**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра САПР**

**ИДЗ**

**ПО ДИСЦИПЛИНЕ «ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ KOTLIN»**

|  |  |  |
| --- | --- | --- |
| Студент гр. 1302 |  | Новиков Г.В. |
| Студентка гр. 1302 |  | Романова О.В. |
| Преподаватель |  | Кулагин М.В. |

Санкт-Петербург

2023

# Разработка приложения для взаимодействия с внешними источниками

# 1.1. Задание

Напишите программу, которая с консоли считывает поисковый запрос пользователя, и выводит результат поиска по Википедии. После выбора нужной статьи программа должна открывать ее в браузере. Программа должна реагировать корректно на любой пользовательский ввод.

Задача разбивается на 5 этапов:

* + - 1. Считать введенные пользователем данные
      2. Сделать запрос к серверу
      3. Распарсить ответ
      4. Вывести результат поиска
      5. Открыть нужную страницу в браузере

Использовать готовые библиотеки для работы с Википедией нельзя.

# 1.2. Спецификация программы

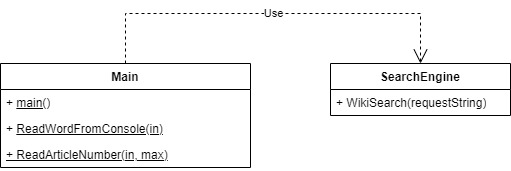


Рис. 1

Классы:

Main – главный класс, с помощью которого происходит запуск программы. Содержит методы ReadWordFromConsole (считывает вводимое пользователем слово из консоли), ReadArticleNumber (выбор заголовка для открытия википедии), main.

SearchEngine – обеспечивает поиск и запрос к серверу.

# 1.3. Описание интерфейса пользователя программы

Пользователь вводит запрос для поиска по Википедии (рис.2).



Рис. 2

В консоль выводятся результаты поиска – заголовки и json (рис.3).

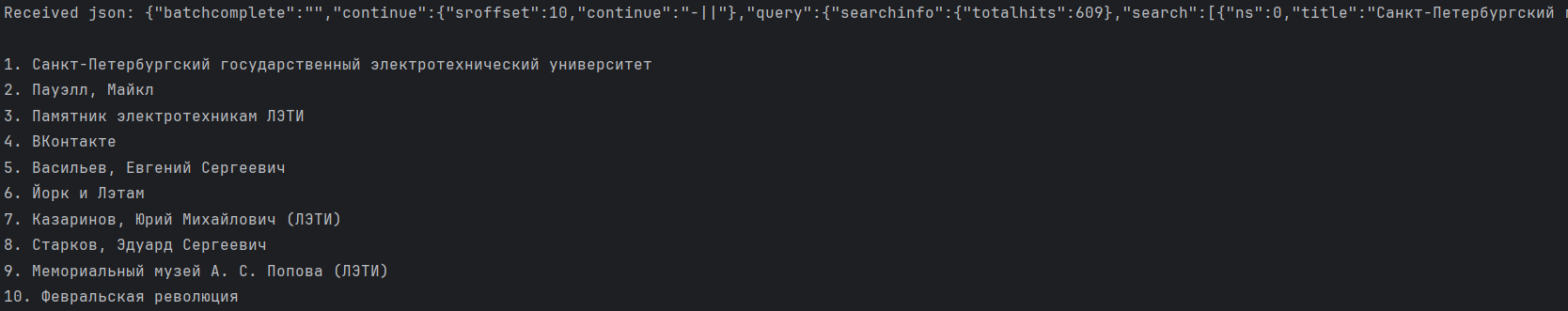


Рис. 3

Пользователь вводит номер заголовка, который необходимо открыть (рис.4).



Рис. 4

В консоль выводится URL и автоматически открывается ссылка (рис.5).



Рис. 5

# 1.4. Текст программы

Main.java:

import java.awt.Desktop;  
import java.io.IOException;  
import java.net.URISyntaxException;  
import java.util.Scanner;  
import com.google.gson.Gson;  
import com.google.gson.GsonBuilder;  
import com.google.gson.JsonObject;  
import com.google.gson.JsonArray;  
import java.net.URI;  
import java.util.regex.Pattern;  
  
public class Main  
{  
 private static String ReadWordFromConsole(Scanner in) {  
 String request;  
 while (true) {  
 System.*out*.print("Enter word: ");  
 request = in.nextLine();  
 if (request.isBlank()) {  
 System.*out*.println("Please enter word to search\n");  
 } else break;  
 }  
 return request;  
 }  
  
 public static int ReadArticleNumber(Scanner in, int max) {  
 String choice\_str;  
 int choice;  
  
 while (true) {  
 System.*out*.print("\nEnter article number: ");  
  
 choice\_str = in.nextLine().strip();  
 Pattern pattern = Pattern.*compile*("\\d+");  
 if (!choice\_str.isBlank() && pattern.matcher(choice\_str).matches()) {  
 choice = Integer.*parseInt*(choice\_str);  
 if (0 < choice && choice <= max) break;  
 }  
 System.*out*.println("Please enter number from 1 to " + max);  
 }  
 return choice;  
 }  
  
 public static void main(String[] args) throws IOException, URISyntaxException, InterruptedException {  
 Scanner in = new Scanner(System.*in*);  
  
 // Search and send request to the server  
 SearchEngine engine = new SearchEngine();  
 String result\_json;  
  
 String request;  
 while (true) {  
 try {  
 request = *ReadWordFromConsole*(in);  
 result\_json = engine.WikiSearch(request);  
 break;  
 } catch (IOException e) {  
 System.*out*.println("Error: cannot connect to Wikipedia. Please check your internet connection");  
 }  
 }  
  
 System.*out*.println("Received json: " + result\_json + "\n");  
  
 // Parsing  
 GsonBuilder builder = new GsonBuilder();  
 Gson gson = builder.create();  
  
 JsonObject json = gson.fromJson(result\_json, JsonObject.class);  
 JsonObject query = json.getAsJsonObject("query");  
 JsonArray result = query.getAsJsonArray("search");  
  
 if (result.isEmpty())  
 {  
 System.*out*.println("No results found");  
 System.*exit*(0);  
 }  
 // Titles output  
 String title;  
 for (int i = 0; i < result.size(); i++)  
 {  
 JsonObject titles = result.get(i).getAsJsonObject();  
 title = titles.get("title").getAsString();  
 System.*out*.println((i + 1) + ". " + title);  
 }  
  
 int choice = *ReadArticleNumber*(in, result.size());  
  
 // Getting pageId  
 JsonObject id = result.get(choice - 1).getAsJsonObject();  
 String pageId = id.get("pageid").getAsString();  
  
 // Open wiki  
 String wiki\_url = "https://ru.wikipedia.org/w/index.php?curid=" + pageId;  
 System.*out*.println("URL:" + wiki\_url);  
 Desktop.*getDesktop*().browse(new URI(wiki\_url));  
  
 in.close();  
 }  
}

SearchEngine.java:

import java.io.IOException;  
import java.net.URI;  
import java.net.URISyntaxException;  
import java.net.URLEncoder;  
import java.net.http.HttpClient;  
import java.net.http.HttpRequest;  
import java.net.http.HttpResponse;  
import java.net.http.HttpResponse.BodyHandlers;  
import java.nio.charset.StandardCharsets;  
  
public class SearchEngine  
{  
 public String WikiSearch(String requestString) throws IOException, InterruptedException {  
 URI uri;  
 try  
 {  
 uri = new URI("https://ru.wikipedia.org/w/api.php?action=query&list=search&utf8=&format=json&srsearch=" + URLEncoder.*encode*(requestString, StandardCharsets.*UTF\_8*));  
 } catch (URISyntaxException e)  
 {  
 throw new RuntimeException(e);  
 }  
 HttpRequest request = HttpRequest.*newBuilder*(uri).build();  
  
 HttpClient client = HttpClient.*newBuilder*()  
 .version(HttpClient.Version.*HTTP\_2*)  
 .build();  
  
 HttpResponse<String> response = client.send(request, BodyHandlers.*ofString*());  
  
 return response.body();  
 }  
}

# 1.5. Выводы

В ходе выполнения лабораторной работы была реализована программа для поиска на википедии через консоль. В работе использовались библиотека Gson и встроенный класс Desktop. Запросы осуществлялись с помощью API википедии и встроенных средств java для веб-запросов (HttpClient, HttpRequest, HttpResponse). Библиотека Gson использовалась для преобразования JSON, класс Desktop – для автоматического открытия статьи Википедии в Google.

# Разработка приложения для обработки данных

# 2.1. Задание

Даны 2 файла-справочника городов. Один файл в формате xml, другой в формате csv.

Необходимо разработать консольное приложение для работы с ними.

После запуска приложение ожидает ввода пути до файла-справочника либо команды на завершение работы (какая-то комбинация клавиш).

По команде завершения работы приложение завершает свою работу.

После ввода пути до файла-справочника приложение формирует сводную статистику:

1) Отображает дублирующиеся записи с количеством повторений.

2) Отображает, сколько в каждом городе: 1, 2, 3, 4 и 5-этажных зданий.

3) Показывает время обработки файла.

После вывода статистики приложение снова ожидает ввода пути до файла-справочника либо команды на завершение работы.

В процессе работы приложение падать не должно, выход только по команде на завершение работы.

# 2.2. Спецификация программы

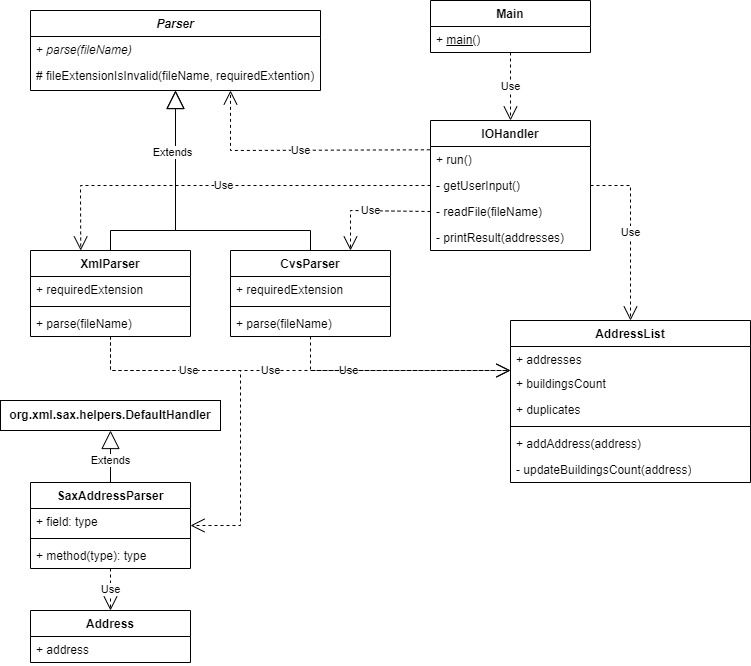


рис. 6

Address – адрес дома.

AddressList – класс для храниния адресов. Содержит методы: addAddress (добавляет адреса домов и дубликаты), updateBuildingsCount (обновляет количество адресов).

IOHandler – Класс для взаимодействия с пользователем. Содержит методы: run, getUserInput (считывает пользовательский ввод из консоли), readFile (считывает путь и название файла), printResult (выводит результаты в консоль).

Main – главный класс, с помощью которого происходит запуск программы.

Parser – абстрактный класс для парсеров.

CsvParser – парсер для csv. Наследует Parser.

SaxAddressParser – Вспомогательный класс для парсинга xml. Наследует org.xml.sax.helpers.DefaultHandler.

XmlParser – парсер xml. Наследует Parser.

# 2.3. Описание интерфейса пользователя программы

Пользователь вводит запрос: адрес файла или слово для выхода (рис.7).

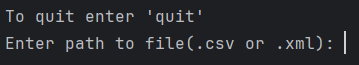


Рис. 7

При вводе адреса в консоль выводится результат обработки файла (дубликаты и количество 1, 2, 3, 4 и 5-этажных зданий) и время обработки файла (рис.8). После этого снова выводится запрос на адрес или выход (рис.9).

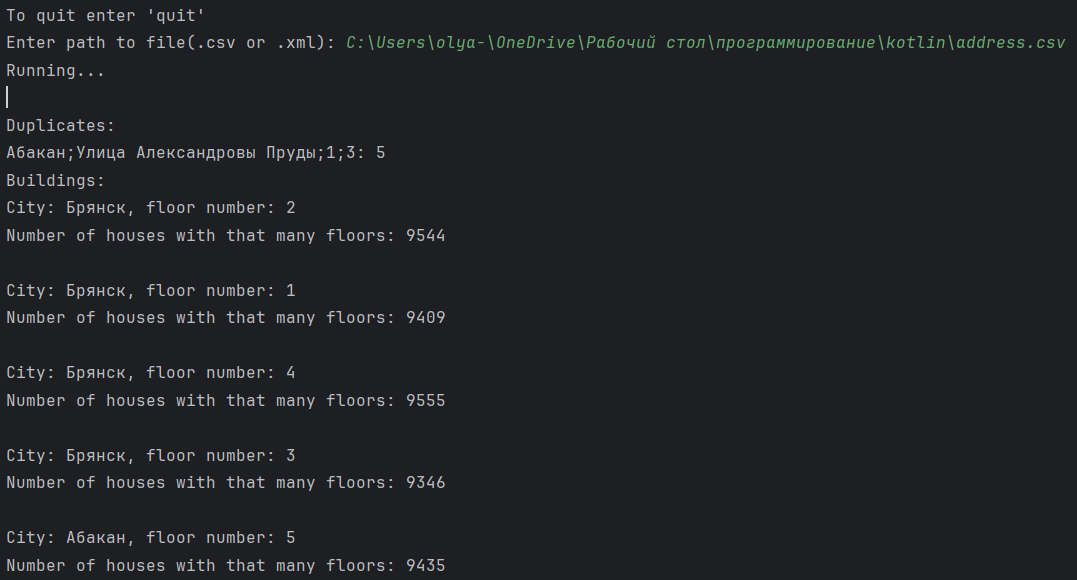


Рис. 8 – дубликаты и количество 1, 2, 3, 4 и 5-этажных зданий

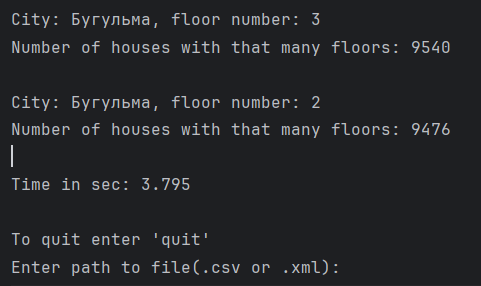


Рис. 9 – время обработки и новый запрос

При вводе слова “quit” программа завершается (рис.10).

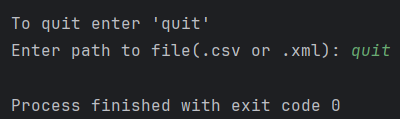


Рис. 10

# 2.4. Текст программы

Main.java:

public class Main {  
 public static void main(String[] args) {  
 IOHandler ioHandler = new IOHandler();  
 ioHandler.run();  
 }  
}

Address.java:

public class Address {  
 private String address;  
 public String getAddress() {  
 return address;  
 }  
 public void setAddress(String address) {  
 this.address = address;  
 }  
}

AddressList.java:

import java.util.HashMap;  
  
public class AddressList {  
 public HashMap<String, Integer> addresses = new HashMap<>();  
 public HashMap<String, Integer> buildingsCount = new HashMap<>();  
 public HashMap<String, Integer> duplicates = new HashMap<>();  
  
 public void addAddress(String address) {  
 updateBuildingsCount(address);  
  
 if (addresses.containsKey(address)) {  
 addresses.put(address, addresses.get(address) + 1);  
 duplicates.put(address, addresses.get(address));  
 }  
 else {  
 addresses.put(address, 1);  
 }  
 }  
  
 private void updateBuildingsCount(String address) {  
 String key = address.substring(0, address.indexOf(";")) +  
 address.substring(address.lastIndexOf(";"));  
 buildingsCount.put(key, buildingsCount.getOrDefault(key, 0) + 1);  
 }  
}

CsvParser.java:

import java.io.BufferedReader;  
import java.io.FileReader;  
import java.io.IOException;  
  
public class CsvParser extends Parser {  
 public String requiredExtension = "csv";  
  
 @Override  
 public AddressList parse(String fileName) throws IllegalArgumentException, IOException {  
 if (fileExtensionIsInvalid(fileName, requiredExtension)) {  
 throw new IllegalArgumentException();  
 }  
  
 AddressList addresses = new AddressList();  
 BufferedReader reader = new BufferedReader(new FileReader(fileName));  
  
 String line;  
 line = reader.readLine();  
  
 while ((line = reader.readLine()) != null) {  
 addresses.addAddress(line.replaceAll("\"", ""));  
 }  
  
 return addresses;  
 }  
}

IOHandler.java:

import java.io.\*;  
import java.util.Scanner;  
  
public class IOHandler {  
  
 */\*\* Starts loop that takes from user a path to scv or xml file, analyses it and writes result to console  
 'quit' exits the loop \*/* public void run() {  
 AddressList addresses;  
  
 String input = getUserInput();  
 while (!input.equals("quit")) {  
 System.*out*.println("Running...\n");  
 try {  
 long startTime = System.*currentTimeMillis*();  
  
 addresses = readFile(input);  
 printResult(addresses);  
  
 long endTime = System.*currentTimeMillis*();  
 long time = endTime - startTime;  
 System.*out*.println("Time in sec: " + (time \* 0.001));  
 } catch (IllegalArgumentException e) {  
 System.*out*.println("Illegal filename");  
 } catch (IOException e) {  
 System.*out*.println("File not found");  
 }  
  
 input = getUserInput();  
 }  
 }  
  
 private String getUserInput()  
 {  
 System.*out*.print("\nTo quit enter 'quit'\n");  
 System.*out*.print("Enter path to file(.csv or .xml): ");  
  
 Scanner in = new Scanner(System.*in*);  
  
 return in.nextLine();  
 }  
  
 private AddressList readFile(String fileName) throws IllegalArgumentException, IOException {  
 int i = fileName.lastIndexOf('.');  
 if (i == -1) {  
 throw new IllegalArgumentException();  
 }  
 String extension = fileName.substring(i + 1);  
  
 Parser parser = switch (extension) {  
 case "csv" -> new CsvParser();  
 case "xml" -> new XmlParser();  
 default -> throw new IllegalArgumentException();  
 };  
  
 return parser.parse(fileName);  
 }  
  
 private void printResult(AddressList addresses) {  
 System.*out*.println("Duplicates:");  
 for (String address : addresses.duplicates.keySet()) {  
 System.*out*.println(address + ": " + addresses.duplicates.get(address));  
 }  
 System.*out*.println("Buildings:");  
 for (String key : addresses.buildingsCount.keySet()) {  
 String city = key.split(";")[0];  
 String floor = key.split(";")[1];  
 System.*out*.println("City: " + city + ", floor number: " + floor);  
 System.*out*.println("Number of houses with that many floors: " + addresses.buildingsCount.get(key) + "\n");  
 }  
 }  
}

Parser.java:

import java.io.IOException;  
  
public abstract class Parser {  
 public abstract AddressList parse(String fileName) throws IllegalArgumentException, IOException;  
  
 protected boolean fileExtensionIsInvalid(String fileName, String requiredExtension) {  
 int i = fileName.lastIndexOf('.');  
 String extension;  
 if (i != -1) {  
 extension = fileName.substring(i + 1);  
 } else {  
 return true;  
 }  
  
 return !extension.equals(requiredExtension);  
 }  
}

SaxAddressParser.java:

import org.xml.sax.Attributes;  
import org.xml.sax.helpers.DefaultHandler;  
  
import java.util.ArrayList;  
  
public class SaxAddressParser extends DefaultHandler {  
 private final ArrayList<Address> addressList = new ArrayList<>();  
 private Address address = null;  
  
 public ArrayList<Address> getAddressList() {  
 return addressList;  
 }  
  
 @Override  
 public void startElement(String uri, String localName, String qName, Attributes attributes) {  
 if (qName.equals("item")) {  
 address = new Address();  
 String city = attributes.getValue("city");  
 String street = attributes.getValue("street");  
 String house = attributes.getValue("house");  
 String floor = attributes.getValue("floor");  
  
 String sb = city + ";" + street + ";" + house + ";" + floor;  
 address.setAddress(sb);  
 }  
 }  
  
 @Override  
 public void endElement(String uri, String localName, String qName) {  
 if (qName.equals("item")) {  
 addressList.add(address);  
 }  
 }  
}

XmlParser.java:

import org.xml.sax.SAXException;  
  
import javax.xml.parsers.\*;  
import java.io.File;  
import java.io.IOException;  
import java.util.ArrayList;  
  
public class XmlParser extends Parser {  
 public String requiredExtension = "xml";  
  
 @Override  
 public AddressList parse(String fileName) throws IllegalArgumentException, IOException {  
 if (fileExtensionIsInvalid(fileName, requiredExtension)) {  
 throw new IllegalArgumentException();  
 }  
  
 AddressList addresses = new AddressList();  
  
 SAXParserFactory saxParserFactory = SAXParserFactory.*newInstance*();  
 try {  
 SAXParser saxParser = saxParserFactory.newSAXParser();  
 SaxAddressParser handler = new SaxAddressParser();  
 saxParser.parse(new File(fileName), handler);  
 ArrayList<Address> addressList = handler.getAddressList();  
  
 for (Address address : addressList) {  
 addresses.addAddress(address.getAddress());  
 }  
 } catch (ParserConfigurationException | SAXException e) {  
 throw new RuntimeException(e);  
 }  
  
 return addresses;  
 }  
}

# 2.5. Выводы

В ходе выполнения лабораторной работы мы поработали с файлами формата csv и xml. Для обработки использовались hashmap. Были использованы библиотеки, связанные с чтением файлов и hashmap.

# Разработка серверного приложения

# 3.1. Задание

В рамках лабораторной работы требуется реализовать программу для взаимодействия с пользователями в Telegram.

Вариант 1 - Бот с расписанием ЛЭТИ

Целью всей работы является создание бота, который бы позволил получить расписание занятий для любой группы. Бот должен понимать следующие команды:

* near\_lesson GROUP\_NUMBER - ближайшее занятие для указанной группы;
* DAY WEEK\_NUMBER GROUP\_NUMBER - расписание занятий в указанный день (monday, thuesday, ...). Неделя может быть четной, нечетной;
* tommorow GROUP\_NUMBER - расписание на следующий день (если это воскресенье, то выводится расписание на понедельник, учитывая, что неделя может быть четной или нечетной);
* all WEEK\_NUMBER GROUP\_NUMBER - расписание на всю неделю.

Для работы с расписанием необходимо использовать API университета:

<https://digital.etu.ru/api/docs-public/>

# 3.2. Спецификация программы

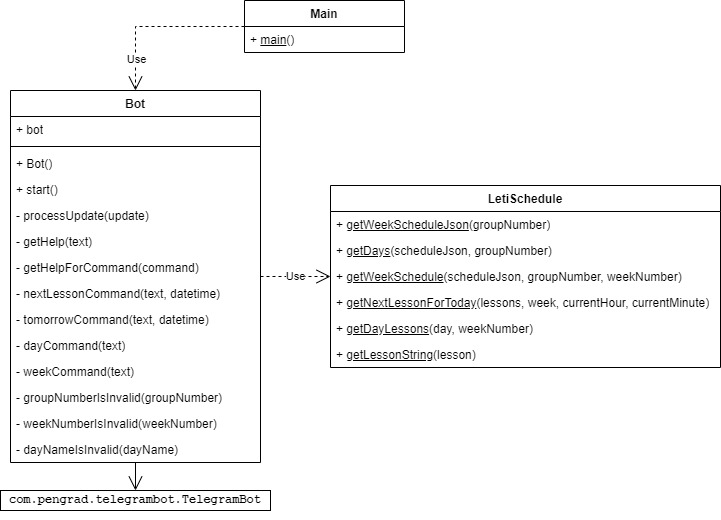


Рис. 11

Main – главный класс, с помощью которого происходит запуск программы.

Bot – обеспечивает работу бота. Содержит методы: Bot, start, processUpdate (установка времени и запуск метода в зависимости от запроса), getHelp, getHelpForCommand, nextLessonCommand, tomorrowCommand, dayCommand, weekCommand, groupNumberIsInvalid, weekNumberIsInvalid, dayNameIsInvalid.

LetiSchedule – обеспечивает поиск на сайте Лэти. Содержит методы: getWeekScheduleJson, getDays, getWeekSchedule, getNextLessonForToday, getDayLessons, getLessonString.

# 3.3. Описание интерфейса пользователя программы

Команды:

* + - 1. next for {group} – следующая пара для группы
      2. tomorrow for {group} – расписание на следующий день для группы

3. week {week} for {group} – расписание на указанную неделю для группы

4. day {day of week} week {week} for {group} – расписание на день (указывается название дня, например “Monday”) с указанием недели для группы

5. /help или /help {command name} – справка или справка для конкретной команды

Номер группы {group} должен состоять из 4 цифр, если группа не найдена, бот присылает сообщение об этом.

Неделя {week} – это одна цифра – либо “1” для нечетной, либо “2” для четной.

День {day of week} – один из следующих вариантов: "monday", "tuesday", "wednesday", "thursday", "friday", "saturday", "sunday" в любом регистре.

Примеры:

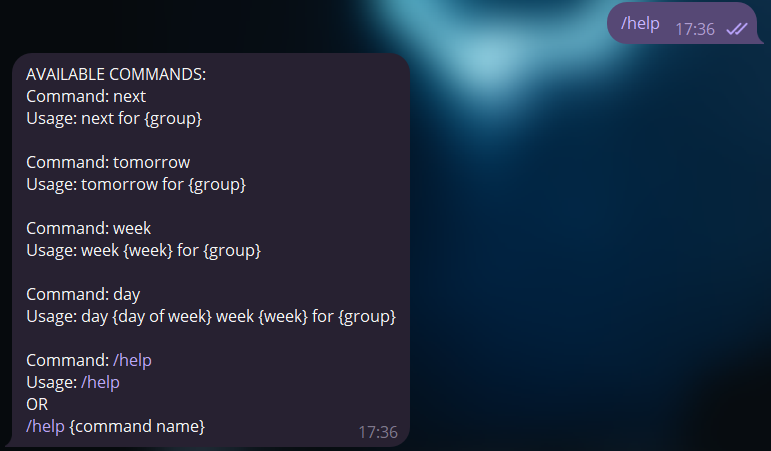


Рис. 12

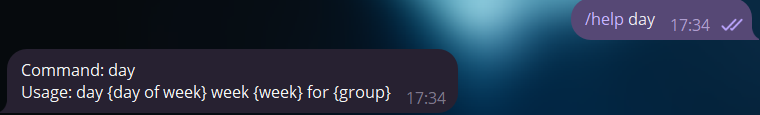


Рис. 13



Рис. 14. Запрос на завтрашнее расписание, сделанное в субботу

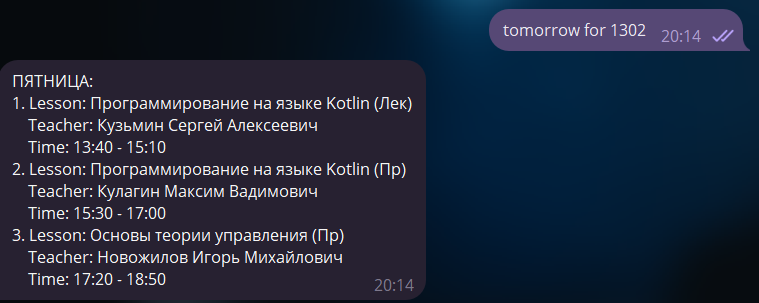


Рис. 15. Запрос на завтрашнее расписание, сделанное в четверг

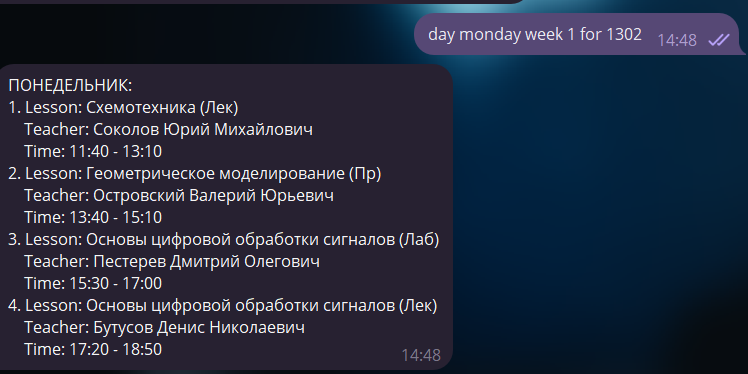


Рис. 16



Рис. 17

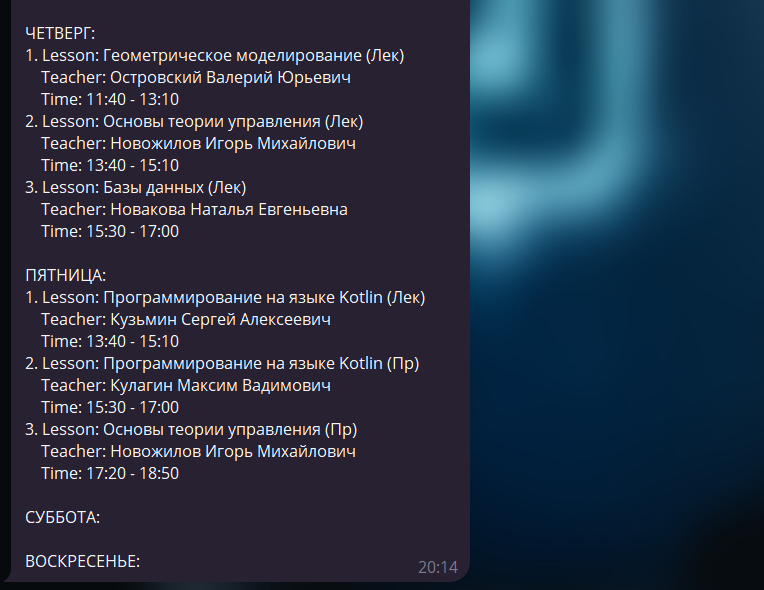


Рис. 18

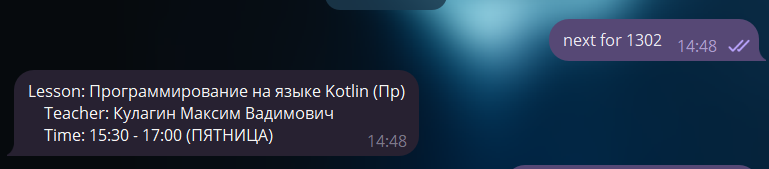


Рис. 19

# 3.4. Текст программы

Main.java:

package org.example;

public class Main {

public static void main(String[] args) {

Bot bot = new Bot();

bot.start();

}

}

LetiSchedule.java:

package org.example;

import com.google.gson.Gson;

import com.google.gson.GsonBuilder;

import com.google.gson.JsonArray;

import com.google.gson.JsonObject;

import okhttp3.HttpUrl;

import okhttp3.OkHttpClient;

import okhttp3.Request;

import okhttp3.Response;

import java.io.IOException;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Calendar;

import java.util.List;

public class LetiSchedule {

public static String getWeekScheduleJson(String groupNumber) {

HttpUrl httpUrl = new HttpUrl.Builder()

.scheme("https")

.host("digital.etu.ru")

.addPathSegment("api")

.addPathSegment("mobile")

.addPathSegment("schedule")

.addQueryParameter("groupNumber", groupNumber)

.build();

Request requestHttp = new Request.Builder()

.url(httpUrl)

.build();

System.out.println("ULR: " + requestHttp);

OkHttpClient httpClient = new OkHttpClient();

try (Response response = httpClient.newCall(requestHttp).execute()) {

if (response.body() != null) {

return response.body().string();

}

} catch (IOException e) {

System.out.println("Error: " + e.getMessage());

}

return "";

}

public static JsonObject getDays(String scheduleJson, String groupNumber) {

Gson gson = new GsonBuilder().create();

return gson.fromJson(scheduleJson, JsonObject.class)

.get(groupNumber).getAsJsonObject()

.get("days").getAsJsonObject();

}

public static String getWeekSchedule(String scheduleJson, String groupNumber, String weekNumber) {

JsonObject days = getDays(scheduleJson, groupNumber);

StringBuilder schedule = new StringBuilder();

JsonObject day;

for (int i = 0; i < 7; i++) {

day = days.get(Integer.toString(i)).getAsJsonObject();

schedule

.append(getDayLessons(day, weekNumber))

.append("\n");

}

return schedule.toString();

}

public static String getNextLessonForToday(JsonArray lessons, String week, byte currentHour, byte currentMinute) {

SimpleDateFormat sdf;

Calendar tmp = new Calendar.Builder().build();

JsonObject lesson;

byte lessonHour;

byte lessonMinute;

for (int i = 0; i < lessons.size(); i++) {

lesson = lessons.get(i).getAsJsonObject();

if (lesson.getAsJsonObject().get("week").getAsString().equals(week)) {

sdf = new SimpleDateFormat("HH:mm");

try {

tmp.setTime(sdf.parse(lesson.get("start\_time").getAsString()));

} catch (ParseException e) {

throw new RuntimeException(e);

}

lessonHour = (byte) tmp.get(Calendar.HOUR\_OF\_DAY);

lessonMinute = (byte) tmp.get(Calendar.HOUR\_OF\_DAY);

if (lessonHour > currentHour || (lessonHour == currentHour && lessonMinute > currentMinute)) {

return getLessonString(lesson);

}

}

}

return "";

}

public static String getDayLessons(JsonObject day, String weekNumber) {

StringBuilder schedule = new StringBuilder();

JsonArray lessons = day.get("lessons").getAsJsonArray();

JsonObject lesson;

String dayName = day.get("name").toString();

schedule.append(dayName.replace("\"", ""))

.append(":\n");

byte lessonNumber = 1;

for (int i = 0; i < lessons.size(); i++) {

lesson = lessons.get(i).getAsJsonObject();

if (lesson.getAsJsonObject().get("week").getAsString().equals(weekNumber)) {

schedule.append(lessonNumber).append(". ")

.append(getLessonString(lesson))

.append("\n");

lessonNumber++;

}

}

return schedule.toString();

}

public static String getLessonString(JsonObject lesson) {

StringBuilder schedule = new StringBuilder();

String name = lesson.get("name").getAsString();

String subjectType = lesson.get("subjectType").getAsString();

String teacher = lesson.get("teacher").getAsString();

String start = lesson.get("start\_time").getAsString();

String end = lesson.get("end\_time").getAsString();

schedule.append("Lesson: ")

.append(name.replace("\"", ""))

.append(" (")

.append(subjectType.replace("\"", ""))

.append(")")

.append("\n Teacher: ")

.append(teacher.replace("\"", ""))

.append("\n Time: ")

.append(start.replace("\"", ""))

.append(" - ")

.append(end.replace("\"", ""));

return schedule.toString();

}

}

Bot.java:

package org.example;

import com.google.gson.\*;

import com.pengrad.telegrambot.TelegramBot;

import com.pengrad.telegrambot.UpdatesListener;

import com.pengrad.telegrambot.model.Update;

import com.pengrad.telegrambot.request.SendMessage;

import com.pengrad.telegrambot.response.SendResponse;

import java.util.Calendar;

import java.util.HashMap;

import java.util.List;

import java.util.TimeZone;

public class Bot {

TelegramBot bot;

String TIMEZONE = "Europe/Moscow";

String INVALID\_COMMAND\_MESSAGE = "Invalid command. Try again";

String INVALID\_GROUP\_NUMBER\_MESSAGE = "Group number must consist of 4 digits";

String INVALID\_WEEK\_NUMBER\_MESSAGE = "Week number must be either 1 for odd or 2 for even";

String INVALID\_DAY\_NAME\_MESSAGE = "Invalid day name. \nVariants: Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday";

List<String> LIST\_OF\_COMMANDS = List.of(new String[] {

"next",

"tomorrow",

"week",

"day",

"/help"

});

HashMap<String, String> COMMAND\_USAGES = new HashMap<>();

List<String> daysOfWeek = List.of(new String[] {

"monday",

"tuesday",

"wednesday",

"thursday",

"friday",

"saturday",

"sunday"

});

public Bot() {

String token = System.getenv("LetiSkedHeadBotToken");

if (token == null) {

throw new RuntimeException("Token is not found in environment variable 'LetiSkedHeadBotToken'");

}

bot = new TelegramBot(token);

COMMAND\_USAGES.put("next", "next for {group}");

COMMAND\_USAGES.put("tomorrow", "tomorrow for {group}");

COMMAND\_USAGES.put("week", "week {week} for {group}");

COMMAND\_USAGES.put("day", "day {day of week} week {week} for {group}");

COMMAND\_USAGES.put("/help", "/help \nOR\n/help {command name}");

}

public void start() {

bot.setUpdatesListener(updates -> {

for (Update update : updates) {

processUpdate(update);

}

return UpdatesListener.CONFIRMED\_UPDATES\_ALL;

});

}

private void processUpdate(Update update) {

if (update.message() == null) return;

String text = update.message().text().trim();

Calendar datetime = new Calendar.Builder()

.setInstant(update.message().date() \* 1000L)

.setTimeZone(TimeZone.getTimeZone(TIMEZONE))

.build();

String response\_text = INVALID\_COMMAND\_MESSAGE;

if (text.matches("next for [0-9]+")) {

response\_text = nextLessonCommand(text, datetime);

} else if (text.matches("day [a-zA-Z]+ week [0-9]+ for [0-9]+")) {

response\_text = dayCommand(text);

} else if (text.matches("tomorrow for [0-9]+")) {

response\_text = tomorrowCommand(text, datetime);

} else if (text.matches("week [0-9]+ for [0-9]+")) {

response\_text = weekCommand(text);

} else if (text.matches("/help( [a-zA-Z]+)?")) {

response\_text = getHelp(text);

}

System.out.println(response\_text);

long chatId = update.message().chat().id();

SendResponse response = bot.execute(new SendMessage(chatId, response\_text));

if (!response.isOk()) {

System.out.println("Telegram api connection error: " + response.errorCode());

}

}

private String getHelp(String text) {

String[] words = text.split("\\s+");

if (words.length == 2) {

return getHelpForCommand(words[1]);

}

StringBuilder helpText = new StringBuilder();

helpText.append("AVAILABLE COMMANDS:\n");

for (String command : LIST\_OF\_COMMANDS) {

helpText.append(getHelpForCommand(command))

.append("\n");

}

return helpText.toString();

}

private String getHelpForCommand(String command) {

if (!LIST\_OF\_COMMANDS.contains(command)) return INVALID\_COMMAND\_MESSAGE;

return "Command: " + command + "\nUsage: " + COMMAND\_USAGES.get(command) + "\n";

}

private String nextLessonCommand(String text, Calendar datetime) {

String[] words = text.split("\\s+");

String groupNumber = words[2];

if (groupNumberIsInvalid(groupNumber)) return INVALID\_GROUP\_NUMBER\_MESSAGE;

String week = String.valueOf(datetime.get(Calendar.WEEK\_OF\_YEAR) % 2);

String today = String.valueOf((datetime.get(Calendar.DAY\_OF\_WEEK) - 2));

String scheduleJson = LetiSchedule.getWeekScheduleJson(groupNumber);

if (scheduleJson.equals("{}")) return "Cannot find group " + groupNumber;

JsonObject days = LetiSchedule.getDays(scheduleJson, groupNumber);

byte currentHour = (byte) datetime.get(Calendar.HOUR\_OF\_DAY);

byte currentMinute = (byte) datetime.get(Calendar.MINUTE);

String nextLesson = "";

String result = "";

int daysToSearch = 14;

for (int i = 0; i < daysToSearch && nextLesson.isEmpty(); i++) {

JsonObject day = days.get(today).getAsJsonObject();

JsonArray lessons = day.get("lessons").getAsJsonArray();

nextLesson = LetiSchedule.getNextLessonForToday(lessons, week, currentHour, currentMinute);

result = nextLesson + " (" + day.get("name").getAsString() + ")";

today = String.valueOf((Integer.parseInt(today) + 1));

if (today.equals("7")) {

today = "0";

week = week.equals("1") ? "2" : "1";

}

currentHour = 0;

currentMinute = 0;

}

System.out.println(result);

return result;

}

private String tomorrowCommand(String text, Calendar datetime) {

String[] words = text.split("\\s+");

String groupNumber = words[2];

if (groupNumberIsInvalid(groupNumber)) return INVALID\_GROUP\_NUMBER\_MESSAGE;

String week = String.valueOf(datetime.get(Calendar.WEEK\_OF\_YEAR) % 2);

String tomorrow = String.valueOf((datetime.get(Calendar.DAY\_OF\_WEEK) - 1));

if (tomorrow.equals("0")) {

week = week.equals("1") ? "2" : "1";

}

String scheduleJson = LetiSchedule.getWeekScheduleJson(groupNumber);

if (scheduleJson.equals("{}")) return "Cannot find group " + groupNumber;

JsonObject days = LetiSchedule.getDays(scheduleJson, groupNumber);

JsonObject day = days.get(tomorrow).getAsJsonObject();

return LetiSchedule.getDayLessons(day, week);

}

private String dayCommand(String text) {

String[] words = text.split("\\s+");

String dayName = words[1].toLowerCase();

if (dayNameIsInvalid(dayName)) return INVALID\_DAY\_NAME\_MESSAGE;

String weekNumber = words[3];

if (weekNumberIsInvalid(weekNumber)) return INVALID\_WEEK\_NUMBER\_MESSAGE;

String groupNumber = words[5];

if (groupNumberIsInvalid(groupNumber)) return INVALID\_GROUP\_NUMBER\_MESSAGE;

String scheduleJson = LetiSchedule.getWeekScheduleJson(groupNumber);

if (scheduleJson.equals("{}")) return "Cannot find group " + groupNumber;

JsonObject days = LetiSchedule.getDays(scheduleJson, groupNumber);

JsonObject day = days.get(Integer.toString(daysOfWeek.indexOf(dayName))).getAsJsonObject();

return LetiSchedule.getDayLessons(day, weekNumber);

}

private String weekCommand(String text) {

String[] words = text.split("\\s+");

String weekNumber = words[1];

if (weekNumberIsInvalid(weekNumber)) return INVALID\_WEEK\_NUMBER\_MESSAGE;

String groupNumber = words[3];

if (groupNumberIsInvalid(groupNumber)) return INVALID\_GROUP\_NUMBER\_MESSAGE;

String scheduleJson = LetiSchedule.getWeekScheduleJson(groupNumber);

if (scheduleJson.equals("{}")) return "Cannot find group " + groupNumber;

return LetiSchedule.getWeekSchedule(scheduleJson, groupNumber, weekNumber);

}

private boolean groupNumberIsInvalid(String groupNumber) {

return groupNumber.length() != 4;

}

private boolean weekNumberIsInvalid(String weekNumber) {

return !weekNumber.matches("[1-2]");

}

private boolean dayNameIsInvalid(String dayName) {

return !daysOfWeek.contains(dayName);

}

}

# 3.5. Выводы

В данной работе был создан телеграм-бот для вывода расписания пар университета ЛЭТИ, которое было получено с помощью специального API. Были реализованы 5 команд: справка, расписание на неделю, расписание на завтра, следующая пара и расписание на конкретный день.

# 4. Подготовка модели данных и подключение СУБД

# 4.1. Задание

В рамках практической работы требуется реализовать взаимодействие разработанного бота с таблицами в базе данных.

# 4.2. Спецификация программы

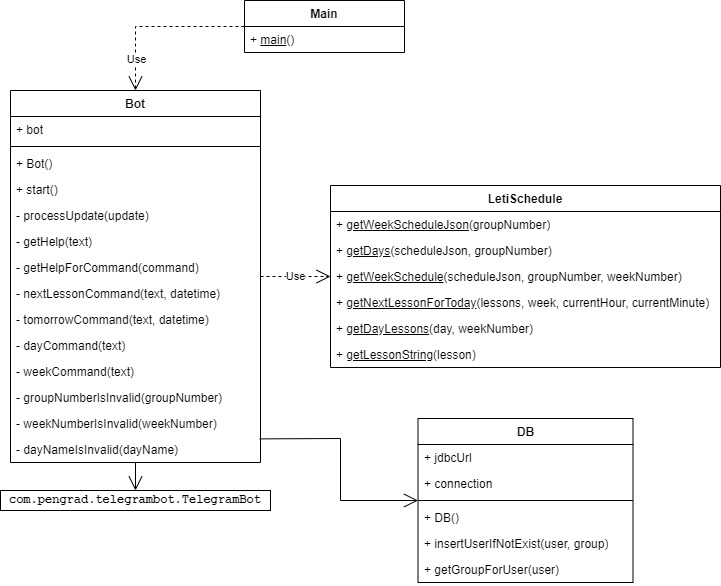


Рис. 20. Диаграмма классов

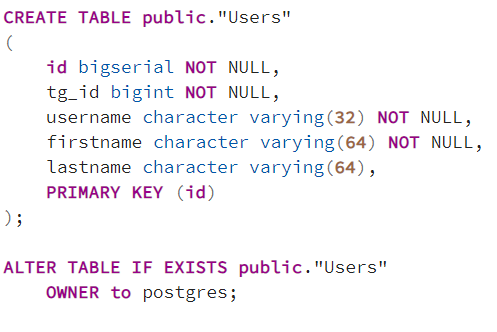


Рис. 21

Main – главный класс, с помощью которого происходит запуск программы.

Bot – обеспечивает работу бота. Содержит методы: Bot, start, processUpdate (установка времени и запуск метода в зависимости от запроса), getHelp, getHelpForCommand, nextLessonCommand, tomorrowCommand, dayCommand, weekCommand, groupNumberIsInvalid, weekNumberIsInvalid, dayNameIsInvalid.

LetiSchedule – обеспечивает поиск на сайте Лэти. Содержит методы: getWeekScheduleJson, getDays, getWeekSchedule, getNextLessonForToday, getDayLessons, getLessonString.

DB – обеспечивает взаимодействие с базой данных.

# 4.3. Описание интерфейса пользователя программы

Команды:

* + - 1. next for {group} OR next– следующая пара для группы (при нахождении пользователя в базе данных можно не вводить номер группы)
      2. tomorrow for {group} OR tomorrow – расписание на следующий день для группы (при нахождении пользователя в базе данных можно не вводить номер группы)
      3. week {week} for {group} OR week {week} – расписание на указанную неделю для группы (при нахождении пользователя в базе данных можно не вводить номер группы)
      4. day {day of week} week {week} for {group} OR day {day of week} week {week} – расписание на день (указывается название дня, например “Monday”) с указанием недели для группы (при нахождении пользователя в базе данных можно не вводить номер группы)
      5. /help или /help {command name} – справка или справка для конкретной команды (при нахождении пользователя в базе данных можно не вводить номер группы)

Номер группы {group} должен состоять из 4 цифр, если группа не найдена, бот присылает сообщение об этом.

Неделя {week} – это одна цифра – либо “1” для нечетной, либо “2” для четной.

День {day of week} – один из следующих вариантов: "monday", "tuesday", "wednesday", "thursday", "friday", "saturday", "sunday" в любом регистре.

После того, как пользователь вызвал команду с указанием группы, он записывается в базу данных. После этого он может не указывать группу при вызове команды. Если пользователь снова вызывает команду с указанием группы, его группа в БД обновляется.

Примеры:

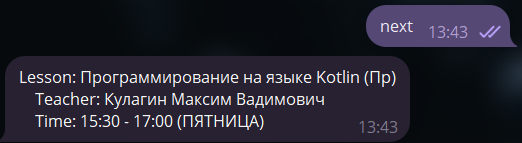
****

Рис. 22. Запрос пользователя, находящегося в базе

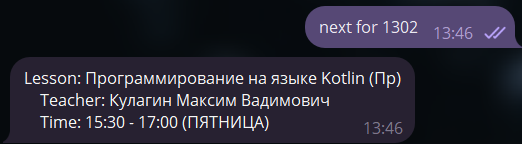


Рис. 23. Запрос пользователя, не находящегося в базе

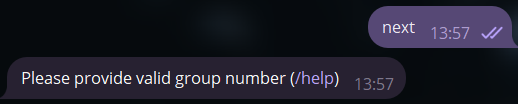


Рис. 24. Запрос пользователя, не находящегося в базе

# 4.4. Текст программы

Main.java:

package org.example;

public class Main {

public static void main(String[] args) {

Bot bot = new Bot();

bot.start();

}

}

LetiSchedule.java:

package org.example;

import com.google.gson.Gson;

import com.google.gson.GsonBuilder;

import com.google.gson.JsonArray;

import com.google.gson.JsonObject;

import okhttp3.HttpUrl;

import okhttp3.OkHttpClient;

import okhttp3.Request;

import okhttp3.Response;

import java.io.IOException;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Calendar;

import java.util.List;

public class LetiSchedule {

public static String getWeekScheduleJson(String groupNumber) {

HttpUrl httpUrl = new HttpUrl.Builder()

.scheme("https")

.host("digital.etu.ru")

.addPathSegment("api")

.addPathSegment("mobile")

.addPathSegment("schedule")

.addQueryParameter("groupNumber", groupNumber)

.build();

Request requestHttp = new Request.Builder()

.url(httpUrl)

.build();

System.out.println("ULR: " + requestHttp);

OkHttpClient httpClient = new OkHttpClient();

try (Response response = httpClient.newCall(requestHttp).execute()) {

if (response.body() != null) {

return response.body().string();

}

} catch (IOException e) {

System.out.println("Error: " + e.getMessage());

}

return "";

}

public static JsonObject getDays(String scheduleJson, String groupNumber) {

Gson gson = new GsonBuilder().create();

return gson.fromJson(scheduleJson, JsonObject.class)

.get(groupNumber).getAsJsonObject()

.get("days").getAsJsonObject();

}

public static String getWeekSchedule(String scheduleJson, String groupNumber, String weekNumber) {

JsonObject days = getDays(scheduleJson, groupNumber);

StringBuilder schedule = new StringBuilder();

JsonObject day;

for (int i = 0; i < 7; i++) {

day = days.get(Integer.toString(i)).getAsJsonObject();

schedule

.append(getDayLessons(day, weekNumber))

.append("\n");

}

return schedule.toString();

}

public static String getNextLessonForToday(JsonArray lessons, String week, byte currentHour, byte currentMinute) {

SimpleDateFormat sdf;

Calendar tmp = new Calendar.Builder().build();

JsonObject lesson;

byte lessonHour;

byte lessonMinute;

for (int i = 0; i < lessons.size(); i++) {

lesson = lessons.get(i).getAsJsonObject();

if (lesson.getAsJsonObject().get("week").getAsString().equals(week)) {

sdf = new SimpleDateFormat("HH:mm");

try {

tmp.setTime(sdf.parse(lesson.get("start\_time").getAsString()));

} catch (ParseException e) {

throw new RuntimeException(e);

}

lessonHour = (byte) tmp.get(Calendar.HOUR\_OF\_DAY);

lessonMinute = (byte) tmp.get(Calendar.HOUR\_OF\_DAY);

if (lessonHour > currentHour || (lessonHour == currentHour && lessonMinute > currentMinute)) {

return getLessonString(lesson);

}

}

}

return "";

}

public static String getDayLessons(JsonObject day, String weekNumber) {

StringBuilder schedule = new StringBuilder();

JsonArray lessons = day.get("lessons").getAsJsonArray();

JsonObject lesson;

String dayName = day.get("name").toString();

schedule.append(dayName.replace("\"", ""))

.append(":\n");

byte lessonNumber = 1;

for (int i = 0; i < lessons.size(); i++) {

lesson = lessons.get(i).getAsJsonObject();

if (lesson.getAsJsonObject().get("week").getAsString().equals(weekNumber)) {

schedule.append(lessonNumber).append(". ")

.append(getLessonString(lesson))

.append("\n");

lessonNumber++;

}

}

return schedule.toString();

}

public static String getLessonString(JsonObject lesson) {

StringBuilder schedule = new StringBuilder();

String name = lesson.get("name").getAsString();

String subjectType = lesson.get("subjectType").getAsString();

String teacher = lesson.get("teacher").getAsString();

String start = lesson.get("start\_time").getAsString();

String end = lesson.get("end\_time").getAsString();

schedule.append("Lesson: ")

.append(name.replace("\"", ""))

.append(" (")

.append(subjectType.replace("\"", ""))

.append(")")

.append("\n Teacher: ")

.append(teacher.replace("\"", ""))

.append("\n Time: ")

.append(start.replace("\"", ""))

.append(" - ")

.append(end.replace("\"", ""));

return schedule.toString();

}

}

Bot.java:

package org.example;

import com.google.gson.\*;

import com.pengrad.telegrambot.TelegramBot;

import com.pengrad.telegrambot.UpdatesListener;

import com.pengrad.telegrambot.model.Update;

import com.pengrad.telegrambot.request.SendMessage;

import com.pengrad.telegrambot.response.SendResponse;

import java.util.Calendar;

import java.util.HashMap;

import java.util.List;

import java.util.TimeZone;

public class Bot {

TelegramBot bot;

String TIMEZONE = "Europe/Moscow";

String INVALID\_COMMAND\_MESSAGE = "Invalid command. Try again";

String INVALID\_GROUP\_NUMBER\_MESSAGE = "Group number must consist of 4 digits";

String INVALID\_WEEK\_NUMBER\_MESSAGE = "Week number must be either 1 for odd or 2 for even";

String INVALID\_DAY\_NAME\_MESSAGE = "Invalid day name. \nVariants: Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday";

List<String> LIST\_OF\_COMMANDS = List.of(new String[] {

"next",

"tomorrow",

"week",

"day",

"/help"

});

HashMap<String, String> COMMAND\_USAGES = new HashMap<>();

List<String> daysOfWeek = List.of(new String[] {

"monday",

"tuesday",

"wednesday",

"thursday",

"friday",

"saturday",

"sunday"

});

public Bot() {

String token = System.getenv("LetiSkedHeadBotToken");

if (token == null) {

throw new RuntimeException("Token is not found in environment variable 'LetiSkedHeadBotToken'");

}

bot = new TelegramBot(token);

COMMAND\_USAGES.put("next", "next for {group}");

COMMAND\_USAGES.put("tomorrow", "tomorrow for {group}");

COMMAND\_USAGES.put("week", "week {week} for {group}");

COMMAND\_USAGES.put("day", "day {day of week} week {week} for {group}");

COMMAND\_USAGES.put("/help", "/help \nOR\n/help {command name}");

}

public void start() {

bot.setUpdatesListener(updates -> {

for (Update update : updates) {

processUpdate(update);

}

return UpdatesListener.CONFIRMED\_UPDATES\_ALL;

});

}

private void processUpdate(Update update) {

if (update.message() == null) return;

String text = update.message().text().trim();

Calendar datetime = new Calendar.Builder()

.setInstant(update.message().date() \* 1000L)

.setTimeZone(TimeZone.getTimeZone(TIMEZONE))

.build();

String response\_text = INVALID\_COMMAND\_MESSAGE;

if (text.matches("next for [0-9]+")) {

response\_text = nextLessonCommand(text, datetime);

} else if (text.matches("day [a-zA-Z]+ week [0-9]+ for [0-9]+")) {

response\_text = dayCommand(text);

} else if (text.matches("tomorrow for [0-9]+")) {

response\_text = tomorrowCommand(text, datetime);

} else if (text.matches("week [0-9]+ for [0-9]+")) {

response\_text = weekCommand(text);

} else if (text.matches("/help( [a-zA-Z]+)?")) {

response\_text = getHelp(text);

}

System.out.println(response\_text);

long chatId = update.message().chat().id();

SendResponse response = bot.execute(new SendMessage(chatId, response\_text));

if (!response.isOk()) {

System.out.println("Telegram api connection error: " + response.errorCode());

}

}

private String getHelp(String text) {

String[] words = text.split("\\s+");

if (words.length == 2) {

return getHelpForCommand(words[1]);

}

StringBuilder helpText = new StringBuilder();

helpText.append("AVAILABLE COMMANDS:\n");

for (String command : LIST\_OF\_COMMANDS) {

helpText.append(getHelpForCommand(command))

.append("\n");

}

return helpText.toString();

}

private String getHelpForCommand(String command) {

if (!LIST\_OF\_COMMANDS.contains(command)) return INVALID\_COMMAND\_MESSAGE;

return "Command: " + command + "\nUsage: " + COMMAND\_USAGES.get(command) + "\n";

}

private String nextLessonCommand(String text, Calendar datetime) {

String[] words = text.split("\\s+");

String groupNumber = words[2];

if (groupNumberIsInvalid(groupNumber)) return INVALID\_GROUP\_NUMBER\_MESSAGE;

String week = String.valueOf(datetime.get(Calendar.WEEK\_OF\_YEAR) % 2);

String today = String.valueOf((datetime.get(Calendar.DAY\_OF\_WEEK) - 2));

String scheduleJson = LetiSchedule.getWeekScheduleJson(groupNumber);

if (scheduleJson.equals("{}")) return "Cannot find group " + groupNumber;

JsonObject days = LetiSchedule.getDays(scheduleJson, groupNumber);

byte currentHour = (byte) datetime.get(Calendar.HOUR\_OF\_DAY);

byte currentMinute = (byte) datetime.get(Calendar.MINUTE);

String nextLesson = "";

String result = "";

int daysToSearch = 14;

for (int i = 0; i < daysToSearch && nextLesson.isEmpty(); i++) {

JsonObject day = days.get(today).getAsJsonObject();

JsonArray lessons = day.get("lessons").getAsJsonArray();

nextLesson = LetiSchedule.getNextLessonForToday(lessons, week, currentHour, currentMinute);

result = nextLesson + " (" + day.get("name").getAsString() + ")";

today = String.valueOf((Integer.parseInt(today) + 1));

if (today.equals("7")) {

today = "0";

week = week.equals("1") ? "2" : "1";

}

currentHour = 0;

currentMinute = 0;

}

System.out.println(result);

return result;

}

private String tomorrowCommand(String text, Calendar datetime) {

String[] words = text.split("\\s+");

String groupNumber = words[2];

if (groupNumberIsInvalid(groupNumber)) return INVALID\_GROUP\_NUMBER\_MESSAGE;

String week = String.valueOf(datetime.get(Calendar.WEEK\_OF\_YEAR) % 2);

String tomorrow = String.valueOf((datetime.get(Calendar.DAY\_OF\_WEEK) - 1));

if (tomorrow.equals("0")) {

week = week.equals("1") ? "2" : "1";

}

String scheduleJson = LetiSchedule.getWeekScheduleJson(groupNumber);

if (scheduleJson.equals("{}")) return "Cannot find group " + groupNumber;

JsonObject days = LetiSchedule.getDays(scheduleJson, groupNumber);

JsonObject day = days.get(tomorrow).getAsJsonObject();

return LetiSchedule.getDayLessons(day, week);

}

private String dayCommand(String text) {

String[] words = text.split("\\s+");

String dayName = words[1].toLowerCase();

if (dayNameIsInvalid(dayName)) return INVALID\_DAY\_NAME\_MESSAGE;

String weekNumber = words[3];

if (weekNumberIsInvalid(weekNumber)) return INVALID\_WEEK\_NUMBER\_MESSAGE;

String groupNumber = words[5];

if (groupNumberIsInvalid(groupNumber)) return INVALID\_GROUP\_NUMBER\_MESSAGE;

String scheduleJson = LetiSchedule.getWeekScheduleJson(groupNumber);

if (scheduleJson.equals("{}")) return "Cannot find group " + groupNumber;

JsonObject days = LetiSchedule.getDays(scheduleJson, groupNumber);

JsonObject day = days.get(Integer.toString(daysOfWeek.indexOf(dayName))).getAsJsonObject();

return LetiSchedule.getDayLessons(day, weekNumber);

}

private String weekCommand(String text) {

String[] words = text.split("\\s+");

String weekNumber = words[1];

if (weekNumberIsInvalid(weekNumber)) return INVALID\_WEEK\_NUMBER\_MESSAGE;

String groupNumber = words[3];

if (groupNumberIsInvalid(groupNumber)) return INVALID\_GROUP\_NUMBER\_MESSAGE;

String scheduleJson = LetiSchedule.getWeekScheduleJson(groupNumber);

if (scheduleJson.equals("{}")) return "Cannot find group " + groupNumber;

return LetiSchedule.getWeekSchedule(scheduleJson, groupNumber, weekNumber);

}

private boolean groupNumberIsInvalid(String groupNumber) {

return groupNumber.length() != 4;

}

private boolean weekNumberIsInvalid(String weekNumber) {

return !weekNumber.matches("[1-2]");

}

private boolean dayNameIsInvalid(String dayName) {

return !daysOfWeek.contains(dayName);

}

}

DB.java

package org.example;

import com.pengrad.telegrambot.model.User;

import java.sql.\*;

public class DB {

String jdbcUrl = "jdbc:postgresql://localhost:5432/LetiSkedHead";

Connection connection;

public DB() {

String password = System.getenv("LetiSkedHeadDBPassword");

if (password == null) {

throw new RuntimeException("Password is not found in environment variable 'LetiSkedHeadDBPassword'");

}

try {

connection = DriverManager.getConnection(

jdbcUrl,

"postgres",

password

);

} catch (SQLException e) {

throw new RuntimeException(e);

}

}

public void insertUserIfNotExist(User user, String group) {

int groupNumber = Integer.parseInt(group);

String statement = String.format(

"insert into users (tg\_id, username, firstname, lastname, group\_number) values (%d, '%s', '%s', '%s', %d) " +

"on conflict (tg\_id) do update " +

"set username = excluded.username, " +

"firstname = excluded.firstname, " +

"lastname = excluded.lastname," +

"group\_number = excluded.group\_number;",

user.id(), user.username(), user.firstName(), user.lastName(), groupNumber

);

try {

PreparedStatement query = connection.prepareStatement(statement);

query.execute();

} catch (SQLException e) {

System.out.println("Database error: " + e.getMessage());

}

}

public String getGroupForUser(User user) {

String statement = String.format("select group\_number from users where tg\_id=%d", user.id());

String groupNumber = "";

try {

PreparedStatement query = connection.prepareStatement(statement);

ResultSet result = query.executeQuery();

result.next();

groupNumber = result.getString("group\_number");

} catch (SQLException e) {

System.out.println("Database error: " + e.getMessage());

}

return groupNumber;

}

}

# 4.5. Выводы

В данной работе был дополнен телеграм-бот из работы три. Была добавлена база данных, содержащая пользователей, и реализованы новые версии для команд (обеспечен ввод без указания группы для пользователей, находящихся в базе данных): расписание на неделю, расписание на завтра, следующая пара и расписание на конкретный день.

# 5. Сборка и публикация результатов

Разработанный программный код собирался с помощью системы автоматизированной сборки Intellij IDEA. Результаты были выложены на Github:

<https://github.com/Gregory-hub/leti_2/tree/main/Kotlin(Java)/lab1>

<https://github.com/Tulenchi/kotlin>