МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра САПР

КУРСОВАЯ РАБОТА

по дисциплине «Объектно-ориентированное программирование»

Тема: Бот с расписанием ЛЭТИ

Студент гр. 4351 Чурзин Г.В.

Преподаватель Кулагин М.В.

Санкт-Петербург

2025

ЗАДАНИЕ
НА КУРСОВУЮ РАБОТУ

Студент Чурзин Г.В.

Группа 4351

Тема работы: Бот с расписанием ЛЭТИ

Исходные данные:
Java 17, Oracle OpenJDK 23.0.1, IntelliJ IDEA

Содержание пояснительной записки:
«Содержание», «Введение», «Диаграмма вариантов использования», «Диаграмма классов объектной модели предметной области», «Спецификация классов», «Код классов объектной модели», «Описание интерфейса пользователя программы», «Заключение», «Список использованных источников», «Приложение»

Предполагаемый объем пояснительной записки:
Не менее 20 страниц.

Дата выдачи задания: 30.09.2025

Дата сдачи реферата: 30.12.2025

Дата защиты реферата: 30.12.2025

Студент Чурзин Г.В.

Преподаватель Кулагин М.В.

АННОТАЦИЯ

В процессе выполнения курсовой работы был создан уникальный Телеграм бот путём изменения класса TelegramLongPollingBot под задачи вывода расписания группы. Бот имеет 5 стандартных команд: вывод ближайшей пары, вывод расписания на день, вывод расписания на завтра, вывод расписания на неделю, вывод чётности текущей недели — и способен обрабатывать случаи некорректных запросов. Бот разработан на Java.

SUMMARY

During the coursework, a unique Telegram bot was created by modifying the TelegramLongPollingBot class to display the group schedule. The bot has five standard commands: display the nearest pair, display the daily schedule, display tomorrow's schedule, display the weekly schedule, and display the current week's parity. It is also capable of handling invalid requests. The bot is developed in Java.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Знание актуального расписания занятий в университете важно для успешной учёбы. Целью работы является разработка Telegram бота, выдающего по различным командам пользователя расписание пар, взятого с API ЛЭТИ. Путём изучения структуры файлов расписания [2] и функций библиотеки Telegram [1] были реализованы 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.

Также была реализована корректная обработка ошибок в запросе.

Диаграмма вариантов использования.

Диаграмма вариантов использования представлена на рисунке 1.

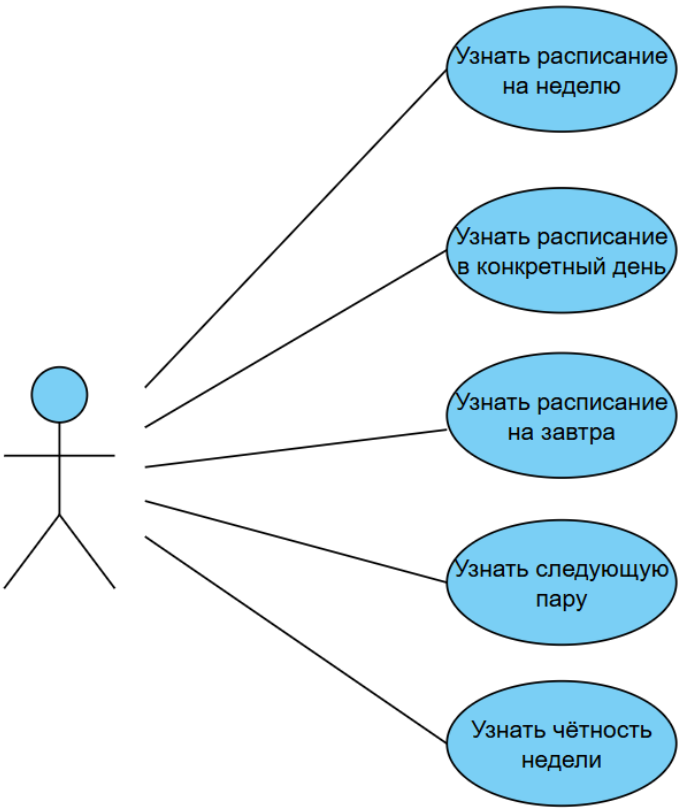

Рисунок 1

Диаграмма классов объектной модели предметной области.

UML-диаграмма классов Telegram бота представлена на рисунке 2.

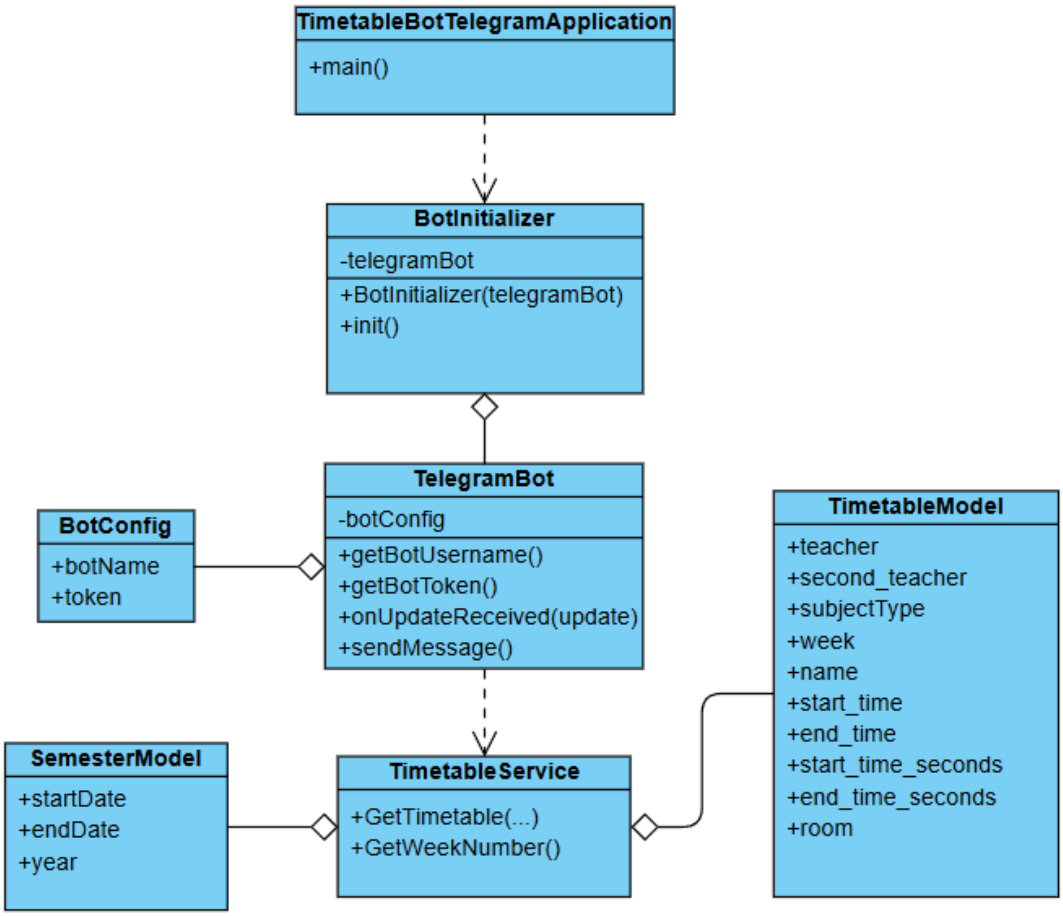

Рисунок 2

Спецификация классов.

Класс TimetableBotTelegramApplication:

Методы:

1. public static void main(String[] args) запускает приложение

Класс BotConfig:

Поля:

1. String botName — логин бота
2. String token — уникальный токен для доступа к боту

Класс BotInitializer:

Поля:

1. TelegramBot telegramBot — экземпляр ТГ бота

Методы:

1. public BotInitializer(TelegramBot telegramBot) инициализирует бота
2. public void init() регистрирует бота

Класс SemesterModel:

Поля:

1. String startDate — дата начала текущего семестра
2. String endDate — дата завершения текущего семестра
3. Integer year — год начала семестра

Класс TimetableModel:

Поля:

1. String teacher — имя преподавателя
2. String second_teacher — имя второго преподавателя
3. String subjectType — тип пары
4. Integer week — чётность недели
5. String name — название предмета
6. String start_time — начало пары
7. String end_time — конец пары
8. Integer start_time_seconds — начало пары в секундах
9. Integer end_time_seconds — конец пары в секундах
10. String room — аудитория

Класс TelegramBot:

Поля:

1. BotConfig botConfig — конфигурация (имя и токен) бота

Методы:

1. public String getBotUsername() возвращает логин
2. public String getBotToken() возвращает токен
3. public void onUpdateReceived(Update update) обрабатывает запрос
4. public void sendMessage(Long chatId, String textToSend) отправляет сообщение

Класс TimetableService:
Методы:

1. public static String GetTimetable(int week, int group, int daySec, boolean near, boolean oneday, int weekday) — находит нужное расписание
2. public static int getWeekNumber() — определяет чётность недели

Код классов объектной модели.

Код основного класса TimetableBotTelegramApplication.java:

```
package com.example.timetable_bot_telegram;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class TimetableBotTelegramApplication {

	public static void main(String[] args) {
		SpringApplication.run(TimetableBotTelegramApplication.class, args);
	}

}
```

Листинг всего кода представлен в приложении А.

Описание интерфейса пользователя программы

На рисунках 3 и 4 представлены примеры использования бота. Для получения ответа необходимо составить запрос согласно шаблону, вызываемому командой /start.

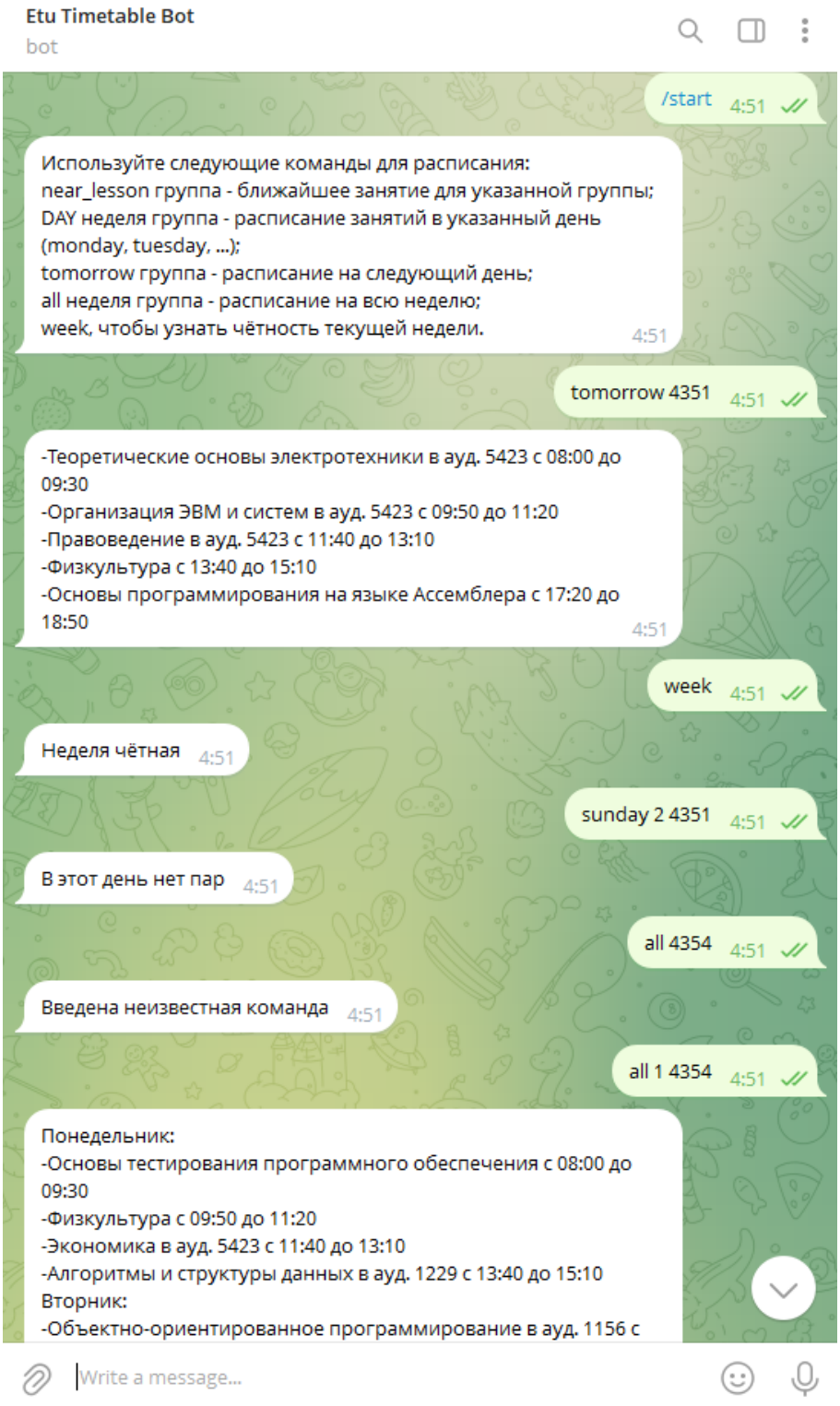

Рисунок 3

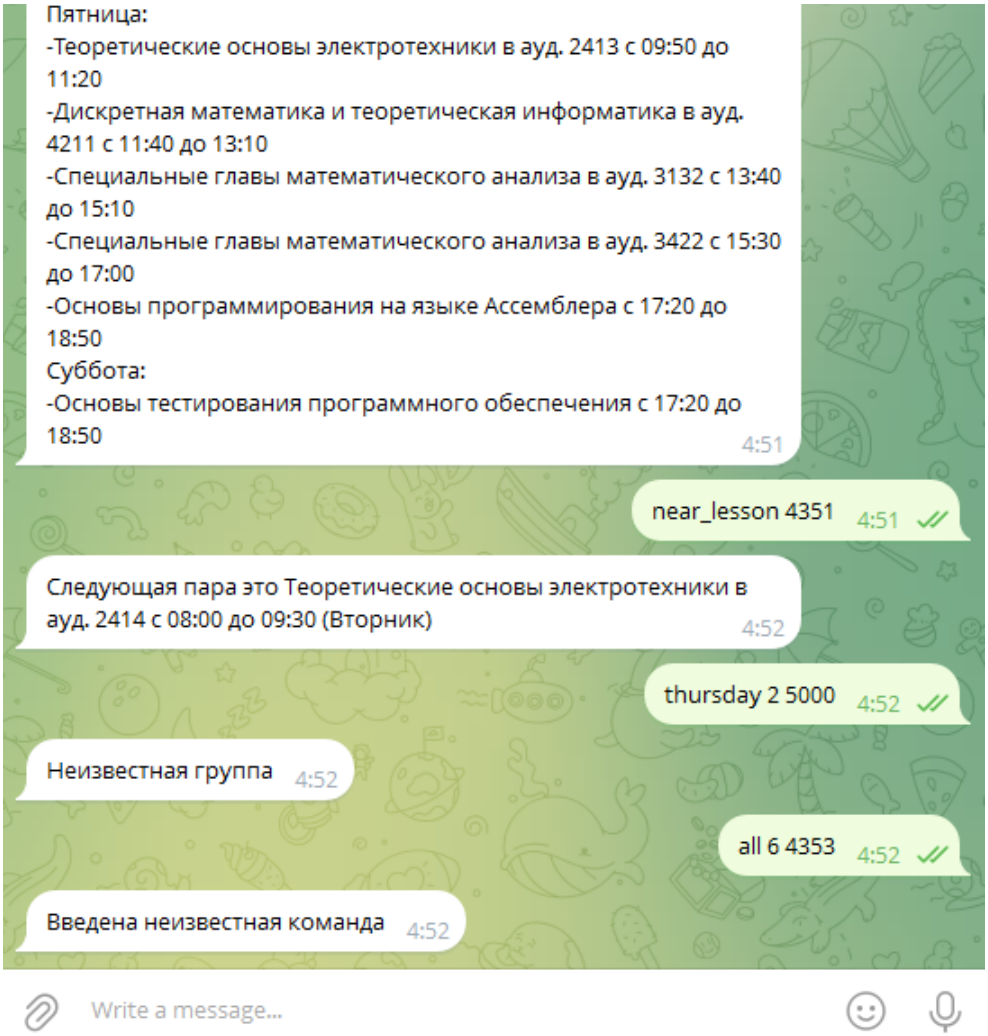
Пятница:
-Теоретические основы электротехники в ауд. 2413 с 09:50 до 11:20
-Дискретная математика и теоретическая информатика в ауд. 4211 с 11:40 до 13:10
-Специальные главы математического анализа в ауд. 3132 с 13:40 до 15:10
-Специальные главы математического анализа в ауд. 3422 с 15:30 до 17:00
-Основы программирования на языке Ассемблера с 17:20 до 18:50
Суббота:
-Основы тестирования программного обеспечения с 17:20 до 18:50
4:51
near_lesson 4351 4:51
Следующая пара это Теоретические основы электротехники в ауд. 2414 с 08:00 до 09:30 (Вторник)
4:52
thursday 2 5000 4:52
Неизвестная группа 4:52
all 6 4353 4:52
Введена неизвестная команда 4:52
Write a message...

Рисунок 4

ЗАКЛЮЧЕНИЕ

В результате выполнения работы были реализованы функции вывода расписания в различных формах для всех обучающихся групп. Реализована обработка некорректного ввода.

Ссылка на репозиторий git https://github.com/ChursStudent/OOPprograms

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Написание простого Telegram бота // habr.com URL: https://habr.com/ru/articles/715384/ (дата обращения: 29.12.2025).

2. API расписания ЛЭТИ // digital.etu.ru URL: https://digital.etu.ru/api/docs-public/ (дата обращения: 30.12.2025).

ПРИЛОЖЕНИЕ А

КОД ПРОГРАММЫ

«TimetableBotTelegramApplication.java»

```
package com.example.timetable_bot_telegram;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class TimetableBotTelegramApplication {

	public static void main(String[] args) {
		SpringApplication.run(TimetableBotTelegramApplication.class, args);
	}

}
```

«BotInitializer.java»

```
package com.example.timetable_bot_telegram.config;

import com.example.timetable_bot_telegram.service.TelegramBot;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.event.ContextRefreshedEvent;
import org.springframework.context.event.EventListener;
import org.springframework.stereotype.Component;
import org.telegram.telegrambots.meta.TelegramBotsApi;
import org.telegram.telegrambots.meta.exceptions.TelegramApiException;
import org.telegram.telegrambots.updatesreceivers.DefaultBotSession;

@Component
public class BotInitializer {
   private final TelegramBot telegramBot;
   @Autowired
   public BotInitializer(TelegramBot telegramBot) {
      this.telegramBot = telegramBot;
   }

   @EventListener({ContextRefreshedEvent.class})
   public void init()throws TelegramApiException {
      TelegramBotsApi telegramBotsApi = new
TelegramBotsApi(DefaultBotSession.class);
      try {
         telegramBotsApi.registerBot(telegramBot);
      } catch (TelegramApiException e){
```

```
            System.out.println("Бот не регистрируется");
        }
    }
}
```

«BotConfig.java»

```
package com.example.timetable_bot_telegram.config;

import lombok.Data;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.PropertySource;

@Configuration
@Data
@PropertySource("application.properties")
public class BotConfig {
    @Value("${bot.name}")
    String botName;
    @Value("${bot.token}")
    String token;
}
```

«SemesterModel.java»

```
package com.example.timetable_bot_telegram.model;

import lombok.Data;

@Data
public class SemesterModel {
    public String startDate;
    public String endDate;
    public Integer year;
}
```

«TimetableModel.java»

```
package com.example.timetable_bot_telegram.model;

import lombok.Data;

@Data
public class TimetableModel {
    public String teacher;
    public String second_teacher;
    public String subjectType;
    public Integer week;
    public String name = "no";
```

```
    public String start_time;
    public String end_time;
    public Integer start_time_seconds;
    public Integer end_time_seconds;
    public String room;
}
```

«TimetableService.java»

```
package com.example.timetable_bot_telegram.service;

import com.example.timetable_bot_telegram.model.SemesterModel;
import com.example.timetable_bot_telegram.model.TimetableModel;
import com.google.gson.Gson;
import com.google.gson.reflect.TypeToken;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.lang.reflect.Type;
import java.net.HttpURLConnection;
import java.net.URL;
import java.text.ParseException;
import java.time.LocalDate;
import java.time.temporal.ChronoUnit;
import java.util.List;

public class TimetableService {
    public static String GetTimetable(int week, int group, int daySec, boolean near,
boolean oneday, int weekday) throws IOException, ParseException {
        String urlString =
"https://digital.etu.ru/api/mobile/schedule?weekDay=&subjectType=&groupNumber
="+group+"&joinWeeks=true";
        URL url = new URL(urlString);
        HttpURLConnection conn = (HttpURLConnection) url.openConnection();
        conn.setRequestMethod("GET");
        conn.setRequestProperty("Accept", "application/json");
        BufferedReader br = new BufferedReader(new
InputStreamReader((conn.getInputStream())));
        StringBuilder jsonString = new StringBuilder();
        String output;
        while ((output = br.readLine()) != null) {
            jsonString.append(output);
        }
        String[] russianDays = {"Понедельник", "Вторник", "Среда", "Четверг",
"Пятница", "Суббота", "Воскресенье"};
        StringBuilder result;
```

```
        StringBuilder currentLessons = new StringBuilder();
        currentLessons.append(jsonString);
        String delimiter = "\"lessons\":";
        int index;
        String botMessage = "";
        int day = 0;
        int dayOfFirst = 0;
        int bracketCount = 0;
        boolean lessonFound = false;
        boolean dayFound = false;
        boolean firstInDay;
        boolean firstInWeek = true;
        Gson gson = new Gson();
        Type listType = new TypeToken<List<TimetableModel>>() {}.getType();
        List<TimetableModel> lessonDay;
        TimetableModel lessonFirstDay = new TimetableModel();
        while (day < 7 && !lessonFound && !dayFound) {
            firstInDay = true;
            result = new StringBuilder();
            index = currentLessons.toString().indexOf(delimiter);
            if (index == -1 && day == 0) {
                botMessage += "Неизвестная группа";
                break;
            }
            currentLessons = new StringBuilder(currentLessons.substring(index +
delimiter.length()));
            for (int i = 0; i < currentLessons.length(); i++) {
                result.append(currentLessons.charAt(i));
                if (currentLessons.charAt(i) == '[') {
                    bracketCount++;
                }
                if (currentLessons.charAt(i) == ']') {
                    bracketCount--;
                }
                if (bracketCount == 0) {
                    break;
                }
            }
            lessonDay = gson.fromJson(result.toString(), listType);
            for (TimetableModel lessonSingle : lessonDay) {
                if (firstInWeek && lessonSingle.week != week) {
                    lessonFirstDay.name = lessonSingle.name;
                    lessonFirstDay.room = lessonSingle.room;
                    lessonFirstDay.start_time = lessonSingle.start_time;
```

```
                lessonFirstDay.end_time = lessonSingle.end_time;
                dayOfFirst = day;
                firstInWeek = false;
            }
            if (lessonSingle.week == 3 || lessonSingle.week == week) {
                if (near) {
                    if ((86400*weekday + daySec - 86400*day -
lessonSingle.end_time_seconds) < 0) {
                        botMessage = "Следующая пара это " + lessonSingle.name
                                + (lessonSingle.room.isEmpty()?"":" в ауд. " +
lessonSingle.room)
                                + " с " + lessonSingle.start_time + " до " +
lessonSingle.end_time + " (" + russianDays[day] + ")\n";
                        lessonFound = true;
                        break;
                    }
                }
                else if (oneday && day == weekday) {
                    botMessage += "-" + lessonSingle.name
                            + (lessonSingle.room.isEmpty()?"":" в ауд. " +
lessonSingle.room)
                            + " с " + lessonSingle.start_time + " до " +
lessonSingle.end_time + "\n";
                    dayFound = true;
                }
                else {
                    if (!oneday) {
                        if (firstInDay) {
                            botMessage += russianDays[day] + ":\n-" + lessonSingle.name
                                    + (lessonSingle.room.isEmpty()?"":" в ауд. " +
lessonSingle.room)
                                    + " с " + lessonSingle.start_time + " до " +
lessonSingle.end_time + "\n";
                            firstInDay = false;
                        } else botMessage += "-" + lessonSingle.name
                                + (lessonSingle.room.isEmpty()?"":" в ауд. " +
lessonSingle.room)
                                + " с " + lessonSingle.start_time + " до " +
lessonSingle.end_time + "\n";
                    }
                }
            }
            day++;
```

```
        }
        if (botMessage.isEmpty() && near) {
            botMessage = "Следующая пара это " + lessonFirstDay.name
                    + (lessonFirstDay.room.isEmpty()?"":" в ауд. " + lessonFirstDay.room)
                    + " с " + lessonFirstDay.start_time + " до " + lessonFirstDay.end_time +
" (" + russianDays[dayOfFirst] + ")\n";
        }
        if (botMessage.isEmpty()) {
            botMessage = "В этот день нет пар";
        }
        return botMessage;
    }

    public static int getWeekNumber() throws IOException {
        String urlString = "https://digital.etu.ru/api/mobile/semester";
        URL url = new URL(urlString);
        HttpURLConnection conn = (HttpURLConnection) url.openConnection();
        conn.setRequestMethod("GET");
        conn.setRequestProperty("Accept", "application/json");
        BufferedReader br = new BufferedReader(new
InputStreamReader((conn.getInputStream())));
        StringBuilder jsonString = new StringBuilder();
        String output;
        while ((output = br.readLine()) != null) {
            jsonString.append(output);
        }
        Gson gson = new Gson();
        SemesterModel Sem = gson.fromJson(jsonString.toString(),
SemesterModel.class);
        int year, month, day;
        year = Sem.year;
        month = 10*(Sem.startDate.charAt(5) - '0') + Sem.startDate.charAt(6) - '0';
        day = 10*(Sem.startDate.charAt(8) - '0') + Sem.startDate.charAt(9) - '0';
        LocalDate today = LocalDate.now();
        LocalDate dateStart = LocalDate.of(year, month, day);
        long daysBetween = ChronoUnit.DAYS.between(dateStart, today);
        return (int)(daysBetween / 7) % 2 + 1;
    }
}
```

«TelegramBot.java»

```
package com.example.timetable_bot_telegram.service;

import com.example.timetable_bot_telegram.config.BotConfig;
import com.google.gson.JsonSyntaxException;
```

```
import lombok.AllArgsConstructor;
import org.springframework.stereotype.Component;
import org.telegram.telegrambots.bots.TelegramLongPollingBot;
import org.telegram.telegrambots.meta.api.methods.send.SendMessage;
import org.telegram.telegrambots.meta.api.objects.Update;
import org.telegram.telegrambots.meta.exceptions.TelegramApiException;
import java.io.IOException;
import java.text.ParseException;
import java.time.LocalTime;
import java.util.Calendar;
import java.util.Scanner;
import static
com.example.timetable_bot_telegram.service.TimetableService.getWeekNumber;

@Component
@AllArgsConstructor
public class TelegramBot extends TelegramLongPollingBot {
    private final BotConfig botConfig;

    @Override
    public String getBotUsername() {
        return botConfig.getBotName();
    }

    @Override
    public String getBotToken() {
        return botConfig.getToken();
    }

    @Override
    public void onUpdateReceived(Update update) {
        String timetable = "";
        if (update.hasMessage() && update.getMessage().hasText()) {
            String messageText = update.getMessage().getText();
            long chatId = update.getMessage().getChatId();
            System.out.println("Получено сообщение: " + messageText + " от " +
chatId);
            String msgWord;
            Calendar calendar = Calendar.getInstance();
            int msgGroup = 0, msgWeek = 0;
            int msgDaySec = LocalTime.now().toSecondOfDay();
            int msgWeekday = calendar.get(Calendar.DAY_OF_WEEK) - 2;
            if (msgWeekday == -1) {
                msgWeekday = 6;
```

```
        }
        boolean msgNear = false, msgOneday = false;
        Scanner scanner = new Scanner(messageText);
        msgWord = scanner.next();
        boolean wrongInput = false;
        switch (msgWord) {
            case "/start":
                sendMessage(chatId, "Используйте следующие команды для
расписания:\n" +
                        "near_lesson группа - ближайшее занятие для указанной
группы;\n" +
                        "DAY неделя группа - расписание занятий в указанный день
(monday, tuesday, ...);\n" +
                        "tomorrow группа - расписание на следующий день;\n" +
                        "all неделя группа - расписание на всю неделю;\n" +
                        "week, чтобы узнать чётность текущей недели.");
                break;
            default: {
                switch (msgWord) {
                    case "near_lesson":
                        if (scanner.hasNextInt()) {
                            msgGroup = scanner.nextInt();
                        } else wrongInput = true;
                        try {
                            msgWeek = getWeekNumber();
                        } catch (IOException e) {
                            throw new RuntimeException(e);
                        }
                        msgNear = true;
                        msgOneday = false;
                        msgWeekday = msgWeekday;
                        break;
                    case "monday":
                        if (scanner.hasNextInt()) {
                            msgWeek = scanner.nextInt();
                        } else wrongInput = true;
                        if (scanner.hasNextInt()) {
                            msgGroup = scanner.nextInt();
                        } else wrongInput = true;
                        msgNear = false;
                        msgOneday = true;
                        msgWeekday = 0;
                        break;
                    case "tuesday":
```

```
    if (scanner.hasNextInt()) {
        msgWeek = scanner.nextInt();
    } else wrongInput = true;
    if (scanner.hasNextInt()) {
        msgGroup = scanner.nextInt();
    } else wrongInput = true;
    msgNear = false;
    msgOneday = true;
    msgWeekday = 1;
    break;
case "wednesday":
    if (scanner.hasNextInt()) {
        msgWeek = scanner.nextInt();
    } else wrongInput = true;
    if (scanner.hasNextInt()) {
        msgGroup = scanner.nextInt();
    } else wrongInput = true;
    msgNear = false;
    msgOneday = true;
    msgWeekday = 2;
    break;
case "thursday":
    if (scanner.hasNextInt()) {
        msgWeek = scanner.nextInt();
    } else wrongInput = true;
    if (scanner.hasNextInt()) {
        msgGroup = scanner.nextInt();
    } else wrongInput = true;
    msgNear = false;
    msgOneday = true;
    msgWeekday = 3;
    break;
case "friday":
    if (scanner.hasNextInt()) {
        msgWeek = scanner.nextInt();
    } else wrongInput = true;
    if (scanner.hasNextInt()) {
        msgGroup = scanner.nextInt();
    } else wrongInput = true;
    msgNear = false;
    msgOneday = true;
    msgWeekday = 4;
    break;
case "saturday":
```

```
            if (scanner.hasNextInt()) {
                msgWeek = scanner.nextInt();
            } else wrongInput = true;
            if (scanner.hasNextInt()) {
                msgGroup = scanner.nextInt();
            } else wrongInput = true;
            msgNear = false;
            msgOneday = true;
            msgWeekday = 5;
            break;
        case "sunday":
            if (scanner.hasNextInt()) {
                msgWeek = scanner.nextInt();
            } else wrongInput = true;
            if (scanner.hasNextInt()) {
                msgGroup = scanner.nextInt();
            } else wrongInput = true;
            msgNear = false;
            msgOneday = true;
            msgWeekday = 6;
            break;
        case "tomorrow":
            if (scanner.hasNextInt()) {
                msgGroup = scanner.nextInt();
            } else wrongInput = true;
            try {
                msgWeek = getWeekNumber();
            } catch (IOException e) {
                throw new RuntimeException(e);
            }
            msgNear = false;
            msgOneday = true;
            msgWeekday = (msgWeekday + 1) % 7;
            if (msgWeekday == 0) {
                msgWeek = (msgWeek==1) ? 2 : 1;
            }
            break;
        case "all":
            if (scanner.hasNextInt()) {
                msgWeek = scanner.nextInt();
            } else wrongInput = true;
            if (scanner.hasNextInt()) {
                msgGroup = scanner.nextInt();
            } else wrongInput = true;
```

```
                    msgNear = false;
                    msgOneday = false;
                    msgWeekday = 0;
                    break;
                case "week":
                    try {
                        if (getWeekNumber() == 1) sendMessage(chatId, "Неделя
нечётная");
                        else sendMessage(chatId, "Неделя чётная");
                        return;
                    } catch (IOException e) {
                        throw new RuntimeException(e);
                    }
                default:
                    wrongInput = true;
                    break;
            }
            if (!(msgWeek == 1 || msgWeek == 2)) {
                wrongInput = true;
            }
            if (!wrongInput) {
                try {
                    timetable = TimetableService.GetTimetable(msgWeek, msgGroup,
msgDaySec, msgNear, msgOneday, msgWeekday);

                } catch (IOException e) {
                    sendMessage(chatId, "Неверные данные");
                } catch (JsonSyntaxException e) {
                    throw new RuntimeException("Unable to parse json");
                } catch (ParseException e) {
                    throw new RuntimeException("Unable to parse data");
                }
                sendMessage(chatId, timetable);
            } else
                sendMessage(chatId, "Введена неизвестная команда");
            break;
        }
    }
  }

  private void sendMessage(Long chatId, String textToSend){
    SendMessage sendMessage = new SendMessage();
    sendMessage.setChatId(String.valueOf(chatId));
```

```
        sendMessage.setText(textToSend);
        try {
            execute(sendMessage);
        } catch (TelegramApiException e) {
            System.out.println("Сообщение не отправилось");
        }
    }
}
```

«application.properties»

```
spring.application.name=timetable-bot-telegram
bot.name=etuTimeBot
bot.token=токен
```