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

КУРСОВАЯ РАБОТА

по дисциплине «Объектно-ориентированное программирование»

Тема: Создание бота в Telegram с расписанием ЛЭТИ

Студентки гр. 4354 Кирсова В.В.
Коробцова Д.А.

Преподаватель Кулагин М.В.

Санкт-Петербург
2025

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

Студентки Кирсова В.В., Коробцова Д.А.

Группа 4354

Тема работы: Создание бота в Telegram с расписанием ЛЭТИ

Исходные данные:

Название и версия языка программирования: Java version "1.8.0_401"

Название ПО для разработки: IntelliJ IDEA

Содержание пояснительной записки:

«Аннотация», «Содержание», «Введение», «Диаграмма вариантов использования», «Диаграмма классов объектной модели предметной области», «Спецификация классов», «Код классов объектной модели», «Описоание интерфейса пользователя», «Заключение», «Список использованных источников», «Приложение».

Предполагаемый объем пояснительной записки:

Не менее 30 страниц.

Дата выдачи задания: 09.10.2025

Дата сдачи реферата: 16.12.2025

Дата защиты реферата: 16.12.2025

Студентки Кирсова В.В.
Коробцова Д.А.

Преподаватель Кулагин М.В.

АННОТАЦИЯ

В рамках курсового проекта разработан Telegram-бот для автоматизированного предоставления актуального расписания занятий студентам СПбГЭТУ «ЛЭТИ». Бот интегрирован с официальным API университета, поддерживает выбор группы, различные режимы отображения расписания (ближайшая пара, завтра, день, неделя) и учитывает чередование чётных/нечётных учебных недель. В работе применены методы объектно-ориентированного программирования, REST API интеграции, асинхронной обработки пользовательских запросов и модульного тестирования. Результатом является полностью функциональное кроссплатформенное приложение, обеспечивающее удобный и быстрый доступ к расписанию через мессенджер Telegram.

SUMMARY

This course project involves the development of a Telegram bot designed to provide students of ETU "LETI" with up-to-date class schedules. The bot is integrated with the university's official API, supports group selection, multiple schedule display modes (next class, tomorrow, daily, weekly), and accounts for even/odd week alternation. The project employs object-oriented programming methods, REST API integration, asynchronous user request processing, and modular testing. The outcome is a fully functional cross-platform application that offers convenient and quick access to schedules via the Telegram messenger.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель работы

Создание Telegram-бота, который позволил бы получить расписание занятий для любой группы университета СПБГТЭУ «ЛЭТИ».

Основные задачи и методы их решения:

1. Доступ к актуальному расписанию университета. Решение: Написание отдельного сервиса, который отправляет запросы на сайт университета, получает расписание в формате JSON и разбирает его в удобную структуру для дальнейшей работы.
2. Хранение и обработка состояния каждого пользователя. Решение: Сохранение для каждого пользователя его группы, выбранного дня и текущего меню в памяти программы.
3. Предоставление удобного интерфейса для выбора действий. Решение: Реализация всплывающей клавиатуры с кнопками: «Ближайшая пара», «Завтра», «Расписание по дням» и другие, чтобы пользователь не вводил команды вручную.
4. Корректное отображение расписания для чётных/нечётных недель. Решение: Алгоритм расчёта от базовой даты (1 сентября), использование кодов недель из данных API.
5. Обеспечение стабильной работы при некорректных данных. Решение: Проверка наличия группы у пользователя, валидация ввода (4 цифры для номера группы), обработка исключений в блоках try-catch, вывод информативных сообщений об ошибках.

1. Диаграмма вариантов использования.

Для определения функциональных возможностей системы и взаимодействия пользователей с Telegram-ботом была разработана диаграмма вариантов использования, которая иллюстрирует основные сценарии работы с приложением.

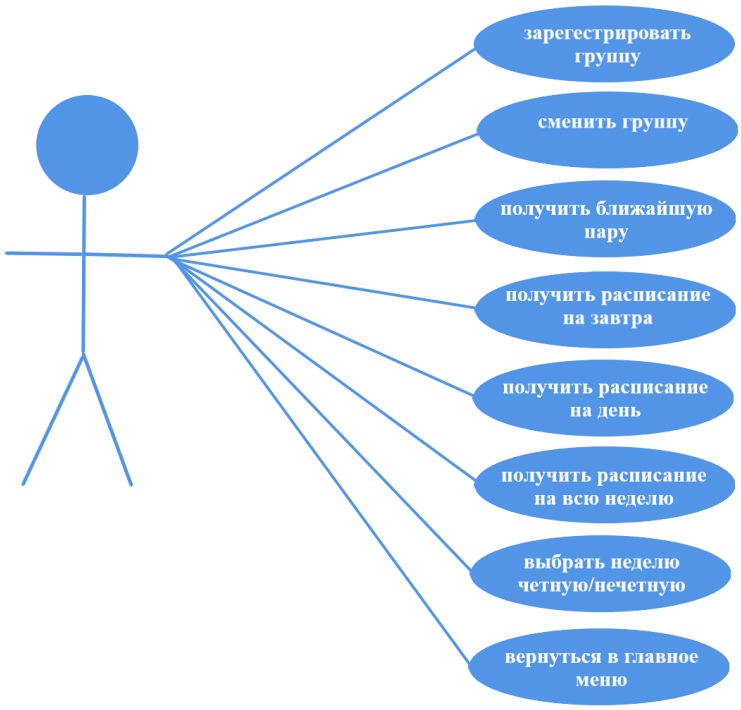


Рис. 1. Диаграмма вариантов использования системы (use cases)

2. Диаграмма классов объектной модели предметной области.

Для наглядного представления структуры системы и взаимосвязей между её компонентами была разработана диаграмма классов, отображающая основные сущности предметной области и их взаимодействие.

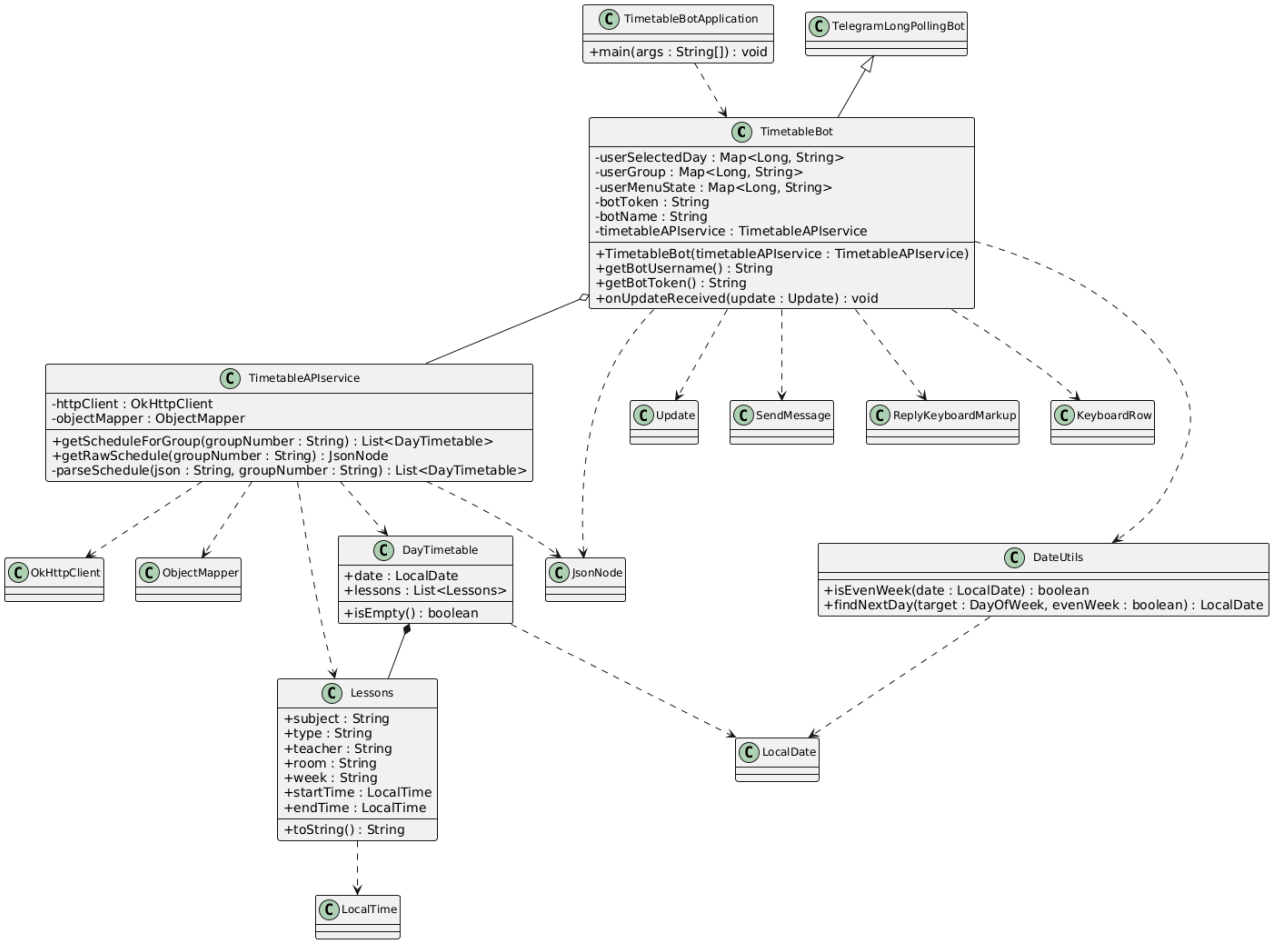


Рис. 2. Диаграмма классов

3. Спецификация классов.

В этом разделе описывается структура классов Telegram-бота для работы с расписанием. Представлены все основные компоненты системы с указанием их полей, методов и функционального назначения.

3.1 Класс TimetableBot

Поля	Описание полей
`Map<Long, String> userSelectedDay`	Хранит выбранный день недели для каждого пользователя
Map<Long, String> userGroup	Хранит номер группы для каждого пользователя
Map<Long, String> userMenuState	Хранит текущее состояние меню для каждого пользователя
String botToken	Токен Telegram-бота
String botName	Имя бота
TimetableAPIservice timetableAPIservice	Сервис для работы с API расписания

Методы	Описание методов
public String getBotUsername()	Возвращает имя бота
ublic String getBotToken()	Возвращает токен бота
public void onUpdateReceived(Update update)	Обрабатывает входящие сообщения от пользователей
private void handleNearLesson(long chatId)	Обрабатывает запрос на ближайшую пару
private void handleTomorrow(long chatId)	Обрабатывает запрос на расписание на завтра

private void handleDayForWeek(long chatId, String weekType)	Обрабатывает запрос на расписание для конкретного дня и типа недели
private void handleFullWeekForType(long chatId, String weekType)	Обрабатывает запрос на расписание на всю неделю
private void showMainMenu(long chatId)	Показывает главное меню
private void showDayMenu(long chatId)	Показывает меню выбора дня недели
private void showWeekSelectionMenu(long chatId, String dayName)	Показывает меню выбора типа недели для дня
private void showWeekSelectionForFullWeek(long chatId)	Показывает меню выбора типа недели для всей недели
private ReplyKeyboardMarkup createMainMenu()	Создает клавиатуру главного меню
private ReplyKeyboardMarkup createDayMenu()	Создает клавиатуру меню дней недели
private ReplyKeyboardMarkup createWeekSelectionMenu(boolean forFullWeek)	Создает клавиатуру выбора типа недели
private String formatLessonWithTime(JsonNode l, int index)	Форматирует информацию о паре для вывода
private String getTeacher(JsonNode l)	Извлекает информацию о преподавателе из JSON
private String getRoom(JsonNode l)	Извлекает информацию об аудитории из JSON
private Integer getDayIndex(String dayName)	Преобразует название дня в индекс (0-5)

private void sendMsg(long chatId, String text)	Отправляет сообщение пользователю
private void appendWeek(StringBuilder sb, JsonNode groupNode, boolean evenWeek)	Добавляет расписание на неделю в StringBuilder
private void appendCombinedWeek(StringBuilder sb, JsonNode groupNode)	Добавляет комбинированное расписание на обе недели

3.2. Класс TimetableAPIservice

Поля	Описание полей
OkHttpClient httpClient	HTTP-клиент для выполнения запросов к API
ObjectMapper objectMapper	JSON-парсер для работы с ответами API

Методы	Описание методов
public List<DayTimetable> getScheduleForGroup(String groupNumber)	Получает расписание для указанной группы
public JsonNode getRawSchedule(String groupNumber)	Получает сырые JSON-данные расписания для группы
private List<DayTimetable> parseSchedule(String json, String groupNumber)	Парсит JSON-ответ и преобразует его в список DayTimetable

3.3. Класс Lessons

Поля	Описание полей
String subject	Название предмета
String type	Тип занятия (лекция, практика и т.д.)

String teacher	Преподаватель
String room	Аудитория
String week	Тип недели
LocalTime startTime	Время начала занятия
LocalTime endTime	Время окончания занятия

Методы	**Описание методов**
public String toString()	Возвращает строковое представление занятия

3.4. Класс DayTimetable

Поля	**Описание полей**
LocalDate date	Дата дня расписания
List<Lessons> lessons	Список занятий в этот день

Методы	**Описание методов**
public boolean isEmpty()	Проверяет, есть ли занятия в этот день

3.5. Класс DateUtils

Поля	**Описание полей**
Нет полей	Вспомогательный класс без состояния

Методы	**Описание методов**
public static boolean isEvenWeek(LocalDate date)	Определяет, является ли неделя четной для указанной даты

public static LocalDate findNextDay(DayOfWeek target, boolean evenWeek)	Находит следующую дату с указанным днем недели и четностью

3.6. Класс TimetableBotApplication

Поля	Описание полей
Нет полей	Главный класс приложения

Методы	Описание методов
public static void main(String[] args)	Точка входа в приложение, запускает Spring Boot контекст и регистрирует бота

4. Код классов объектной модели.

В данном разделе представлены основные классы объектной модели, используемые в системе Telegram-бота.

```
package ru.etu.timetable_bot.bot;

import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.LocalTime;
import java.time.format.DateTimeFormatter;
import java.util.*;
import java.util.concurrent.ConcurrentHashMap;
import org.springframework.beans.factory.annotation.Value;
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 ru.etu.timetable_bot.api.TimetableAPIservice;
import ru.etu.timetable_bot.utils.DateUtils;
import com.fasterxml.jackson.databind.JsonNode;
import
org.telegram.telegrambots.meta.api.objects.replykeyboard.ReplyKeyboardMarkup;
import
org.telegram.telegrambots.meta.api.objects.replykeyboard.buttons.KeyboardRow;

@Component
public class TimetableBot extends TelegramLongPollingBot {
   private final Map<Long, String> userGroup = new ConcurrentHashMap<>();
   private final Map<Long, String> selectedDay = new ConcurrentHashMap<>();
   private final Map<Long, String> menuState = new ConcurrentHashMap<>();

   @Value("${token}") private String botToken;
   @Value("${telegram.bot-name}") private String botName;
   private final TimetableAPIservice timetableAPIservice;

   public TimetableBot(TimetableAPIservice timetableAPIservice) {
      this.timetableAPIservice = timetableAPIservice;
   }

   @Override public String getBotUsername() { return botName; }
   @Override public String getBotToken() { return botToken; }
```

```
    @Override
    public void onUpdateReceived(Update update) {
        if (!update.hasMessage() || !update.getMessage().hasText()) return;

        long chatId = update.getMessage().getChatId();
        String text = update.getMessage().getText().trim();

        try {
            if (text.equals("/start")) handleStart(chatId);
            else if (text.matches("\\d{4}")) handleGroupInput(chatId, text);
            else if (text.equals("Назад")) handleBack(chatId);
            else handleCommand(chatId, text);
        } catch (Exception e) {
            e.printStackTrace();
            sendMsg(chatId, "Ошибка: " + e.getMessage());
        }
    }

    private void handleStart(long chatId) {
        clearUserData(chatId);
        sendMsg(chatId, "Привет! Укажите номер вашей группы (4 цифры,
например: 4354):");
    }

    private void handleGroupInput(long chatId, String group) {
        userGroup.put(chatId, group);
        sendMsg(chatId, "Группа сохранена: " + group);
        showMainMenu(chatId);
    }

    private void handleBack(long chatId) {
        String state = menuState.get(chatId);
        if ("week_selection".equals(state)) showDayMenu(chatId);
        else showMainMenu(chatId);
    }

    private void handleCommand(long chatId, String command) throws Exception {
        switch (command) {
            case "Сменить группу" -> handleChangeGroup(chatId);
            case "Ближайшая пара" -> handleNearLesson(chatId);
            case "Завтра" -> handleTomorrow(chatId);
            case "Вся неделя" -> showWeekSelectionForFullWeek(chatId);
            case "Расписание по дням" -> showDayMenu(chatId);
            case "Нечетная неделя", "Четная неделя", "Обе недели" ->
```

```
            handleWeekSelection(chatId, getWeekType(command));
        default -> handleDaySelection(chatId, command);
    }
}

private void handleChangeGroup(long chatId) {
    clearUserData(chatId);
    sendMsg(chatId, "Введите новый номер группы (4 цифры):");
}

private void handleDaySelection(long chatId, String dayName) throws Exception {
    if (isDayName(dayName)) {
        selectedDay.put(chatId, dayName);
        showWeekSelectionMenu(chatId, dayName);
    } else {
        sendMsg(chatId, "Пожалуйста, используйте кнопки.");
    }
}

private String getWeekType(String command) {
    return switch (command) {
        case "Нечетная неделя" -> "odd";
        case "Четная неделя" -> "even";
        case "Обе недели" -> "both";
        default -> "both";
    };
}

private boolean isDayName(String text) {
    return List.of("Понедельник", "Вторник", "Среда",
                   "Четверг", "Пятница", "Суббота").contains(text);
}

// Основные методы обработки расписания
private void handleNearLesson(long chatId) throws Exception {
    String group = userGroup.get(chatId);
    if (group == null) { sendMsg(chatId, "Сначала укажите группу."); return; }

    JsonNode schedule = timetableAPIservice.getRawSchedule(group).get(group);
    if (schedule == null) { sendMsg(chatId, "Расписание не найдено."); return; }

    LocalDateTime now = LocalDateTime.now();
    for (int offset = 0; offset < 14; offset++) {
        LocalDate date = now.toLocalDate().plusDays(offset);
```

```
            int dayIndex = date.getDayOfWeek().getValue() - 1;
            if (dayIndex >= 6) continue;

            JsonNode dayNode = schedule.path("days").path(String.valueOf(dayIndex));
            if (!dayNode.has("lessons")) continue;

            JsonNode lesson = findNextLessonInDay(dayNode, date);
            if (lesson != null) {
                String formattedDate =
date.format(DateTimeFormatter.ofPattern("dd.MM.yyyy"));
                sendMsg(chatId, "□ Ближайшая пара (" + formattedDate + ")\n\n" +
formatLesson(lesson, 1));
                return;
            }
        }
        sendMsg(chatId, "Ближайшие 2 недели — занятий нет.");
    }

    private JsonNode findNextLessonInDay(JsonNode dayNode, LocalDate date) {
        Map<String, List<JsonNode>> slots = groupLessonsByTime(dayNode);
        boolean isEvenDate = DateUtils.isEvenWeek(date);
        LocalTime now = LocalTime.now();

        for (Map.Entry<String, List<JsonNode>> entry : slots.entrySet()) {
            LocalTime start = LocalTime.parse(entry.getKey().split("-")[0]);
            if (date.equals(LocalDate.now()) && start.isBefore(now)) continue;

            return selectLessonForWeek(entry.getValue(), isEvenDate);
        }
        return null;
    }

    private Map<String, List<JsonNode>> groupLessonsByTime(JsonNode dayNode)
{
        Map<String, List<JsonNode>> slots = new LinkedHashMap<>();
        for (JsonNode lesson : dayNode.get("lessons")) {
            String key = lesson.get("start_time").asText() + "-" +
lesson.get("end_time").asText();
            slots.computeIfAbsent(key, k -> new ArrayList<>()).add(lesson);
        }
        return slots;
    }
```

```
    private JsonNode selectLessonForWeek(List<JsonNode> lessons, boolean
isEvenWeek) {
        if (isEvenWeek) {
            for (JsonNode l : lessons) if (l.get("week").asText().equals("2")) return l;
            for (JsonNode l : lessons) if (l.get("week").asText().matches("[13]")) return l;
        } else {
            for (JsonNode l : lessons) if (l.get("week").asText().matches("[13]")) return l;
            for (JsonNode l : lessons) if (l.get("week").asText().equals("2")) return l;
        }
        return null;
    }

    private void handleTomorrow(long chatId) throws Exception {
        String group = userGroup.get(chatId);
        if (group == null) { sendMsg(chatId, "Сначала укажите группу."); return; }

        LocalDate tomorrow = LocalDate.now().plusDays(1);
        if (tomorrow.getDayOfWeek().getValue() == 7) { // Воскресенье
            sendMsg(chatId, "Завтра воскресенье - занятий нет."); return;
        }

        JsonNode schedule = timetableAPIservice.getRawSchedule(group).get(group);
        if (schedule == null) { sendMsg(chatId, "Расписание не найдено."); return; }

        int dayIndex = tomorrow.getDayOfWeek().getValue() - 1;
        JsonNode dayNode = schedule.path("days").path(String.valueOf(dayIndex));

        if (!dayNode.has("lessons")) {
            sendMsg(chatId, "Завтра занятий нет."); return;
        }

        Map<String, List<JsonNode>> slots = groupLessonsByTime(dayNode);
        boolean isEvenWeek = DateUtils.isEvenWeek(tomorrow);
        List<JsonNode> resultLessons = new ArrayList<>();

        for (List<JsonNode> slot : slots.values()) {
            JsonNode lesson = selectLessonForWeek(slot, isEvenWeek);
            if (lesson != null) resultLessons.add(lesson);
        }

        if (resultLessons.isEmpty()) {
            sendMsg(chatId, "Завтра занятий нет."); return;
        }
```

```
        resultLessons.sort(Comparator.comparing(l -> l.get("start_time").asText()));
        String dayName = getDayName(tomorrow.getDayOfWeek());
        String formattedDate =
tomorrow.format(DateTimeFormatter.ofPattern("dd.MM.yyyy"));

        StringBuilder sb = new StringBuilder();
        sb.append("□ Завтра - ").append(dayName).append("
(").append(formattedDate).append(")\n\n");
        for (int i = 0; i < resultLessons.size(); i++) {
            sb.append(formatLesson(resultLessons.get(i), i + 1)).append("\n");
        }
        sendMsg(chatId, sb.toString());
    }

    private void handleWeekSelection(long chatId, String weekType) throws
Exception {
        String state = menuState.get(chatId);
        if ("week_selection_for_full".equals(state)) {
            handleFullWeekForType(chatId, weekType);
        } else {
            if ("both".equals(weekType)) {
                sendMsg(chatId, "Для дня недели выберите нечетную или четную
неделю."); return;
            }
            handleDayForWeek(chatId, weekType);
        }
    }

    private void handleDayForWeek(long chatId, String weekType) throws Exception
{
        String group = userGroup.get(chatId);
        String dayName = selectedDay.get(chatId);
        if (group == null || dayName == null) {
            sendMsg(chatId, "Сначала выберите группу и день."); return;
        }

        JsonNode schedule = timetableAPIservice.getRawSchedule(group).get(group);
        if (schedule == null || !schedule.has("days")) {
            sendMsg(chatId, "Расписание не найдено."); return;
        }

        int dayIndex = getDayIndex(dayName);
        if (dayIndex == -1) { sendMsg(chatId, "Неизвестный день."); return; }
```

```
        JsonNode dayNode = schedule.path("days").path(String.valueOf(dayIndex));
        if (!dayNode.has("lessons")) {
            String weekStr = "odd".equals(weekType) ? "нечетной" : "четной";
            sendMsg(chatId, "В " + dayName.toLowerCase() + " на " + weekStr + "
неделе занятий нет.");
            return;
        }

        Map<String, List<JsonNode>> slots = groupLessonsByTime(dayNode);
        boolean isEvenRequest = "even".equals(weekType);
        List<JsonNode> resultLessons = new ArrayList<>();

        for (List<JsonNode> slot : slots.values()) {
            JsonNode lesson = selectLessonForWeek(slot, isEvenRequest);
            if (lesson != null) resultLessons.add(lesson);
        }

        if (resultLessons.isEmpty()) {
            String weekStr = isEvenRequest ? "четной" : "нечетной";
            sendMsg(chatId, "В " + dayName.toLowerCase() + " на " + weekStr + "
неделе занятий нет.");
            return;
        }

        resultLessons.sort(Comparator.comparing(l -> l.get("start_time").asText()));
        String weekTitle = isEvenRequest ? "четная" : "нечетная";
        StringBuilder sb = new StringBuilder();
        sb.append("□ ").append(dayName).append("\n(неделя:
").append(weekTitle).append(")\n\n");
        for (int i = 0; i < resultLessons.size(); i++) {
            sb.append(formatLesson(resultLessons.get(i), i + 1)).append("\n");
        }
        sendMsg(chatId, sb.toString());
    }

    private void handleFullWeekForType(long chatId, String weekType) throws
Exception {
        String group = userGroup.get(chatId);
        if (group == null) { sendMsg(chatId, "Сначала укажите группу."); return; }

        JsonNode schedule = timetableAPIservice.getRawSchedule(group).get(group);
        if (schedule == null || !schedule.has("days")) {
            sendMsg(chatId, "Расписание не найдено."); return;
        }
```

```
        StringBuilder sb = new StringBuilder();
        if ("both".equals(weekType)) {
            sb.append("□ Расписание на обе недели:\n\n");
            appendCombinedWeek(sb, schedule);
        } else {
            boolean isEven = "even".equals(weekType);
            sb.append("□ Расписание на ").append(isEven ? "четную" :
"нечетную").append(" неделю:\n\n");
            appendWeek(sb, schedule, isEven);
        }
        sendMsg(chatId, sb.toString());
    }

    private void appendWeek(StringBuilder sb, JsonNode schedule, boolean
evenWeek) {
        String[] dayNames = {"Понедельник", "Вторник", "Среда", "Четверг",
"Пятница", "Суббота"};
        for (int i = 0; i < 6; i++) {
            JsonNode dayNode = schedule.path("days").path(String.valueOf(i));
            if (!dayNode.has("lessons")) continue;

            Map<String, List<JsonNode>> slots = groupLessonsByTime(dayNode);
            List<JsonNode> resultLessons = new ArrayList<>();

            for (List<JsonNode> slot : slots.values()) {
                JsonNode lesson = selectLessonForWeek(slot, evenWeek);
                if (lesson != null) resultLessons.add(lesson);
            }

            if (!resultLessons.isEmpty()) {
                resultLessons.sort(Comparator.comparing(l ->
l.get("start_time").asText()));
                sb.append("□ ").append(dayNames[i]).append("\n");
                for (int j = 0; j < resultLessons.size(); j++) {
                    sb.append(formatLesson(resultLessons.get(j), j + 1)).append("\n");
                }
                sb.append("\n");
            }
        }
    }

    private void appendCombinedWeek(StringBuilder sb, JsonNode schedule) {
```

```
        String[] dayNames = {"Понедельник", "Вторник", "Среда", "Четверг",
"Пятница", "Суббота"};
        for (int i = 0; i < 6; i++) {
            JsonNode dayNode = schedule.path("days").path(String.valueOf(i));
            if (!dayNode.has("lessons")) continue;

            sb.append("□ ").append(dayNames[i]).append("\n");
            Map<String, List<JsonNode>> slots = groupLessonsByTime(dayNode);
            int index = 1;

            for (Map.Entry<String, List<JsonNode>> entry : slots.entrySet()) {
                List<JsonNode> slot = entry.getValue();
                if (slot.size() == 1) {
                    sb.append(formatLesson(slot.get(0), index)).append("\n");
                } else {
                    sb.append(formatCombinedLesson(slot, index)).append("\n");
                }
                index++;
            }
            sb.append("\n");
        }
    }

    private String formatCombinedLesson(List<JsonNode> slot, int index) {
        JsonNode odd = null, even = null;
        for (JsonNode l : slot) {
            String w = l.get("week").asText();
            if (w.matches("[13]")) odd = l;
            else if (w.equals("2") || w.equals("4")) even = l;
        }

        StringBuilder sb = new StringBuilder();
        sb.append(index).append(". ");
        if (odd != null && even != null) {
            sb.append(odd.get("name").asText()).append("
(").append(odd.get("subjectType").asText()).append(") (нечетная) / ")
              .append(even.get("name").asText()).append("
(").append(even.get("subjectType").asText()).append(") (четная)\n");
        } else if (odd != null) {
            sb.append(odd.get("name").asText()).append("
(").append(odd.get("subjectType").asText()).append(") (нечетная)\n");
        } else if (even != null) {
            sb.append(even.get("name").asText()).append("
(").append(even.get("subjectType").asText()).append(") (четная)\n");
```

```
        }

        String start = slot.get(0).get("start_time").asText();
        String end = slot.get(0).get("end_time").asText();
        sb.append("□ ").append(start).append(" - ").append(end).append("\n");

        String teacherOdd = odd != null ? getTeacher(odd) : "";
        String teacherEven = even != null ? getTeacher(even) : "";
        if (!teacherOdd.isEmpty() && !teacherEven.isEmpty() &&
!teacherOdd.equals(teacherEven)) {
            sb.append("Преподаватель: ").append(teacherOdd).append(" (нечетная) /
").append(teacherEven).append(" (четная)\n");
        } else if (!teacherOdd.isEmpty()) sb.append("Преподаватель:
").append(teacherOdd).append("\n");
        else if (!teacherEven.isEmpty()) sb.append("Преподаватель:
").append(teacherEven).append("\n");

        String roomOdd = odd != null ? getRoom(odd) : "";
        String roomEven = even != null ? getRoom(even) : "";
        if (!roomOdd.isEmpty() && !roomEven.isEmpty() &&
!roomOdd.equals(roomEven)) {
            sb.append("Ауд. ").append(roomOdd).append(" (нечетная) / Ауд.
").append(roomEven).append(" (четная)\n");
        } else if (!roomOdd.isEmpty() && !roomOdd.equals("—")) sb.append("Ауд.
").append(roomOdd).append("\n");
        else if (!roomEven.isEmpty() && !roomEven.equals("—")) sb.append("Ауд.
").append(roomEven).append("\n");
        else if (roomOdd.equals("онлайн") || roomEven.equals("онлайн"))
sb.append("Форма: дистанционно\n");

        return sb.toString();
    }

    // Форматирование занятия
    private String formatLesson(JsonNode lesson, int index) {
        StringBuilder sb = new StringBuilder();
        sb.append(index).append(". ").append(lesson.get("name").asText())
          .append(" (").append(lesson.get("subjectType").asText()).append(")\n")
          .append("□ ").append(lesson.get("start_time").asText())
          .append(" - ").append(lesson.get("end_time").asText()).append("\n");

        String teacher = getTeacher(lesson);
        if (!teacher.isEmpty()) sb.append("Преподаватель:
").append(teacher).append("\n");
```

```
        String room = getRoom(lesson);
        if (room.equals("онлайн")) sb.append("Форма: дистанционно\n");
        else if (!room.isEmpty() && !room.equals("—")) sb.append("Ауд.
").append(room).append("\n");

        String url = lesson.path("url").asText("");
        if (!url.isEmpty() && !url.equals("null") && !url.equals("—")) {
            sb.append("Ссылка: ").append(url).append("\n");
        }
        return sb.toString();
    }

    private String getTeacher(JsonNode l) {
        String main = l.path("teacher").asText("").trim();
        String second = l.path("second_teacher").asText("").trim();
        if (main.isEmpty() && second.isEmpty()) return "";
        return second.isEmpty() ? main : main + ", " + second;
    }

    private String getRoom(JsonNode l) {
        String form = l.path("form").asText("");
        if (form.equalsIgnoreCase("online") || form.equalsIgnoreCase("онлайн")) return
"онлайн";
        String room = l.path("room").asText("");
        return room.isEmpty() ? "—" : room;
    }

    private int getDayIndex(String dayName) {
        return switch (dayName) {
            case "Понедельник" -> 0;
            case "Вторник" -> 1;
            case "Среда" -> 2;
            case "Четверг" -> 3;
            case "Пятница" -> 4;
            case "Суббота" -> 5;
            default -> -1;
        };
    }

    private String getDayName(java.time.DayOfWeek dayOfWeek) {
        return switch (dayOfWeek) {
            case MONDAY -> "Понедельник";
            case TUESDAY -> "Вторник";
```

```
            case WEDNESDAY -> "Среда";
            case THURSDAY -> "Четверг";
            case FRIDAY -> "Пятница";
            case SATURDAY -> "Суббота";
            default -> "День";
        };
    }

    // Меню
    private void showWeekSelectionForFullWeek(long chatId) {
        menuState.put(chatId, "week_selection_for_full");
        boolean isEven = DateUtils.isEvenWeek(LocalDate.now());
        String message = "□ Вся неделя\nСейчас идёт " + (isEven ? "четная" :
"нечетная") + " неделя.\nКакую неделю показать?";
        sendMessageWithKeyboard(chatId, message, createWeekSelectionMenu(true));
    }

    private void showWeekSelectionMenu(long chatId, String dayName) {
        selectedDay.put(chatId, dayName);
        menuState.put(chatId, "week_selection");
        boolean isEven = DateUtils.isEvenWeek(LocalDate.now());
        String message = "□ " + dayName + "\nСейчас идёт " + (isEven ? "четная" :
"нечетная") + " неделя.\nКакую неделю показать?";
        sendMessageWithKeyboard(chatId, message,
createWeekSelectionMenu(false));
    }

    private void showMainMenu(long chatId) {
        menuState.put(chatId, "main");
        sendMessageWithKeyboard(chatId, "Выберите действие:",
createMainMenu());
    }

    private void showDayMenu(long chatId) {
        menuState.put(chatId, "day_selection");
        sendMessageWithKeyboard(chatId, "Выберите день:", createDayMenu());
    }

    // Создание клавиатур
    private ReplyKeyboardMarkup createMainMenu() {
        return createKeyboard(new String[][]{
            {"Ближайшая пара", "Завтра"},
            {"Расписание по дням", "Вся неделя"},
            {"Сменить группу"}
```

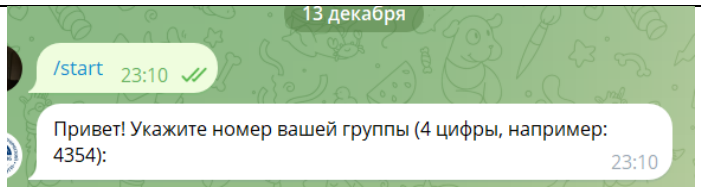
```
        });
    }

    private ReplyKeyboardMarkup createDayMenu() {
        return createKeyboard(new String[][]{
            {"Понедельник", "Вторник", "Среда"},
            {"Четверг", "Пятница", "Суббота"},
            {"Назад"}
        });
    }

    private ReplyKeyboardMarkup createWeekSelectionMenu(boolean forFullWeek)
{
        List<String[]> rows = new ArrayList<>();
        rows.add(new String[]{"Нечетная неделя", "Четная неделя"});

        List<String> lastRow = new ArrayList<>();
        if (forFullWeek) lastRow.add("Обе недели");
        lastRow.add("Назад");
        rows.add(lastRow.toArray(new String[0]));

        return createKeyboard(rows.toArray(new String[0][]));
    }

    private ReplyKeyboardMarkup createKeyboard(String[][] buttons) {
        ReplyKeyboardMarkup keyboard = new ReplyKeyboardMarkup();
        List<KeyboardRow> rows = new ArrayList<>();

        for (String[] rowButtons : buttons) {
            KeyboardRow row = new KeyboardRow();
            for (String button : rowButtons) {
                row.add(button);
            }
            rows.add(row);
        }

        keyboard.setKeyboard(rows);
        keyboard.setResizeKeyboard(true);
        return keyboard;
    }

    // Вспомогательные методы
    private void sendMsg(long chatId, String text) {
        try {
```

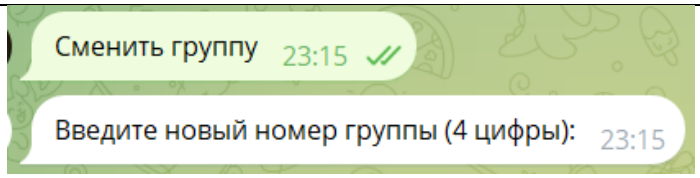
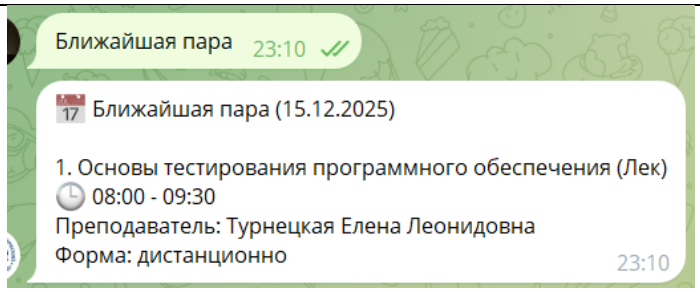
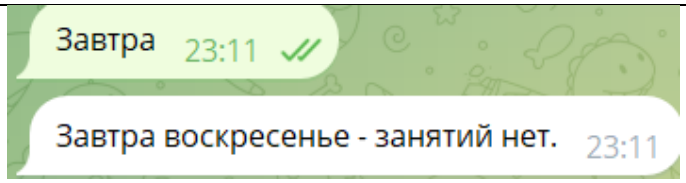
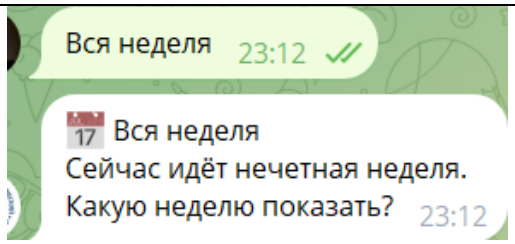
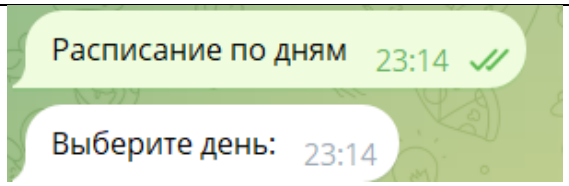
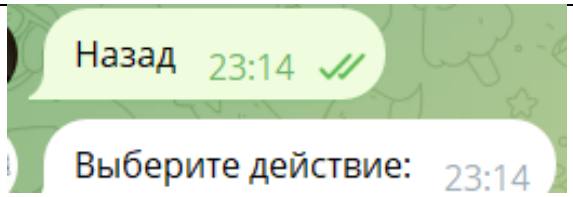
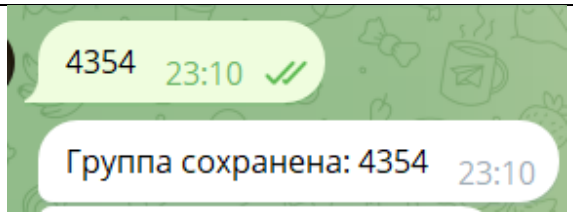
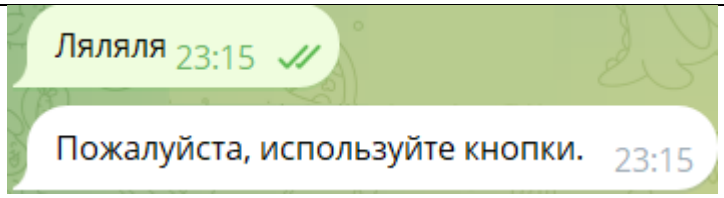
```
            execute(SendMessage.builder()
                .chatId(String.valueOf(chatId))
                .text(text)
                .build());
        } catch (TelegramApiException e) {
            e.printStackTrace();
        }
    }

    private void sendMessageWithKeyboard(long chatId, String text,
ReplyKeyboardMarkup keyboard) {
        try {
            execute(SendMessage.builder()
                .chatId(String.valueOf(chatId))
                .text(text)
                .replyMarkup(keyboard)
                .build());
        } catch (TelegramApiException e) {
            e.printStackTrace();
        }
    }

    private void clearUserData(long chatId) {
        userGroup.remove(chatId);
        selectedDay.remove(chatId);
        menuState.remove(chatId);
    }
}

package ru.etu.timetable_bot.api;

import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import okhttp3.OkHttpClient;
import okhttp3.Request;
import okhttp3.Response;
import org.springframework.stereotype.Service;

@Service
public class TimetableAPIservice {
    private final OkHttpClient httpClient = new OkHttpClient();
    private final ObjectMapper objectMapper = new ObjectMapper();

    public JsonNode getRawSchedule(String groupNumber) throws Exception {
```

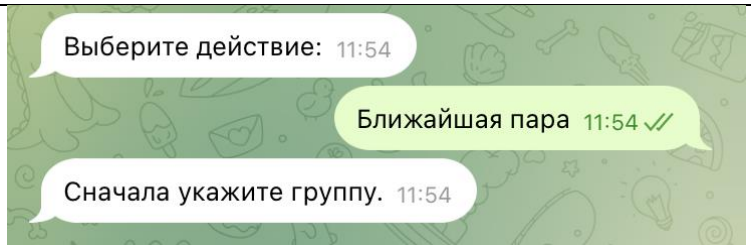
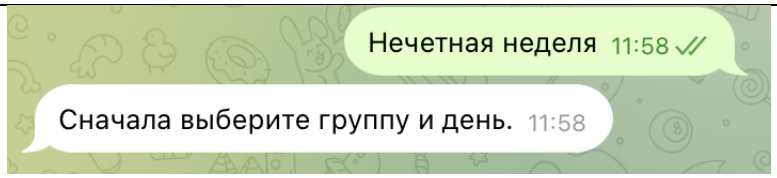
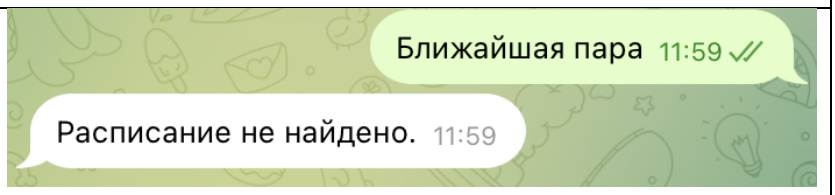
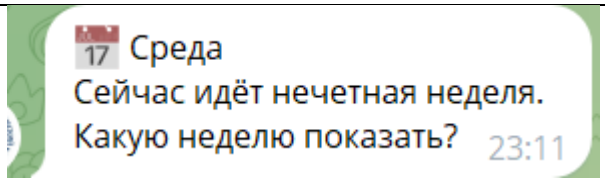
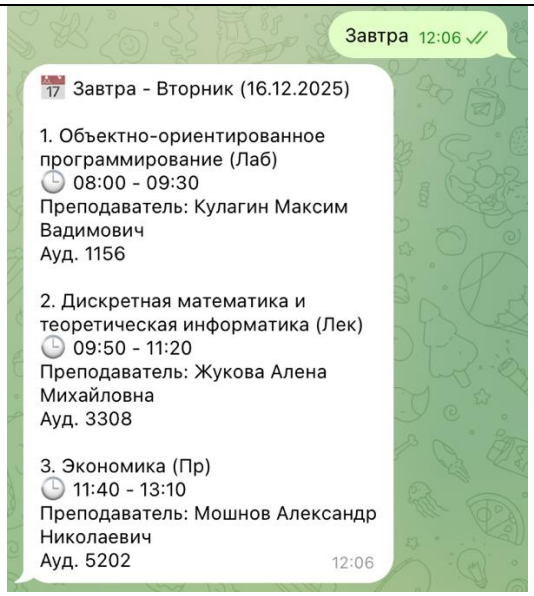
```
        String url = String.format(

"https://digital.etu.ru/api/mobile/schedule?groupNumber=%s&season=autumn&year
=2025&joinWeeks=true&withURL=true",
            groupNumber
        );

        Request request = new Request.Builder()
            .url(url)
            .addHeader("User-Agent", "Telegram Bot")
            .build();

        try (Response response = httpClient.newCall(request).execute()) {
            if (!response.isSuccessful()) {
                throw new RuntimeException("API error: " + response.code());
            }
            return objectMapper.readTree(response.body().string());
        }
    }
}
```

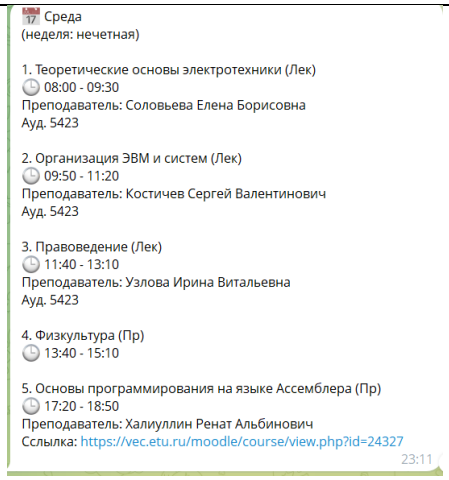
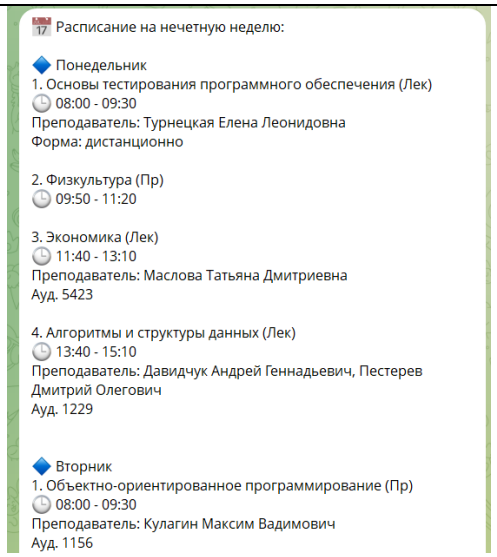
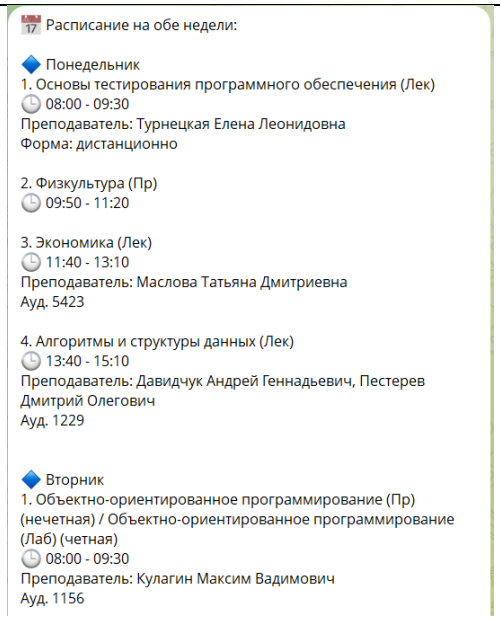
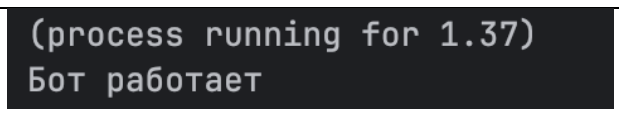
5. Описание интерфейса пользователя программы

Telegram-бот для получения расписания реализует интуитивно понятный интерфейс на основе клавиатурных меню, который позволяет пользователям легко получать информацию о занятиях без необходимости запоминания специальных команд. Ниже представлено описание этого интерфейса:

Inp/Outp	Пример
Inp1: /start Outp1: Привет! Укажите номер вашей группы (4 цифры, например: 4354):	

Inp2: Сменить группу Outp2: Введите новый номер группы (4 цифры):	
Inp3: Ближайшая пара Outp3: Ближайшая пара (дд.мм.гггг) Форматированная информация о паре.	
Inp4: Завтра Outp4: Завтра воскресенье – занятий нет.	
Inp5: Вся неделя Outp5: Вся неделя Сейчас идет четная/нечетная неделя. Какую неделю показать?	
Inp6: Расписание по дням Outp6: Выберете день:	
Inp7: Назад Outp7: Выберете действие:	
Inp8: \d{4}(ввод группы) Outp8: Группа сохранена: номер группы	
Outp9: Пожалуйста, используйте кнопки.	

Outp12: Сначала укажите группу.	
Outp13: Сначала выберите группу и день.	
Outp14: Расписание не найдено.	
Outp17: Ближайшие 2 недели --- занятий нет.	Сейчас невозможен вывод данного сообщения
Outp18: Завтра занятий нет.	Сейчас невозможен вывод данного сообщения
Outp20: [день недели] Сейчас идёт [четная/нечетная] неделя. Какую неделю показать?	
Outp21: Завтра - [день недели] (дд.мм.гггг) Форматированное расписание на день.	

Outp22: [день недели] (неделя: [четная/нечетная]) Форматированное расписание на день.	
Outp23: Расписание на [четную/нечетную] неделю: Форматированное расписание на неделю	
Outp24: Расписание на обе недели: Комбинированное расписание на обе недели.	
Консольные выводы	
Outp25: Бот работает	

ЗАКЛЮЧЕНИЕ

В рамках курсовой работы разработан Telegram-бот для автоматизированного предоставления расписания занятий студентам ЛЭТИ. Бот успешно интегрирован с API университета, поддерживает выбор группы, отображение расписания в различных режимах и учёт чётности недель. Реализованы все основные функции: удобная кнопочная навигация, обработка состояния пользователей, корректное форматирование вывода. Система работает стабильно, готова к использованию и может быть расширена в будущем. Разработанное решение демонстрирует практическую значимость и подтверждает эффективность использования современных цифровых инструментов для оптимизации учебного процесса.

Ссылка на гитхаб:

https://github.com/varkirsova/oop_timetableBot.git

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Официальное API расписания СПбГЭТУ «ЛЭТИ» // Digital ETU. URL: https://digital.etu.ru/api/mobile/schedule (дата обращения: 10.12.2024).
2. Jackson Project – JSON для Java // GitHub. URL: https://github.com/FasterXML/jackson (дата обращения: 10.12.2024).
3. Spring Boot 3.0 документация // Spring Framework. URL: https://docs.spring.io/spring-boot/docs/3.0.0/reference/htmlsingle/ (дата обращения: 10.12.2024)
4. ГОСТ 7.32-2017-1. Межгосударственный стандарт. ОТЧЕТ О НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ. Структура и правила оформления.
5. Требования к оформлению учебных работ в СПбГЭТУ «ЛЭТИ» // Официальный сайт университета. URL: https://etu.ru/ru/obrazovanie/trebovaniya-k-oformleniyu-rabot (дата обращения: 10.12.2025).
6. TelegramBots: библиотека для создания ботов на Java // GitHub. URL: https://github.com/rubenlagus/TelegramBots (дата обращения: 10.12.2025).

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД

TimetableBot.java

```
package ru.etu.timetable_bot.bot;

import java.time.format.DateTimeFormatter;
import java.util.*;
import java.util.concurrent.ConcurrentHashMap;
import java.time.*;
import org.springframework.beans.factory.annotation.Value;
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 ru.etu.timetable_bot.api.TimetableAPIservice;

import ru.etu.timetable_bot.utils.DateUtils;
import com.fasterxml.jackson.databind.JsonNode;

import org.telegram.telegrambots.meta.api.objects.replykeyboard.ReplyKeyboardMarkup;
import org.telegram.telegrambots.meta.api.objects.replykeyboard.buttons.KeyboardRow;
import java.util.ArrayList;

import java.util.List;

@Component
public class TimetableBot extends TelegramLongPollingBot {

    private final Map<Long, String> userSelectedDay = new ConcurrentHashMap<>();
    private final Map<Long, String> userGroup = new ConcurrentHashMap<>();
    private final Map<Long, String> userMenuState = new ConcurrentHashMap<>();


    @Value("${token}")
    private String botToken;

    @Value("${telegram.bot-name}")
    private String botName;

    private final TimetableAPIservice timetableAPIservice;


    public TimetableBot(TimetableAPIservice timetableAPIservice) {
        this.timetableAPIservice = timetableAPIservice;
    }

    @Override
    public String getBotUsername() {
```

```
        return botName;
    }

    @Override
    public String getBotToken() {
        return botToken;
    }

    @Override
    public void onUpdateReceived(Update update) {
        if (update.hasMessage() && update.getMessage().hasText()) {
            String text = update.getMessage().getText().trim();
            long chatId = update.getMessage().getChatId();

            try {
                if (text.equals("/start")) {
                    String group = userGroup.get(chatId);
                    userGroup.remove(chatId);
                    userSelectedDay.remove(chatId);
                    userMenuState.remove(chatId);
                    if (group == null) {
                        sendMsg(chatId, "Привет! Укажите номер вашей группы (4 цифры, например:
4354):");
                    } else {
                        sendMsg(chatId, "Ваша группа: " + group + "\nВыберите действие:");
                        showMainMenu(chatId);
                    }
                } else if (text.matches("\\d{4}")) {
                    userGroup.put(chatId, text);
                    sendMsg(chatId, "Группа сохранена: " + text);
                    showMainMenu(chatId);
                } else if (text.equals("Сменить группу")) {
                    sendMsg(chatId, "Введите новый номер группы (4 цифры):");
                } else if (text.equals("Ближайшая пара")) {
                    handleNearLesson(chatId);
                } else if (text.equals("Завтра")) {
                    handleTomorrow(chatId);
                } else if (text.equals("Вся неделя")) {
                    showWeekSelectionForFullWeek(chatId);
                } else if (text.equals("Расписание по дням")) {
                    showDayMenu(chatId);
                } else if (List.of("Понедельник", "Вторник", "Среда", "Четверг", "Пятница",
"Суббота").contains(text)) {
                    showWeekSelectionMenu(chatId, text);
                } else if (text.equals("Нечетная неделя")) {
                    handleWeekSelection(chatId, "odd");
                } else if (text.equals("Четная неделя")) {
                    handleWeekSelection(chatId, "even");
                } else if (text.equals("Обе недели")) {
                    handleWeekSelection(chatId, "both");
                } else if (text.equals("Назад")) {
                    String state = userMenuState.get(chatId);
```

```
                if ("week_selection_for_full".equals(state)) {
                    showMainMenu(chatId);
                } else if ("week_selection".equals(state)) {
                    showDayMenu(chatId);
                } else if ("day_selection".equals(state)) {
                    showMainMenu(chatId);
                } else {
                    showMainMenu(chatId);
                }
            } else {
                sendMsg(chatId, "Пожалуйста, используйте кнопки.");
            }
        } catch (Exception e) {
            e.printStackTrace();
            sendMsg(chatId, "Ошибка: " + e.getMessage());
        }
    }
}


private void handleDayForWeek(long chatId, String weekType) throws Exception {
    String group = userGroup.get(chatId);
    String dayName = userSelectedDay.get(chatId);
    if (group == null || dayName == null) {
        sendMsg(chatId, "Сначала выберите группу и день.");
        return;
    }

    JsonNode rawSchedule = timetableAPIservice.getRawSchedule(group);
    JsonNode groupNode = rawSchedule.get(group);
    if (groupNode == null || !groupNode.has("days")) {
        sendMsg(chatId, "Расписание не найдено.");
        return;
    }

    Integer dayIndex = getDayIndex(dayName);
    if (dayIndex == null) {
        sendMsg(chatId, "Неизвестный день.");
        return;
    }

    JsonNode dayNode = groupNode.path("days").path(String.valueOf(dayIndex));
    if (!dayNode.has("lessons") || dayNode.get("lessons").size() == 0) {
        String weekStr = "odd".equals(weekType) ? "нечетной" : "четной";
        sendMsg(chatId, "В " + dayName.toLowerCase() + " на " + weekStr + " неделе занятий
нет.");
        return;
    }

    Map<String, List<JsonNode>> slots = new LinkedHashMap<>();
    for (JsonNode l : dayNode.get("lessons")) {
        String key = l.get("start_time").asText() + "-" + l.get("end_time").asText();
```

```
    slots.computeIfAbsent(key, k -> new ArrayList<>()).add(l);
}

List<JsonNode> resultLessons = new ArrayList<>();
boolean isEvenRequest = "even".equals(weekType);

for (List<JsonNode> slot : slots.values()) {
    JsonNode chosen = null;

    if (isEvenRequest) {
        for (JsonNode l : slot) {
            String w = l.get("week").asText();
            if ("2".equals(w)) {
                chosen = l;
                break;
            }
        }
        if (chosen == null) {
            for (JsonNode l : slot) {
                String w = l.get("week").asText();
                if ("1".equals(w) || "3".equals(w)) {
                    chosen = l;
                    break;
                }
            }
        }
    } else {
        for (JsonNode l : slot) {
            String w = l.get("week").asText();
            if ("1".equals(w) || "3".equals(w)) {
                chosen = l;
                break;
            }
        }
        if (chosen == null) {
            for (JsonNode l : slot) {
                String w = l.get("week").asText();
                if ("2".equals(w)) {
                    chosen = l;
                    break;
                }
            }
        }
    }

    if (chosen != null) {
        resultLessons.add(chosen);
    }
}

if (resultLessons.isEmpty()) {
    String weekStr = isEvenRequest ? "четной" : "нечетной";
```

```
            sendMsg(chatId, "В " + dayName.toLowerCase() + " на " + weekStr + " неделе занятий
нет.");
            return;
        }

        resultLessons.sort(Comparator.comparing(l -> l.get("start_time").asText()));

        StringBuilder sb = new StringBuilder();
        String weekTitle = isEvenRequest ? "четная" : "нечетная";
        sb.append("⌨ ").append(dayName).append("\n(неделя:
").append(weekTitle).append(")\n\n");
        int index = 1;
        for (JsonNode l : resultLessons) {
            sb.append(formatLessonWithTime(l, index)).append("\n");
            index++;
        }

        sendMsg(chatId, sb.toString());
    }

    private void handleNearLesson(long chatId) throws Exception {
        String group = userGroup.get(chatId);
        if (group == null) {
            sendMsg(chatId, "Сначала укажите группу.");
            return;
        }

        JsonNode rawSchedule = timetableAPIservice.getRawSchedule(group);
        JsonNode groupNode = rawSchedule.get(group);
        if (groupNode == null || !groupNode.has("days")) {
            sendMsg(chatId, "Расписание не найдено.");
            return;
        }

        LocalDateTime now = LocalDateTime.now();

        for (int offset = 0; offset < 14; offset++) {
            LocalDate date = now.toLocalDate().plusDays(offset);
            int dayIndex = date.getDayOfWeek().getValue() - 1;
            if (dayIndex >= 6) continue;

            JsonNode dayNode = groupNode.path("days").path(String.valueOf(dayIndex));
            if (!dayNode.has("lessons")) continue;

            Map<String, List<JsonNode>> slots = new LinkedHashMap<>();
            for (JsonNode l : dayNode.get("lessons")) {
                String key = l.get("start_time").asText() + "-" + l.get("end_time").asText();
                slots.computeIfAbsent(key, k -> new ArrayList<>()).add(l);
            }

            List<Map.Entry<String, List<JsonNode>>> sortedSlots = new
ArrayList<>(slots.entrySet());
```

```
sortedSlots.sort(Map.Entry.comparingByKey());

for (Map.Entry<String, List<JsonNode>> entry : sortedSlots) {
    LocalTime start = LocalTime.parse(entry.getKey().split("-")[0]);
    LocalDateTime lessonTime = date.atTime(start);
    if (lessonTime.isBefore(now)) {
        continue;
    }

    JsonNode chosen = null;
    boolean isEvenDate = DateUtils.isEvenWeek(date);

    if (isEvenDate) {
        for (JsonNode l : entry.getValue()) {
            String w = l.get("week").asText();
            if ("2".equals(w)) {
                chosen = l;
                break;
            }
        }
        if (chosen == null) {
            for (JsonNode l : entry.getValue()) {
                String w = l.get("week").asText();
                if ("1".equals(w) || "3".equals(w)) {
                    chosen = l;
                    break;
                }
            }
        }
    } else {
        for (JsonNode l : entry.getValue()) {
            String w = l.get("week").asText();
            if ("1".equals(w) || "3".equals(w)) {
                chosen = l;
                break;
            }
        }
        if (chosen == null) {
            for (JsonNode l : entry.getValue()) {
                String w = l.get("week").asText();
                if ("2".equals(w)) {
                    chosen = l;
                    break;
                }
            }
        }
    }

    if (chosen != null) {
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd.MM.yyyy");
        String formattedDate = date.format(formatter);
```

```
            StringBuilder sb = new StringBuilder();
            sb.append("📅 Ближайшая пара (").append(formattedDate).append(") \n\n");
            sb.append(formatLessonWithTime(chosen, 1));

            sendMsg(chatId, sb.toString());
            return;
        }
      }
    }

    sendMsg(chatId, "Ближайшие 2 недели — занятий нет.");
}

private void handleTomorrow(long chatId) throws Exception {
    String group = userGroup.get(chatId);
    if (group == null) {
        sendMsg(chatId, "Сначала укажите группу.");
        return;
    }

    LocalDate today = LocalDate.now();
    LocalDate tomorrow = today.plusDays(1);

    DayOfWeek tomorrowDayOfWeek = tomorrow.getDayOfWeek();

    if (tomorrowDayOfWeek == DayOfWeek.SUNDAY) {
        sendMsg(chatId, "Завтра воскресенье - занятий нет.");
        return;
    }

    int dayIndex = tomorrowDayOfWeek.getValue() - 1; // Monday=0, Tuesday=1, ..Saturday=5

    boolean isEvenWeek = DateUtils.isEvenWeek(tomorrow);

    JsonNode rawSchedule = timetableAPIservice.getRawSchedule(group);
    JsonNode groupNode = rawSchedule.get(group);
    if (groupNode == null || !groupNode.has("days")) {
        sendMsg(chatId, "Расписание не найдено.");
        return;
    }

    JsonNode dayNode = groupNode.path("days").path(String.valueOf(dayIndex));
    if (!dayNode.has("lessons")) {
        sendMsg(chatId, "Завтра занятий нет.");
        return;
    }

    Map<String, List<JsonNode>> slots = new LinkedHashMap<>();
    for (JsonNode l : dayNode.get("lessons")) {
        String key = l.get("start_time").asText() + "-" + l.get("end_time").asText();
        slots.computeIfAbsent(key, k -> new ArrayList<>()).add(l);
```

```
}

List<JsonNode> resultLessons = new ArrayList<>();

for (List<JsonNode> slot : slots.values()) {
    JsonNode chosen = null;

    if (isEvenWeek) {
        for (JsonNode l : slot) {
            String w = l.get("week").asText();
            if ("2".equals(w)) {
                chosen = l;
                break;
            }
        }
        if (chosen == null) {
            for (JsonNode l : slot) {
                String w = l.get("week").asText();
                if ("1".equals(w) || "3".equals(w)) {
                    chosen = l;
                    break;
                }
            }
        }
    } else {
        for (JsonNode l : slot) {
            String w = l.get("week").asText();
            if ("1".equals(w) || "3".equals(w)) {
                chosen = l;
                break;
            }
        }
        if (chosen == null) {
            for (JsonNode l : slot) {
                String w = l.get("week").asText();
                if ("2".equals(w)) {
                    chosen = l;
                    break;
                }
            }
        }
    }

    if (chosen != null) {
        resultLessons.add(chosen);
    }
}

if (resultLessons.isEmpty()) {
    sendMsg(chatId, "Завтра занятий нет.");
    return;
}
```

```
        String dayName = switch (tomorrowDayOfWeek) {
            case MONDAY -> "Понедельник";
            case TUESDAY -> "Вторник";
            case WEDNESDAY -> "Среда";
            case THURSDAY -> "Четверг";
            case FRIDAY -> "Пятница";
            case SATURDAY -> "Суббота";
            default -> "День";
        };

        DateTimeFormatter dateFormatter = DateTimeFormatter.ofPattern("dd.MM.yyyy");
        String formattedDate = tomorrow.format(dateFormatter);

        StringBuilder sb = new StringBuilder();
        sb.append("📅 Завтра - ").append(dayName)
                .append(" (").append(formattedDate).append(")\n\n");

        int index = 1;
        for (JsonNode l : resultLessons) {
            sb.append(formatLessonWithTime(l, index)).append("\n");
            index++;
        }

        sendMsg(chatId, sb.toString());
    }

    private void appendWeek(StringBuilder sb, JsonNode groupNode, boolean evenWeek) {
        String[] dayNames = {"Понедельник", "Вторник", "Среда", "Четверг", "Пятница",
"Суббота"};

        for (int i = 0; i < 6; i++) {
            JsonNode dayNode = groupNode.path("days").path(String.valueOf(i));
            if (!dayNode.has("lessons")) continue;

            Map<String, List<JsonNode>> slots = new LinkedHashMap<>();
            for (JsonNode l : dayNode.get("lessons")) {
                String timeKey = l.get("start_time").asText() + "-" + l.get("end_time").asText();
                slots.computeIfAbsent(timeKey, k -> new ArrayList<>()).add(l);
            }

            List<JsonNode> resultLessons = new ArrayList<>();
            for (List<JsonNode> slot : slots.values()) {
                JsonNode chosen = null;

                if (evenWeek) {
                    for (JsonNode l : slot) {
                        String w = l.get("week").asText();
                        if ("2".equals(w)) {
                            chosen = l;
                            break;
                        }
```

```
                }
                if (chosen == null) {
                    for (JsonNode l : slot) {
                        String w = l.get("week").asText();
                        if ("1".equals(w) || "3".equals(w)) {
                            chosen = l;
                            break;
                        }
                    }
                }
            } else {
                for (JsonNode l : slot) {
                    String w = l.get("week").asText();
                    if ("1".equals(w) || "3".equals(w)) {
                        chosen = l;
                        break;
                    }
                }
                if (chosen == null) {
                    for (JsonNode l : slot) {
                        String w = l.get("week").asText();
                        if ("2".equals(w)) {
                            chosen = l;
                            break;
                        }
                    }
                }
            }

            if (chosen != null) {
                resultLessons.add(chosen);
            }
        }

        if (!resultLessons.isEmpty()) {
            resultLessons.sort(Comparator.comparing(l -> l.get("start_time").asText()));

            sb.append("\uD83D\uDD37").append(" ").append(dayNames[i]).append("\n");
            int index = 1;
            for (JsonNode l : resultLessons) {
                sb.append(formatLessonWithTime(l, index)).append("\n");
                index++;
            }
            sb.append("\n");
        }
    }

    if (sb.length() > 0 && sb.charAt(sb.length() - 1) == '\n') {
        sb.setLength(sb.length() - 1);
    }
}
```

```
private void handleWeekSelection(long chatId, String weekType) throws Exception {
    String state = userMenuState.get(chatId);
    if ("week_selection_for_full".equals(state)) {
        handleFullWeekForType(chatId, weekType);
    } else {
        if ("both".equals(weekType)) {
            sendMsg(chatId, "Для дня недели выберите нечетную или четную неделю.");
            return;
        }
        String group = userGroup.get(chatId);
        String dayName = userSelectedDay.get(chatId);
        if (group == null || dayName == null) {
            sendMsg(chatId, "Сначала выберите группу и день.");
            return;
        }
        handleDayForWeek(chatId, weekType);
    }
}


private void showWeekSelectionForFullWeek(long chatId) {
    userMenuState.put(chatId, "week_selection_for_full");
    boolean isEven = DateUtils.isEvenWeek(LocalDate.now());
    String currentWeek = isEven ? "четная" : "нечетная";


    String message = String.format(
            "⌨ Вся неделя\nСейчас идёт %s неделя.\nКакую неделю показать?",
            currentWeek
    );

    SendMessage msg = SendMessage.builder()
            .chatId(String.valueOf(chatId))
            .text(message)
            .replyMarkup(createWeekSelectionMenu(true))
            .build();

    try {
        execute(msg);
    } catch (TelegramApiException e) {
        e.printStackTrace();
    }
}

private void showWeekSelectionMenu(long chatId, String dayName) {
    userSelectedDay.put(chatId, dayName);
    userMenuState.put(chatId, "week_selection");
    boolean isEven = DateUtils.isEvenWeek(LocalDate.now());
    String currentWeek = isEven ? "четная" : "нечетная";

    String message = String.format(
```

```
            "📅 %s\nСейчас идёт %s неделя.\nКакую неделю показать?",
            dayName, currentWeek
    );

    SendMessage msg = SendMessage.builder()
            .chatId(String.valueOf(chatId))
            .text(message)
            .replyMarkup(createWeekSelectionMenu(false))
            .build();

    try {
        execute(msg);
    } catch (TelegramApiException e) {
        e.printStackTrace();
    }
}

private ReplyKeyboardMarkup createWeekSelectionMenu(boolean forFullWeek) {
    ReplyKeyboardMarkup keyboard = new ReplyKeyboardMarkup();
    List<KeyboardRow> rows = new ArrayList<>();

    KeyboardRow r1 = new KeyboardRow();
    r1.add("Нечетная неделя");
    r1.add("Четная неделя");
    rows.add(r1);

    KeyboardRow r2 = new KeyboardRow();
    if (forFullWeek) {
        r2.add("Обе недели");
    }
    r2.add("Назад");
    rows.add(r2);

    keyboard.setKeyboard(rows);
    keyboard.setResizeKeyboard(true);
    return keyboard;
}

private void handleFullWeekForType(long chatId, String weekType) throws Exception {
    String group = userGroup.get(chatId);
    if (group == null) {
        sendMsg(chatId, "Сначала укажите группу.");
        return;
    }

    JsonNode rawSchedule = timetableAPIservice.getRawSchedule(group);
    JsonNode groupNode = rawSchedule.get(group);
    if (groupNode == null || !groupNode.has("days")) {
        sendMsg(chatId, "Расписание не найдено.");
        return;
    }
```

```
        StringBuilder sb = new StringBuilder();
        if ("both".equals(weekType)) {
            sb.append("🗓 Расписание на обе недели:\n\n");
            appendCombinedWeek(sb, groupNode);
        } else {
            boolean isEven = "even".equals(weekType);
            String title = isEven ? "четную" : "нечетную";
            sb.append("🗓 Расписание на ").append(title).append(" неделю:\n\n");
            appendWeek(sb, groupNode, isEven);
        }
        sendMsg(chatId, sb.toString());
    }

    private void appendCombinedWeek(StringBuilder sb, JsonNode groupNode) {
        String[] dayNames = {"Понедельник", "Вторник", "Среда", "Четверг", "Пятница",
"Суббота"};

        for (int i = 0; i < 6; i++) {
            JsonNode dayNode = groupNode.path("days").path(String.valueOf(i));
            if (!dayNode.has("lessons")) continue;

            Map<String, List<JsonNode>> slots = new LinkedHashMap<>();
            for (JsonNode l : dayNode.get("lessons")) {
                String key = l.get("start_time").asText() + "-" + l.get("end_time").asText();
                slots.computeIfAbsent(key, k -> new ArrayList<>()).add(l);
            }

            if (slots.isEmpty()) continue;

            sb.append("\uD83D\uDD37").append(" ").append(dayNames[i]).append("\n");
            int index = 1;
            for (List<JsonNode> slot : slots.values()) {
                if (slot.size() == 1) {
                    sb.append(formatLessonWithTime(slot.get(0), index));
                } else {
                    String start = slot.get(0).get("start_time").asText();
                    String end = slot.get(0).get("end_time").asText();

                    JsonNode oddLesson = null, evenLesson = null;
                    for (JsonNode l : slot) {
                        String w = l.get("week").asText();
                        if ("1".equals(w) || "3".equals(w)) {
                            oddLesson = l;
                        } else if ("2".equals(w) || "4".equals(w)) {
                            evenLesson = l;
                        }
                    }

                    sb.append(index).append(". ");
                    if (oddLesson != null && evenLesson != null) {
```

```
                sb.append(oddLesson.get("name").asText()).append("
(").append(oddLesson.get("subjectType").asText()).append(") (нечетная) / ")
                        .append(evenLesson.get("name").asText()).append("
(").append(evenLesson.get("subjectType").asText()).append(") (четная)\n");
            } else if (oddLesson != null) {
                sb.append(oddLesson.get("name").asText()).append("
(").append(oddLesson.get("subjectType").asText()).append(") (нечетная)\n");
            } else if (evenLesson != null) {
                sb.append(evenLesson.get("name").asText()).append("
(").append(evenLesson.get("subjectType").asText()).append(") (четная)\n");
            }

            sb.append("🕓 ").append(start).append(" - ").append(end).append("\n");

            String teacherOdd = oddLesson != null ? getTeacher(oddLesson) : null;
            String teacherEven = evenLesson != null ? getTeacher(evenLesson) : null;
            if (teacherOdd != null && teacherEven != null && !teacherOdd.equals(teacherEven))
{
                sb.append("Преподаватель: ").append(teacherOdd).append(" (нечетная) /
").append(teacherEven).append(" (четная)\n");
            } else if (teacherOdd != null) {
                sb.append("Преподаватель: ").append(teacherOdd).append("\n");
            } else if (teacherEven != null) {
                sb.append("Преподаватель: ").append(teacherEven).append("\n");
            }

            String roomOdd = oddLesson != null ? getRoom(oddLesson) : null;
            String roomEven = evenLesson != null ? getRoom(evenLesson) : null;
            if (roomOdd != null && roomEven != null && !roomOdd.equals(roomEven)) {
                sb.append("Ауд. ").append(roomOdd).append(" (нечетная) / Ауд.
").append(roomEven).append(" (четная)\n");
            } else if (roomOdd != null && !"—".equals(roomOdd)) {
                sb.append("Ауд. ").append(roomOdd).append("\n");
            } else if (roomEven != null && !"—".equals(roomEven)) {
                sb.append("Ауд. ").append(roomEven).append("\n");
            } else if ("онлайн".equals(roomOdd) || "онлайн".equals(roomEven)) {
                sb.append("Форма: дистанционно\n");
            }
        }
        index++;
        sb.append("\n");
    }
    sb.append("\n");
  }

  if (sb.length() > 0 && sb.charAt(sb.length() - 1) == '\n') {
    sb.setLength(sb.length() - 1);
  }
 }


 private String formatLessonWithTime(JsonNode l, int index) {
```

```
    String start = l.get("start_time").asText();
    String end = l.get("end_time").asText();
    String subject = l.get("name").asText();
    String type = l.get("subjectType").asText();
    String teacher = getTeacher(l);
    String room = getRoom(l);

    StringBuilder sb = new StringBuilder();

    sb.append(index).append(". ").append(subject).append(" (").append(type).append(")\n");

    sb.append("🕒 ").append(start).append(" - ").append(end).append("\n");

    if (!teacher.isEmpty()) {
        sb.append("Преподаватель: ").append(teacher).append("\n");
    }

    if ("онлайн".equalsIgnoreCase(room)) {
        sb.append("Форма: дистанционно\n");
    } else if (!room.isEmpty() && !room.equals("—")) {
        sb.append("Ауд. ").append(room).append("\n");
    }

    JsonNode urlNode = l.path("url");
    String url = null;
    if (!urlNode.isMissingNode() && !urlNode.isNull() && urlNode.asText() != null) {
        url = urlNode.asText().trim();
    }
    if (url != null && !url.isEmpty() && !url.equals("null") && !url.equals("—")) {
        sb.append("Сслылка: ").append(url).append("\n");
    }

    return sb.toString();
}


private String getTeacher(JsonNode l) {
    String main = l.path("teacher").asText("").trim();
    String second = l.path("second_teacher").asText("").trim();

    if (main.isEmpty() && second.isEmpty()) {
        return "";
    }
    if (second.isEmpty()) {
        return main;
    }
    return main + ", " + second;
}

private String getRoom(JsonNode l) {
    String form = l.path("form").asText("");
    if ("online".equalsIgnoreCase(form) || "онлайн".equalsIgnoreCase(form)) {
```

```
            return "онлайн";
        }
        String room = l.path("room").asText("");
        return room.isEmpty() ? "—" : room;
    }

    private Integer getDayIndex(String dayName) {
        return switch (dayName) {
            case "Понедельник" -> 0;
            case "Вторник" -> 1;
            case "Среда" -> 2;
            case "Четверг" -> 3;
            case "Пятница" -> 4;
            case "Суббота" -> 5;
            default -> null;
        };
    }


    private void sendMsg(long chatId, String text) {
        SendMessage message = SendMessage.builder()
                .chatId(String.valueOf(chatId))
                .text(text)
                .build();
        try {
            execute(message);
        } catch (TelegramApiException e) {
            e.printStackTrace();
        }
    }

    private void showMainMenu(long chatId) {
        userMenuState.put(chatId, "main");
        SendMessage msg = SendMessage.builder()
                .chatId(String.valueOf(chatId))
                .text("Выберите действие:")
                .replyMarkup(createMainMenu())
                .build();
        try {
            execute(msg);
        } catch (TelegramApiException e) {
            e.printStackTrace();
        }
    }

    private ReplyKeyboardMarkup createMainMenu() {
        ReplyKeyboardMarkup keyboard = new ReplyKeyboardMarkup();
        List<KeyboardRow> rows = new ArrayList<>();

        KeyboardRow row1 = new KeyboardRow();
        row1.add("Ближайшая пара");
        row1.add("Завтра");
```

```
        rows.add(row1);

        KeyboardRow row2 = new KeyboardRow();
        row2.add("Расписание по дням");
        row2.add("Вся неделя");
        rows.add(row2);

        KeyboardRow row3 = new KeyboardRow();
        row3.add("Сменить группу");
        rows.add(row3);

        keyboard.setKeyboard(rows);
        keyboard.setResizeKeyboard(true);
        return keyboard;
    }

    private void showDayMenu(long chatId) {
        userMenuState.put(chatId, "day_selection");
        SendMessage msg = SendMessage.builder()
                .chatId(String.valueOf(chatId))
                .text("Выберите день:")
                .replyMarkup(createDayMenu())
                .build();
        try {
            execute(msg);
        } catch (TelegramApiException e) {
            e.printStackTrace();
        }
    }

    private ReplyKeyboardMarkup createDayMenu() {
        ReplyKeyboardMarkup keyboard = new ReplyKeyboardMarkup();
        List<KeyboardRow> rows = new ArrayList<>();

        KeyboardRow r1 = new KeyboardRow();
        r1.add("Понедельник");
        r1.add("Вторник");
        r1.add("Среда");
        rows.add(r1);

        KeyboardRow r2 = new KeyboardRow();
        r2.add("Четверг");
        r2.add("Пятница");
        r2.add("Суббота");
        rows.add(r2);

        KeyboardRow r3 = new KeyboardRow();
        r3.add("Назад");
        rows.add(r3);

        keyboard.setKeyboard(rows);
        keyboard.setResizeKeyboard(true);
```

```
        return keyboard;
    }
}
```

TimetableBotApplication.java

```
package ru.etu.timetable_bot;


import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.ConfigurableApplicationContext;
import org.telegram.telegrambots.meta.TelegramBotsApi;
import org.telegram.telegrambots.meta.exceptions.TelegramApiException;
import org.telegram.telegrambots.updatesreceivers.DefaultBotSession;
import ru.etu.timetable_bot.bot.TimetableBot;

@SpringBootApplication
public class TimetableBotApplication {

    public static void main(String[] args) {
        ConfigurableApplicationContext ctx = SpringApplication.run(TimetableBotApplication.class,
args);

        TimetableBot bot = ctx.getBean(TimetableBot.class);
        try {
            TelegramBotsApi botsApi = new TelegramBotsApi(DefaultBotSession.class);
            botsApi.registerBot(bot);
            System.out.println("Бот работает");
        } catch (TelegramApiException e) {
            e.printStackTrace();
        }
    }
}
```

Lessons.java

```
package ru.etu.timetable_bot.model;

import java.time.LocalTime;

public class Lessons {
    public String subject;
    public String type;
    public String teacher;
    public String room;
    public String week;
    public LocalTime startTime;
    public LocalTime endTime;
```

```
    @Override
    public String toString() {
        return String.format(
                "%s–%s %s (%s), %s, %s",
                startTime, endTime, subject, type, teacher, room
        );
    }
}
```

DateUtils.java

```
package ru.etu.timetable_bot.utils;

import java.time.DayOfWeek;
import java.time.LocalDate;

public class DateUtils {

    public static boolean isEvenWeek(LocalDate date) {
        LocalDate start = LocalDate.of(2025, 9, 1);
        if (date.isBefore(start)) {
            start = LocalDate.of(2024, 9, 2);
        }

        LocalDate firstMonday =
start.with(java.time.temporal.TemporalAdjusters.previousOrSame(DayOfWeek.MONDAY));
        long days = java.time.temporal.ChronoUnit.DAYS.between(firstMonday, date);
        int weekNumber = (int) (days / 7) + 1;
        return weekNumber % 2 == 0;
    }

    public static LocalDate findNextDay(DayOfWeek target, boolean evenWeek) {
        LocalDate today = LocalDate.now();
        for (int i = 0; i < 14; i++) {
            LocalDate candidate = today.plusDays(i);
            if (candidate.getDayOfWeek() == target && isEvenWeek(candidate) == evenWeek) {
                return candidate;
            }
        }
        return today.with(java.time.temporal.TemporalAdjusters.nextOrSame(target));
    }
}
```

DayTimetable.java

```
package ru.etu.timetable_bot.model;

import java.time.LocalDate;
import java.util.List;

public class DayTimetable {
```

```
    public LocalDate date;
    public List<Lessons> lessons;

    public boolean isEmpty() {
        return lessons == null || lessons.isEmpty();
    }
}
```

TimetableAPIservice.java

```
package ru.etu.timetable_bot.api;

import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import okhttp3.OkHttpClient;
import okhttp3.Request;
import okhttp3.Response;
import org.springframework.stereotype.Service;
import ru.etu.timetable_bot.model.DayTimetable;
import ru.etu.timetable_bot.model.Lessons;

import java.time.DayOfWeek;
import java.time.LocalDate;
import java.time.LocalTime;
import java.util.*;

@Service
public class TimetableAPIservice {

    private final OkHttpClient httpClient = new OkHttpClient();
    private final ObjectMapper objectMapper = new ObjectMapper();

    public List<DayTimetable> getScheduleForGroup(String groupNumber) throws Exception {
        String url = String.format(

"https://digital.etu.ru/api/mobile/schedule?groupNumber=%s&season=autumn&year=2025&joinW
eeks=true&withURL=true",
                groupNumber
        );

        Request request = new Request.Builder()
                .url(url)
                .addHeader("User-Agent", "Spring Boot Telegram Bot (ru.etu)")
                .build();

        try (Response response = httpClient.newCall(request).execute()) {
            if (!response.isSuccessful()) {
                throw new RuntimeException("Ошибка API: " + response.code());
            }
            String json = response.body().string();
            return parseSchedule(json, groupNumber);
        }
```

```
    }

    private List<DayTimetable> parseSchedule(String json, String groupNumber) throws Exception
{
        JsonNode root = objectMapper.readTree(json);
        JsonNode groupNode = root.get(groupNumber);

        if (groupNode == null || !groupNode.has("days")) {
            return Collections.emptyList();
        }

        JsonNode days = groupNode.get("days");
        List<DayTimetable> result = new ArrayList<>();
        LocalDate monday = LocalDate.now()
                .with(java.time.temporal.TemporalAdjusters.previousOrSame(DayOfWeek.MONDAY));

        for (int i = 0; i <= 5; i++) {
            String dayKey = String.valueOf(i);
            if (!days.has(dayKey)) continue;

            JsonNode day = days.get(dayKey);
            DayTimetable ds = new DayTimetable();
            ds.date = monday.plusDays(i);
            ds.lessons = new ArrayList<>();

            if (day.has("lessons")) {
                for (JsonNode lessonNode : day.get("lessons")) {
                    Lessons lesson = new Lessons();
                    lesson.subject = lessonNode.get("name").asText();
                    lesson.type = lessonNode.get("subjectType").asText();
                    lesson.startTime = LocalTime.parse(lessonNode.get("start_time").asText());
                    lesson.endTime = LocalTime.parse(lessonNode.get("end_time").asText());

                    String main = lessonNode.get("teacher").asText();
                    String second = lessonNode.path("second_teacher").asText();
                    lesson.teacher = second.isEmpty() ? main : main + ", " + second;

                    String form = lessonNode.get("form").asText();
                    if ("online".equals(form)) {
                        lesson.room = "онлайн";
                    } else {
                        lesson.room = lessonNode.path("room").asText();
                        if (lesson.room.isEmpty()) lesson.room = "—";
                    }

                    lesson.week = lessonNode.get("week").asText();

                    ds.lessons.add(lesson);
                }
                ds.lessons.sort(Comparator.comparing(l -> l.startTime));
            }
            result.add(ds);
```

```
        }
        return result;
    }

    public JsonNode getRawSchedule(String groupNumber) throws Exception {
        String url = String.format(

"https://digital.etu.ru/api/mobile/schedule?groupNumber=%s&season=autumn&year=2025&joinWeeks=true&withURL=true",
                groupNumber
        );

        Request request = new Request.Builder()
                .url(url)
                .addHeader("User-Agent", "Telegram Bot")
                .build();

        try (Response response = httpClient.newCall(request).execute()) {
            if (!response.isSuccessful()) {
                throw new RuntimeException("Ошибка API: " + response.code());
            }
            String json = response.body().string();
            return objectMapper.readTree(json);
        }
    }
}
```

Pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
<parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>4.0.0</version>
        <relativePath/>
</parent>
<groupId>ru.etu</groupId>
<artifactId>timetable-bot</artifactId>
<version>0.0.1-SNAPSHOT</version>
<name>timetable-bot</name>
<description>Demo project for Spring Boot</description>
<url/>
<licenses>
        <license/>
</licenses>
<developers>
        <developer/>
```

```
</developers>
<scm>
        <connection/>
        <developerConnection/>
        <tag/>
        <url/>
</scm>
<properties>
        <java.version>17</java.version>
</properties>
<dependencies>
      <dependency>
        <groupId>com.fasterxml.jackson.core</groupId>
        <artifactId>jackson-databind</artifactId>
        <version>2.17.0</version>
      </dependency>
      <dependency>
        <groupId>com.fasterxml.jackson.datatype</groupId>
        <artifactId>jackson-datatype-jsr310</artifactId>
        <version>2.17.0</version>
      </dependency>

      <dependency>
        <groupId>org.telegram</groupId>
        <artifactId>telegrambots-spring-boot-starter</artifactId>
        <version>5.4.0.1</version>
      </dependency>

        <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-webmvc</artifactId>
        </dependency>

        <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-webmvc-test</artifactId>
                <scope>test</scope>
        </dependency>

      <dependency>
        <groupId>com.squareup.okhttp3</groupId>
        <artifactId>okhttp</artifactId>
        <version>4.11.0</version>
      </dependency>
</dependencies>

<build>
        <plugins>
                <plugin>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-maven-plugin</artifactId>
                </plugin>
```

```
        </plugins>
</build>

</project>
```