УДК 004.624

Создание IRC бота отслеживания ошибок

Кизянов Антон Олегович
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В данной статье описан процесс создания IRC-бота для отслеживания ошибок. Для создания используется DocumentDB как место хранения информации об ошибках и NodeJS для описания логики работы IRC-бота. Созданный бот позволяет оставлять пользователям информацию о найденных багах в канале IRC, тем самым сообщая разработчикам о существующих ошибках.
Ключевые слова: IRC, NodeJS, бот

Building an IRC bug tracking bot

Kizyanov Anton Olegovich
Sholom-Aleichem Priamursky State University
Student

Abstract
This article will walk you through the process of creating an IRC bug tracking bot. For creation, DocumentDB is used as a place to store information about errors and NodeJS to describe the logic of the IRC bot. The created bot allows users to leave information about the bugs found in the IRC channel, thereby informing developers about existing bugs.
Keywords: IRC, NodeJS, bot

InternetRelayChat (IRC) позволяет общаться в режиме реального времени в виде текста. Этот чат работает по протоколу TCP в модели клиент-сервер. IRC поддерживает групповые сообщения, которые называются каналами, а также поддерживает личные сообщения.

IRC организован во множество сетей с разными аудиториями. IRC является клиентским сервером, поэтому пользователям нужны клиенты IRC для подключения к серверам IRC. Клиентское программное обеспечение IRC поставляется как в виде пакетного программного обеспечения, так и в виде веб-клиентов. Некоторые браузеры также предоставляют IRC-клиенты в качестве надстроек. Пользователи могут установить их в своих системах, а затем использовать их для подключения к серверам IRC. При подключении к этим IRC-серверам пользователи должны указать уникальный псевдоним и выбрать существующий канал для связи, или пользователи могут начать новый канал при подключении к этим серверам.

В данной статье будет разработан IRC-бот для отслеживания ошибок. Этот бот будет предоставлять информацию об ошибках, а также подробную информацию о конкретной ошибке. Все это будет происходить без проблем в самом IRC-канале.

Цель исследования – создать IRC-бота для отслеживания ошибок для команд разработчиков.

Ранее этим вопросом интересовались Л.А. Сазанова, М.А. Зенков развивали тему «Чат-боты как современный инструмент бизнес-анализа» [1] в которой обсуждается использование чат-ботов для решения бизнес-задач. Рассмотрены примеры применения данного инструмента анализа в маркетинге и кадровых службах. Показаны преимущества чат-ботов перед мобильными приложениями и другими электронными средствами коммуникации. А. Слободской с темой «IRC-сервер» [2], а подробнее про администрирование небольшой локальной сети. Так или иначе пользователям хочется общаться. Такие программы, как lantalk, естественно, не подходят, потому что рассылают широковещательные пакеты по всей сети, что не есть хорошо. Выход – установка irc-сервера, который лишен очень многих недостатков и имеет массу преимуществ. Например, быстрота работы по сравнению с www-чатами – ведь клиенту не приходится грузить громоздкую страницу, чаще всего наполненную всяческими баннерами; расходование меньшего количества трафика, что не может не радовать. Существует несколько разновидностей irc-серверов, имеющих некие свои особенности, но похожие друг на друга в целом, например, unreal ircd, bahamut ircd и так далее. М.В. Жигулин, А.В. Коломеец, Н.Г. Кушик, А.В. Шабалдин опубликовали статью «Тестирование программной реализации протокола irc на основе модели расширенного автомата» [3] описали тестирование программной реализации ngIRCd телекоммуникационного протокола IRC на основе теста, построенного по модели расширенного автомата. Компьютерные эксперименты иллюстрируют эффективность предложенного подхода.

Для инициирования IRC-коммуникации нужен IRC-клиент и сервер или сеть, к которой будет подключен клиент. Будет использоваться сеть freenode для подключения клиента. Freenode - крупнейшая бесплатная IRC-сеть с открытым исходным кодом, ориентированная на программное обеспечение.

В качестве веб-клиента IRC через URL-адрес (https://webchat.freenode.net/). После открытия URL-адреса должна появится форма как на рисунке 1.

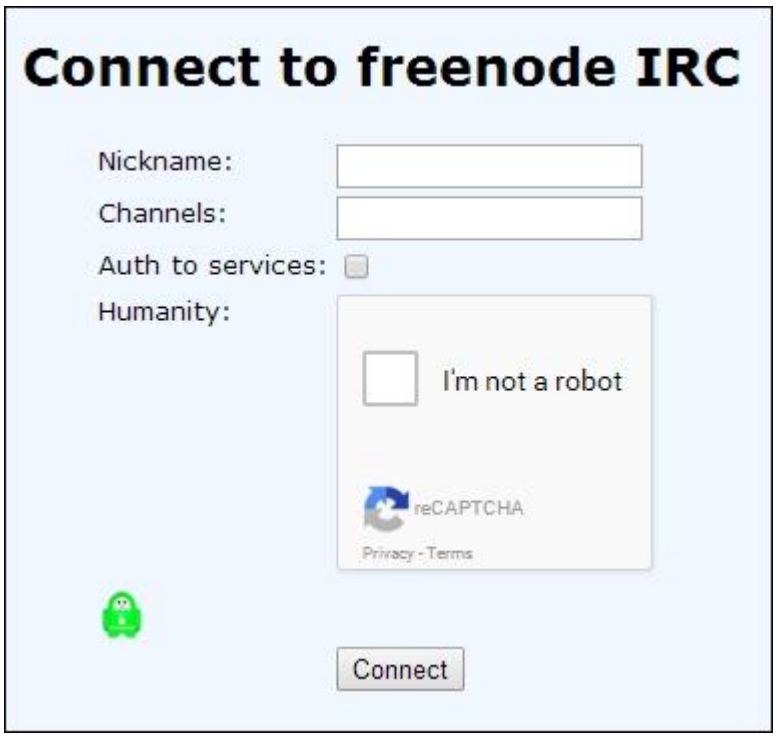

Рис. 1 Форма входа на IRC сервер

В качестве логина указали «Madan», в качестве канали был указан «#BugsChannel». В IRC каналы всегда обозначаются знаком #. Все разработчики или члены команды могут аналогичным образом указать свои псевдонимы и название канала, чтобы присоединиться к общению.

После подключения должна появится страница как на рисунке 2.

Рис. 2 Главная страница канала

Таким образом, IRC-клиент подключается к сети freenode. Также можно наблюдать за всеми пользователя в канале в колонке справа, сейчас только 1 пользователь «Medan».

IRC-бот — это программа, которая подключается к IRC как один из клиентов и отображается как один из пользователей в каналах IRC. Эти IRC-боты используются для предоставления услуг IRC или для размещения пользовательских реализаций на основе чата, которые помогут командам эффективно сотрудничать.

Начать следует с создания папки на локальном диске для хранения программы-бота из командной строки:

```
mkdir ircbot
cd ircbot
```

В системе должен быть установлен NodeJS и npm для работы бота. Дальше нужно создать файл package.json, используя команду ниже.

```
npm init
```

После выполнения команды npm init в консоли будет сообщение примерно, как на рисунке 3.

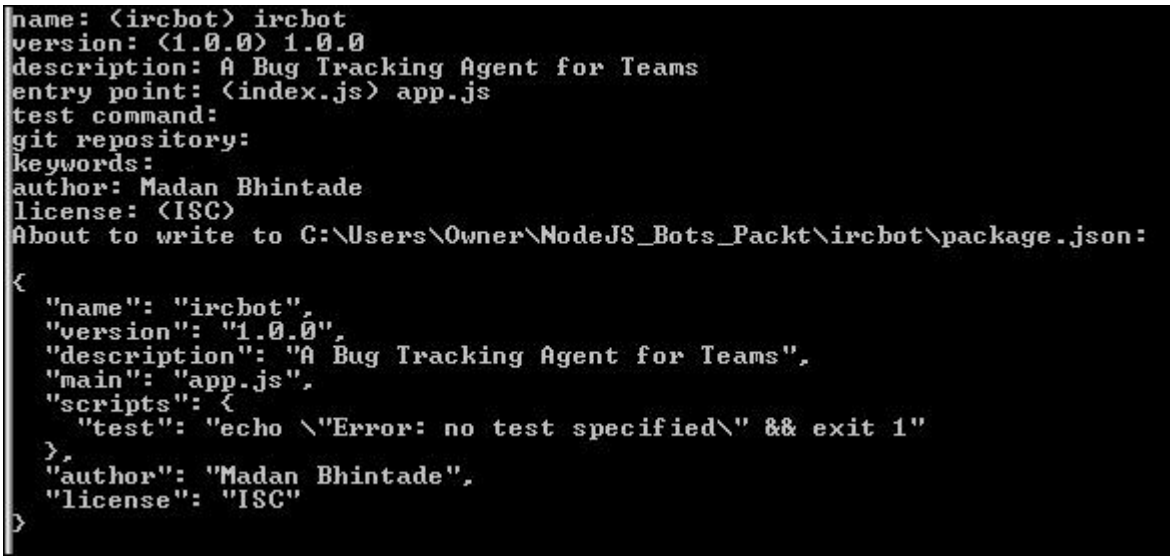

Рис. 3 Вывод команды

Нужно установить пакет irc из npm. Его можно найти по адресу https://www.npmjs.com/package/irc.

Чтобы установить его, нужно выполнить команду ниже.

```
npm install --save irc
```

После успешной установки необходимо обновить package.json, чтобы включить "engines" атрибут. Нужно открыть package.json файл в текстовом редакторе и обновите его следующим образом:

```
"engines": {
    "node": ">=5.6.0"
```

```
}
```

Полностью файл package.json должен выглядеть как на рисунке 4.

```
{
  "name": "ircbot",
  "version": "1.0.0",
  "description": "A Bug Tracking Agent for Teams",
  "main": "app.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Madan Bhintade",
  "license": "ISC",
  "dependencies": {
    "irc": "^0.5.0"
  },
  "engines": {
    "node": ">=5.6.0"
}
}
```

Рис. 4 Полное содержимое файла package.json

Дальше нужен файл app.js, который будет точкой входа для бота. Этот файл должен содержать такой код.

```
var irc = require('irc');
var client = new irc.Client('irc.freenode.net', 'BugTrackerIRCBot', {
  autoConnect: false});
client.connect(5, function(serverReply) {
  console.log("Connected!\n", serverReply);
  client.join('#BugsChannel', function(input) {console.log("Joined #BugsChannel");
    client.say('#BugsChannel', "Hi, there. I am an IRC Bot which track bugs or
defects for your team.\n I can help you using following commands.\n BUGREPORT \n
BUG # <BUG. NO>");});});});
```

Теперь нужно запустить программу Node.js и посмотреть, как выглядит консоль. Если все работает хорошо, консоль должна показать, что бот подключен к требуемой сети, а также подключен к каналу. Консоль выглядит как на рисунке 5.

```
C:\Users\Owner\NodeJS_Bots_Packt\ircbot>node app.js
Connected!
 { prefix: 'verne.freenode.net',
  server: 'verne.freenode.net',
  command: 'rpl_welcome',
  rawCommand: '001',
  commandType: 'reply',
  args:
   [ 'BugTrackerIRCBot',
     'Welcome to the freenode Internet Relay Chat Network BugTrackerIRCBot' ] }
Joined #BugsChannel
```

Рис. 5 Вывод первого запуска бота

Теперь, если посмотрите канал #BugsChannel в веб-клиенте, можно заметить, что бот присоединился к нему, а также отправил приветственное сообщение как на рисунке 6.

Рис. 6 Первое сообщение от бота

Для работы бота использовалась библиотека irc как показано ниже.

Мы использовали irc библиотеку со следующей строчкой:

```
var irc = require('irc');
```

Используя irc библиотеку, был создан экземпляр клиента для подключения к одной из сетей IRC, используя следующий фрагмент кода:

```
var client = new irc.Client('irc.freenode.net', 'BugTrackerIRCBot', {
  autoConnect: false});
```

Здесь бот подключился к сети irc.freenode.net и предоставил никнейм BugTrackerIRCBot. Это имя было дано, чтобы бот отслеживал и сообщал обо всех обнаруженных ошибках в будущем. Теперь бот подключается к определенному каналу, используя следующий фрагмент кода:

```
client.connect(5, function(serverReply) {
  console.log("Connected!\n", serverReply);
  client.join('#BugsChannel', function(input) {
    console.log("Joined #BugsChannel");
    client.say('#BugsChannel', "Hi, there. I am an IRC Bot which track bugs or
defects for your team.\n I can help you using following commands.\n BUGREPORT \n
BUG # <BUG. NO>");});});});
```

В предыдущем фрагменте кода после подключения клиента был получен ответ от сервера. Этот ответ отображается в консоли. После успешного подключения бот присоединяется к каналу, используя следующую строку кода:

```
client.join('#BugsChannel', function(input) {
```

Теперь, используя client.join(), бот присоединяется к тому же каналу. После того, как бот присоединился, бот передает приветственное сообщение в том же канале, используя функцию client.say().

Разработчики всегда хотели бы знать, как работают программы или система. Для этого группы тестирования обычно проводят тестирование системы или программы и регистрируют ошибки или дефекты в программном пакете или системе для отслеживания ошибок. Позже разработчики могут изучить эти ошибки и устранить их в рамках жизненного цикла разработки. Во время этого путешествия разработчики будут сотрудничать и общаться через платформы обмена сообщениями, такие как IRC. В этом канале будут сообщаться об ошибках и сообщаться по запросам разработчиков. Кроме того, если разработчикам нужна дополнительная информация об ошибке, чат-бот может помочь им, предоставив URL-адрес из системы отслеживания ошибок.

Для долгосрочного хранения информации об ошибках нужно место, в качество него будет использоваться DocumentDB.

DocumentDB — это NoSQL база данных, в которой данные хранятся в документах JSON, и работает на платформе Microsoft Azure. Нужно завести учетную запись в Azure, чтобы запустить на нем DocumentDB. Для бота будет учетная запись с именем botdb. Нужно выбрать NoSQL API в качестве DocumentDB. Выбрать подходящую подписку и ресурсы. После ввода всей необходимой информации, нужно нажать на кнопку «Создать» внизу, чтобы создать новую учетную запись для DocumentDB, как на рисунке 7

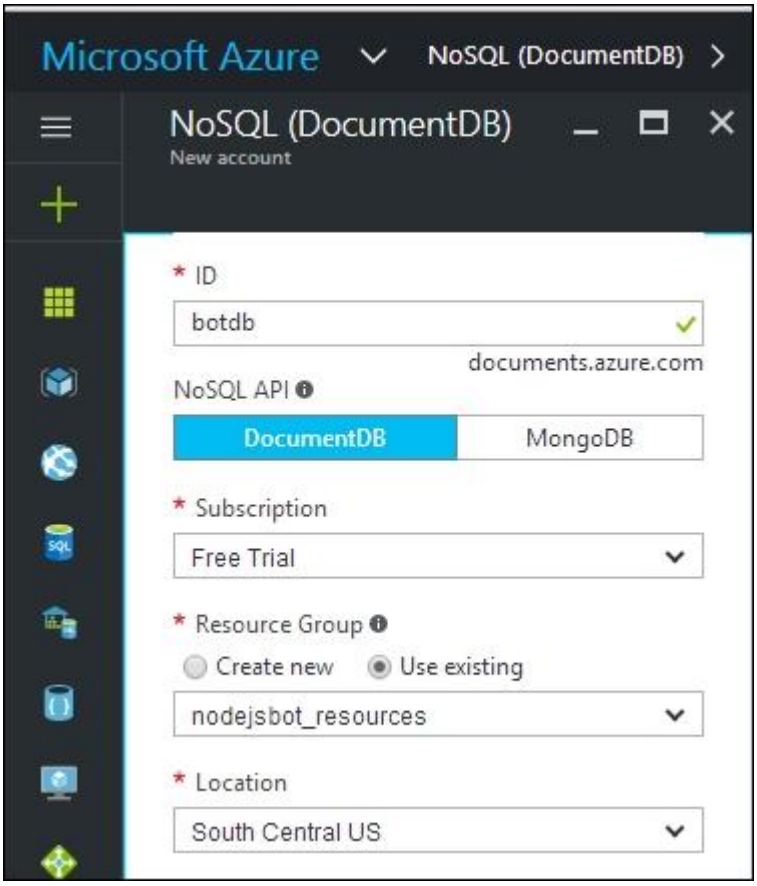

Рис. 7 Форма создания DocumentDB

В текущей учетной записи нужно создать коллекцию для хранения данных об ошибках. Чтобы создать новую коллекцию, нужно нажать на опцию «Add Collection», как показано на рисунке 8.

Рис. 8 Кнопка создания новой коллекции

При нажатии на опцию «Add Collection» в правой части экрана отобразится следующий экран, как на рисунке 9.

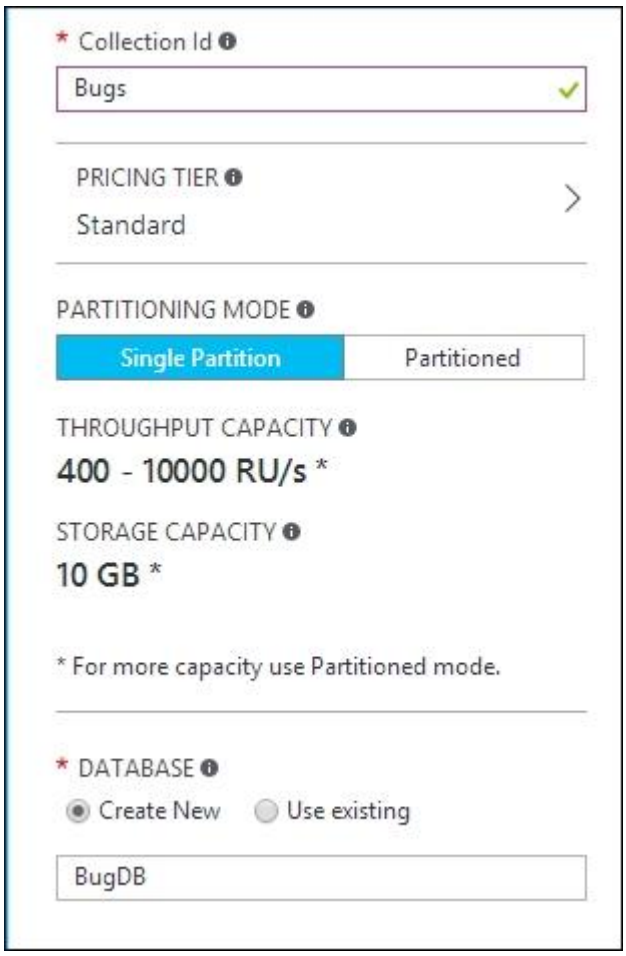

Рис. 9 Форма настройки DocumentDB

Эта новая база данных будет названа BugDB. Как только эта база данных будет создана, в будущем можно добавлять в нее другие коллекции, связанные с ошибками. После того, как все данные введены нужно нажать «ОК», чтобы создать базу данных, как на рисунке 10.

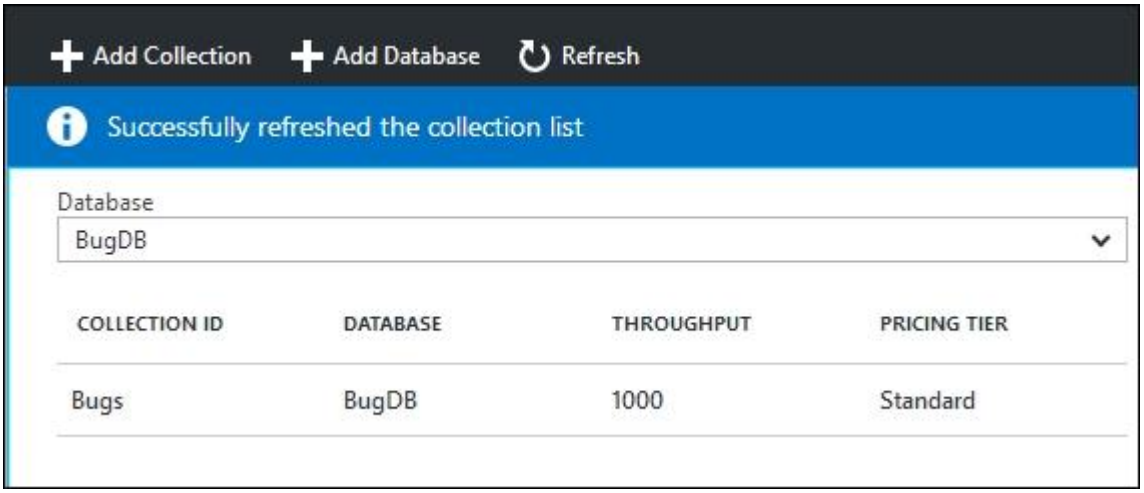

Рис. 10 Список баз данных

Теперь есть BugsDB коллекция ошибок, в которой будут храниться все данные об ошибках. Чтобы добавить элемент данных, нужно воспользоваться опцией меню Document Explorer, показанной на рисунке 11.

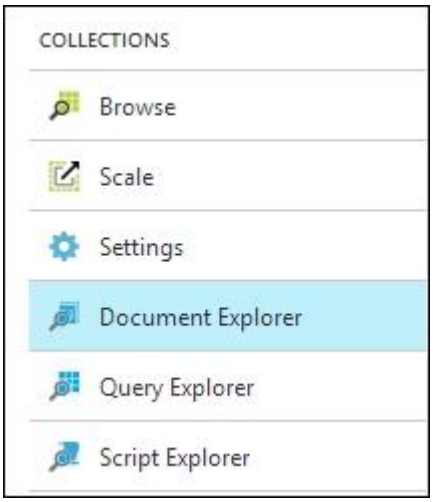

Рис. 11 Выбор коллекции

Откроется экран со списком созданных баз данных и коллекций. Чтобы создать документ JSON для коллекции ошибок, нужно нажать кнопку «Создать». Откроется форма для ввода данных в формате JSON. Нужно ввести туда данные как на рисунке 12.

```
{
    "id": "123",
    "status" : "Open",
    "title" : "Sign In Failing for Google Accounts",
    "description" : "Sign In is failing for Google Accounts.
error could not log in.",
    "priority" : "High",
    "assignedto" : "Madan Bhintade",
    "url" : "http://mybugsystem.net/bugs/123"
}
```

Рис. 12 Данные для заполнения формы

В коллекции будет хранится поля id, status, title, description, priority, assignedto, и url атрибуты, которые будут храниться в коллекции ошибок.

Таким образом, можно создать образцы записей в коллекции ошибок, которые позже будут включены в программу Node.js. Примерный список ошибок можно увидеть на рисунке 13.

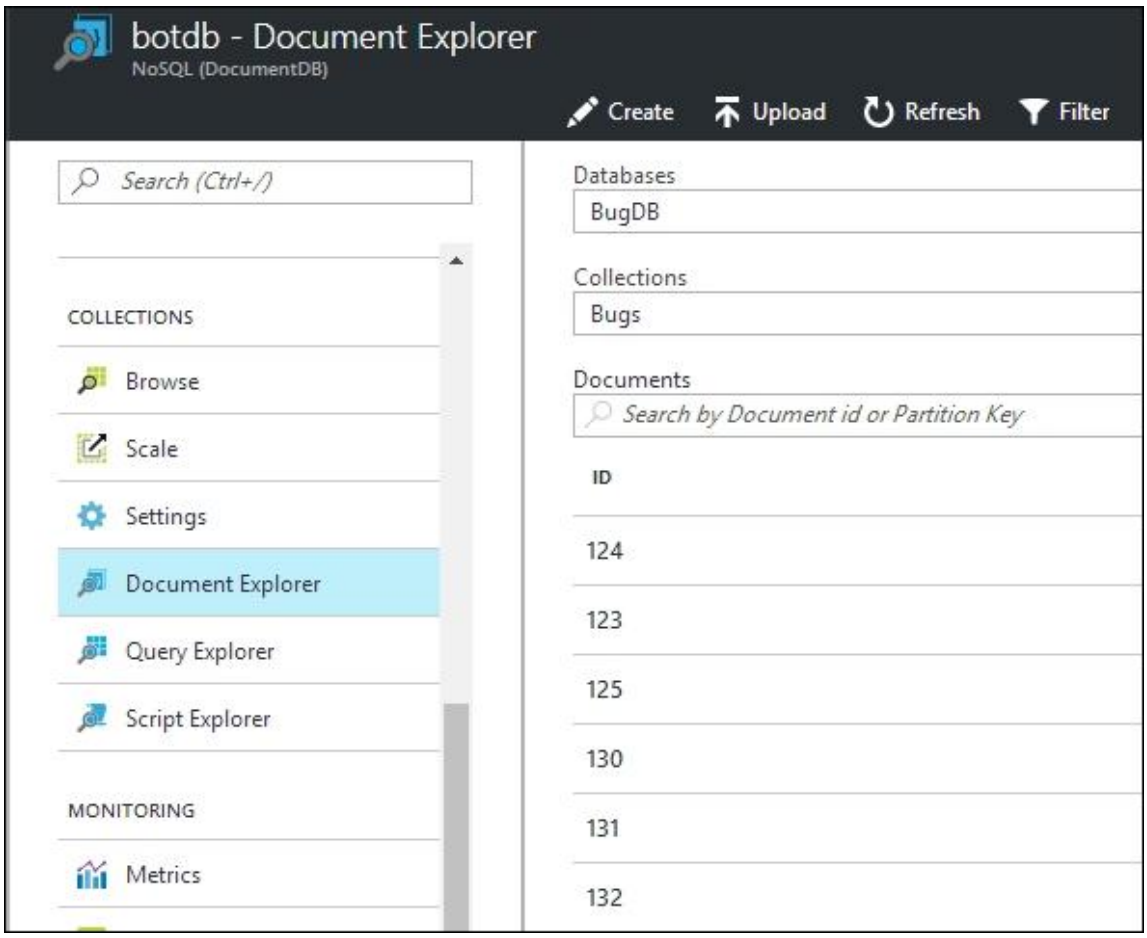

Рис. 13 Список заполненных ошибок

Для подключения базы данных к боту нужно установить пакет documentdb в nodeJS. Он находится по адресу https://www.npmjs.com/package/documentdb.

Чтобы установить его, нужно выполнить команду ниже.

```
npm install documentdb --save
```

Выводом команды будет что-то похожее на рисунок 14.

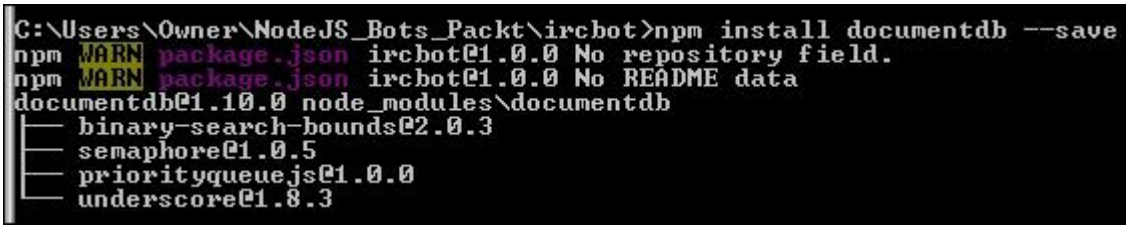

Рис. 14 Установка documentdb

Дальше нужно обновить файл app.js, чтобы бот мог получить доступ к данным на основе DocumentDB с помощью API-интерфейсов DocumentDB.

Чтобы связать DocumentDB с Node.js, будет использоваться следующий код:

```
var DocumentClient = require('documentdb').DocumentClient;
var host = "https://botdb.documents.azure.com:443/";
```

```
var masterKey = "<YOUR PRIMARY KEY>";
var docclient = new DocumentClient(host, {masterKey: masterKey});
docclient.readDocuments('dbs/BugDB/colls/Bugs').toArray(function (err, docs) {
    console.log(docs.length + ' Documents found');});});
```

Для создания экземпляра DocumentClient понадобится host и masterkey учетной записи DocumentDB. Найти их можно в разделе «Keys», как показано на рисунке 15.

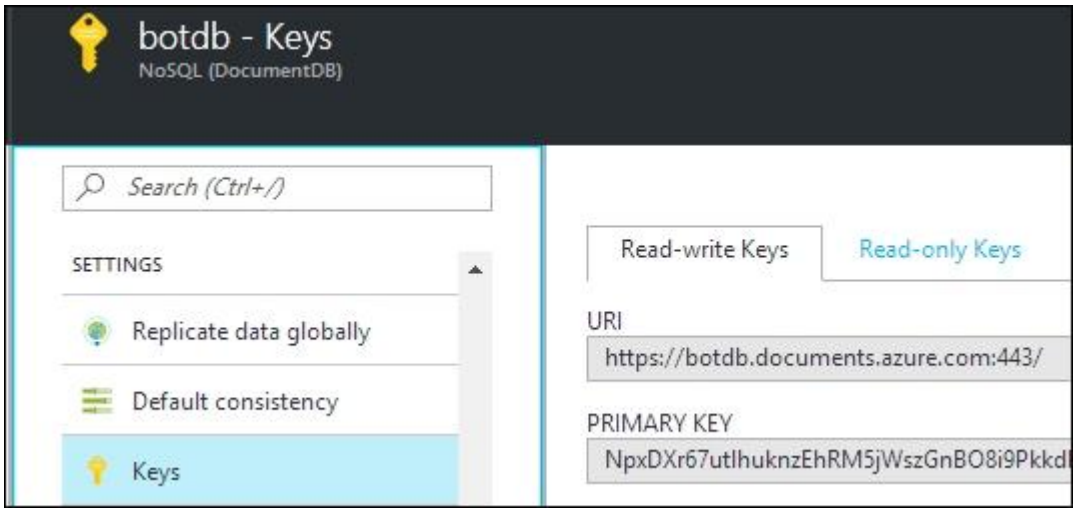

Рис. 15 Вкладка с ключами доступа

Чтобы прочитать все документы из коллекции, можно использовать следующие строки кода.

```
docclient.readDocuments('dbs/BugDB/colls/Bugs').toArray(function (err, docs) {
    console.log(docs.length + ' Documents found');});});
```

В readDocuments() требуется аргумент ссылки на коллекцию. Эта ссылка на коллекцию — это просто путь к коллекции. Это выглядит следующим образом

```
dbs/<Your Database>/colls/<Your Collection ID>
```

app.js сейчас должно выглядеть так.

```
var irc = require('irc');
var client = new irc.Client('irc.freenode.net', 'BugTrackerIRCBot', {
  autoConnect: false});
client.connect(5, function(serverReply) {
  console.log("Connected!\n", serverReply);
  client.join('#BugsChannel', function(input) {
    console.log("Joined #BugsChannel");
    client.say('#BugsChannel', "Hi, there. I am an IRC bot which track bugs or
defects for your team.\n I can help you using following commands.\n BUGREPORT \n
BUG # <BUG. NO>");});});});
var DocumentClient = require('documentdb').DocumentClient;
var host = "https://botdb.documents.azure.com:443/";
var masterKey = "<YOUR PRIMARY KEY>";
var docclient = new DocumentClient(host, {masterKey: masterKey});
docclient.readDocuments('dbs/BugDB/colls/Bugs').toArray(function (err, docs) {
    console.log(docs.length + ' Documents found');
```

```
});
```

Как только запустить app.js, программа подключится к коллекции с помощью Microsoft Azure DocumentDB Node.js SDK. После прочтения документов в командной строке можно заметить колличество прочитанных документов, как на рисунке 16.

Рис. 16 Вывод количества прочитанных документов

На данный момент бот подключается к DocumentDB и получает оттуда документы.

Чтобы связать все вместе, нужно изменим файл app.js, чтобы он был как показано ниже.

```
var irc = require('irc');
var client = new irc.Client('irc.freenode.net', 'BugTrackerIRCBot', {
    autoConnect: false});
client.connect(5, function(serverReply) {
    console.log("Connected!\n", serverReply);
    client.join('#BugsChannel', function(input) {
        console.log("Joined #BugsChannel");
        client.say('#BugsChannel', "Hi, there. I am an IRC Bot which track bugs or
defects for your team.\n I can help you using following commands.\n BUGREPORT \n
BUG # <BUG. NO>");});});});
var DocumentClient = require('documentdb').DocumentClient;
var host = "https://botdb.documents.azure.com:443/";
var masterKey = "<PRIMARY KEY>";
var docclient = new DocumentClient(host, {masterKey: masterKey});
client.addListener('message', function (from, to, text) {
    var str = text;
    if (str.indexOf('BUGREPORT') === -1){
        if (str.indexOf('BUG #') === -1){
            client.say('#BugsChannel', "I could not get that!\n Send me commands
like,\n BUGREPORT \n BUG # <BUG. NO>");
        }else {
            client.say('#BugsChannel', "So you need info about "+text);
            client.say('#BugsChannel', "Wait for a moment!");
            var t= text.substring(6,text.length);
            var temp = t.trim();
            var querySpec = {
                query: 'SELECT * FROM Bugs b WHERE b.id= @id',
                parameters: [{name: '@id', value: temp}]};
            docclient.queryDocuments('dbs/BugDB/colls/Bugs',
querySpec).toArray(function (err, results) {
                if (results.length>0){
                    client.say('#BugsChannel', "["+ results[0].url+"] [Status]:
"+results[0].status+" [Title]:"+results[0].title);}
                else{client.say('#BugsChannel', 'No bugs found.'); }});}}
    else{
        client.say('#BugsChannel', "So you need a Bug Report!");
        client.say('#BugsChannel', "Wait for a moment!");
        var querySpec = {
            query: 'SELECT * FROM Bugs b WHERE b.status= @status',
            parameters: [{name: '@status', value: 'Open'}]};
```

```
        docclient.queryDocuments('dbs/BugDB/colls/Bugs',
querySpec).toArray(function (err, results) {
            client.say('#BugsChannel','Total Open Bugs:'+results.length);
        });
        var querySpec = {
            query: 'SELECT * FROM Bugs b WHERE b.status= @status',
            parameters: [{name: '@status', value: 'Closed'}]};
        docclient.queryDocuments('dbs/BugDB/colls/Bugs',
querySpec).toArray(function (err, results) {
            client.say('#BugsChannel','Total Closed Bugs:'+results.length);});});});
```

После запуска обновленного app.js в консоли должны быть сообщения как на рисунке 17.

```
C:\Users\Owner\NodeJS_Bots_Packt\ircbot>node app.js
Connected!
 { prefix: 'verne.freenode.net',
  server: 'verne.freenode.net',
  command: 'rpl_welcome',
  rawCommand: '001',
  commandType: 'reply',
  args:
   [ 'BugTrackerIRCBot',
     'Welcome to the freenode Internet Relay Chat Network BugTrackerIRCBot' ] }
Joined #BugsChannel
```

Рис. 17 Запуск готовой версии

Теперь на канале «#BugsChannel» IRC-клиента должен быть активен бот, как на рисунке 18.

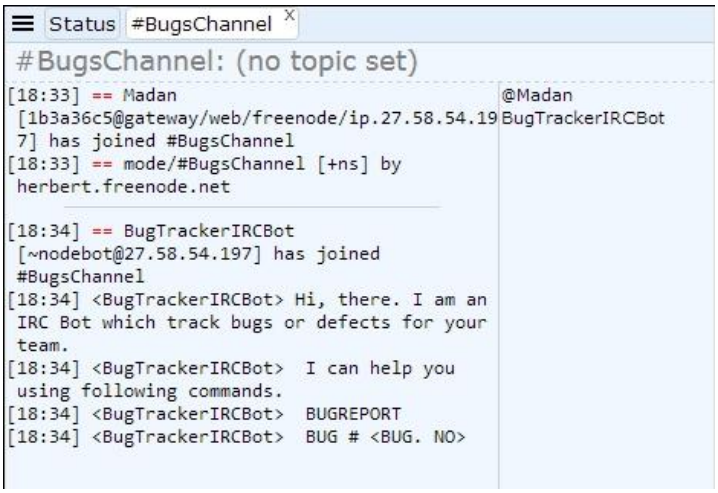

Рис. 18 Работа обновленного бота в чате

При вводе команды «BUGREPORT», бот сгенерирует отчет по всем существующим ошибкам, как на рисунке 19.

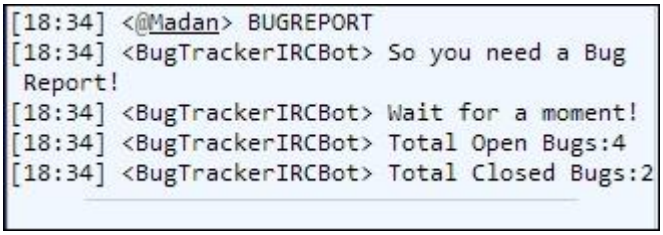

Рис. 19 Вывод статистика ботом

При запросе информации про конкретную ошибку, например «BUG#125» бот выдаст информация с ссылкой на подробную информацию как на рисунке 20.

```
[19:34] <@Madan> BUG # 125
[19:34] <BugTrackerIRCBot> So you need info
 about BUG # 125
[19:34] <BugTrackerIRCBot> Wait for a moment!
[19:34] <BugTrackerIRCBot>
 [http://mybugsystem.net/bugs/125] [Status]:
 Open [Title]:Left menu is not collapsing.
```

Рис. 20 Вывод информации о конкретной ошибке

При переходе по данной ботом ссылке, открывается страница с Json информацией по ошибке, как на рисунке 21.

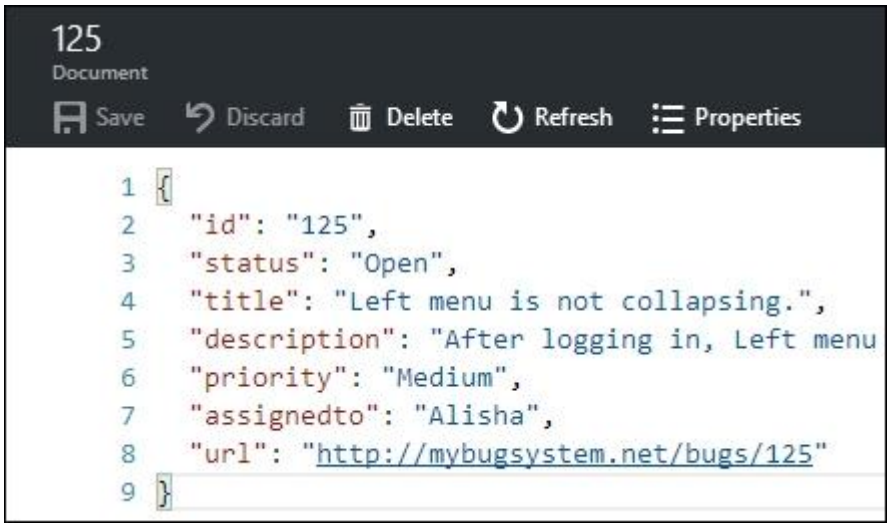

Рис. 21 Подробная информация по ошибке

При запросе ошибки, которой нет в базе, бот ответит соответствующим образом как на рисунке 22.

```
[19:25] <@Madan> BUG # 12345
[19:25] <BugTrackerIRCBot> So you need info
 about BUG # 12345
[19:25] <BugTrackerIRCBot> Wait for a
 moment!
[19:25] <BugTrackerIRCBot> No bugs found.
```

Рис. 22 Сообщение о несуществующей ошибки

Вывод

Каждой команде разработчиков нужны инструменты для отслеживания ошибок и отчетов. Обычно необходимы сообщения об ошибках и их назначение. В случае критических проектов эти потребности становятся очень важными для сроков проекта. В этой статье был создан очень простой IRC-бот в Node.js и протестирован, как он может

взаимодействовать внутри канала с помощью веб-клиента IRC. Затем расширен базовый функционал бота так, что по запросу пользователя бот быстро и легко предоставлял информацию. Также использовалось облачное хранилище на основе Azure для хранения базы данных ошибок.

В сегодняшнем мире сотрудничества команды разработчиков, использующие такие интеграции и автоматизацию, будут эффективны и действенны при предоставлении своих качественных продуктов.

Библиографический список

1. Сазанова Л.А., Зенков М.А. Чат-боты как современный инструмент бизнес-анализа // В сборнике: BI-технологии и корпоративные информационные системы в оптимизации бизнес-процессов цифровой экономики. Материалы VI Международной научно-практической очно-заочной конференции. 2019. С. 22-23. URL: https://www.elibrary.ru/item.asp?id=37638725 (Дата обращения: 29.01.2021)
2. Жигулин М.В., Коломеец А.В., Кушик Н.Г., Шабалдин А.В. Тестирование программной реализации протокола irc на основе модели расширенного автомата // Известия Томского политехнического университета. 2011. Т. 318. № 5. С. 81-84. URL: https://www.elibrary.ru/item.asp?id=16398372 (Дата обращения: 29.01.2021)
3. Слободской А. IRC-сервер // Системный администратор. 2003. № 7 (8). С. 28-30. URL: https://www.elibrary.ru/item.asp?id=20343448 (Дата обращения: 29.01.2021)