Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«САРАТОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н.Г. ЧЕРНЫШЕВСКОГО»

Кафедра дискретной математики и
информационных технологий

RK СЕРИИ "КОННОР" ДЛЯ ДЛЯ ПОМОЩИ В РАССЛЕДОВАНИИ ДЕЛ, СВЯЗАННЫХ С ДЕВИАНТНЫМИ АНДРОИДАМИ

КУРСОВАЯ РАБОТА

Студента 3 курса 321 группы
направления 09.03.01 — Информатика и вычислительная техника
факультета КНиИТ
Экгарт Викентия Александровича

Научный руководитель
ассистент _____ П.П. Поздняков

Заведующий кафедрой
к.ф.-м.н. доцент _____ Л.Б. Тяпаев

Саратов 2019

Содержание

ВВЕДЕНИЕ

Целью данной курсовой работы является разработка чат-бота, работающего с API социальной сети "Вконтакте"для помощи старосте в оповещении студентов о различных мероприятиях и прочих объявлениях, а также для мгновенного получения акутальной информации о порядке текущей недели в расписании, без необходимости ручного высчитывания или обращения к сторонним ресурсам. Данная социальная сеть была выбрана, поскольку количество пользователей (в том числе и студентов ВУЗа) в ней максимально, в отличае от других социальных сетей. Также, данная социальная сеть полностью поддерживает законодательство РФ. В работе будет рассмотрен полный путь от знакомста с API социальной сети и, до конечного запуска бота на сервере.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Дать определение понятию «API».
2. Ознакомиться с официальной документацией API от выбранной социальной сети.
3. Выбрать технологии для решения поставленной задачи.
4. Разработать чат-бота.

1 Понятие «API»

API — application programming interface, по русски - программный интерфейс приложения, Т.е. это описание способов (набор определенных "открытых"методов с парамаетрами, констант для изменения) которыми одно компьютерное программное обеспечение может взаимодействовать с другим компьютерным ПО. При этом, как правило, при хорошем проектировании API, не имеет значения на каком языке будет написана программа, которая будет обращаться к этому API. Так, например, программа написанная на C++ может работать с API программы, написанной на Java.

2 Ознакомление с официальной документацией API Вконтакте

Итак, документация данной социальной сети находится по слудующему адресу https://vk.com/dev/manuals.

Для решения поставленной задачи, заходим в раздел с чат ботами, и видим, что для взаимодействия бота и сервера социальной сети предусмотрено два способа общения "клиент-сервер":

— Callback API.

— Long Poll API.

Callback API — согласно официальной документации, работает следующим образом - как только в сообществе происходит нужное событие - от сервера Вконтакте приходит уведомление на наш сервер. При этом, событие может быть каким угодно: комментарий к фотографии, новая запись на стене, вступление в сообщество, отправка сообщения, и многое другое. [2].

Второй способ получения обновлений — это подключение к Long Poll серверу. В отличие от Callback API, Long Poll сервер будет присылать на наш сервер только те обновления, которые связаны с сообщениями. Никаких других событий из сообщества в нём нет. То есть, узнать о том, присоединился ли пользователь к сообществу - нельзя.

Исходя из поставленной задачи, а также из-за некоторых преимуществ в скорости Long Poll API над обычным [1], принято решение использовать Long Poll API.

Теперь необходимо создать сообщество в Вконтакте, от имени которого будет писать чат бот.

3 Выбор технологии для решения поставленной задачи

Выбранный мною функционал чат-бота предполагает следующий сценарий использования : Создатель беседы (беседа - чат между несколькими пользователями) или любой другой её участник добавлеят бота в беседу. Далее администатор беседы открывает боту доступ ко всем сообщениям в ней и делает бота администратором (необходимо для работы функции "позови всех")

После этого доступны следующие команды боту :

— какая неделя [сейчас, завтра, следующая]

— позови всех

— объявление

— инфо, помощь, что ты умеешь

Первая команда должна вернуть информацию о том, какая сейчас (или иная, зависит от запроса пользователя) неделя. Например, бот может ответить "сейчас числитель что означает что в данный момент актуальны пары из верхних ячеек расписания.

Вторая команда "позови всех"отправляет каждому пользователю уведомление, даже если уведомления у этого пользователя отключены. Полезно, так как многие пользователи в виду частых и неинфрмативных сообщени (флуд) отключают уведомления из беседы и могут пропустить что-то важное.

Третья команда "объявление— отправляет каждому пользователю уведомление с текстом, идущем после этой команды. Работет даже если уведомления у этого пользователя отключены. Как в случае и с предидущим пунктом, защищает пользователей от несвоевременного информирования при отключенных уведомлениях.

Последняя команда выводит информацию о доступных командах на данный момент.

Для достижения такого механизма работы мной были выбраны следующие технологии:

1. enviroment.
   — Node JS. (серверный интрператор JS)
   — PM2. (перезапуск процесса в случае ошибки, а также более продвинутое логирование)

2. bot.
    - NPM (пакетный менеджер для работы с зависимостями).
    - vk-node-sdk (библиотека для взаимодействием с LongPoll API).
    - util (библиотека для логирования ошибок)
    - JavaScript (язык программирования).
3. Сервер.
    - Ubuntu 17.10 64bit (512 МБ RAM 20 ГБ SSD 1 CPU).

Посльку не требуется хранить какие-либо данные, полученные от пользователя, база данных не учавствует в проекте.

В качестве IDE была использована VS Code от ©Microsoft. Данная IDE имеет встроенную поддержку JavaScript, системы контроля версий, а также распространяется бесплатно.

4 Разработка чат бота

Разработка велась в три этапа — программирование и создание серверной части - самого бота, загрузка проекта на сервер и ручное тестирование функционала.

4.1 чат-бот

С помощью CLI (command line interface) была создана директория для разработки и далее была введена команда ‘npm init‘ , которая создает конфигурационный файл трбуемого приложения. Полученный файл оказался таким

```
1
2   {
3     "name": "bot_vk",
4     "version": "1.0.0",
5     "description": "konnor testing",
6     "main": "konnor.js",
7     "dependencies": {
8       "node-vk-bot": "1.2.1",
9       "util": "^0.11.0",
10      "vk-node-sdk": "^0.1.8",
11    },
12    "scripts": {
13      "test": "echo \"Error: no test specified\" && exit 1"
14    },
15    "author": "",
16    "license": "ISC"
17  }
```

Далее, был создан главный js файл — "konnor.js в котором необходимо с помощью библиотеки "vk-node-sdk"подключиться к серверверу Вконтакте для приёма уведомлений.

Для этого, создал файл с константами, где будут указаны токены для работы с ботом, а также ID сообщества, от чьего имени будет отвечать бот. Разнесение архитектуры приложения на несколько файлов упростит его поддежку в дальнейшем и поможет делать важные изменения без затрагивания большого количства кода.

Вот как выглядит данный файл сейчас:

```
1  const vkGroupFullRight = 'group on vk token fith full rights and
       long-poll';
2  const groupId = 000;
3
4  module.exports = {
5      vkGroupFullRight: vkGroupFullRight,
6      groupId: groupId,
7  }
```

В главном файле, импортируем этот файл и инициализируем бота.

```
1
2  const DEBUG_MODE = require('./konnor_config');
3  const util = require('util');
4  const { Bot } = require('node-vk-bot');
5
6  //don't forget to add tokens in file and rename him
7  const TOKENS = require('./secret_tokens');
8
9  const regName = /коннор|connor|конор|андроид/i;
10
11
12 const debugConsole = (variable, depth) => {
13     DEBUG_MODE && console.log('debug: ' + util.inspect(variable,
           false, depth = 8));
14 }
15
16 const bot = new Bot({
17     token: TOKENS.vkGroupFullRight,
18     group_id: TOKENS.groupId,
19 }).start()
20
21 console.log('bot started');
22
23
24 bot.on('poll-error', error => {
25     console.error('error occurred on a working with the Long Poll
           server ' +
26         `(${util.inspect(error)})`)
27 })
```

Для запуска бота введем команду "node ./konnor.js". Бот запущен!

Прежде чем писать реакцию на команды, в соотвествии с принципами "Банды четырех" [3] — выносим каждую команду в отдельный модуль и делаем "горячее подключение"модулей. То есть не требуется переписывать общую логику бота, при добавлении новой команды в будущем.

Каждая команда - это объект, вида:

```
  const имя_команды = {
    callName: 'триггер срабатывания, может быть регулярным
       выражением',
    action: (bot, message, TOKENS) => {
      дейсвие - что она делает
    }
}
```

Также, подключение команд реализуем в отдельном файле, дабы не трогать основной.

Сама же логика бота - на какую команду он среагирует будет осуществляться следующим образом — бот получает список всех доступных команд и входящее сообщение. Далее простым циклом проходясь по списку команд, он сверяет "триггер"(регулярное выражение) каждой команды с входным сообщением. И если регулярное выражение проходит (результат true), то бот исполняет нужную команду и передаёт свой ответ на отправку.

Данный паттерн называется "Стратегия"и очень сильно экономит ресуры и облегчает добваление новых команд.

Маршрутов четыре — в приложении четыре страницы:

— Поиск встречи.

— Создание встречи.

— О приложении.

— Просмотр конкретной встречи.

```
import Vue from 'vue'
import App from './components/App.vue'
import VueRouter from 'vue-router'
import Vuetify from 'vuetify'
import './stylus/main.styl'
import VueResource from 'vue-resource'

import FindEvent from './components/pages/FindEvent.vue'
import CreateEvent from './components/pages/CreateEvent.vue'
```

```
import Event from './components/pages/Event.vue'
import About from './components/pages/About.vue'

Vue.use(Vuetify);
Vue.use(VueRouter);
Vue.use(VueResource);

let router = new VueRouter({
    routes: [
        {
            name: 'find',
            path: '/',
            components: { default: FindEvent }
        }, {
            name: 'create',
            path: '/create',
            components: { default: CreateEvent }
        }, {
            name: 'event',
            path: '/event/:id',
            components: { default: Event}
        }, {
            name: 'about',
            path: '/about',
            components: {default: About}
        }
    ]
});

new Vue({
    el: '#app',
    router: router,
    render: h => h(App)
});
```

После создадим компоненты — главный компонент — App, в который встроим четыре основных - на каждую страницу [?].

Компонент App.vue, в котором будет отображаться боковое меню и верхнее содержимое, а также отображаться кнопка прокрутки вверх, когда пользователь прокручивает контент вниз. В соответствии с концепцией Vue — содержимое данного компонента будет доступно конечному пользователю

на любом экране.

```
<template>
  <v-app id="inspire" v-scroll="onScroll">
    <v-navigation-drawer
      fixed
      v-model="drawer"
      app
    >
      <v-list dense>
        <v-list-tile @click="switchPage('/')">
          <v-list-tile-action>
            <v-icon>place</v-icon>
          </v-list-tile-action>
          <v-list-tile-content>
            <v-list-tile-title>Найти встерчу</v-list-tile-title>
          </v-list-tile-content>
        </v-list-tile>
        <v-list-tile @click="switchPage('/create')">
          <v-list-tile-action>
            <v-icon>edit</v-icon>
          </v-list-tile-action>
          <v-list-tile-content>
            <v-list-tile-title>Создать встречу</v-list-tile-title>
          </v-list-tile-content>
        </v-list-tile>
        <v-list-tile @click="switchPage('/about')">
          <v-list-tile-action>
            <v-icon>help</v-icon>
          </v-list-tile-action>
          <v-list-tile-content>
            <v-list-tile-title>О проекте</v-list-tile-title>
          </v-list-tile-content>
        </v-list-tile>
      </v-list>
    </v-navigation-drawer>
    <v-toolbar color="indigo" dark fixed app>
      <v-toolbar-side-icon @click.stop="drawer = !drawer"></v-
          toolbar-side-icon>
      <v-toolbar-title>Meet&Greet</v-toolbar-title>
    </v-toolbar>

```

```
40      <main>
41        <v-content>
42          <keep-alive>
43            <router-view></router-view>
44          </keep-alive>
45        </v-content>
46      </main>
47
48      <v-fab-transition>
49        <v-btn style="bottom: 30px"
50               absolute
51               dark
52               fab
53               fixed
54               bottom
55               right
56               v-show="showBtnUp"
57               class='indigo'
58               v-on:click="scrollToTop"
59        >
60          <v-icon>keyboard_arrow_up</v-icon>
61        </v-btn>
62      </v-fab-transition>
63      <v-footer color="indigo" app>
64        <span class="white--text">&copy; course project. 2018</span>
65      </v-footer>
66    </v-app>
67  </template>
68
69  <script>
70    export default {
71      data: () => ({
72        drawer: null,
73        showBtnUp: false,
74        offsetTop: 0,
75      }),
76      methods: {
77        scrollToTop: function () {
78          window.scroll({ top: 0, left: 0, behavior: 'smooth' });
79        },
80        onScroll (e) {
```

```
81          this.offsetTop = window.pageYOffset;
82          if (this.offsetTop>=280){
83            this.showBtnUp = true
84          }
85          else this.showBtnUp = false
86        },
87        switchPage: function(link) {
88          scrollTo(0,0);
89          this.$root.$router.push({ path: link });
90        },
91      }
92    }
93 </script>
```

После были созданы страничных компоненты, которые будут вложены в главный компонент — App.

С помощью

<style src="../css/Events.css»</style>

есть возможность подключения таблиц стилей в конкретный компонент.

С помощью команды npm build — скомпилируем клиентскую часть, в особом режиме — production mode. Данная команда собирает все скрипты в один файл и оптимизирует их, удаляя лишние символы и обфусифицирует код — заменяет понятные программисту переменные вроде showButton на ab, тем самым защищая код от неправомерного использования [?].

4.2 Развертка приложения на сервере

Запустим команду gradle build [?]. Она соберет все зависимости для backend части, а также уже собранную клиентскую часть в один файл-пакет meetAndGreet-0.0.1-SNAPSHOT.jar.

После этого, необходимо запустить файл, для убеждения в том, что приложение работает. Запускается оно с помощью команды (разработка ведется под ОС Windows, поэтому команда именно такая).

```
1 C:\ProgramData\Oracle\Java\javapath\java.exe -jar meetAndGreet
    -0.0.1-SNAPSHOT.jar
```

Приложение запуститься на 80 порту, что весьма удобно — не нужно указывать порт в адресной строке.

Далее необходимо подключиться к серверу по протоколу ssh и скопировать файл приложения в рабочую директорию. Если просто запустить такую команду

java -jar meetgreet.jar

то после закрытия консоли и отключения от сервера, приложение перестанет работать. Необходимо создать bash скрипт, в котором не только будет запуск приложения, но и логгирование в файл.

Содержимое bash скрипта

```
#!/bin/bash
java -jar meetgreet.jar > log.txt
```

После запуска этого скрипта необходимо отключиться от сервера и проверить доступность приложения с помощью браузеров по IP адресу сервера.

Ниже на Рисунке 1 и Рисунке 2 приведены скриншоты приложения.

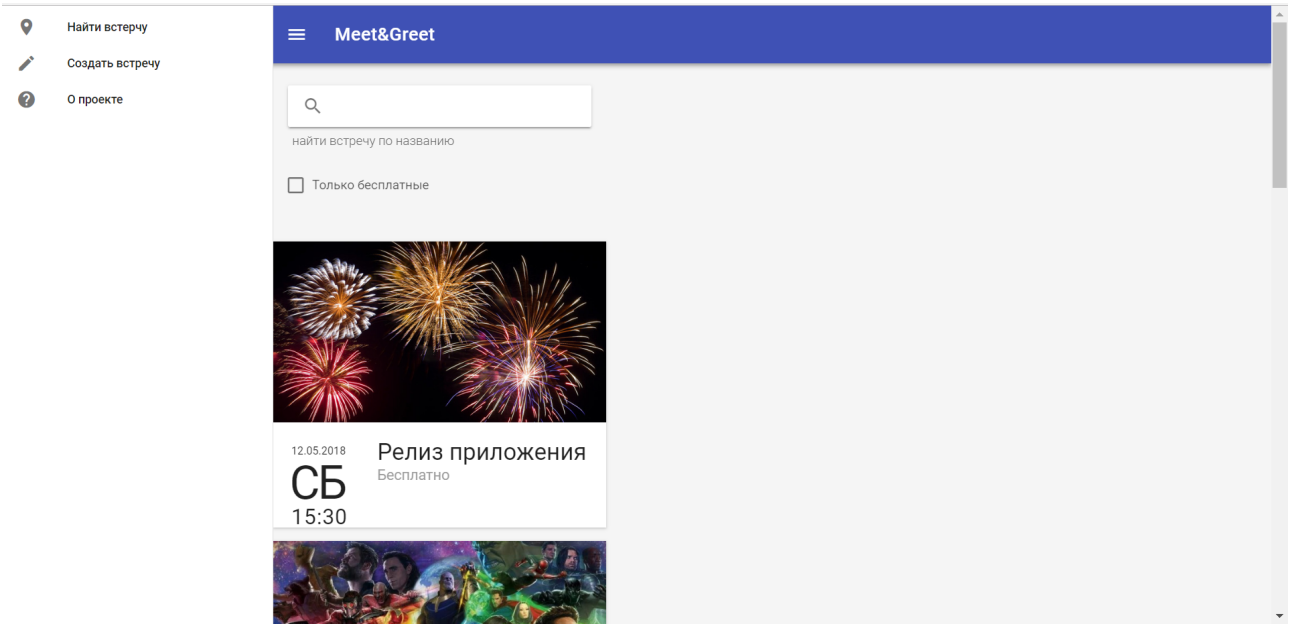

Рисунок 1 – Приложение при просмотре с настольных компьютеров.

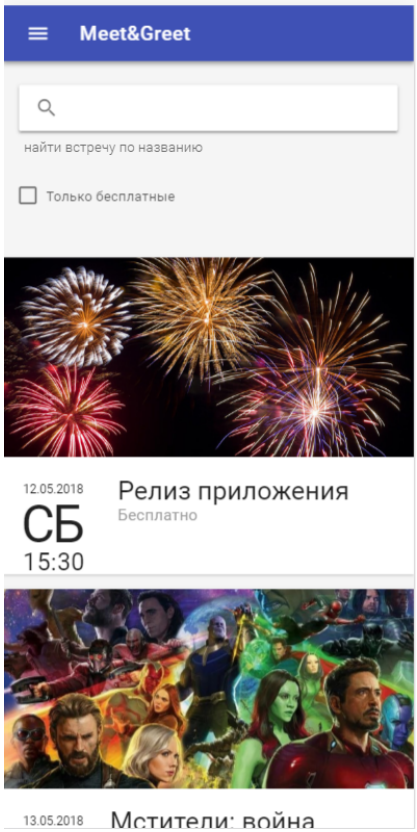

Рисунок 2 – Приложение при просмотре с мобильных устройств.

ЗАКЛЮЧЕНИЕ

При выполнении данной курсовой работы, были рассмотрены современные Web технологии для построения SPA. Было разработано приложение с адаптивной верстkой которое работает одинаково хорошо как на мобильных устройствах, так и на настольных. Исходный код приложения доступен в репозитории GitHub:

https://github.com/vikegart/meetAndGreet

В дальнейшем планируется добавить такие действия со встречами как обновление и удаление. Также планируется изучить Spring Security и добавить авторизацию.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Bots Long Poll API [Электронный ресурс]: URL: https://vk.com/dev/bots_longpoll (Дата обращения: 18.02.2019) Загл. с экрана. Яз. англ;

2 API для чат-ботов [Электронный ресурс]: URL: https://vk.com/dev/bots_docs (Дата обращения: 18.02.2019) Загл. с экрана. Яз. англ;

3 Приемы объектно-ориентированного проектирования. Паттерны проектирования. Влиссидес Джон , Джонсон Р. , Хелм Ричард , Гамма Эрих Яз. англ стр [50-120];