

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

РК 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	3
1 Понятие «API»	4
2 Ознакомление с официальной документацией API Вконтакте	5
3 Выбор технологии для решения поставленной задачи	6
4 Разработка чат бота	8
4.1 frontend	8
4.2 Развертка приложения на сервере	14
ЗАКЛЮЧЕНИЕ	16
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	17

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 актуальной 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 отличие от других социальных сетей. Также, данная социальная сеть полностью поддерживает законодательство РФ. В работе будет рассмотрен полный путь от знакомства 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 параметрами, 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 следующему 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 добавляет бота в беседу. Далее администратор 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 неинформативных сообщений (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 предыдущим 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 интерпретатор 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 взаимодействия 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 участвует 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 frontend

С помощью CLI (command line interface) была создана директория frontend и далее была введена команда ‘npm init’, которая создает конфигурационный файл трбуемого приложения.

```
1 {
2   "name": "meet",
3   "version": "1.0.0",
4   "description": "meetings in Saratov",
5   "author": "vikegart <vikegart@gmail.com>",
6   "private": true,
7   "scripts": {
8     "dev": "webpack-dev-server --inline --progress --config build/
9       webpack.dev.conf.js",
10    "start": "npm run dev",
11    "build": "node build/build.js"
12  },
13  "dependencies": {
14    "moment": "^2.20.1",
15    "vue": "^2.5.3",
16    "vue-resource": "^1.3.4",
17    "vue-router": "^3.0.1",
18    "vuetify": "^0.17.3",
19    "vuex": "^3.0.1"
20  },
21  "devDependencies": {
22    "autoprefixer": "^7.1.2",
23    "babel-core": "^6.22.1",
24    "babel-loader": "^7.1.1",
25    "babel-plugin-add-filehash": "^6.9.4",
26    "babel-plugin-transform-runtime": "^6.22.0",
27    "babel-preset-env": "^1.3.2",
28    "babel-preset-stage-2": "^6.22.0",
29    "babel-register": "^6.22.0",
30    "chalk": "^2.0.1",
```



```

30   "copy-webpack-plugin": "^4.0.1",
31   "css-loader": "^0.28.0",
32   "eventsource-polyfill": "^0.9.6",
33   "extract-text-webpack-plugin": "^3.0.0",
34   "file-loader": "^1.1.4",
35   "friendly-errors-webpack-plugin": "^1.6.1",
36   "html-webpack-plugin": "^2.30.1",
37   "webpack-bundle-analyzer": "^2.9.0",
38   "node-notifier": "^5.1.2",
39   "postcss-import": "^11.0.0",
40   "postcss-loader": "^2.0.8",
41   "semver": "^5.3.0",
42   "shelljs": "^0.7.6",
43   "optimize-css-assets-webpack-plugin": "^3.2.0",
44   "ora": "^1.2.0",
45   "stylus": "^0.54.5",
46   "stylus-loader": "^3.0.1",
47   "rimraf": "^2.6.0",
48   "url-loader": "^0.5.8",
49   "vue-loader": "^13.3.0",
50   "vue-style-loader": "^3.0.1",
51   "vue-template-compiler": "^2.5.3",
52   "portfinder": "^1.0.13",
53   "webpack": "^3.6.0",
54   "webpack-dev-server": "^2.9.1",
55   "webpack-merge": "^4.1.0"
56 },
57 "engines": {
58   "node": ">= 4.0.0",
59   "npm": ">= 3.0.0"
60 },
61 "browserslist": [
62   "> 1%",
63   "last 2 versions",
64   "not ie <= 8"
65 ]
66 }

```

Для запуска клиентской части требуется ввести в консоль команду `npm start` [?].

Далее, был создан главный js файл — точка входа в SPA приложение,

в котором необходимо настроить маршруты — routings.

Маршрутов четыре — в приложении четыре страницы:

- Поиск встречи.
- Создание встречи.
- О приложении.
- Просмотр конкретной встречи.

```
1 import Vue from 'vue'
2 import App from './components/App.vue'
3 import VueRouter from 'vue-router'
4 import Vuetify from 'vuetify'
5 import './stylus/main.styl'
6 import VueResource from 'vue-resource'
7
8 import FindEvent from './components/pages/FindEvent.vue'
9 import CreateEvent from './components/pages/CreateEvent.vue'
10 import Event from './components/pages/Event.vue'
11 import About from './components/pages/About.vue'
12
13 Vue.use(Vuetify);
14 Vue.use(VueRouter);
15 Vue.use(VueResource);
16
17 let router = new VueRouter({
18   routes: [
19     {
20       name: 'find',
21       path: '/',
22       components: { default: FindEvent }
23     }, {
24       name: 'create',
25       path: '/create',
26       components: { default: CreateEvent }
27     }, {
28       name: 'event',
29       path: '/event/:id',
30       components: { default: Event }
31     }, {
32       name: 'about',
33       path: '/about',
34       components: { default: About }
```

```

35     }
36   ]
37 });
38
39 new Vue({
40   el: '#app',
41   router: router,
42   render: h => h(App)
43 });

```

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.

```

1 <template>
2   <v-app id="inspire" v-scroll="onScroll">
3     <v-navigation-drawer
4       fixed
5       v-model="drawer"
6       app
7     >
8     <v-list dense>
9       <v-list-tile @click="switchPage('/')">
10        <v-list-tile-action>
11          <v-icon>place</v-icon>
12        </v-list-tile-action>
13        <v-list-tile-content>
14          <v-list-tile-title>Найти встречу</v-list-tile-title>
15        </v-list-tile-content>
16      </v-list-tile>
17      <v-list-tile @click="switchPage('/create')">
18        <v-list-tile-action>
19          <v-icon>edit</v-icon>
20        </v-list-tile-action>
21        <v-list-tile-content>
22          <v-list-tile-title>Создать встречу</v-list-tile-title>
23        </v-list-tile-content>

```

```

24     </v-list-tile>
25     <v-list-tile @click="switchPage('/about')">
26         <v-list-tile-action>
27             <v-icon>help</v-icon>
28         </v-list-tile-action>
29         <v-list-tile-content>
30             <v-list-tile-title>О проекте</v-list-tile-title>
31         </v-list-tile-content>
32     </v-list-tile>
33 </v-list>
34 </v-navigation-drawer>
35 <v-toolbar color="indigo" dark fixed app>
36     <v-toolbar-side-icon @click.stop="drawer = !drawer"></v-
37         toolbar-side-icon>
38     <v-toolbar-title>Meet&Greet</v-toolbar-title>
39 </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 запускаться 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

```
1 #!/bin/bash
2 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.

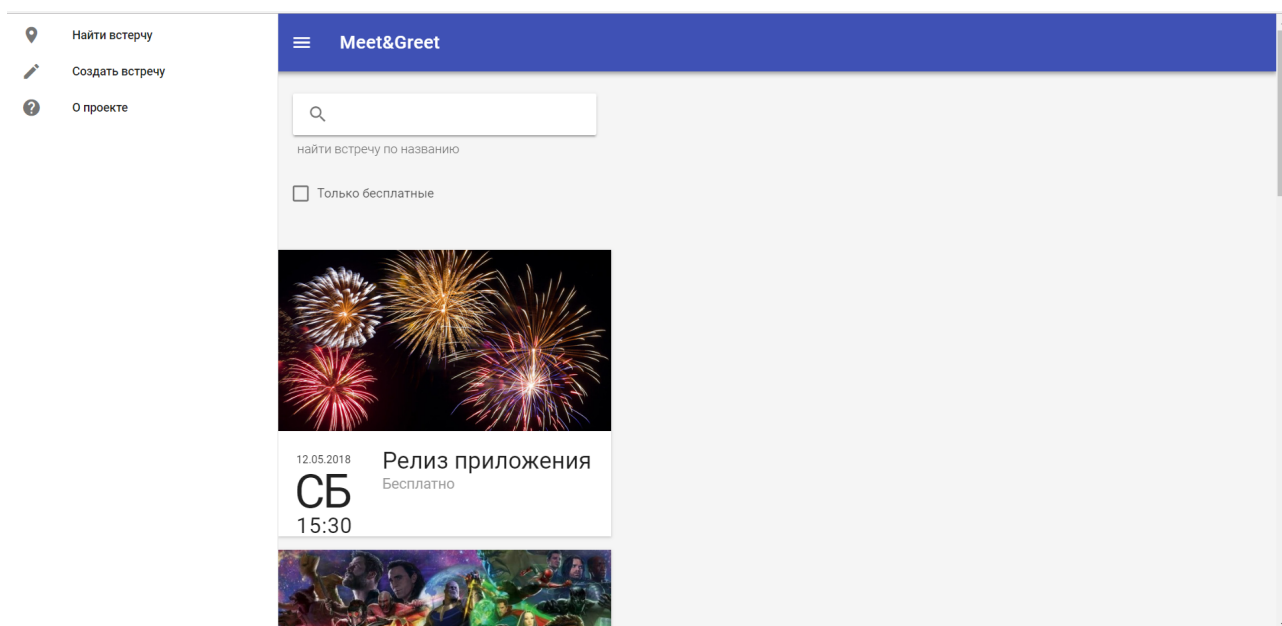


Рисунок 1 – Приложение при просмотре с настольных компьютеров.

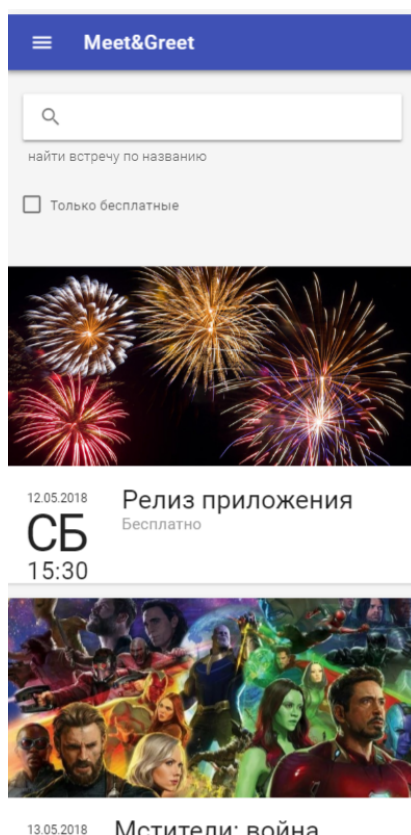


Рисунок 2 – Приложение при просмотре с мобильных устройств.

ЗАКЛЮЧЕНИЕ

При выполнении данной курсовой работы, были рассмотрены современные Web технологии для построения SPA. Было разработано приложение с адаптивной версткой 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;