Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования «Уральский федеральный университет имени первого Президента России Б.Н.Ельцина»

Институт радиоэлектроники и информационных технологий – РТФ Центр ускоренного обучения

**НАПИСАНИЕ БОТА ДЛЯ TELEGRAM НА JAVA**

КУРСОВОЙ ПРОЕКТ

По дисциплине “Программирование”

Пояснительная записка

09.03.03 58.29.21 013 ПЗ

Ст. преподаватель: Н.А. Архипов

Нормоконтролёр: С.И. Тимошенко

Студент группы РИЗ-400028у: В.А. Кудрявцев

Екатеринбург 2023

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc159262558)

[1 Постановка задачи 4](#_Toc159262559)

[1.1 Регистрация бота в Telegram 4](#_Toc159262560)

[2 Создание программы 6](#_Toc159262561)

[2.1 Написание классов бота в Intellij IDEA 6](#_Toc159262562)

[3 Размещение программы в сети Интернет 12](#_Toc159262563)

[3.1 Публикация проекта 12](#_Toc159262564)

[3.2 Альтернативные языки для написания telegram-бота 12](#_Toc159262565)

[3.3 Востребованность на рынке труда специальности, связанной с написанием чат-ботов для социальной сети 14](#_Toc159262566)

[ЗАКЛЮЧЕНИЕ 15](#_Toc159262567)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 16](#_Toc159262568)

[ПРИЛОЖЕНИЕ А (обязательное) Текст программ 17](#_Toc159262569)

[А.1 Класс бота Bot.java 17](#_Toc159262570)

[А.2 Класс Weather.java 20](#_Toc159262571)

[А.3 Класс Model.java 22](#_Toc159262572)

[ПРИЛОЖЕНИЕ Б (справочное) Скриншот с сайта Антиплагиат 24](#_Toc159262573)

ВВЕДЕНИЕ

Telegram является одной из ведущих площадок медиа пространства. Важно отметить, что Telegram – это один из самых популярных мессенджеров на территории России и стран СНГ, является российской разработкой от создателя социальной сети ВКонтакте Павла Дурова. Изначально, мессенджер Telegram был создан с целью обмена сообщениями между пользователями. С течением времени, функционал мессенджера претерпел изменения и улучшения, а именно собеседником любого пользователя может стать автоматическая программа - робот, иными словами бот.

В Telegram существует три вида медиасервисов: чаты, каналы и боты. Телеграм-чаты создаются для обмена сообщениями между всеми участниками беседы. Телеграм-канал подразумевает отправку информации от одного лица, который является администратором. Бот – это собеседник в мессенджере, робот, отвечающий на запросы автоматически. Он заранее запрограммирован выдавать пользователю определенную информацию.

Актуальность исследования данной курсовой работы обосновывается необходимостью создания чат-бота для предоставления информации пользователю посредством ответа в сообщении в социальной сети Telegram.

В качестве языка программирования для создания данного сервиса используется мультипарадигмальный высокоуровневый язык программирования Java. Объектом исследования в курсовой работе является процесс формирования запроса информации, поиск необходимой информации в Интернете, предоставление её пользователю по запросу чат-ботом. Предмет исследования представляет собой программа на языке программирования Java, содержащая в себе компоненты для полноценной работы чат-бота, предоставляющего актуальную информацию пользователю в виде сообщения.

Задачей исследования в курсовой работе является создание автоматизированного сервиса для предоставления пользователю актуальных данных о погоде в заданной географической точке по названию места на карте с помощью парсинга данных веб-сайта, имеющем в своём содержании данную информацию. После выполнения задачи, разработанный проект можно разместить на веб-сервисе для дальнейшего использования.

# 1 Постановка задачи

## 1.1 Регистрация бота в Telegram

Для чат-бота необходима регистрация на сервисе Telegram. Данная процедура выполняется с помощью бота @BotFather, который в свою очередь, позволяет получить токен для использования ботами в пространстве социальной сети Telegram.

Регистрация проходит очень просто и может быть выполнена за несколько минут пользователем. Для этого необходимо иметь на руках следующую информацию: имя бота и имя пользователя. Диалог с ботом @BotFather представлен на рисунке 1 ниже:

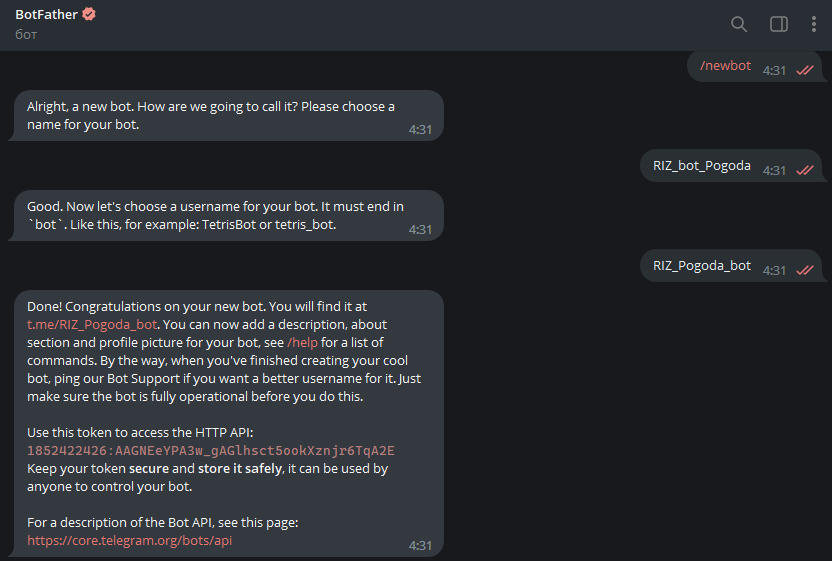


Рисунок 1 – Регистрация бота

После регистрации бота необходимо непосредственное создание классов в среде разработки.

Для написания классов создаваемого чат-бота используем среду разработки Intellij IDEA community Edition 2020.1. На ПК установлена Java JDK 15 (x64). Переменная Path прописана корректно в папку с JDK на жёстком диске, что позволяет запускать написанные классы на виртуальной машине Java в ПК.

Данные будут браться с веб-сервиса openweathermap, сервис предоставляет данных о погоде по заданному названию населённого пункта, который позволяет использовать API с данными о погоде.

Общую концепцию работы бота можно представить в виде блок-схемы (рисунок 2):

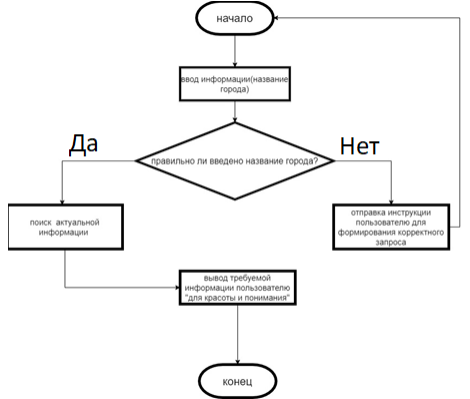


Рисунок 2 – Работа бота в виде блок-схемы

Исходя из блок-схемы, бот будет выводить запрашиваемую информацию только в случае корректного ввода пользователем начальных данных, а именно название города или посёлка.

Использование бота поможет выяснять значения погоды для любого места на карте, информация будет предоставляться в онлайн режиме, при обращении к одному и тому же городу значения показателей будут меняться.

# 2 Создание программы

## 2.1 Написание классов бота в Intellij IDEA

Основным рабочим классом бота является класс Bot.java, который содержит в себе методы, которыми описываются запрос на веб-сервис, вывод информации в чат, кнопки клавиатуры для диалога с ботом, а также метод, возвращающий имя пользователя и метод, содержащий токен, выданный @BotFather при регистрации бота.

Для создания приложения необходимо подключить фрейворк Maven, который поможет нам автоматически создавать необходимые зависимости, поэтому при создании проекта выбираем соответствующую конфигурацию. Таким образом необходимые библиотеки подтянутся в проект, а в папке с проектом сформируется файл pom.xml, отображающий зависимости, работающие в проекте. Далее можно создавать основные классы.

Также для корректной работы сервиса необходим импорт библиотек, позволяющих нашему боту понимать информацию с серверов Telegram:

import org.telegram.telegrambots.ApiContextInitializer;  
import org.telegram.telegrambots.TelegramBotsApi;  
import org.telegram.telegrambots.api.methods.send.SendMessage;  
import org.telegram.telegrambots.api.objects.Message;  
import org.telegram.telegrambots.api.objects.Update;  
import org.telegram.telegrambots.api.objects.replykeyboard.ReplyKeyboardMarkup;  
import org.telegram.telegrambots.api.objects.replykeyboard.buttons.KeyboardButton;  
import org.telegram.telegrambots.api.objects.replykeyboard.buttons.KeyboardRow;  
import org.telegram.telegrambots.bots.TelegramLongPollingBot;  
import org.telegram.telegrambots.exceptions.TelegramApiException;  
import org.telegram.telegrambots.exceptions.TelegramApiRequestException;

Также необходимы библиотеки для работы с массивом данных, получаемых с API веб-сервиса:

import java.io.IOException;  
import java.util.ArrayList;  
import java.util.List;

В теле основного класса требуется определить методы, работающие в классе, т.е. метод для распознавания сообщения, отправленного боту, а также работу кнопок клавиатуры, размещённых в окне диалога с ботом, у нас их будет две: «Инструкция», которая позволит ориентировать пользователя по вводу и «Настройки», которая в дальнейшем может быть использована для работы с методом настроек в боте.

В методе public void onUpdateReceived будет формироваться объект сообщения, отправляемого пользователю ботом, в котором будет формироваться запрашиваемая информация о состоянии погоды, а также сообщения от бота в случае нажатия кнопок клавиатуры, класс Weather будет описан ниже и будет содержать информацию, какую назначим ему мы для вывода.

Вторым основным классом в программе будет Weather.java, который в себе содержит запрос на веб-ресурс. Для корректной работы также необходим импорт библиотек:

import org.json.JSONArray;  
import org.json.JSONObject;  
  
import java.io.IOException;  
import java.io.InputStream;  
  
import java.net.URL;  
import java.util.Scanner;

В этом классе присутствует переменная величина, которая получает значения с веб-сервиса и формирует массив данных, которые необходимо преобразовать в форму, удобную для вывода в сообщении пользователю, «для красоты и понимания». Поэтому необходимо создание объекта, который заполнится необходимыми данными, таким объектом является JSONObject object.

public class Weather {  
  
public static String getWeather(String message, Model model) throws IOException {  
 URL url = new URL("http://api.openweathermap.org/data/2.5/weather?q=" + message + "&units=metric&appid=6fff53a641b9b9a799cfd6b079f5cd4e");  
  
 Scanner in = new Scanner((InputStream) url.getContent());  
 String result = "";  
 while (in.hasNext()) {  
 result += in.nextLine();  
 }  
  
 JSONObject object = new JSONObject(result);  
 model.setName(object.getString("name"));  
  
 JSONObject main = object.getJSONObject("main");  
 model.setTemp(main.getDouble("temp"));  
 model.setHumidity(main.getDouble("humidity"));  
  
 JSONArray getArray = object.getJSONArray("weather");  
 for (int i = 0; i < getArray.length(); i++) {  
 JSONObject obj = getArray.getJSONObject(i);  
 model.setIcon((String) obj.get("icon"));  
 model.setMain((String) obj.get("main"));  
 }  
  
 return "Город: " + model.getName() + "\n" +  
 "Температура: " + model.getTemp() + "C" + "\n" +  
 "Влажность: " + model.getHumidity() + "%" + "\n" +  
 "Небо: " + model.getMain() + "\n" ;  
 }  
}

При обращении бота к сайту с запросом погоды по введённому названию места на карте, запрос выглядит следующим образом:

http://api.openweathermap.org/data/2.5/weather?q=" + message + "&units=metric&applied=6fff53a641b9b9a799cfd6b079f5cd4e

В данном запросе в качестве объекта message будет введённое пользователем название города, в конце запроса содержится ключ-токен, который мы получаем в процессе регистрации на веб-ресурсе, откуда берём данные.

Для получения информации с ресурса www.openweathermap, то для получения токена необходима регистрация, в процессе регистрации мы получаем свой токен, прикрепляемый к запросу.

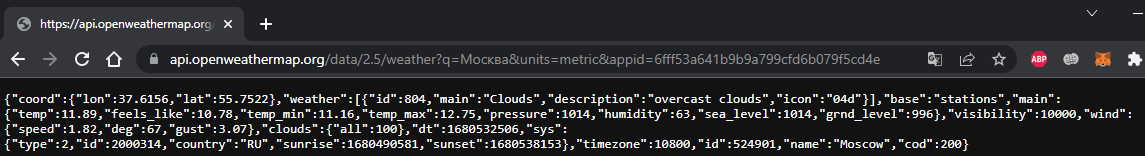


Рисунок 3 – Запрос на получение погоды по городу Москва

Для этого нужно разложить полученный с сервера запрос с целью создания нового объекта Model.java, который мы можем настроить должным образом для вывода нужной информации. Для вывода необходимой информации необходимо инициализировать переменные, которым мы будем задавать значения в соответствии с получаемыми данными: место, температура, влажность, наличие/отсутствие осадков. Создаваемый класс примет вид:

public class Model {  
 private String name;  
 private Double temp;  
 private Double humidity;  
 private String icon;  
 private String main;  
 public String getName() {  
 return name;  
 }  
 public void setName(String name) {  
 this.name = name;  
 }  
 public Double getTemp() {  
 return temp;  
 }  
 public void setTemp(Double temp) {  
 this.temp = temp;  
 }  
 public Double getHumidity() {  
 return humidity;  
 }  
 public void setHumidity(Double humidity) {  
 this.humidity = humidity;  
 }

public String getIcon() {  
 return icon;  
 }  
 public void setIcon(String icon) {  
 this.icon = icon;  
 }  
 public String getMain() {  
 return main;  
 }  
 public void setMain(String main) {  
 this.main = main;  
 }  
}

Сгенерированные методы возвращают пользователю только те данные, которые описаны в данном классе. Если требуется отображение каких-либо других данных, тогда требуется ввести необходимые переменные и добавить их в класс Model.java. В итоге, в классе Weather.java фактом отработки всех действий формируется сообщение пользователю по его запросу (рисунок 4).

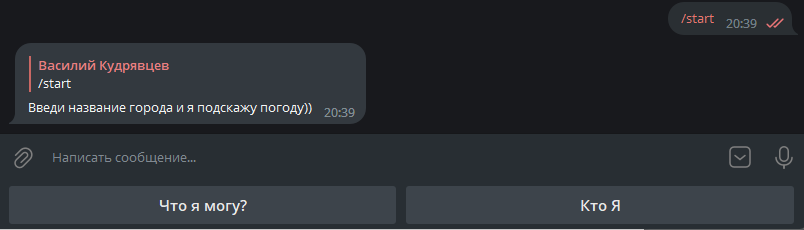


Рисунок 4 – Описание принципа работы бота

Для визуального отображения информации бот разные типы данных: вещественные числа, строки (рисунок 5):

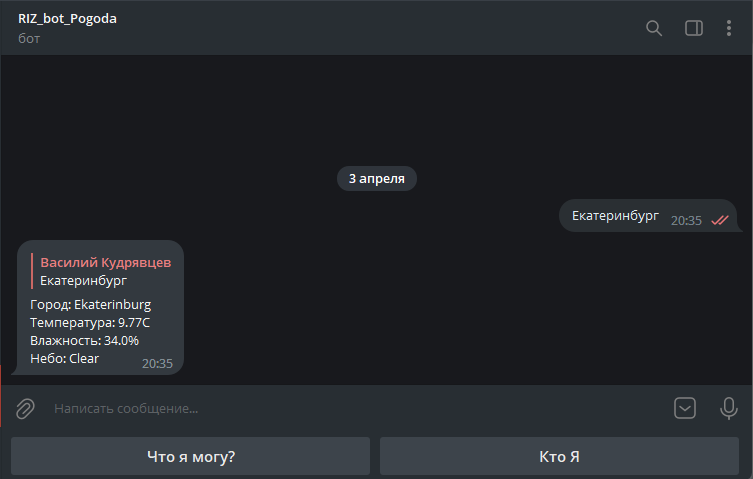


Рисунок 5 – Результат работы бота

# 3 Размещение программы в сети Интернет

## 3.1 Публикация проекта

Существует множество площадок для публикации подобных ботов, которые позволяют, размещая в интернете проект, иметь доступ к нему в любое время с любого устройства.

Можно привести список сервисов и способы размещения, большинство сервисов в сети подразумевают под собой аренду оборудования, а именно сервера с настраиваемыми параметрами.

Если мы имеем для запуска проект, сосредотачивающий работу с сервисами с участием большого количества пользователей, необходимо подсчитать параметры оборудования, которое обработает с необходимой нам скоростью запросы пользователей. Множество коммерческих площадок позволяют вручную настроить параметры виртуальной машины, которую вы можете взять в аренду и разместить на нём свою программу, в данном случае бота.

Также имеются альтернативные условно бесплатные сервисы, например heroku.com, на которую можно загрузить свой проект прямо с площадки GitHub, инициировать запуск приложения и пользоваться без привлечения физического оборудования.

При размещении проекта на реальном сервере необходимо учесть тот факт, что на сервере присутствовали компоненты, необходимые для запуска. В противном случае перед запуском необходимо вручную их устанавливать.

Java – не единственный язык, на котором можно создавать Telegram-ботов, подобных ботов пишут также на Python, JavaScript, C#, C++ и др.

## 3.2 Альтернативные языки для написания telegram-бота

Как и сказано выше, на выбор можно взять любой язык программирования для написания подобного бота. На данный момент особо популярным языком для такого проекта многие признают Python, их выбор основывается на том, что у Python более низкий порог для вхождения в процесс программирования. С современным развитием информатики, даже многие школьники разрабатывают свои первые программы именно на Python. Так как работа бота завязывается на фрейворках, то при написании данной 16 программы нужно учитывать наличие необходимых компонентов в системе для корректной работы приложения. Поскольку для размещения ПО в интернете при выборе платформы многие выбирают дистрибутивы Linux, необходимые библиотеки для Python присутствуют в системе.

Другим конкурентом для разработки данного бота может послужить JavaScript. «Младший брат» Java тоже может являться отличным решением для написания бота и дальнейшем его использовании, при наличии навыка у пользователя в написании скриптов на данном языке программирования, бот также будет стабильно работать с API указанных веб-ресурсов.

Вообще выбор языка программирования для написания подобного бота (как и многих других) весьма разнообразен: всё зависит здесь только от самого программиста, ведь лучше всего можно описать логику работы бота на языке, который вы знаете, а ерунда может получиться в любом языке, поэтому создание ботов стоит проводить только в том языке, синтаксис и принципы работы логики объектов вам хорошо известны.

## 3.3 Востребованность на рынке труда специальности, связанной с написанием чат-ботов для социальной сети

Выполнять работы по созданию данной программы может junior- разработчик. Выбор уровня разработки обусловлен тем, что даже для написания чат-бота необходимы базовые знания работы языка программирования и среды разработки, поэтому написание классов программы осилит начинающий программист. Наличие справочного материала на просторах интернета позволяют самостоятельно написать и наладить работу сервиса, связанного с социальной сетью.

Средний уровень оплаты труда junior-разработчика в России на 2023 год изображен на гистограмме ниже:

Диаграмма 1 – Средняя оплата труда junior-разработчика

ЗАКЛЮЧЕНИЕ

В вопросах создания данного инструмента, как чат-бот необходимо отталкиваться от специфики предоставляемой услуги, ведь от количества необходимых функций зависит наличие тех или иных классов ив тексте программы.

Выбор языка программирования для создания сложного бота зависит от навыков разработчика, под этим понимается степень осведомлённости о работе механизмов бота при запуске его в глобальной сети.

Размещение готового бота можно производить на различных площадках: от свободно используемых в сети, до коммерческих сервисов с набором функций по обслуживанию программного продукта.

На языке программирования Java в ходе курсовой работы создан бот для социальной сети Telegram, который предназначен для сообщения пользователю по запросу сведений о погоде по заданному названию места на карте. Алгоритм реализован на механизме парсинга данных, получаемых от веб-сервиса, и выводится на экран пользователю в виде текстового сообщения с данными, также реализована возможность передачи статусной картинки погоды в сообщении. В активном окне диалога присутствуют две интерактивные кнопки, при нажатии на которые, пользователю приходят автоматически сформированные сообщения роботом. Включена возможность обработки исключений- в случае некорректного ввода названия пользователем, бот предупреждает в сообщении о некорректном вводе.

В приложении к курсовой работе находятся тексты основных классов бота, написанные на языке программирования Java.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Козлов А. А., Телеграм-бот как простой и удобный способ получения информации [Электронный ресурс] / А. А. Козлов, А. В. Батищев //Территория науки. – 2017. – №5. – с. 55-64. – Режим доступа: https://cyberleninka.ru/article/v/telegram-bot-kakprostoy-i-udobnyy-sposob-polucheniya-informatsii (дата обращения 10.03.2021).
2. Давыдов, С. IntelliJ IDEA. Профессиональное программирование на Java / С. Давыдов. – СПб.: BHV, 2005. - 800 c
3. Машнин, Т.С. Web-сервисы Java. Профессиональное программирование / Т.С. Машнин. – СПб.: BHV, 2012. - 560 c.
4. Давыдов, С. IntelliJ IDEA. Профессиональное программирование на Java / С. Давыдов. – СПб.: BHV, 2005. - 800 c.
5. Аванесян Н. Л., Telegram, как пример мессенджера: возможности и перспективы развития. [Электронный ресурс] / Н. Л. Аванесян // Научный потенциал XXI века. – 2017. – Режим доступа: https://elibrary.ru/download/elibrary\_29653726\_34734017.pdf (дата обращения 25.05.2021).
6. Герберт, Шилдт Java 8. Руководство для начинающих / Шилдт Герберт. – М.: Диалектика / Вильямс, 2015. - 899 c.
7. Официальный сайт Heroku [Электронный ресурс]: Облачная PaaS-платформа. – Режим доступа: https://www.heroku.com/ (дата обращения 10.06.2021).
8. Архаков, Д. NodeJS: Делаем кнопки в Telegram API (inline-keyboards) [Электронный ресурс] / Д. Архаков // Блог о программировании. – 2016. – Режим доступа: https://archakov.im/post/nodejs-make-buttons-on-telegram-api.html (дата обращения 30.05.2021).

ПРИЛОЖЕНИЕ А  
(обязательное)  
Текст программ

## А.1 Класс бота Bot.java

import org.telegram.telegrambots.ApiContextInitializer;  
import org.telegram.telegrambots.TelegramBotsApi;  
import org.telegram.telegrambots.api.methods.send.SendMessage;  
import org.telegram.telegrambots.api.objects.Message;  
import org.telegram.telegrambots.api.objects.Update;  
import org.telegram.telegrambots.api.objects.replykeyboard.ReplyKeyboardMarkup;  
import org.telegram.telegrambots.api.objects.replykeyboard.buttons.KeyboardButton;  
import org.telegram.telegrambots.api.objects.replykeyboard.buttons.KeyboardRow;  
import org.telegram.telegrambots.bots.TelegramLongPollingBot;  
import org.telegram.telegrambots.exceptions.TelegramApiException;  
import org.telegram.telegrambots.exceptions.TelegramApiRequestException;  
import java.io.IOException;  
import java.util.ArrayList;  
import java.util.List;  
public class Bot extends TelegramLongPollingBot {  
 public static void main(String[] args) {  
 ApiContextInitializer.*init*();  
 TelegramBotsApi telegramBotsApi = new TelegramBotsApi();  
 try {  
 telegramBotsApi.registerBot(new Bot());  
 } catch (TelegramApiRequestException e) {  
 e.printStackTrace();  
 }  
 }  
 public void sendMsg(Message message, String text) {  
 SendMessage sendMessage = new SendMessage();  
 sendMessage.enableMarkdown(true);  
 sendMessage.setChatId(message.getChatId().toString());  
 sendMessage.setReplyToMessageId(message.getMessageId());  
 sendMessage.setText(text);  
 try {  
 setButtons(sendMessage);  
 sendMessage(sendMessage);  
 } catch (TelegramApiException e) {  
 e.printStackTrace();  
 }  
 }  
 public void onUpdateReceived(Update update) {  
 Model model = new Model();  
 Message message = update.getMessage();  
 if (message != null && message.hasText()) {  
 switch (message.getText()) {  
 case "Что я могу?":  
 sendMsg(message, "Введи название города и я подскажу погоду))");  
 break;  
 case "/":  
 sendMsg(message, "Введи название города и я подскажу погоду))");  
 break;  
  
 case "/start":  
 sendMsg(message, "Введи название города и я подскажу погоду))");  
 break;  
 case "Кто Я":  
 sendMsg(message, "Я - бот, который подсказывает погоду в разных городах)");  
 break;  
 default:  
 try {  
 sendMsg(message, Weather.*getWeather*(message.getText(), model));  
 } catch (IOException e) {  
 sendMsg(message, "Город не найден!");  
 }  
 }  
 }  
 }  
 public void setButtons(SendMessage sendMessage) {  
 ReplyKeyboardMarkup replyKeyboardMarkup = new ReplyKeyboardMarkup();  
 sendMessage.setReplyMarkup(replyKeyboardMarkup);  
 replyKeyboardMarkup.setSelective(true);  
 replyKeyboardMarkup.setResizeKeyboard(true);  
 replyKeyboardMarkup.setOneTimeKeyboard(false);  
  
 List<KeyboardRow> keyboardRowList = new ArrayList<>();  
 KeyboardRow keyboardFirstRow = new KeyboardRow();  
  
 keyboardFirstRow.add(new KeyboardButton("Что я могу?"));  
 keyboardFirstRow.add(new KeyboardButton("Кто Я"));  
  
 keyboardRowList.add(keyboardFirstRow);  
 replyKeyboardMarkup.setKeyboard(keyboardRowList);  
  
 }  
 public String getBotUsername() {  
 return "RIZ\_Pogoda\_bot";  
 }  
 public String getBotToken() {  
 return "1852422426:AAGNEeYPA3w\_gAGlhsct5ookXznjr6TqA2E";  
 }  
}

## А.2 Класс Weather.java

import org.json.JSONArray;  
import org.json.JSONObject;  
import java.io.IOException;  
import java.io.InputStream;  
import java.net.URL;  
import java.util.Scanner;  
public class Weather {  
 public static String getWeather(String message, Model model) throws IOException {  
 URL url = new URL("http://api.openweathermap.org/data/2.5/weather?q=" + message + "&units=metric&appid=6fff53a641b9b9a799cfd6b079f5cd4e");  
 Scanner in = new Scanner((InputStream) url.getContent());  
 String result = "";  
 while (in.hasNext()) {  
 result += in.nextLine();  
 }  
 JSONObject object = new JSONObject(result);  
 model.setName(object.getString("name"));  
 JSONObject main = object.getJSONObject("main");  
 model.setTemp(main.getDouble("temp"));  
 model.setHumidity(main.getDouble("humidity"));  
 JSONArray getArray = object.getJSONArray("weather");  
 for (int i = 0; i < getArray.length(); i++) {  
 JSONObject obj = getArray.getJSONObject(i);  
 model.setIcon((String) obj.get("icon"));  
 model.setMain((String) obj.get("main"));  
 }  
 return "Город: " + model.getName() + "\n" +  
 "Температура: " + model.getTemp() + "C" + "\n" +  
 "Влажность: " + model.getHumidity() + "%" + "\n" +  
 "Небо: " + model.getMain() + "\n" ; } }

## А.3 Класс Model.java

public class Model {  
 private String name;  
 private Double temp;  
 private Double humidity;  
 private String icon;  
 private String main;  
 public String getName() {  
 return name;  
 }  
 public void setName(String name) {  
 this.name = name;  
 }  
 public Double getTemp() {  
 return temp;  
 }  
 public void setTemp(Double temp) {  
 this.temp = temp;  
 }  
 public Double getHumidity() {  
 return humidity;  
 }  
 public void setHumidity(Double humidity) {  
 this.humidity = humidity;  
 }  
 public String getIcon() {  
 return icon;  
 }  
 public void setIcon(String icon) {  
 this.icon = icon;  
 }  
 public String getMain() {  
 return main;  
 }  
 public void setMain(String main) {  
 this.main = main;  
 }  
}

ПРИЛОЖЕНИЕ Б  
(справочное)  
Скриншот с сайта Антиплагиат

Для проверки был использован ресурс antiplagiat.ru бесплатной версии, последнее обновление базы данных от 2021 года.

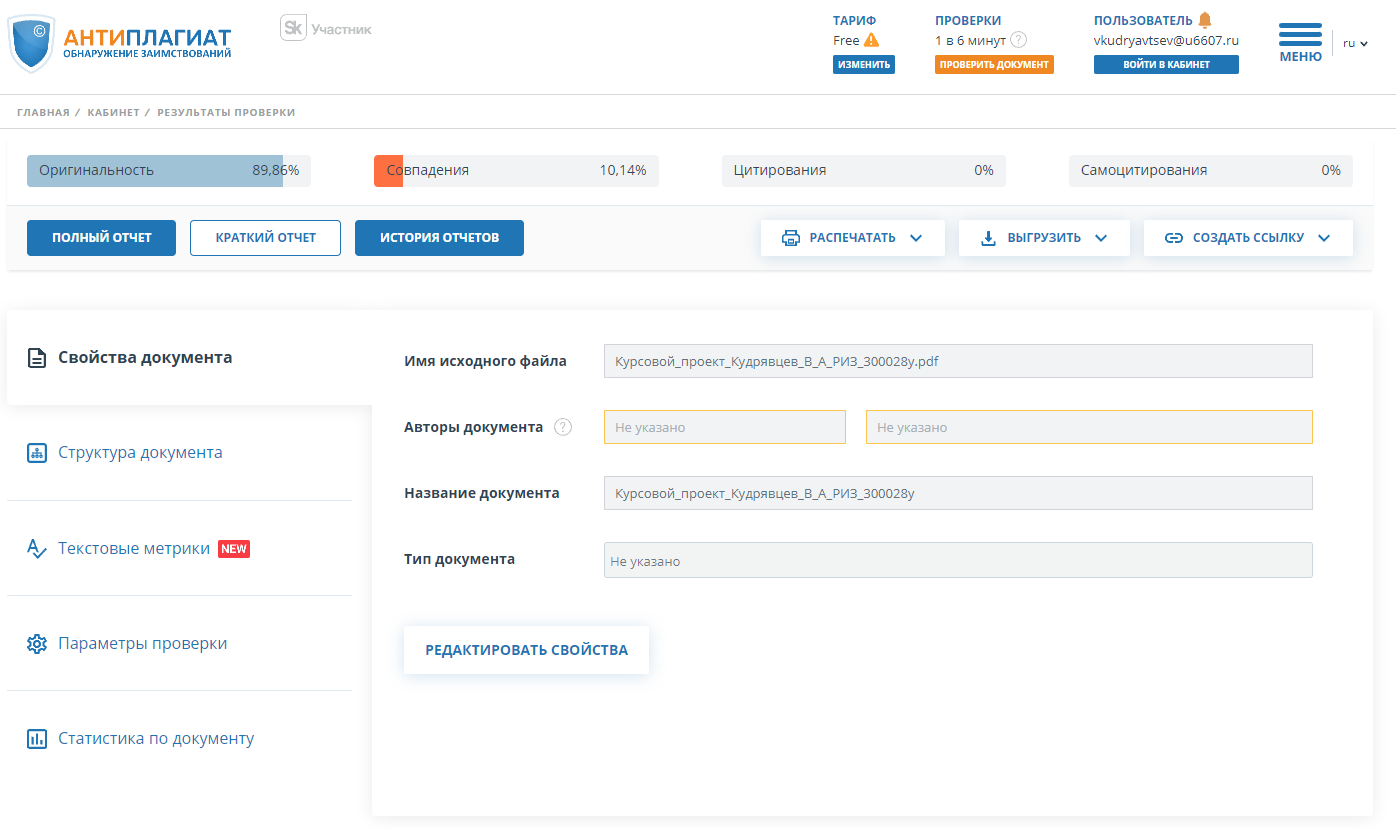


Рисунок 6 – Скриншот с сайта Антиплагиат