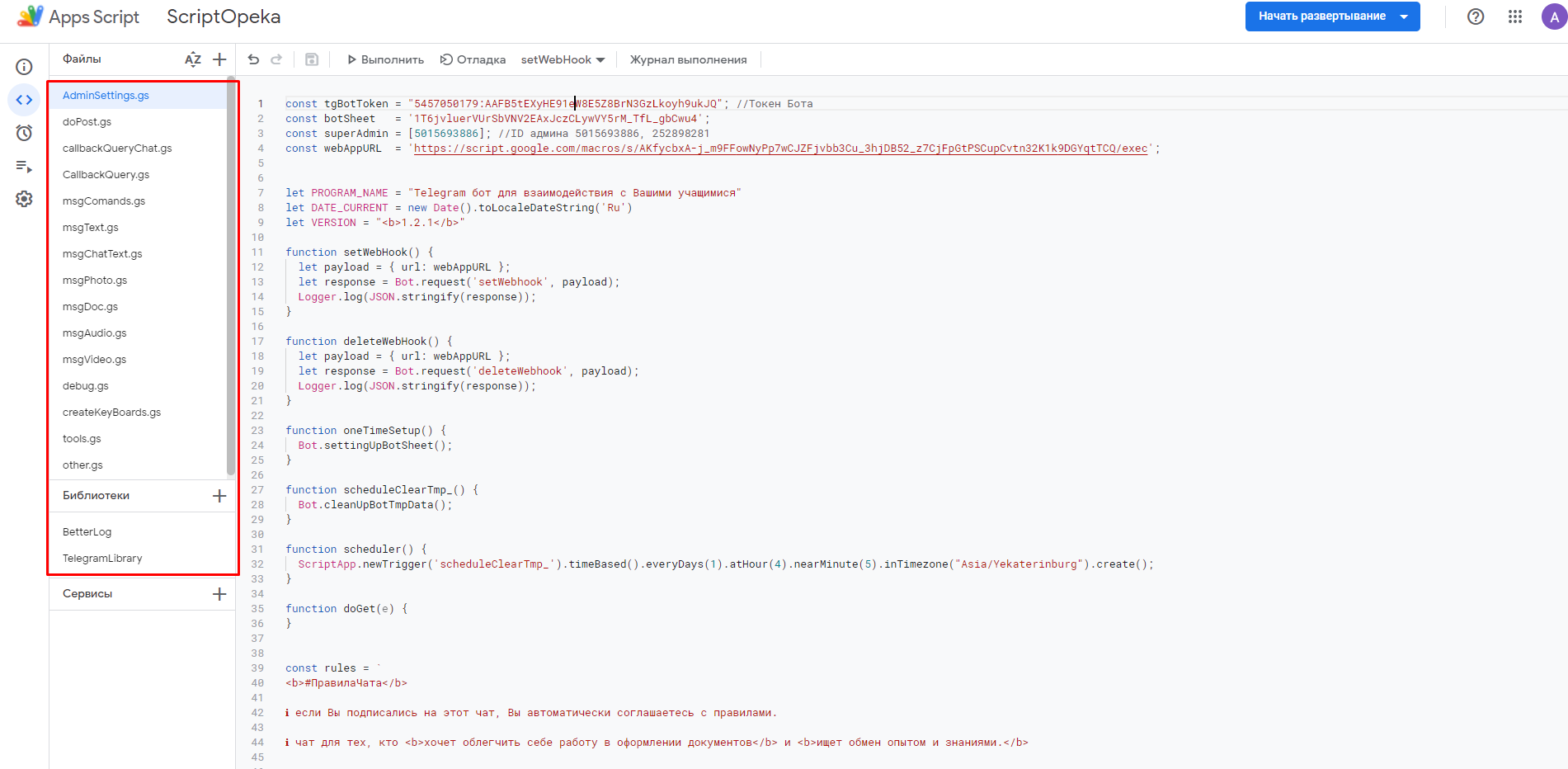
«Информационно-образовательная программа автоматизированного сбора информации в организациях образования».

**Жукенова Г.Б., Куликов А.А.**

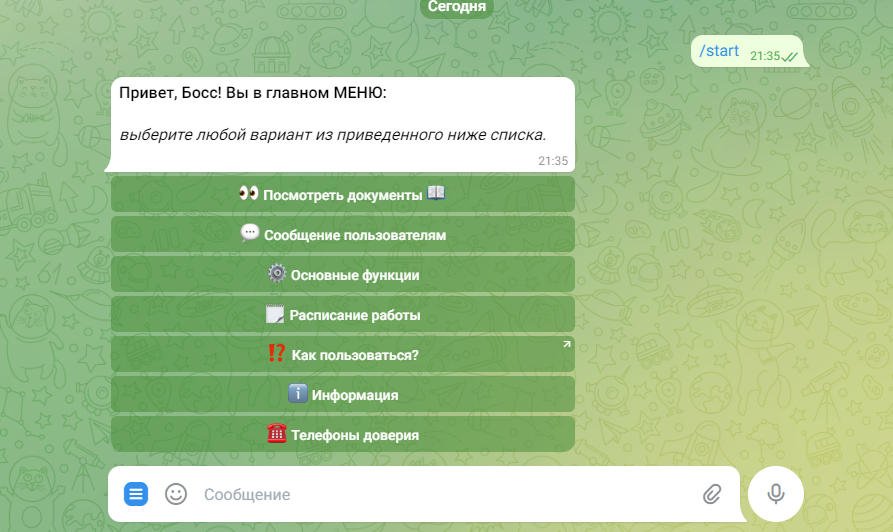
**Применимо:** в любой образовательной или социальной организации, где требуются сборы документов с граждан

**СКРИНШОТЫ:**

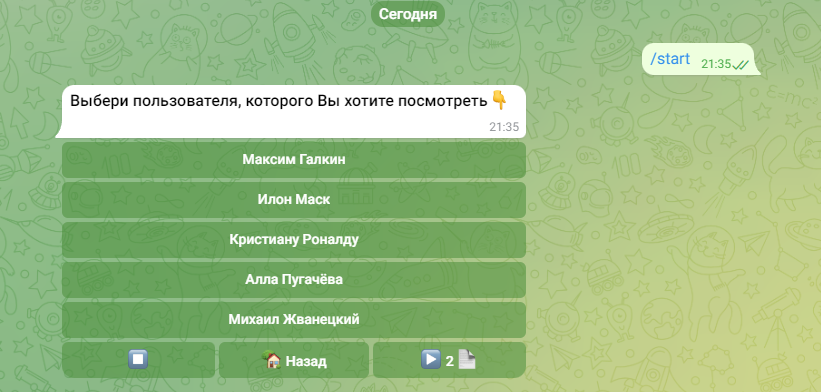
Скриншот 1



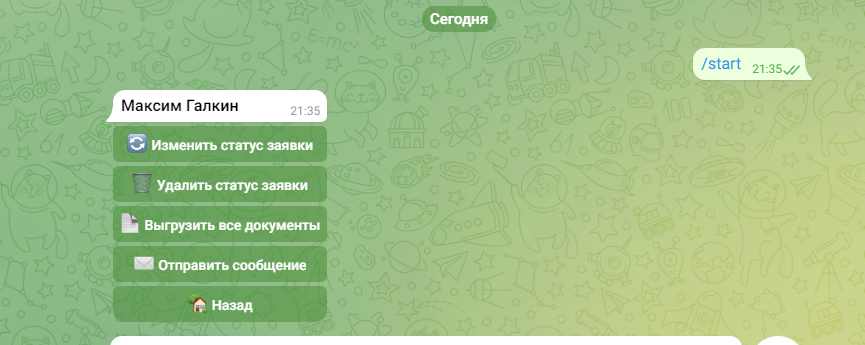
Скриншот 2 – главное меню



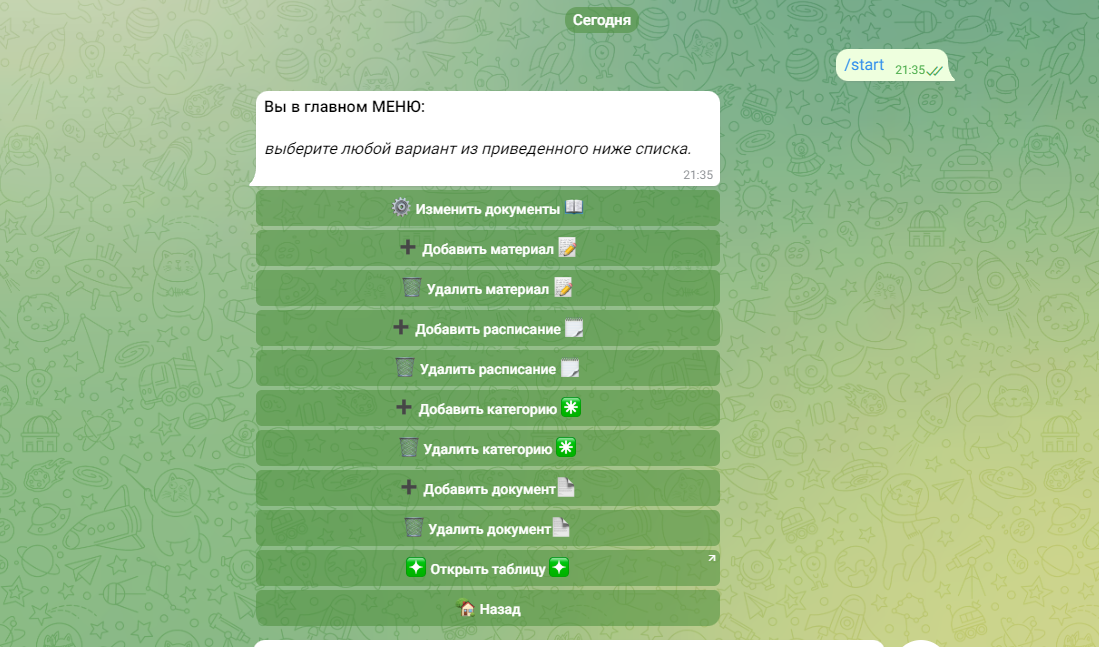
Скриншот 3 – Функция посмотреть документы у пользователя



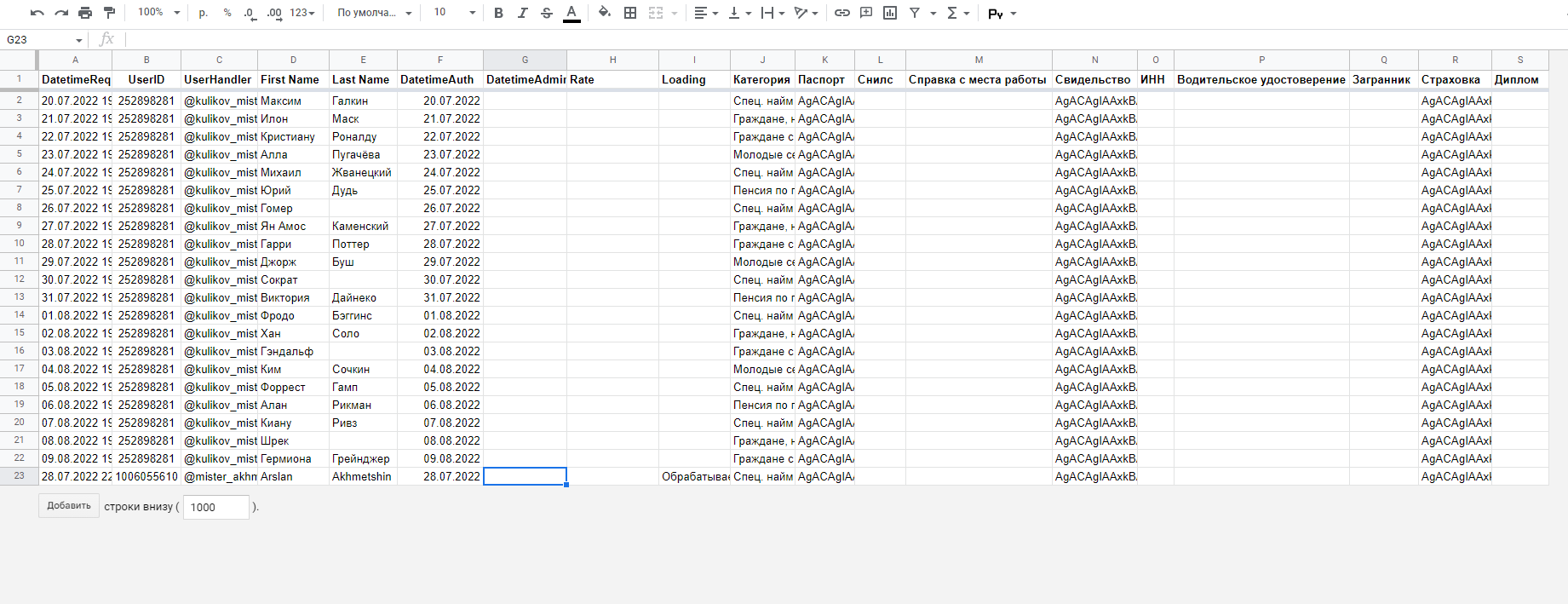
Скриншот 4 - при выборе человека



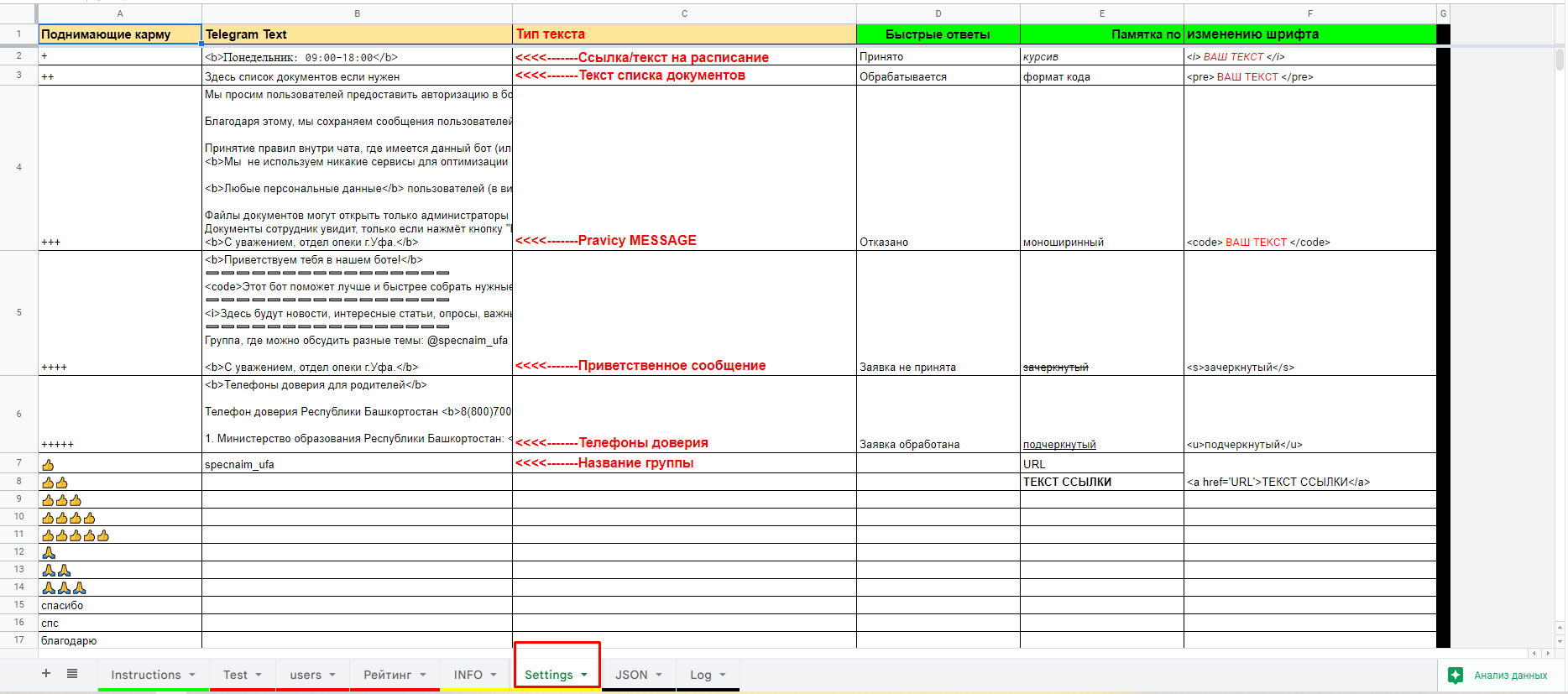
Скриншот 5 – основные функции



Скриншот 6 - вот так данные представлены в таблице



Скриншот 7 – настройки бота можно производить из таблицы



**1. Файл AdminSettings.gs**

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

const botSheet   = '1T6jvluerVUrSbVNV2EAxJczCLywVY5rM\_TfL\_gbCwu4';

const superAdmin = [ID админа]; //ID админа

const webAppURL  = 'https://script.google.com/macros/s/AKfycbxA-j\_m9FFowNyPp7wCJZFjvbb3Cu\_3hjDB52\_z7CjFpGtPSCupCvtn32K1k9DGYqtTCQ/exec';

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

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

let VERSION = "<b>1.2.1</b>"

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 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 работу в оформлении документов</b> и <b>ищет обмен опытом и знаниями.</b>

----

⚠️ <b>Пишите вопросы когда:</b>

ℹ️ Вы уже пытались найти ответ внутри чата.

Вопросы типа: "найдите за меня.." -- 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 прочитали всю информацию внутри бота и не нашли ответа.

----

🚫  искать исполнителей,

    что-то рекламировать,

    предлагать свои услуги в чате

🚫 политика | мат | оскорбления

----

⚠️ <b>Бот для cбора документов: @docs\_opeka\_bot</b>;

ℹ️ О том, как пользоваться ботом, можно прочитать: <b><a href='https://telegra.ph/Kak-polzovatsya-botom-otdela-Opeki-07-18'>▶️Здесь</a></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.

▶️ Команда: /privacy — выводит информацию о том, какие данные используются.

----

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

----

Чтобы писать в чат жми на кнопку👇`;

/\*<b>bold</b>, <strong>bold</strong>

<i>italic</i>, <em>italic</em>

<u>underline</u>, <ins>underline</ins>

<s>strikethrough</s>, <strike>strikethrough</strike>, <del>strikethrough</del>

<span class="tg-spoiler">spoiler</span>, <tg-spoiler>spoiler</tg-spoiler>

<b>bold <i>italic bold <s>italic bold strikethrough <span class="tg-spoiler">italic bold strikethrough spoiler</span></s> <u>underline italic bold</u></i> bold</b>

<a href="http://www.example.com/">inline URL</a>

<a href="tg://user?id=123456789">inline mention of a user</a>

<code>inline fixed-width code</code>

<pre>pre-formatted fixed-width code block</pre>

<pre><code class="language-python">pre-formatted fixed-width code block written in the Python programming language</code></pre>\*/

**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); //дебаг сообщений

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

      callbackQueryChat(TelegramJSON);

      return;

    }

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

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

        msgComands(TelegramJSON);

        return;

      }

    }

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

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

        msgText(TelegramJSON);

        return;

      }

      else {

        msgChatText(TelegramJSON);

        return;

      }

    }

    if (Bot.isNewChatMember() || Bot.isLeftChatMember()) { //если сообщение о новом/удаленеии члене чата/группы

      Bot.deleteMessage(TelegramJSON.message.message\_id); //удаление предыдущего сообщения - очистка мусора

      msgChatText(TelegramJSON);

      return

    }

    if (Bot.isPhoto()) {//если Фото

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

        msgPhoto(TelegramJSON);

        return;

      }

    }

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

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

        msgDoc(TelegramJSON);

        return;

      }

    }

    if (Bot.isAudio()) {//если Аудио

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

        msgAudio(TelegramJSON);

        return;

      }

    }

    if (Bot.isVideo()) {//если Видео

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

        msgVideo(TelegramJSON);

        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': [[{'text': '☑️ Принять', 'callback\_data': `agreeRules\_${TelegramJSON.callback\_query.from.id}`}]]});

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

    return;

  }

  if (options.data.split('\_')[0] == 'agreeRules') {

    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,

      }

    }

    if (options.data.split('\_')[1] == option.user\_id) {

      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[6],"resize\_keyboard": true});

      }

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

      let ind = search\_row(option.user\_id, 'users', 2); //индекс пользователя в базе данных

      if(ind){

        let date = Utilities.formatDate(new Date(), "GMT", "dd.MM.yyyy")

        let val = [date,option.user\_id,'@'+TelegramJSON.callback\_query.from.username,TelegramJSON.callback\_query.from.first\_name,TelegramJSON.callback\_query.from.last\_name,date]

        setSheetValues('users', val);

      }

      answer = {

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

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

        show\_alert: true

      }

      Bot.request('answerCallbackQuery',answer);

      return;

    }

    else{

      Bot.request('answerCallbackQuery', { callback\_query\_id: TelegramJSON.callback\_query.id, text: 'Это не ваша Кнопка', show\_alert: true });

      return;

    }

  }

  if (options.data == 'pass') {

    try{

      Bot.request('answerCallbackQuery',answer);

    }

    catch{

      Bot.request('answerCallbackQuery', { callback\_query\_id: TelegramJSON.callback\_query.id, text: '✅ Теперь можешь писать в чат 📝', show\_alert: true });

    }

    return;

  }

  else{//остальные нажатия кнопок отрабатываются внутри бота

    return callbackQuery(TelegramJSON);

  }

}

**Файл callbackQuery.gs**

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

  let cb = TelegramJSON.callback\_query;

  let options = {

    chat\_id: cb.message.chat.id,

    message\_id: cb.message.message\_id,

    data: cb.data,

    parse\_mode: 'HTML',

    disable\_web\_page\_preview: true,

    disable\_notification: true,

    text: '' || cb.message.text,

    reply\_markup: '',

  }

  let userProperties = PropertiesService.getUserProperties();

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

  let msg, msg2 = '';

  if (Bot.getSystemUser() && Bot.getSystemUser().isAuth || superAdmin.includes(options.chat\_id)) { // остальные сообщения для авторизованных пользователей

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

      userProperties.deleteProperty(options.chat\_id);

      if (Bot.getSystemUser().isAdmin || superAdmin.includes(options.chat\_id)) {//если админ возвращается в меню

        Bot.sendChatAction();

        msg = "<b>Привет, Босс! 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[2]});

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

      }

      else{

        let ind = search\_row(options.chat\_id, 'users', 2); //индекс пользователя в базе данных

        Bot.sendChatAction();

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

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

        if (getSheetVal('users', ind, 9) === '') {// если статус заявки пустой, показывается полная клавиатура

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

        }

        else{

          Bot.editMessageKeyboard(msg, options.message\_id, null, {'inline\_keyboard': menu[0].slice(2)});

        }

      }

      return;

    }

    if (options.data === 'stop') {

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

      return;

    }

    if (options.data === 'Hotlines'){

      Bot.sendChatAction();

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

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

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

      }

      else{

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

      }

      return;

    }

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

      Bot.sendChatAction();

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

      Bot.editMessageKeyboard('<b>ℹ️ Информация</b>', options.message\_id, null, {"inline\_keyboard": menu[3] });

      return;

    }

    if (options.data === 'MoreInfo'){ //Доп информация

      Bot.sendChatAction();

      if (Bot.getSystemUser().isAdmin || superAdmin.includes(options.chat\_id)) {

        try{

          Bot.editMessageKeyboard(getInfoValues(), options.message\_id, null, {"inline\_keyboard": [[{text: '🏡 Назад', callback\_data: 'Information'}]] });

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

        }

        catch{

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

        }

        return;

      }

      else{

        try{

          Bot.editMessageKeyboard(getInfoValues(), options.message\_id, null, {"inline\_keyboard": [[{text: '🏡 Назад', callback\_data: 'menu\_start'}]] });

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

        }

        catch{

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

        }

        return;

      }

    }

    if (options.data === 'Articles'){ // архив статей

      Bot.sendChatAction();

      try{

        Bot.editMessageKeyboard('🗂️ <b>Архив статей</b>\n'+getInfoArticles(), options.message\_id, null, {"inline\_keyboard": [[{text: '🏡 Назад', callback\_data: 'Information'}]] });

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

      }

      catch{

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

      }

      return;

    }

    if (options.data === 'schedule'){ //расписание работы

      Bot.sendChatAction();

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

        msg = 'Расписание работы🗒';

        try{

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

        }

        catch{

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

        }

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

      }

      else{

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

      }

      return;

    }

    if (options.data === 'loading') { //Пользователь хочет проверить состояние заявки

      Bot.sendChatAction();

      let ind = search\_row(options.chat\_id, 'users', 2)

      if (ind == undefined){ //если пользователь не зарегистрирован

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

      }

      else{

        let data = getSheetValue('users', ind, 9);

        if (data != ''){ //если состояние заявки присутствует

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

          Bot.editMessageKeyboard(`<b>Cостояние заявки:</b> ${String(data)}`, options.message\_id, null, {'inline\_keyboard': menu[1]});

        }

        else{ //если состояние заявки отсутствует

          Bot.request('answerCallbackQuery', { callback\_query\_id: cb.id, text: '⚠️ Документов на проверку не отправлялось!'});

        }

      }

      return;

    }

    if (options.data === 'sendWait') {//Пользователь хочет отправить заявку

      Bot.sendChatAction();

      let ind = search\_row(options.chat\_id, 'users', 2); //индекс пользователя в базе данных

      if (ind == undefined){ //если пользователь не зарегистрирован

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

      }

      else{

        if (getSheetVal('users', ind, 9) === '') {//если заявка не отправлялась

          let category = getSheetVal('users', ind, 10);//категория пользователя - название

          if (category){

            let indCategory = search\_row(category, 'Test', 1);

            let a = Sheet('users').getRange(1, 11, 1, Sheet('users').getLastColumn()-10).getValues().flat();//названия категорий

            let b = Sheet('Test').getRange(indCategory, 2, 1, Sheet('Test').getLastColumn()-1).getValues().flat();//список обязательных документов

            let c = Sheet('users').getRange(ind, 11, 1, Sheet('users').getLastColumn()-10).getValues().flat();//массив с данными пользователя

            const categoriesArr = b //новый список обязательных документов

              .filter(document => document.includes('✅'))

              .map(document => document.slice(2))

            let arr = [];

            for (let i = 0; i < c.length; i++) {//список имеющихся документов у пользователя

              if (c[i]){ //если нет какого-то документа

                arr.push(a[i]);

              }

            }

            const diff = function (a1, a2) {//сравнение двух массивов

              return a1.filter(doc => !a2.includes(doc))

            }

            const diffCategories = diff(categoriesArr, arr);//список с недостающими документами

            if (diffCategories.length) {

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

              msg = `<b>Внимание:</b> после отправки документов на проверку, возможность изменять документы будет заблокирована.\n\n<i>Если согласны, нажмите:</i> <b>Продолжить</b>`;

              Bot.editMessageKeyboard(msg, options.message\_id, null, {'inline\_keyboard': menu[10]});

            }

            else{

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

              msg = `<b>Внимание:</b> после отправки документов на проверку, возможность изменять документы будет заблокирована.\n\n<i>Если согласны, нажмите:</i> <b>Продолжить</b>`;

              Bot.editMessageKeyboard(msg, options.message\_id, null, {'inline\_keyboard': menu[11]});

            }

          }

          else{

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

            msg = `<b>Прежде, чем отправить заявку для проверки документов, необходимо:\n\n1.</b> Выбрать категорию, к которой Вы принадлежите;\n<b>2.</b> Загрузить все необходимые документы;\n<b>3.</b> Повторно отправить заявку на проверку.`

            keys = [[{'text': '✳️ Установить категорию', 'callback\_data': `newCategory`}],

                    [{'text': '📨 Отправить документы', 'callback\_data': 'Отправить документы'}],

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

            Bot.editMessageKeyboard(msg, options.message\_id, null, {'inline\_keyboard': keys});

          }

        }

      return;

      }

    }

    if (options.data === 'waiting') { //отправка заявки

      Bot.sendChatAction();

      let ind = search\_row(options.chat\_id, 'users', 2); //индекс пользователя в базе данных

        if (getSheetVal('users', ind, 9) === '') {//если заявка не отправлялась

          let category = getSheetVal('users', ind, 10);//категория пользователя - название

          if (category){

            let indCategory = search\_row(category, 'Test', 1);

            let a = Sheet('users').getRange(1, 11, 1, Sheet('users').getLastColumn()-10).getValues().flat();//названия категорий

            let b = Sheet('Test').getRange(indCategory, 2, 1, Sheet('Test').getLastColumn()-1).getValues().flat();//список обязательных документов

            let c = Sheet('users').getRange(ind, 11, 1, Sheet('users').getLastColumn()-10).getValues().flat();//массив с данными пользователя

            const categoriesArr = b //новый список обязательных документов

              .filter(document => document.includes('✅'))

              .map(document => document.slice(2))

            let arr = [];

            for (let i = 0; i < c.length; i++) {//список имеющихся документов у пользователя

              if (c[i]){ //если нет какого-то документа

                arr.push(a[i]);

              }

            }

            const diff = function (a1, a2) {//сравнение двух массивов

              return a1.filter(doc => !a2.includes(doc))

            }

            const diffCategories = diff(categoriesArr, arr);//список с недостающими документами

            if (diffCategories.length) {//если имеются недостающие документы

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

              msg = '<b>Загрузите следующие документы:</b> \n'

              for (let i = 0; i < diffCategories.length; i++) {msg += `<b>${i+1}.</b> ${diffCategories[i]}\n`};//формирование сообщения

              Bot.editMessageKeyboard(msg, options.message\_id, null, {'inline\_keyboard': menu[9]});

            }

            else{

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

              setSheetVal('users', ind, 9, 'Заявка отправлена');

              msg = `<b>✅ Ваша заявка отправлена!</b><i>\nКогда статус изменится, Вам придёт сообщение!</i>`;

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

              msg2 = "<b>Босс, новая заявка от " + "<a href='" + Bot.mentionByID() +"'>" + Bot.getUserFullName() + "</a></b>";

              let key = [[{'text': '🔄 Изменить статус заявки', 'callback\_data': `setStatus\_${options.chat\_id}` }],

                        [{'text': '📄 Выгрузить все документы', 'callback\_data': `DownloadDocs\_${options.chat\_id}` }],

                        [{'text': '✉️ Отправить сообщение', 'callback\_data': `sendmsg\_${options.chat\_id}`}]];

              Bot.sendMessageTo(superAdmin[0], msg2, {'reply\_markup': {'inline\_keyboard': key}});

            }

          }

          else{

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

            msg = `<b>Прежде, чем отправить заявку для проверки документов, необходимо:\n\n1.</b> Выбрать категорию, к которой Вы принадлежите;\n<b>2.</b> Загрузить все необходимые документы;\n<b>3.</b> Повторно отправить заявку на проверку.`

            keys = [[{'text': '✳️ Установить категорию', 'callback\_data': `newCategory`}],

                    [{'text': '📨 Отправить документы', 'callback\_data': 'Отправить документы'}],

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

            Bot.editMessageKeyboard(msg, options.message\_id, null, {'inline\_keyboard': keys});

          }

        }

        else{//если заявка уже отправлялась ранее

          Bot.request('answerCallbackQuery', { callback\_query\_id: cb.id, text: '⚠️ Заявка уже отправлялась ранее!'});

          msg = `<b>⚠️ Заявка уже отправлялась ранее!</b> \n<i>Будьте терпеливы, Ваша заявка уже на рассмотрении.\nКогда статус изменится, Вам придёт сообщение!</i>`;

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

        }

      return;

    }

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

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

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

      let data = getDataColumnSheet('Instructions');

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

      try{

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

        return;

      }

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

        Bot.deleteMessage(options.message\_id);

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

        return;

      }

    }

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

      let user = {callback\_query\_data: options.data};

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

        let data = getDataUsers();

        if(data.length > 0){//проверка на наличие пользователей

          msg = msg = '<b>Выбери пользователя, которого Вы хотите посмотреть</b>👇';

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

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

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

        }

        else{

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

        }

        return;

      }

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

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

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

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

      }

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

        let data = getDataColumnSheet('Instructions');

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

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

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

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

      }

      else{

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

      }

      return;

    }

    if (Bot.getUsersID().includes(+options.data)) {//если выбран юзер из списка

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

      let keys = [[{'text': '🔄 Изменить статус заявки', 'callback\_data': `setStatus\_${+options.data}` }],

                  [{'text': '🗑️ Удалить статус заявки', 'callback\_data': `delStatus\_${+options.data}` }],

                  [{'text': '📄 Выгрузить все документы', 'callback\_data': `DownloadDocs\_${+options.data}` }],

                  [{'text': '✉️ Отправить сообщение', 'callback\_data': `sendmsg\_${options.data}`}],

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

      msg = Bot.getSystemUser(options.data).firstName +' '+ Bot.getSystemUser(options.data).lastName

      Bot.editMessageKeyboard(`<b>${msg}</b>` , options.message\_id, null, {'inline\_keyboard': keys});

      return;

    }

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

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

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

      Bot.editMessageText('<b>Напишите название для Вашей новой категории.</b>', options.message\_id);

      return;

    }

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

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

      if (getSheetVal('Test', 1, 1)) {

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

        let categories = getDataColumnSheet('Test');

        msg = '<b>Выберите категорию, которую Вы хотите удалить</b>👇'

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

        return;

      }

      else{

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

        return;

      }

    }

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

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

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

      Bot.editMessageText('<b>Напишите название нового документа.</b>', options.message\_id);

      return;

    }

    if (options.data.split('\_')[0] === 'sendmsg') { //отпрака сообщения от имени бота пользователю

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

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

      msg = '<b>Напишите Ваше сообщение, которое Вы хотите отправить пользовтателю.</b>👇';

      Bot.sendMessage(msg);

      return;

    }

    if (options.data === 'sendmsgAdmin') { //отпрака сообщения администратору

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

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

      msg = '<b>Напишите Ваше сообщение, которое Вы хотите отправить администратору.</b>👇';

      Bot.editMessageKeyboard( msg, options.message\_id, null, {'inline\_keyboard': [[{'text': '🏡 Назад', 'callback\_data': 'menu\_start'}]]});

      return;

    }

    if (options.data === 'sendAllmsg') { //отпрака сообщения от имени бота всем пользователям

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

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

      msg = '<b>Напишите Ваше сообщение, которое Вы хотите отправить всем пользовтателям.</b>👇';

      Bot.editMessageKeyboard( msg, options.message\_id, null, {'inline\_keyboard': [[{'text': '🏡 Назад', 'callback\_data': 'sendmsg'}]]});

      return;

    }

    if (options.data === 'sendAllmsgTo') { //отпрака сообщения от имени бота всем пользователям

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

      let categories = getDataColumnSheet('Test');

      msg = '<b>Выберите категорию, которой Вы хотите отправить Ваше сообщение.</b>';

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

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

      return;

    }

              /\*                          АДМИНИСТРАТОРСКИЕ ФУНКЦИИ                           \*/

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

      Bot.sendChatAction();

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

      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[8]});

      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 расписание или прямую ссылку на Ваше расписание</b>', options.message\_id, null, {'inline\_keyboard': menu[1]});

      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!'});

      }

      else {

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

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

      }

      return;

    }

    if (options.data.includes('setStatus')) {

      Bot.sendChatAction();

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

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

      msg = '<b>Босс, выберите любой вариант из приведенного ниже списка.</b>';

      keys = getDataColumnSheet('Settings', row=2,col=4);

      Bot.sendMessageCustomKeyboard(msg, keys, 'Босс, выбери статус!');

      return;

    }

    if (options.data.includes('delStatus')) {

      Bot.sendChatAction();

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

      let ind = search\_row(userData.callback\_query\_data.split('\_')[1], 'users', 2);

      msg = '✅<b>Cтатус успешно изменён.</b> \n\n<i>Используйте: /start, чтобы снова перейти в главное меню</i>';

      Bot.sendMessage(msg);

      setSheetVal('users', ind, 9, '');

      msg2 = `✉️<b>Статус Вашей заявки обновлён:</b> \n<i>Текущий статус:</i> <b>${text}\n\nС уважением, отдел опеки г.Уфа</b>`;

      Bot.sendMessageTo(userData.callback\_query\_data.split('\_')[1], msg2);

      return;

    }

    if (options.data.includes('newCategory')) {

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

      userProperties.setProperty(options.chat\_id, JSON.stringify({callback\_query\_data: 'setCategory'}));

      msg = '<b>Установите категорию из приведённого ниже списка👇</b>';

      let categories = getDataColumnSheet('Test');

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

      return;

    }

    if (options.data === 'editDocs') { //если нажата кнопка для изменения документов по умолчанию

      Bot.sendChatAction();

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

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

      let categories = getDataColumnSheet('Test');//категории

      msg = '<b>Выберите категорию, в которой Вы хотите изменить документы</b>👇'

      Bot.editMessageKeyboard( msg, options.message\_id, null, getkeys(categories, 2, return\_key = "Functions"));

      return;

    }

    if (options.data === 'editDocsSendAll') {//отправка уведомления пользователям об изменении категории

      Bot.sendChatAction();

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

      let category = userData.callback\_query\_data.split('\_')[1];

      let users = getDataColumnSheet('users', 2, 2);

      let ind = search\_row(category, 'Test', 1); //строка сохранённой категории

      let docs = getDataRowSheet('Test', ind, 2).filter(document => document.includes('✅'));//список обязательных документов

      let txt = `⚠️ Внимание, список документов в категории:<b> ${category}</b> был изменён!\n\n<b>Текущий список:</b>\n`;

      for (i in docs) {txt+=`${docs[i]}\n`};

      for (i in users) {

        Bot.sendMessageTo(users[i], txt);

      }

      msg2 = `Проинформировано: <b>${+i+1}</b> из <b>${users.length}</b>`;

      Bot.sendMessage(msg2);

      return;

    }

    if (options.data >= 0) {//изменение документов по умолчанию

      Bot.sendChatAction();

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

      if (userData.callback\_query\_data.includes('editDocs') ) {

        let ind = search\_row(userData.callback\_query\_data.split('\_')[1], 'Test', 1); //Cтрока выбранной категории

        let docs = getDataRowSheet('Test', ind, 2);

        if (docs[options.data].split(' ')[0] == '✅') {

          msg = docs[options.data].replace(/✅/, '➖')

        }

        else {

          msg = docs[options.data].replace(/➖/, '✅')

        }

        setSheetVal('Test', ind, +options.data+2, msg);

        docs = getDataRowSheet('Test', ind, 2);//обновление переменной docs

        Bot.editMessageKeyboard(`Ваш Выбор: <b>${msg}</b> сохранён!` , options.message\_id, null,  getkeys(docs, 1, "Functions", `editDocsSendAll`));

        return;

      }

    }

    if (getDataColumnSheet('Test').includes(options.data)) { //если выбрана категория

      Bot.sendChatAction();

      if (userData.callback\_query\_data === 'editDocs') { //если выбрана категория для изменения документов по умолчанию

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

        let ind = search\_row(options.data, 'Test', 1); //Cтрока выбранной категории

        let docs = getDataRowSheet('Test', ind, 2);

        msg = '<b>Выберите документ, который Вы хотите изменить</b>👇'

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

        Bot.editMessageKeyboard( msg, options.message\_id, null, getkeys(docs, 1, "Functions", `editDocsSendAll`));

        return;

      }

      if (userData.callback\_query\_data === 'sendAllmsgTo') { //если выбрана категория для отправки сообщения категории пользователей

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

        msg = '<b>Напишите Ваше сообщение, которое Вы хотите отправить пользовтателям.</b>👇';

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

        Bot.editMessageKeyboard( msg, options.message\_id, null, {'inline\_keyboard': [[{'text': '🏡 Назад', 'callback\_data': 'sendmsg'}]]});

        return;

      }

      if (userData.callback\_query\_data === 'Удалить категорию') { //если выбрана категория для удаления

        try{

          Sheet('Test').deleteRow(search\_row(options.data, 'Test', 1));

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

          let categories = getDataColumnSheet('Test');

          msg = '<b>Выберите категорию, которую Вы хотите удалить</b>👇'

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

        }

        catch{

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

        }

        return;

      }

      if (userData.callback\_query\_data === 'setCategory') { //если выбрана категория для установки

        if (!superAdmin.includes(options.chat\_id)){

          try {

            let ur = search\_row(options.chat\_id, 'users', 2);

            setSheetVal('users', ur, 10, options.data); //установка категории

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

            userProperties.setProperty(options.chat\_id, JSON.stringify({callback\_query\_data: 'Отправить документы'}));

            msg = '<b>Выберите документ из приведённого ниже списка👇</b>';

            Bot.editMessageKeyboard(msg, options.message\_id, null, getkeyFlags(options.chat\_id, return\_key = "menu\_start"));

            return

          }

          catch{

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

            return;

          }

        }

        return;

      }

    }

     /\*                          РАБОТА С ДОКУМЕНТАМИ                           \*/

    if (options.data === 'Отправить документы') { //отправка документов

      let ind = search\_row(options.chat\_id, 'users', 2);//строка пользователя в таблице

      if (getSheetVal('users', ind, 9) !== '') {//если заявка уже отправлялась

        Bot.sendChatAction();

        Bot.request('answerCallbackQuery', { callback\_query\_id: cb.id, text: '⚠️ Отправка документов запрещена' });

        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].slice(2)});

      }

      else{

        if (getSheetVal('users', ind, 10) !== '') {//если категория граждан в таблице имеется

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

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

          msg = '<b>Выберите документ из приведённого ниже списка👇</b>';

          Bot.editMessageKeyboard(msg, options.message\_id, null, getkeyFlags(options.chat\_id, return\_key = "menu\_start"));

          return;

        }

        else {//если пользователь не выбрал категорию

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

          userProperties.setProperty(options.chat\_id, JSON.stringify({callback\_query\_data: 'setCategory'}));

          msg = '<b>Установите категорию из приведённого ниже списка👇</b>';

          let categories = getDataColumnSheet('Test');

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

          return;

        }

      }

    }

    let passlist = Sheet('Users').getRange(1, 11, 1, Sheet('Users').getLastColumn()-10).getValues().flat();

    if (passlist.includes(options.data)) { //если выбран документ из списка

      if (userData.callback\_query\_data === 'Отправить документы') {

        Bot.request('answerCallbackQuery', { callback\_query\_id: cb.id, text: `Отправьте: ${options.data}`});

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

        let ur = search\_row(options.chat\_id, 'users', 2);//строка пользователя

        let dcol = search\_col(options.data, 'users', 1);//колонка надйенного документа

        if (getSheetVal('users', ur, dcol) === '') {//если документ уже имеется в списке

          msg = '<b>Пришлите Ваш документ в ответном сообщении.</b> \n\n<i>Поддерживаются следующие форматы</i> \n <b>| PDF | DOC | JPEG | JPG | PNG |</b>';

          let key = [[{ text: '🏡 Назад', callback\_data: 'Отправить документы'}]];

          Bot.editMessageKeyboard(msg, options.message\_id, null, {'inline\_keyboard': key });

          return;

        }

        else{

          msg = '<b>Данный документ уже есть в системе.</b> \nЧтобы заменить документ, пришлите Ваш новый документ в ответном сообщении. \n\n<i>Поддерживаются следующие форматы</i> \n <b>| PDF | DOC | JPEG | JPG | PNG |</b>';

          let key = [

            [{ text: 'Удалить документ', callback\_data: `deleteDoc\_${options.data}`}],

            [{ text: '🏡 Назад', callback\_data: 'Отправить документы'}]

          ]

          Bot.editMessageKeyboard(msg, options.message\_id, null, {'inline\_keyboard': key });

          return;

        }

      }

    }

    if (options.data.includes('deleteDoc')) { //удаление документа из меню пользователя, до подачи заявки

      let ur = search\_row(options.chat\_id, 'users', 2);

      let dcol = search\_col(options.data.split('\_')[1], 'users', 1);

      setSheetVal('users', ur, dcol, null);

      Bot.request('answerCallbackQuery', { callback\_query\_id: cb.id, text: `✅Документ успешно удалён.` });

      Bot.deleteMessage(options.message\_id);

      msg = `✅Документ: <b>${userData.callback\_query\_data.split('\_')[1]}</b> успешно удалён.`;

      Bot.sendMessage(msg);

      msg2 = '<b>Выберите документ из приведённого ниже списка👇</b>';

      userProperties.setProperty(options.chat\_id, JSON.stringify({callback\_query\_data: 'Отправить документы'}));

      Bot.sendMessage(msg2, {'reply\_markup': getkeyFlags(options.chat\_id, return\_key = "menu\_start")});

      return;

    }

    if (options.data === 'delete'){

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

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

      return;

    }

    if (options.data.includes('DownloadDocs')) { //выгрузка всех документов пользователя

      Bot.sendChatAction();

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

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

      let ind = search\_row(userID, 'users', 2); //индекс пользователя в базе данных

      let usersDocs = Sheet('users').getRange(ind, 11, 1, Sheet('Users').getLastColumn()-10).getValues().flat();

      let data1 = Sheet('users').getRange(1, 11, 1, Sheet('Users').getLastColumn()-10).getValues().flat();

      Bot.sendMessage("<b>Начало выгрузки для <a href='tg://user?id=" + userID +"'>" + [Bot.getSystemUser(userID).firstName +' '+Bot.getSystemUser(userID).lastName] + "</a></b>");

      for (let i = 0; i < usersDocs.length; i++) {

        if (usersDocs[i]) {

          msg = "<b>" + data1[i] + "</b> <a href='tg://user?id=" + userID +"'><b>" + [Bot.getSystemUser(userID).firstName +' '+Bot.getSystemUser(userID).lastName] + "</b></a>";

          try{

            Bot.sendPhoto(msg, usersDocs[i]);

          }

          catch{

            Bot.sendDocument(msg, usersDocs[i]);

          }

        }

      }

      Bot.sendMessage("<b>Выгрузка завершена для <a href='tg://user?id=" + userID +"'>" + [Bot.getSystemUser(userID).firstName +' '+Bot.getSystemUser(userID).lastName] + "</a></b>");

      return;

    }

                 /\*                          ДОПОЛНИТЕЛЬНЫЕ ФУНКЦИИ                           \*/

    if (getDataColumnSheet('Instructions').includes(options.data)) {//если пользователь выбрал материал в боте

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

      if (userData.callback\_query\_data === 'Удалить материал') {//если пользователь желает удалить материал

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

        Sheet("Instructions").deleteRow(search\_row(options.data, "Instructions", 1));

        let data = getDataColumnSheet("Instructions");

        msg = 'Выбери материал, который Вы хотите <b>удалить</b>👇';

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

        return;

      }

      if(userData.callback\_query\_data === 'Instructions'){

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

        let row = search\_row(options.data, 'Instructions', 1);

        let lastpost = Sheet('Instructions').getRange(row, 1, 1,  4).getValues()[0];

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

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

        if (lastpost[3] == 'Текст') {

          Bot.editMessageKeyboard(msg, options.message\_id, null, {'inline\_keyboard': menu[7]}, {'disable\_web\_page\_preview': false});

          return;

        }

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

          Bot.deleteMessage(options.message\_id);

          Bot.sendPhoto(msg, lastpost[2], menu[7], {'disable\_web\_page\_preview': false});

          return;

        }

        if (lastpost[3] == 'Видео') {

          Bot.deleteMessage(options.message\_id);

          Bot.sendVideo(msg, lastpost[2], menu[7], {'disable\_web\_page\_preview': false});

          return;

        }

        if (lastpost[3] == 'Аудио') {

          Bot.deleteMessage(options.message\_id);

          Bot.sendAudio(msg, lastpost[2], menu[7], {'disable\_web\_page\_preview': false});

          return;

        }

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

          Bot.deleteMessage(options.message\_id);

          Bot.sendDocument(msg, lastpost[2], menu[7], {'disable\_web\_page\_preview': false});

          return;

        }

      }

    }

    if (options.data.includes('user') ) {//авторизация пользователя

      if (options.data.includes('approve') ) {//принятие авторизации

        let idn = options.data.split('\_')[2];

        Bot.authSystemUser(idn, 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.";

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

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

      }

      if ( options.data.includes('deny') ) {//отклонение авторизации

        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.includes('page')) {//если нажата кнопка перелистнуть страницу в любом из списков

      let page = options.data.substr(5);

      let x = userData.callback\_query\_data;

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

      switch(x) {

        case 'Посмотреть документы':

          msg = '<b>Выбери пользователя, которого Вы хотите посмотреть</b>👇';

          data = getDataUsers("users");

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

          break;

        case 'sendAllmsgTo':

          msg = 'Выберите категорию, которой Вы хотите <b>отправить сообщение</b>👇';

          data = getDataColumnSheet('Test');

          Bot.editMessageKeyboard(msg, options.message\_id, null, keySheets(data, page, return\_key = 'sendmsg').keyboard);

          break;

        case 'Instructions':

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

          data = getDataColumnSheet('Instructions');

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

          break;

        case 'Удалить категорию':

          msg = 'Выберите категорию, которую Вы хотите <b>удалить</b>👇';

          data = getDataColumnSheet('Test');

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

          break;

        case 'setCategory':

          msg = 'Выберите категорию, которую Вы хотите <b>установить</b>👇';

          data = getDataColumnSheet('Test');

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

          break;

        case 'Удалить материал':

          msg = 'Выбери материал, который Вы хотите <b>удалить</b>👇';

          data = getDataColumnSheet('Instructions');

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

          break;

        default:

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

          break;

      }

      return;

    }

  }

  else{//если пользователь не зарегистрирован в боте

    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 msgComands.gs**

function msgComands(TelegramJSON) {

  let text = TelegramJSON.message.text;

  let chatid = TelegramJSON.message.chat.id;

  let userProperties = PropertiesService.getUserProperties();

  let msg, msg2 = '';

  if (text === '/addme'){

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

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

      Bot.sendMessage(msg);

      return;

    }

    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 администраторам

    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);

      return;

    }

  }

  if (text === '/privacy') {

    Bot.sendMessage(privacy\_msg);

    return;

  }

  if (Bot.getSystemUser().isAuth || superAdmin.includes(chatid)) {

    let userData = JSON.parse(userProperties.getProperty(chatid));

    if (text === '/test') {

      Bot.sendMessage(userData.callback\_query\_data);

      Bot.sendMessage(search\_row(userData.callback\_query\_data.split('\_')[1], 'Test', 1));

      return;

    }

    if (text === '/cancel') { // отмена операций

      if (userData.callback\_query\_data.includes('sendoc')) { //если отмена в меню отправить документы

        return;

      }

    }

    if (text === '/test2') {

      let users = Sheet('users').getRange(2, 2, Sheet('users').getLastRow(), 9).getValues();

      msg = text +'\n<b>С уважением, отдел опеки г.Уфа</b>';

      Bot.sendMessage(userData.callback\_query\_data.split('\_')[1]);

      Bot.sendMessage(users[1][8]);

      for (i in users) {

        if (users[i][8] === userData.callback\_query\_data.split('\_')[1]){

          Bot.sendMessageTo(users[i][0], msg);

        }

      }

      return;

    }

    userProperties.deleteProperty(chatid);

    if (text === '/start') {

      if (superAdmin.includes(chatid)) {

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

        Bot.sendMessage(msg, {'reply\_markup': {'inline\_keyboard': menu[2]} } );

      }

      else{

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

        if (getSheetVal('users', search\_row(chatid, 'users', 2), 9) === ''){// если статус заявки пустой, показывается полная клавиатура

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

        }

        else{

          Bot.sendMessage(msg, {'reply\_markup': {'inline\_keyboard': menu[0].slice(2)}});

        }

      }

      return;

    }

    if (text === '/send\_rules' && superAdmin.includes(chatid) ) {

      const groupId = getSheetVal('Settings', 7, 2);

      if(groupId){

        Bot.sendMessageTo(`@${groupId}`, rules, {'reply\_markup': {'inline\_keyboard': menu[6]}});

        return;

      }

      else{

        Bot.sendMessage('Добавьте имя для группы в листе Settings');

        return;

      }

    }

    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 авторизацию \n /rate - поставить рейтинг💹 \n /privacy - какие данные используются \n /cancel - отмена";

      Bot.sendMessage(msg);

      return;

    }

    if (text === '/donate') {

      Bot.sendMessage(msgd);

      return;

    }

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

      if (leaderBoard()) {

        let msg = leaderBoard();

        Bot.sendMessage(msg);

      }

      else{

        Bot.sendMessage("Список самых активных пуст!");

      }

      return;

    }

    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);

      return;

    }

    if (text === '/addme'){

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

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

        Bot.sendMessage(msg);

        return;

      }

      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 администраторам

      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);

        return;

      }

    }

    if (text === '/rate'){

      let keyboard = ['⭐️','⭐️⭐️','⭐️⭐️⭐️','⭐️⭐️⭐️⭐️','⭐️⭐️⭐️⭐️⭐️'];

      msg = "Как вы оцениваете этого бота?";

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

      return;

    }

  }

  else {//если неавторизованный пользователь

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

    Bot.sendMessage(msg);

    return;

  }

}

**Файл msgText.gs**

function msgText(TelegramJSON) {

  let text = TelegramJSON.message.text;

  let chatid = TelegramJSON.message.chat.id;

  let userProperties = PropertiesService.getUserProperties();

  let userData = JSON.parse(userProperties.getProperty(chatid));

  Bot.sendChatAction();

  let msg, msg2 = '';

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

    setSheetVal('users', search\_row(chatid, 'users', 2), 8, text)//вставка рейтинга напротив id пользователя

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

    return;

  }

  if (text && userData.callback\_query\_data.includes('setStatus')){

    Bot.sendChatAction();

    if (getDataColumnSheet('Settings', row=2,col=4).includes(text)) {

      let ind = search\_row(userData.callback\_query\_data.split('\_')[1], 'users', 2);

      msg = '✅<b>Cтатус успешно изменён.</b> \n\n<i>Используйте: /start, чтобы снова перейти в главное меню</i>';

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

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

      Bot.sendMessage(msg, {reply\_markup: { remove\_keyboard: true }});

      setSheetVal('users', ind, 9, text);

      msg2 = `✉️<b>Статус Вашей заявки обновлён:</b> \n<i>Текущий статус:</i> <b>${text}\n\nС уважением, отдел опеки г.Уфа</b>`;

      Bot.sendMessageTo(userData.callback\_query\_data.split('\_')[1], msg2);

      return;

    }

    else {

      msg = '⚠️ Выберите статус из списка снизу';

      Bot.sendMessage(msg);

      return;

    }

  }

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

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

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

    userProperties.deleteProperty(chatid);

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

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

    Bot.sendMessage(msg);

    return;

  }

  if (text && userData.callback\_query\_data === 'Добавить материал') {

    if ( text.split('///')[0].length > 30 ) {

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

      Bot.sendMessage(msg);

    }

    if (text.split('///')[1].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);

    }

    else {//если название от 1 до 30 и описание до 1024

      Sheet("Instructions").appendRow([text.split('///')[0], text.split('///')[1], '', 'Текст']);

      let msg = `✅Текст успешно добавлен. \n\n<i>Продолжайте присылать материалы для добавления,\n/start, чтобы снова перейти в главное меню</i>`;

      //userProperties.deleteProperty(chatid);

      Bot.sendMessage(msg);

    }

    return;

  }

  if (text && userData.callback\_query\_data === 'Добавить категорию') {

    if ( text.length > 30 ) {

      let msg = `⚠️<b>Название не должно превышать 30 символов.</b>\n\n<i>Если название будет больше 30 символов, текст выйдет за границы экрана в одной строке, что делает его менее читабельным</i>`;

      Bot.sendMessage(msg);

    }

    else {//если название от 1 до 30

      let data = [text];

      let data1 = Sheet('Users').getRange(1, 11, 1, Sheet('Users').getLastColumn()-10).getValues().flat();

      for(let i = 0; i < data1.length; i++) {

        data.push('➖ '+data1[i]);

      }

      Sheet("Test").appendRow(data);//добавление кнопок по умолчанию

      //userProperties.deleteProperty(chatid);

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

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

      let msg = `✅Категория успешно добавлена. \n\n<i>Продолжайте присылать названия для добавления новых категорий,\n/start, чтобы снова перейти в главное меню</i>`;

      Bot.sendMessage(msg);

    }

    return;

  }

  if (text && userData.callback\_query\_data === 'Добавить документ') {

    if ( text.length > 30 ) {

      let msg = `⚠️<b>Название не должно превышать 30 символов.</b>\n\n<i>Если название будет больше 30 символов, текст выйдет за границы экрана в одной строке, что делает его менее читабельным</i>`;

      Bot.sendMessage(msg);

    }

    else {//если название от 1 до 30

      setSheetVal('users', 1, Sheet('users').getLastColumn()+1, text);

      const lastColumn = Sheet('Test').getLastColumn()+1;

      const lastRow = Sheet('Test').getLastRow();

      for(let i = 1; i <= lastRow; i++) {

        setSheetVal('Test', i, lastColumn, `➖ `+text);

      }

      let msg = `✅Документ успешно добавлен. \n\n<i>Продолжайте присылать названия для добавления новых категорий,\n/start, чтобы снова перейти в главное меню</i>`;

      //userProperties.deleteProperty(chatid);

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

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

      Bot.sendMessage(msg);

    }

    return;

  }

  if (text && userData.callback\_query\_data.includes('sendmsgAdmin') ) { // отправка сообщения администратору от пользователя

    msg2 = "<b>Босс, новое сообщение от: " + "<a href='" + Bot.mentionByID() +"'>" + Bot.getUserFullName() + "</a></b>\n\n✉️" + text;

    let key = [[{'text': '✉️ Ответить', 'callback\_data': `sendmsg\_${chatid}`}]];

    Bot.sendMessageTo(superAdmin[0], msg2, {'reply\_markup': {'inline\_keyboard': key}});

    Bot.deleteMessage(TelegramJSON.message.message\_id-1);

    msg = `✅<b>Ваше сообщение успешно отправлено. </b> \n\n<i>/start, чтобы снова перейти в главное меню</i>`;

    userProperties.deleteProperty(chatid);

    Bot.sendMessage(msg);

    return;

  }

  if (text && userData.callback\_query\_data === 'sendAllmsg' ) { // отправка сообщения всем пользоватеям

    let users = getDataColumnSheet('users', 2, 2);//пользователи

    msg = text +'\n<b>С уважением, отдел опеки г.Уфа</b>';

    for (i in users) {

      Bot.sendMessageTo(users[i], msg);

    }

    msg = `Проинформировано: <b>${+i+1}</b> из <b>${users.length}</b>\n\n<i>/start, чтобы снова перейти в главное меню</i>`;

    Bot.sendMessage(msg);

    //userProperties.deleteProperty(chatid);

    return;

  }

  if (text && userData.callback\_query\_data.includes('sendAllmsgTo') ) { // отправка сообщения категории пользователей

    let users = Sheet('users').getRange(2, 2, Sheet('users').getLastRow(), 9).getValues();

    msg = text +'\n\n<b>С уважением, отдел опеки г.Уфа</b>';

    let c = 0;

    for (i in users) {

      if (users[i][8] === userData.callback\_query\_data.split('\_')[1]){

        Bot.sendMessageTo(users[i][0], msg); c+=1;

      }

    }

    msg = `Проинформировано: <b>${c}</b> из <b>${users.length}</b>\n\n<i>/start, чтобы снова перейти в главное меню</i>`;

    Bot.sendMessage(msg);

    //userProperties.deleteProperty(chatid);

    return;

  }

  if (text && userData.callback\_query\_data.includes('sendmsg') ) { // отправка сообщения пользователю

    try{

      msg2 = text +'\n\n<b>С уважением, отдел опеки г.Уфа</b>';

      Bot.sendMessageTo(userData.callback\_query\_data.split('\_')[1], msg2);

      msg = `✅<b>Сообщение успешно отправлено. </b> \n\n<i>/start, чтобы снова перейти в главное меню</i>`;

      //userProperties.deleteProperty(chatid);

      Bot.sendMessage(msg);

    }

    catch{

      msg = `⚠️Произошла ошибка`;

      //userProperties.deleteProperty(chatid);

      Bot.sendMessage(msg);

    }

    return;

  }

}

**Файл msgChatText.gs**

function msgChatText(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;

      let chatName = TelegramJSON.message.reply\_to\_message.chat.title;

      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, chatName)}</b>`;

        let trophy = getTrophy(chat\_idReply);

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

        Bot.sendMessage(msg);

        return;

      }

    }

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

      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()) { // если новый пользователь в чате

    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);

    return;

  }

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

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

    Bot.sendMessage(rules);

    return;

  }

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

    if (leaderBoard()) {

      let msg = leaderBoard();

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

      Bot.sendMessage(msg);

      return;

    }

    else{

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

      return;

    }

  }

}

**Файл msgPhoto.gs**

//отправка фото

function msgPhoto(TelegramJSON) {

  let chatid = TelegramJSON.message.chat.id;

  let userProperties = PropertiesService.getUserProperties();

  let userData = JSON.parse(userProperties.getProperty(chatid));

  let photo = TelegramJSON.message.photo;

  Bot.sendChatAction();

  if (photo && userData.callback\_query\_data === 'Добавить материал') {

    if ( TelegramJSON.message.caption.split('///')[0].length > 30 ) {

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

      Bot.sendMessage(msg);

      return;

    }

    if (TelegramJSON.message.caption.split('///')[1].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);

      return;

    }

    else {//если название от 1 до 30 и описание до 1024

      let sheet = SpreadsheetApp.openById(botSheet).getSheetByName("Instructions");

      sheet.appendRow([TelegramJSON.message.caption.split('///')[0], TelegramJSON.message.caption.split('///')[1], photo[3].file\_id, 'Изображение']);

      let msg = `✅Изображение с описанием успешно добавлено. \n\n<i>Продолжайте присылать материалы для добавления,\n/start, чтобы снова перейти в главное меню</i>`;

      //userProperties.deleteProperty(chatid);

      Bot.sendMessage(msg);

      return;

    }

  }

  if (photo && userData.callback\_query\_data.includes('sendoc') ) {

    let ur = search\_row(chatid, 'users', 2);

    let dcol = search\_col(userData.callback\_query\_data.split('\_')[1], 'users', 1);

    setSheetVal('users', ur, dcol, photo[3].file\_id);

    msg = `✅Документ: <b>${userData.callback\_query\_data.split('\_')[1]}</b> успешно добавлен.`;

    Bot.deleteMessage(TelegramJSON.message.message\_id-1);

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

    //userProperties.deleteProperty(chatid);

    Bot.sendMessage(msg);

    let msg2 = '<b>Выберите документ из приведённого ниже списка👇</b>';

    userProperties.setProperty(chatid, JSON.stringify({callback\_query\_data: 'Отправить документы'}));

    Bot.sendMessage(msg2, {'reply\_markup': getkeyFlags(chatid, return\_key = "menu\_start")});

    return;

  }

}

**Файл msgDoc.gs**

function msgDoc(TelegramJSON) { // отправка документа

  let chatid = TelegramJSON.message.chat.id;

  let userProperties = PropertiesService.getUserProperties();

  let userData = JSON.parse(userProperties.getProperty(chatid));

  let document = TelegramJSON.message.document;

  let mimetype = TelegramJSON.message.document.mime\_type;

  let msg, msg2 = '';

  Bot.sendChatAction();

  if (document && userData.callback\_query\_data === 'Добавить материал') {

    if ( TelegramJSON.message.caption.split('///')[0].length > 30 ) {

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

      Bot.sendMessage(msg);

      return;

    }

    if (TelegramJSON.message.caption.split('///')[1].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);

      return;

    }

    else {//если название от 1 до 30 и описание до 1024

      let sheet = SpreadsheetApp.openById(botSheet).getSheetByName("Instructions");

      sheet.appendRow([TelegramJSON.message.caption.split('///')[0], TelegramJSON.message.caption.split('///')[1], document.file\_id, 'Документ']);

      let msg = `Документ с описанием успешно добавлено. \n\n<i>Продолжайте присылать материалы для добавления,\n/menu, чтобы снова перейти в главное меню</i>`;

      //userProperties.deleteProperty(chatid);

      Bot.sendMessage(msg);

      return;

    }

  }

  if (document && userData.callback\_query\_data.includes('sendoc') ) {

    if (mimetype === 'application/pdf' || mimetype === 'application/msword' || mimetype === 'application/vnd.openxmlformats-officedocument.wordprocessingml.document') { //если прислали файл pdf, doc, docx

      let ur = search\_row(chatid, 'users', 2);

      let dcol = search\_col(userData.callback\_query\_data.split('\_')[1], 'users', 1);

      setSheetVal('users', ur, dcol, document.file\_id);

      //userProperties.deleteProperty(chatid);

      Bot.deleteMessage(TelegramJSON.message.message\_id-1);

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

      msg = `✅Документ: <b>${userData.callback\_query\_data.split('\_')[1]}</b> успешно добавлен.`;

      Bot.sendMessage(msg);

      msg2 = '<b>Выберите документ из приведённого ниже списка👇</b>';

      userProperties.setProperty(chatid, JSON.stringify({callback\_query\_data: 'Отправить документы'}));

      Bot.sendMessage(msg2, {'reply\_markup': getkeyFlags(chatid, return\_key = "menu\_start")});

      return;

    }

    else{//если прислали другой формат

      Bot.deleteMessage(TelegramJSON.message.message\_id-1);

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

      msg = `⚠️<b>Неверный формат документа.</b> \n\n<i>Поддерживаются следующие форматы:</i> \n<b>| PDF | DOC | JPEG | JPG | PNG |</b> \n\n/start - чтобы перейти в меню`;

      Bot.sendMessage(msg, {'reply\_markup': {'inline\_keyboard': [[{'text': '🏡 Назад', 'callback\_data': 'Отправить документы'}]] } });

      return;

    }

  }

}

**Файл msgAudio.gs**

function msgAudio(TelegramJSON) {

  let chatid = TelegramJSON.message.chat.id;

  let userProperties = PropertiesService.getUserProperties();

  let audio = TelegramJSON.message.audio;

  let userData = JSON.parse(userProperties.getProperty(chatid));

  //let mimetype = audio.mime\_type;

  Bot.sendChatAction();

  if ( audio && userData.callback\_query\_data === "Добавить материал") {

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

    if (name\_text.length > 30) {

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

      Bot.sendMessage(msg);

    }

    let caption = TelegramJSON.message.caption;

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

    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);

    }

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

      let sheet = SpreadsheetApp.openById(botSheet).getSheetByName("Instructions");

      sheet.appendRow([name\_text, caption\_text, audio.file\_id, 'Изображение'])

      let msg = `✅ Аудио с описанием успешно добавлено. \n\n<i>Продолжайте присылать материалы для добавления,\n/menu, чтобы снова перейти в главное меню</i>`;

      //userProperties.deleteProperty(chatid);

      Bot.sendMessage(msg);

    }

    return;

  }

}

**Файл msgVideo.gs**

function msgVideo(TelegramJSON) {

  let chatid = TelegramJSON.message.chat.id;

  let userProperties = PropertiesService.getUserProperties();

  let video = TelegramJSON.message.video;

  let userData = JSON.parse(userProperties.getProperty(chatid));

  //let mimetype = video.mime\_type;

  Bot.sendChatAction();

  if( userData.callback\_query\_data === "Добавить материал" ) {

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

    if ( name\_text.length > 30 ) {

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

      Bot.sendMessage(msg);

    }

    let caption = TelegramJSON.message.caption;

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

    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);

    }

    else {//если название от 1 до 30 и Описание меньше 1024 символов

      let sheet = SpreadsheetApp.openById(botSheet).getSheetByName("Instructions");

      sheet.appendRow([name\_text, caption\_text, video.file\_id, 'Видео']);

      let msg = `✅ Видео с описанием успешно добавлено. \n\n<i>Продолжайте присылать материалы для добавления,\n/menu, чтобы снова перейти в главное меню</i>`;

      //userProperties.deleteProperty(chatid);

      Bot.sendMessage(msg);

    }

    return;

  }

}

**Файл debug.gs**

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

  setSheetVal('JSON', 1, 1, JSON.stringify(TelegramJSON, null, 7))

  return;

}

**Файл createKeyboards.gs**

const menu = [

 [[{'text': '📨 Отправить документы', 'callback\_data': 'Отправить документы'}],

  [{'text': '🙋‍♂️Я собрал все документы🙋‍♀️', 'callback\_data': 'sendWait'}],

  [{'text': '🔄 Посмотреть состояние', 'callback\_data': 'loading'}],

  [{'text': '💬 Cообщение администратору', 'callback\_data': 'sendmsgAdmin'}],

  [{'text': '🗒 Расписание работы', 'callback\_data': 'schedule'}],

  [{'text': '⁉️ Как пользоваться?', 'url': 'https://telegra.ph/Kak-polzovatsya-botom-otdela-Opeki-07-18'}],

  [{'text': 'ℹ️ Информация об организации', 'callback\_data': 'MoreInfo'}],

  [{'text': '☎️ Телефоны доверия', 'callback\_data': 'Hotlines'}]],

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

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

   [{'text': '💬 Cообщение пользователям', 'callback\_data': 'sendmsg'}],

   [{'text': '⚙️ Основные функции', 'callback\_data': 'Functions'}],

   [{'text': '🗒 Расписание работы', 'callback\_data': 'schedule'}],

   [{'text': '⁉️ Как пользоваться?', 'url': 'https://telegra.ph/Kak-polzovatsya-botom-otdela-Opeki-07-18'}],

   [{'text': 'ℹ️ Информация', 'callback\_data': 'Information'}],

   [{'text': '☎️ Телефоны доверия', 'callback\_data': 'Hotlines'}]],

  [[{'text': '📋 Инструкции', 'callback\_data': 'Instructions'}],

   [{'text': '🗂️ Архив статей', 'callback\_data': 'Articles'}],

   [{'text': '🪧 Информация об организации', 'callback\_data': 'MoreInfo'}],

   [{'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': 'Instructions'}]],

  [[{'text': '⚙️ Изменить документы 📖', 'callback\_data': 'editDocs'}],

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

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

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

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

   [{'text': '➕ Добавить категорию ✳️', 'callback\_data': 'Добавить категорию'}],

   [{'text': '🗑️ Удалить категорию ✳️', 'callback\_data': 'Удалить категорию'}],

   [{'text': '➕ Добавить документ📄', 'callback\_data': 'Добавить документ'}],

   [{'text': '🗑️ Удалить документ📄', 'callback\_data': 'Удалить документ'}],

   [{'text': '❇️ Открыть таблицу ❇️', 'url': 'https://docs.google.com/spreadsheets/d/'+botSheet}],

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

  [[{'text': '📨 Отправить документы', 'callback\_data': 'Отправить документы'}],

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

  [[{'text': '☑️ Продолжить', 'callback\_data': 'waiting'}],

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

  [[{'text': '✅ Продолжить', 'callback\_data': 'waiting'}],

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

  [[{'text': '💬 Cообщение всем пользователям', 'callback\_data': 'sendAllmsg'}],

   [{'text': '💬 Cообщение отдельной категории', 'callback\_data': 'sendAllmsgTo'}],

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

];

//массив кнопок в зависимости от наличия значений таблицы

function getkeyFlags(chatid, return\_key = "menu\_start"){

  //chatid=252898281;

  let ind = search\_row(chatid, 'users', 2); //индекс пользователя в базе данных

  let category = getSheetVal('users', ind, 10);//категория пользователя - название

  let indCategory = search\_row(category, 'Test', 1);

  let a = Sheet('users').getRange(1, 11, 1, Sheet('users').getLastColumn()-10).getValues().flat();//названия категорий

  let b = Sheet('Test').getRange(indCategory, 2, 1, Sheet('Test').getLastColumn()-1).getValues().flat(); //список обязательных документов

  let c = Sheet('users').getRange(ind, 11, 1, Sheet('users').getLastColumn()-10).getValues().flat();//массив с данными пользователя

  const categories = b //новый список обязательных документов

    .filter(document => document.includes('✅'))

    .map(document => document.slice(2));

  let arrUCatNames = []; //список имеющихся документов у пользователя

  for (let i = 0; i < c.length; i++) {

    if (c[i]){ //если документ

      arrUCatNames.push(a[i]); //формирование списка названий по наличию документа

    }

  }

  //функция проверки массивов на уникальность

  function unique(arr, arr2) {

    let result = [];

    for (let str of arr) {

      if (arr2.includes(str)) {

        result.push([{ text: '✅ '+str, callback\_data: str}]);

      }

      else{

        result.push([{ text: '➖ '+str, callback\_data: str}])

      }

    }

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

    return result;

  }

  let keyboards = {

    inline\_keyboard: [...unique(categories, arrUCatNames)]

  }

  //console.log(keys);

  return keyboards

}

//массив кнопок из значений таблицы с колбеком номера из списка

function getkeys(arraykeys, k, return\_key = "menu\_start", access\_key){

  //arraykeys = Sheet('Test').getRange(3, 2, 1, Sheet('Test').getLastColumn()-1).getValues().flat(); //список обязательных документов

  let keys = [];

  if (k == 1){//режим 1

    for (i in arraykeys) {

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

    }

    keys.push([{ text: '💬 Cообщить пользователям', callback\_data: access\_key}])

    keys.push([{ text: '🏡 Назад', callback\_data: return\_key }])//кнопка "назад" по-умолчанию

  }

  if (k == 2){ //режим 2

    for (i in arraykeys) {

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

    }

    keys.push([{ text: '🏡 Назад', callback\_data: return\_key }])//кнопка "назад" по-умолчанию

  }

  let keyboards = {

    inline\_keyboard: [...keys]

  }

  //console.log(keys);

  return keyboards

}

//функция создания массива кнопок при просмотре списка материалов

function keySheets(arraykeys, page = "page-0", return\_key = "menu\_start", r=1) {

  let container = [];

  let keyboardLength = 5

  let pageArr = listArray(keyboardLength, arraykeys, container)

  page = +page.replace(/[^0-9]/g, '');

  let keys = [];

  if (r === 1) { //режим 1

    for (i in pageArr[page]) {

      let id = pageArr[page][i]; //id пользователя из списка

      if (id.includes('\*~')) {//если кнопки = массив пользователей

        keys.push([{ text: `${id.split('\*~')[1]} ${id.split('\*~')[2]}`, callback\_data: id.split('\*~')[0]}]);

      }

      else {

        keys.push([{ text: id, callback\_data: id}]);

      }

    }

  }

  if (r === 2) { //режим 2

    for (i in pageArr[page]) {

      keys.push([{ text: String(arraykeys[i]), callback\_data: String(arraykeys[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 == pageArr.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**

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

 \*           Функционал для кармы           \*

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

const wsData = Sheet('Рейтинг');

const wsSetting = Sheet('Settings');

function setCarma(name, chatId, chatName) {

  let arrChatId = wsData.getRange(2, 5, wsData.getLastRow()).getValues().flat();

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

  if (idRow == 1) {

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

    let currentCarma = 1

    return currentCarma

  }

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

  currentCarma = currentCarma + 1

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

  wsData.getRange('B2:E').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

}

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

 \*                Bot функции               \*

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

function Sheet(name) {//открытие листа текущей таблицы

  return SpreadsheetApp.openById(botSheet).getSheetByName(name);

}

//функция подписки

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

  let row = undefined

  if(row\_t){

    row = row\_t

  }

  else{

    row = search\_row(chat\_id, 'users', 2)

  }

  if(row){

    return 0

  }

  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 = search\_row(chat\_id, 'users', 2)

  }

  if(row){

    Sheet("users").deleteRow(row)

    Bot.sendMessage("❌Отписаны❌")

  }

  return 0

}

//функция получения количества подписчиков

function fan\_num(\_chat\_id){

  return Sheet("users").getLastRow()-1

}

//переменная сообщения donate

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)

}

//получение значений с одного полного столбца с конкретного листа

function getDataColumnSheet(sheetName, row=1,col=1) {

  //sheetName = 'Test';

  try{

    const list = Sheet(sheetName).getSheetValues(row, col, Sheet(sheetName).getLastRow(), 1).filter(n => n!='').flat();

    //console.log(list);

    return list

  }

  catch{

    return null

  }

}

//Получение значений начиная со стартовго столбца и до конца строки

function getDataRowSheet(sheetName, row=1,col=1) {

  //sheetName = 'Test';

  try{

    const list = Sheet(sheetName).getSheetValues(row, col, 1, Sheet(sheetName).getLastColumn()).filter(n => n!='').flat();

    //console.log(list);

    return list

  }

  catch{

    return null

  }

}

function getDataUsers(sheetname) {

  sheetname = 'users';

  try{

    let ss = SpreadsheetApp.openById(botSheet);

    let sheet = ss.getSheetByName(sheetname);

    let data = sheet.getRange(2, 2, sheet.getLastRow()-1, 4).getValues();

    let lists = []

    for (i in data) {

      lists.push(`${data[i][0]}\*~${data[i][2]}\*~${data[i][3]}`);

    }

    list = lists.flat()

    console.log(list)

    return list

  }

  catch{

    return 0

  }

}

//чтение данных на листе

function getDataSheet(sheetName) {

  //sheetname = 'Instructions'

  return Sheet(sheetName).getRange().getValues();

}

//добавление массива в виде новой строки на листе

function addDataSheet(sheetName, data) {

  Sheet(sheetName).appendRow(data)

}

//взятие одного значения с ячейки на листе

function getSheetValue(sheetName, row, col){

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

}

//функция взятия значений в нужном листe

function getSheetVal(sheetName, row, col){

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

}

//функция установки значений в нужный лист

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

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

}

//функция добавления строки с контентом из массива по ячейкам через запятую

function setSheetValues(sheetName, val){

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

  return Sheet(sheetName).appendRow(values)

}

//функция удаления ячейки из листа

function deleteSheetVal(sheetName, row, col){

  return Sheet(sheetName).getRange(row, col).deleteCells();

}

function delSheet(sheetName) {

  //sheetName = 'лист';

  SpreadsheetApp.openById(botSheet).setActiveSheet(Sheet(sheetName), true);

  return SpreadsheetApp.openById(botSheet).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 строки на листе по значению

function search\_row(text, list, index, start\_row = 1){

  //text = "Вторая";

  //list = 'test';

  let lastRow = Sheet(list).getLastRow();

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

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

    if(this\_text == String(text)){

      //console.log(start\_row)

      return start\_row

    }

    else undefined

  }

}

//функция поиска колонки на листе по значению

function search\_col(text, list, index, start\_col = 1){

  //text = "Паспорт";

  //list = 'users';

  let lastColumn = Sheet(list).getLastColumn();

  for(start\_col=1; start\_col<=lastColumn; start\_col++){

    this\_text = getSheetVal(list, index, start\_col)

    if(this\_text == String(text)){

      //console.log(start\_col)

      return start\_col

    }

    else undefined

  }

}

//взятие массива значений

function getSheetValues(sheetName, row, col, cols=1){

  return Sheet(sheetName).getSheetValues(row, col, Sheet(sheetName).getLastRow(), cols)

}

function getInfoArticles(){

  let values = Sheet('Архив').getRange(1, 1, Sheet('Архив').getLastRow(), Sheet('Архив').getLastColumn()).getValues();

  let board = []

  for (let i = 0; i < Sheet('Архив').getLastRow(); i++) {

    let info = (`<a href="${values[i][1]}">${values[i][0]}</a>\n`)

    board += info;

  }

  return board.toString();

}

function getInfoValues(){ //телфоны доверия

  let values = Sheet('INFO').getRange(1, 1, Sheet('INFO').getLastRow(), Sheet('INFO').getLastColumn()-1).getValues();

  let board = []

  for (let i = 0; i < Sheet('INFO').getLastRow(); i++) {

    if(values[i][0] == 'Телефоны доверия' || values[i][0] == 'Все об Уфе'){

      let info = (`<a href="${values[i][1].substring(19)}">${values[i][0]}</a>\n`);

      board += info;

    }

    else{

      let info = (`<a href="${values[i][1]}">${values[i][0]}</a>\n`)

      board += info;

    }

  }

  return board.toString();

}

**Файл other.gs**

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 hotlines\_msg = getSheetValue("Settings", 6, 2).toString();

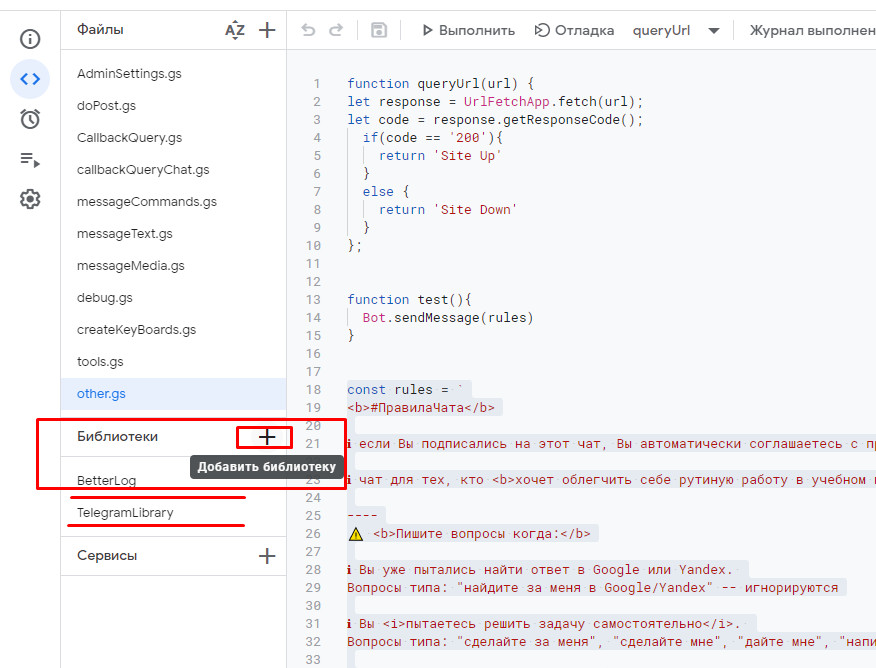
let version\_msg = 'PROGRAM NAME:' + '1' + '\n\nТекущая версия на' + '<u>' + '1' + '</u>' + ":\n" + '1';

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 отдела опеки г. Уфа.\n\nАвтор: <b>Куликов Александр</b><a href="https://t.me/kulikov\_mister">.</a> - Магистрант БГПУ им.М.Акмуллы';

**ВАЖНО:**

Необходимо также подключить две библиотеки для полноценной работы скрипта.

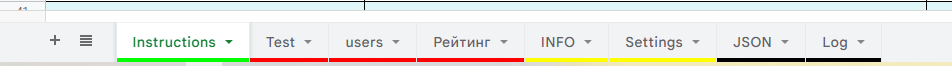


1. **1DSyxam1ceq72bMHsE6aOVeOl94X78WCwiYPytKi7chlg4x5GqiNXSw0l** – библиотека для отладки кода

(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.

