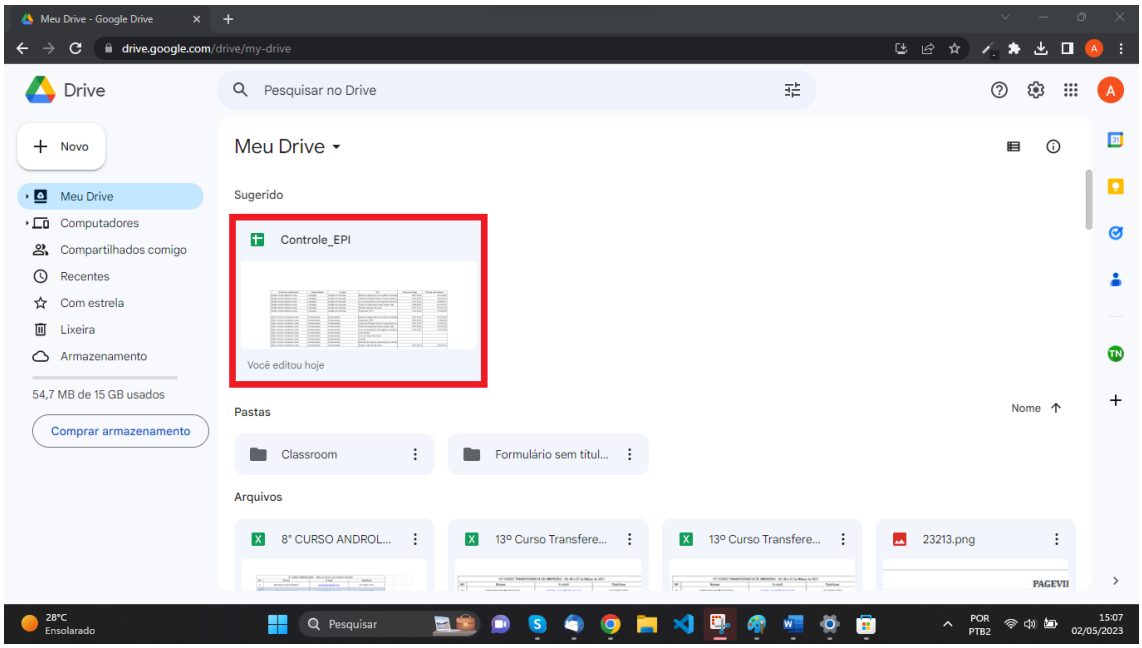
Sumário

INTRODUÇÃO

Esta automação foi criada utilizando as ferramentas do Google: **Apps Script, Google Agenda e Google Sheets.**

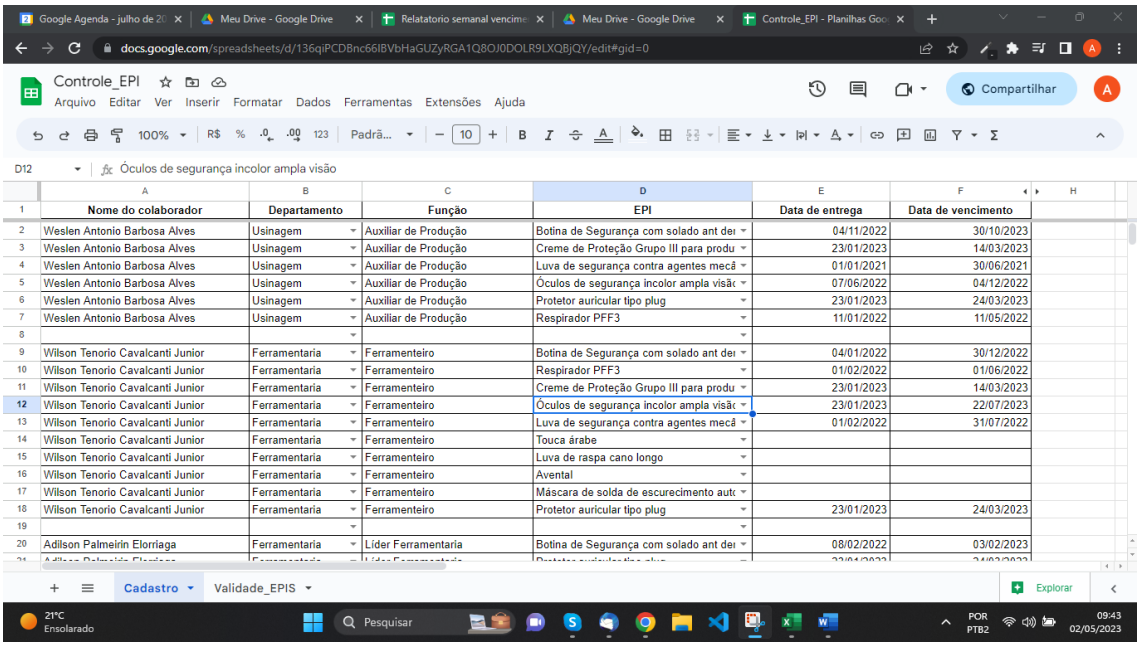
GOOGLE SHEETS

A planilha utilizada nesta automação se encontra no google drive da conta agendawta@gmail.com. A planilhas utilizada se chama **“Controle_EPI”**.



Esta planilha serve como uma base de dados, ela é composta por duas páginas: **“Cadastro”** e “**Validade_EPIS”.**

A página **“Cadastro”** contém os dados: **Nome do colaborador, Departamento, Função, EPI, Data de entrega, Data de Vencimento.**



	A	B	C	D	E	F
1	Nome do colaborador	Departamento	Função	EPI	Data de entrega	Data de vencimento
2	Weslen Antonio Barbosa Alves	Usinagem	Auxiliar de Produção	Botina de Segurança com solado ant de	04/11/2022	30/10/2023
3	Weslen Antonio Barbosa Alves	Usinagem	Auxiliar de Produção	Creme de Proteção Grupo III para produ	23/01/2023	14/03/2023
4	Weslen Antonio Barbosa Alves	Usinagem	Auxiliar de Produção	Luva de segurança contra agentes mecâ	01/01/2021	30/06/2021
5	Weslen Antonio Barbosa Alves	Usinagem	Auxiliar de Produção	Óculos de segurança incolor ampla visã	07/06/2022	04/12/2022
6	Weslen Antonio Barbosa Alves	Usinagem	Auxiliar de Produção	Protetor auricular tipo plug	23/01/2023	24/03/2023
7	Weslen Antonio Barbosa Alves	Usinagem	Auxiliar de Produção	Respirador PFF3	11/01/2022	11/05/2022
8						
9	Wilson Tenorio Cavalcanti Junior	Ferramentaria	Ferramenteiro	Botina de Segurança com solado ant de	04/01/2022	30/12/2022
10	Wilson Tenorio Cavalcanti Junior	Ferramentaria	Ferramenteiro	Respirador PFF3	01/02/2022	01/06/2022
11	Wilson Tenorio Cavalcanti Junior	Ferramentaria	Ferramenteiro	Creme de Proteção Grupo III para produ	23/01/2023	14/03/2023
12	Wilson Tenorio Cavalcanti Junior	Ferramentaria	Ferramenteiro	Óculos de segurança incolor ampla visã	23/01/2023	22/07/2023
13	Wilson Tenorio Cavalcanti Junior	Ferramentaria	Ferramenteiro	Luva de segurança contra agentes mecâ	01/02/2022	31/07/2022
14	Wilson Tenorio Cavalcanti Junior	Ferramentaria	Ferramenteiro	Touca árabe		
15	Wilson Tenorio Cavalcanti Junior	Ferramentaria	Ferramenteiro	Luva de raspa cano longo		
16	Wilson Tenorio Cavalcanti Junior	Ferramentaria	Ferramenteiro	Avental		
17	Wilson Tenorio Cavalcanti Junior	Ferramentaria	Ferramenteiro	Máscara de solda de escurecimento aut		
18	Wilson Tenorio Cavalcanti Junior	Ferramentaria	Ferramenteiro	Protetor auricular tipo plug	23/01/2023	24/03/2023
19						
20	Adilson Palmeirin Elorriaga	Ferramentaria	Líder Ferramentaria	Botina de Segurança com solado ant de	08/02/2022	03/02/2023

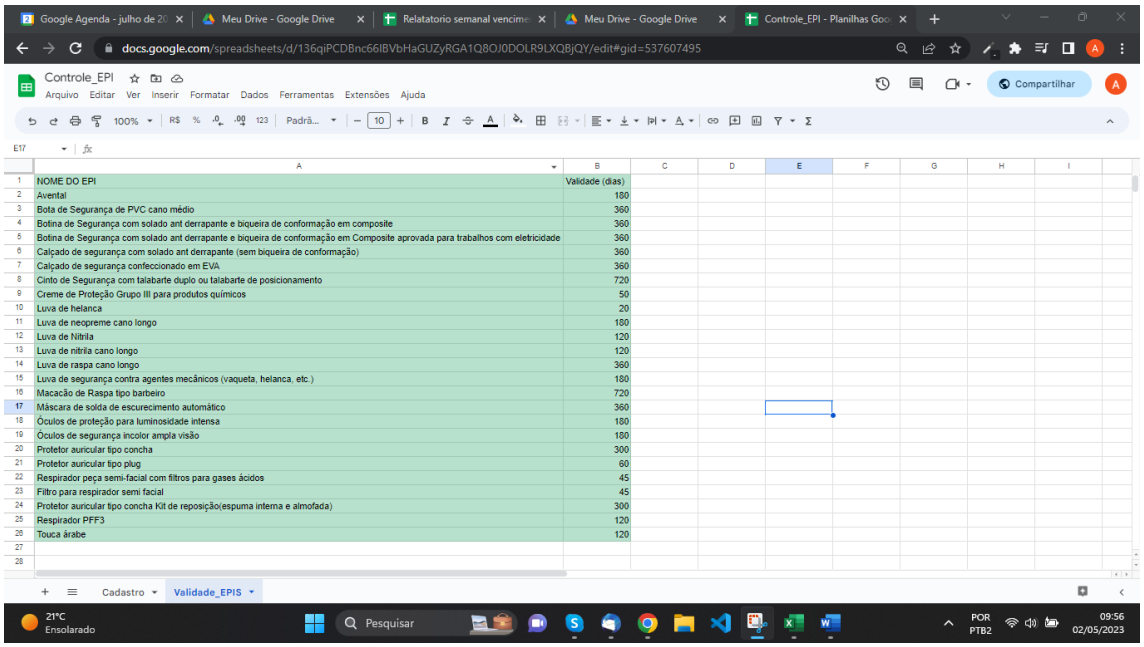
OBS.: A coluna data de vencimento calcula as datas automaticamente através da fórmula:

```
=SE(OU(ÉCÉL.VAZIA(D2); ÉCÉL.VAZIA(E2)); ""; E2+PROCV(D2;Validade_EPIS!$A$1:$B$28;2;0))
```

Esta parte da fórmula verifica se as colunas EPI e Data de entrega estão preenchidas e retorna vazio caso as colunas estejam em branco.

Esta parte realiza a soma do prazo de validade à data de entrega tendo como resultado a data de vencimento.

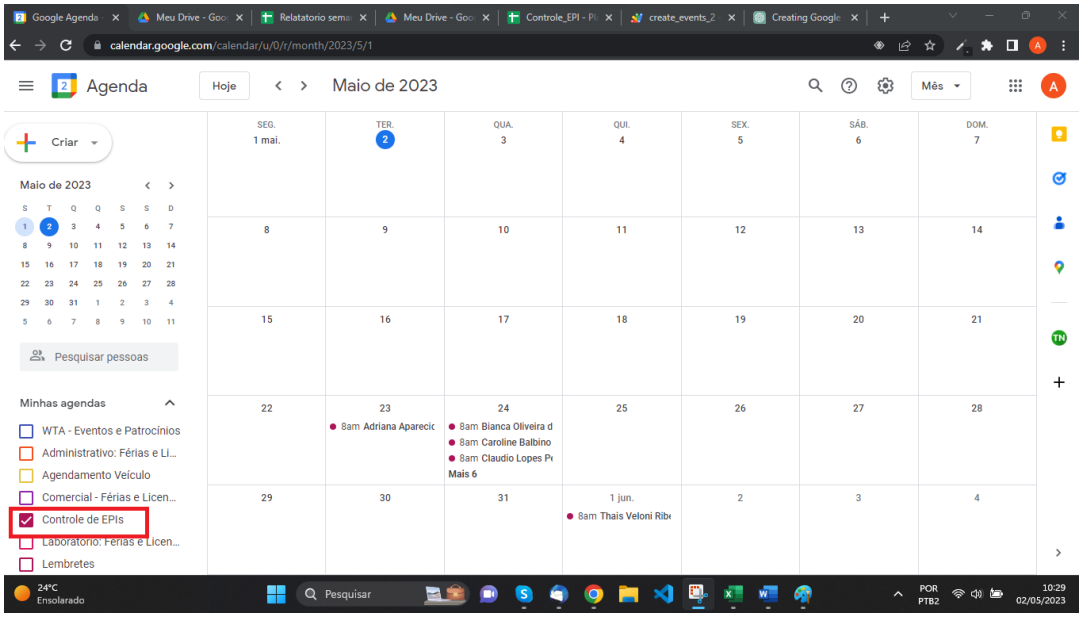
A página **“Validade_EPIS”** contém as informações referentes aos EPIS que são utilizados, informações como o nome do EPI e o prazo de validade.



Controle_EPI

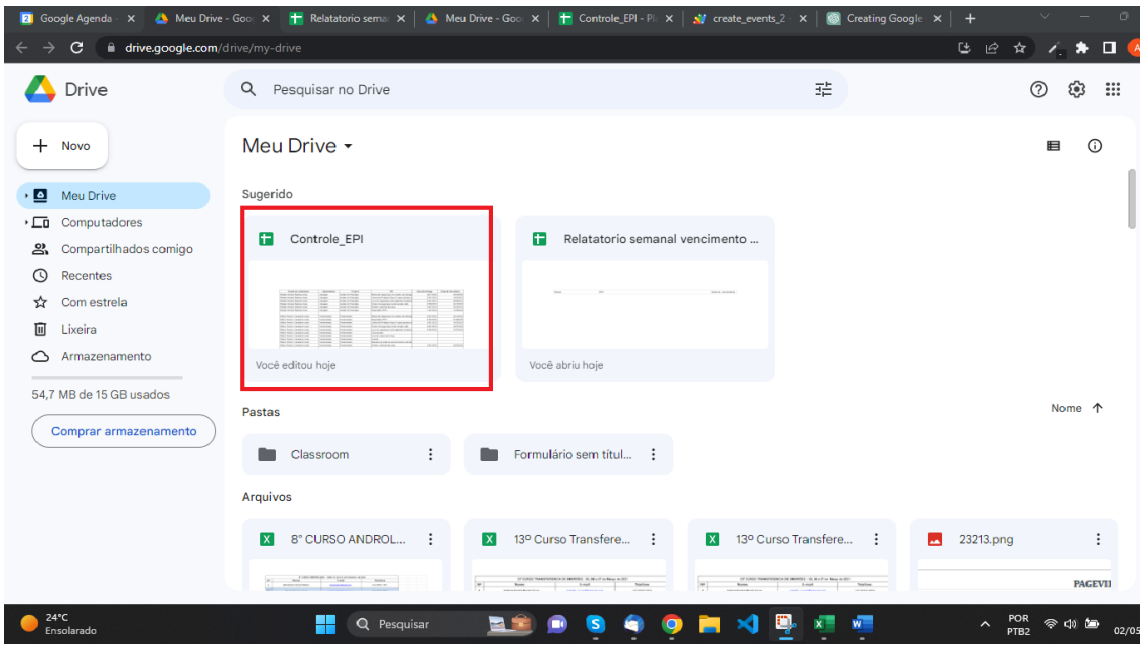
	A	B
1	NOME DO EPI	Validade (dias)
2	Avental	180
3	Bota de Segurança de PVC cano médio	360
4	Botina de Segurança com solado ant derrapante e biqueira de conformação em composite	360
5	Botina de Segurança com solado ant derrapante e biqueira de conformação em Composite aprovada para trabalhos com eletricidade	360
6	Calçado de segurança com solado ant derrapante (sem biqueira de conformação)	360
7	Calçado de segurança confeccionado em EVA	360
8	Cinto de Segurança com talabarte duplo ou talabarte de posicionamento	720
9	Creme de Proteção Grupo III para produtos químicos	50
10	Luva de helanca	20
11	Luva de neopreme cano longo	180
12	Luva de Nitrila	120
13	Luva de nitrila cano longo	120
14	Luva de raspa cano longo	360
15	Luva de segurança contra agentes mecânicos (vaqueta, helanca, etc.)	180
16	Macacão de Raspa tipo barbeiro	720
17	Máscara de solda de escurecimento automático	360
18	Óculos de proteção para luminosidade intensa	180
19	Óculos de segurança incolor ampla visão	180
20	Protetor auricular tipo concha	300
21	Protetor auricular tipo plug	60
22	Respirador peça semi-facial com filtros para gases ácidos	45
23	Filtro para respirador semi facial	45
24	Protetor auricular tipo concha Kit de reposição(espuma interna e almofada)	300
25	Respirador PFF3	120
26	Touca árabe	120
27		
28		

Esta planilha tem como função coletar e armazenar os dados para que um código os leia e crie eventos na data do vencimento do EPI na agenda **“Controle de EPIs”** da conta **agendawta@gamail.com**

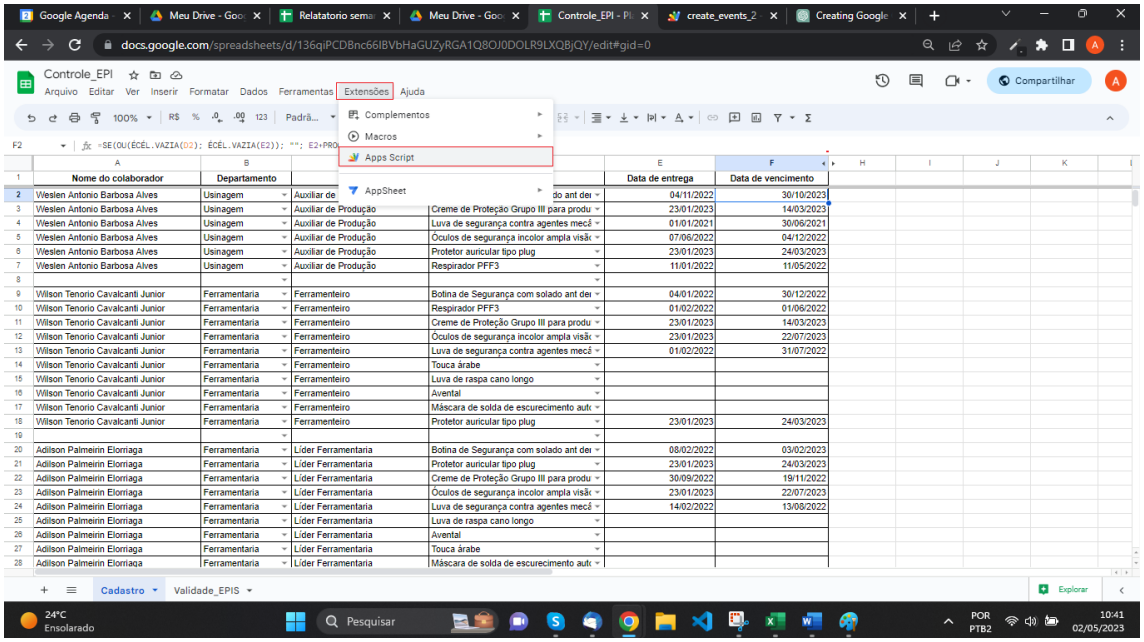


GOOGLE APPS SCRIPT

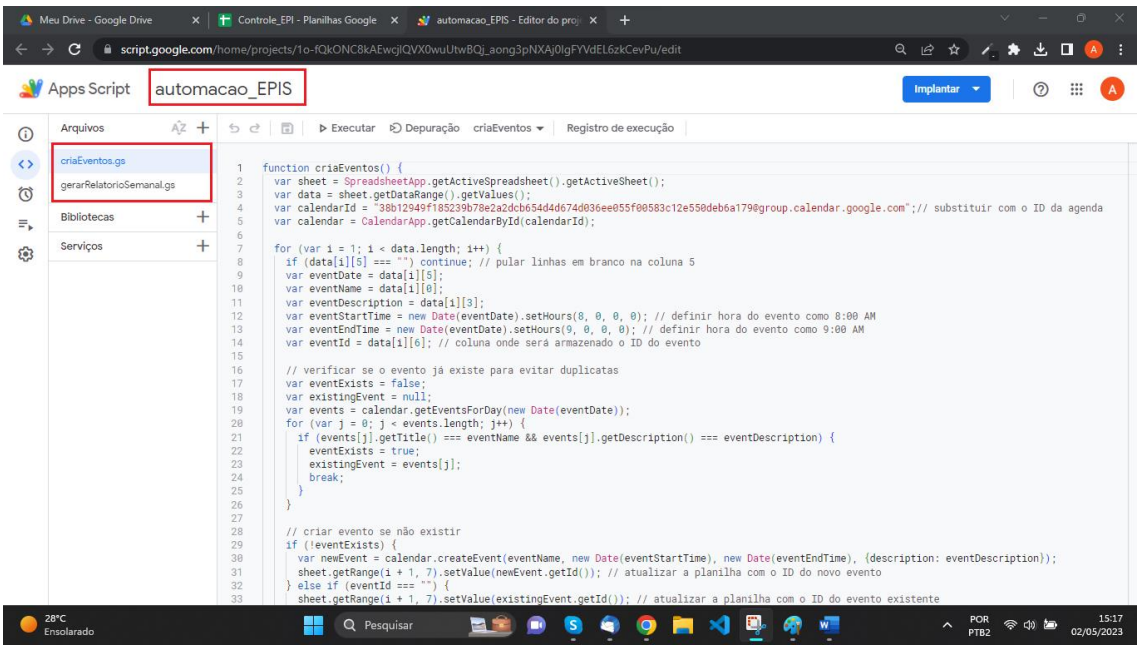
A automação funciona com dois códigos escritos em Javascript, para acessá-los, abra a planilha **“Controle_EPI”**;



Clique na aba **“Extensões”** e clique em **“Apps Script”**.



Aqui você encontrará um projeto chamado “**automacao_EPIS”** que contém os dois códigos Javascript: **criaEventos.gs** e **gerarRelatorioSemanal.gs**



FUNCIONAMENTO DO CÓDIGO criaEventos

O código **criaEventos** é responsável por ler os dados da planilha **“Controle_EPI”** e criar eventos no google agenda.

```
1  function criaEventos() {
2    var sheet = SpreadsheetApp.getActiveSpreadsheet().getActiveSheet();
3    var data = sheet.getDataRange().getValues();
4    var calendarId = "38b12949f185239b78e2a2dcb654d4d674d036ee055f00583c12e550deb6a179@group.calendar.google.com"
5    var calendar = CalendarApp.getCalendarById(calendarId);
6
```

- A linha 1 define a função “**criaEventos**”.
- A linha 2 obtém a planilha ativa atual, no caso a planilha **“ContrIole_EPI”** e armazena na variável **“sheet”**.
- A linha 3 obtém todos os valores contidos na planilha e armazena na variável **“data”**.
- A linha 4 define o ID da agenda que será utilizada para realizar os agendamentos e armazena na variável **“calendarId”**.
- A linha 5 usa o ID da agenda para buscar a agenda em que os eventos serão criados, no caso a agenda **“Controle de EPIs”**, e armazená-la na variável **‘calendar’**.

```
7  for (var i = 1; i < data.length; i++) {
8    if (data[i][5] === "") continue; // pular linhas em branco na coluna 5
9    var eventDate = data[i][5];
10   var eventName = data[i][0];
11   var eventDescription = data[i][3];
12   var eventStartTime = new Date(eventDate).setHours(8, 0, 0, 0); // definir hora do evento como 8:00 AM
13   var eventEndTime = new Date(eventDate).setHours(9, 0, 0, 0); // definir hora do evento como 9:00 AM
14   var eventId = data[i][6]; // coluna onde será armazenado o ID do evento
15
16   // verificar se o evento já existe para evitar duplicatas
17   var eventExists = false;
18   var existingEvent = null;
19   var events = calendar.getEventsForDay(new Date(eventDate));
20   for (var j = 0; j < events.length; j++) {
21     if (events[j].getTitle() === eventName && events[j].getDescription() === eventDescription) {
22       eventExists = true;
23       existingEvent = events[j];
24       break;
25     }
26   }
27
28   // criar evento se não existir
29   if (!eventExists) {
30     var newEvent = calendar.createEvent(eventName, new Date(eventStartTime), new Date(eventEndTime), {description: eventDescription});
31     sheet.getRange(i + 1, 7).setValue(newEvent.getId()); // atualizar a planilha com o ID do novo evento
32   } else if (eventId === "") {
33     sheet.getRange(i + 1, 7).setValue(existingEvent.getId()); // atualizar a planilha com o ID do evento existente
34   }
35 }
36 }
37
```

- A linha 7 inicia um loop **for** que percorre cada linha dos dados da planilha a partir da segunda linha, ou seja, a partir do índice 1 (já que a primeira linha normalmente contém os títulos das colunas).

- A linha 8 verifica se a data na coluna 5 está vazia. Se estiver, a execução passa para a próxima iteração do loop usando o comando **continue**.
- As linhas 9,10 e 11 obtêm as datas do evento, o nome do evento e a descrição do evento, contidas nas colunas da planilha e armazena nas suas respectivas variáveis, **lembre-se que o índice das colunas na variável data começam no índice 0.**
- As linhas 12 e 13 definem o horário de início do evento e o horário de término.
- A linha 14 armazena o ID do evento criado na variável **“eventId”** e preenche a linha atual na coluna 6 da planilha com esse ID para que não sejam criados eventos duplicados.
- As linhas 17 cria uma variável booleana chamada **“eventsExists”**
- A linha 18 cria uma variável chamada **“existingEvent”** para o evento existente, se houver.
- A linha 19 cria uma variável chamada **“events”** que usa o método **“getEventsForDay”** para obter todos os eventos que ocorreram na data especificada pelo evento em questão
- Nas linhas 20 a 35 o código entra em um loop "for" que percorre cada evento e verifica se o título e a descrição do evento são iguais aos do evento em questão. Se um evento com o mesmo título e descrição for encontrado, a variável **"eventExists"** é definida como **true** e a variável "existingEvent" é definida como o evento encontrado. O loop é interrompido usando o comando "break", porque não há necessidade de continuar verificando outros eventos se já encontramos um evento existente com as mesmas informações. Se nenhum evento existente for encontrado, a variável **"eventExists"** permanecerá como **false** e o código criará um novo evento com o método **"createEvent"** e alocará o novo ID de evento para a célula na coluna 6 da linha atual na planilha. Caso contrário, se um evento existente for encontrado, o código verificará se o ID do evento já está armazenado na planilha na coluna 6. Se não houver um ID de evento, o código alocará o ID do

evento existente na célula da coluna 6 da linha atual na planilha. Isso garante que a planilha tenha sempre o ID mais atualizado para cada evento existente ou novo que foi criado.

FUNCIONAMENTO DO CÓDIGO gerarRelatorioSemanal

O código **gerarRelatorioSemanal** é responsável por ler a agenda "**Controle de EPIs",** identificar os EPIS que vencem na semana atual, gerar um relatório em forma de uma tabela HTML e mandar este relatório por Email.

```
1  function gerarRelatorioSemanal() {
2    var hoje = new Date();
3    var primeiroDiaSemana = new Date(hoje.getFullYear(), hoje.getMonth(), hoje.getDate() - hoje.getDay() + 1);
4    var ultimoDiaSemana = new Date(hoje.getFullYear(), hoje.getMonth(), primeiroDiaSemana.getDate() + 6);
5    var calendario = CalendarApp.getCalendarById('38b12949f185239b78e2a2dcb654d4d674d036ee055f00583c12e550deb6a179@group.calendar.google.com');
6    var eventosSemanaAtual = calendario.getEvents(primeiroDiaSemana, ultimoDiaSemana);
```

- A linha 1 define a função gerarRelatorioSemanal.
- A linha 2 cria uma variável chamada **"hoje"** que cria um objeto Date e atribui a ele a data e hora atual do sistema. Essa variável será usada para criar a linha de assunto do e-mail.
- A linha 3 cria um objeto "**Date"** e atribui a ele a data do primeiro dia da semana atual. Ele faz isso subtraindo o número de dias passados desde o início da semana **(hoje.getDay())** do dia atual **(hoje.getDate())** e adicionando 1 para chegar ao primeiro dia da semana (+ 1). O método **getMonth()** é usado para obter o mês do objeto Date criado anteriormente. Essa variável será usada para obter os eventos do calendário para a semana atual.
- A linha 4 cria um objeto "**Date"** e atribui a ele a data do último dia da semana atual. Ele faz isso adicionando 6 dias (+ 6) à data do primeiro dia da semana **(primeiroDiaSemana.getDate())** e usando o mesmo mês e ano do objeto date criado anteriormente.
- A linha 5 define uma variável calendario e atribui a ela o calendário do Google Agenda a ser usado para buscar eventos. O ID do calendário é fornecido como uma string.
- A linha 6 usa o método **getEvents()** do objeto Calendar para buscar eventos no calendário para a semana atual. Os argumentos são os objetos **Date** criados anteriormente **("primeiroDiaSemana"** e **"ultimoDiaSemana")**.

```
8   var html = "<table><tr><th>Data</th><th>Nome</th><th>EPI</th></tr>";
9   eventosSemanaAtual.forEach(function(evento) {
10    var dataEvento = evento.getStartTime().toLocaleDateString("pt-BR");
11    var nomeEvento = evento.getTitle();
12    var epiEvento = evento.getDescription();
13
```

- A linha 8 define uma variável **“html”** e atribui a ela o início de uma tabela HTML. A tabela terá três colunas: Data, Nome e EPI.
- A linha 9 começa um loop **forEach()** que iterará sobre os eventos da semana atual.
- A linha 10 cria uma variável “**dataEvento”** e atribui a ela a data do início do evento formatada como uma **string** de data no formato "dd/mm/yyyy". A função **toLocaleDateString()** é usada para formatar a data.
- A linha 11 cria uma variável **“nomeEvento”** e atribui a ela o nome do evento.
- A linha 12 obtém a descrição do evento atual e armazena-a na variável **“epiEvento”**.

```
14    // Verifica se a hora de início é anterior à hora de término
15    if (evento.getStartTime() >= evento.getEndTime()) {
16      // Se não, mostra uma mensagem de erro na tabela HTML
17      html += "<tr style='color:red;'><td>" + dataEvento + "</td><td>" + nomeEvento + "</td><td>Erro: a hora de início é posterior à hora de
        término.</td></tr>";
18    } else {
19      // Se sim, adiciona a linha normalmente
20      html += "<tr><td>" + dataEvento + "</td><td>" + nomeEvento + "</td><td>" + epiEvento + "</td></tr>";
21    }
22  });
23  html += "</table>";
24
25  MailApp.sendEmail({
26    to: 'suporte.comercial@wtavet.com.br,inovacao@wtavet.com.br,administrativo@wtavet.com.br,administrativo1@wtavet.com.br,negocios@wtavet.com.br',
27    subject: 'Grupo WTA: Relatório Semanal de Vencimento de EPIS '+ hoje,
28    htmlBody: html
29  });
30  }
31
```

- Nas linhas 15 – 21 a função **getStartTime** é usada para obter a hora de início do evento e **getEndTime** é usada para obter a hora de término do evento. Esses dois valores são comparados usando o operador >= para verificar se a hora de início é maior ou igual à hora de término. Se a condição no **if** for verdadeira, ou seja, se a hora de início for maior ou igual à hora de término, o código dentro do bloco **if** será executado. Essa seção adiciona uma nova linha à tabela HTML com uma mensagem de erro indicando que a hora de início é posterior à hora de término. A cor da linha é definida como vermelho para indicar um erro.

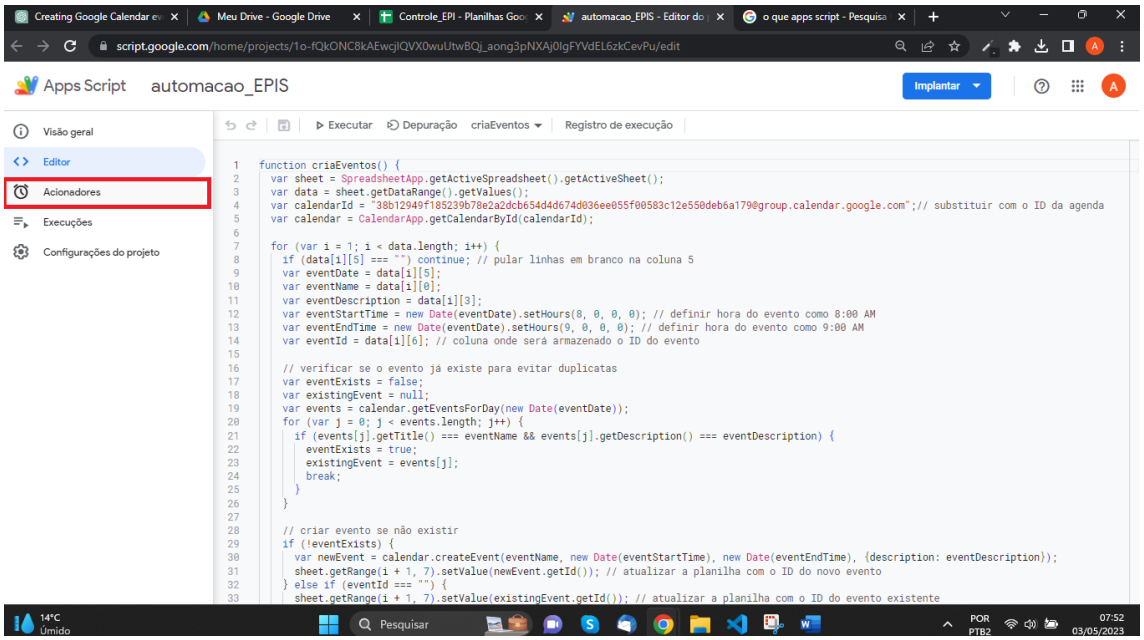
Se a condição no **if** for falsa, ou seja, se a hora de início for menor do que a hora de término, o código dentro do bloco **else** será executado. Essa seção adiciona uma nova linha à tabela HTML com as informações do evento atual, incluindo a data, o título e a descrição. (**Verificação para prevenir erros).**

```
25  MailApp.sendEmail({
26    to: 'suporte.comercial@wtavet.com.br,inovacao@wtavet.com.br,administrativo@wtavet.com.br,administrativo1@wtavet.com.br,negocios@wtavet.com.br',
27    subject: 'Grupo WTA: Relatório Semanal de Vencimento de EPIS '+ hoje,
28    htmlBody: html
29  });
30  }
31
```

- As linhas 25 - 29 são responsáveis por mandar o Email com o relatório, o campo **“to:”** contém os E-mails dos destinatários, o campo **“subject:”** contém o assunto do Email, o campo **“htmlBody:”** especifica o tipo do arquivo contido no corpo do Email.

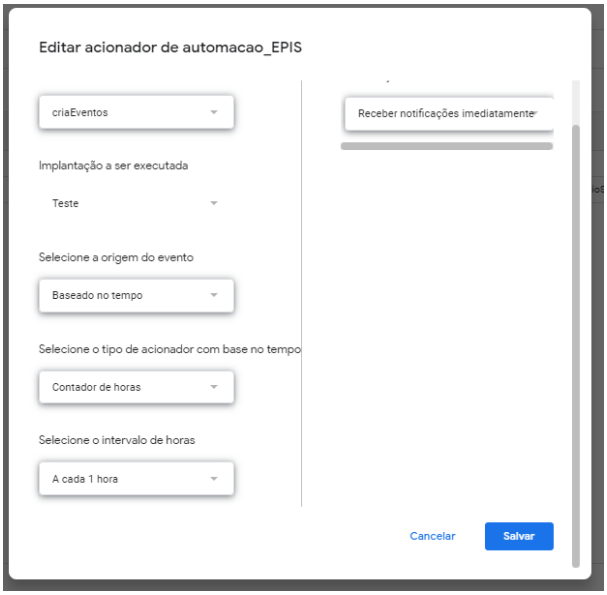
ACIONADORES

Para definir a frequência que os códigos devem ser executados utilizei os acionadores na plataforma **“Apps Script”**



O código **“criaEventos”** está com um acionador configurado para que ele seja executado a cada hora. Desta maneira ele sempre vai manter a agenda atualizada, pois a cada hora ele vai ler a planilha e caso a planilha tenha sido atualizada ele criará novos eventos na agenda.

Configuração do acionador:



O código **gerarRelatorioSemanal** está configurado para que seja executado todos os domingos entre as 8h e as 9h. Dessa forma será gerado e enviado por e-mail um relatório dos vencimentos dos EPIs toda semana.

Configuração do acionador: