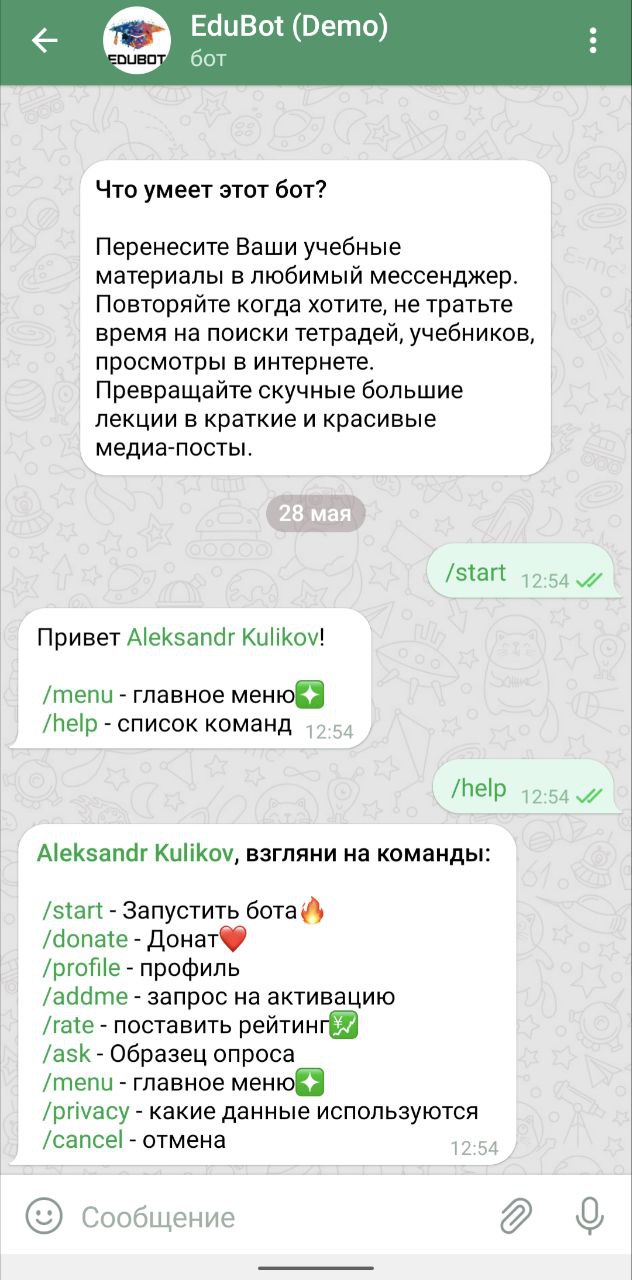
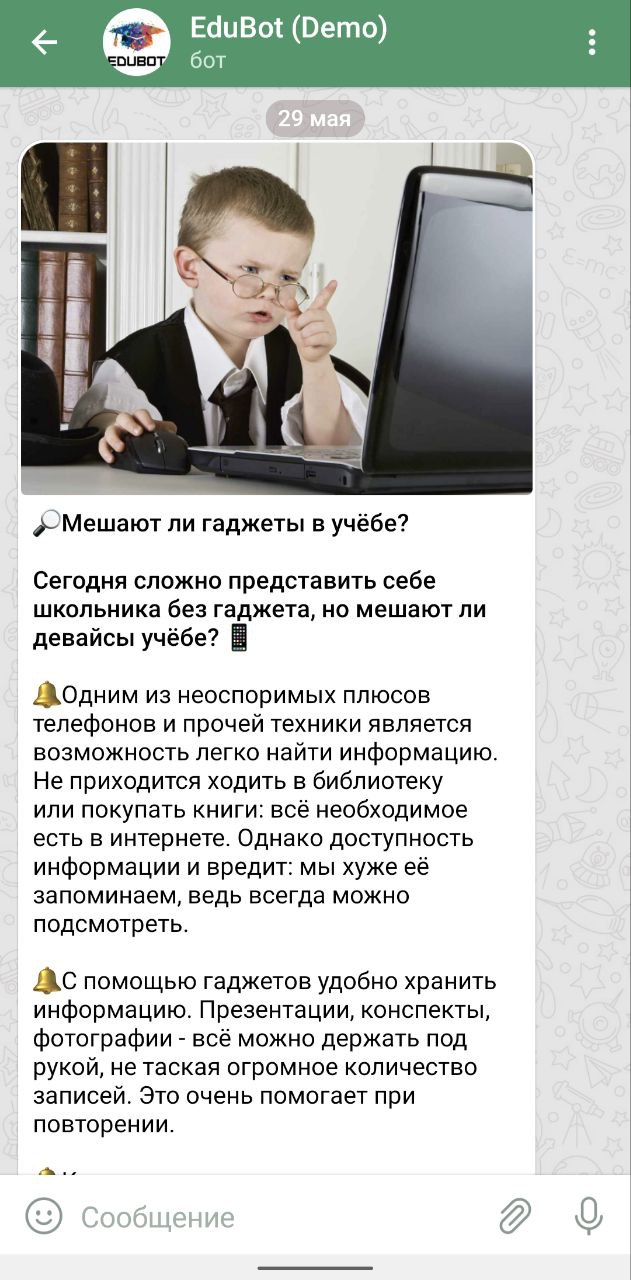
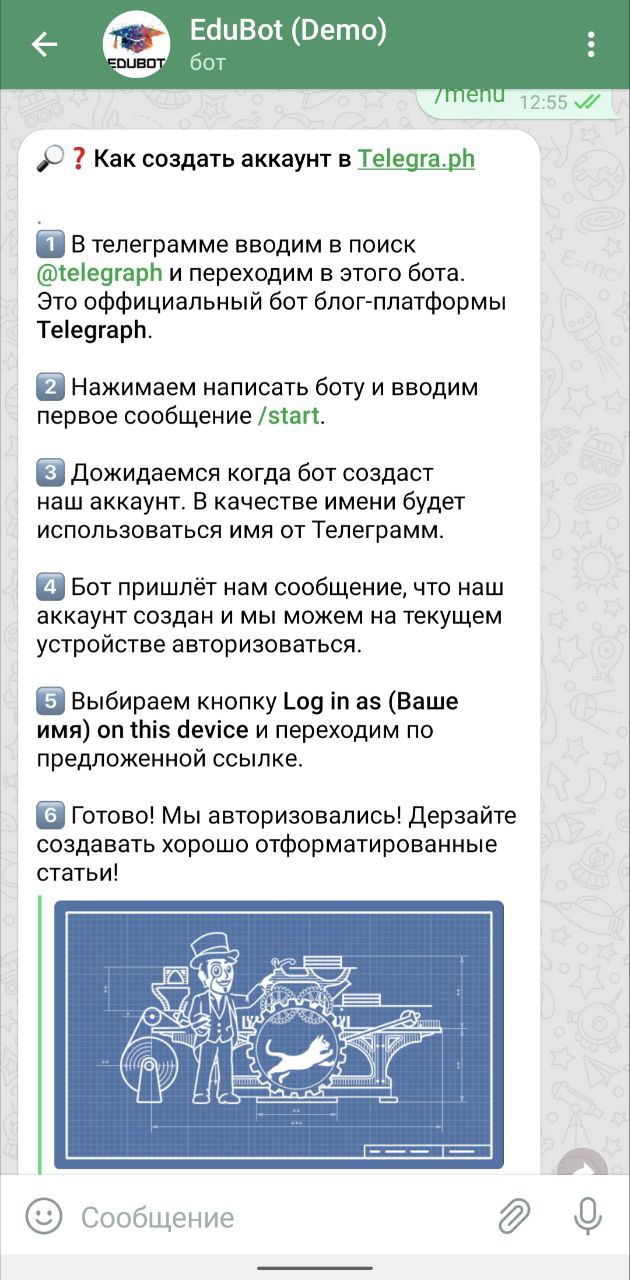
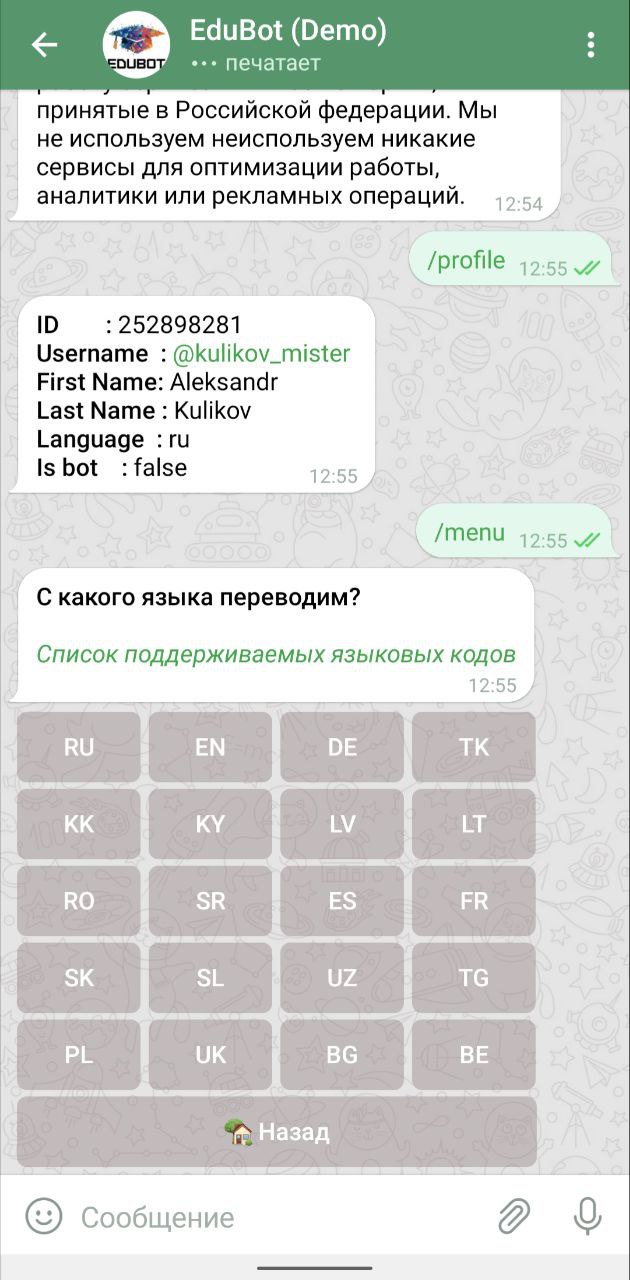
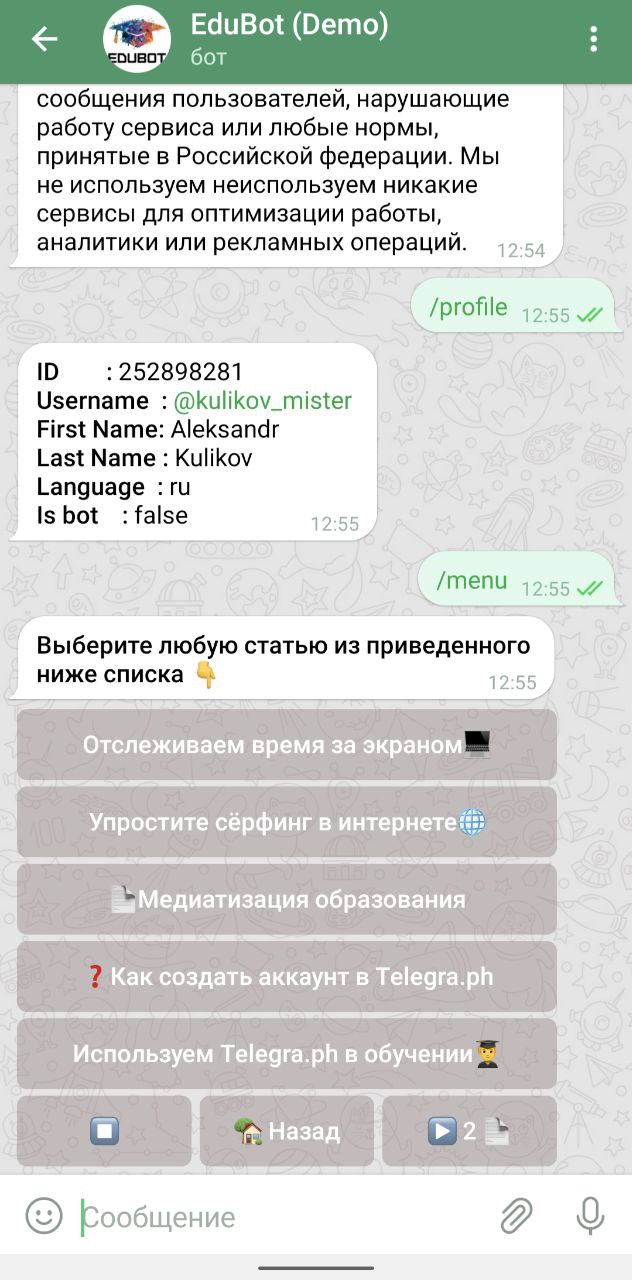
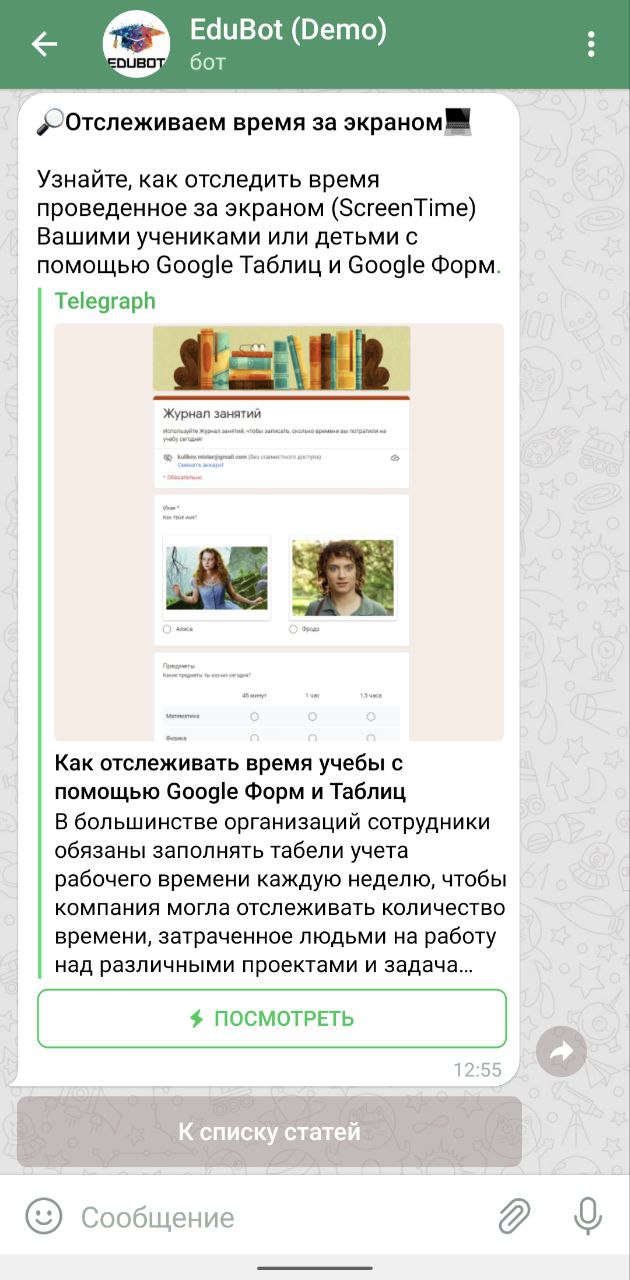
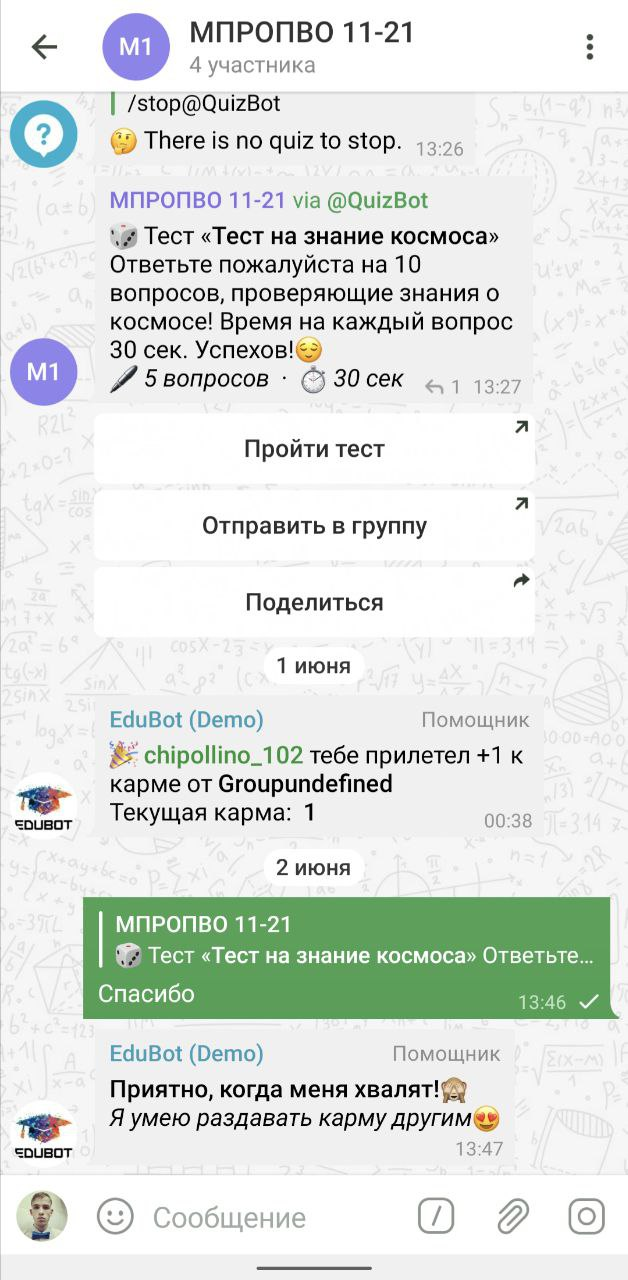
Информационно-образовательная программа «Мобильная система для медиаобразования»

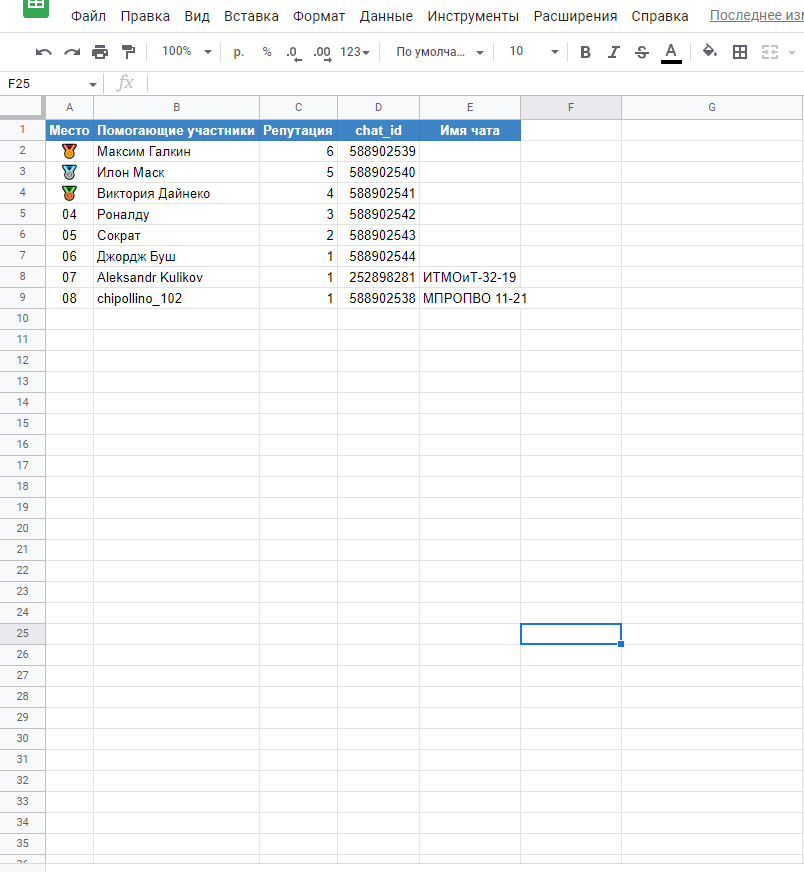
****\

**СКРИНШОТЫ:**





**Рейтинг самых активных учащихся с именами чатов**



Файл **AdminSettings.gs**

const tgBotToken = "???"; //Токен Бота

const botSheet   = ???'; //ID таблицы

const superAdmin = [???]; //ID админа

const webAppURL  = '???'; // URL развёртывания

let PROGRAM\_NAME = "Telegram бот для взаимодействия с Вашими учащимися"

let DATE\_CURRENT = new Date().toLocaleDateString('Ru')

let VERSION = "<b>1.2.1</b>" //вводится рандомно, если планируются доработки кода

function setWebHook() {

  let payload = {

    url: webAppURL

  };

  let response = Bot.request('setWebhook', payload);

  Logger.log(JSON.stringify(response));

}

function deleteWebHook() {

  let payload = {

    url: webAppURL

  };

  let response = Bot.request('deleteWebhook', payload);

  Logger.log(JSON.stringify(response));

}

function oneTimeSetup() {

  Bot.settingUpBotSheet();

}

function scheduleClearTmp\_() {

  Bot.cleanUpBotTmpData();

}

function scheduler() {

  ScriptApp.newTrigger('scheduleClearTmp\_').timeBased().everyDays(1).atHour(4).nearMinute(5).inTimezone("Asia/Yekaterinburg").create();

}

function doGet(e) {

}

const threaded = [//образец многопоточного разговора

  { q: 'Назови мне свое полное имя.' },

  { q: 'Введите Ваш пол?',

    o: {'reply\_markup': {'keyboard': [[{ 'text': 'Мужской' }],[{ 'text': 'Женский' }]],'resize\_keyboard': true,'one\_time\_keyboard': true,'input\_field\_placeholder': 'Мужской или Женский?'}},

    v: '^Мужской$|^Женский$',

    w: '\_Нажмите на клавиатуру ниже.\_' },//warning

  { q: 'Какой у вас номер идентификационной карты?'},

  { q: 'Введите Ваш номер телефона',

    o: {'reply\_markup': {'keyboard': [[{ 'text': 'Отправить номер телефона', 'request\_contact': true }]],'resize\_keyboard': true,'one\_time\_keyboard': true,'input\_field\_placeholder': 'Номер телефона'}},

    v: '^\\d{11}$',

    w: '\_Неверный номер телефона. Пожалуйста, введите снова.\_'}

];

Файл **doPost.gs**

// https://github.com/peterherrmann/BetterLog

let Logger = BetterLog.useSpreadsheet(botSheet);

let Bot = TelegramLibrary.createBot(tgBotToken, botSheet);

let TelegramJSON;

function doPost(e) {

  if(e.postData.type == "application/json") {

    TelegramJSON = JSON.parse(e.postData.contents);

    Bot.getUpdate(TelegramJSON);

    Logger.log(JSON.stringify(TelegramJSON));

    debug(TelegramJSON);

    if(Bot.isCallbackQuery()) {//если callback\_query

      callbackQueryChat(TelegramJSON);

    }

    if(Bot.isBotCommand(TelegramJSON)) {// если команды боту

      if (Bot.isChatType('private')) {//если команда в личный чат

        message\_commands(TelegramJSON)

      }

      else{//если команда в другой чат

      }

    }

    //если пользователь Ведет Многопоточный Разговор

    let tc = Bot.userHasThreadedConversation();

    // многопоточный разговор

    if(tc.found) {

      if(tc.step == threaded.length) {

        let ans = Bot.endThreadedConversation(threaded,tc.step);

        if(ans) {

          // выполните обработку здесь

          Logger.log(ans);

          let msg = "Данные были записаны. Спасибо.";

          Bot.sendMessageKeyboardRemove(msg);

        }

      }

      else //следующее Сообщение В Многопоточном Разговоре

        Bot.nextMessageInThreadedConversation(threaded, tc.step);

    }

    if (Bot.isNewChatMember() || Bot.isLeftChatMember()) {

      let resp = Bot.deleteMessage(TelegramJSON.message.message\_id);

      console.log(resp.result);

      message\_chat(TelegramJSON);

      return

    }

    if(Bot.isTextMessage()) {// обычные сообщения

      if (Bot.isChatType('private')) {//если в личный чат

        msgText(TelegramJSON)

      }

      else {

        message\_chat(TelegramJSON)

      }

    }

    if(Bot.isMap()) {// координаты, локации

      if (Bot.isChatType('private')) {

        let lat, long  = [TelegramJSON.message.location.latitude, TelegramJSON.message.location.longitude];

        Bot.sendVenue(lat, long, lat+','+long, 'Моё местоположение в данный момент.');

        return

      }

    }

    if(Bot.isPhoto()){//если фото

      if (Bot.isChatType('private')) {//если в личный чат

        msgPhoto(TelegramJSON)

        return

      }

    }

    if(Bot.isVideo()){//если видео

      if (Bot.isChatType('private')) {//если в личный чат

        msgVideo(TelegramJSON)

        return

      }

    }

    if(Bot.isAudio()){//если аудио

      if (Bot.isChatType('private')) {//если в личный чат

        msgAudio(TelegramJSON)

        return

      }

    }

    if(Bot.isDocument()){//если Документ

      if (Bot.isChatType('private')) {//если в личный чат

        msgDoc(TelegramJSON)

        return

      }

    }

  }

}

Файл **CallbackQuery.gs**

function callbackQuery(TelegramJSON) {//обработка нажатий в личном чате пользователя

  let options = {

    chat\_id: TelegramJSON.callback\_query.message.chat.id,

    message\_id: TelegramJSON.callback\_query.message.message\_id,

    data: TelegramJSON.callback\_query.data,

    parse\_mode: 'HTML',

    disable\_web\_page\_preview: true,

    disable\_notification: true,

    text: '' || TelegramJSON.callback\_query.message.text,

    reply\_markup: '',

  }

  let userProperties = PropertiesService.getUserProperties();

  // callback\_query

  if(Bot.isCallbackQuery()) {

    let a = Bot.getSystemUser();

    if(a && a.isAuth) {

      let cb = TelegramJSON.callback\_query;

      let msg = '', msg2 = '';

      if(options.data === 'menu\_1') {

        if (superAdmin.includes(Bot.getUserID())) {

          Bot.sendChatAction();

          msg = "<b>Вы в главном МЕНЮ:</b>\n\n<i>выберите любой вариант из приведенного ниже списка.</i>";

          Bot.editMessageKeyboard(msg, options.message\_id, null, {'inline\_keyboard': menu[4]});

          Bot.request('answerCallbackQuery', { callback\_query\_id: cb.id});

          return;

        }

        else{

          Bot.sendChatAction();

          msg = "<b>Вы в главном МЕНЮ:</b>\n\n<i>выберите любой вариант из приведенного ниже списка.</i>";

          Bot.editMessageKeyboard(msg, options.message\_id, null, {'inline\_keyboard': menu[1]});

          Bot.request('answerCallbackQuery', { callback\_query\_id: cb.id});

          return;

        }

      }

      if(options.data === 'menu\_start'){

        Bot.sendChatAction();

        msg = "<b>Вы в главном МЕНЮ:</b>\n\n<i>выберите любой вариант из приведенного ниже списка.</i>";

        Bot.editMessageKeyboard(msg, options.message\_id, null, {'inline\_keyboard': menu[0]});

        Bot.request('answerCallbackQuery', { callback\_query\_id: cb.id});

        return;

      }

      if(options.data === 'profile') {

        Bot.sendChatAction();

        let isb;

        if (cb.from.is\_bot === 'true'){isb = " Да"}else{isb = " Нет"};

        let msg = "<i><b>ID        :</b></i> " + Bot.getUserID() + "\n" +

                  "<i><b>Username  :</b></i> " + Bot.getUsername() + "\n" +

                  "<i><b>First Name:</b></i> " + Bot.getUserFirstName() + "\n" +

                  "<i><b>Last Name :</b></i> " + [cb.from.last\_name || ""] + "\n" +

                  "<i><b>Бот       :</b></i> " + isb

        Bot.editMessageKeyboard(msg, options.message\_id, null, {'inline\_keyboard': menu[2]});

        Bot.request('answerCallbackQuery', { callback\_query\_id: cb.id});

        return;

      }

      if (options.data === 'donate') {

        Bot.sendChatAction();

        Bot.editMessageKeyboard(msgd, options.message\_id, null, {'inline\_keyboard': menu[2]});

        Bot.request('answerCallbackQuery', { callback\_query\_id: cb.id});

        return;

      }

      if (options.data === 'stop') {

        Bot.request('answerCallbackQuery', { callback\_query\_id: cb.id, text: '🥺 Дальше пусто' });

        return;

      }

      if (options.data === 'Settings') {

        Bot.sendChatAction();

        Bot.editMessageKeyboard('Вы в меню: <b>НАСТРОЙКИ</b>', options.message\_id, null, {'inline\_keyboard': menu[3]});

        Bot.request('answerCallbackQuery', { callback\_query\_id: cb.id});

        return;

      }

      if (options.data === 'translate') {

        Bot.sendChatAction();

        let lang\_msg = `<a href="https://telegra.ph/List-Language-05-21"><i>Список поддерживаемых языковых кодов</i></a>`;

        Bot.editMessageKeyboard(`<b>C какого языка переводим?</b>\n\n${lang\_msg}`, options.message\_id, null, {'inline\_keyboard': menu[9]});

        Bot.request('answerCallbackQuery', { callback\_query\_id: cb.id});

        return;

      }

      if (options.data.split('\_')[1] === 'sourceL') {

        Bot.sendChatAction();

        let lang\_msg = `<a href="https://telegra.ph/List-Language-05-21"><i>Список поддерживаемых языковых кодов</i></a>`;

        sl = options.data.split('\_')[0];

        let keys = [[{text: 'RU', callback\_data: `${sl}\_tlang\_ru`},{text: 'EN', callback\_data: `${sl}\_tlang\_en`},{text: 'DE', callback\_data: `${sl}\_tlang\_de`},{text: 'TK', callback\_data: `${sl}\_tlang\_tk`}],

  [{text: 'KK', callback\_data: `${sl}\_tlang\_kk`},{text: 'KY', callback\_data: `${sl}\_tlang\_ky`},{text: 'LV', callback\_data: `${sl}\_tlang\_lv`},{text: 'LT', callback\_data: `${sl}\_tlang\_lt`}],

  [{text: 'RO', callback\_data: `${sl}\_tlang\_ro`},{text: 'SR', callback\_data: `${sl}\_tlang\_sr`},{text: 'ES', callback\_data: `${sl}\_tlang\_es`},{text: 'FR', callback\_data: `${sl}\_tlang\_fr`}],

  [{text: 'SK', callback\_data: `${sl}\_tlang\_sk`},{text: 'SL', callback\_data: `${sl}\_tlang\_sl`},{text: 'UZ', callback\_data: `${sl}\_tlang\_uz`},{text: 'TG', callback\_data: `${sl}\_tlang\_tg`}],

  [{text: 'PL', callback\_data: `${sl}\_tlang\_pl`},{text: 'UK', callback\_data: `${sl}\_tlang\_uk`},{text: 'BG', callback\_data: `${sl}\_tlang\_bg`},{text: 'BE', callback\_data: `${sl}\_tlang\_be`}],

  [{text: '🏡 Назад', callback\_data: 'translate'}]];

        Bot.editMessageKeyboard(`<b>На какой язык переводим?</b>\n\n${lang\_msg}`, options.message\_id, null, {'inline\_keyboard': keys,});

        Bot.request('answerCallbackQuery', { callback\_query\_id: cb.id});

        return;

      }

      if (options.data.split('\_')[1] === 'tlang') {//если выбран втрой язык для перевода

        let user = {

          callback\_query\_data: options.data,

        };

        userProperties.setProperty(options.chat\_id, JSON.stringify(user));

        Bot.editMessageText(`Введите текст для перевода. (${options.data.split('\_')[0]} => ${options.data.split('\_')[2]})`, options.message\_id, null);

        Bot.request('answerCallbackQuery', { callback\_query\_id: cb.id});

        return;

      }

      if (options.data === 'schedule'){

        Bot.sendChatAction();

        if(schedule\_msg !== ''){ //если ссылка для расписания имеется

          let msg = 'Расписание🗒';

          Bot.editMessageKeyboard(msg, options.message\_id, null, {"inline\_keyboard": [[{ text: 'Открыть', url: schedule\_msg }],[{text: '🏡 Назад', callback\_data: 'menu\_1'}]] });

          Bot.request('answerCallbackQuery', { callback\_query\_id: cb.id});

          return;

        }

        else{

          Bot.request('answerCallbackQuery', { callback\_query\_id: cb.id, text: '🥺 Расписание отсутсвует!'});

          return;

        }

      }

      if (options.data === 'courses') {

        Bot.sendChatAction();

        let res = [];

        nameOfSheets().forEach(function(item){res.push(`\n${item}`)});

        Bot.editMessageKeyboard(`<b>Список предметов:</b> \n${res}`, options.message\_id, null, {'inline\_keyboard': menu[8]});

        Bot.request('answerCallbackQuery', { callback\_query\_id: cb.id});

        return;

      }

      if (options.data === 'Добавить расписание'){

        let user = {

          callback\_query\_data: 'Добавить расписание',

        };

        userProperties.setProperty(options.chat\_id, JSON.stringify(user));

        Bot.editMessageKeyboard('<b>Пришлите прямую ссылку на Ваше расписание</b>', options.message\_id, null, {'inline\_keyboard': menu[8]});

        Bot.request('answerCallbackQuery', { callback\_query\_id: cb.id});

        return;

      }

      if (options.data === 'Удалить расписание'){

        if (schedule\_msg.trim() == '') { //если ссылка для расписания имеется

          Bot.request('answerCallbackQuery', { callback\_query\_id: cb.id, text: '⚠️ Расписание отсутсвует!'});

          return;

        }

        else {

          setSheetVal('Settings', 2, 1, '');//установка в графу с расписанием пустого значения

          Bot.request('answerCallbackQuery', { callback\_query\_id: cb.id, text: '✅ Расписание успешно удалено!'});

          return;

        }

      }

      if (options.data === 'Examples') {

        let user = {

          callback\_query\_data: options.data,

        };

        userProperties.setProperty(options.chat\_id, JSON.stringify(user));

        let data = getDataShhet('Examples');

        let msg = '<b>Выберите любую статью из приведенного ниже списка </b>👇';

        try{

          Bot.editMessageKeyboard(msg, options.message\_id, null, keySheets(data, undefined, return\_key = "Settings").keyboard);

        }

        catch{//если не получается изменить сообщение для возврата назад

          Bot.deleteMessage(options.message\_id);

          Bot.sendMessage(msg, {'reply\_markup': keySheets(data, undefined, return\_key = "Settings").keyboard});

        }

        Bot.request('answerCallbackQuery', { callback\_query\_id: cb.id});

      }

      if (options.data === 'Добавить предмет') {

        let user = {

          callback\_query\_data: options.data,

        };

        userProperties.setProperty(options.chat\_id, JSON.stringify(user));

        Bot.editMessageText('Напишите название предмета.', options.message\_id, null);

        Bot.request('answerCallbackQuery', { callback\_query\_id: cb.id});

        return;

      }

      if (options.data === 'Посмотреть лекции' || options.data === 'Удалить материал' || options.data === 'Удалить предмет' || options.data === 'Добавить материал') {

        let user = {

          callback\_query\_data: options.data,

        };

        if (options.data === 'Добавить материал') {

          msg = 'Выбери предмет, в который нужно добавить материал 👇';

          userProperties.setProperty(options.chat\_id, JSON.stringify(user));

          Bot.editMessageKeyboard(msg, options.message\_id, null, keySheets(nameOfSheets(), undefined, return\_key = "menu\_1").keyboard);

          Bot.request('answerCallbackQuery', { callback\_query\_id: cb.id});

        }

        if (options.data === 'Посмотреть лекции') {

          msg = 'Выбери предмет, который Вы хотите посмотреть👇'};

        if (options.data === 'Удалить материал') {

          msg = 'Выбери предмет, в котором Вы хотите <b>удалить</b> материал👇'};

        if (options.data === 'Удалить предмет') {

          msg = `Выберите предмет, который хотите удалить`};

        if(nameOfSheets().length > 0){//проверка на наличие предметов

          userProperties.setProperty(options.chat\_id, JSON.stringify(user));

          Bot.editMessageKeyboard(msg, options.message\_id, null, keySheets(nameOfSheets(), undefined, return\_key = "menu\_1").keyboard);

          Bot.request('answerCallbackQuery', { callback\_query\_id: cb.id});

        }

        else{

          Bot.request('answerCallbackQuery', { callback\_query\_id: cb.id, text: '🥺 Здесь пусто!' });

        }

        return;

      }

      let sheets = keySheets(nameOfSheets()).nameOfSheets;

      let userData = JSON.parse(userProperties.getProperty(options.chat\_id));

      if (sheets.includes(options.data) && userData.callback\_query\_data === 'Удалить предмет'){

        delSheet(options.data);

        Bot.editMessageKeyboard(`Выберите предмет, который хотите <b>удалить</b>👇`, options.message\_id, null, keySheets(nameOfSheets(), undefined, return\_key = "menu\_1").keyboard);

      }

      if (sheets.includes(options.data)){//если выбран предмет из списка

        let data = getDataShhet(options.data);

        if (data !== 0) { //если в списке предметов есть информация

          let x = userData.callback\_query\_data;

          switch(x) {

            case 'Посмотреть лекции':

              userProperties.setProperty(options.chat\_id, JSON.stringify({callback\_query\_data: `find\_${msg}`}));

              Bot.editMessageKeyboard(`<b>${options.data}</b>` , options.message\_id, null, keySheets(data).keyboard);

              Bot.request('answerCallbackQuery', { callback\_query\_id: cb.id});

              break;

            case 'Удалить предмет':

              delSheet(options.data);

              Bot.editMessageKeyboard(`Выберите предмет, который хотите <b>удалить</b>👇`, options.message\_id, null, keySheets(nameOfSheets(), undefined, return\_key = "menu\_1").keyboard);

              Bot.request('answerCallbackQuery', { callback\_query\_id: cb.id});

              break;

            case 'Удалить материал':

              userProperties.setProperty(options.chat\_id, JSON.stringify({callback\_query\_data: `deletem\_${msg}`}));

              Bot.editMessageKeyboard(`Выберите предмет, в котором Вы хотите <b>удалить</b> материал👇` , options.message\_id, null, keySheets(data).keyboard);

              Bot.request('answerCallbackQuery', { callback\_query\_id: cb.id});

              break;

            default:

              Bot.request('answerCallbackQuery', { callback\_query\_id: cb.id});//ответный запрос, чтобы не было зацикливаний

              break;

          }

        }

        if (userData.callback\_query\_data === 'Добавить материал'){//если в списке предметов есть информация/или нет

          userProperties.setProperty(options.chat\_id, JSON.stringify({callback\_query\_data: `addm\_${options.data}`}));

          Bot.editMessageText('<b>Пришлите Ваш новый материал.</b>', options.message\_id);

          Bot.request('answerCallbackQuery', { callback\_query\_id: cb.id});

        }

        else {

          Bot.request('answerCallbackQuery', { callback\_query\_id: cb.id, text: '🥺 Здесь пусто!' });

        }

      }

      if (options.data.includes('page')) {//если нажата кнопка перелистнуть страницу в списке лекций

        let page = options.data.substr(-1);

        let x = userData.callback\_query\_data;

        Bot.request('answerCallbackQuery', { callback\_query\_id: cb.id});//ответный запрос, чтобы не было зацикливаний

        switch(x) {

          case 'Посмотреть лекции':

            msg = 'Выбери предмет, который Вы хотите <b>посмотреть</b>👇';

            Bot.editMessageKeyboard(msg, options.message\_id, null, keySheets(nameOfSheets(), page, return\_key = "menu\_1").keyboard);

            break;

          case 'Добавить материал':

            msg = 'Выбери предмет, в который нужно <b>добавить</b> материал👇';

            Bot.editMessageKeyboard(msg, options.message\_id, null, keySheets(nameOfSheets(), page, return\_key = "menu\_1").keyboard);

            break;

          case 'Удалить предмет':

            msg = 'Выберите предмет, который хотите <b>удалить</b>👇';

            Bot.editMessageKeyboard(msg, options.message\_id, null, keySheets(nameOfSheets(), page, return\_key = "menu\_1").keyboard);

            break;

          case 'Удалить материал':

            msg = 'Выберите предмет, в котором Вы хотите <b>удалить</b> материал👇';

            Bot.editMessageKeyboard(msg, options.message\_id, null, keySheets(nameOfSheets(), page, return\_key = "menu\_1").keyboard);

            break;

          case 'Examples':

            msg = 'Выберите любую статью из приведенного ниже списка 👇';

            data = getDataShhet('Examples');

            Bot.editMessageKeyboard(msg, options.message\_id, null, keySheets(data, page, return\_key = 'Settings').keyboard, {'disable\_web\_page\_preview': true});

            break;

          default:

            Bot.request('answerCallbackQuery', { callback\_query\_id: cb.id});//ответный запрос, чтобы не было зацикливаний

            break;

        }

      }

      if (options.data.includes('page') && sheets.includes(options.text)) {//если текст статьи совпадает с текстом кнопки, чтобы правильно корректно срабатывали нажатия

        let page = options.data.substr(-1);

        let data = getDataShhet(options.text);

        let msg = '<b>'+userData.callback\_query\_data.split('\_')[1]+'</b>';

        Bot.editMessageKeyboard(msg, options.message\_id, null, keySheets(data, page).keyboard);

        Bot.request('answerCallbackQuery', { callback\_query\_id: cb.id});

        return;

      }

      if(options.data.split('\_')[0] === 'user') {//авторизация пользователя

        if(options.data.split('\_')[1] === 'approve') {

          Bot.authSystemUser(options.data.split('\_')[2], true);

          msg  = "<i>Вы были авторизованы.</i> Приятного пользования!";

          msg2 = "Запрос от <a href='tg://user?id="+options.data.split('\_')[2] + "'>"+options.data.split('\_')[2]+"</a> был <b>одобрен</b> вами.";

        }

        else if(options.data.split('\_')[1] === 'deny') {

          Bot.authSystemUser(options.data.split('\_')[2], false);

          msg  = "<i>К сожалению, Ваш запрос был отклонен!</i>";

          msg2 = "Запрос от <a href='tg://user?id="+options.data.split('\_')[2] + "'>"+options.data.split('\_')[2]+"</a> был <b>отклонен</b> вами.";

        }

        Bot.sendMessage(msg, { chat\_id: options.data.split('\_')[2] });

        Bot.editMessageText(msg2, options.message\_id);

        Bot.request('answerCallbackQuery', { callback\_query\_id: cb.id});

        return;

      }

      if(options.data.split('\_')[0] === 'article') {//кнопка "назад" при просмотре материала

        let data = getDataShhet(options.data.split('\_')[1]);

        let msg = `<b>${options.data.split('\_')[1]}</b>`;

        try{

          Bot.editMessageKeyboard(msg, options.message\_id, null, keySheets(data).keyboard);

        }

        catch{

          Bot.deleteMessage(options.message\_id);

          Bot.sendMessage(msg, {'reply\_markup': keySheets(data).keyboard});

        }

        Bot.request('answerCallbackQuery', { callback\_query\_id: cb.id});

        return;

      }

      if(options.data === 'delete'){

        Bot.deleteMessage(TelegramJSON.callback\_query.message.message\_id);

        Bot.request('answerCallbackQuery', { callback\_query\_id: cb.id});

        return;

      }

      if (search\_row(options.data, 'Examples') >= 0 && userData.callback\_query\_data === 'Examples'){//если пользователь просматривает Examples

        let lastpost = SpreadsheetApp.openById(botSheet).getSheetByName('Examples').getRange(search\_row(options.data, 'Examples'), 1, 1,  4).getValues()[0]

        let msg = '<b>🔎' + lastpost[0] + '</b>' + '\n\n'+ lastpost[1];

        Bot.request('answerCallbackQuery', { callback\_query\_id: cb.id});

        if (lastpost[3] == 'Текст') {

          Bot.editMessageKeyboard(msg, options.message\_id, null, {'inline\_keyboard': [[{ text: 'К списку статей', 'callback\_data': `Examples` }]]}, {'disable\_web\_page\_preview': false});

        }

        if (lastpost[3] == 'Изображение') {

          Bot.deleteMessage(options.message\_id);

          Bot.sendPhoto(msg, String(lastpost[2]), menu[10], {'disable\_web\_page\_preview': false});

        }

        if (lastpost[3] == 'Видео') {

          Bot.deleteMessage(options.message\_id);

          Bot.sendVideo(msg, String(lastpost[2]), menu[10], {'disable\_web\_page\_preview': false});

        }

        if (lastpost[3] == 'Аудио') {

          Bot.deleteMessage(options.message\_id);

          Bot.sendAudio(msg, String(lastpost[2]), menu[10], {'disable\_web\_page\_preview': false});

        }

        if (lastpost[3] == 'Документ') {

          Bot.deleteMessage(options.message\_id);

          Bot.sendDocument(msg, String(lastpost[2]), menu[10], {'disable\_web\_page\_preview': false});

        }

      }

      let msgdata = userData.callback\_query\_data.split('\_');

      if (search\_row(options.data, options.text) >= 0) {

        if (msgdata[0] === 'find'){//если пользователь просматривает материалы

          let lastpost = SpreadsheetApp.openById(botSheet).getSheetByName(options.text).getRange(search\_row(options.data, options.text), 1, 1,  4).getValues()[0]

          let msg = '<b>🔎' + lastpost[0] + '</b>' + '\n\n'+ lastpost[1];

          let x = lastpost[3]

          Bot.request('answerCallbackQuery', { callback\_query\_id: cb.id});

          switch(x) {

            case 'Изображение':

              Bot.deleteMessage(options.message\_id);

              Bot.sendPhoto(msg, String(lastpost[2]), [[{'text': 'К списку материалов', 'callback\_data': `article\_${options.text}`}]]);

              break;

            case 'Видео':

              Bot.deleteMessage(options.message\_id);

              Bot.sendVideo(msg, String(lastpost[2]), [[{'text': 'К списку материалов', 'callback\_data': `article\_${options.text}`}]]);

              break;

            case 'Аудио':

              Bot.deleteMessage(options.message\_id);

              Bot.sendAudio(msg, String(lastpost[2]), [[{'text': 'К списку материалов', 'callback\_data': `article\_${options.text}`}]]);

              break;

            case 'Документ':

              Bot.deleteMessage(options.message\_id);

              Bot.sendDocument(msg, String(lastpost[2]), [[{'text': 'К списку материалов', 'callback\_data': `article\_${options.text}`}]]);

              break;

            case 'Текст':

              Bot.editMessageKeyboard(msg, options.message\_id, null,{'inline\_keyboard': [[{ text: 'К списку материалов', callback\_data: `article\_${options.text}`}]],

                                                                     'disable\_web\_page\_preview': false});

              break;

            default:

              Bot.request('answerCallbackQuery', { callback\_query\_id: cb.id, text: '🥺 Какая-то ошибка!' });

              break;

          }

        }

        if (msgdata[0] === 'deletem') {//если пользователь желает удалить материал

          let SpreadSheet = SpreadsheetApp.openById(botSheet);

          let Sheet = SpreadSheet.getSheetByName(options.text);

          Sheet.deleteRow(search\_row(options.data, options.text));

          let data = getDataShhet(msgdata[1]);

          Bot.editMessageKeyboard(`${msgdata[1]}` , options.message\_id, null, keySheets(data, undefined, return\_key = 'Удалить материал').keyboard);

          Bot.request('answerCallbackQuery', { callback\_query\_id: cb.id});

        }

      }

    }

    else{//если пользователь не зарегистрирован в боте

      let msg = "Вы не зарегестрированы. \n<i>Пожалуйста, нажмите /addme, чтобы отправить запрос на авторизацию.</i>";

      Bot.sendMessage(msg);

      Bot.request('answerCallbackQuery', { callback\_query\_id: cb.id});

      return;

    }

  }

}

Файл **callbackQueryChat.gs**

function callbackQueryChat(TelegramJSON) {//обработка нажатий в чатах

  let options = {

    chat\_id: TelegramJSON.callback\_query.message.chat.id,

    message\_id: TelegramJSON.callback\_query.message.message\_id,

    data: TelegramJSON.callback\_query.data,

    parse\_mode: 'HTML',

    disable\_web\_page\_preview: true,

    disable\_notification: true,

    text: '' || TelegramJSON.callback\_query.message.text,

    reply\_markup: '',

  }

  if (options.data == 'riding\_rules') {

    Bot.editMessageKeyboard(rules, options.message\_id, options.chat\_id, {'inline\_keyboard': menu[6]});

    Bot.request('answerCallbackQuery', {callback\_query\_id: TelegramJSON.callback\_query.id})

  }

  if (options.data == 'agree with rules') {

    let option = {

      user\_id: TelegramJSON.callback\_query.from.id,

      parse\_mode: 'HTML',

      disable\_web\_page\_preview: true,

      disable\_notification: true,

      permissions: {

        can\_send\_messages: true,

        can\_send\_media\_messages: true,

        can\_send\_polls: true,

        can\_send\_other\_messages: true,

        can\_add\_web\_page\_previews: true,

        can\_invite\_users: true,

      }

    }

    let cache = CacheService.getScriptCache();

    let cacheData = JSON.parse(cache.get(TelegramJSON.callback\_query.from.id));

    if (cacheData !== null) {

      option.chat\_id = cacheData.chat\_id

      option.message\_id = cacheData.message\_id

      option.parse\_mode = 'HTML',

      option.text = cacheData.text

      Bot.editMessageKeyboard(option.text, option.message\_id, option.chat\_id, {"inline\_keyboard": menu[7],"resize\_keyboard": true});

    }

    Bot.restrictChatMember(options.chat\_id, option.user\_id, option.permissions, null);

    answer = {

      callback\_query\_id: TelegramJSON.callback\_query.id,

      text: '✅ Теперь можешь писать в чат 📝',

      show\_alert: true

    }

    Bot.request('answerCallbackQuery',answer)

  }

  if (options.data == 'pass') {

    Bot.request('answerCallbackQuery',answer)

  }

  else{//остальные нажатия кнопок отрабатываются внутри бота

    callbackQuery(TelegramJSON)

  }

}

Файл **messageCommands.gs**

 function message\_commands(TelegramJSON) {

  let userProperties = PropertiesService.getUserProperties();

  userProperties.deleteProperty(TelegramJSON.message.chat.id);

  let text = TelegramJSON.message.text;

  if (text === '/start'){

    let msg = "Привет <a href='" + Bot.mentionByID() +"'>" + Bot.getUserFullName() +"</a>! \n\n /menu - главное меню❇️ \n /help - список команд";

    Bot.sendMessage(msg);

  }

  if (text === '/cancel' && !Bot.userHasThreadedConversation().found){

    Bot.sendMessage(`<i>Пользователь отменил операцию</i> \n\n/menu - главное меню❇️`);

  }

  if (text === '/help'){

    let msg = "<b><a href='" + Bot.mentionByID() +"'>" + Bot.getUserFullName() + "</a>, взгляни на команды:</b>\n\n /start - Запустить бота🔥 \n /donate - Донат❤️ \n /profile - профиль \n /addme - запрос на активацию \n /rate - поставить рейтинг💹 \n /ask - Образец опроса \n /menu - главное меню❇️ \n /privacy - какие данные используются \n /cancel - отмена";

    Bot.sendMessage(msg);

  }

  if (text === '/privacy'){

    Bot.sendMessage(privacy\_msg);

  }

  if (text === '/donate'){

    Bot.sendMessage(msgd);

  }

  if (text === '/schedule'){

    if(schedule\_msg != ""){

      keyboard = {

      reply\_markup: {"inline\_keyboard": [[{ text: 'Открыть', url: schedule\_msg }]]}

      }

      let msg = 'Расписание🗒'

      Bot.sendMessage(msg, keyboard)

    }

    else{

      let msg = 'Расписание сейчас отсутствует. Используйте /addschedule, чтобы добавить ссылку на расписание'

      Bot.sendMessage(msg);

    }

  }

  if (text === '/menu'){

    if(Bot.getSystemUser().isAuth) {//если суперадмин

      msg = "<b>Вы в главном МЕНЮ:</b>\n\n<i>выберите любой вариант из приведенного ниже списка.</i>";

      Bot.sendMessage(msg, {'reply\_markup': {'inline\_keyboard': menu[0]}});

    }

    else{//если неавторизованный пользователь

      let msg = "Вы не зарегестрированы. \n<i>Пожалуйста, нажмите /addme, чтобы отправить запрос на авторизацию.</i>";

      Bot.sendMessage(msg);

    }

  }

  if (text === '/set\_lang'){

    Bot.sendChatAction();

    let lang\_msg = `<a href="https://telegra.ph/List-Language-05-21"><i>Список поддерживаемых языковых кодов</i></a>`;

    Bot.sendMessage(`<b>C какого языка переводим</b>\n\n${lang\_msg}`, {'reply\_markup': {'inline\_keyboard': menu[9]}});

    return

  }

  if (text === '/ask'){

    Bot.startThreadedConversation(threaded);

    return

  }

  if(text === '/top'){//команда для топлиста

    let msg = leaderBoard();

    Bot.sendMessage(msg);

  }

  if (text === '/profile'){

    msg = "<b>ID        :</b> " + Bot.getUserID() + "\n" +

          "<b>Username  :</b> " + "<a href='" + Bot.mentionByID() +"'>" + Bot.getUsername() + "</a>\n" +

          "<b>First Name:</b> " + Bot.getUserFirstName() + "\n" +

          "<b>Last Name :</b> " + Bot.getUserLastName() + "\n" +

          "<b>Language  :</b> " + TelegramJSON.message.from.language\_code + "\n" +

          "<b>Is bot    :</b> " + TelegramJSON.message.from.is\_bot;

    Bot.sendMessage(msg);

  }

  if (text === '/addme'){

    if(Bot.getSystemUser() && Bot.getSystemUser().isAuth) {

      let msg = "Вы уже являетесь авторизованным пользователем.";

      Bot.sendMessage(msg);

      return;

    }

    else if(Bot.getSystemUser()) {

      let msg = "Вы уже делали запрос раньше. Пожалуйста, дождитесь ответа администратора.";

      Bot.sendMessage(msg);

      return;

    }

    Bot.addSystemUser();

    msg = "Ваш запрос был отправлен администратору.";

    Bot.sendMessage(msg);

    // отправка сообщение с запросом администратору

    let sendTo = superAdmin || Bot.getAdminsID();

    const len = sendTo.length;

    for(let i = 0; i < len; i++) {

      let options = {

        'chat\_id': sendTo[i],

        'reply\_markup': {

          'inline\_keyboard': [

            [

              { 'text': '⛔️Отклонить', 'callback\_data': 'user\_deny\_' + Bot.getUserID() },

              { 'text': '✅Принять', 'callback\_data': 'user\_approve\_' + Bot.getUserID() }

            ]

          ]

        }

      };

      msg1 = "Этот пользователь запрашивает у вас разрешение\n\n" +

            "<i>ID        :</i> " + "<a href='" + Bot.mentionByID() +"'>" + Bot.getUserID() + "</a>\n" +

            "<i>Username  :</i> " + "<a href='" + Bot.mentionByID() +"'>" + Bot.getUsername() + "</a>\n" +

            "<i>First Name:</i> " + Bot.getUserFirstName() + "\n" +

            "<i>Last Name :</i> " + Bot.getUserLastName() + "\n" +

            "<i>Language  :</i> " + TelegramJSON.message.from.language\_code + "\n" +

            "<i>Is bot    :</i> " + TelegramJSON.message.from.is\_bot;

      Bot.sendMessage(msg1, options);

    }

  }

  if (text === '/rate'){

    let keyboard = '⭐️,⭐️⭐️;⭐️⭐️⭐️,⭐️⭐️⭐️⭐️,⭐️⭐️⭐️⭐️⭐️';

    msg = "Как вы оцениваете этого бота?";

    Bot.sendMessageCustomKeyboard(msg, keyboard, 'Дайте мне свои звезды...');

  }

}

Файл **messageText.gs**

function message\_chat(TelegramJSON) {//функция для обработки сообщений в  групповом чате

  let text = TelegramJSON.message.text;

  if (!Bot.isNewChatMember()) { //если сообщения не о новом участнике

    if (Bot.isReply()) { //если пересланное сообщение

      let chat\_idReply = TelegramJSON.message.reply\_to\_message.from.id;

      let first\_nameReply = TelegramJSON.message.reply\_to\_message.from.first\_name;

      let last\_nameReply = TelegramJSON.message.reply\_to\_message.from.last\_name;

      let usernameReply = TelegramJSON.message.reply\_to\_message.from.username;

      user = getUserInfo(first\_nameReply, last\_nameReply, usernameReply);

      let usetReply = user.username;

      if (user.username == '') usetReply = user.first\_name + ' ' + user.last\_name;

      if (matchesThanks(text).length !== 0) {

        if (TelegramJSON.message.reply\_to\_message.from.is\_bot) {

          msg = '<b>Приятно, когда меня хвалят!</b>🙈\n<i>Я умею раздавать карму другим😍</i>';

          Bot.sendMessage(msg);

          return

        }

        if (Bot.getUserID() == chat\_idReply) {

          msg = 'Ха... Себе карму хочешь повысить? 😜 Ну-ну...\nПомогай другим в чате и карма быстро вырастит';

          Bot.sendMessage(msg);

          return

        }

        let usetReplyFullName = user.first\_name + user.last\_name;

        msg = `🎉 <b><a href='tg://user?id=${chat\_idReply}'>${usetReply}</a></b> тебе прилетел +1 к карме от <b>${TelegramJSON.message.from.first\_name + TelegramJSON.message.from.last\_name}</b>\nТекущая карма: <b> ${setCarma(usetReplyFullName, chat\_idReply)}</b>`;

        let trophy = getTrophy(chat\_idReply);

        if (trophy != false) msg += '\nТрофей:' + trophy

        Bot.sendMessage(msg);

      }

    }

    if (!Bot.isReply()) { //если непересланное сообщение

      if (matchesThanks(text).length !== 0) {

        msg= `<a href='${Bot.mentionByID()}'>${Bot.getUserFirstName()}</a>, чтобы поблагодарить участника чата, ответь на его сообщение, так я смогу повысить его карму.😌`

        Bot.sendMessage(msg);

        return

      }

    }

  }

  if (Bot.isNewChatMember()) {

    namemsg = `<b>🎉 ${Bot.getUserFullName()}</b>, добро пожаловать`

    let msg = `<b>${Bot.getUserFirstName()}</b>, ты в чате!\nСейчас ты можешь пользоватся только поиском.🔍\n📝 <b>Чтобы писать в чат</b>, необходимо ознакомится с правилами чата.🧾`

    let keyBoard = {

      "inline\_keyboard": [

        [{ "text": "Правила чата", 'callback\_data': 'riding\_rules'}]],

      "resize\_keyboard": true}

    let options = {

      chat\_id: TelegramJSON.message.chat.id,

      user\_id: TelegramJSON.message.from.id,

      parse\_mode: 'HTML',

      disable\_web\_page\_preview: true,

      disable\_notification: true,

      reply\_markup: keyBoard,

      permissions: {

        can\_send\_messages: false,

        can\_invite\_users: true

      },

      until\_date: ~~(Date.now()/100) + 604800

    }

    Bot.restrictChatMember(options.chat\_id, options.user\_id, options.permissions, options.until\_date);

    resp = Bot.sendMessage(msg, options);

    let cache = CacheService.getScriptCache();

    let data = {

      chat\_id: resp.result.chat.id,

      message\_id: resp.result.message\_id,

      text: namemsg + '<b>!</b>'

    }

    cache.put(options.user\_id, JSON.stringify(data), 21600)

  }

if(text === '/rules' || text === '/rules@EduMobile\_bot'){//команда для правил

    Bot.deleteMessage(TelegramJSON.message.message\_id);

    Bot.sendMessage(rules);

}

  if(text === '/top' || text === '/top@EduMobile\_bot'){//команда для топлиста

    let msg = leaderBoard();

    Bot.deleteMessage(TelegramJSON.message.message\_id);

    Bot.sendMessage(msg);

  }

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 \*                                     ОБРАБОТКА СООБЩЕНИЙ В ЛИЧНОМ ЧАТЕ                                              \*

 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

function msgText(TelegramJSON) {

  let text = TelegramJSON.message.text;

  let userProperties = PropertiesService.getUserProperties();

  let user = JSON.parse(userProperties.getProperty(TelegramJSON.message.chat.id));

  if(text === '⭐️' || text === '⭐️⭐️' || text === '⭐️⭐️⭐️' || text === '⭐️⭐️⭐️⭐️' || text === '⭐️⭐️⭐️⭐️⭐️') {

    setSheetVal('users', fan\_row(TelegramJSON.message.chat.id), 8, text)//вставка рейтинга напротив id пользователя

    Bot.sendMessageKeyboardRemove('Спасибо Вам за Ваш рейтинг!☺️');

  }

  if(text && user.callback\_query\_data === 'Добавить расписание') {

    let msg = '✅<b>Расписание успешно добавлено.</b> \n\n<i>Используйте: /menu, чтобы снова перейти в главное меню</i>';

    setSheetVal('Settings', 2, 1, text)

    userProperties.deleteProperty(TelegramJSON.message.chat.id);

    Bot.deleteMessage(TelegramJSON.message.message\_id);

    Bot.deleteMessage(TelegramJSON.message.message\_id-1);//удаление 2-х предыдущих сообщений, для очистки мусора

    Bot.sendMessage(msg);

  }

  if(text && user.callback\_query\_data === 'Добавить предмет') {

    Bot.sendChatAction();

    if(text.length > 30) {

      let msg = '⚠️Название Предмета не должно превышать 30 символов';

      Bot.sendMessage(msg);

    }

    else{

      createSheet(text);

      let msg = '<b>✅Предмет успешно добавлен.</b> \n\n<i>Продолжайте вводить названия предметов для добавления,\n/menu, чтобы снова перейти в главное меню</i>';

      //userProperties.deleteProperty(TelegramJSON.message.chat.id);

      Bot.sendMessage(msg);

    }

  }

  let msgdata = user.callback\_query\_data.split('\_');

  if(text && msgdata[0] == "addm") {//если пришло сообщение в режиме добавления материала для лекций

    let name\_text = text.split('///')[0];

    if (name\_text.length > 30){

      let msg = `<b>⚠️Название материала не должно превышать 30 символов.</b> \n\n<i>Перед основным текстом лекции ограничьте название тремя чертами:\n<b>Например:</b>Лекция1///Текст лекции 1...</i>`;

      Bot.sendMessage(msg);

    }

    else{//если название более 1, но менее 30 символов

      Bot.sendChatAction();

      let sheet = SpreadsheetApp.openById(botSheet).getSheetByName(msgdata[1]);

      sheet.appendRow([name\_text, text.split('///')[1], '', 'Текст'])

      let msg = `✅Метериал в виде текста по предмету: <b>${msgdata[1]}</b> успешно добавлен! \n\n<i>/cancel -  остановить добавление</i>`;

      //userProperties.deleteProperty(TelegramJSON.message.chat.id);

      Bot.sendMessage(msg);

    }

  }

  if(text && msgdata[1] === 'tlang') {//если пришло сообщение после выбора target language

    Bot.sendChatAction();

    try{

      let translatedText = LanguageApp.translate(text, msgdata[0], msgdata[2]); //текст, с какого языка, на какой язык;

      Bot.sendMessage(`<code>${translatedText}</code>\n\n <i>/set\_lang - установить язык заново\n/cancel - остановить перевод</i>`);

      Bot.sendMessage(`Введите текст для перевода. (${msgdata[0]} => ${msgdata[2]})`)

    }

    catch{

      Bot.sendMessage(`⚠️Произошла ошибка перевода. Попробуйте снова или позже.\n\n <i>/cancel - остановить перевод</i>`);

    }

  }

}

Файл **messageMedia.gs**

//отправка фото

function msgPhoto(TelegramJSON) {

  let userProperties = PropertiesService.getUserProperties();

  let user = JSON.parse(userProperties.getProperty(TelegramJSON.message.chat.id));

  let fileid = TelegramJSON.message.photo[2].file\_id;

  let caption = TelegramJSON.message.caption;

  let name\_text = caption.split('///')[0];

  let caption\_text = caption.split('///')[1];

  let msgdata = user.callback\_query\_data.split('\_');

  if(TelegramJSON.message.photo && user.callback\_query\_data.split('\_')[0] === "addm"){

    if (name\_text.length > 30){

      let msg = `⚠️<b>Название материала не должно превышать 30 символов.</b> \n\n<i>Перед основным текстом лекции ограничьте название тремя чертами:\n<b>Например:</b>Лекция1///Текст лекции 1...</i>`;

      Bot.sendMessage(msg);

    }

    if(caption\_text.length > 1024){

      let msg = `⚠️<b>Описание материала не должно превышать 1024 символа.</b> \n\n<i>Если у Вас большой материал: используйте @telegraph для сохрания материала в виде статьи, в сообщении боту пришлите ссылку на материал. Так материал не будет сразу отпугивать по объёму и будет наиболее читаемый в мобильных устройствах</i>`;

      Bot.sendMessage(msg);

    }

    if (name\_text.length <= 30 && caption\_text.length <= 1024){//если название более 1, но менее 30 символов

      let sheet = SpreadsheetApp.openById(botSheet).getSheetByName(msgdata[1]);

      sheet.appendRow([caption.split('///')[0], caption.split('///')[1], fileid, 'Изображение'])

      let msg = `✅Изображение с описанием по предмету: <b>${msgdata[1]}</b> успешно добавлено. \n\n<i>Продолжайте присылать материалы для добавления,\n/menu, чтобы снова перейти в главное меню</i>`;

      //userProperties.deleteProperty(TelegramJSON.message.chat.id);

      Bot.sendMessage(msg);

    }

  }

}

//отправка видео

function msgVideo(TelegramJSON) {

  let userProperties = PropertiesService.getUserProperties();

  let user = JSON.parse(userProperties.getProperty(TelegramJSON.message.chat.id));

  let fileid = TelegramJSON.message.video.file\_id;

  let caption = TelegramJSON.message.caption;

  let name\_text = caption.split('///')[0];

  let caption\_text = caption.split('///')[1];

  let msgdata = user.callback\_query\_data.split('\_');

  if(TelegramJSON.message.video && user.callback\_query\_data.split('\_')[0] === "addm"){

    if (name\_text.length > 30){

      let msg = `⚠️<b>Название материала не должно превышать 30 символов.</b> \n\n<i>Перед основным текстом лекции ограничьте название тремя чертами:\n<b>Например:</b>Лекция1///Текст лекции 1...</i>`;

      Bot.sendMessage(msg);

    }

    if(caption\_text.length > 1024){

      let msg = `⚠️<b>Описание материала не должно превышать 1024 символа.</b> \n\n<i>Если у Вас большой материал: используйте @telegraph для сохрания материала в виде статьи, в сообщении боту пришлите ссылку на материал. Так материал не будет сразу отпугивать по объёму и будет наиболее читаемый в мобильных устройствах</i>`;

      Bot.sendMessage(msg);

    }

    if (name\_text.length <= 30 && caption\_text.length <= 1024){//если название более 1, но менее 30 символов

      let sheet = SpreadsheetApp.openById(botSheet).getSheetByName(msgdata[1]);

      sheet.appendRow([caption.split('///')[0], caption.split('///')[1], fileid, 'Видео']);

      let msg = `✅Видео с описанием по предмету: <b>${msgdata[1]}</b> успешно добавлено. \n\n<i>Продолжайте присылать материалы для добавления,\n/menu, чтобы снова перейти в главное меню</i>`;

      //userProperties.deleteProperty(TelegramJSON.message.chat.id);

      Bot.sendMessage(msg);

    }

  }

}

//отправка аудио

function msgAudio(TelegramJSON) {

  let userProperties = PropertiesService.getUserProperties();

  let user = JSON.parse(userProperties.getProperty(TelegramJSON.message.chat.id));

  let fileid = TelegramJSON.message.audio.file\_id;

  let caption = TelegramJSON.message.caption;

  let name\_text = caption.split('///')[0];

  let caption\_text = caption.split('///')[1];

  let msgdata = user.callback\_query\_data.split('\_');

  if(TelegramJSON.message.audio && user.callback\_query\_data.split('\_')[0] === "addm"){

    if (name\_text.length > 30){

      let msg = `⚠️<b>Название материала не должно превышать 30 символов.</b> \n\n<i>Перед основным текстом лекции ограничьте название тремя чертами:\n<b>Например:</b>Лекция1///Текст лекции 1...</i>`;

      Bot.sendMessage(msg);

    }

    if(caption\_text.length > 1024){

      let msg = `⚠️<b>Описание материала не должно превышать 1024 символа.</b> \n\n<i>Если у Вас большой материал: используйте @telegraph для сохрания материала в виде статьи, в сообщении боту пришлите ссылку на материал. Так материал не будет сразу отпугивать по объёму и будет наиболее читаемый в мобильных устройствах</i>`;

      Bot.sendMessage(msg);

    }

    if (name\_text.length <= 30 && caption\_text.length <= 1024){//если название более 1, но менее 30 символов

      let sheet = SpreadsheetApp.openById(botSheet).getSheetByName(msgdata[1]);

      sheet.appendRow([caption.split('///')[0], caption.split('///')[1], fileid, 'Аудио'])

      let msg = `✅Аудио с описанием по предмету: <b>${msgdata[1]}</b> успешно добавлено. \n\n<i>Продолжайте присылать материалы для добавления,\n/menu, чтобы снова перейти в главное меню</i>`;

      //userProperties.deleteProperty(TelegramJSON.message.chat.id);

      Bot.sendMessage(msg);

    }

  }

}

//отправка документа

function msgDoc(TelegramJSON) {

  let userProperties = PropertiesService.getUserProperties();

  let user = JSON.parse(userProperties.getProperty(TelegramJSON.message.chat.id));

  let fileid = TelegramJSON.message.document.file\_id;

  let caption = TelegramJSON.message.caption;

  let name\_text = caption.split('///')[0];

  let caption\_text = caption.split('///')[1];

  let msgdata = user.callback\_query\_data.split('\_');

  if(TelegramJSON.message.document && user.callback\_query\_data.split('\_')[0] === "addm"){

    if (name\_text.length > 30){

      let msg = `⚠️<b>Название материала не должно превышать 30 символов.</b> \n\n<i>Перед основным текстом лекции ограничьте название тремя чертами:\n<b>Например:</b>Лекция1///Текст лекции 1...</i>`;

      Bot.sendMessage(msg);

    }

    if(caption\_text.length > 1024){

      let msg = `⚠️<b>Описание материала не должно превышать 1024 символа.</b> \n\n<i>Если у Вас большой материал: используйте @telegraph для сохрания материала в виде статьи, в сообщении боту пришлите ссылку на материал. Так материал не будет сразу отпугивать по объёму и будет наиболее читаемый в мобильных устройствах</i>`;

      Bot.sendMessage(msg);

    }

    if (name\_text.length <= 30 && caption\_text.length <= 1024){//если название более 1, но менее 30 символов

      let sheet = SpreadsheetApp.openById(botSheet).getSheetByName(msgdata[1]);

      sheet.appendRow([caption.split('//')[0], caption.split('///')[1], fileid, 'Документ'])

      let msg = `✅Документ с описанием по предмету: <b>${msgdata[1]}</b> успешно добавлен. \n\n<i>Продолжайте присылать материалы для добавления,\n/menu, чтобы снова перейти в главное меню</i>`;

      //userProperties.deleteProperty(TelegramJSON.message.chat.id);

      Bot.sendMessage(msg);

    }

  }

}

//отправка документа

function msgMedia(TelegramJSON) {

  let userProperties = PropertiesService.getUserProperties();

  let user = JSON.parse(userProperties.getProperty(TelegramJSON.message.chat.id));

  let fileid = TelegramJSON.message.document.file\_id;

  let caption = TelegramJSON.message.caption;

  let name\_text = caption.split('///')[0];

  let caption\_text = caption.split('///')[1];

  let msgdata = user.callback\_query\_data.split('\_');

  if(TelegramJSON.message.document && user.callback\_query\_data.split('\_')[0] === "addm"){

    if (name\_text.length > 30){

      let msg = `⚠️<b>Название материала не должно превышать 30 символов.</b> \n\n<i>Перед основным текстом лекции ограничьте название тремя чертами:\n<b>Например:</b>Лекция1///Текст лекции 1...</i>`;

      Bot.sendMessage(msg);

    }

    if(caption\_text.length > 1024){

      let msg = `⚠️<b>Описание материала не должно превышать 1024 символа.</b> \n\n<i>Если у Вас большой материал: используйте @telegraph для сохрания материала в виде статьи, в сообщении боту пришлите ссылку на материал. Так материал не будет сразу отпугивать по объёму и будет наиболее читаемый в мобильных устройствах</i>`;

      Bot.sendMessage(msg);

    }

    if (name\_text.length <= 30 && caption\_text.length <= 1024){//если название более 1, но менее 30 символов

      let sheet = SpreadsheetApp.openById(botSheet).getSheetByName(msgdata[1]);

      sheet.appendRow([caption.split('//')[0], caption.split('///')[1], fileid, 'Документ'])

      let msg = `✅Файлы с описанием по предмету: <b>${msgdata[1]}</b> успешно добавлен. \n\n<i>Продолжайте присылать материалы для добавления,\n/menu, чтобы снова перейти в главное меню</i>`;

      //userProperties.deleteProperty(TelegramJSON.message.chat.id);

      Bot.sendMessage(msg);

    }

  }

}

Файл **debug.gs**

function debug(TelegramJSON) {//json файл последнего действия в листе JSON

  let ss = SpreadsheetApp.openById(botSheet);

  ss.getSheetByName("JSON").getRange(1, 1).setValue(JSON.stringify(TelegramJSON, null, 7));

}

Файл **createKeyBoards.gs**

const menu = [

 [[{'text': '🔥 Меню', 'callback\_data': 'menu\_1'}],

  [{'text': 'ℹ️ Профиль', 'callback\_data': 'profile'}],

  [{'text': '⚙️ Настройки', 'callback\_data': 'Settings'}],

  [{'text': '💳 Donate', 'callback\_data': 'donate'}]],

  [[{'text': '👀 Посмотреть лекции 📖', 'callback\_data': 'Посмотреть лекции'}],

   [{'text': '📚 Список предметов', 'callback\_data': 'courses'}],

   [{'text': '👀 Посмотреть расписание 🗒', 'callback\_data': 'schedule'}],

   [{'text': '🅰️ Переводчик 🈂️', 'callback\_data': 'translate'}],

   [{'text': '🏡 Назад', 'callback\_data': 'menu\_start'}]],

  [[{'text': '🏡 Назад', 'callback\_data': 'menu\_start'}]],

  [[{'text': '🛃 Написать разработчику', 'url': 't.me/freetech\_support'}],

   [{'text': '💡 Examples', 'callback\_data': 'Examples'}],

   [{'text': '⁉️ Как пользоваться?', 'url': 'https://telegra.ph/Kak-polzovatsya-botom-dlya-obrazovaniya-02-23'}],

   [{'text': '🏡 Назад', 'callback\_data': 'menu\_start'}]],

  [[{'text': '👀 Посмотреть лекции 📖', 'callback\_data': 'Посмотреть лекции'}],

   [{'text': '➕ Добавить материал 📝', 'callback\_data': 'Добавить материал'}],

   [{'text': '🗑️ Удалить материал 📝', 'callback\_data': 'Удалить материал'}],

   [{'text': '📚 Список предметов', 'callback\_data': 'courses'}],

   [{'text': '➕ Добавить предмет 📚', 'callback\_data': 'Добавить предмет'}],

   [{'text': '🗑️ Удалить предмет 📚', 'callback\_data': 'Удалить предмет'}],

   [{'text': '👀 Посмотреть расписание 🗒', 'callback\_data': 'schedule'}],

   [{'text': '➕ Добавить расписание 🗒', 'callback\_data': 'Добавить расписание'}],

   [{'text': '🗑️ Удалить расписание 🗒', 'callback\_data': 'Удалить расписание'}],

   [{'text': '🅰️ Переводчик 🈂️', 'callback\_data': 'translate'}],

   [{'text': '🏡 Назад', 'callback\_data': 'menu\_start'}]],

  [[{'text': '📄 Правила чата', 'callback\_data': 'riding\_rules'}]],

  [[{'text': '☑️ Принять', 'callback\_data': 'agree with rules'}]],

  [[{ "text": "✅ С правилами ознакомлен", 'callback\_data': 'pass' }]],

  [[{'text': '🏡 Назад', 'callback\_data': 'menu\_1'}]],

  [[{text: 'RU', callback\_data: 'ru\_sourceL'},{text: 'EN', callback\_data: 'en\_sourceL'},{text: 'DE', callback\_data: 'de\_sourceL'},{text: 'TK', callback\_data: 'tk\_sourceL'}],

   [{text: 'KK', callback\_data: 'kk\_sourceL'},{text: 'KY', callback\_data: 'ky\_sourceL'},{text: 'LV', callback\_data: 'lv\_sourceL'},{text: 'LT', callback\_data: 'lt\_sourceL'}],

   [{text: 'RO', callback\_data: 'ro\_sourceL'},{text: 'SR', callback\_data: 'sr\_sourceL'},{text: 'ES', callback\_data: 'es\_sourceL'},{text: 'FR', callback\_data: 'fr\_sourceL'}],

   [{text: 'SK', callback\_data: 'sk\_sourceL'},{text: 'SL', callback\_data: 'sl\_sourceL'},{text: 'UZ', callback\_data: 'uz\_sourceL'},{text: 'TG', callback\_data: 'tg\_sourceL'}],

   [{text: 'PL', callback\_data: 'pl\_sourceL'},{text: 'UK', callback\_data: 'uk\_sourceL'},{text: 'BG', callback\_data: 'bg\_sourceL'},{text: 'BE', callback\_data: 'be\_sourceL'}],

   [{text: '🏡 Назад', callback\_data: 'menu\_1'}]],

   [[{'text': 'Назад в 💡 Examples', 'callback\_data': 'Examples'}]]

];

//функция создания массива кнопок при просмотре списка материалов

function keySheets(arraykeys, page = "page-0", return\_key = "Посмотреть лекции") {

  //

  let container = [];

  let keyboardLength = 5

  let nameOfSheetsArr = listArray(keyboardLength, arraykeys, container)

  console.log(arraykeys)

  page = +page.substr(-1)

  let keys = [];

  for (i in nameOfSheetsArr[page]) {

    keys.push([{ text: String(nameOfSheetsArr[page][i]), callback\_data: nameOfSheetsArr[page][i] }])

  }

  if (arraykeys.length > keyboardLength) {

    if (page == 0) {

      keys.push([{ text: '⏹', callback\_data: "stop" }, { text: '🏡 Назад', callback\_data: return\_key }, { text: '▶️ ' + (page + 2) + " 📄", callback\_data: "page-" + (page + 1) }])

    }

    else if (page == nameOfSheetsArr.length - 1) {

      keys.push([{ text: "📄 " + (page) + ' ◀️', callback\_data: "page-" + (page - 1) }, { text: '🏡 Назад', callback\_data: return\_key }, { text: '⏹', callback\_data: "stop" }])

    }

    else {

      keys.push([{ text: "📄 " + (page) + ' ◀️', callback\_data: "page-" + (page - 1) }, { text: '🏡 Назад', callback\_data: return\_key }, { text: '▶️ ' + (page + 2) + " 📄", callback\_data: "page-" + (page + 1) }])

    }

  }

  else{//если список кнопок меньше заданного значения 'keyboardlength'

    keys.push([{ text: '🏡 Назад', callback\_data: return\_key }])

  }

  let keyboards = {

    inline\_keyboard: [...keys]

  }

  console.log(JSON.stringify(keyboards))

  let obj = { nameOfSheets: arraykeys, keyboard: keyboards }

  return obj

}

Файл **tools.gs**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 \*                Bot функции               \*

 \* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//функция подписки

function join(chat\_id, name, last\_name, row\_t){

  let row = undefined

  if(row\_t){

    row = row\_t

  }

  else{

    row = fan\_row(chat\_id)

  }

  if(row){

    return 0

  }

  // SpreadsheetApp.openById(botSheet).getSheetByName("Fans list");

  // let lastRow = Sheet.getLastRow();

  let add\_info = [chat\_id, name, last\_name, DATE\_CURRENT]

  setSheetValues("users", add\_info)

  sendText(chat\_id, "✅Подписаны✅")

  return 0

}

//функция отписки

function leave(chat\_id, row\_t){

  let row = undefined

  if(row\_t){

    row = row\_t

  }

  else{

    row = fan\_row(chat\_id)

  }

  if(row){

    let SpreadSheet = SpreadsheetApp.openById(botSheet);

    let Sheet = SpreadSheet.getSheetByName("users");

    Sheet.deleteRow(row)

    Bot.sendMessage("❌Отписаны❌")

  }

  return 0

}

//функция получения фанатской строки

function fan\_row(chat\_id){

  let SpreadSheet = SpreadsheetApp.openById(botSheet);

  let Sheet = SpreadSheet.getSheetByName("users");

  let lastRow = Sheet.getLastRow();

  let start\_row = 2

  for(start\_row=2; start\_row<=lastRow; start\_row++){

    this\_chat\_id = getSheetVal("users", start\_row, \_getItemColInFansList("chat\_id"))

    if(this\_chat\_id == String(chat\_id)){

      return start\_row

    }

  }

  return undefined

}

function \_getItemColInFansList(item){

  if(item == "chat\_id"){

    return 2

  }

  let msg = "[error] Wrong column: "+item

  Logger.log.ERR(msg, "tools.\_getItemColInFansList")

  return -1

}

//функция получения количества подписчиков

function fan\_num(\_chat\_id){

  return SpreadsheetApp.openById(botSheet).getSheetByName("users").getLastRow()-1

}

let msgd = TelegramLibrary.donate\_msg();

//функция отправки количества подписчиков админу или фанатам

function send\_fans\_number(chat\_id){

  let keyboard = undefined

  if(Admins\_UID.indexOf(chat\_id) < 0){

    keyboard = keyboard\_fans

  }

  else{

    keyboard = keyboard\_admins

  }

  sendTextKeyboard(chat\_id, "Количество подписчиков: "+String(fan\_num(chat\_id)), keyboard)

}

function createSheet(sheetname) {

  // sheetname = 'Новый лист'

  let spreadsheet = SpreadsheetApp.openById(botSheet);

  let sheets = spreadsheet.getSheets();

  sheets[sheets.length - 7].activate();

  spreadsheet.insertSheet(sheets.length - 7);

  spreadsheet.getActiveSheet().setName(sheetname);

  spreadsheet.getRange('E:Z').activate();

  spreadsheet.getActiveSheet().deleteColumns(spreadsheet.getActiveRange().getColumn(), spreadsheet.getActiveRange().getNumColumns());

  spreadsheet.getRange('A:D').activate();

  spreadsheet.getActiveSheet().setColumnWidths(1, 4, 148);

  spreadsheet.getActiveSheet().setColumnWidths(1, 4, 210);

  let sheet = spreadsheet.getActiveSheet();

  sheet.getRange(1, 1, sheet.getMaxRows(), sheet.getMaxColumns()).activate();

  spreadsheet.getRange('1:1000').applyRowBanding(SpreadsheetApp.BandingTheme.LIGHT\_GREY);

  let banding = spreadsheet.getRange('A1:D1000').getBandings()[0];

  banding.setHeaderRowColor(null)

  .setFirstRowColor('#ffffff')

  .setSecondRowColor('#f3f3f3')

  .setFooterRowColor(null);

  banding = spreadsheet.getRange('A1:D1000').getBandings()[0];

  banding.setHeaderRowColor(null)

  .setFirstRowColor('#ffffff')

  .setSecondRowColor('#e0f7fa')

  .setFooterRowColor(null);

}

function nameOfSheets() {

  let ss = SpreadsheetApp.openById(botSheet);

  let sheets = ss.getSheets();

  let sheetsName = []

  for (i in sheets) {

    if (i < sheets.length - 7) sheetsName.push(sheets[i].getSheetName());

  }

  console.log(sheetsName)

  return sheetsName

}

function getDataShhet(sheetname) {

  //sheetname = 'лист2'

  try{

    let ss = SpreadsheetApp.openById(botSheet);

    let sheet = ss.getSheetByName(sheetname);

    let data = sheet.getRange(1, 1, sheet.getLastRow()).getValues();

    let lists = []

    for (i in data) {

      lists.push(data[i]);

    }

    list = lists.flat()

    console.log(list)

    return list

  }

  catch{

    return 0

  }

}

function getDataSheet(sheetname) {

  // sheetname = 'Листочек'

  let ss = SpreadsheetApp.getActiveSpreadsheet();

  let sheet = ss.getSheetByName(sheetname);

  let data = sheet.getDataRange().getValues();

  console.log(data)

  return data

}

function addDataSheet(sheetname, data) {

  let ss = SpreadsheetApp.openById(botSheet);

  let sheet = ss.getSheetByName(sheetname);

  sheet.appendRow(data)

}

//функция взятия значений в нужный лист

function getSheetVal(sheetName, row, col){

  let SpreadSheet = SpreadsheetApp.openById(botSheet);

  let Sheet = SpreadSheet.getSheetByName(sheetName);

  return Sheet.getSheetValues(row, col, 1, 1)[0][0]

}

//функция установки значений в нужный лист

function setSheetVal(sheetName, row, col, val){

  let SpreadSheet = SpreadsheetApp.openById(botSheet);

  let Sheet = SpreadSheet.getSheetByName(sheetName);

  Sheet.getRange(row, col).setValue(val)

  return 0

}

//функция добавления строки с контентом из массива

function setSheetValues(sheetName, val){

  let values = val.toString().split(',');

  SpreadsheetApp.openById(botSheet).getSheetByName(sheetName).appendRow(values)

  return 0

}

function delSheet(sheetName) {

  //sheetName = 'лист';

  let spreadsheet = SpreadsheetApp.openById(botSheet);

  spreadsheet.setActiveSheet(spreadsheet.getSheetByName(sheetName), true);

  return spreadsheet.deleteActiveSheet();

}

// функция деления массива (длина элементов, массив который нужно разделить, новый список после деления)

function listArray(num, array, container) {

  for (d = 0; d < array.length; d += num) {

    container.push(array.slice(d, d + num));

  }

  return container

}

function finalCountdown() {

  let token = tgBotToken;

  let chatID = TelegramJSON.message.chat.id;

  let urls = 'https://api.telegram.org/bot' + token +

      '/sendMessage?chat\_id=' + chatID + '&text=It\'s the final countdown!';

  let response = UrlFetchApp.fetch(urls);

  response = JSON.parse(response.getContentText());

  let msgID = response.result.message\_id;

  Utilities.sleep(1000);

  let baseUrl = 'https://api.telegram.org/bot' + token +

      '/editMessageText?chat\_id=' + chatID + '&message\_id=' + msgID + '&text=';

  let w = { '0': "0️⃣", '1': "1️⃣", '2': "2️⃣", '3': "3️⃣", '4': "4️⃣", '5': "5️⃣", '6': "6️⃣", '7': "7️⃣", '8': "8️⃣", '9': "9️⃣", ":": "▪️" }

  let url;

  for (let i = 0; i < 15; ++i) {

      let date = new Date();

      let timestamp = date.getTime();

      let timezone = Session.getScriptTimeZone()

      let times = Utilities.formatDate(date, timezone, "HH:mm:ss");

      url = baseUrl + [...times.toString()].map(e => w[e]).join();

      UrlFetchApp.fetch(url);

      Utilities.sleep(1000);

  }

  url = baseUrl + '💣Boom!';

  UrlFetchApp.fetch(url);

}

//функция поиска строк статей и уроков

function search\_row(text, list){

  //text = "Лекция 1";

  //list = 'лист2';

  let Sheet = SpreadsheetApp.openById(botSheet).getSheetByName(list);

  let lastRow = Sheet.getLastRow();

  let start\_row = 1

  for(start\_row=1; start\_row<=lastRow; start\_row++){

    this\_text = getSheetVal(list, start\_row, 1)

    if(this\_text == String(text)){

      //console.log(start\_row)

      return start\_row

    }

    else undefined

  }

}

function getSheetValue(sheetName, row, col){

  let SpreadSheet = SpreadsheetApp.openById(botSheet);

  let Sheet = SpreadSheet.getSheetByName(sheetName);

  return Sheet.getSheetValues(row, col, 1, 1)

}

let schedule\_msg = getSheetValue("Settings", 2, 2).toString()

let welcome\_msg = getSheetValue("Settings", 5, 2).toString()

let setting\_msg = getSheetValue("Settings", 3, 2).toString()

let privacy\_msg = getSheetValue("Settings", 4, 2).toString()

let version\_msg = 'PROGRAM NAME:' + '1' + '\n\nТекущая версия на' + '<u>' + '1' + '</u>' + ":\n" + '1'

let howto\_msg = '🤓<b>Инструкция по использованию</b>📃<a href="https://telegra.ph/Kak-polzovatsya-botom-dlya-obrazovaniya-02-23">.</a>'

let author\_msg = 'Это бот был создан специально для образования.\n\nАвтор: <b>Куликов Александр</b><a href="https://t.me/kulikov\_mister">.</a> - Магистрант БГПУ им.М.Акмуллы'

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 \*           Функционал для кармы           \*

 \* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

const wsData = SpreadsheetApp.openById(botSheet).getSheetByName('Рейтинг');

const wsSetting = SpreadsheetApp.openById(botSheet).getSheetByName('Settings');

function setCarma(name, chatId) {

  let arrChatId = wsData.getRange(2, 4, wsData.getLastRow()).getValues().flat();

  let idRow = arrChatId.indexOf(+chatId) + 2;

  if (idRow == 1) {

    wsData.appendRow(['', name, 1, chatId]);

    let currentCarma = 1

    return currentCarma

  }

  let currentCarma = wsData.getRange(idRow, 3).getValue();

  currentCarma = currentCarma + 1

  wsData.getRange(idRow, 3).setValue(currentCarma);

  wsData.getRange('B2:D').sort({ column: 3, ascending: false });

  return currentCarma

}

function leaderBoard() {

  let dataTop = wsData.getRange(2, 1, wsData.getLastRow(), 3).getDisplayValues()

  let board = []

  for (let i = 0; i < dataTop.length - 1; i++) {

    board.push(`<code>${dataTop[i][0]}</code> <b>${dataTop[i][1]}: ${dataTop[i][2]}</b>\n`)

  }

  let text = '🏆 <b>Таблица лидеров</b>\n\n' + board.slice(0, 20).join('');

  console.log(text)

  return text

}

function thanks(query) {

  let dataThanks = wsSetting.getRange(2, 1, wsSetting.getLastRow() - 1).getValues().flat();

  return dataThanks.filter(function (el) {

    return el.toLowerCase().indexOf(query.toLowerCase()) > -1;

  })

}

function matchesThanks(query) {

  // query = '+'

  query = query.toLowerCase().replace(/[\s.,№#"'%?!&()<>]/g, ' ').split(' ').filter(Boolean)

  let dataThanks = wsSetting.getRange(2, 1, wsSetting.getLastRow() - 1).getValues().flat();

  let matches = dataThanks.filter(function (item) {

    return query.indexOf(item.toLowerCase()) > -1

  })

  return matches

}

function getTrophy(chatId) {

  let arrChatId = wsData.getRange(2, 4, wsData.getLastRow()).getValues().flat();

  let idRow = arrChatId.indexOf(+chatId) + 2;

  let trophy = false

  if (idRow > 1 && idRow < 5) {

    trophy = wsData.getRange(idRow, 1).getValue();

  }

  return trophy

}

function getUserInfo(first\_name = '', last\_name = '', username = '') {

  let user = {

    first\_name: first\_name,

    last\_name: last\_name == '' ? '' : ' ' + last\_name,

    username: username

  }

  return user

}

Файл **other.gs**

const rules = `

<b>#ПравилаЧата</b>

ℹ️ если Вы подписались на этот чат, Вы автоматически соглашаетесь с правилами.

ℹ️ чат для тех, кто <b>хочет облегчить себе рутиную работу в учебном процессе</b> и <b>ищет обмен опытом и знаниями.</b>

----

⚠️ <b>Пишите вопросы когда:</b>

ℹ️ Вы уже пытались найти ответ в Google или Yandex.

Вопросы типа: "найдите за меня в Google/Yandex" -- игнорируются

ℹ️ Вы <i>пытаетесь решить задачу самостоятельно</i>.

Вопросы типа: "сделайте за меня", "сделайте мне", "дайте мне", "напишите мне" и т.д -- игнорируются

ℹ️ Вы подготовили пример задания.

----

🚫  искать исполнителей,

    что-то рекламировать,

    предлагать свои услуги в чате

🚫 политика | мат | оскорбления

----

⚠️ <b>Бот для обучения: @EduMobile\_bot</b>

ℹ️ Любые благодарственные слова,  в ответ (свайп влево на телефоне) на сообщение участнику чата, повышают его карму.

👍 Не забывайте поблагодарить человека, который вам помог.💪

▶️ Команда: /top — выводит топ-лист самых помогающих.

----

<i>💬 "Навык поиска информации гораздо Важнее самой информации!"</i>

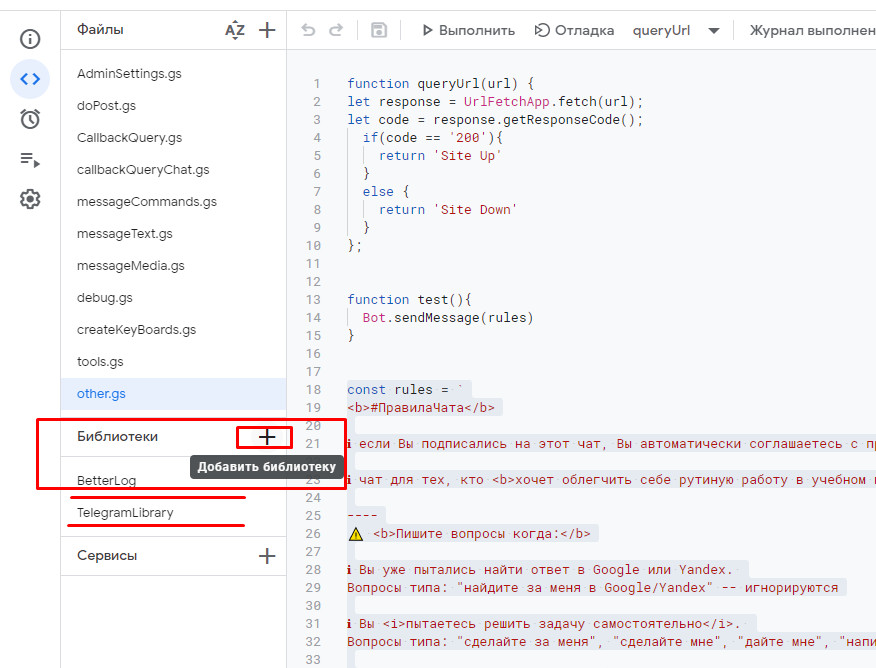
----

Чтобы писать в чат жми на кнопку👇

`;

**ВАЖНО:**

Необходимо также подключить две библиотеки для полноценной работы скрипта.



1. **1DSyxam1ceq72bMHsE6aOVeOl94X78WCwiYPytKi7chlg4x5GqiNXSw0l** – библиотека для отладки кода

(ссылка: <https://github.com/peterherrmann/BetterLog>),

2. **1-IPgOJBfdSpS3LicfDCg8BbA69zJkUIug6T\_p4CA-brdmzSaZQQYO4mI** – библиотека, поддерживающая все методы Telegram Api – можно использовать в разработке Telegram-ботов на Google Apps Script. (автор: Куликов Александр Александрович – БГПУ им. М. Акмуллы)

3. К привязанной Google-таблице добавить следующие листы:



Лист **Examples** – образцы материалов, можно посмотреть в боте

Лист **users** – сохраните своих пользователей

Лист Рейтинг – контролируйте активность в чатах

Настройки из листа **Settings:**

Информация:

**Столбец A** - слова благодарности, на которые ориентируется бот в чате

**Столбец B** – текст сообщений (необязательно к изменению)

**Столбцы D-E** – столбцы с памяткой к форматированию текста в материалах

В остальных листах используются материалы педагога.

