

Содержание

ВВЕДЕНИЕ	3
1. ПОСТАНОВКА ЗАДАЧИ К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ.....	6
1.1. Постановка задачи	6
1.2. Основные требования к работе	7
1.3. Изучение предметной области	8
2. РАЗРАБОТКА ПРИЛОЖЕНИЯ	10
2.1. Выбор средств разработки	10
2.2. Составление дерева сценариев	12
2.3 Разработка базы данных.....	14
2.4 Программная реализация	19
3. ТЕСТИРОВАНИЕ	29
4. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ	38
ЗАКЛЮЧЕНИЕ	43
СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ	44
ПРИЛОЖЕНИЕ А	45
ПРИЛОЖЕНИЕ Б.....	46
ПРИЛОЖЕНИЕ В	52

ВВЕДЕНИЕ

Для начала обозначим понятие, что такое социальная сеть.

Социальная сеть — это платформа, онлайн-сервис или сайт, предназначенный для построения, отражения и организации социальных взаимоотношений, визуализацией которых являются социальные графы. [1]

В современном мире практически каждый из нас использует различные социальные сети, например, «ВКонтакте», «Одноклассники» и пр. На данный момент «ВКонтакте» пользуются 97 миллионов человек в месяц [6]. В них мы находим и общаемся с новыми и старыми друзьями, можем купить товары или найти ответы на интересующие нас вопросы, состоим в группа по интересам.

На момент 15.06.2019 «ВКонтакте» насчитывает 183 481 723 групп и сообществ.

С ростом пользователей социальных сетей у групп возникла потребность в помощнике, который смог быстро и информативно отвечать на часто задаваемые вопросы пользователей. Это и привело к созданию чат-бот.

Чат-бот — это программа, работающая внутри мессенджера. Такая программа способна отвечать на вопросы, а также самостоятельно задавать их. Чат-боты используются в разных сферах для решения типовых задач. [2]

Многие крупные компании, также используют чат-ботов, например, «Мегафон», «Сбербанк», «Ozone» и пр. Они раздают стикеры, показывают рекламные ролики, связанные с их компаниями, а также отвечают на часто задаваемые вопросы пользователей. Более подробно о функционале этих чат-ботов можно ознакомиться в официальных группах этих компаний.

Давайте рассмотрим подробнее функционал чат-бота от компании «Сбербанк» под названием «СберКот». Данный чат-бот выполняет следующие действия:

- Отправляет информацию о новых услугах банка
- Учит финансовой грамотности, при помощи статей и забавных комиксов с участием «СберКота».

Также в нем есть специальная клавиатура для пользователей, чтобы им было удобно и быстро оформить карту или найти отделение и банкомат «Сбербанк».

У нашего университета СибГУТИ, также имеется группы во «ВКонтакте». Самой большой по количеству пользователей (8285 за 15.06.2019) является группа «СибГУТИ». Эта группа имеет исключительно информационную составляющую, полезную в учёбе, студенческой жизни, новых технологий и пр. Электронный адрес этой группы: https://vk.com/sibsutis_universitet_svyazi

Вторая по количеству пользователей (4970 за 15.06.2019) является «Подслушано СибГУТИ». В ней студенты общаются между собой о учебе и о личном. Абитуриенты могут задать вопросы другим студентам. Электронный адрес этой группы: <https://vk.com/overhearsibsutis>

Исходя из статистики данных двух групп, можно сделать вывод, что суммарное количество студентов и абитуриентов составляет от 5000 до 8000 пользователей.

Чат-бота ни в одной официальной группе нашего ВУЗа нет. Это и послужило мотивацией для создания помощника, который бы помогал студентам и абитуриентам отвечать на вопросы, связанные с обучением и ВУЗом.

Прежде чем выбрать средства разработки чат-бота, необходимо определиться, какие команды будет выполнять данный помощник. Для этого пришлось провести небольшой опрос у студентов и приемной комиссии. Они описали, на какие основные вопросы хотят получать ответы студенты и абитуриенты. Вопросы приведены в разделе «Постановка задачи».

В разделе «Изучение предметной области», рассмотрены какие технологии используются для создания чат-ботов. Чтобы разработчикам было проще понять, с чем им надо будет работать, специально для них «ВКонтакте» создали документацию по различным технологиям, которые можно внедрить в эту социальную сеть. Подробнее можно ознакомиться по электронному адресу: <https://vk.com/dev/>

Раздел «Выбор средств разработки» описывает следующие пункты, которые необходимы для создания чат-бота:

- Язык программирования
- Среда выполнения кода
- Способ общения с сервером «ВКонтакте»
- База данных
- Воспитательные средства для создания программы

В самом разделе определены конкретные решения по всем пунктам и предоставлена подробная информация по этим технологиям.

Самым необходимым является раздел «Составление дерева сценариев». В нем подробно расписано следующее:

- Логика чат-бота
- Все возможные исходы диалога
- Как составлять дерево сценариев

Разделы «Разработка базы данных» и «Программная реализация» являются технической реализацией данного помощника. Подробно расписаны пункты:

- Описание используемой базы данных
- Как пользоваться Node.js, npm и MySQL
- Заполнение, добавление значений в MySQL, используя Node.js
- Подробно расписано как настроить группу «ВКонтакте» для чат-бота
- Как в программе реализовать логику чат-бота и какие методы использовались для этого

В третьей главе предоставлены результаты тестирования программы, которые выполнены посредством ручного тестирования. Это нужно для того, чтобы проверить правильность программной реализации данного чат-бота. Описаны какие данные и какие результаты получались при вводе различных значений.

Четвертая глава написана для пользователей чат-бота. Подробно описано следующее:

- Начало работы с чат-ботом
- Навигация по чат-боту
- Как правильно вводить значения для поиска
- Как пользоваться разделом для студентов и абитуриентов

Это руководство позволит пользователям правильно и быстро работать с чат-ботом. Особенно оно будет полезно тем, кто еще ни разу не пользовался данными помощниками.

1. ПОСТАНОВКА ЗАДАЧИ К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

1.1. Постановка задачи

Необходимо разработать чат-бот для СибГУТИ, который будет помогать студентам и абитуриентам отвечать на основные вопросы, которые они задают. Для этого необходимо их определить.

Вопросы абитуриентов:

- поступление
- частые вопросы
- новости
- где находится корпус
- направления
- военная кафедра
- контактные данные

Вопросы студентов:

- поиск аудитории
- расписание
- какая сейчас неделя
- мероприятия
- информация о деканатах
- часы работы Здравпункта

Отвечая на эти вопросы, помощник облегчит поиск информации по учебе, работу с сайтом ВУЗа, а также сделает официальную группу СибГУТИ более современной.

1.2. Основные требования к работе

Обозначим требования для реализации данной работы:

- интуитивно понятное меню для поиска нужного вопроса
- информативный ответ пользователю
- быстрота отклика чат-бота
- описать все возможные исходы диалога
- изучить VK API
- Оформление группы и настройка для чат-бота
- Составление руководства пользователя

1.3. Изучение предметной области

Прежде чем перейти к средствам разработки, необходимо понять, как работает чат-бот «ВКонтакте». Для этого необходимо разобрать основные термины.

Прежде чем мы обозначим основной термин VK API, необходимо понять, что такое API.

API — это посредник между разработчиком приложений и какой-либо средой, с которой это приложение должно взаимодействовать. API упрощает создание кода, поскольку предоставляет набор готовых классов, функций или структур для работы с имеющимися данными. [3]

VK API — это интерфейс, который позволяет получать информацию из базы данных vk.com с помощью http-запросов к специальному серверу. Вам не нужно знать в подробностях, как устроена база, из каких таблиц и полей каких типов она состоит — достаточно того, что API-запрос об этом «знает». Синтаксис запросов и тип возвращаемых ими данных строго определены на стороне самого сервиса. [4]

Если мы отправляем какой-то запрос, то мы ждем на него ответ. Это называется JSON-объект (формат записи данных в виде пар «имя свойства»: «значение» [4]). Если произошла ошибка, то придёт соответствующее сообщение и причина. В ином случае мы получаем запрошенные данные.

Здесь, как и в любой другой социальной сети, необходимо авторизоваться. Это нужно, чтобы сервер понимал, кто отправляет запрос и кому отправлять ответ. Поэтому приходится создать токен.

Токен - это строка из цифр и латинских букв, которую Вы передаете на сервер вместе с запросом. Из этой строки сервер получает всю нужную ему информацию. Есть разные способы получения токена, более того, он может быть выдан не только пользователю, но и сообществу, и сразу всему приложению. [4]

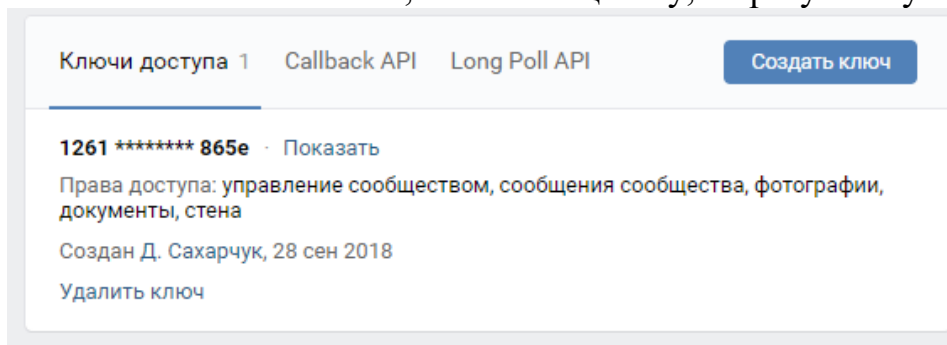


Рисунок 1.3.1 – Создание токена

Данный ключ доступа можно создать в группе в разделе «Управление», далее «Работа с API», в соответствии с рисунком 1.3.1.

Есть два способа реализовать взаимодействие с VK API:

- 1) Callback API
- 2) Bots Long Poll API

Для работы Callback API необходимо иметь сервер, который будет принимать и обрабатывать информацию, которую ему передали. Этот сервер должен находиться в сети интернета. Также, кроме токена, для дополнительной защиты можно указать секретный ключ.

Bots Long Poll API работает на компьютере пользователя. В данном способе мы не ждем, когда к нам придёт запрос, а сами с определенной частотой отправляем их на сервер «ВКонтакте». Очередь событий хранятся, на стороне «ВКонтакте», то есть мы просто читаем нужные нам события, которые происходят на сервере «ВКонтакте». Когда получили информацию, отправляем запрос и ждем ответа от сервера. Благодаря этому отправка сообщения чат-бот происходит быстрее, чем в Callback API. Пример такого соединения можно увидеть на рисунке 1.3.2.

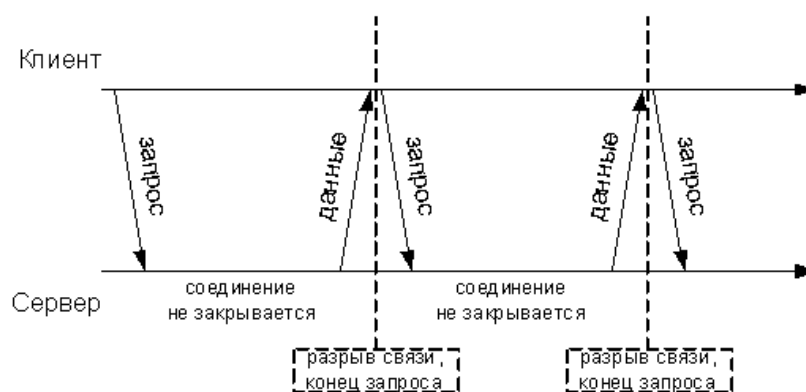


Рисунок 1.3.2 – Long-Poll соединение

2. РАЗРАБОТКА ПРИЛОЖЕНИЯ

2.1. Выбор средств разработки

Чтобы начать писать скрипт чат-бота, необходимо выбрать, язык программирования, способ взаимодействия с VK API и другие вспомогательные средства.

Для данной работы было выбран язык JavaScript, платформа Node.js, Bots Long Poll API, npm и MySQL.

JavaScript – это легковесный, интерпретируемый, объектно-ориентированный язык. Наиболее широкое применение находит как язык сценариев веб-страниц, но также используется и в других программных продуктах, например, node.js или Apache CouchDB. [9]

Node.js – это программная платформа, основанная на движке V8 (транслирующем JavaScript в машинный код), превращающая JavaScript из узкоспециализированного языка в язык общего назначения. Node.js добавляет возможность JavaScript взаимодействовать с устройствами ввода-вывода через свой API (написанный на C++), подключать другие внешние библиотеки, написанные на разных языках, обеспечивая вызовы к ним из JavaScript-кода. Node.js применяется преимущественно на сервере, выполняя роль веб-сервера, но есть возможность разрабатывать на Node.js и десктопные оконные приложения (при помощи NW.js, AppJS или Electron для Linux, Windows и macOS) и даже программировать микроконтроллеры (например, tessel и espruino). В основе Node.js лежит событийно-ориентированное и асинхронное программирование с неблокирующим вводом/выводом. [7]

MySQL – это быстрая, надежная, открыто распространяемая СУБД. MySQL, как и многие другие СУБД, функционирует по модели "клиент/сервер". Под этим подразумевается сетевая архитектура, в которой компьютеры играют роли клиентов, либо серверов. На рисунке 2.1.1 изображена схема передачи информации между компьютером клиента и жестким диском сервера.

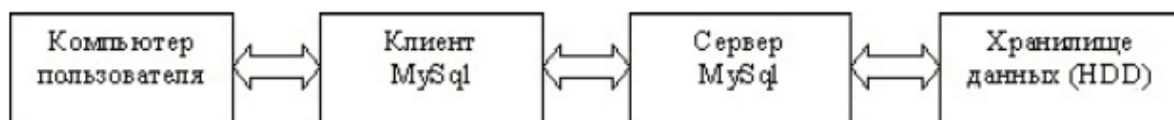


Рисунок 2.1.1 – Схема передачи данных в архитектуре "клиент/сервер"

СУБД управляет одной или несколькими базами данных. База данных представляет собой совокупность информации, организованной в виде множеств. Каждое множество содержит записи унифицированного вида. Сами записи состоят из полей. Обычно множества называют таблицами, а записи — строками таблиц. Такова логическая модель данных. На жестком диске вся база

данных может находиться в одном файле. В MySQL для каждой базы данных создаётся отдельный каталог, а каждой таблице соответствуют три файла. [8]

Npm – это менеджер пакетов для Node.js, представляет собой библиотеки, которые создаются сообществами программистов. В них можно найти пакеты, которые упростят написание скриптов и баз данных, что позволит проработать логику и функционал программы быстрее и более четко.

Является крупнейший в мире реестр программного обеспечения. Разработчики с открытым исходным кодом со всех континентов используют npm для обмена и заимствования пакетов, а многие организации также используют npm для управления частной разработкой.

Npm состоит из трех отдельных компонентов:

- Веб-сайт. Используется для поиска пакетов
- Интерфейс командной строки. Работа с терминала. Помогает разработчикам взаимодействовать с npm
- Реестр. Является большая общественная база данных программного обеспечения JavaScript и метаданных вокруг него. [10]

Для реализации задачи нужны три пакета:

- 1) Moment – позволяет работать и манипулировать с датами и временем.
- 2) Node-vk-bot-api – основной пакет, на котором строится весь функционал программы
- 3) Mysql2 – помогает работать с базами данных, используя платформу Node.js.

2.2. Составление дерева сценариев

Определившись с выбором средств разработки, надо продумать, как будет работать чат-бот. Чтобы не расписывать этапы виде списка, проще всего изобразить дерево сценариев, которые будет отражать его логику.

Первое и самое главное с чего начинается чат-бот – это меню, которое будет являться корневым узлом дерева сценариев.

В поставленной задаче нам необходимо отвечать на вопросы абитуриентов и студентов. Создадим эти два элемента дерева, которые будут исходить из меню.

Теперь, когда основные ветки созданы, можно прописать вопросы по разделам, на которые будет отвечать чат-бот. Для удобства обозначаем каждый уровень дерева одним цветом, чтобы легко ориентироваться в нем. Отметим направления стрелок, чтобы не появлялись тупиковые ветки. На дуге стрелки пишем букву N, если происходит ввод сообщения для поиска. Чтобы не загромождать дерево сценариев лишним текстом, пропишем только вопросы.

В соответствии с рисунком 2.2.1, можно легко реализовать логику чат-бота в скрипте, прописать все исходы события, которые могут произойти, и исключить те случаи, когда диалог может зайти в тупик. Данное дерево составлялась в сервисе Draw.io, который базируется на создание диаграмм и схем.

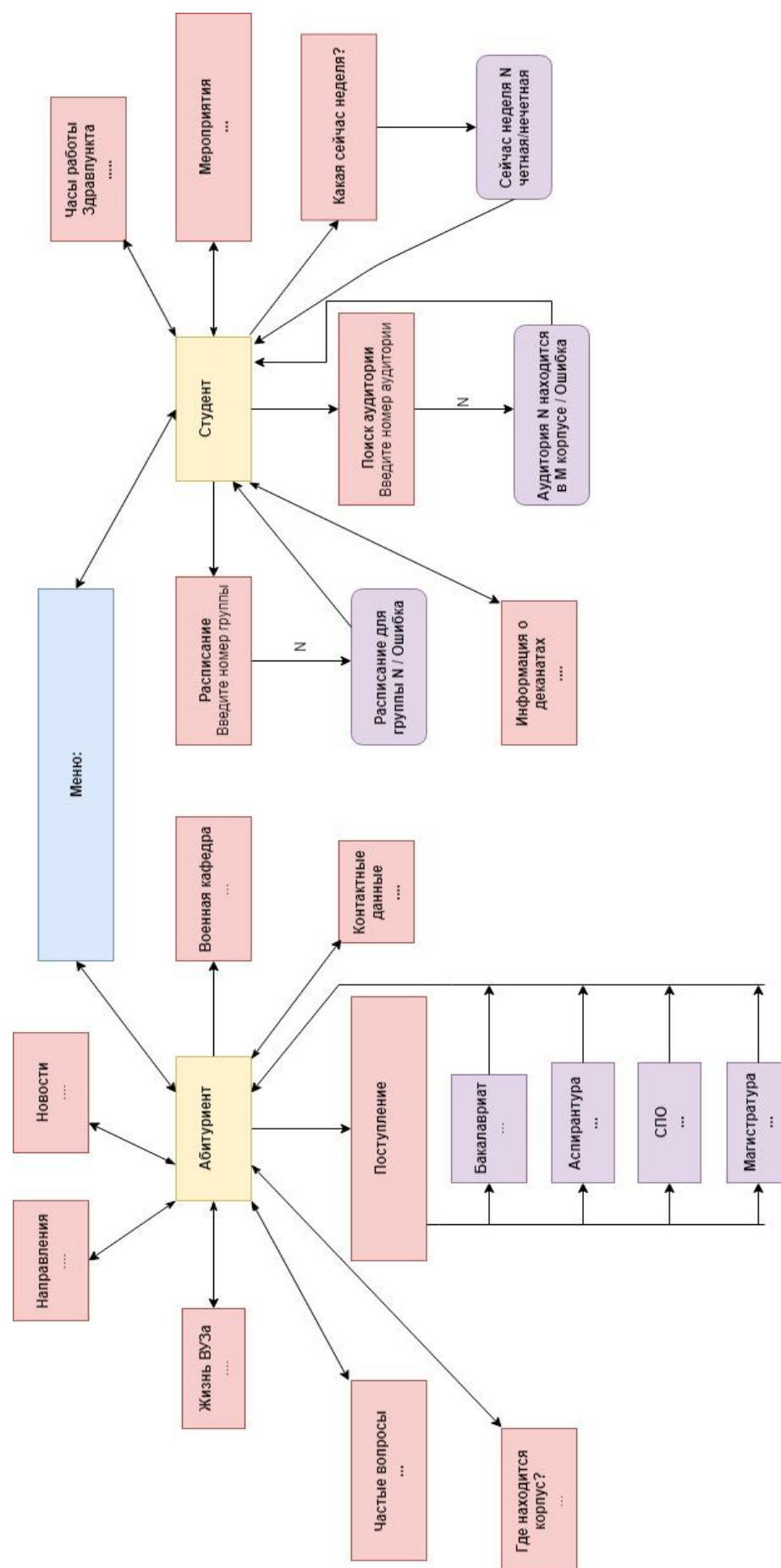


Рисунок 2.2.1 – Основное дерево сценариев

2.3 Разработка базы данных

Для реализации задачи нужно создать две базы данных – расписание и список аудиторий. Обозначим табличные значения, которые будут храниться в них.

Расписание:

- Номер группы
- Адрес хранения расписания на сервере «ВКонтакте»

Потребовалось вручную загрузить все документы в группу чат-бота и прописать адреса в базе данных. Способ такого хранения намного безопаснее и надежнее, чем передавать файлы, который будут храниться на локальном диске компьютера.

Аудитории:

- Номер аудитории
- Назначение
- Корпус

Многу была произведена работа по сбору и записи всех аудиторий, которые есть в нашем ВУЗе. Получился справочник, который отображает всю основную информацию для поиска.

Установим и настроим сервер MySQL на компьютер. На рисунке 2.3.1 пример в хода на сервер.

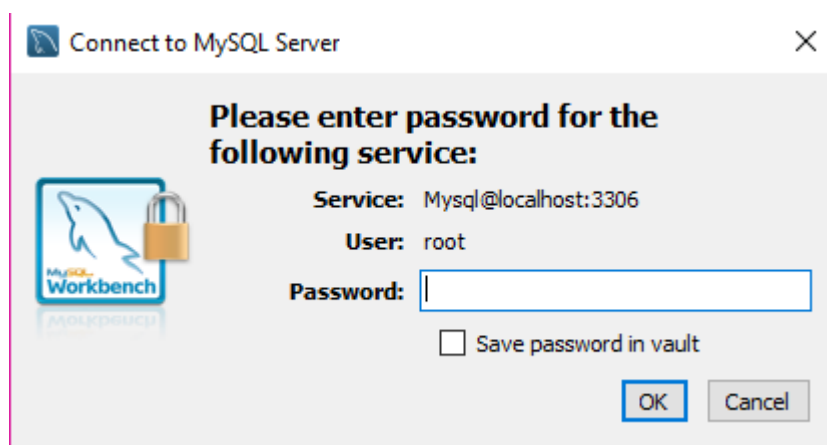


Рисунок 2.3.1 – Соединение с сервером

Чтобы начать работу с Node.js нужно скачать его с официального сайта. Все действия выполняем в командной строке компьютера. Также при установке, по умолчанию устанавливается npm. Пример сложения чисел с помощью Node.js на рисунке 2.3.2.

```
PS E:\Diplom\database> node
> 2+2
4
> 3+0
3
>
```

Рисунок 2.3.2 – Сложение чисел с помощью Node.js

В начале разработки мы не всегда можем предположить сколько пакетов npm мы будем использовать для нашего проекта, и поэтому необходимо создать файл `package.json`, который будет содержать информацию о этих пакетах и зависимости, которыми они обладают. Это позволит переносить или выставлять в интернет проект без используемых пакетов, что сократит объем передаваемой информации. На рисунке 2.3.3 показано инициализация `package.json`. С помощью опции `--yes` заполняем файл данными по умолчанию.

```
PS E:\Diplom\database> npm init --yes
Wrote to E:\Diplom\database\package.json:

{
  "name": "database",
  "version": "1.0.0",
  "description": "",
  "main": "mydb.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

Рисунок 2.3.3 – Инициализация `package.json`

Для работы с базой данных MySQL проще всего использовать пакет `npm mysql2`. Пример записи и установки на рисунке 2.3.4

```
PS E:\Diplom\database> npm install --save mysql2
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN database@1.0.0 No description
npm WARN database@1.0.0 No repository field.

+ mysql2@1.6.5
added 13 packages from 19 contributors and audited 16 packages in 4.631s
found 0 vulnerabilities
```

Рисунок 2.3.4 – Установка и запись пакета `mysql2`

Теперь создадим соединение с сервером MySQL. Для этого рассмотрим листинг 2.3.1.

Листинг 2.3.1 – Соединение с сервером через Node.js

```
const mysql = require("mysql2");

const connection = mysql.createConnection({
  host: "localhost",
  user: "root",
  database: "dbtable",
  password: "mIY6F}IS6h|wa$v"
});
```

Разберем параметры из листинга 2.3.1:

- host: хост, на котором находится сервер MySQL. По умолчанию выставляется значение "localhost"
- user: имя пользователя MySQL, которое использует подключение к серверу. В нашем случае "root "
- database: имя базы данных, которому хотим подключиться. На сервере ее назвали "dbtable". Пример создания в листинге 2.3.2. Если этот параметр не прописан, то идет подключение ко всему серверу
- password: пароль пользователя MySQL. Его мы записали при настройке сервера

Листинг 2.3.2 – Пример создания базы данных

```
const mysql = require("mysql2");

const connection = mysql.createConnection({
  host: "localhost",
  user: "root",
  password: "mIY6F}IS6h|wa$v"
});

connection.query("CREATE DATABASE dbtable",
  function(err, results) {
    if(err) console.log(err);
    else console.log("База данных создана");
  });

connection.end();
```

Когда создана база данных, необходимо сделать две таблицы, где будут храниться данные по расписанию и аудиториям. Пример создания для аудиторий в листинге 2.3.3. Аналогичным образом будет создана таблица для расписания, но иметь другие названия, названия и типы переменных.

Листинг 2.3.3 – Пример создания таблицы для аудиторий

```
const mysql = require("mysql2");

const connection = mysql.createConnection({
  host: "localhost",
  user: "root",
  password: "mIY6F}IS6h|wa$v"
});

const sql = `create table if not exists auditorium(
  number int not null,
  purpose varchar(255),
  corps varchar(255) not null
)`;

connection.query(sql, function(err, results) {
  if(err) console.log(err);
  else console.log("Таблица создана");
});

connection.end();
```

Заполним созданные таблицы значениями. Пример добавления значения в листинге 2.3.4.

Листинг 2.3.4 – Добавление значения в таблицу аудиторий

```
const mysql = require("mysql2");

const connection = mysql.createConnection({
  host: "localhost",
  user: "root",
  password: "mIY6F}IS6h|wa$v"
});

const users = [
  ["415", "деканат факультета заочного обучения", "Корпус №2 (заочка и дистанционка)"]
];

const sql = `INSERT INTO auditorium(number, purpose, corps) VALUES ?`;

connection.query(sql, [users], function(err, results) {
  if(err) console.log(err);
  console.log(results);
});
```


В результате в MySQL таблица аудиторий должна выглядеть согласно рисунку 2.3.5.

number	purpose	corps
415	деканат факультета заочного обучения	Корпус №2 (заочка и дистанционка)
414	факультет заочного обучения	Корпус №2 (заочка и дистанционка)
413	факультет заочного обучения	Корпус №2 (заочка и дистанционка)
412	факультет заочного обучения	Корпус №2 (заочка и дистанционка)
411	факультет заочного обучения (аудитория)	Корпус №2 (заочка и дистанционка)
403	научно-образовательный центр для наукоё...	Корпус №2 (заочка и дистанционка)
403	факультет заочного обучения (лекционка)	Корпус №2 (заочка и дистанционка)
404	научно-образовательный центр для наукоё...	Корпус №2 (заочка и дистанционка)
404	факультет заочного обучения (аудитория)	Корпус №2 (заочка и дистанционка)
402	факультет заочного обучения (аудитория)	Корпус №2 (заочка и дистанционка)
401	факультет заочного обучения (аудитория)	Корпус №2 (заочка и дистанционка)

Рисунок 2.3.5 – Таблица аудиторий

В данной базе данных нет связей между таблицами. На рисунке 2.3.6 предоставлена схема базы данных.

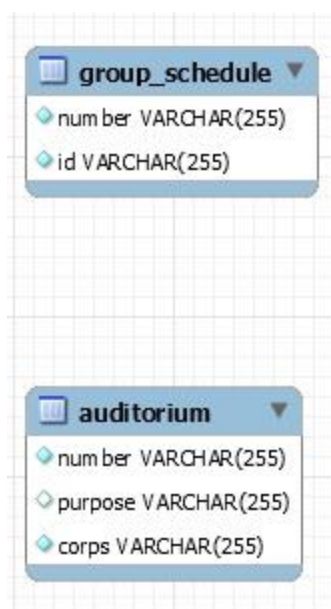
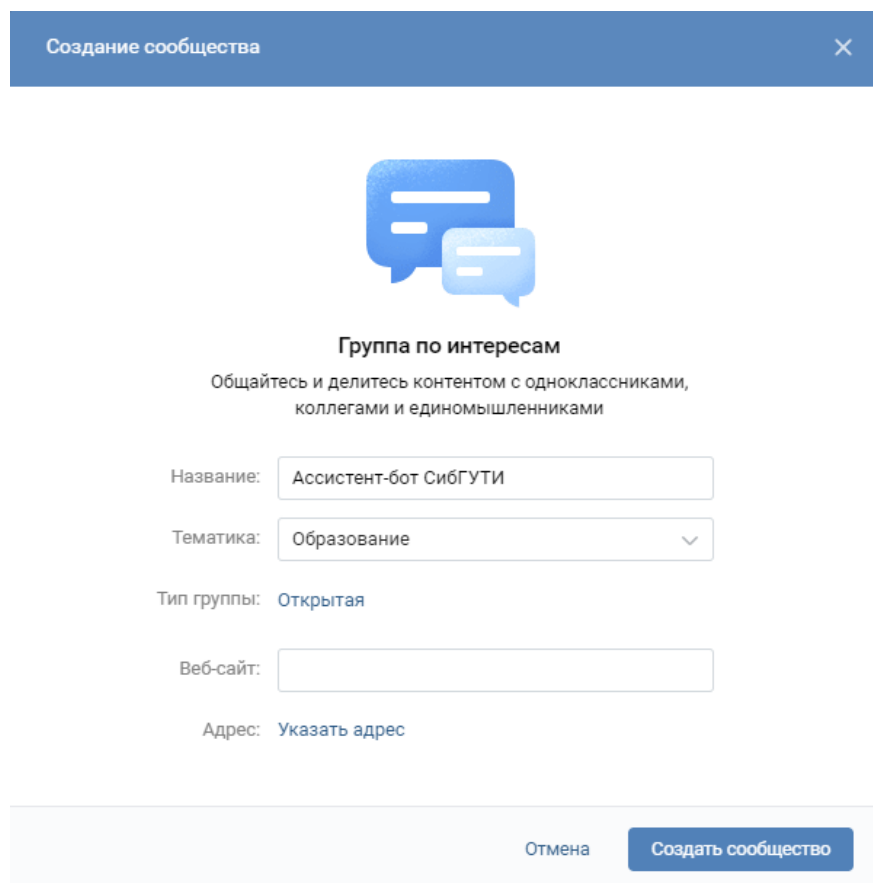


Рисунок 2.3.6 – Схема базы данных

Когда заполнение двух таблиц закончено, можно приступить к написанию скрипта бота. Введённые значения нам еще понадобятся в программной реализации.

2.4 Программная реализация

Прежде чем начать писать скрипт чат-бота, необходимо быть пользователем социальной сети «ВКонтакте» и создать группу, где будет располагаться он будет располагаться. Когда мы создали аккаунт, переходим в раздел «Группы» и нажимаем на кнопку «Создать сообщество». Выбираем соответствующий тип группы и прописываем поля. Пример на рисунке 2.4.1.



The screenshot shows the 'Создание сообщества' (Create Community) form in VKontakte. At the top is a blue header bar with the title and a close button. Below it is a blue speech bubble icon. The form is titled 'Группа по интересам' (Group by interests) with a subtitle 'Общайтесь и делитесь контентом с одноклассниками, коллегами и единомышленниками' (Communicate and share content with classmates, colleagues, and like-minded people). The form contains several fields: 'Название:' (Name) with the value 'Ассистент-бот СибГУТИ', 'Тематика:' (Topic) with a dropdown menu showing 'Образование' (Education), 'Тип группы:' (Group type) with the selected option 'Открытая' (Open), 'Веб-сайт:' (Website) with an empty field, and 'Адрес:' (Address) with a link 'Указать адрес' (Specify address). At the bottom are two buttons: 'Отмена' (Cancel) and 'Создать сообщество' (Create community).

Рисунок 2.4.1 – Создание группы

Созданную группу оформляем – добавляем картинку, пишем статус и заполняем основную информацию. Главное писать все по тематике, которая будет соответствовать чат-боту. Пример на рисунке 2.4.2.

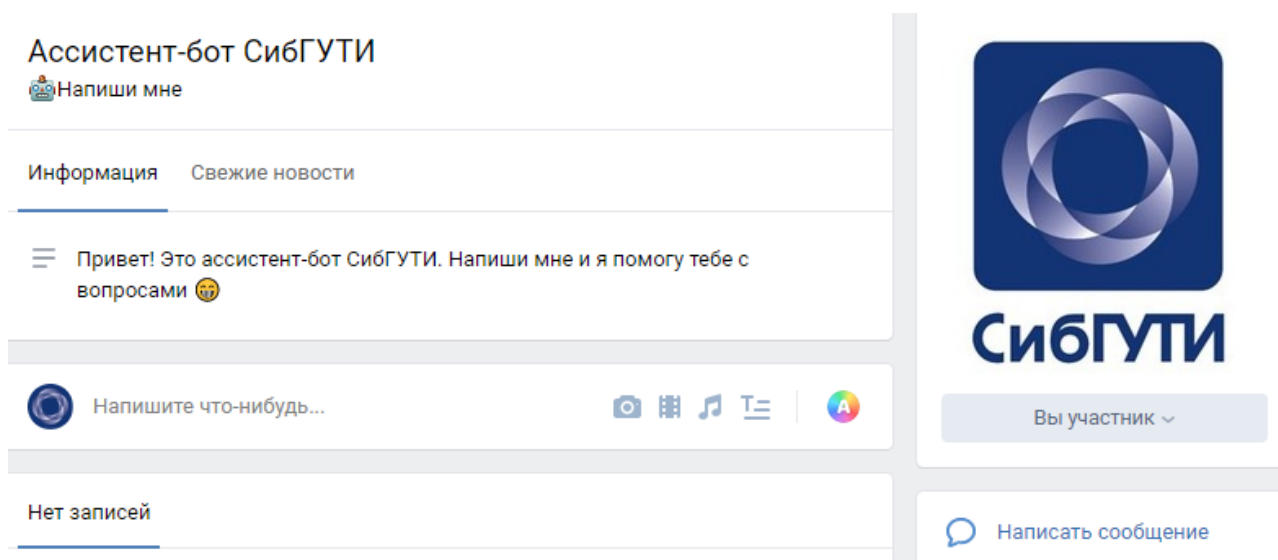


Рисунок 2.4.2 – Оформление группы

Для создания чат-бота нам потребуется создать токен. Его можно получить, совершив следующие действия:

- 1) Заходим в раздел «Управление»
- 2) Переходим в «Работа с API»
- 3) Нажимаем кнопку «Создать ключ»
- 4) Во всплывающем окне проставить везде галочке
- 5) Нажать кнопку «Создать»
- 6) К вашему аккаунту должен быть привязан телефон. Через него произойдёт подтверждение пользователя
- 7) Теперь мы получили токен. Пример результата на рисунке 2.4.3

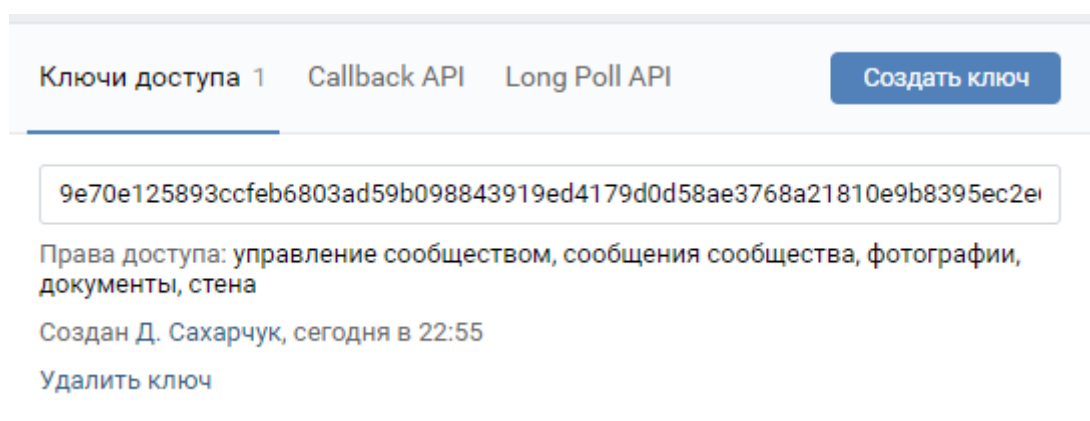


Рисунок 2.4.3 – Получение токена

Чтобы чат-бот мог присылать пользователю клавиатуру, совершить следующие действия:

- 1) Заходим в раздел «Управление»
- 2) Переходим в «Сообщения»
- 3) Поменять значение «Сообщения сообщества» на «Включены»

- 4) Нажать на галочку в параметре «Виджет сообщений»
- 5) Можно прописать следующие пункты:
 - а) «Приветствие»
 - б) «Краткая информация»
 - в) «Статус не в сети»
- 6) Нажимать кнопку «Сохранить»
- 7) Переходим в раздел «Настройки для бота»
- 8) Поменять значение «Возможности ботов» на «Включены»
- 9) Проставить галочку на параметре «Добавить кнопку «Начать»» и нажимать кнопку «Сохранить»

Результат действий продемонстрирован на рисунке 2.4.4.

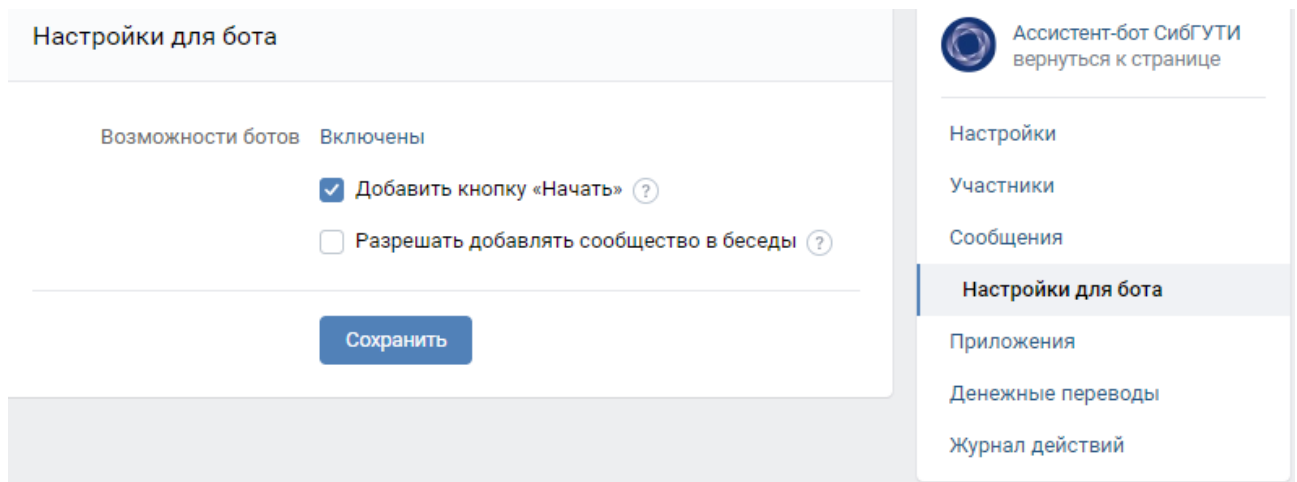


Рисунок 2.4.4 – Настройка группы

Пропишем, на какие события Long Poll API. Наш чат-бот будет реагировать на входящие сообщения. Выполняем следующие действия:

- 1) Заходим в раздел «Управление»
- 2) Переходим в «Работа с API»
- 3) Заходим в раздел «Long Poll API»
- 4) Переходим во вкладку «Тип событий»
- 5) Отмечаем галочкой пункт «Входящее сообщение»

Результат представлен на рисунке 2.4.5.

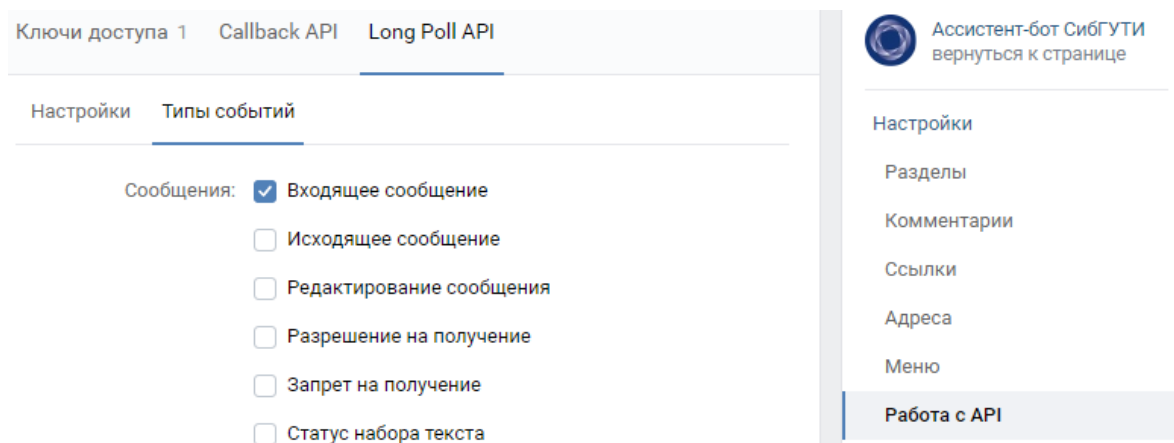


Рисунок 2.4.5 – Настройка Long Poll API

После настройки группы, нужно инициализировать `package.json`, скачать и записать нужные нам пакеты в папку. Последовательность выполнения команд:

- 1) `npm init --yes`
- 2) `npm install --save mysql2 node-vk-bot-api moment`

Теперь переходим к написанию скрипта. Назовём наш файл `bot` с расширением `js`. Пропишем переменные, которые обращаются к пакетам и методам из них. Также создадим и файл `functions` с расширением `js`, в котором будут храниться часто используемые функции, и пропишем к нему обращение. Пример в листинге 2.4.1.

Листинг 2.4.1 – Обращение к пакетам и методам

```
const VkBot = require('node-vk-bot-api/lib');
const Session = require('node-vk-bot-api/lib/session');
const Stage = require('node-vk-bot-api/lib/stage');
const Scene = require('node-vk-bot-api/lib/scene');
const moment = require('moment');
const mysql = require('mysql2');
const whatsdo = require('./functions');
```

Далее прописываем соединение с базой данных MySQL и переменную, которая будет общаться с сервером «ВКонтакте» методом `Bots Long Poll API`. В эту переменную передаем токен группы. И пропишем функцию, которая будет отправлять запросы на сервер «ВКонтакте». Пример в листинге 2.4.2.

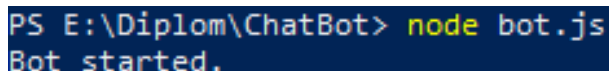
Листинг 2.4.2 – Подключение к базе данных и серверу «ВКонтакте»

```
const connection = mysql.createConnection({
  host: "localhost",
  user: "root",
  database: "dbtable",
  password: "mIY6F}IS6h|wa$V"
});

const bot = new
VkBot('12617858a5c31f15050b6968efb86634a1358926bc9f9bfa5db1220abcb
f1402593f735b9af067f76865e');

bot.startPolling(() => {
  console.log('Bot started.');//вывод на консоль
});
```

Проверим правильность выполненных действий. Запустим скрипт в командной строке. Результат и выполнение должен быть, как на рисунке 2.4.6.



```
PS E:\Diplom\ChatBot> node bot.js
Bot started.
```

Рисунок 2.4.6 – Результат выполнения первого запуска скрипта

Далее в скрипте будем использовать методы из пакета node-vk-bot-api. Подробнее ознакомится с ними можно по адресу: <https://www.npmjs.com/package/node-vk-bot-api>

В пункте 2.2 прописали дерево сценариев диалога. Его и возьмем за основу логики чат-бота. И в файле пропишем functions начнем прописывать основные моменты из дерева. Используем метод Markup, который будет отправлять пользователю клавиатуру. Рассмотрим листинг 2.4.3, в котором прописана создание клавиатуры меню чат-бота.

Листинг 2.4.3 – Клавиатура меню

```
const Markup = require('node-vk-bot-api/lib/markup');

module.exports = {
  start: function (ctx) {
    ctx.reply('●Меню●', null, Markup
      .keyboard([
        Markup.button('🎓Студент', 'primary'),
        Markup.button('📄Абитуриент', 'positive'),
      ])
      .oneTime());
  }
};
```

Создаем переменную Markup, которая будет обращаться к методу markup. Прописываем module.exports, который создает внешний доступ к написанным нами модулям. Модуль меню назовём start. Существуют всего четыре значения цвета кнопки клавиатуры:

- 1) primary – синий
- 2) secondary – белый
- 3) negative – красный
- 4) positive – зеленый

Вспомогательный метод “oneTime()” нужен для того, чтобы создавать одноразовую клавиатуру. При отправке сообщения, где нет клавиатуры, последняя отправленная не сохранится у пользователя.

Не забываем добавлять в текст отправленных сообщений и клавиатуры тематические эмоджи. Они придают тексту индивидуальности и помогают интуитивно сориентировать в тексте.

Эмоджи – язык идеограмм и смайликов, используемый в электронных сообщениях и веб-страницах. Этот графический язык, где вместо слов используются сочетания картинок. [5]

Синим цветом будем обозначать кнопку «Студент» и вопросы студентов. Зеленым «Абитуриент» и их вопросы. Кнопку «Назад» обозначим красным цветом. Это визуально придаст кнопкам значимости и логику.

Аналогично пропишем клавиатуру для студентов, абитуриентов и поступающих. Обращаться к созданным модулям в основном скрипте будем через переменную whatsdo. Пропишем команду, которая будет реагировать на любое содержание сообщения и отправлять клавиатуру меню. Пример в листинге 2.4.4, а результат изображен на рисунке 2.4.7.

Листинг 2.4.4 – Реакция чат-бота на любые сообщения

```
bot.on((ctx) => {
  ctx.reply(whatsdo.start(ctx));
});
```

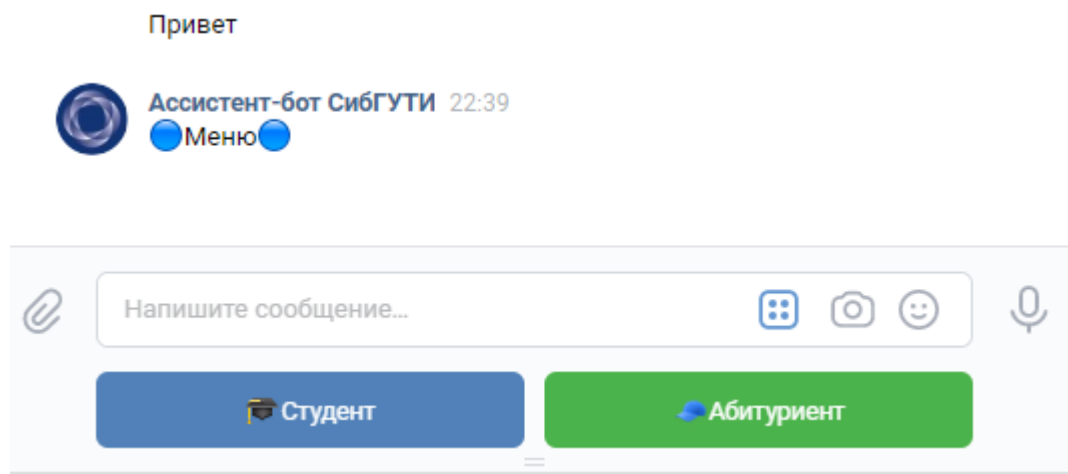


Рисунок 2.4.7 – Меню чат-бота

Чат-бот реагирует на триггеры или другими словами на определенные сообщения. Поэтому нужно писать листинг 2.4.4 после всех триггеров. Пример триггера для кнопки «Назад» в листинге 2.4.5.

Листинг 2.4.5 – Реализация кнопки «Назад»

```
bot.command('↩️Назад', (ctx) => {  
    ctx.reply(whatsdo.start(ctx));  
});
```

Аналогичным методом прописываются остальные триггеры чат-бота. Но есть три триггера, которые имеют другую реализацию. Рассмотрим сначала триггер на вопрос «Какая сейчас неделя?». Его реализация представлена в листинге 2.4.6.

Листинг 2.4.6 – Реализация вопроса «Какая сейчас неделя?»

```
bot.command('🗉Какая сейчас неделя?', (ctx) => {
    week_start = moment("11-02-2019", "DD-MM-YYYY").format("w");
    week_now = moment().format("w");
    parity = 2;
    if ((week_start % 2 == 0 && parity == 2) || (week_start % 2 !=
0 && parity == 1)) {
        if (week_now % 2 == 0) {
            ctx.reply('Сейчас 🗉неделя (четная)');
        } else {
            ctx.reply('Сейчас 🗑неделя (нечетная)');
        }
    } else {
        if (week_now % 2 == 0) {
            ctx.reply('Сейчас 🗑неделя (нечетная)');
        } else {
            ctx.reply('Сейчас 🗉неделя (четная)');
        }
    }
    ctx.reply(whatsdo.student(ctx));
});
```

Рассмотрим метод и переменные:

- `format("w")`: возвращает номер недели в году в формате числа
- `week_start`: неделя начала учебы. При изменении можно изменить значение даты "11-02-2019" на любое другое в формате "DD-MM-YYYY".
- `week_now`: текущая неделя
- `parity`: выставляется четность или нечетность недели начала учебы (четная – 2, нечетная – 1). При изменении поменять цифру

Для реализации мы воспользовались пакетом `moment`. Результат скрипта представлен на рисунке 2.4.8.

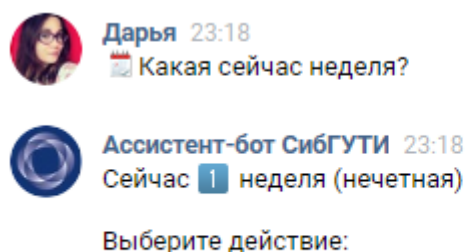


Рисунок 2.4.8 – Результат триггера «Какая сейчас неделя?»

Теперь рассмотрим триггеры «Расписание» и «Поиск аудитории». Для этого нам понадобятся методы:

- Scene: создаёт сценарий действий
- Session: запоминает переменные данной сессии
- Stage: запоминает стадии сцены

Реализация ввода значений для поиска расписания подробно представлен в листинге 2.4.7.

Листинг 2.4.7 – Реализация поиска расписания

```
const session = new Session();
const scene = new Scene('input',
  (ctx) => {
    ctx.session.text = ""
    ctx.session.text = ctx.message.text;
    ctx.scene.next();
  },
  (ctx) => {
    ctx.scene.leave();
    if(ctx.session.text == "📅Расписание") {
      const sql = `SELECT * FROM group_schedule WHERE number=?`;
      const filter = ctx.message.text.toUpperCase();
      connection.query(sql, filter, function(err, results) {
        if(err) console.log(err);
        const data = results;
        output = ""
        for(let i=0; i < data.length; i++) {
          output = output + data[i].id;
        }
        if (output != ""){
          bot.sendMessage(ctx.message.peer_id, 'Ваше
Расписание!', output)
          ctx.reply(whatsdo.student(ctx));
        }
        else{
          ctx.reply("Вы ввели группу не верно или такой группы
нет")
          ctx.reply(whatsdo.student(ctx));
        }
      });
    }
  });
const stage = new Stage(scene);
bot.use(session.middleware());
bot.use(stage.middleware());
```

В переменную `ctx.session.text` записываем какой триггер обратился к сценарию. Это необходимо для того, чтобы понять к какой базе данных должны обращаться, чтобы получить результат.

Далее пользователь должен ввести значение, которое он хочет найти. Его записываем в переменную `filter` в верхнем регистре. Выполняем поиск в базе

данных расписания (group_schedule) по табличному значению номера группы (number).

Записываем результат в переменную output. В ней будет находиться адрес, где на сервере «ВКонтакте» хранится расписание. Если она оказалась пустая, то выводим сообщение об ошибке, в ином случае отправляем документ. Результат работы приведен на рисунке 2.4.9.

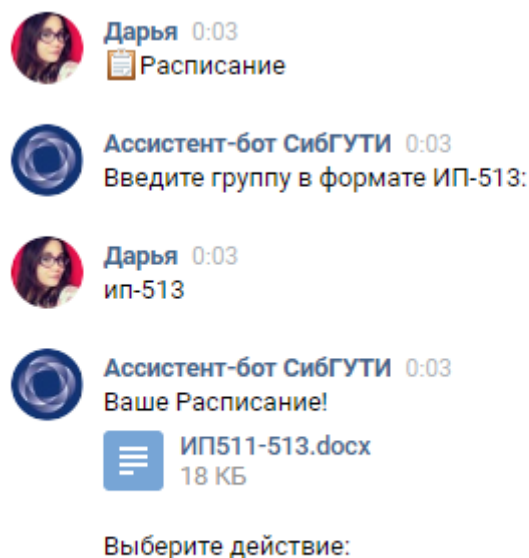


Рисунок 2.4.9 – Поиск расписания

Аналогичным образом прописываем скрипт для поиска аудитории. Пример результата на рисунке 2.4.10.

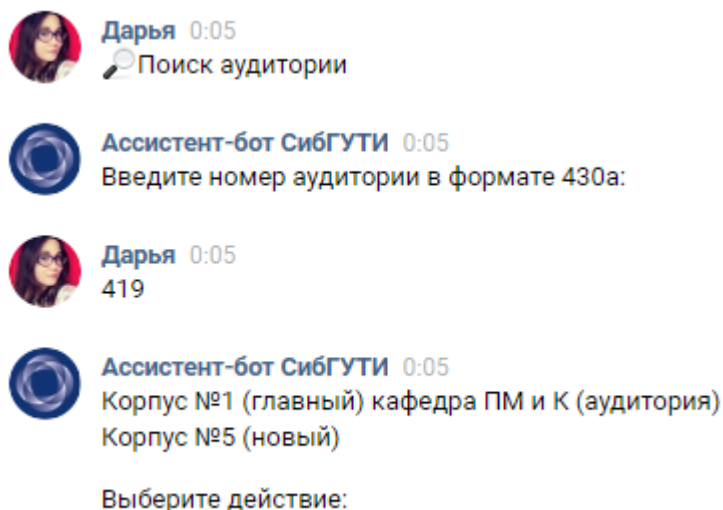


Рисунок 2.4.10 – Поиск аудитории

3. ТЕСТИРОВАНИЕ

После того, как чат-бот уже готов к работе и программная реализация закончена, нужно провести тесты, чтобы убедиться, что все триггеры работают корректно.

Для начала очистим всю историю сообщений с чат-ботом со стороны пользователя и со стороны группы. Теперь можно запустить данного помощника и начать его тестировать.

Перейдя в диалог с чат-ботом, мы должным увидеть кнопку «Начать». Это говорит о том, что мы правильно настроили группу. На рисунке 3.1 представлен результат начала диалога.



Рисунок 3.1 – Начало диалога

Если мы нажмем на кнопку «Начать», мы должно перейти в меню чат-бота. Это связано с тем, что у нас нет такого триггера «Начать», и чат-бот отправляет меню на незнакомые ему триггеры. Результат на рисунке 3.2.

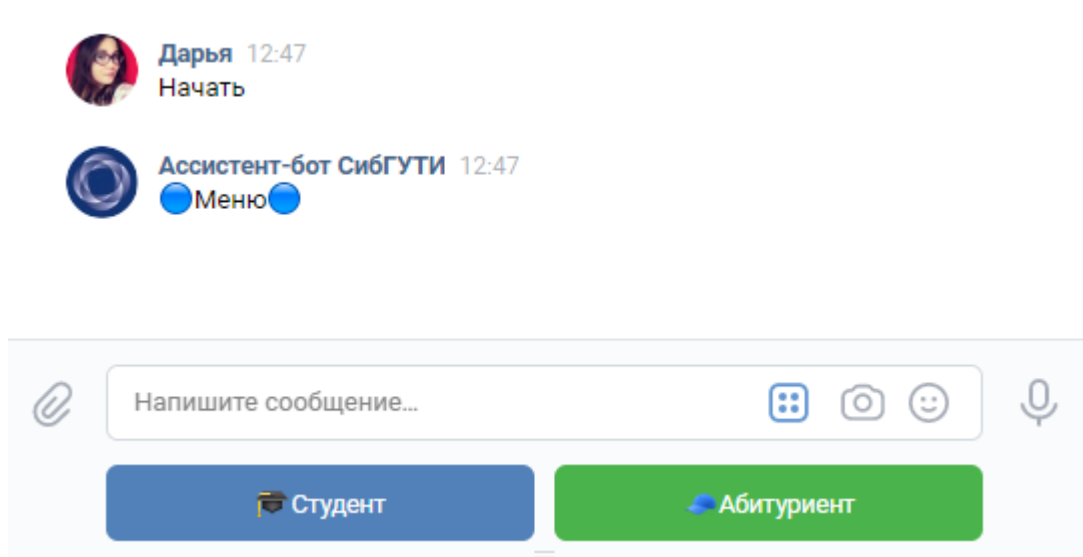


Рисунок 3.2 – Результат нажатия кнопки «Начать»

Теперь перейдем в раздел «Студент» и начнем его тестировать.

Когда мы нажмем на этот раздел, мы должны получить именно клавиатуру студента, без пересечения с другими клавиатурами. Результат предоставлен на рисунке 3.3.

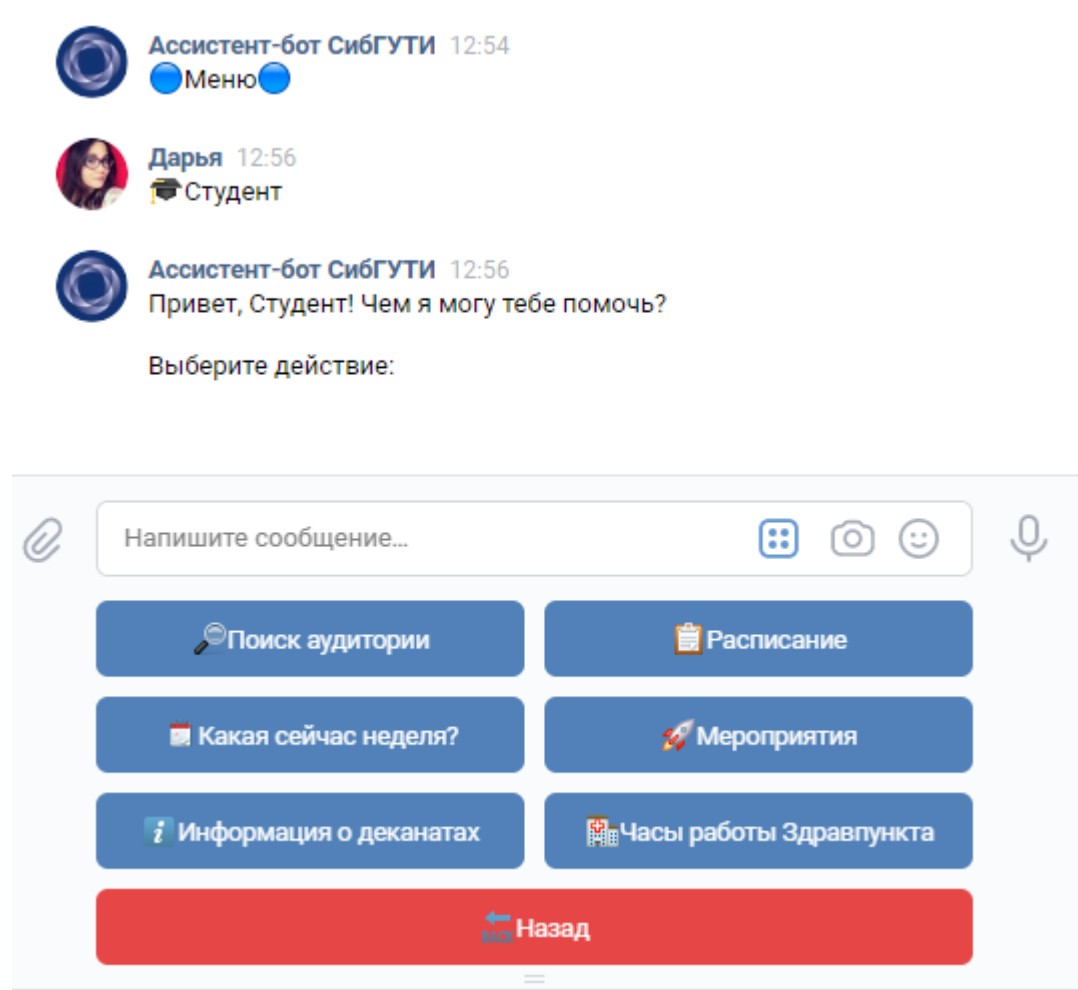


Рисунок 3.3 – Результат нажатия кнопки «Студент»

Проведём тестирование функции «Поиск аудитории». Для этого введём значение «333». Посмотрим, что записано в базе данных. Результат на рисунке 3.4.

number	purpose	corps
323	кафедра физики (лаборатория)	Корпус №5 (новый)
330	кафедра РТС (лаборатория)	Корпус №3 (старый)
331	зав кафедрой радиотехнических систем	Корпус №3 (старый)
332	кафедра РТС (лаборатория)	Корпус №3 (старый)
333		Корпус №3 (старый)
333	аудитория	Корпус №5 (новый)
335	кафедра Радиотехнических систем (препод...	Корпус №3 (старый)
335	кафедра ТЭЦ (лаборатория)	Корпус №5 (новый)

Рисунок 3.4 – Просмотр таблицы аудиторий

Чат-бот должен выдать следующий результат:

- Корпус №3 (старый)
- Корпус №5 (новый) аудитория

Проверим это в диалоге. Все совпадает. Результат выполнения предоставлен на рисунке 3.5.

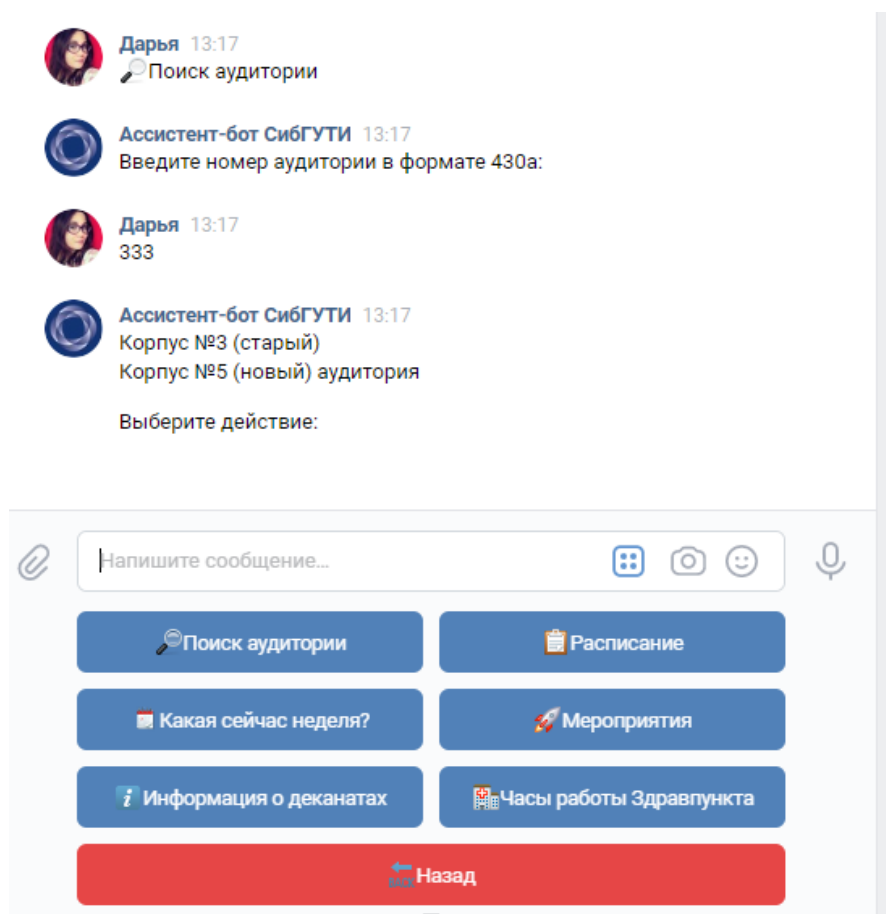


Рисунок 3.5 – Ввод значения «333»

Теперь введем произвольную строку «333абвгд». Чат-бот должен сообщить об ошибке. Это действительно так. Результат на рисунке 3.6.

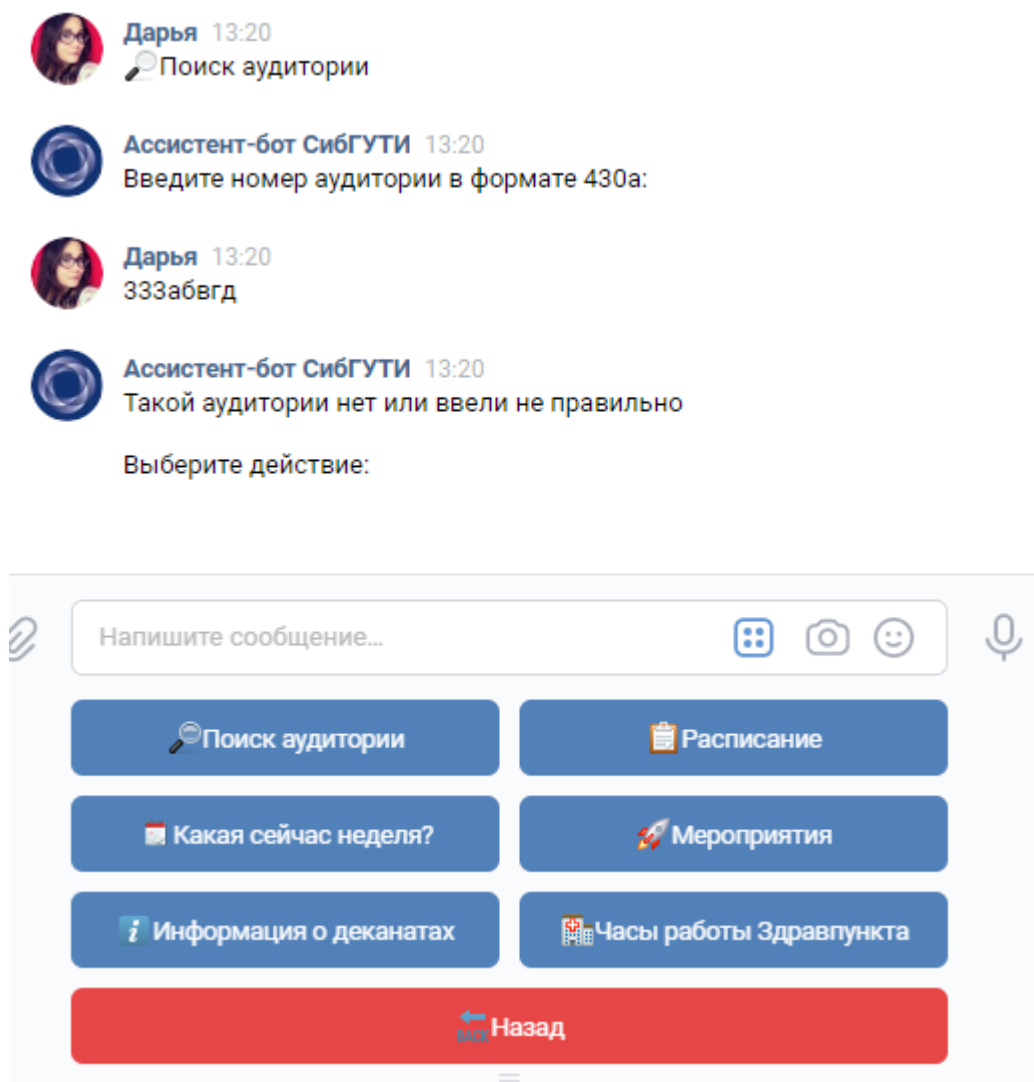


Рисунок 3.6 – Ввод неверного значения аудитории

Стоит также заметить, что после каждого ответа, чат-бота выводит верную клавиатуру пользователю.

Теперь перейдём к тестированию функции поиска расписания. Для этого нажмем на кнопку «Расписание». Чат-бот предложит ввести номер группы. Давайте найдем расписание для группы ИП-513. Посмотрим какой адрес на сервере «ВКонтакте» у этого документа. Для этого обратимся к базе данных. Результат на рисунке 3.7.

	number	id
	ИП-512	doc-170978780_498964126
▶	ИП-513	doc-170978780_498964126
	ИП-611	doc-170978780_498964179
	ИП-612	doc-170978780_498964179
	ИП-613	doc-170978780_498964179
	ИП-711	doc-170978780_498964271
	ИП-712	doc-170978780_498964271
	ИП-713	doc-170978780_498964271
	ИП-714	doc-170978780_498964342
	ИП-715	doc-170978780_498964342

Рисунок 3.7 – Просмотр таблицы расписаний

Если мы перейдем по этому адресу получим расписание для данной группы. Результат на рисунке 3.8.

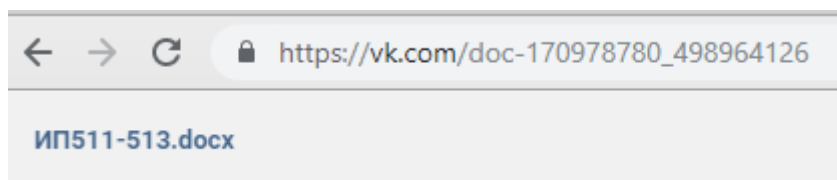


Рисунок 3.8 – Проверка расписания и значений из базы данных

Проверим, что ответит чат-бот на запрос. Все совпадает. Результат приведен на рисунке 3.9.

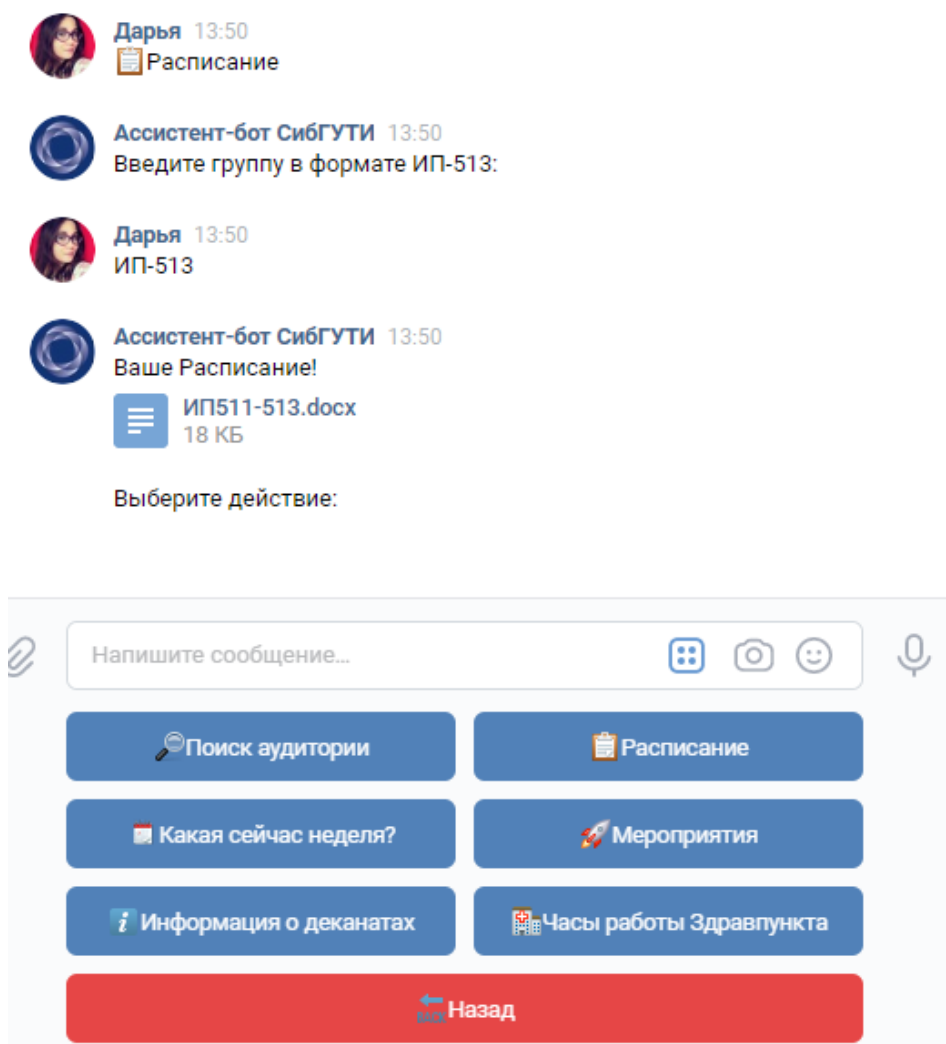


Рисунок 3.9 – Ввод значения «ИП-513»

Теперь введем произвольную строку «ИП-513абвгд». Чат-бот должен сообщить об ошибке. Это действительно так. Результат на рисунке 3.10.

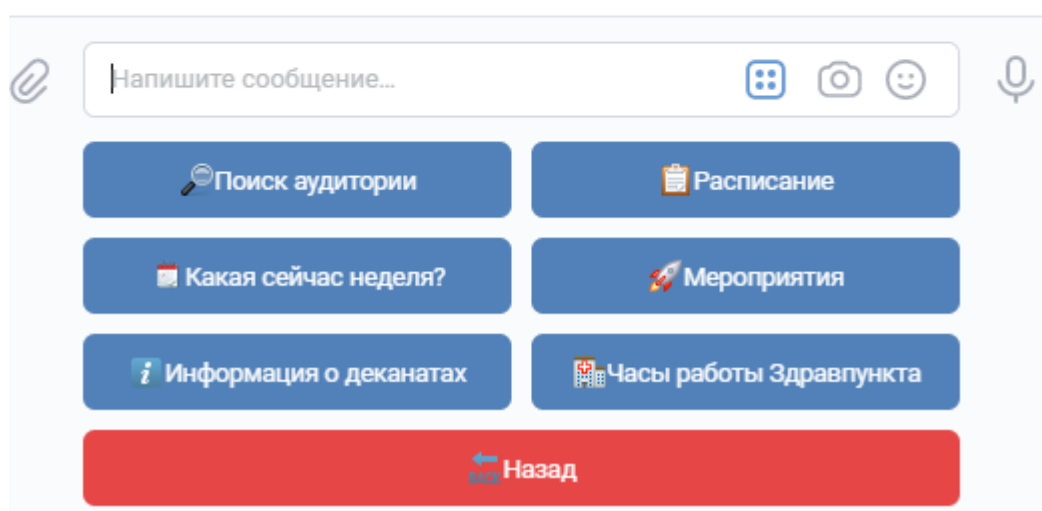
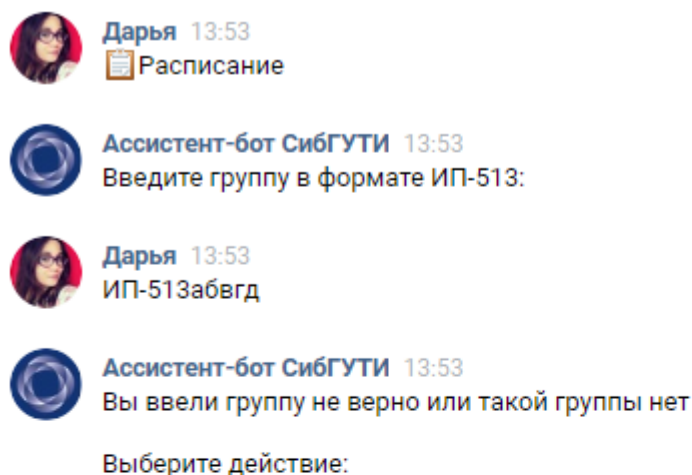


Рисунок 3.10 - Ввод неверного значения аудитории

Если теперь мы нажмем на кнопку «Назад», то чат-бот должен вернуть нас в меню. Это действительно так. Результат на рисунке 3.11.

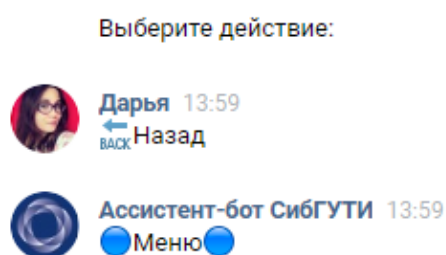


Рисунок 3.11 – Результат нажатия кнопки «Назад»

Теперь перейдем в раздел «Абитуриент» и начнем его тестировать.

Когда мы нажмем на этот раздел, мы должны получить именно клавиатуру абитуриента, без пересечения с другими клавиатурами. Результат предоставлен на рисунке 3.12.

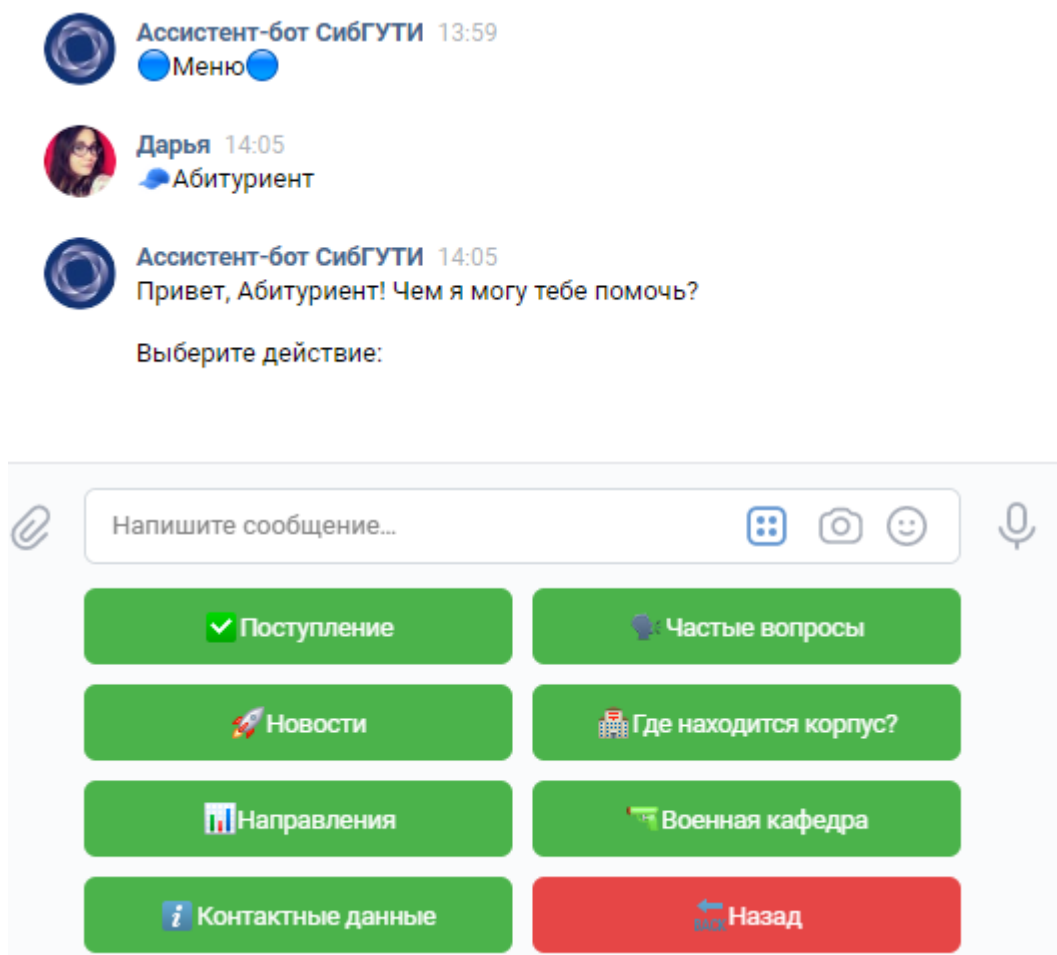


Рисунок 3.12 - Результат нажатия кнопки «Абитуриент»

Проверим функцию «Поступление». Чат-бот должен отправить клавиатуру в котором четыре значения:

- 1) Бакалавриат
- 2) Магистратура
- 3) Аспирантура
- 4) СПО

Это действительно так. Результат на рисунке 3.13.

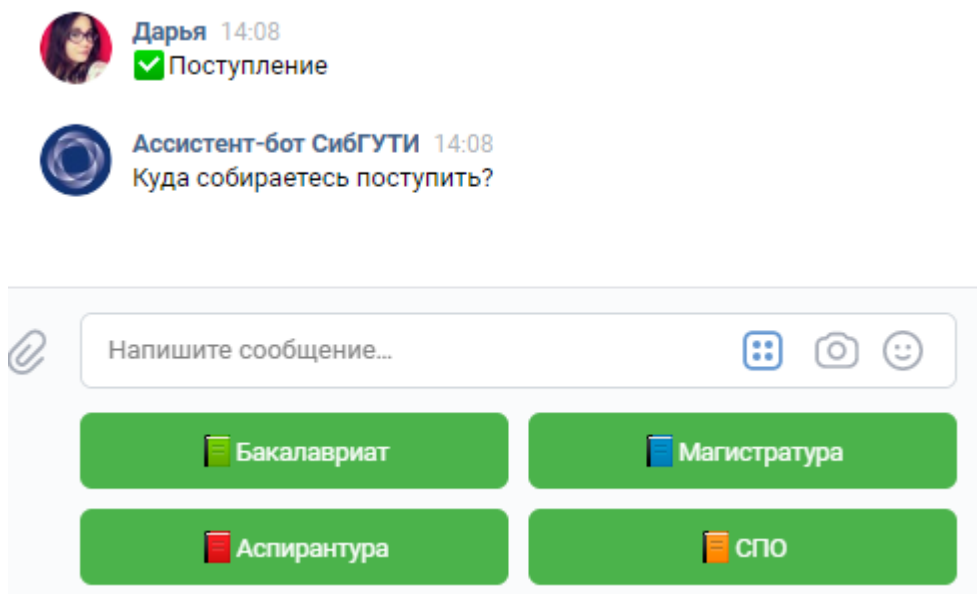


Рисунок 3.13 – Клавиатура для «Поступление»

В результате выполнения данных тестов, можно сделать вывод, что чат-бот работает корректно.

4. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Чтобы новому пользователю было проще ориентироваться в чат-боте, напомним руководство по его пользованию.

Лучше всего это руководство написать в виде статьи и закрепить в группе данного чат-бота.

Сделаем следующие действия:

- Напишем заголовок статьи
- Приведем небольшой девиз статьи
- Прикрепим логотип СибГУТИ

Результат выполнения на рисунке 4.1.

Ассистент-Бот СибГУТИ: Руководство Пользователя

| Делаем пользование Ассистент-бота проще!



Рисунок 4.1 – Начало руководства

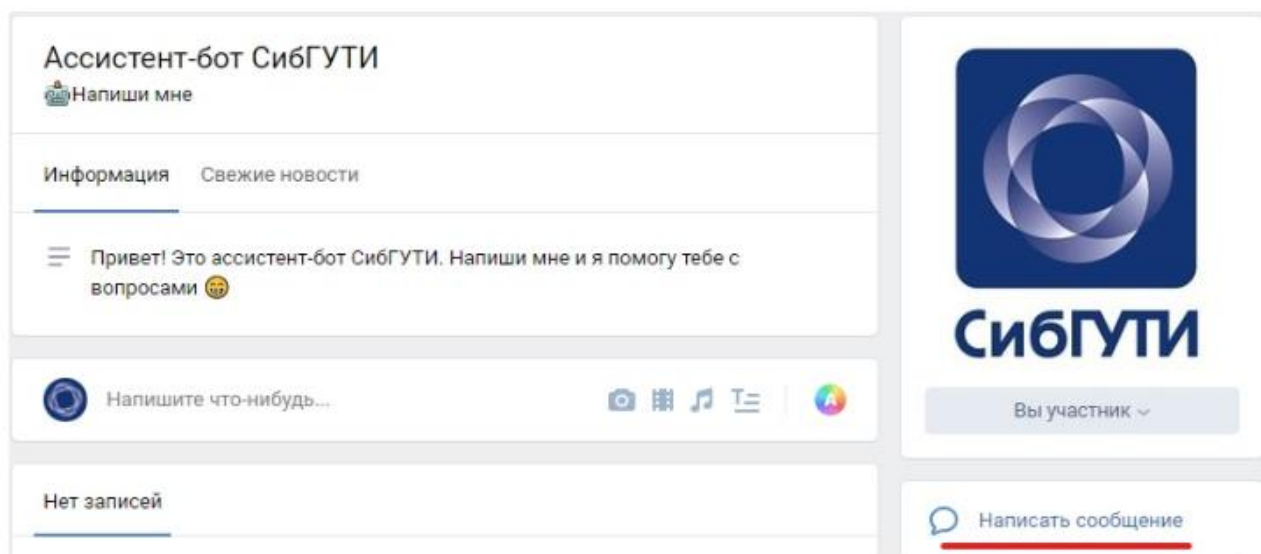
Теперь начнем писать раздел «Начало Работы с Ассистент-ботом». Обозначим и выделим заголовок. Пример на рисунке 4.2.

Начало Работы с Ассистент-ботом

Рисунок 4.2 – Заголовок «Начало Работы с Ассистент-ботом»

В этом разделе будет подробно расписано и показано, как пользователю начать диалог с чат-ботом. Небольшой отрывок из этой части представлен на рисунке 4.3.

Чтобы начать диалог, нужно нажать на кнопку «Написать сообщение» в группе этого сообщества.



В открывшемся окне нажмите на «Перейти к диалогу с сообществом»

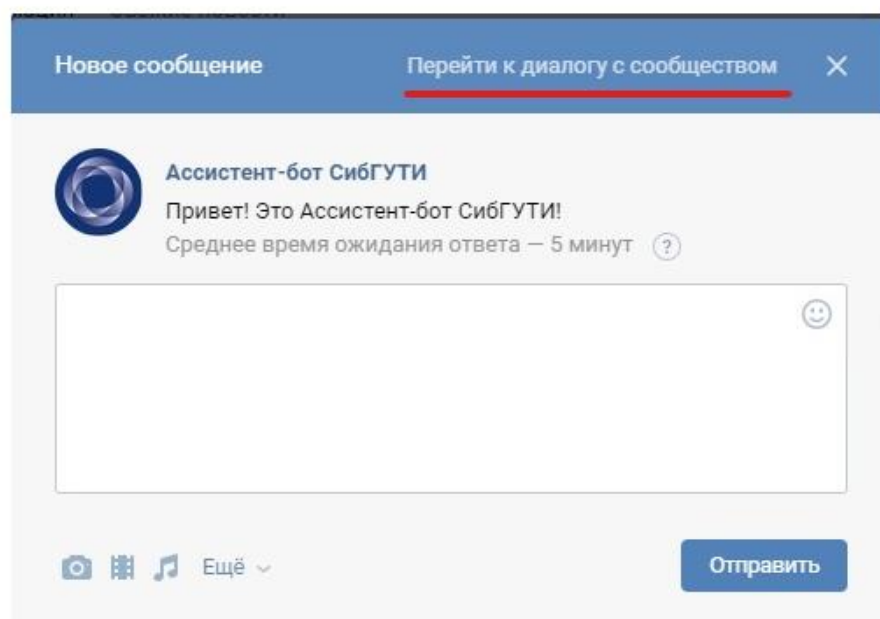


Рисунок 4.3 – Отрывок из раздела «Начало Работы с Ассистент-ботом»

После того, когда начался диалог с чат-ботом, пользователю необходимо понять, как работают разделы для студентов и абитуриентов. Для этого добавим в руководство два раздела «Студент» и «Абитуриент».

Для начала добавим и выделим заголовок «Студент». Пример представлен на рисунке 4.4

Студент

Рисунок 4.4 - Заголовок «Студент»

В этом разделе будет подробно расписано, как пользоваться функциями для поиска расписания и аудиторий, а также другие функции. Отрывок из этого раздела можно увидеть на рисунке 4.5.

Поиск аудитории

Нажмите на кнопку «Поиск аудитории». Введите номер аудитории. В ответ Вы получите, в каком корпусе находится аудитория, и какое она имеет назначение. Формат ввода: цифры или цифры и буква. Если номер аудитории, которую Вы ищете содержит букву, то ее стоит писать слитно с цифрами.

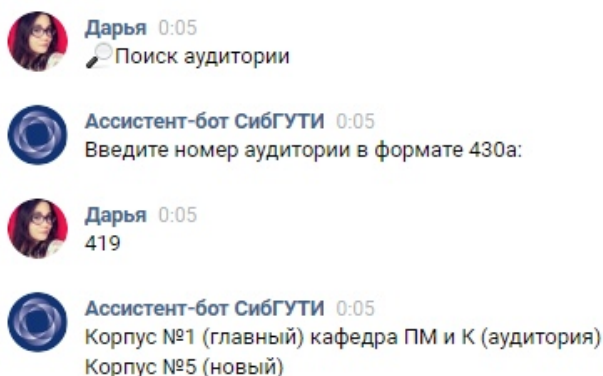


Рисунок 4.5 - Отрывок из раздела «Студент»

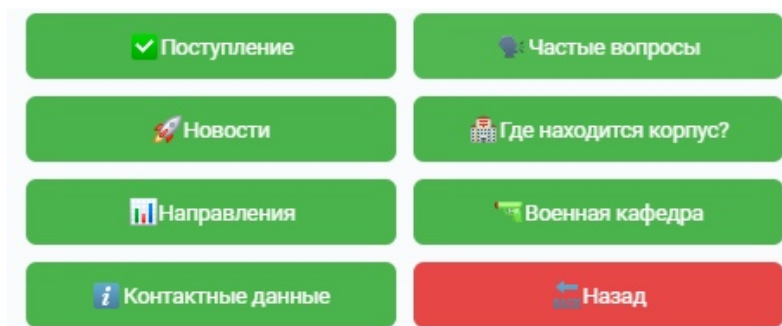
Теперь создадим раздел «Абитуриент». Для начала нужно добавить и выделить заголовок «Абитуриент». Пример на рисунке 4.6.

Абитуриент

Рисунок 4.6 - Заголовок «Абитуриент»

В этом разделе будет подробно расписано, как пользоваться функцией «Поступление», а также другие функции. Отрывок из этого раздела можно увидеть на рисунке 4.7.

Выберите раздел в меню «Абитуриент». Ассистент-бот предложит самые популярные вопросы, которые задают абитуриенты. Перейдя по ним вы можете получить ответ на них. После ответа Ассистент-бот возвращает Вас, ко всем вопросам абитуриента.



Поступление

Нажмите на кнопку «Поступление». Ассистент-бот предложит выбрать, куда Вы собираетесь поступать. Сделав выбор, получите информацию о том, куда собираетесь поступать. Пример для бакалавриата.



Рисунок 4.7 - Отрывок из раздела «Абитуриент»

Подробнее прочитать и ознакомиться со статьей «Ассистент-Бот СибГУТИ: Руководство Пользователя» можно по электронному адресу: <https://vk.com/@-170978780-rukovodstvo-polzovately>

ЗАКЛЮЧЕНИЕ

В результате данной работы был разработан чат-бот для СибГУТИ, создана и настроена группа «ВКонтакте» для него.

Было разработано интуитивно понятное меню для пользователя, чтобы ему было проще ориентироваться в поиске ответа на интересующие вопросы.

Также необходимо было сделать ответы на вопросы таким образом, чтобы они были максимально краткими и информативными.

Чтобы получать ответ на вопрос быстро, были прописаны скрипты диалога исходя из заложенных вопросов. Можно получить ответ только на них. На любые другие вопросы он будет отправлять пользователю меню чат-бота.

Для того, чтобы диалог не зашел в тупик, был разработано и схематически изображено дерево сценариев диалога. Благодаря ему можно легко организовать логику чат-бота и предусмотреть все возможные исходы диалога. А также исключить тупиковые исходы событий.

В ходе разработки я ознакомилась с новым для себя языком программирования, а также сервисами, платформой и СУБД. А конкретно с JavaScript, Node.js, npm и MySQL.

В процессе создания чат-бота, также мною было изучено VK API, которое позволяет получать информацию из баз данных «ВКонтакте».

Чтобы пользователю было понятно, как работать с чат-ботом, мною было составлено подробное руководство пользователя. Оно оформлено в виде статьи и выложено в группе чат-бота.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

- 1 Лямин, А. В. Использование социальных сетей в образовании [Электронный ресурс]: URL: <http://www.iprbookshop.ru/66487.html> (дата посещения : 23.05.2019)
- 2 Свободная энциклопедия Википедия, статья «Бот_(программа)» [Электронный ресурс]: URL: [https://ru.wikipedia.org/wiki/%D0%91%D0%BE%D1%82 \(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0\)](https://ru.wikipedia.org/wiki/%D0%91%D0%BE%D1%82 (%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0)) (дата посещения сайта 23.05.2019)
- 3 Свободная энциклопедия Википедия, статья «API» [Электронный ресурс]: URL: <https://ru.wikipedia.org/wiki/API> (дата посещения сайта 23.05.2019)
- 4 Документация для разработчиков, статья «Знакомство с API ВКонтакте» [Электронный ресурс]: URL: https://vk.com/dev/first_guide (дата посещения сайта 23.05.2019)
- 5 Свободная энциклопедия Википедия, статья «Эмоджи» [Электронный ресурс]: URL: <https://ru.wikipedia.org/wiki/%D0%AD%D0%BC%D0%BE%D0%B4%D0%B7%D0%B8> (дата посещения сайта 13.06.2019)
- 6 Свободная энциклопедия Википедия, статья «ВКонтакте» [Электронный ресурс]: URL: <https://ru.wikipedia.org/wiki/%D0%92%D0%9A%D0%BE%D0%BD%D1%82%D0%B0%D0%BA%D1%82%D0%B5> (дата посещения сайта 15.06.2019)
- 7 Свободная энциклопедия Википедия, статья «Node.js» [Электронный ресурс]: URL: <https://ru.wikipedia.org/wiki/Node.js> (дата посещения сайта 15.06.2019)
- 8 Введение в СУБД MySQL /Национальный Открытый Университет "ИНТУИТ" [Электронный ресурс]: URL: <http://www.iprbookshop.ru/73650.html> (дата посещения : 15.06.2019)
- 9 Документация для разработчиков, статья «JavaScript» [Электронный ресурс]: URL: <https://developer.mozilla.org/ru/docs/Web/JavaScript> (дата посещения сайта 15.06.2019)

ПРИЛОЖЕНИЕ А

Наиболее употребляемые текстовые сокращения

СибГУТИ – Сибирский государственный университет телекоммуникаций и информатики

ВУЗ – высшее учебное заведение

API – application programming interface (программный интерфейс приложения)

СУБД – система управления базами данных

Js – JavaScript

Npm – node package manager

ПРИЛОЖЕНИЕ Б

Листинг файла bot.js

```
const VkBot = require('node-vk-bot-api/lib');
const Session = require('node-vk-bot-api/lib/session');
const Stage = require('node-vk-bot-api/lib/stage');
const Scene = require('node-vk-bot-api/lib/scene');
const moment = require('moment');
const mysql = require("mysql2");
const whatsdo = require('./functions');

const connection = mysql.createConnection({
  host: "localhost",
  user: "root",
  database: "dbtable",
  password: "mIY6F}IS6h|wa$v"
});

const bot = new
VkBot('12617858a5c31f15050b6968efb86634a1358926bcaf9bfa5db1220abcb
f1402593f735b9af067f76865e');

const session = new Session();
const scene = new Scene('input',
  (ctx) => {
    ctx.session.text = ""
    ctx.session.text = ctx.message.text;
    //console.log(ctx.session.text + ' 1input');
    ctx.scene.next();
  },
  (ctx) => {
    ctx.scene.leave();
    console.log(ctx.session.text + ' 2input')
    if(ctx.session.text == "■Расписание") {
      //console.log(ctx.message.text + ' input Расписание');
      const sql = `SELECT * FROM group_schedule WHERE number=?`;
      const filter = ctx.message.text.toUpperCase();
      connection.query(sql, filter, function(err, results) {
        if(err) console.log(err);
        const data = results;
        output = ""
        for(let i=0; i < data.length; i++) {
          output = output + data[i].id;
        }
        if (output != ""){
          //console.log(output + ' *output');
          bot.sendMessage(ctx.message.peer_id, 'Ваше
Расписание!', output)
          ctx.reply(whatsdo.student(ctx));
        }
      }
    }
  })
}
```

```

else{
    //console.log(output + '*output ошибка');
    ctx.reply("Вы ввели группу не верно или такой группы
нет")
    ctx.reply(whatsdo.student(ctx));
}
});
}

if(ctx.session.text == "🔍Поиск аудитории") {
    //console.log(ctx.message.text + ' input Поиск аудитории')
    const sql = `SELECT * FROM auditorium WHERE number=?`;
    const filter = ctx.message.text.toUpperCase();
    connection.query(sql, filter, function(err, results) {
        if(err) console.log(err);
        const data = results;
        output = ""
        for(let i=0; i < data.length; i++) {
            output = output + data[i].corps + " " +
data[i].purpose + "\n";
        }
        if (output != "") {
            //console.log(output + ' *output');
            ctx.reply(output);
            ctx.reply(whatsdo.student(ctx));
        }
        else {
            //console.log(output + ' *output ошибка');
            ctx.reply("Такой аудитории нет или ввели не
правильно")
            ctx.reply(whatsdo.student(ctx));
        }
    });
}
});

const stage = new Stage(scene);
bot.use(session.middleware());
bot.use(stage.middleware());

bot.command('👤Студент', (ctx) => {
    //console.log(ctx.message.text + ' *Студент*');
    ctx.reply('Привет, Студент! Чем я могу тебе помочь?');
    ctx.reply(whatsdo.student(ctx));
});

bot.command('🎓Абитуриент', (ctx) => {
    //console.log(ctx.message.text + ' *Абитуриент*');
    ctx.reply('Привет, Абитуриент! Чем я могу тебе помочь?');
    ctx.reply(whatsdo.matriculant(ctx));
});

```

```

bot.command('⬅️Назад', (ctx) => {
    //console.log(ctx.message.text + ' *Назад*');
    ctx.reply(whatsdo.start(ctx));
});

//*****Студент*****
bot.command('🔍Поиск аудитории', (ctx) => {
    //console.log(ctx.message.text + ' *Поиск аудитории*');
    ctx.reply('Введите номер аудитории в формате 430a:');
    ctx.scene.enter('input');
});

bot.command('📅Расписание', (ctx) => {
    //console.log(ctx.message.text + ' *Расписание*');
    ctx.reply('Введите группу в формате ИП-513:');
    ctx.scene.enter('input');
});

bot.command('📅Какая сейчас неделя?', (ctx) => {
    //console.log(ctx.message.text + ' *Какая сейчас неделя?');
    week_start = moment("11-02-2019", "DD-MM-YYYY").format("w");
    //дата начала учебы
    week_now = moment().format("w");
    parity = 2; //четность нечетность
    //console.log(week_start + ' *week_start ' + moment("11-02-2019", "DD-MM-YYYY").format("DD-MM-YYYY"));
    //console.log(week_now + ' *week_now ' + moment().format("DD-MM-YYYY"));
    if ((week_start % 2 == 0 && parity == 2) || (week_start % 2 != 0 && parity == 1)) {
        if (week_now % 2 == 0) {
            ctx.reply('Сейчас 2📅неделя (четная)');
        } else {
            ctx.reply('Сейчас 1📅неделя (нечетная)');
        }
    } else {
        if (week_now % 2 == 0) {
            ctx.reply('Сейчас 1📅неделя (нечетная)');
        } else {
            ctx.reply('Сейчас 2📅неделя (четная)');
        }
    }
    ctx.reply(whatsdo.student(ctx));
});

```

```

bot.command('📅Мероприятия', (ctx) => {
    //console.log(ctx.message.text + ' *Мероприятия');
    ctx.reply('Всю актуальную информацию можно узнать здесь:');
    ctx.reply('https://sibsutis.ru/science/novosti.php');
    ctx.reply(whatsdo.student(ctx));
});

bot.command('ℹ️Информация о деканатах', (ctx) => {
    //console.log(ctx.message.text + ' *Информация о деканатах');
    ctx.reply('ИВТ:');
    ctx.reply('https://sibsutis.ru/faculties/ivt/contacts.php');
    ctx.reply('АЭС:');
    ctx.reply('https://sibsutis.ru/faculties/aes/contacts.php');
    ctx.reply('МТС:');
    ctx.reply('https://sibsutis.ru/faculties/mts/contacts.php');
    ctx.reply('ГФ:');
    ctx.reply('https://sibsutis.ru/faculties/gf/contacts.php');
    ctx.reply(whatsdo.student(ctx));
});

bot.command('🕒Часы работы Здравпункта', (ctx) => {
    //console.log(ctx.message.text + ' *Часы работы Здравпункта');
    ctx.reply('Часы приёма - 10:00 - 14:00');
    ctx.reply('Выходной суббота, воскресенье');
    ctx.reply(whatsdo.student(ctx));
});

//*****Абитуриент*****
bot.command('✅Поступление', (ctx) => {
    //console.log(ctx.message.text + ' *Поступление');
    ctx.reply(whatsdo.arrival(ctx));
});

bot.command('❓Частые вопросы', (ctx) => {
    //console.log(ctx.message.text + ' *Частые вопросы');
    ctx.reply('Задать вопрос и найти ответ можно здесь:');

    ctx.reply('https://sibsutis.ru/applicants/wg/group/48/forum/');
    ;
    ctx.reply(whatsdo.matriculant(ctx));
});

bot.command('📰Новости', (ctx) => {
    //console.log(ctx.message.text + ' *Новости');
    ctx.reply('Актуальные новости:');
    ctx.reply('https://sibsutis.ru/applicants/news/');
    ctx.reply(whatsdo.matriculant(ctx));
});

```



```

bot.command('Где находится корпус?', (ctx) => {
    //console.log(ctx.message.text + ' *Где находится корпус?');
    bot.sendMessage(ctx.message.peer_id, 'Карта корпусов:',
'photo-170978780_456239017')
    ctx.reply(whatsdo.matriculant(ctx));
});

bot.command('Направления', (ctx) => {
    console.log(ctx.message.text + ' *Направления');
    //ctx.reply('Основная информация:');
    ctx.reply('https://www.sibsutis.ru/applicants/abiturient/facul
tets/');
    ctx.reply(whatsdo.matriculant(ctx));
});

bot.command('Военная кафедра', (ctx) => {
    //console.log(ctx.message.text + ' *Военная кафедра');
    ctx.reply('Правила поступления')
    ctx.reply('http://uvc.sibsutis.ru/conditions')
    ctx.reply(whatsdo.matriculant(ctx));
});

bot.command('Контактные данные', (ctx) => {
    //console.log(ctx.message.text + ' *Контактные данные');
    ctx.reply('Основная контакты:');
    ctx.reply('https://sibsutis.ru/about/contacts/')
    ctx.reply(whatsdo.matriculant(ctx));
});

//*****Поступление*****
bot.command('Бакалавриат', (ctx) => {
    //console.log(ctx.message.text + ' *Бакалавриат');
    ctx.reply('https://sibsutis.ru/applicants/priem-2019.php');
    ctx.reply(whatsdo.matriculant(ctx));
});

bot.command('Магистратура', (ctx) => {
    //console.log(ctx.message.text + ' *Магистратура');

    ctx.reply('https://sibsutis.ru/applicants/abiturient/magistr/'
);
    ctx.reply(whatsdo.matriculant(ctx));
});

bot.command('Аспирантура', (ctx) => {
    //console.log(ctx.message.text + ' *Аспирантура');
    ctx.reply('https://sibsutis.ru/applicants/abiturient/aspirant/
priyem-2019-aspirantura.php');
    ctx.reply(whatsdo.matriculant(ctx));
});

```

```
bot.command('📄СПО', (ctx) => {
    //console.log(ctx.message.text + ' *СПО');
    ctx.reply('https://sibsutis.ru/applicants/priyem-2019-
programmy-spo.php');
    ctx.reply(whatsdo.matriculant(ctx));
});

bot.on((ctx) => {
    //console.log(ctx.message.text + ' *что угодно*');
    ctx.reply(whatsdo.start(ctx));
});

bot.startPolling(() => {
    console.log('Bot started.');//вывод на консоль
});
```

ПРИЛОЖЕНИЕ В

Листинг файла functions.js

```
const Markup = require('node-vk-bot-api/lib/markup');

module.exports = {
  start: function (ctx) {
    ctx.reply('●Меню●', null, Markup
      .keyboard([
        Markup.button('🎓Студент', 'primary'),
        Markup.button('□Абитуриент', 'positive'),
      ])
      .oneTime());
  },

  student: function (ctx) {
    ctx.reply('Выберите действие:', null, Markup
      .keyboard([
        Markup.button('🔍Поиск аудитории', 'primary'),
        Markup.button('📅Расписание', 'primary'),
        Markup.button('📅Какая сейчас неделя?', 'primary'),
        Markup.button('🎊Мероприятия', 'primary'),
        Markup.button('📰Информация о деканатах', 'primary'),
        Markup.button('🕒Часы работы Здравпункта', 'primary'),
        Markup.button('⬅️Назад', 'negative'),
      ], { columns: 2 })
      .oneTime());
  },

  matriculant: function (ctx) {
    ctx.reply('Выберите действие:', null, Markup
      .keyboard([
        Markup.button('✔️Поступление', 'positive'),
        Markup.button('💡Частые вопросы', 'positive'),
        Markup.button('📰Новости', 'positive'),
        Markup.button('📍Где находится корпус?', 'positive'),
        Markup.button('🏠Направления', 'positive'),
        Markup.button('👤Военная кафедра', 'positive'),
        Markup.button('👤Контактные данные', 'positive'),
        Markup.button('⬅️Назад', 'negative'),
      ], { columns: 2 })
      .oneTime());
  },
}
```

```

arrival: function (ctx) {
  ctx.reply('Куда собираетесь поступить?', null, Markup
    .keyboard([
      Markup.button('🎓Бакалавриат', 'positive'),
      Markup.button('🎓Магистратура', 'positive'),
      Markup.button('🎓Аспирантура', 'positive'),
      Markup.button('🎓СПО', 'positive'),
    ], { columns: 2 })
    .oneTime());
}
};

```