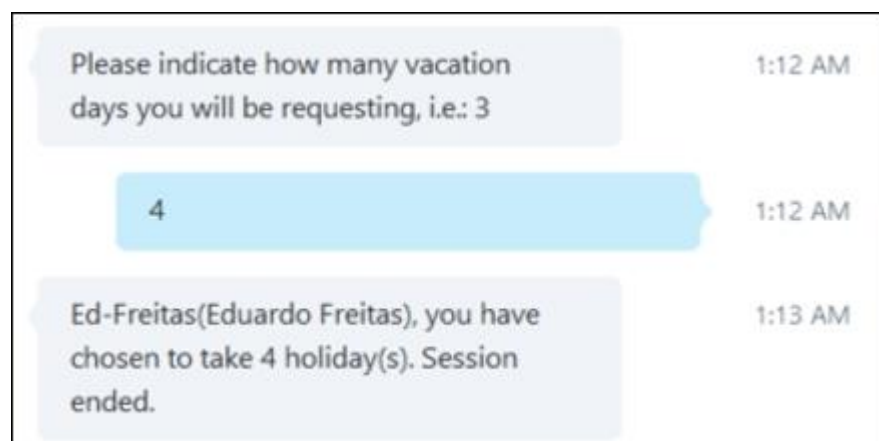


Рис. 30 Выбор количества дней отпуска

Важно отметить, что есть место для добавления дополнительной логики и выполнения дополнительных операций, таких как фактическое изменение значений `AZURE_TABLE` после ввода запроса дней, поэтому есть



много возможностей для продолжения изучения и расширения функциональности бота.

#### Вывод

В этой статье рассказано о том, как подключаться и взаимодействовать со службами Skype, а также о том, как создать бота для использования некоторых базовых, но интересных и интерактивных функций. Можно рассмотреть, как настроить бота со Skype, как установить соответствующие пакеты npm и реализовать базовый скелет и структуру для приложения.

В дополнение к этому также узнали, как создать мозг бота, чтобы выполнять определенные задачи и отправлять правильный ответ на основе полученных данных.

#### Библиографический список

1. Потапова Р.К., Собакин А.Н., Маслов А.В. Возможность идентификации говорящего по голосу в системе интернет-телефонии skype // Вестник Московского государственного лингвистического университета. 2013. № 13 (673). С. 177-188. URL: <https://www.elibrary.ru/item.asp?id=19316021> (Дата обращения: 29.01.2021)
2. Прохорова С.Н. Использование скайп-технологий в дистанционном обучении // Вестник Воронежского государственного университета. Серия: Проблемы высшего образования. 2014. № 3. С. 83-86. URL: <https://www.elibrary.ru/item.asp?id=21847796> (Дата обращения: 29.01.2021)
3. Григорьева С.В., Игошкина Н.Г., Васильева О.Г. Организация мониторинга сервиса skype for business для оптимизации работы отдела поддержки сервисов // Вопросы региональной экономики. 2018. № 2 (35). С. 25-31. URL: <https://www.elibrary.ru/item.asp?id=35171731> (Дата обращения: 29.01.2021)