МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ

ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТКафедра «Вычислительная техника»

Дисциплина «Системное программное обеспечение».

Пояснительная записка к курсовому проекту.

**Тема: «Приложение для получения программы телепередач с использованием технологии TelegramBot»**

Выполнил:

студент группы ИВТВМбд-31

Захарычев Никита.

Проверил:

преподаватель кафедры

«Вычислительная техника»

Макаров Павел Сергеевич.

г.Ульяновск, 2017

**Оглавление**

**Введение**

Бот, а также интернет-бот, www-бот и т. п. (англ. bot, сокр.) — специальная программа, выполняющая автоматически и/или по заданному расписанию какие-либо действия через интерфейсы, предназначенные для людей. Обычно боты предназначаются для выполнения работы, однообразной и повторяемой, с максимально возможной скоростью (очевидно, намного выше возможностей человека).

В данном курсовом проекте под словом бот подразумевается чат-бот, который по запросу пользователя присылает необходимую ему информацию.

В чем преимущества **Telegram-бота**?

**Telegram-Боты** – это маленькие умные помощники. Их прелесть в том, что вам не нужно покидать экосистему мессенджера, чтобы решить элементарную задачу: спросить, уточнить, узнать, посчитать. Каждый бот сам по себе, но вместе они образуют вашу личную армию ассистентов, каждому из которых вы даете поручение, а они безропотно и четко выполнят ваши пожелания.

**Telegram-Боты** – это небольшие шестеренки и приводные ремни к вашим задачам, которые могут действовать слаженно, если с умом подогнать их друг к другу. Скажем, на какие-то конкретные действия вы можете настроить ботов к определенному времени. Например, доставлять вам с утра свежие новости, прогноз погоды, ситуацию по пробкам. А вечером – рекомендацию по фильму, смешные анекдоты и рецепт дня. Пользователи могут взаимодействовать с ботами при помощи сообщений, отправляемых через обычные или групповые чаты. Логика бота контролируется при помощи HTTPS запросов к нашему API для ботов.

В рамках данного проекта будет использоваться технология TelegramBot API.

**1. Формирование требований**

**Цель курсовой работы** — ознакомится с технологией написания ботов, улучшить навыки программирования на выбранном языке.

**Задачи данной работы**:

* определить какие виды программ относятся к «ботам» изучить функции этих программ
* ознакомится с функционалом (API), который предоставляет Telegram для создания бота
* ознакомиться с выбранной предметной областью
* определить функционал своего приложения
* спроектировать архитектуру приложения
* выбрать язык программирования, на котором будет реализован бот
* написать программу
* отладка и тестирование бота

**2. Обоснование целесообразности и осуществимости проекта**

Все меньше людей читают газеты, смотрят телевизор, ведь можно посмотреть необходимую информацию в интернете. Но все-таки есть категория людей, которые и «сидят» в интернете и также смотрят телевизор. Столкнувшись с проблемой постоянного поиска газеты или же программы передач в интернете, который занимает довольно длительное время, было решено реализовать данный проект. Телефон всегда находится «под рукой», большую часть времени люди проводят в социальных сетях, мессенджерах, поэтому использование бота в качестве источника информации экономит огромное количество времени.

Целесообразность осуществления проекта обусловлена тем, что в настоящее время боты стали очень популярны. Они буквально повсюду. Бот является интерактивным участником чата: чтобы он вступил в диалог, достаточно дать простую команду. А это значит, что он на один шаг ближе к пользователю, чем мобильное приложение, и на два, — чем веб-сайт.

**3. Техническое задание**

Был определён функционал и средства разработки. Составлено следующее техническое задание.

**Название проекта** - приложение для программы телепередач с использованием технологии TelegramBot.

**Применение -** получение актуальной информации в удобном виде не выходя из мессенджера.

**Состав**

Функции бота:

* возможность получения полного списка передач по каналам на день запроса.
* возможность получения списка передач по жанру (фильмы, сериалы, спорт, детям) на текущий день.

Техническая часть:

* Использование TelegramAPI
* Язык программирования Node.js
* Web-приложение, реализующее логику.

**Основные требование по назначению**

Система должна обеспечивать выполнение всех вышеперечисленных функциональных возможностей.

**Реализация**

Проект реализован на операционной системе Windows 10 при помощи языка JavaScript (Node.JS).

**Ресурсные ограничения**

* Поддержка и оптимизация приложения мессенджера Telegram.
* Ресурсные ограничения идентичны ресурсным ограничениям приложения мессенджера Telegram.

**4. Описание Telegram Bot API**

Bot API представляет собой HTTP-интерфейс для работы с ботами в Telegram. При создании бота ему присваивается уникальный токен, который нужен для доступа к боту из программы.

111111:AAHlxTLpuPokbEQgLYKHwMMl\_T\_EVG4IArM.

Все запросы к Telegram Bot API должны осуществляться через HTTPS в следующем виде: https://api.telegram.org/bot<token>/НАЗВАНИЕ\_МЕТОДА.

**Например:**

https://api.telegram.org/111111:AAHlxTLpuPokbEQgLYKHwMMl\_T\_EVG4IArM /getMe

Допускаются GET и POST запросы. Для передачи параметров в Bot API доступны 4 способа:

* Запрос в URL;
* application/x-www-form-urlencoded;
* application/json (не подходит для загрузки файлов);
* multipart/form-data (для загрузки файлов).

Ответ придёт в виде JSON-объекта, в котором всегда будет булево поле ok и опциональное строковое поле description, содержащее человекочитаемое описание результата. Если поле ok истинно, то запрос прошёл успешно и результат его выполнения можно увидеть в поле result. В случае ошибки поле ok будет равно false, а причины ошибки будут описаны в поле description. Кроме того, в ответе будет целочисленное поле error\_code.

**5. Отличие бота от классического аккаунта.**

У роботов нет статусов «онлайн» и «был в сети», вместо этого отображается надпись «робот». Для роботов выделено ограниченное место на серверах — все сообщения будут удалены по прошествии определённого срока после обработки.

Роботы не могут сами начать общение с пользователем. Пользователь должен либо добавить робота в группу, либо первым начать с ним диалог. Для этого можно использовать ссылки вида: telegram.me/<bot\_username> или поиск по имени пользователя.

Имя пользователя у робота должно заканчиваться на «bot».

При добавлении в конференцию, по умолчанию робот не получает всех сообщений.

**6.** **Архитектура**

Приложение включает следующие компоненты:

* модуль TeleBot автоматизирующий взаимодействие с Telegram Bot API;
* сайт для парсинга программы телепередач;
* исходный код приложения, реализующий логику взаимодействия;
* аккаунт бота на платформе Telegram.

**7. Реализация**

В проекте использовались следующие библиотеки:

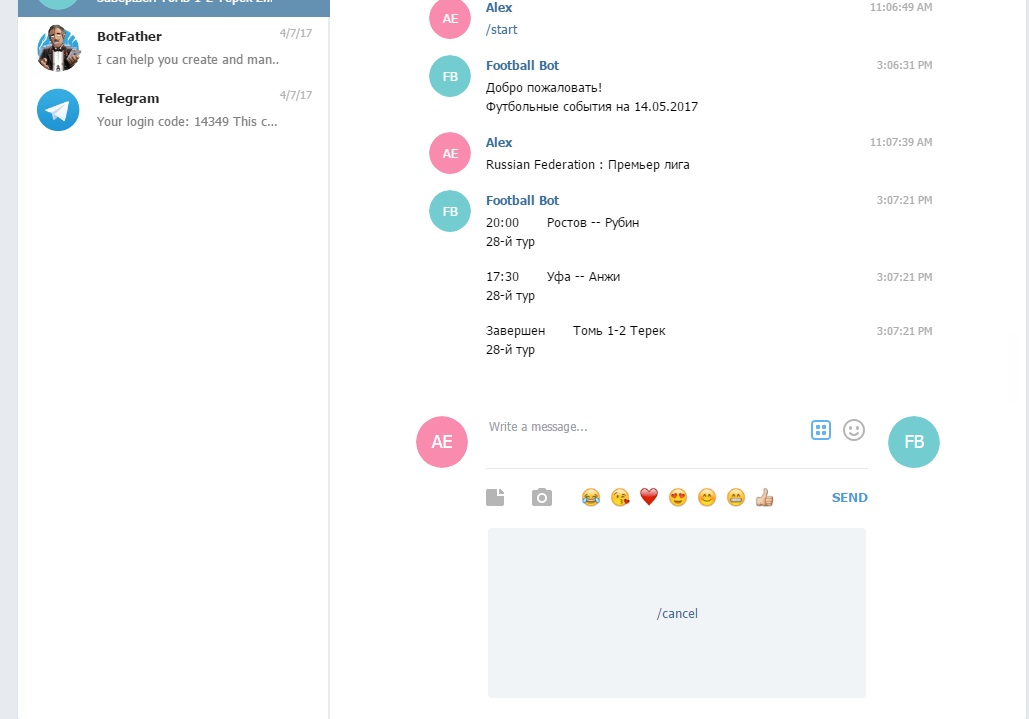
* **Node-telegram-bot-api** *–* модуль упрощающий взаимодействие с Telegram Bot API.
* **Request** *-*  модуль для простого создания http запросов. Использовался для получения страниц сайта.
* **Cheerio** *–* реализация фреймворка Jquery для Node.js. Использовался для парсинга страниц.

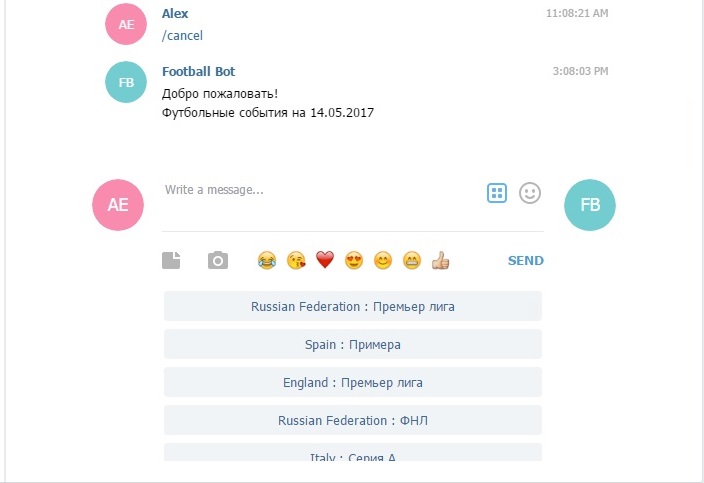
**Описание команд**

/start – команда запуска бота (после исполнения выводится начальное меню).

/cancel – команда выхода из пунктов меню.

Для команд получения списка событий используется кнопочный интерфейс.

****

****

**8. Руководство для разработчика**

Представление объектов проекта:

*const TelegramBot = require('node-telegram-bot-api')*– подключение модуля для работы с Telegram Bot API;

*const request = require("request")* – подключение модуля для работы с запросами;

*const cheerio = require("cheerio")* – подключение модуля для работы с Document Object Model(Библиотека реализует подмножество ядра jQuery. Функционал позволяет работать с очень простой, последовательной моделью DOM. В результате синтаксического анализа, манипулирование и рендеринг становятся невероятно эффективными);

*const url = "http://soccer365-1.xyz/online/"* – определение константы с адресом сайта для осуществления парсинга;

*const token = '340282230:AAHlxTLpuPokbEQgLYKHwMMl\_T\_EVG4IArM'* – определение константы с токеном(идентификатором) бота;

*const welcome = "Добро пожаловать!\nФутбольные события на " + getDate()* – определение константы содержащее приветствие, отправляемое собеседнику в момент начала диалога;

const bot = new TelegramBot(token, {polling: true}) – определение константы с экземпляром бота(Первым параметром передается токен бота, вторым параметром – объект со свойством polling : true. Данное свойство определяет циклический период обновления бота);

*var cncl = [['/cancel']]*– определение переменной, содержащей кнопку отмены;

*var leaguesNames = []* – определение массива, содержащего имена актуальных футбольных лиг;

*var leaguesNamesKey = []*– определение массива, содержащего кнопки актуальных футбольных лиг;

*var leagues = []*– определение массива, содержащего DOM-объекты актуальных футбольных лиг;

*var countMatches = []*– определение массива, содержащего количества матчей каждой лиги;

*var $ = cheerio.load(body)*– определение переменной, результат метода *load()* библиотеки *cheerio*, а именно происходит инициализация тела документа.

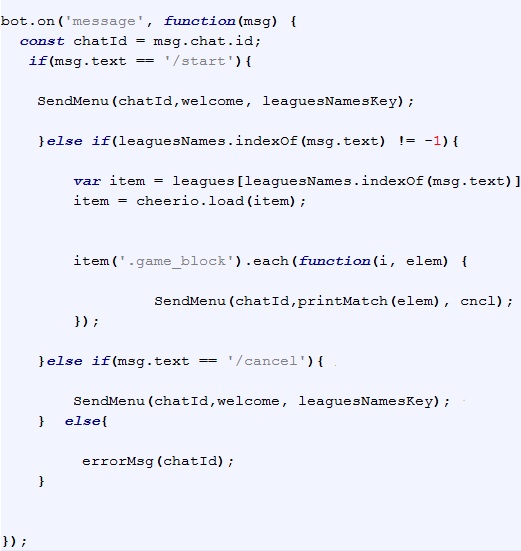
Назначения функций проекта:

*сheerio*.load(obj) – Данный метод производит загрузку DOM-объекта, тем самым инициализирует объект для дальнейшей обработки его с помощью функционала библиотеки;

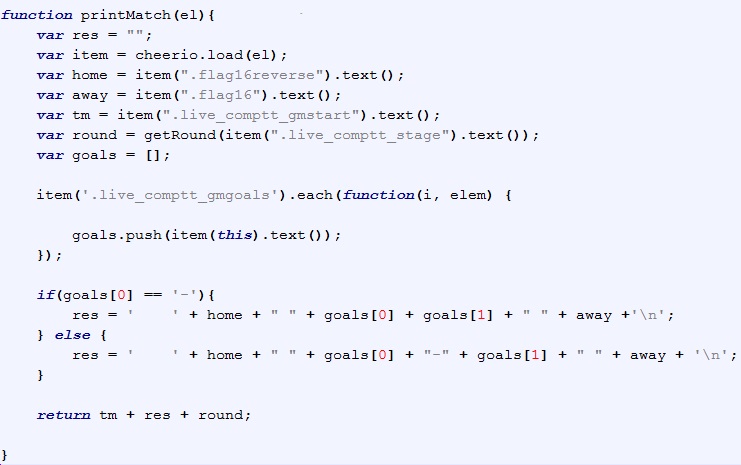
*request(url, function (error, response, body){…})*– метод, осуществляющий запрос к web-странице для получения тела DOM и дальнейшей предобработки документа(первым параметром передается адрес сайта, вторым – callback функция. Данная функция выполняет формирования списка лиг и списка кнопок с лигами);



*bot.on('message', function(msg){…})*– метод, осуществляющий обработку входящего сообщения. Первым параметром передается тип события, вторым – callback функция в которой в зависимости от водящего сообщения msg бот принимает решение на ответ;



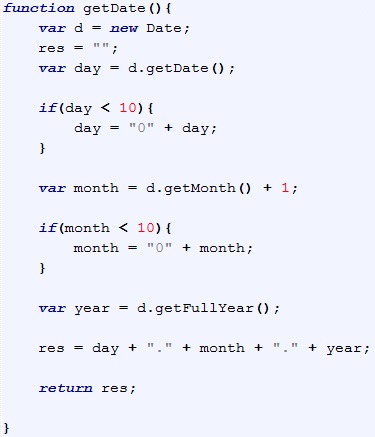
*printMatch(el)*– метод принимает объект актуального матча, в зависимости от статуса матча, формирует строку из команды хозяев, команды гостей, времени начала матча, либо сыгранного времени на данный момент и счета, если матч не завершен;



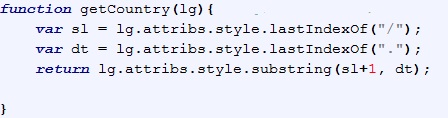
*SendMenu(id,msg, arr)* – метод, осуществляющий вывод ответа и комбинации кнопок в зависимости от полученного сообщения;

*bot.sendMessage(id, msg, {reply\_markup: JSON.stringify({keyboard: arr})})} –* метод, осуществляющий отправку сообщения. Первым параметром передается идентификатор чата, вторым – сообщение, третьим дополнительным параметром передается объект, в свойстве keyboard которого указан массив для отображения в виде кнопок;

*getDate()* – метод, осуществляющий формирование текущей даты в формате DD.MM.YYYY;



*getCountry(lg) –* метод, осуществляющий формирование названия страны из свойства style, содержащего абсолютный путь до изображения флага. Входным параметром является DOM-объект лиги, страну которой нужно получить;



**9. Тестирование**

Тестирование проводилось для всех используемых команд. Проверялась реакция приложения на несуществующие команды, текст.

На всех тестах программа отработала корректно.

**Заключение**

В ходе данной работы освоены новые технологи по созданию чат-ботов. Улучшены навыки программирования на языке Node.js. Получен опыт использования дополнительных библиотек и модулей.

Также написано приложение, которое может быть полезным людям, которые пользуются ресурсом *habrahabr.ru/*

**Список литературы**

1. <https://tlgrm.ru/docs/bots> — документация по созданию Telegram бота
2. <https://core.telegram.org/bots> — Telegram API
3. <https://github.com/yagop/node-telegram-bot-api> - node-telegram-bot-api
4. <https://github.com/request/request> — request
5. <https://github.com/cheeriojs/cheerio> - Cheerio

**Приложение 1. Исходный код**

const TelegramBot = require('node-telegram-bot-api'); // подключаем библиотеку для работы с ботом

const request = require("request"); // подключаем библиотеку для запросов

const cheerio = require("cheerio"); // подключаем библиотеку для работы с DOM

const url = "http://soccer365-1.xyz/online/"; // сайт для извлечения данных

const token = '340282230:AAHlxTLpuPokbEQgLYKHwMMl\_T\_EVG4IArM'; // токен бота

const welcome = "Добро пожаловать!\nФутбольные события на " + getDate(); // приветствие

const bot = new TelegramBot(token, {polling: true}); // инициализация бота

var cncl = [['/cancel']]; // кнопка отмены

var leaguesNames = []; // массив имен лиг

var leaguesNamesKey = []; // массив имен лиг для кнопок

var leagues = []; // массив объектов лиг

var countMatches = []; // массив количества матчей каждой лиги

request(url, function (error, response, body) { // находим нужную таблицу и формируем лиги и их имена

if (!error) {

var $ = cheerio.load(body);

var data = $("#result\_data");

leagues = $(data).children(".live\_comptt\_bd");

$('.block\_header .flag16').each(function(i, elem) {

var country = getCountry(this);

leaguesNames.push(country + " : " + $(this).text());

});

leaguesNamesKey = arrOfArr(leaguesNames);

}else {

bot.sendMessage(chatId, "Произошла ошибка: " + error);

}

});

bot.on('message', function(msg) { // обработка полученного сообщения

const chatId = msg.chat.id; // идентификатор чата

if(msg.text == '/start'){ // если старт

SendMenu(chatId,welcome, leaguesNamesKey); // выводим список лиг

}else if(leaguesNames.indexOf(msg.text) != -1){ // если название лиги

var item = leagues[leaguesNames.indexOf(msg.text)]; // находим в массиве лиг нужную

item = cheerio.load(item);

item('.game\_block').each(function(i, elem) { // ищем все блоки с игрой и для каждой игры

SendMenu(chatId,printMatch(elem), cncl); // выводим название команд и результат

});

}else if(msg.text == '/cancel'){ // если отмена то

SendMenu(chatId,welcome, leaguesNamesKey); // выводим список лиг

} else{

errorMsg(chatId); // иначе неизвестная команда

}

});

function printMatch(el){ // формирование матча

var res = "";

var item = cheerio.load(el);

var home = item(".flag16reverse").text();

var away = item(".flag16").text();

var tm = item(".live\_comptt\_gmstart").text();

var round = getRound(item(".live\_comptt\_stage").text());

var goals = [];

item('.live\_comptt\_gmgoals').each(function(i, elem) {

goals.push(item(this).text());

});

if(goals[0] == '-'){

res = ' ' + home + " " + goals[0] + goals[1] + " " + away +'\n';

} else {

res = ' ' + home + " " + goals[0] + "-" + goals[1] + " " + away + '\n';

}

return tm + res + round;

}

function arrOfArr(arr){ // из одномерного в двумерный(для кнопок)

var res = [];

for(var i = 0; i < arr.length; i++){

var tmp = [];

tmp.push(arr[i]);

res.push(tmp);

}

return res;

}

function errorMsg(id){ // вывод сообщения о неизвестной команде

bot.sendMessage(id, "Неизвестная команда",{

reply\_markup: JSON.stringify({

keyboard: [['/start']]

})

});

}

function SendMenu(id,msg, arr){ // вывод сообщения

bot.sendMessage(id, msg, {

reply\_markup: JSON.stringify({

keyboard: arr

})

});

}

function getDate(){ // получение даты в нужном формате

var d = new Date;

res = "";

var day = d.getDate();

if(day < 10){

day = "0" + day;

}

var month = d.getMonth() + 1;

if(month < 10){

month = "0" + month;

}

var year = d.getFullYear();

res = day + "." + month + "." + year;

return res;

}

function getCountry(lg){ // формируем из пути для флага страны ее название

var sl = lg.attribs.style.lastIndexOf("/");

var dt = lg.attribs.style.lastIndexOf(".");

return lg.attribs.style.substring(sl+1, dt);

}

function getRound(round){ // формируем строку названия тура

if(round.lastIndexOf(",") !== -1){

var ch = round.lastIndexOf(",");

round = round.substring(0,ch);

}

return round;

}