**Sumário**

[INTRODUÇÃO 1](#_Toc134175162)

[GOOGLE APPS SCRIPT 4](#_Toc134175163)

[FUNCIONAMENTO DO CÓDIGO criaEventos 6](#_Toc134175164)

[FUNCIONAMENTO DO CÓDIGO gerarRelatorioSemanal 8](#_Toc134175165)

[ACIONADORES 10](#_Toc134175166)

# INTRODUÇÃO

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

**GOOGLE SHEETS**

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamenteA 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.**

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Texto

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

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

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

**Tela de computador

Descrição gerada automaticamente**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”**;

**Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente**

Tela de computador

Descrição gerada automaticamenteClique na aba **“Extensões”** e clique em **“Apps Script”**.

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamenteAqui 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.

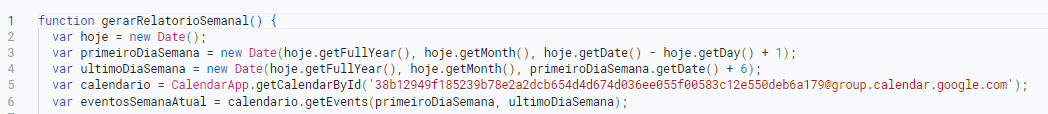
**Uma imagem contendo Texto

Descrição gerada automaticamente**

* A linha 1 define a função “**criaEventos**”.
* A linha 2 obtém a planilha ativa atual, no caso a planilha **“Contrlole\_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”**.
* Interface gráfica do usuário, Texto, Aplicativo

  Descrição gerada automaticamenteA 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’**.
* 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.

* 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

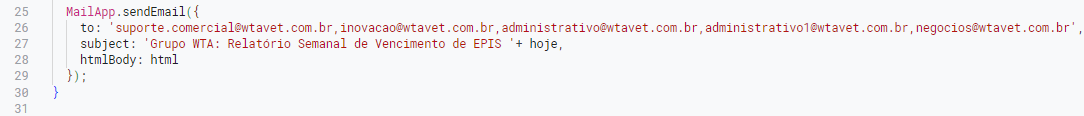
**“ultimoDiaSemana”)**.

Texto

Descrição gerada automaticamente

* 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.
* **Interface gráfica do usuário, Texto, Aplicativo

  Descrição gerada automaticamente com confiança média**A linha 12 obtém a descrição do evento atual e armazena-a na variável **“epiEvento”**.
* 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).**

* 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

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamentePara 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.

**Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamenteConfiguração 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.

**Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamenteConfiguração do acionador:**