

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

КУРСОВАЯ РАБОТА
по дисциплине «Объектно-ориентированное программирование»
Тема: «Создание Telegram-бота
с расписанием занятий ЛЭТИ»

Студенты гр. 4354

Преподаватель

Чучалин И.В.
Шупряков В.В.
Кулагин М.В.

Санкт-Петербург

2025

ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ

Студенты Шупряков В.В. и Чучалин И.В.

Группа 4354

Тема работы: создание Telegram-бота для студентов ЛЭТИ с расписанием занятий

Исходные данные:

Программа написана на языке Java 25. Среда разработки IntelliJ IDEA Community Edition 2025.2.1

Содержание пояснительной записки:

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

Предполагаемый объем пояснительной записки:

Не менее 20 страниц.

Дата выдачи задания: 30.09.2025

Дата сдачи реферата: 23.12.2025

Дата защиты реферата: 29.12.2025

Студент

Шупряков В.В.

Чучалин И.В.

Преподаватель

Кулагин М.В.

АННОТАЦИЯ

В рамках курсового проекта был разработан Telegram-бот для предоставления актуального расписания занятий Санкт-Петербургского государственного электротехнического университета «ЛЭТИ». Бот получает данные напрямую с официального сайта вуза через API в формате JSON, что обеспечивает работу с информацией в реальном времени. Основная функция — отправка расписания по номеру учебной группы по запросу пользователя. Реализованы возможности просмотра занятий на текущий день, на всю неделю, а также поиск ближайшего предстоящего занятия. Для удобства восприятия в начале каждого сообщения автоматически указывается четность учебной недели.

Разработка выполнена на языке Java с применением библиотеки telegrams API для интеграции с платформой Telegram. Бот оснащен интуитивным кнопочным интерфейсом, поддерживает автоматическое определение текущей недели и сохраняет пользовательские настройки. В системе предусмотрена корректная обработка ошибок, включая случаи потери подключения к сети или проблем с получением данных от сервера университета.

SUMMARY

Within the course project, a Telegram bot was developed to provide up-to-date class schedules for the Saint Petersburg State Electrotechnical University "LETI". The bot retrieves data directly from the official university website via an API in JSON format, ensuring real-time information processing. Its core function is to send the schedule based on a user-provided study group number. Implemented features include viewing classes for the current day, for the entire week, and finding the nearest

upcoming class. For user convenience, the parity (even/odd) of the academic week is automatically indicated at the beginning of each message.

The development was carried out using the Java programming language with the telegrambots API library for integration with the Telegram platform. The bot features an intuitive button-based interface, supports automatic determination of the current week, and saves user preferences. The system includes proper error handling for cases such as network connection loss or issues with retrieving data from the university server.

СОДЕРЖАНИЕ

Введение	6
1. Диаграмма вариантов использования	7
2. Диаграмма классов объектной модели предметной области	8
3. Спецификация классов	8
4. Описание интерфейса пользователя программы	12
5. Код классов объектной модели	15
Заключение	32
Список использованных источников	33

ВВЕДЕНИЕ

Целью данной курсовой работы является разработка Telegram-бота для получения актуального расписания занятий в университете «ЛЭТИ». Бот предназначен для упрощения доступа студентов к информации о расписании.

К основным задачам проекта относится разработка трёх ключевых компонентов: архитектуры, функционала и пользовательского интерфейса бота. В части архитектуры необходимо решить задачи по организации безопасного хранения конфиденциальных данных, настройке HTTP-запросов к официальному API университета, парсингу и обработке получаемых JSON-ответов, а также надёжной обработке различных форматов данных с корректным управлением исключениями. Функциональная часть включает реализацию отображения расписания на текущий и следующий день, а также на всю неделю, точное определение чётности учебной недели и функцию поиска ближайшего предстоящего занятия. В рамках создания интерфейса требуется разработать интуитивное кнопочное меню с возможностью ввода номера учебной группы.

Для решения этих задач будут применяться следующие методы: объектно-ориентированное программирование на Java для построения модульной и масштабируемой архитектуры, использование библиотеки TelegramBots для интеграции с Telegram API, применение HTTP-клиента для выполнения запросов к API ЛЭТИ и получения актуальных данных в формате JSON, а также библиотеки Gson для эффективного парсинга и сериализации данных.

1. Диаграмма вариантов использования.

Диаграмма вариантов использования включает: просмотр расписания на сегодня, просмотр расписания на завтра, просмотр ближайшего занятия, просмотр расписания на неделю, настройку группы с возможностью ручного ввода или выбора из предложенных вариантов, просмотр меню с командами бота, просмотр расписания по дням недели.

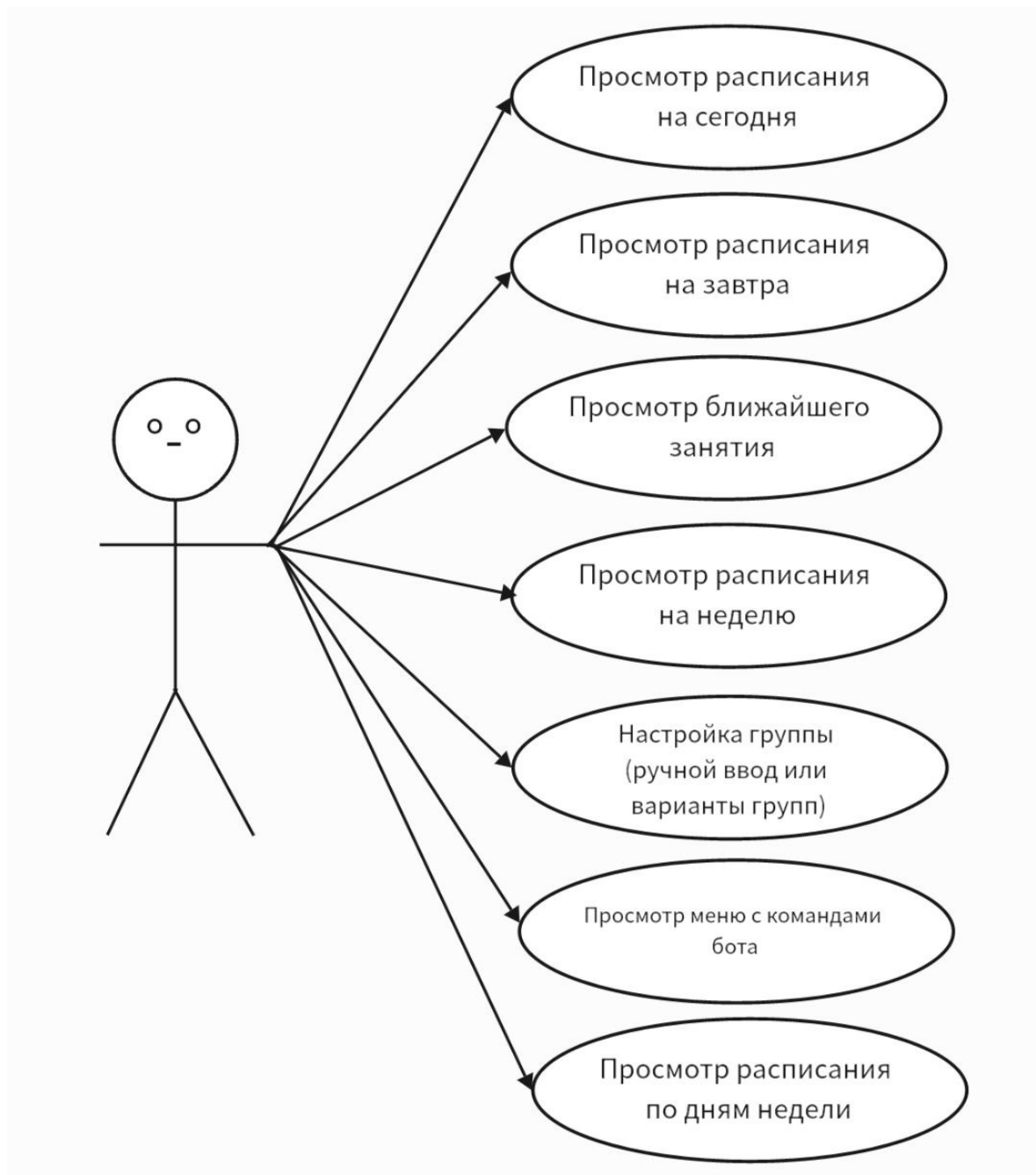


Рис. 1 Диаграмма вариантов использования системы

2. Диаграмма классов объектной модели предметной области.



Рис. 2. UML-диаграммы классов

3. Спецификация классов.

Класс ScheduleBot - Основной класс бота, обрабатывает взаимодействие с Telegram

Метод	Описание
String getBotUsername()	Возвращает имя бота из конфигурации
String getBotToken()	Возвращает токен бота из конфигурации
void onUpdateReceived(Update update)	Главный метод обработки входящих сообщений
void handleMessage(Update update)	Обрабатывает текстовые сообщения от пользователя

<code>void handleToday(long chatId, long userId)</code>	Отправляет расписание на текущий день
<code>void handleTomorrow(long chatId, long userId)</code>	Отправляет расписание на следующий день
<code>void handleWeek(long chatId, long userId)</code>	Отправляет расписание на всю неделю
<code>void handleNear(long chatId, long userId)</code>	Находит и отправляет ближайшее занятие
<code>void handleDay(long chatId, long userId, String day)</code>	Отправляет расписание на указанный день недели
<code>void handleSettings(long chatId, long userId)</code>	Показывает меню настройки группы
<code>void handleSetGroup(long chatId, long userId, String groupNumber)</code>	Устанавливает номер группы пользователя
<code>void sendMessageWithKeyboard(long chatId, String text, ReplyKeyboardMarkup keyboard)</code>	Отправляет сообщение с клавиатурой

Класс BotConfig - Класс для управления конфигурацией и данными пользователей

Поле	Описание
<code>static Properties properties</code>	Хранит настройки из config.properties

<code>static Map<Long, String> userGroups</code>	Хранит соответствие ID пользователей и их групп
--	---

Методы	Описание
<code>static String getBotUsername()</code>	Возвращает имя бота
<code>static String getBotToken()</code>	Возвращает токен бота
<code>static void setUserGroup(long userId, String groupNumber)</code>	Сохраняет номер группы для пользователя
<code>static String getUserGroup(long userId)</code>	Возвращает номер группы пользователя
<code>static void loadConfig()</code>	Загружает конфигурацию из файла

Класс `ScheduleService` - Класс для работы с API расписания ЛЭТИ

Методы	Описание
<code>static String getScheduleForGroup(String groupId)</code>	Получает JSON-расписание для указанной группы
<code>static String parseScheduleForDay(String json, String day, String groupNumber)</code>	Парсит расписание на конкретный день
<code>static String getWeekSchedule(String json, String groupNumber)</code>	Формирует расписание на всю неделю

<code>static String findNearestLesson(String json, String groupNumber)</code>	Находит ближайшее занятие для группы
<code>static String getCurrentWeekParity()</code>	Определяет четность текущей недели
<code>static String getCurrentWeekInfo()</code>	Возвращает информацию о четности недели
<code>static String getTodayDayName()</code>	Возвращает английское название текущего дня
<code>static String getTomorrowDayName()</code>	Возвращает английское название завтрашнего дня
<code>static String fetchJsonFromUrl(String url)</code>	Выполняет HTTP-запрос и получает JSON
<code>static String formatNearestLesson(JsonObject lesson, String day)</code>	Форматирует информацию о ближайшем занятии
<code>static boolean isLessonForCurrentWeek(String weekType)</code>	Проверяет, относится ли занятие к текущей неделе
<code>static String getSafeString(JsonObject obj, String key, String defaultValue)</code>	Получает строку из JSON

Класс `ScheduleException` - Пользовательское исключение для ошибок расписания

Конструктор	Описание
<code>ScheduleException(String message)</code>	Создает исключение с сообщением об ошибке
<code>ScheduleException(String message, Throwable cause)</code>	Создает исключение с сообщением и причиной

Класс `KeyboardService` - Класс для создания клавиатур интерфейса

Методы	Описание
<code>static ReplyKeyboardMarkup getMainKeyboard()</code>	Создает главное меню бота
<code>static ReplyKeyboardMarkup getGroupSetupKeyboard()</code>	Создает клавиатуру для настройки группы
<code>static ReplyKeyboardMarkup getBackKeyboard()</code>	Создает клавиатуру с кнопкой "Назад"

Класс `DevmarkscheduleBot` - Главный класс приложения, точка входа

Метод	Описание
<code>static void main(String[] args)</code>	Запускает Telegram-бота, проверяет конфигурацию

4. Описание интерфейса пользователя программы.

1)Общая концепция интерфейса

Интерфейс реализован в виде меню на основе кнопок, что обеспечивает интуитивную навигацию без необходимости запоминания команд. Бот следует принципам контекстного взаимодействия — в зависимости от состояния диалога пользователю предлагаются соответствующие клавиатуры.

2)Стартовый экран

При вызове команд /start, /help или нажатии кнопки «Назад в меню» пользователь видит приветственное сообщение с названием бота. Если группа уже установлена, отображается информация о текущей группе; в противном случае предлагается ввести номер группы. Сообщение сопровождается главным меню кнопок для дальнейшей навигации.

Клавиатура представляет собой главное меню с пятью рядами кнопок: первый ряд содержит «Сегодня» и «Завтра», второй — «Ближайшее» и «Неделя», третий — «Понедельник», «Вторник» и «Среда», четвёртый — «Четверг», «Пятница» и «Суббота», пятый — «Настройка группы».

3)Меню настройки группы

При нажатии кнопки «Настройка группы» пользователь переходит в специализированное меню выбора группы. Отображается информация о текущей установленной группе (если есть) и предлагаются варианты: четыре предустановленные группы 4351–4354, кнопка «Ввести свою группу» для ручного ввода и кнопка «Назад в меню» для возврата без изменений.

Выбор предустановленной группы приводит к её автоматическому сохранению и возврату в главное меню с подтверждающим сообщением. При выборе «Ввести свою группу» активируется режим ручного ввода.

4)Режим ввода группы

В режиме ручного ввода пользователю предлагается ввести номер группы в текстовом поле. Клавиатура сокращается до одной кнопки «Назад в меню». Система валидирует ввод: при корректном четырёхзначном номере группа сохраняется, пользователь возвращается в главное меню с подтверждением; при некорректном вводе выводится сообщение об ошибке с предложением повторить ввод.

5)Формат вывода расписания

Сообщения с расписанием имеют единую структурированную форму. В начале указывается текущая чётность недели, затем название дня. Каждое занятие форматируется с указанием времени начала и окончания, типа недели (если применимо), названия предмета, типа занятия, имени преподавателя (при наличии) и места проведения — номера аудитории или пометки «Онлайн» для дистанционных форматов.

6)Обработка ошибок и уведомления

При отсутствии установленной группы для запросов расписания выводится сообщение «Сначала установите группу». Если занятий в выбранный период нет, отображается соответствующее уведомление. Ошибки ввода группы сопровождаются пояснением о требуемом формате. Все сообщения об ошибках сохраняют доступ к основной навигации через прикрепленные клавиатуры

5. Код классов объектной модели.

```
//DevmarkscheduleBote
package leti.schedule.bot;

import org.telegram.telegrambots.meta.TelegramBotsApi;
import org.telegram.telegrambots.meta.exceptions.TelegramApiException;
import org.telegram.telegrambots.updatesreceivers.DefaultBotSession;

public class DevmarkschebuleBot {
    public static void main(String[] args) {
        String token = BotConfig.getBotToken();
        if (token == null || token.contains("${}") || token.equals("telegram.bot.token")) {
            System.err.println("Токен не найден, ошибка");
            System.exit(1);
        }

        try {
            TelegramBotsApi botsApi = new TelegramBotsApi(DefaultBotSession.class);
            botsApi.registerBot(new ScheduleBot());
            System.out.println("Бот запущен! Имя: " + BotConfig.getBotUsername());
        } catch (TelegramApiException e) {
            e.printStackTrace();
            System.err.println("Ошибка при запуске бота: " + e.getMessage());
        }
    }
}

//BotConfig
package leti.schedule.bot;

import java.io.*;
import java.util.HashMap;
import java.util.Map;
import java.util.Properties;

public class BotConfig {
    private static final Properties properties = new Properties();
    private static final Map<Long, String> userGroups = new HashMap<>();

    static {
        loadConfig();
    }

    private static void loadConfig() {
        try (InputStream input =
            BotConfig.class.getClassLoader().getResourceAsStream("Application.properties")) {
            if (input == null) {
                throw new RuntimeException("Файл Application.properties не найден");
            }
        }
    }
}
```

```

    }
    properties.load(input);

    String botToken = System.getenv("BOT_TOKEN");
    if (botToken != null && !botToken.trim().isEmpty()) {
        properties.setProperty("telegram.bot.token", botToken);
    }

} catch (IOException e) {
    throw new RuntimeException("Ошибка чтения конфигурации", e);
}
}

public static String getBotUsername() {
    return properties.getProperty("telegram.bot.username");
}

public static String getBotToken() {
    return properties.getProperty("telegram.bot.token");
}

public static void setUserGroup(long userId, String groupNumber) {
    if (groupNumber.matches("\\d{4}")) {
        userGroups.put(userId, groupNumber);
    }
}

public static String getUserGroup(long userId) {
    return userGroups.get(userId);
}
}

//KeyboardService
package leti.schedule.bot;

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;

public class KeyboardService {

    public static ReplyKeyboardMarkup getMainKeyboard() {
        ReplyKeyboardMarkup keyboardMarkup = new ReplyKeyboardMarkup();
        keyboardMarkup.setResizeKeyboard(true);

        List<KeyboardRow> keyboard = new ArrayList<>();

        KeyboardRow row1 = new KeyboardRow();
        row1.add("Сегодня");
        row1.add("Завтра");
    }
}

```

```

keyboard.add(row1);

KeyboardRow row2 = new KeyboardRow();
row2.add("Ближайшее");
row2.add("Неделя");
keyboard.add(row2);

KeyboardRow row3 = new KeyboardRow();
row3.add("Понедельник");
row3.add("Вторник");
row3.add("Среда");
keyboard.add(row3);

KeyboardRow row4 = new KeyboardRow();
row4.add("Четверг");
row4.add("Пятница");
row4.add("Суббота");
keyboard.add(row4);

KeyboardRow row5 = new KeyboardRow();
row5.add("Настройка группы");
keyboard.add(row5);

keyboardMarkup.setKeyboard(keyboard);
return keyboardMarkup;
}

public static ReplyKeyboardMarkup getGroupSetupKeyboard() {
    ReplyKeyboardMarkup keyboardMarkup = new ReplyKeyboardMarkup();
    keyboardMarkup.setSelective(true);
    keyboardMarkup.setResizeKeyboard(true);
    keyboardMarkup.setOneTimeKeyboard(true);

    List<KeyboardRow> keyboard = new ArrayList<>();

    KeyboardRow row1 = new KeyboardRow();
    row1.add("4351");
    row1.add("4352");
    row1.add("4353");
    row1.add("4354");
    keyboard.add(row1);

    KeyboardRow row2 = new KeyboardRow();
    row2.add("Ввести свою группу");
    row2.add("Назад в меню");
    keyboard.add(row2);

    keyboardMarkup.setKeyboard(keyboard);
    return keyboardMarkup;
}

public static ReplyKeyboardMarkup getBackKeyboard() {

```

```

        ReplyKeyboardMarkup keyboardMarkup = new ReplyKeyboardMarkup();
        keyboardMarkup.setSelective(true);
        keyboardMarkup.setResizeKeyboard(true);
        keyboardMarkup.setOneTimeKeyboard(false);

        List<KeyboardRow> keyboard = new ArrayList<>();
        KeyboardRow row = new KeyboardRow();
        row.add("Назад в меню");
        keyboard.add(row);

        keyboardMarkup.setKeyboard(keyboard);
        return keyboardMarkup;
    }
}

//ScheduleBot
package leti.schedule.bot;

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.api.objects.replykeyboard.ReplyKeyboardMarkup;
import org.telegram.telegrambots.meta.exceptions.TelegramApiException;

import java.io.IOException;

public class ScheduleBot extends TelegramLongPollingBot {

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

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

    @Override
    public void onUpdateReceived(Update update) {
        try {
            if (update.hasMessage() && update.getMessage().hasText()) {
                handleMessage(update);
            }
        } catch (TelegramApiException e) {
            System.err.println("Telegram API: " + e.getMessage());
        }
    }

    private void handleMessage(Update update) throws TelegramApiException {
        String messageText = update.getMessage().getText().trim();
        long chatId = update.getMessage().getChatId();
    }
}

```

```

long userId = update.getMessage().getFrom().getId();

try {
    if (messageText.equals("/start") || messageText.equals("/help") ||
messageText.equals("Назад в меню")) {
        sendWelcomeMessage(chatId, userId);
    } else if (messageText.equals("Сегодня")) {
        handleToday(chatId, userId);
    } else if (messageText.equals("Завтра")) {
        handleTomorrow(chatId, userId);
    } else if (messageText.equals("Неделя")) {
        handleWeek(chatId, userId);
    } else if (messageText.equals("Ближайшее")) {
        handleNear(chatId, userId);
    } else if (messageText.equals("Настройка группы")) {
        handleSettings(chatId, userId);
    } else if (messageText.equals("Понедельник")) {
        handleDay(chatId, userId, "monday");
    } else if (messageText.equals("Вторник")) {
        handleDay(chatId, userId, "tuesday");
    } else if (messageText.equals("Среда")) {
        handleDay(chatId, userId, "wednesday");
    } else if (messageText.equals("Четверг")) {
        handleDay(chatId, userId, "thursday");
    } else if (messageText.equals("Пятница")) {
        handleDay(chatId, userId, "friday");
    } else if (messageText.equals("Суббота")) {
        handleDay(chatId, userId, "saturday");
    } else if (messageText.equals("Ввести свою группу")) {
        requestGroupInput(chatId);
    } else if (messageText.matches("\\d{2,5}")) {
// любое число от 2 до 5 цифр
if (messageText.matches("\\d{4}")) {
    handleSetGroup(chatId, userId, messageText);
} else {
    sendMessageWithKeyboard(chatId,
        "Некорректный номер группы.\nНомер группы должен состоять из 4 цифр.\n\n"
+
        "Пожалуйста, введите правильный номер группы:",
        KeyboardService.getBackKeyboard());
}
    } else {
        sendMessageWithKeyboard(chatId, "Используйте кнопки меню",
KeyboardService.getMainKeyboard());
    }
//    } else if (messageText.matches("\\d{4}")) {
//        handleSetGroup(chatId, userId, messageText);
//    } else {
//        sendMessageWithKeyboard(chatId, "Используйте кнопки меню",
KeyboardManager.getMainKeyboard());
//    }
} catch (ScheduleException e) {

```

```

        sendMessageWithKeyboard(chatId, "Ошибка: " + e.getMessage(),
KeyboardService.getMainKeyboard());
    } catch (IOException e) {
        sendMessageWithKeyboard(chatId, "Ошибка подключения к серверу",
KeyboardService.getMainKeyboard());
    } catch (TelegramApiException e) {
        throw e;
    }
}

private void sendWelcomeMessage(long chatId, long userId) throws TelegramApiException {
    String userGroup = BotConfig.getUserGroup(userId);

    if (userGroup != null) {
        String welcomeText = "*Бот расписания ЛЭТИ*\n\n" +
            "*Ваша группа:* " + userGroup + "\n\n" +
            "Используйте кнопки ниже для получения расписания";
        sendMessageWithKeyboard(chatId, welcomeText, KeyboardService.getMainKeyboard());
    } else {
        String welcomeText = "*Бот расписания ЛЭТИ*\n\n" +
            "Добро пожаловать! Этот бот поможет вам получить расписание занятий:\n\n" +
            "*Введите номер своей группы* (4 цифры, например: 4354)\n\n" +
            "Или выберите группу из списка в меню 'Настройка группы';
        sendMessageWithKeyboard(chatId, welcomeText, KeyboardService.getMainKeyboard());
    }
}

private void handleToday(long chatId, long userId) throws ScheduleException, IOException,
TelegramApiException {
    String userGroup = BotConfig.getUserGroup(userId);
    if (userGroup == null) {
        sendMessageWithKeyboard(chatId, "Сначала установите группу",
KeyboardService.getMainKeyboard());
        return;
    }

    String json = ScheduleService.getScheduleForGroup(userGroup);
    String today = ScheduleService.getTodayDayName();
    String schedule = ScheduleService.parseScheduleForDay(json, today, userGroup);
    sendMessageWithKeyboard(chatId, schedule, KeyboardService.getMainKeyboard());
}

private void handleTomorrow(long chatId, long userId) throws ScheduleException, IOException,
TelegramApiException {
    String userGroup = BotConfig.getUserGroup(userId);
    if (userGroup == null) {
        sendMessageWithKeyboard(chatId, "Сначала установите группу",
KeyboardService.getMainKeyboard());
        return;
    }
}

```

```

        String json = ScheduleService.getScheduleForGroup(userGroup);
        String tomorrow = ScheduleService.getTomorrowDayName();
        String schedule = ScheduleService.parseScheduleForDay(json, tomorrow, userGroup);
        sendMessageWithKeyboard(chatId, schedule, KeyboardService.getMainKeyboard());
    }

    private void handleWeek(long chatId, long userId) throws ScheduleException, IOException,
TelegramApiException {
        String userGroup = BotConfig.getUserGroup(userId);
        if (userGroup == null) {
            sendMessageWithKeyboard(chatId, "Сначала установите группу",
KeyboardService.getMainKeyboard());
            return;
        }

        String json = ScheduleService.getScheduleForGroup(userGroup);
        String schedule = ScheduleService.getWeekSchedule(json, userGroup);
        sendMessageWithKeyboard(chatId, schedule, KeyboardService.getMainKeyboard());
    }

    private void handleNear(long chatId, long userId) throws ScheduleException, IOException,
TelegramApiException {
        String userGroup = BotConfig.getUserGroup(userId);
        if (userGroup == null) {
            sendMessageWithKeyboard(chatId, "Сначала установите группу",
KeyboardService.getMainKeyboard());
            return;
        }

        String json = ScheduleService.getScheduleForGroup(userGroup);
        String nearest = ScheduleService.findNearestLesson(json, userGroup);
        sendMessageWithKeyboard(chatId, nearest, KeyboardService.getMainKeyboard());
    }

    private void handleDay(long chatId, long userId, String day) throws ScheduleException,
IOException, TelegramApiException {
        String userGroup = BotConfig.getUserGroup(userId);
        if (userGroup == null) {
            sendMessageWithKeyboard(chatId, "Сначала установите группу",
KeyboardService.getMainKeyboard());
            return;
        }

        String json = ScheduleService.getScheduleForGroup(userGroup);
        String schedule = ScheduleService.parseScheduleForDay(json, day, userGroup);
        sendMessageWithKeyboard(chatId, schedule, KeyboardService.getMainKeyboard());
    }

    private void handleSettings(long chatId, long userId) throws TelegramApiException {
        String userGroup = BotConfig.getUserGroup(userId);
        String currentGroupInfo = userGroup != null ? "Текущая группа: *" + userGroup + " *\n\n" :
"";

```

```

        String settingsText = "*Настройка группы*\n\n" + currentGroupInfo +
            "Выберите группу или введите свою";
        sendMessageWithKeyboard(chatId, settingsText,
            KeyboardService.getGroupSetupKeyboard());
    }

    private void handleSetGroup(long chatId, long userId, String groupNumber) throws
        TelegramApiException {
        if (groupNumber.matches("\\d{4}")) {
            BotConfig.setUserGroup(userId, groupNumber);
            sendMessageWithKeyboard(chatId, "Группа установлена: *" + groupNumber + "*",
                KeyboardService.getMainKeyboard());
        } else {
            sendMessageWithKeyboard(chatId,
                "Некорректный номер группы.\nНомер группы должен состоять из 4 цифр.\n\n" +
                "Пожалуйста, введите номер группы еще раз:",
                KeyboardService.getBackKeyboard());
        }
    }

    private void requestGroupInput(long chatId) throws TelegramApiException {
        SendMessage message = new SendMessage();
        message.setChatId(String.valueOf(chatId));
        message.setText("Введите номер группы (4 цифры):");
        execute(message);
    }

    private void sendMessageWithKeyboard(long chatId, String text, ReplyKeyboardMarkup
        keyboard) throws TelegramApiException {
        SendMessage message = new SendMessage();
        message.setChatId(String.valueOf(chatId));
        message.setText(text);
        message.setParseMode("Markdown");
        message.setReplyMarkup(keyboard);
        execute(message);
    }
}

//ScheduleService
package leti.schedule.bot;

import com.google.gson.*;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.util.EntityUtils;
import org.apache.http.HttpResponse;

import java.io.IOException;
import java.time.*;
import java.time.temporal.ChronoUnit;

```

```

import java.util.*;

import java.time.DateTimeException;
import java.time.format.TextStyle;

public class ScheduleService {

    public static String getScheduleForGroup(String groupId) throws IOException,
ScheduleException {
        String apiUrl = String.format(

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

        return fetchJsonFromUrl(apiUrl);
    }

    private static String fetchJsonFromUrl(String url) throws IOException, ScheduleException {
        try (CloseableHttpClient httpClient = HttpClients.createDefault()) {
            HttpGet request = new HttpGet(url);
            HttpResponse response = httpClient.execute(request);

            if (response.getStatusLine().getStatusCode() != 200) {
                throw new ScheduleException("API вернуло ошибку: " +
response.getStatusLine().getStatusCode());
            }

            return EntityUtils.toString(response.getEntity(), "UTF-8");
        }
    }

    public static String getCurrentWeekParity() {
        LocalDate semesterStartDate = LocalDate.of(2025, 9, 2);
        long weeksSinceStart = ChronoUnit.WEEKS.between(semesterStartDate, LocalDate.now());
        boolean isOddWeek = weeksSinceStart % 2 == 0;
        return isOddWeek ? "нечётная" : "чётная";
    }

    public static String getCurrentWeekInfo() {
        return "*Текущая неделя:* " + getCurrentWeekParity();
    }

    public static String parseScheduleForDay(String jsonData, String dayName, String
groupNumber) throws ScheduleException {
        try {
            JsonObject rootObject = JsonParser.parseString(jsonData).getAsJsonObject();

            if (!rootObject.has(groupNumber)) {
                throw new ScheduleException("Группа не найдена");
            }
        }
    }
}

```

```

JsonObject groupData = rootObject.getAsJsonObject(groupNumber);
JsonObject daysSchedule = groupData.getAsJsonObject("days");
int dayIndex = getDayIndex(dayName);

if (dayIndex == -1) {
    throw new ScheduleException("Неверный день недели");
}

String dayKey = String.valueOf(dayIndex);
if (!daysSchedule.has(dayKey)) {
    return "В этот день занятий нет.";
}

JsonObject daySchedule = daysSchedule.getAsJsonObject(dayKey);
if (!daySchedule.has("lessons") || daySchedule.get("lessons").isJsonNull()) {
    return "В этот день занятий нет.";
}

JsonArray lessonsArray = daySchedule.getAsJsonArray("lessons");
if (lessonsArray.isEmpty()) {
    return "В этот день занятий нет.";
}

List<JsonObject> lessonsList = new ArrayList<>();
for (JsonElement lesson : lessonsArray) {
    lessonsList.add(lesson.getAsJsonObject());
}

lessonsList.sort((lesson1, lesson2) -> {
    String time1 = getSafeString(lesson1, "start_time", "00:00");
    String time2 = getSafeString(lesson2, "start_time", "00:00");
    return time1.compareTo(time2);
});

StringBuilder scheduleText = new StringBuilder();
String displayDayName = daySchedule.has("name") ?
    daySchedule.get("name").getString() : getRussianDayName(dayIndex);

scheduleText.append(getCurrentWeekInfo()).append("\n");
scheduleText.append("*").append(displayDayName).append("*\n\n");

for (JsonObject lesson : lessonsList) {
    String startTime = getSafeString(lesson, "start_time", "?:?:??");
    String endTime = getSafeString(lesson, "end_time", "?:?:??");
    String subjectName = getSafeString(lesson, "name", "Предмет не указан");
    String lessonType = getSafeString(lesson, "subjectType", "");
    String teacherName = getSafeString(lesson, "teacher", "");
    String classroom = getSafeString(lesson, "room", "");
    String lessonFormat = getSafeString(lesson, "form", "");
    String weekType = getSafeString(lesson, "week", "");
}

```

```

        String weekInfoText = "";
        if (!weekType.isEmpty() && !weekType.equals("null")) {
            weekInfoText = getWeekTypeInfo(weekType);
        }

        scheduleText.append("*").append(startTime).append(" - ").append(endTime);
        if (!weekInfoText.isEmpty()) {
            scheduleText.append(" ").append(weekInfoText);
        }
        scheduleText.append("*\n");

        scheduleText.append(" ").append(subjectName);
        if (!lessonType.isEmpty()) {
            scheduleText.append(" ").append(lessonType).append(" ");
        }
        scheduleText.append("\n");

        if (!teacherName.isEmpty() && !teacherName.equals("null")) {
            scheduleText.append(" ").append(teacherName).append("\n");
        }

        if ("online".equalsIgnoreCase(lessonFormat) || "distant".equalsIgnoreCase(lessonFormat))
        {
            scheduleText.append(" Онлайн");
        } else if (!classroom.isEmpty() && !classroom.equals("null")) {
            scheduleText.append(" Ауд. ").append(classroom);
        }

        scheduleText.append("\n\n");
    }

    return scheduleText.toString();

} catch (JsonSyntaxException e) {
    throw new ScheduleException("Ошибка формата данных");
}
}

public static String getWeekSchedule(String jsonData, String groupNumber) throws
ScheduleException {
    try {
        JsonObject rootObject = JsonParser.parseString(jsonData).getAsJsonObject();

        if (!rootObject.has(groupNumber)) {
            throw new ScheduleException("Группа не найдена");
        }

        JsonObject groupData = rootObject.getAsJsonObject(groupNumber);
        JsonObject daysSchedule = groupData.getAsJsonObject("days");

        StringBuilder weekScheduleText = new StringBuilder();
        weekScheduleText.append(getCurrentWeekInfo()).append("\n");
    }
}

```

```

        weekScheduleText.append("*Расписание для группы
").append(groupNumber).append("*\n\n");

        String[] russianDayNames = {"Понедельник", "Вторник", "Среда", "Четверг",
"Пятница", "Суббота"};

        boolean hasAnyLessons = false;

        for (int dayIdx = 0; dayIdx < 6; dayIdx++) {
            String dayKey = String.valueOf(dayIdx);
            if (daysSchedule.has(dayKey)) {
                JsonObject daySchedule = daysSchedule.getAsJsonObject(dayKey);
                if (daySchedule.has("lessons") && !daySchedule.get("lessons").isJsonNull()) {
                    JsonArray lessonsArray = daySchedule.getAsJsonArray("lessons");
                    if (!lessonsArray.isEmpty()) {
                        hasAnyLessons = true;

                        String displayDayName = daySchedule.has("name") ?
                            daySchedule.get("name").getString() : russianDayNames[dayIdx];
                        weekScheduleText.append("*").append(displayDayName).append(":\n");

                        for (JsonElement lesson : lessonsArray) {
                            JsonObject lessonObj = lesson.getAsJsonObject();
                            String startTime = getSafeString(lessonObj, "start_time", "?:?:?");
                            String endTime = getSafeString(lessonObj, "end_time", "?:?:?");
                            String subjectName = getSafeString(lessonObj, "name", "Предмет");
                            String lessonType = getSafeString(lessonObj, "subjectType", "");
                            String classroom = getSafeString(lessonObj, "room", "");

                            weekScheduleText.append(" • ").append(startTime).append("-
").append(endTime);
                            weekScheduleText.append(" - ").append(subjectName);

                            if (!lessonType.isEmpty()) {
                                weekScheduleText.append(" (").append(lessonType).append(")");
                            }

                            if (!classroom.isEmpty() && !classroom.equals("null")) {
                                weekScheduleText.append(" (").append(classroom).append(")");
                            }

                            weekScheduleText.append("\n");
                        }
                        weekScheduleText.append("\n");
                    }
                }
            }
        }

        if (!hasAnyLessons) {
            return "На этой неделе занятий нет.";
        }

```

```

        return weekScheduleText.toString();

    } catch (JsonSyntaxException e) {
        throw new ScheduleException("Ошибка формата данных");
    }
}

public static String findNearestLesson(String jsonData, String groupNumber) throws
ScheduleException {
    try {
        LocalDateTime currentTime = LocalDateTime.now();
        LocalDate currentDate = LocalDate.now();
        DayOfWeek currentDayOfWeek = currentDate.getDayOfWeek();

        JsonObject rootObject = JsonParser.parseString(jsonData).getAsJsonObject();

        if (!rootObject.has(groupNumber)) {
            throw new ScheduleException("Группа не найдена. Попробуйте изменить номер
группы.");
        }

        JsonObject groupData = rootObject.getAsJsonObject(groupNumber);
        JsonObject daysSchedule = groupData.getAsJsonObject("days");

        // Ищем на сегодня
        int todayIndex = currentDayOfWeek.getValue() - 1;
        String todayKey = String.valueOf(todayIndex);

        if (daysSchedule.has(todayKey)) {
            JsonObject todaySchedule = daysSchedule.getAsJsonObject(todayKey);
            if (todaySchedule.has("lessons") && !todaySchedule.get("lessons").isJsonNull()) {
                JsonArray lessonsArray = todaySchedule.getAsJsonArray("lessons");

                JsonObject nearestLesson = null;
                LocalDateTime nearestLessonTime = null;

                for (JsonElement lesson : lessonsArray) {
                    JsonObject lessonObj = lesson.getAsJsonObject();
                    String startTimeStr = getSafeString(lessonObj, "start_time", "");
                    if (!startTimeStr.isEmpty()) {
                        try {
                            LocalDateTime lessonTime = LocalDateTime.parse(startTimeStr);

                            String weekType = getSafeString(lessonObj, "week", "");
                            if (isLessonForCurrentWeek(weekType)) {
                                if (lessonTime.isAfter(currentTime) || lessonTime.equals(currentTime)) {
                                    if (nearestLessonTime == null || lessonTime.isBefore(nearestLessonTime))
                                    {
                                        nearestLessonTime = lessonTime;
                                        nearestLesson = lessonObj;
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}

```

```

        }
    }
    } catch (DateTimeException ignored) {
        // Пропускаем невалидное время и продолжаем цикл
    }
}

if (nearestLesson != null) {
    return formatNearestLesson(nearestLesson, "сегодня");
}

}

// Если на сегодня не нашли, ищем на ближайшие дни
for (int daysOffset = 1; daysOffset <= 7; daysOffset++) {
    int nextDayIndex = (todayIndex + daysOffset) % 7;
    String nextDayKey = String.valueOf(nextDayIndex);

    if (daysSchedule.has(nextDayKey)) {
        JsonObject daySchedule = daysSchedule.getAsJsonObject(nextDayKey);
        if (daySchedule.has("lessons") && !daySchedule.get("lessons").isJsonNull()) {
            JsonArray lessonsArray = daySchedule.getAsJsonArray("lessons");

            if (!lessonsArray.isEmpty()) {
                JsonObject firstLesson = lessonsArray.get(0).getAsJsonObject();

                String dayDisplayName;
                switch (daysOffset) {
                    case 1 -> dayDisplayName = "завтра";
                    case 2 -> dayDisplayName = "послезавтра";
                    default -> {
                        LocalDate targetDate = currentDate.plusDays(daysOffset);
                        String rawDayName =
targetDate.getDayOfWeek().getDisplayName(TextStyle.FULL, new Locale("ru"));
                        dayDisplayName = rawDayName.substring(0, 1).toUpperCase() +
rawDayName.substring(1);
                    }
                }

                return formatNearestLesson(firstLesson, dayDisplayName);
            }
        }
    }
}

return "Ближайших занятий не найдено";

} catch (JsonSyntaxException e) {
    throw new ScheduleException("Ошибка формата данных");
}
}

```

```

private static String formatNearestLesson(JsonObject lesson, String dayDisplayName) {
    String startTime = getSafeString(lesson, "start_time", "?:?:?");
    String endTime = getSafeString(lesson, "end_time", "?:?:?");
    String subjectName = getSafeString(lesson, "name", "Предмет не указан");
    String lessonType = getSafeString(lesson, "subjectType", "");
    String teacherName = getSafeString(lesson, "teacher", "");
    String classroom = getSafeString(lesson, "room", "");
    String lessonFormat = getSafeString(lesson, "form", "");
    String weekType = getSafeString(lesson, "week", "");

    StringBuilder resultText = new StringBuilder();
    resultText.append("*Ближайшее занятие*\n\n");
    resultText.append("*").append(dayDisplayName).append("*\n");
    resultText.append("*").append(startTime).append(" - ").append(endTime).append("*\n");
    resultText.append(" ").append(subjectName);

    if (!lessonType.isEmpty()) {
        resultText.append(" (").append(lessonType).append(")");
    }
    resultText.append("\n");

    if (!teacherName.isEmpty() && !teacherName.equals("null")) {
        resultText.append(" ").append(teacherName).append("\n");
    }

    if ("online".equalsIgnoreCase(lessonFormat) || "distant".equalsIgnoreCase(lessonFormat)) {
        resultText.append(" Онлайн");
    } else if (!classroom.isEmpty() && !classroom.equals("null")) {
        resultText.append(" Ауд. ").append(classroom);
    }

    if (!weekType.isEmpty() && !weekType.equals("null")) {
        resultText.append("\n ").append(getWeekTypeInfo(weekType));
    }

    return resultText.toString();
}

private static boolean isLessonForCurrentWeek(String weekType) {
    if (weekType.isEmpty() || weekType.equals("null") || weekType.equals("3")) {
        return true;
    }

    String currentParity = getCurrentWeekParity();
    boolean isCurrentOdd = currentParity.equals("нечётная");

    return (isCurrentOdd && weekType.equals("1")) || (!isCurrentOdd &&
weekType.equals("2"));
}

private static String getWeekTypeInfo(String weekType) {

```

```

return switch (weekType) {
    case "1" -> "(Нечётная неделя)";
    case "2" -> "(Чётная неделя)";
    case "3" -> "(Все недели)";
    default -> "";
};
}

private static int getDayIndex(String dayName) {
    return switch (dayName.toLowerCase()) {
        case "monday" -> 0;
        case "tuesday" -> 1;
        case "wednesday" -> 2;
        case "thursday" -> 3;
        case "friday" -> 4;
        case "saturday" -> 5;
        default -> -1;
    };
}

private static String getRussianDayName(int index) {
    String[] russianDays = {"Понедельник", "Вторник", "Среда", "Четверг", "Пятница",
"Суббота"};
    return (index >= 0 && index < russianDays.length) ? russianDays[index] : "День недели";
}

private static String getSafeString(JsonObject jsonObj, String key, String defaultValue) {
    if (jsonObj.has(key) && !jsonObj.get(key).isJsonNull()) {
        String value = jsonObj.get(key).getString();
        return (value == null || value.equals("null") || value.trim().isEmpty()) ?
            defaultValue : value.trim();
    }
    return defaultValue;
}

public static String getTomorrowDayName() {
    LocalDate tomorrow = LocalDate.now().plusDays(1);
    DayOfWeek day = tomorrow.getDayOfWeek();
    return translateDay(day.getDisplayName(TextStyle.FULL, new Locale("ru")).toLowerCase());
}

public static String getTodayDayName() {
    LocalDate today = LocalDate.now();
    DayOfWeek day = today.getDayOfWeek();
    return translateDay(day.getDisplayName(TextStyle.FULL, new Locale("ru")).toLowerCase());
}

private static String translateDay(String russianDayName) {
    return switch (russianDayName) {
        case "понедельник" -> "monday";
        case "вторник" -> "tuesday";
        case "среда" -> "wednesday";
    };
}

```

```

        case "четверг" -> "thursday";
        case "пятница" -> "friday";
        case "суббота" -> "saturday";
        default -> russianDayName;
    };
}
}

class ScheduleException extends Exception {
    public ScheduleException(String message) {
        super(message);
    }
}

//Application.properties

spring.application.name=bot

telegram.bot.username=devmark_scheduleLeti_bot
telegram.bot.token= //не покажу)

leti.api.url=https://digital.etu.ru/api/schedule/groups/

```

ЗАКЛЮЧЕНИЕ

В рамках курсовой работы был разработан Telegram-бот для предоставления расписания занятий студентов СПбГЭТУ «ЛЭТИ». Система интегрирована с университетским порталом через REST API, что обеспечивает получение актуальных данных в реальном времени. Бот позволяет пользователям получать расписание на текущий и следующий день, просматривать занятия на всю неделю, выбирать конкретный день недели, а также находить ближайшее предстоящее занятие. Для корректного отображения расписания в зависимости от учебного графика был реализован алгоритм автоматического определения чётности недели. Пользователи могут выбрать или ввести номер своей учебной группы, после чего все запросы к расписанию выполняются с учётом выбранной группы.

Результаты были выложены на Github:
https://github.com/diwkos/OOP_discipline/tree/main/kursov_oop

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Watson, J. Создание Telegram-бота на Java: от идеи до деплоя [Электронный ресурс] // JavaRush. – URL: <https://javarush.com/groups/posts/504-sozдание-telegram-bota-na-java-ot-idei-do-deploja>
2. Пишем простой Telegram bot на java [Электронный ресурс] // Habr. – URL: <https://habr.com/ru/articles/715384/>
3. Создаем Telegram Бота с Нуля на Java и Spring Boot [Электронный ресурс] // YouTube. – URL: <https://www.youtube.com/watch?v=kgkT-FrSBuk>
4. Добавляем телеграм-бота на проект - "Java-проект от А до Я" [Электронный ресурс] // JavaRush. – URL: <https://javarush.com/groups/posts/3204-java-proekt-ot-a-do-ja-dobavljajem-telegram-bota-na-proekt>
5. Расписания занятий и экзаменов [Электронный ресурс] // Санкт-Петербургский государственный электротехнический университет «ЛЭТИ». – URL: <https://etu.ru/ru/studentam/raspisaniya-zanyatiy-i-ekzamenov/>
6. Гамма, Э. Приемы объектно-ориентированного проектирования. Паттерны проектирования / Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес. – СПб.: Питер, 2020.