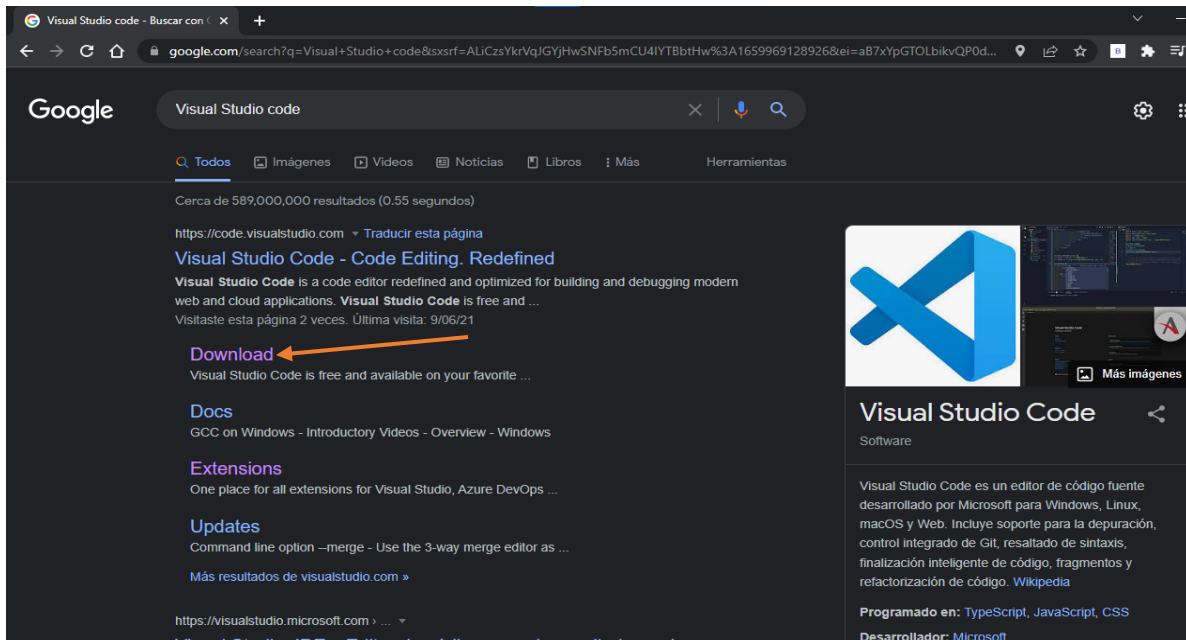
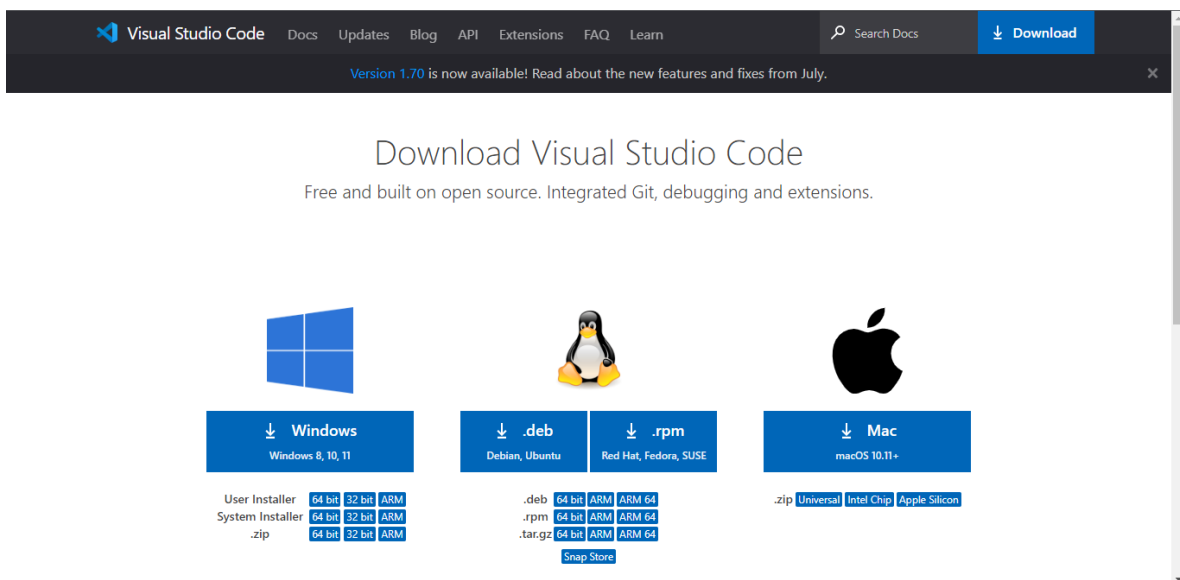


Proceso instalación Visual Studio Code

Paso 1: Escribimos en Google Visual Studio Code y seleccionamos donde dice “Download”.



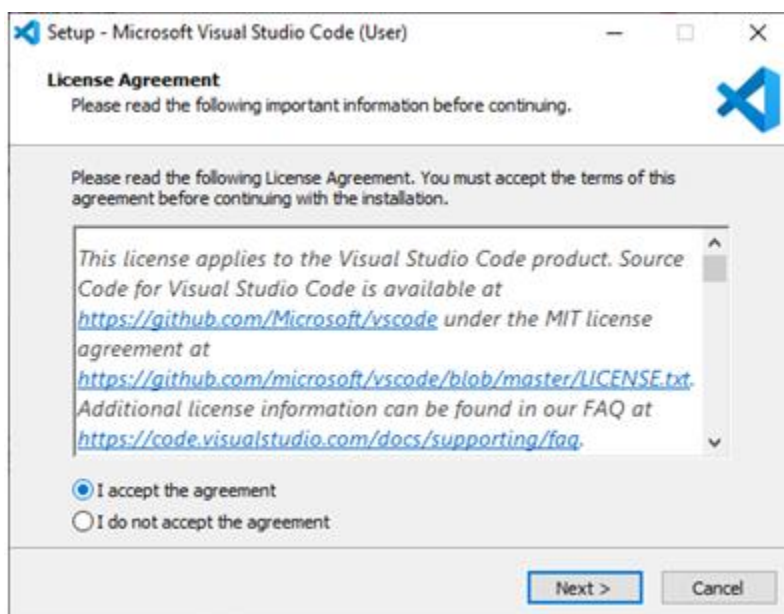
Paso 2: Seleccionamos el sistema operativo que tenemos y lo descargamos.



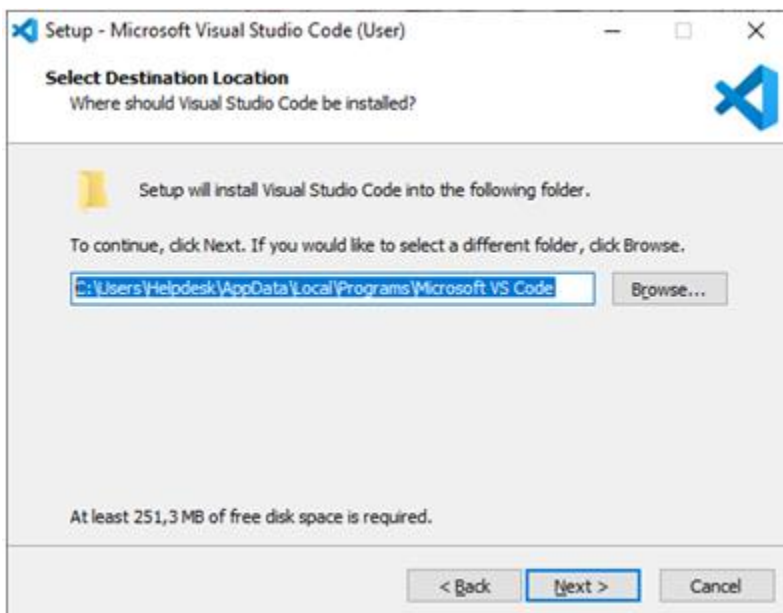
Paso 3: Al darle clic nos descargará un .exe, al cual le daremos clic encima.



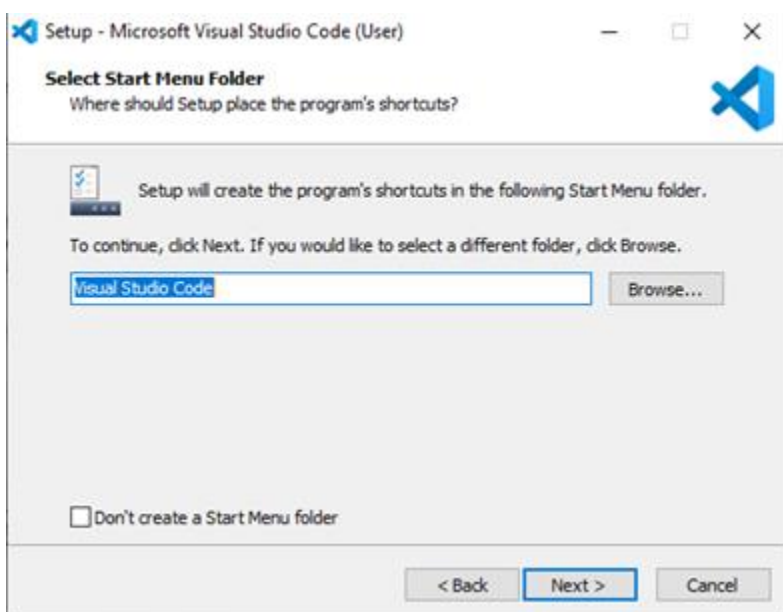
Paso 4: Lee y acepta el acuerdo de licencia. Haz clic en Next para continuar.



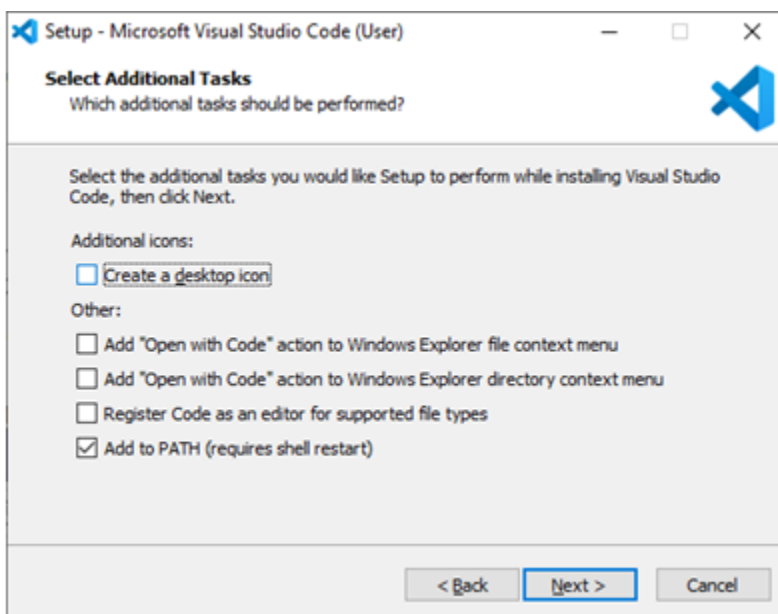
Paso 5: Puedes cambiar la ubicación de la carpeta de instalación o mantener la configuración predeterminada. Haz clic en Next para continuar.



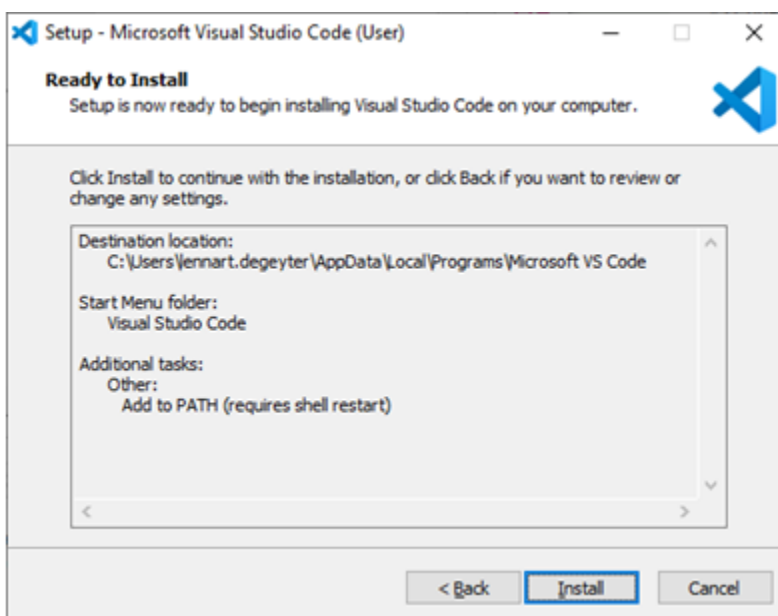
Paso 6: Elige si deseas cambiar el nombre de la carpeta de accesos directos en el menú Inicio o si no deseas instalar accesos directos en absoluto. Haz clic en Next.



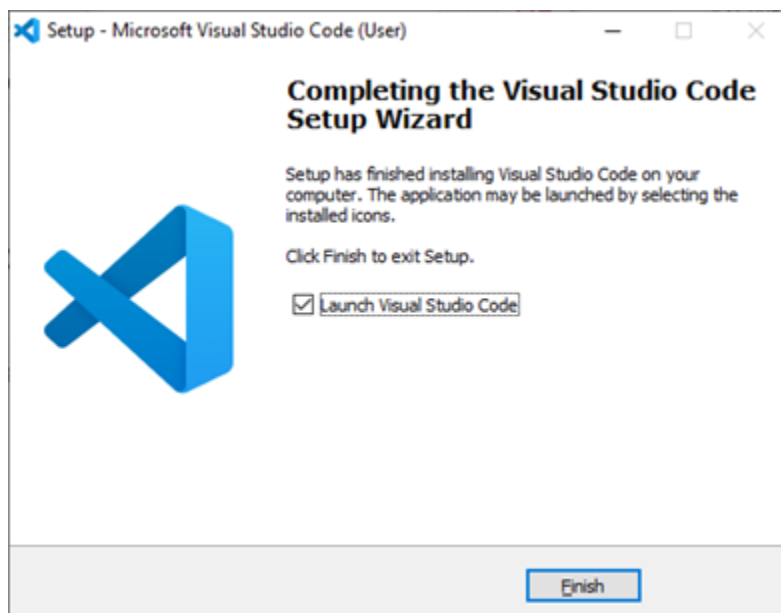
Paso 7: Selecciona las tareas adicionales, por ej. crear un icono en el escritorio o añadir opciones al menú contextual de Windows Explorer. Haz clic en Next.



Paso 8: Haz clic en Install para iniciar la instalación.

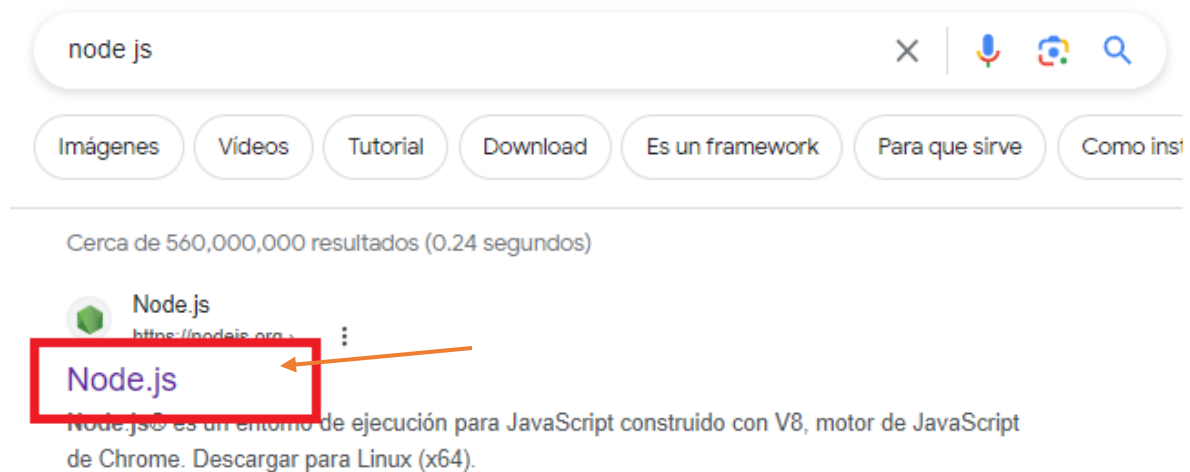


Paso 9: El programa está instalado y listo para usar. Haz clic en Finish para finalizar la instalación y lanzar el programa.



Proceso instalación Node JS

Paso 1: Escribimos en Google node js y seleccionamos la primera opción.



Paso 2: Ahora vamos a seleccionar la opción “LTS” ya que es más estable que la nueva

Node.js® es un entorno de ejecución para JavaScript construido con V8, motor de JavaScript de Chrome.

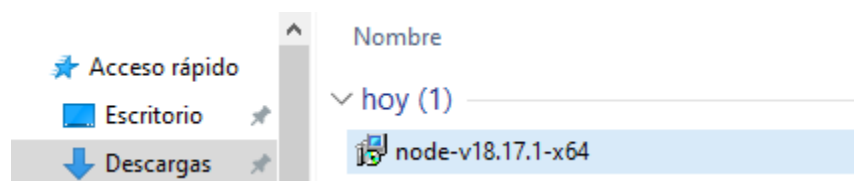
Descargar para Windows (x64)



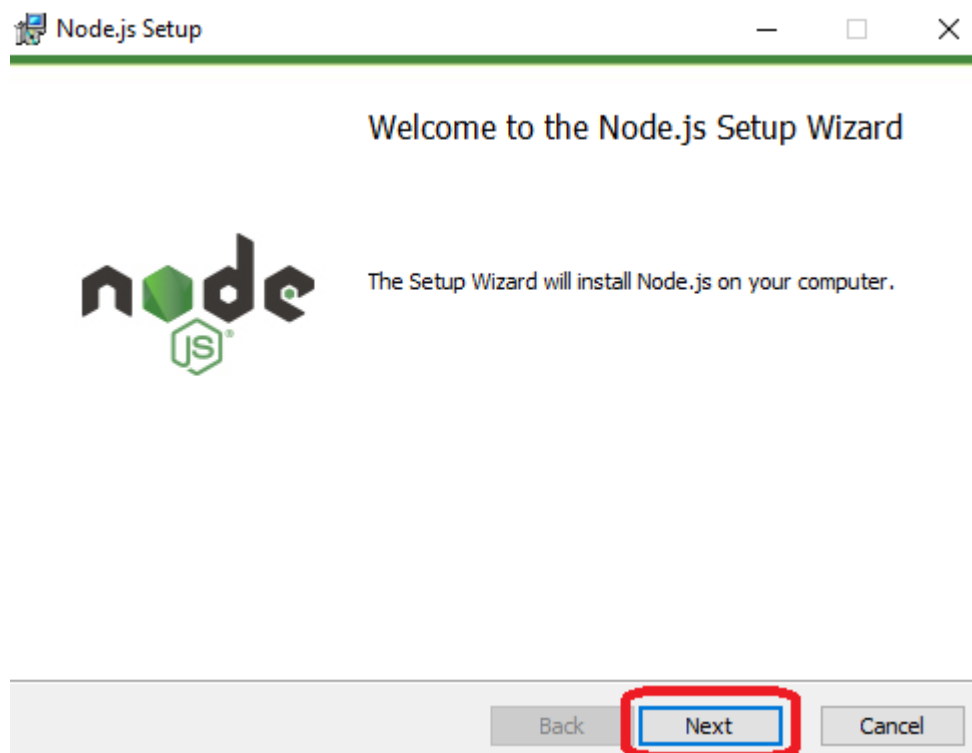
[Otras Descargas](#) | [Cambios](#) | [Documentación de la API](#) [Otras Descargas](#) | [Cambios](#) | [Documentación de la API](#)

O eche un vistazo al [Programa de soporte a largo plazo \(LTS\)](#).

Paso 3: Buscaremos en las descargas el archivo que hemos descargado y le damos doble click.



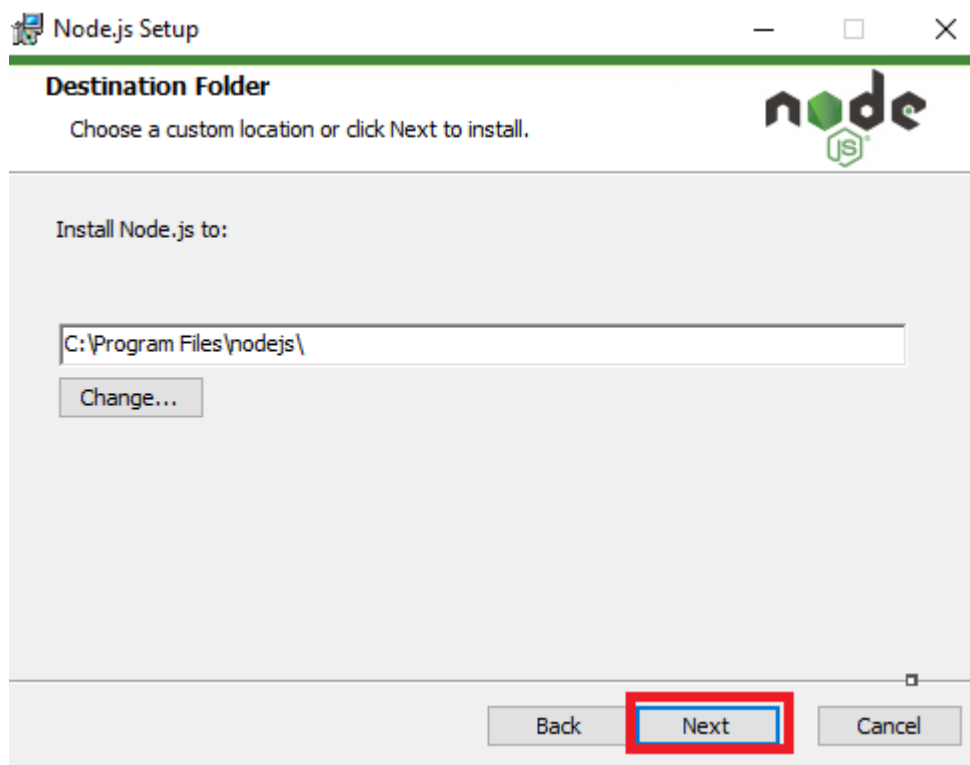
Paso 4: Le damos click en “Next”



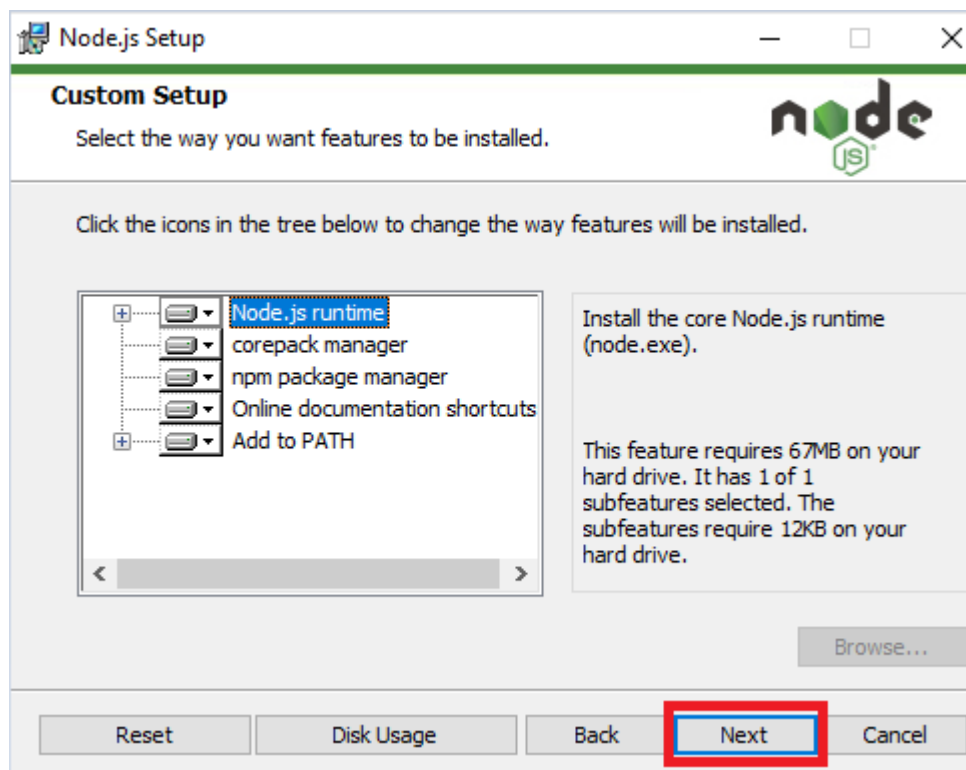
Paso 5: Lee los términos y condiciones le damos click en aceptar y le damos “next”.



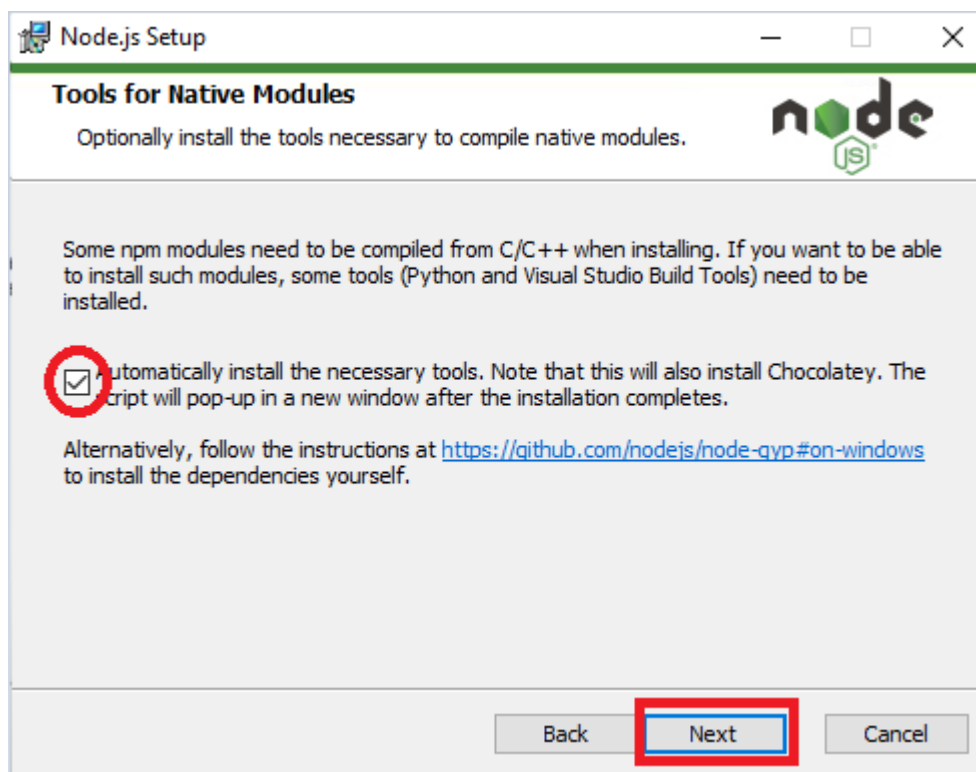
Paso 6: Le damos click en “Next” de nuevo



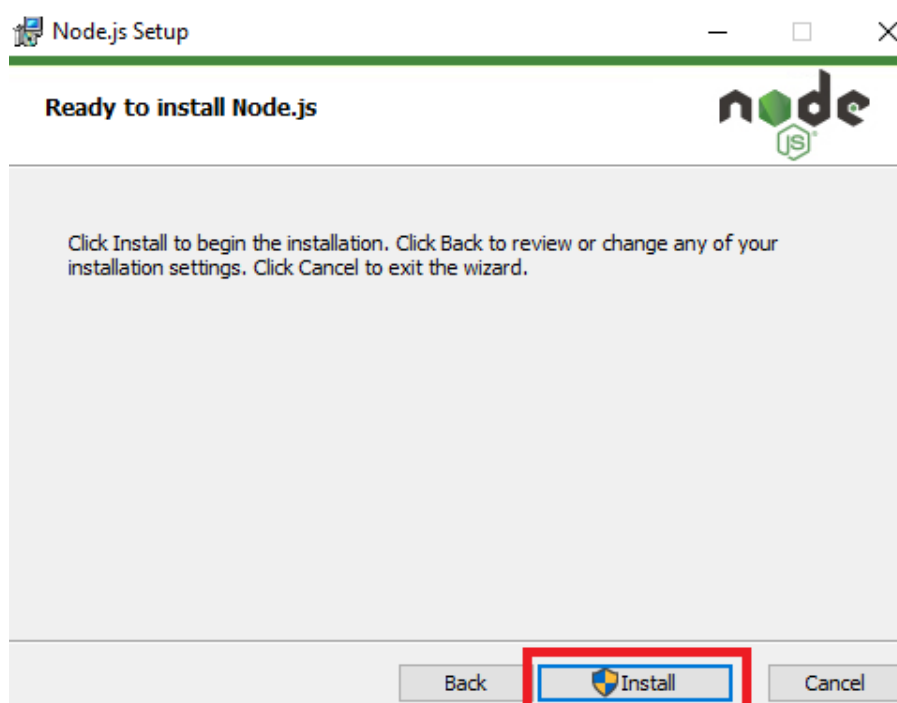
Paso 7: Ahora de acá no vamos a mover nada y simplemente daremos “Next”



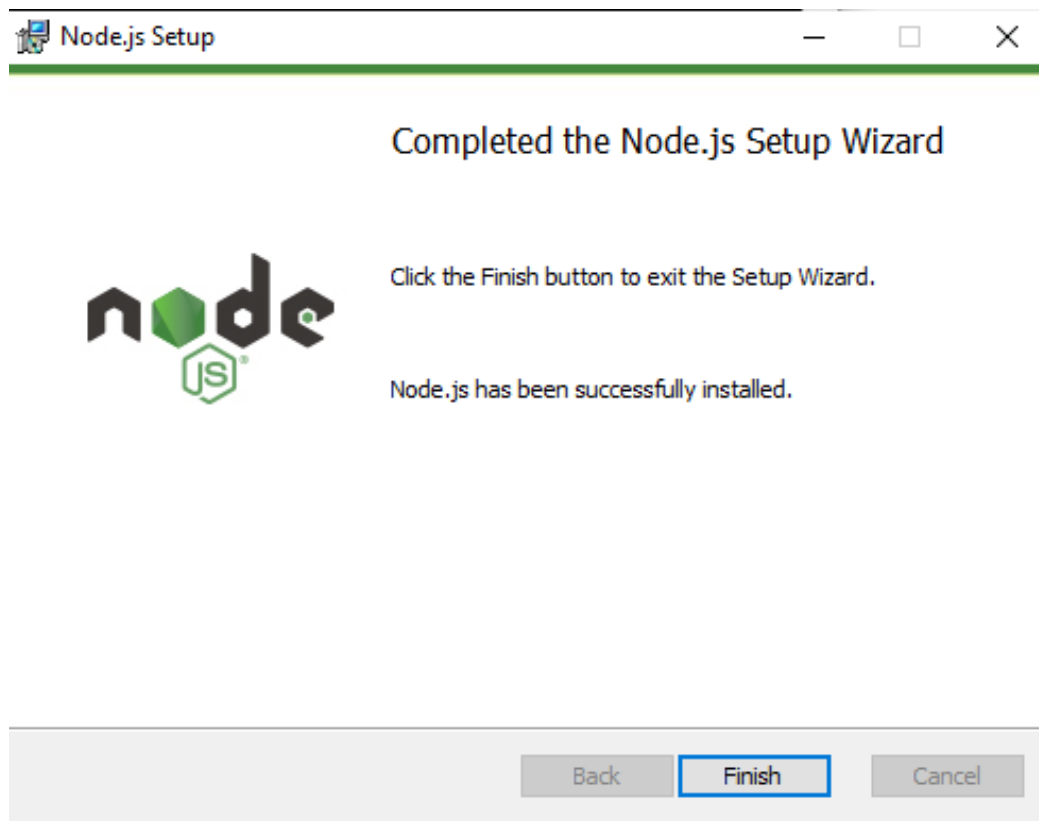
Paso 8: Ahora vamos a instalar algunos módulos de “npm” chuleando la opción que nos da y luego damos “Next”



Paso 9: Ahora damos click en “Install”



Paso 10: Con esto hemos instalado “Node js” en caso de que te salgan más pestañas como les suele pasar a algunos equipos para instalar simplemente a todo le das enter y se instalará correctamente



Creación de archivos y carpetas

Paso 1: Creamos una carpeta raíz. En nuestro caso la llamaremos “calendariogoogle”.

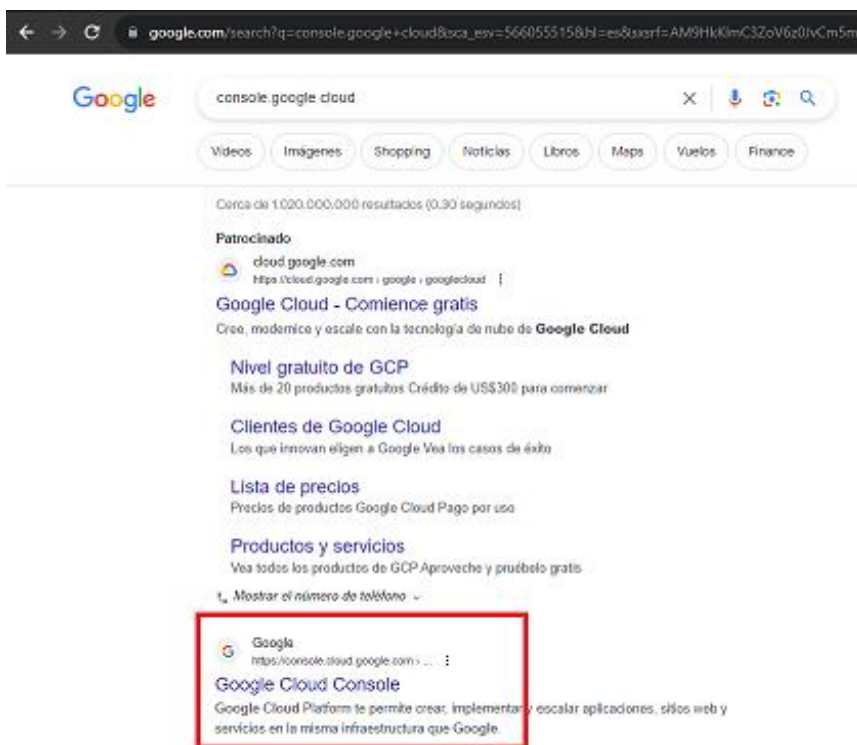
 calendariogoogle

Calendario De Google

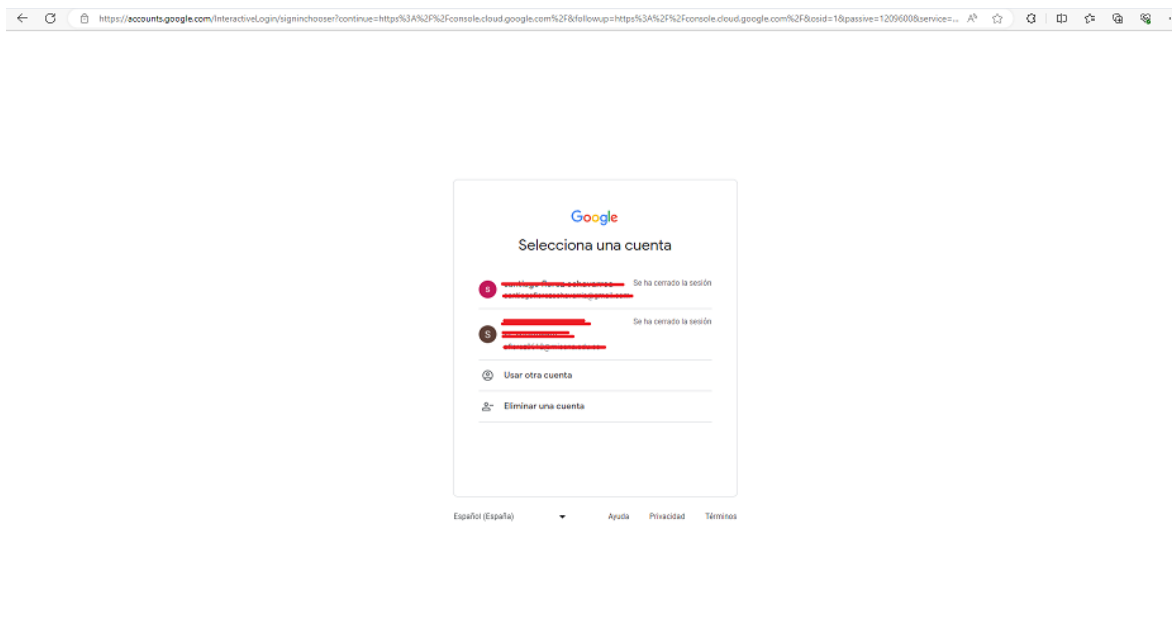
Paso 1: Vamos a comenzar yendo a google.



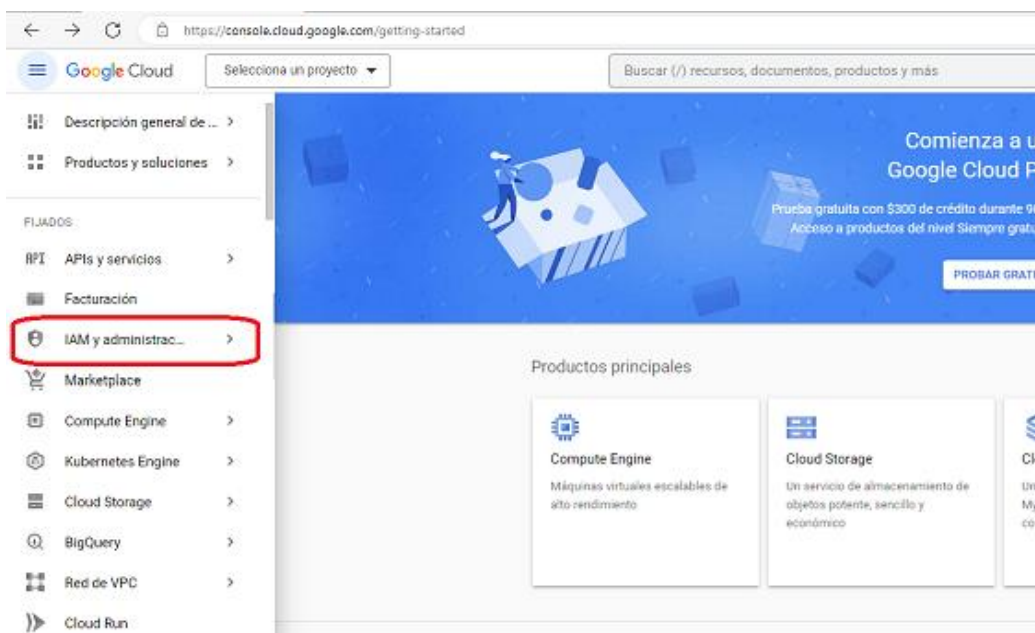
Paso 2: Buscaremos console.google cloud y escogemos la señalada por la imagen



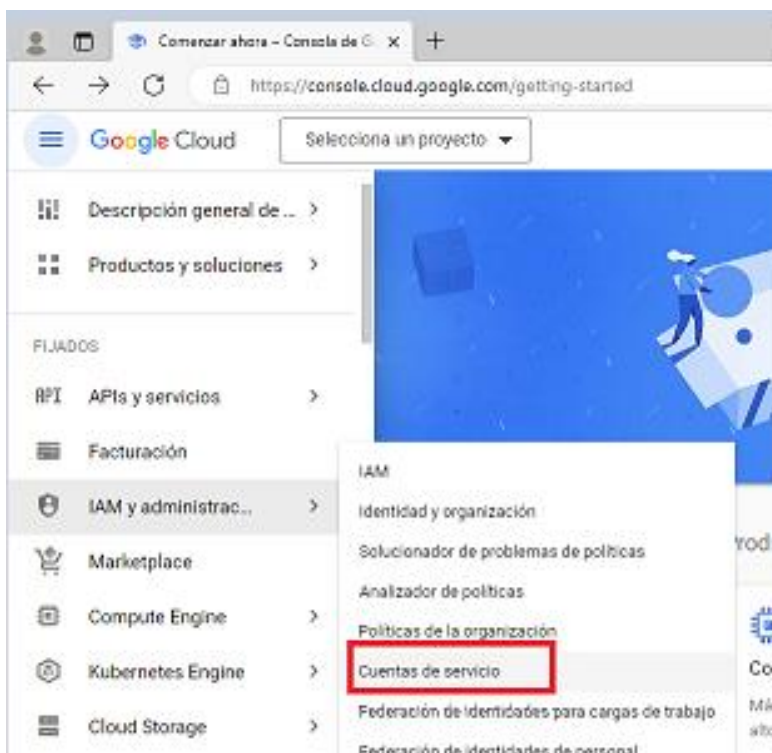
Paso 3: Nos pedirá iniciar sesión con Google así que lo haremos



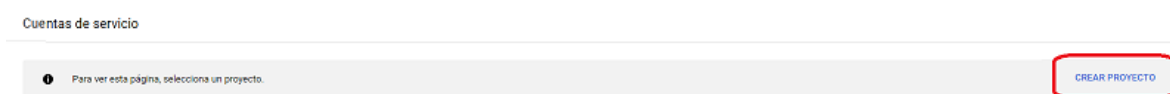
Paso 4: Ahora nos mostrara esta pestaña he iremos a “IAM y administración”



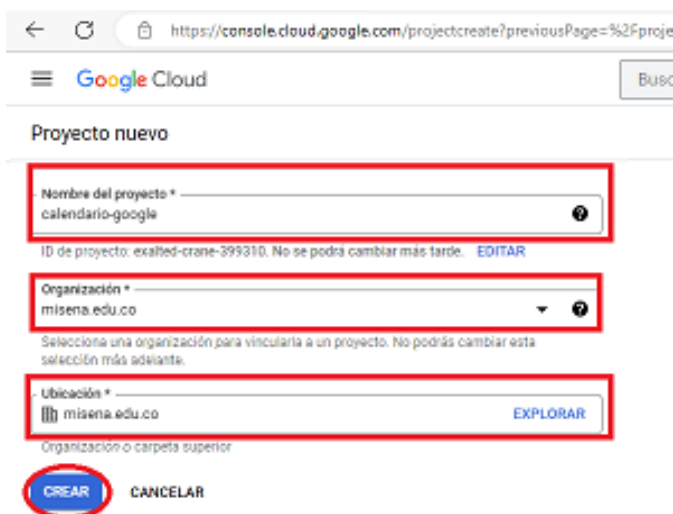
Paso 5: Y seleccionamos la opción de “Cuentas de servicio”



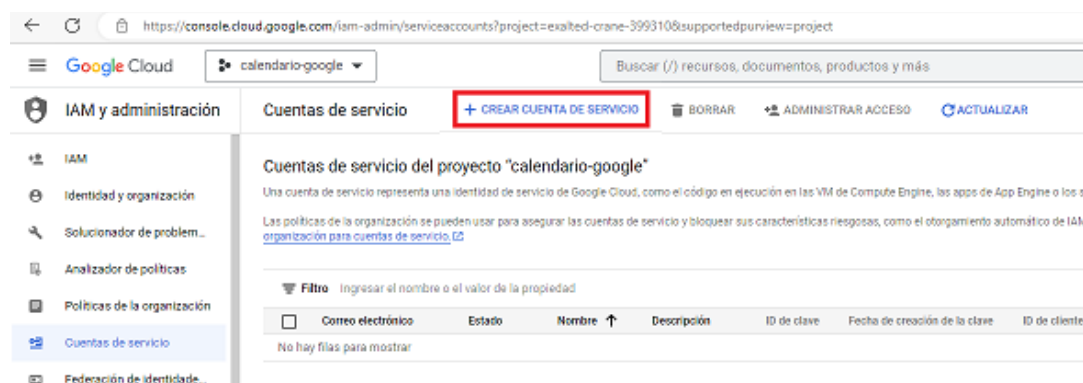
Paso 6: Nos mostrará esta pestaña y seleccionamos “crear proyecto”



Paso 7: Llenamos la información solicitada y le damos “Crear”



Paso 8: Ahora nos sale esta pestaña y seleccionamos “crear cuenta de servicio”



Paso 9: Ahora llenamos de nuevo la información solicitada y seleccionamos “CREAR Y CONTINUAR”

The screenshot shows the 'Crear cuenta de servicio' (Create Service Account) wizard. The first step, 'Detalles de la cuenta de servicio', is active. It contains the following fields:

- Nombre de la cuenta de servicio:** api-calendario
- ID de la cuenta de servicio:** api-calendario
- Dirección de correo electrónico:** api-calendario@exalted-crane-399310.iam.gserviceaccount.com
- Descripción de la cuenta de servicio:** Este es un servicio para probar el calendario de google con api

The 'CREAR Y CONTINUAR' button is highlighted with a red box. Below the details section, there are two optional steps:

- Otorga a esta cuenta de servicio acceso al proyecto (opcional)
- Otorga a usuarios acceso a esta cuenta de servicio (opcional)

At the bottom, there are 'LISTO' and 'CANCELAR' buttons.


Paso 10: Ahora vamos a seleccionar la opción de “selecciona un rol”

← Crear cuenta de servicio

✓ **Detalles de la cuenta de servicio**

2 **Otorga a esta cuenta de servicio acceso al proyecto (opcional)**

Otorga a esta cuenta de servicio acceso a calendario-google a fin de que tenga permiso para completar acciones específicas en los recursos de tu proyecto. [Más información](#)

selecciona un rol Condición de IAM (opcional) ? 

+ AGREGAR CONDICIÓN DE IAM

+ AGREGAR OTRO ROL

CONTINUAR

3 **Otorga a usuarios acceso a esta cuenta de servicio (opcional)**

LISTO CANCELAR

Paso 11: Escogemos “propietario”

Filtro | Escribir para filtrar

Acceso rápido	Roles
Usadas actualmente	Editor
Básico	Navegador
Por producto o servicio	Propietario
Acceso a VPC sin servidores	Visualizador
Acceso Context Manager	

ADMINISTRAR FUNCIONES

Propietario

Acceso completo a la mayoría de los recursos de Google Cloud. Consulta la lista de permisos incluidos.

Paso 12: Ahora damos “continuar”

← Crear cuenta de servicio

✓ Detalles de la cuenta de servicio

2 Otorga a esta cuenta de servicio acceso al proyecto (opcional)

Otorga a esta cuenta de servicio acceso a calendario-google a fin de que tenga permiso para completar acciones específicas en los recursos de tu proyecto. [Más información](#)

Rol: Propietario Condición de IAM (opcional) + AGREGAR CONDICIÓN DE IAM

Acceso completo a la mayoría de los recursos de Google Cloud. Consulta la lista de permisos incluidos.

+ AGREGAR OTRO ROL

CONTINUAR

3 Otorga a usuarios acceso a esta cuenta de servicio (opcional)

LISTO CANCELAR

Paso 13: Ahora no llenamos nada de esta información y le damos “listo”

← Crear cuenta de servicio

✓ Detalles de la cuenta de servicio

✓ Otorga a esta cuenta de servicio acceso al proyecto (opcional)

3 Otorga a usuarios acceso a esta cuenta de servicio (opcional)

Grant access to users or groups that need to perform actions as this service account. [Learn more](#)

Función de los usuarios de cuentas de servicio

Otorga a los usuarios los permisos para implementar los trabajos y las VM con esta cuenta de servicio

Función de los administradores de cuentas de servicio

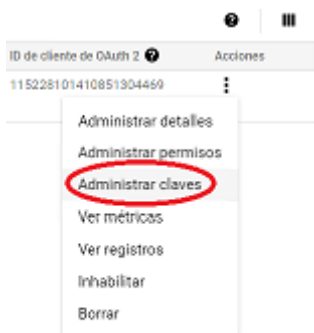
Otorga a los usuarios permisos para administrar esta cuenta de servicio

LISTO CANCELAR

Paso 14: Ahora podemos ver como se ve la cuenta que acabamos de crear. Vamos a seleccionar los tres puntos de la derecha.

Cuentas de servicio							
+ CREAR CUENTA DE SERVICIO BORRAR ADMINISTRAR ACCESO ACTUALIZAR APRENDIZAJE							
Cuentas de servicio del proyecto "calendario-google"							
Una cuenta de servicio representa una identidad de servicio de Google Cloud, como el código en ejecución en las VM de Compute Engine, las apps de App Engine o los sistemas que se ejecutan fuera de Google. Obtén más información sobre las cuentas de servicio.							
Las políticas de la organización se pueden usar para asegurar las cuentas de servicio y bloquear sus características riesgosas, como el otorgamiento automático de IAM, la creación y carga de claves, o la creación misma de cuentas de servicio. Obtén más información sobre las políticas de la organización para cuentas de servicio.							
Filtro Ingresar el nombre o el valor de la propiedad							
<input type="checkbox"/>	Comeo electrónico	Estado	Nombre ↑	Descripción	ID de clave	Fecha de creación de la clave	ID de cliente de OAuth 2.0
<input type="checkbox"/>	api-calendario@exalted-crane-999910.iam.gserviceaccount.com	Habilitado	api-calendario	Este es un servicio para probar el calendario de google con api	No hay claves		115228101410851304469
							Acciones

Paso 15: Ahora le damos “Administrar claves”



Paso 16: Seleccionamos “Agregar clave”



Paso 17: Y escogemos "Crear clave nueva"



Paso 18: Escogemos que la clave será tipo "JSON" y damos click en "CREAR"

Crear clave privada para "api-calendario"

Descarga un archivo que contiene la clave privada. Almacena el archivo en un lugar seguro, ya que no es posible recuperar la clave si se pierde.

Tipo de clave

☒ JSON

Recomendado

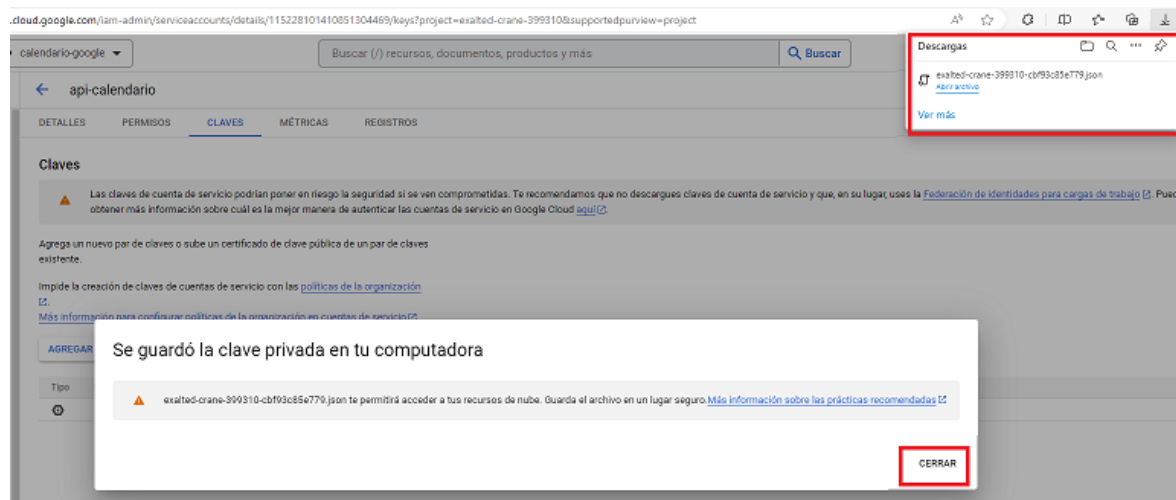
☐ P12

Para compatibilidad inversa con código en formato P12

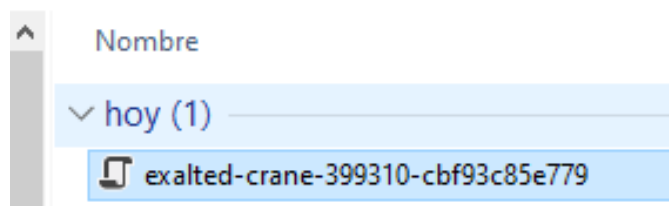
CANCELAR

CREAR

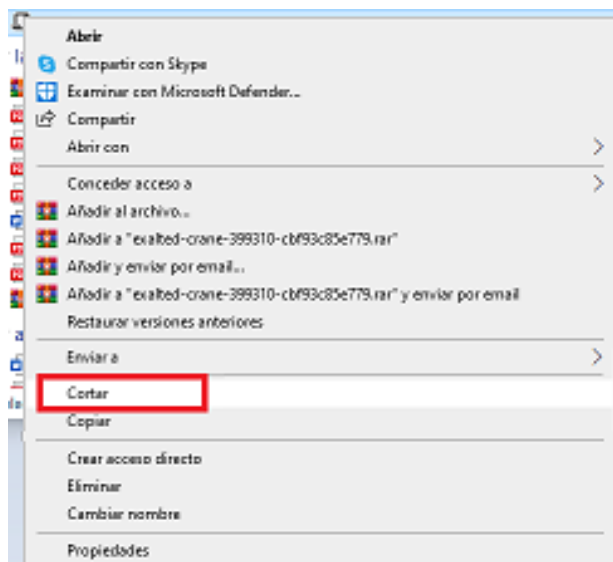
Paso 19: Una vez hemos creado la llave se te descargará un archivo en tu computador.



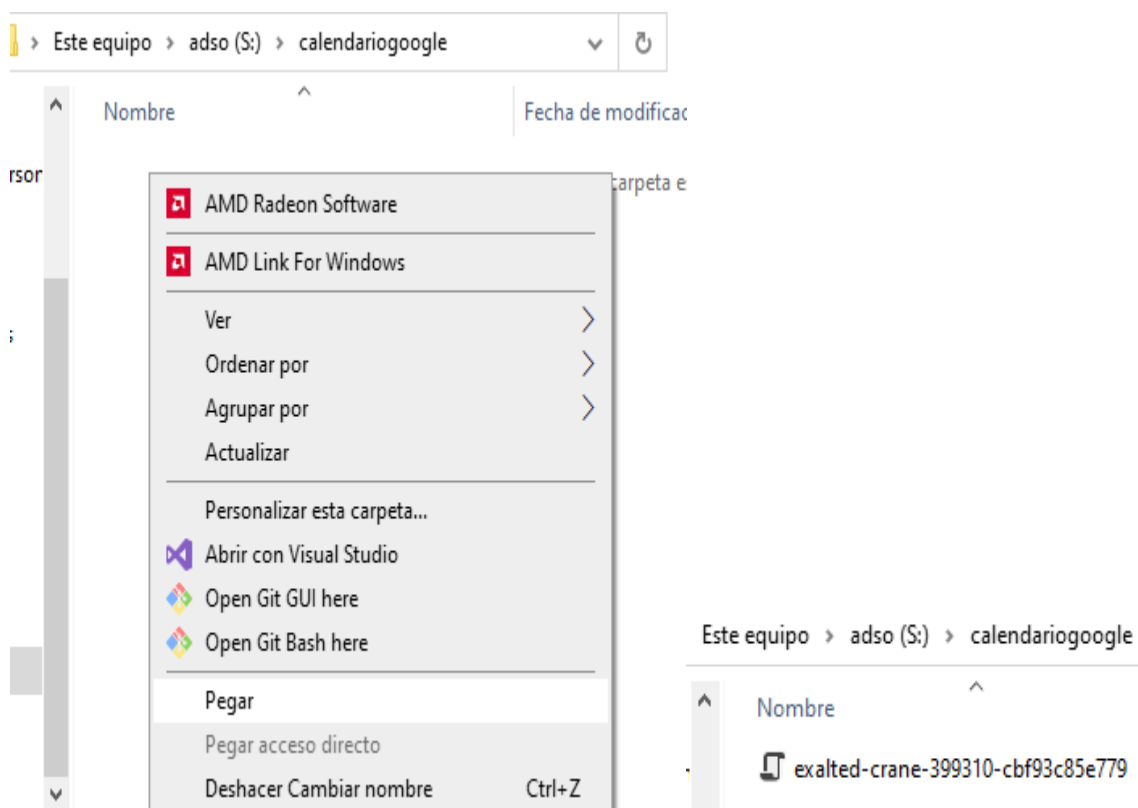
Este equipo > Descargas >



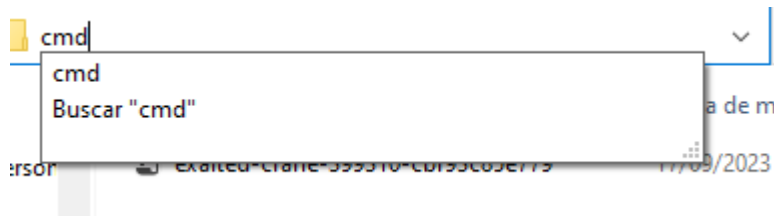
Paso 20: Ese archivo que se descargo vamos a cortarlo de la carpeta de descargas. Dándole click derecho al archivo y luego izquierdo en cortar



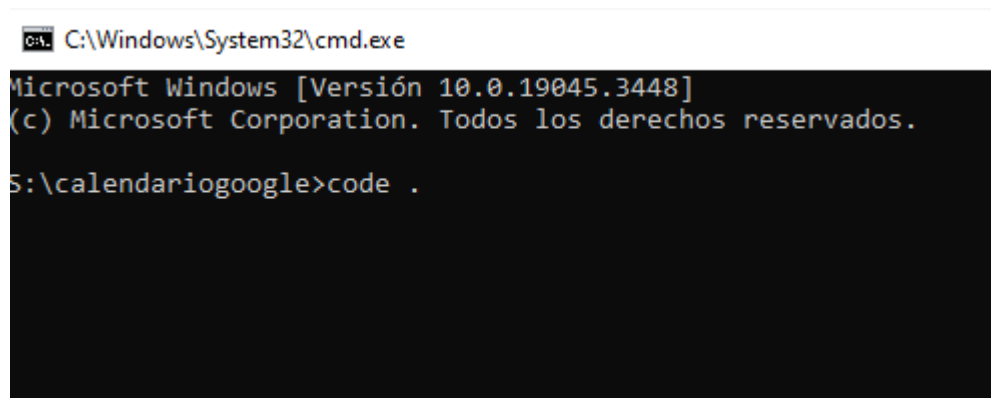
Paso 21: Iremos a nuestra carpeta raíz “calendariogoogle” y pegamos ahí el archivo.



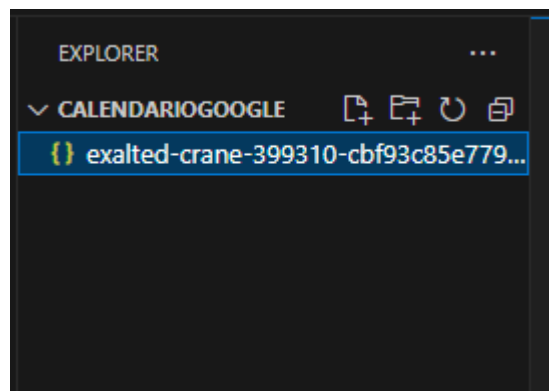
Paso 22: Para abrir el Visual Studio Code, haremos lo siguiente: Dentro de la carpeta raíz, en la barra superior, escribiremos cmd y le damos enter.



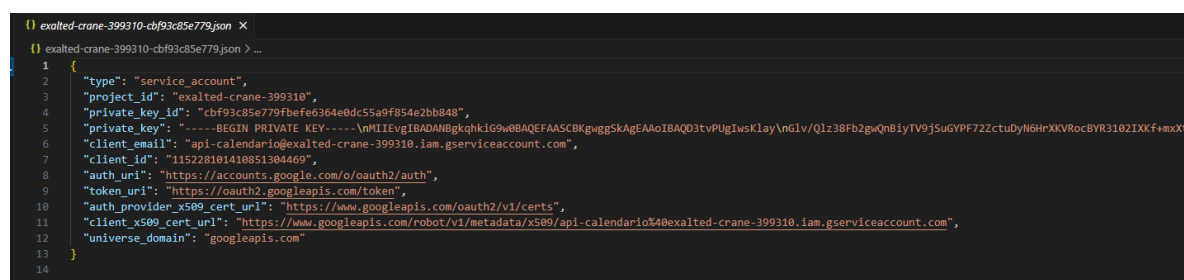
Paso 23: Esto nos abrirá la consola donde colocamos “code .” para abrir el Visual Studio Code



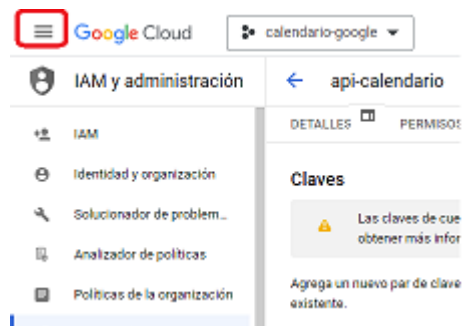
Paso 24: Ahora vemos el archivo que hemos puesto y lo seleccionamos



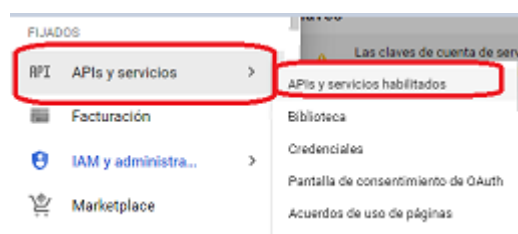
Paso 25: Así se ve actualmente nuestro archivo.



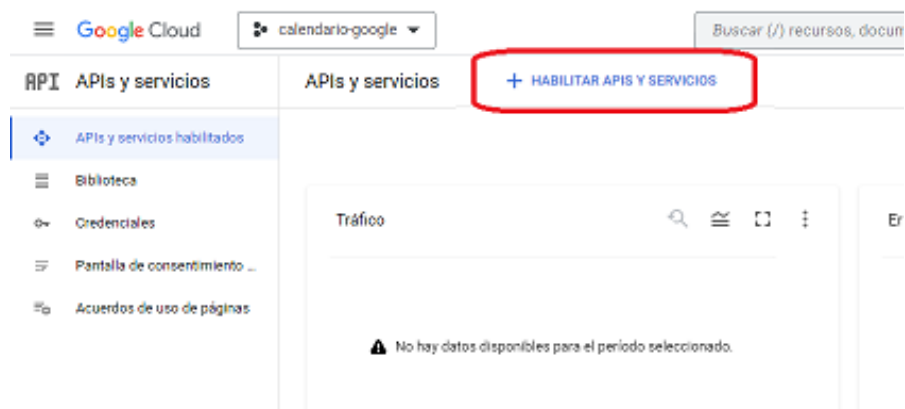
Paso 26: Ahora iremos a Google de nuevo y seleccionamos las tres líneas de arriba a la izquierda



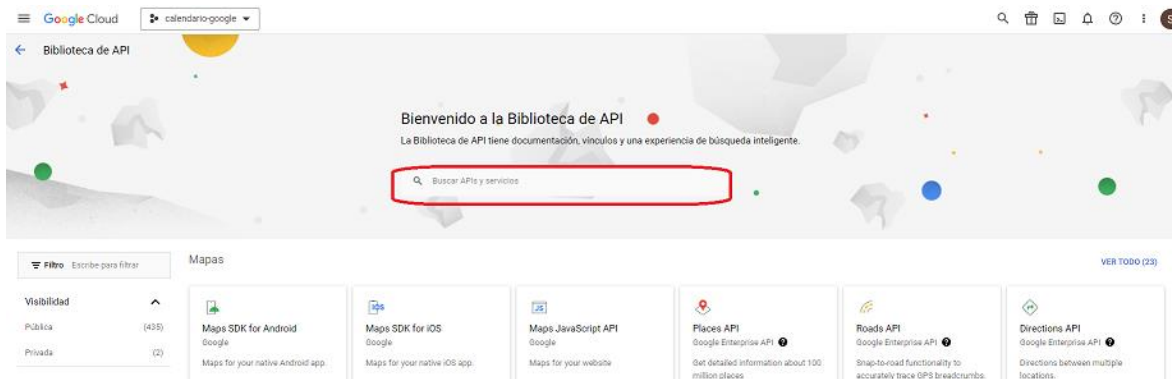
Paso 27: Ahora escogemos “APIs y servicios” y luego “APIS y servicios habilitados”



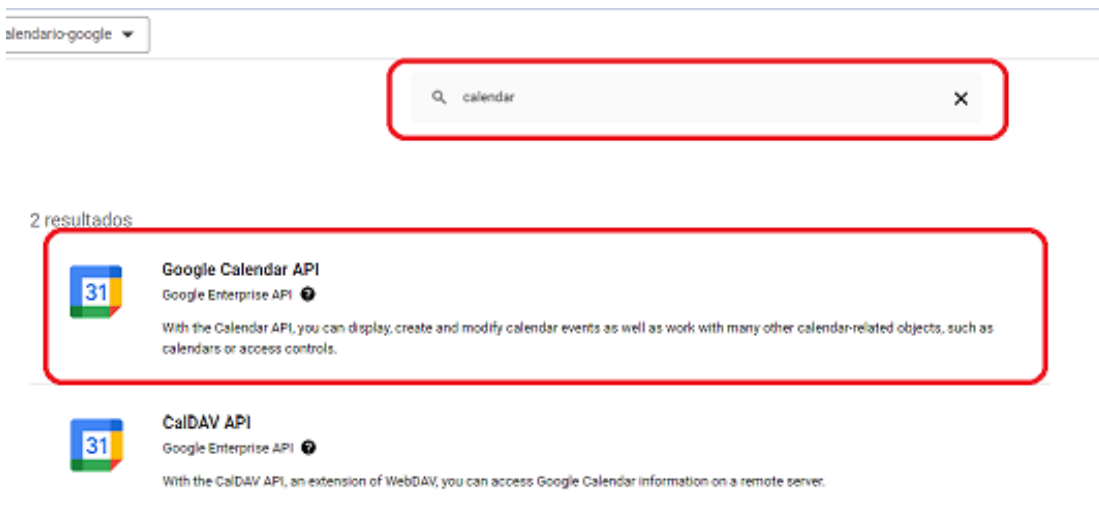
Paso 28: Nos aparece esta pestaña y seleccionamos “HABILITAR APIS Y SERVICIOS”



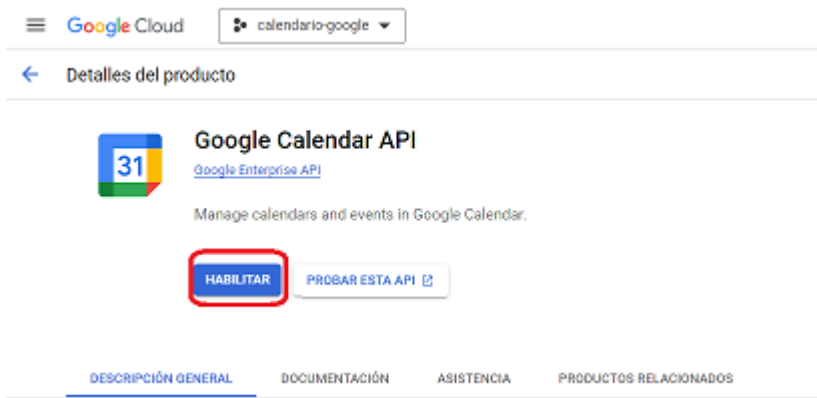
Paso 29: Ahora nos sale esta pestaña y vamos a la barra de búsqueda y colocamos “calendar”



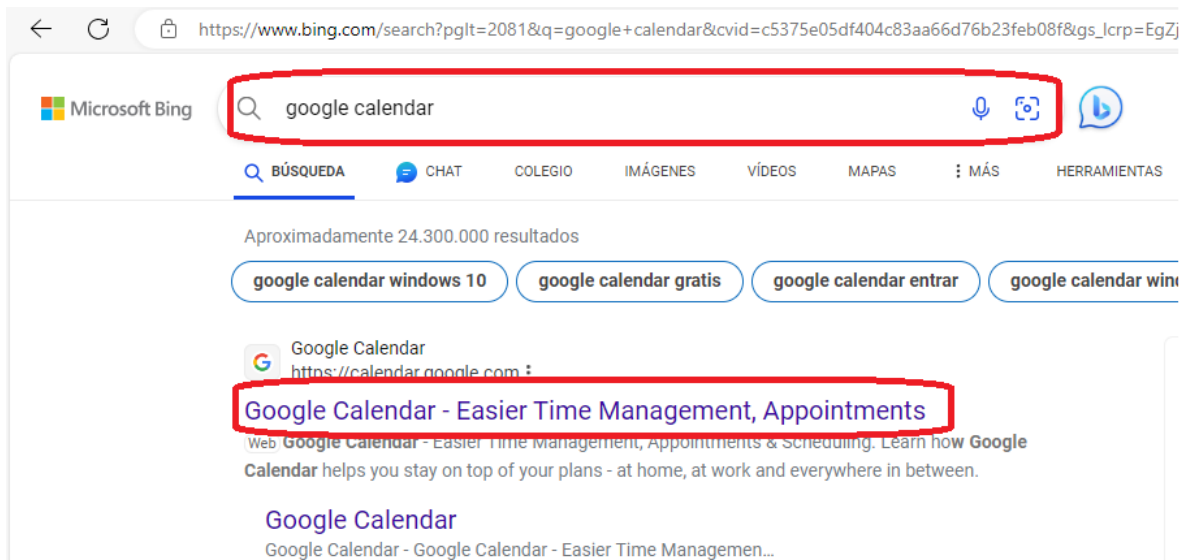
Paso 30: Ahora seleccionamos la opción de Google Calendar API



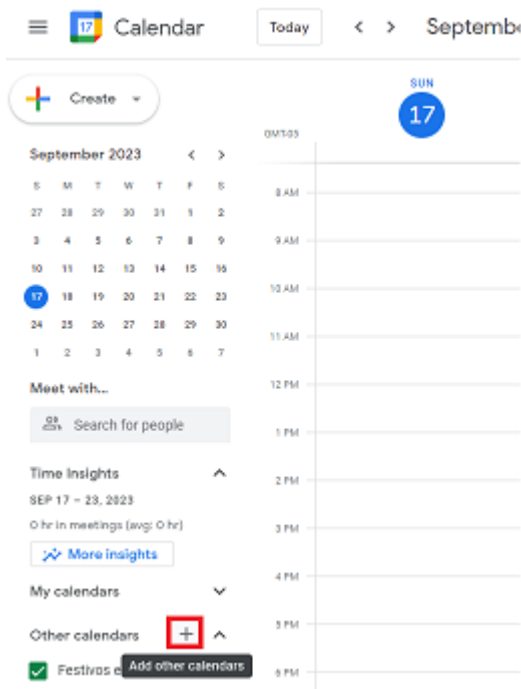
Paso 31: Nos sale esta api y le damos “habilitar” para habilitar la api en nuestro proyecto



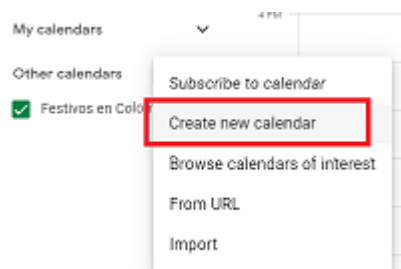
Paso 32: Una vez habilitada la Api volvemos a Google y buscamos “Google calendar” donde seleccionamos la primera opción



Paso 33: Nos aparecerá un calendario y vamos a la izquierda donde dice “other calendars” y escogemos el “+”



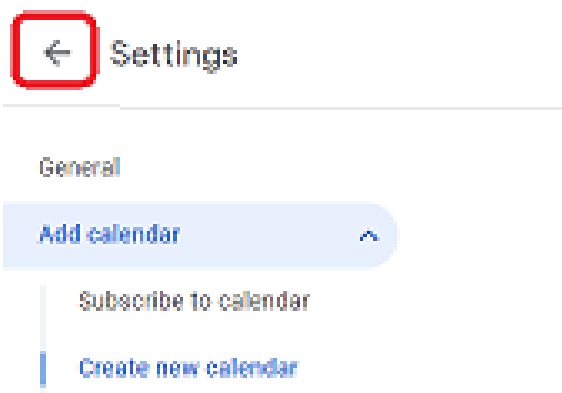
Paso 34: Ahora nos salen estas opciones y seleccionamos "Create new calendar"



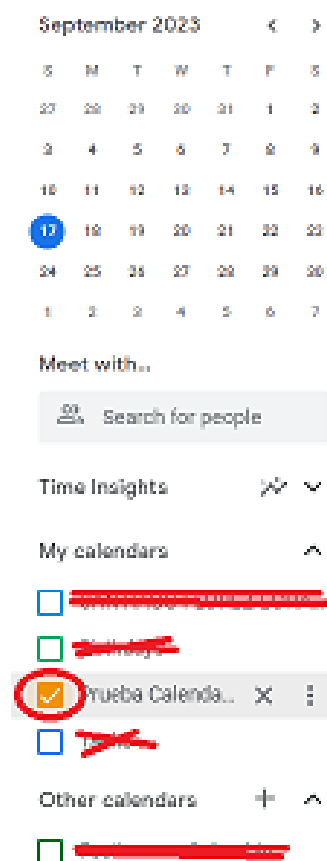
Paso 35: Ahora vamos a rellenar la información solicitada y vamos a escoger "Create calendar" donde nos saldrá que fue creado correctamente.

A screenshot of the 'Create new calendar' form in Google Calendar. The form has several fields: 'Name' with the value 'Prueba Calendario', 'Description' with the value 'Este es un calendario de pruebas', 'Time zone' with a dropdown menu showing '(GMT-05:00) Colombia Standard Time', 'Owner' with the name 'SANTIAGO FLOREZ ECHAVARRIA', and 'Organization' with the email 'misena.edu.co'. At the bottom, there is a blue button labeled 'Create calendar' which is highlighted with a red rectangle. Below the form, there is a success message in a dark box: '"Prueba Calendario" successfully created' followed by 'Configure' and a close button 'X'. The entire success message box is also highlighted with a red rectangle.

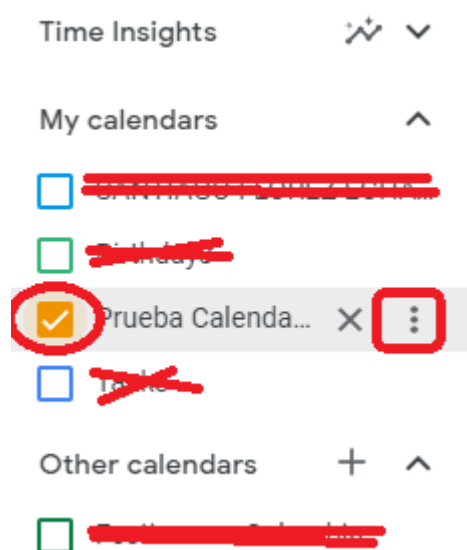
Paso 36: Ahora iremos a escoger la flecha para ir atrás



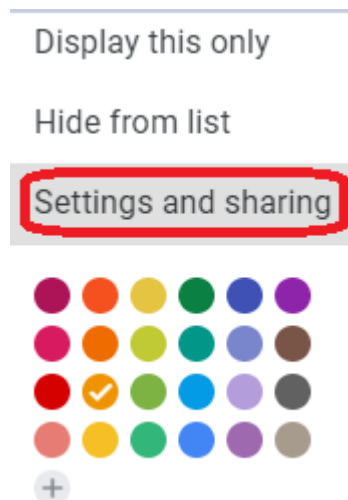
Paso 37: Donde se ven todos los calendarios que tenemos creados vamos a quitar todos los que tenemos excepto el que acabamos de crear que lo vamos a chulear



Paso 38: Al calendario que acabamos de crear vamos a seleccionar los tres puntos de la derecha



Paso 39: Escogemos la opción “Setting and sharing”



Paso 40: Nos aparece esta página y bajamos

The screenshot shows the 'Calendar settings' page for a calendar named 'Prueba Calendario'. The page is divided into several sections: 'Calendar settings', 'Auto-accept invitations', and 'Access permissions for events'. Two red arrows point downwards, indicating the scroll direction. The 'Calendar settings' section includes fields for 'Name' (Prueba Calendario), 'Description' (Este es un calendario de pruebas), 'Time zone' (GMT-05:00 Colombia Standard Time), and 'Organization' (misena.edu.co). There is an 'Export calendar' button and a link to 'Learn more about exporting your calendar'. The 'Auto-accept invitations' section has a dropdown menu set to 'Automatically add all invitations to this calendar'. The 'Access permissions for events' section shows two permissions: 'Make available to public' (unchecked) and 'Make available for Servicio Nacional de Aprendizaje SENA' (checked). Each permission has a 'See all event details' link.

Calendar settings

Name
Prueba Calendario

Description
Este es un calendario de pruebas

Time zone
(GMT-05:00) Colombia Standard Time

Organization
misena.edu.co

Export calendar

Learn more about [exporting your calendar](#)

Auto-accept invitations

Automatically add all invitations to this calendar

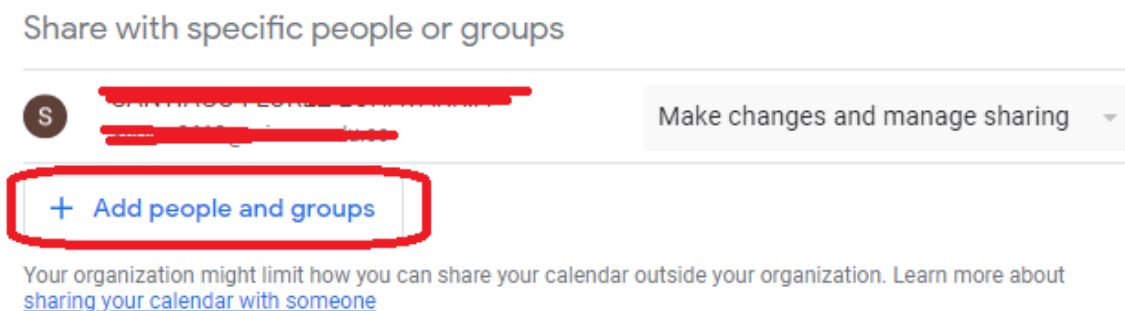
Calendars for resources can auto-accept invitations. [Learn more about auto-accept invitations](#)

Access permissions for events

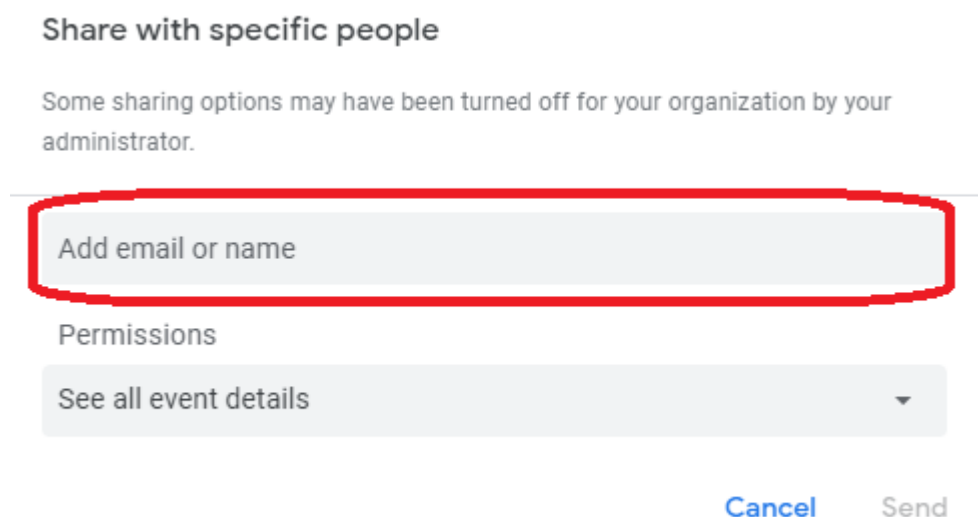
☐ Make available to public [See all event details](#)

☒ Make available for Servicio Nacional de Aprendizaje SENA [See all event details](#)

Paso 41: Donde dice esta opción llamada “Share with specific people or groups” aparecemos nosotros y le damos a “Add people and groups”



Paso 42: Ahora vemos que nos pide añadirlo, no añadimos nada de momento.



Paso 43: Volvemos al archivo del código y buscamos en (línea 6) la donde dice “client_email” y copiamos lo que dice entre las “” quitando las comillas

```
1 {
2   "type": "service_account",
3   "project_id": "exalted-crane-399310",
4   "private_key_id": "cbf93c85e779fbef6364e0dc55a9f854e2bb848",
5   "private_key": "-----BEGIN PRIVATE KEY-----\nMIIEEgIBADANBgkqhkiG9w0BAQEFAAQgSkAgEAAoIBAQD3tvPugIwsKlay\n6   "client_email": "api-calendario@exalted-crane-399310.iam.gserviceaccount.com",
7   "client_id": "115228101410851304469",
8   "auth_uri": "https://accounts.google.com/o/oauth2/auth",
9   "token_uri": "https://oauth2.googleapis.com/token",
10  "auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1/certs",
11  "client_x509_cert_url": "https://www.googleapis.com/robot/v1/metadata/x509/api-calendario%40exalted-crane-399310.iam.gserviceaccount.com",
12  "universe_domain": "googleapis.com"
13 }
```

Paso 44: Lo colocamos en la página y en permisos le damos click

Share with specific people

Some sharing options may have been turned off for your organization by your administrator.

api-calendario@exalted-crane-399310.iam.gserviceaccount.com

Permissions

See all event details ▼

[Cancel](#) [Send](#)

Paso 45: Vamos a seleccionar “Make changes to events”

Make changes to events ▲

See only free/busy (hide details)

See all event details

Make changes to events

Make changes and manage sharing

Paso 46: Ahora que lo tenemos preparado le damos click en “Send”

Share with specific people

Some sharing options may have been turned off for your organization by your administrator.

api-calendario@exalted-crane-399310.iam.gserviceaccount.com

Permissions

Make changes to events

Cancel

Send

Paso 47: Ahora vemos como se agregó.

Share with specific people or groups



[Redacted email address]

Make changes and manage sharing



api-calendario@exalted-crane-399310.iam.gserviceacco...

Make changes to events



+ Add people and groups

Paso 48: Ahora vamos a abrir de nuevo la consola en nuestro proyecto y veremos si tenemos node instalado colocando “node -v”

C:\Windows\System32\cmd.exe

```
Microsoft Windows [Versión 10.0.19045.3448]
(c) Microsoft Corporation. Todos los derechos reservados.

S:\calendariogoogle>node -v
```

Paso 49: En caso de que este instalado nos saldrá la versión de node que tenemos, también vamos a mirar si tenemos “NPM” instalado colocando “npm -v”

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Versión 10.0.19045.3448]
(c) Microsoft Corporation. Todos los derechos reservados.

S:\calendariogoogle>node -v
v18.17.1

S:\calendariogoogle>npm -v
```

Paso 50: Y si esta instalado nos sale la versión y vamos a abrir el código una vez más

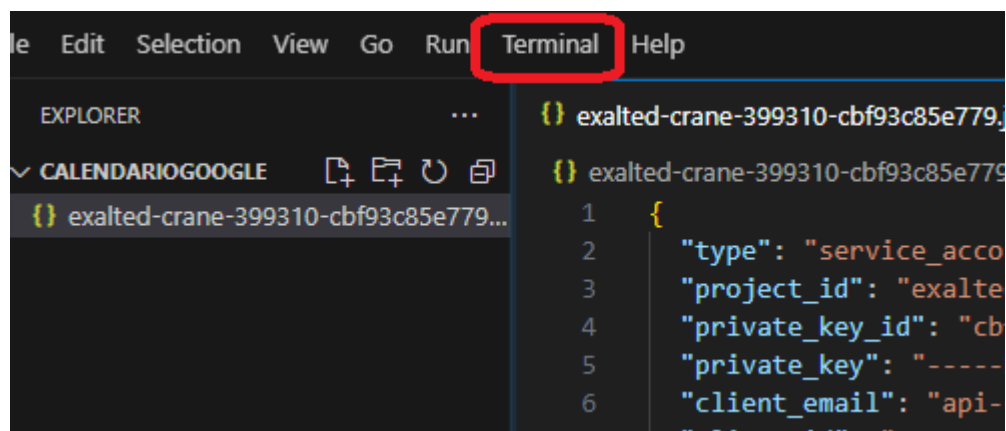
```
C:\Windows\System32\cmd.exe
Microsoft Windows [Versión 10.0.19045.3448]
(c) Microsoft Corporation. Todos los derechos reservados.

S:\calendariogoogle>node -v
v18.17.1

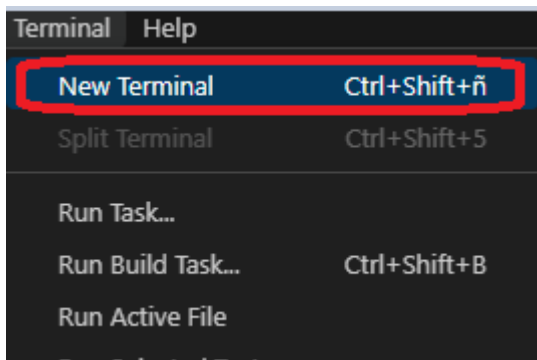
S:\calendariogoogle>npm -v
9.6.7

S:\calendariogoogle>code .
```

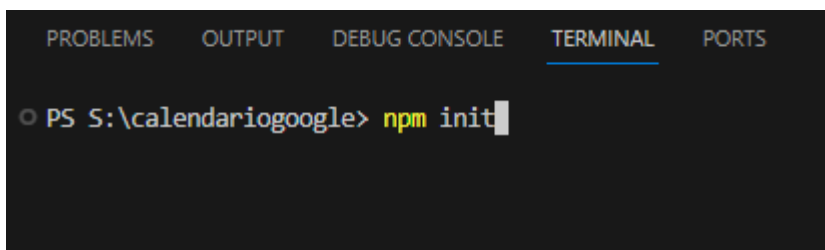
Paso 51: Ahora en nuestro proyecto arriba vamos a seleccionar “Terminal”



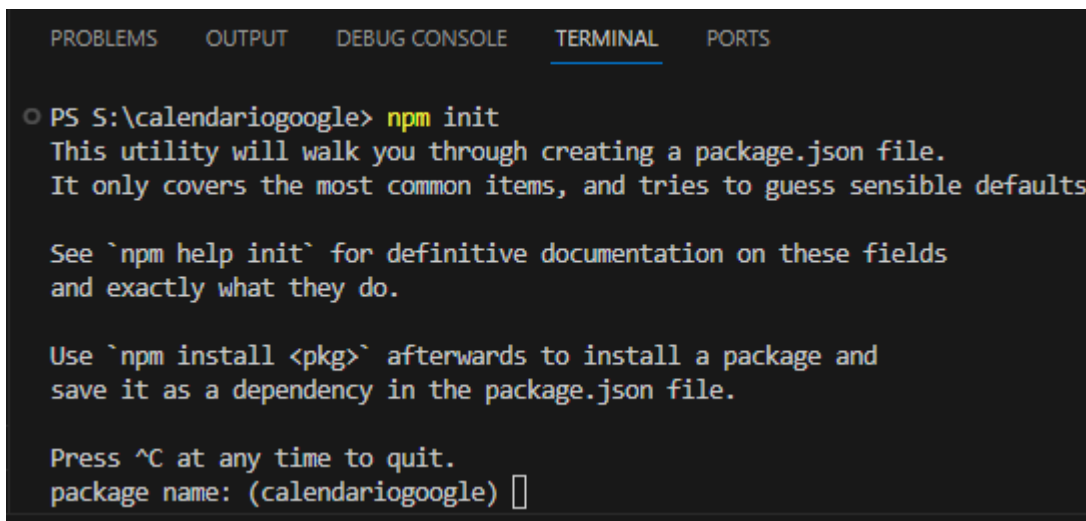
Paso 52: Ahora seleccionamos “New Terminal” para abrir la terminal



Paso 53: Gracias a que tenemos node js y npm instalados vamos a colocar en la terminal “npm init”



Paso 54: Comenzará a crear los archivos json y nos pedirá información vamos a dar enter a todo excepto a el autor



Paso 55: En Author vamos a colocar nuestro nombre y daremos enter de nuevo hasta que se termine de instalar.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (calendariogoogle)
version: (1.0.0)
description:
entry point: (index.js)
test command:
git repository:
keywords:
author: Santiago Florez Echavarria
```

Paso 56: Nos mostrará los datos sobre el archivo y luego nos dirá si está bien, escribiremos “yes”

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

{
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Santiago Florez Echavarria",
  "license": "ISC"
}

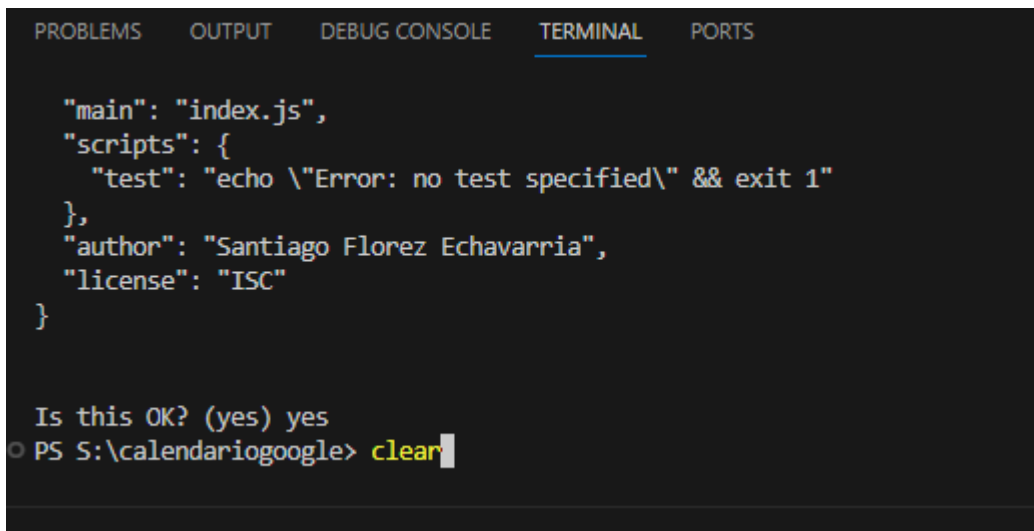
Is this OK? (yes) yes
```

Paso 57: Ahora si se instalo correctamente vamos a ver que se creo un archivo llamado “package.json”

```
EXPLORER  ...

▼ CALENDARIOGOOGLE
  {} exalted-crane-399310-cbf93c85e779...
  {} package.json
```

Paso 58: Limpiaremos la terminal con “clear”

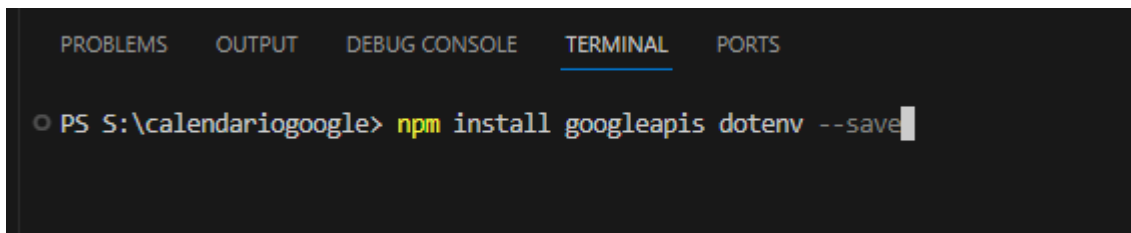


```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Santiago Florez Echavarria",
  "license": "ISC"
}

Is this OK? (yes) yes
PS S:\calendariogoogle> clear
```

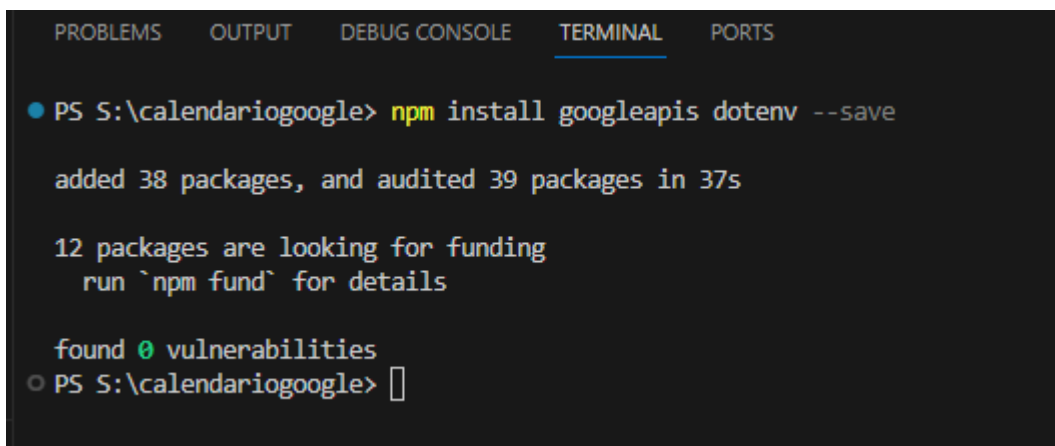
Paso 59: Ahora vamos a instalar la biblioteca de Google apis y también vamos a instalar “dotenv” el cual es una biblioteca de node js que sirve para cargar variables de entorno desde un archivo llamado “.env” que crearemos más adelante. Y si tienes una versión de npm superior a 5.0.0 no es necesario colocar --save el cual sirve para que personas que clonen el proyecto o trabajen en el puedan instalar las dependencias fácilmente sin embargo luego de la versión 5.0.0 no es necesario ya que npm guarda automáticamente las dependencias en “package.json”



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS S:\calendariogoogle> npm install googleapis dotenv --save
```

Paso 60: Veremos si se instaló correctamente.



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

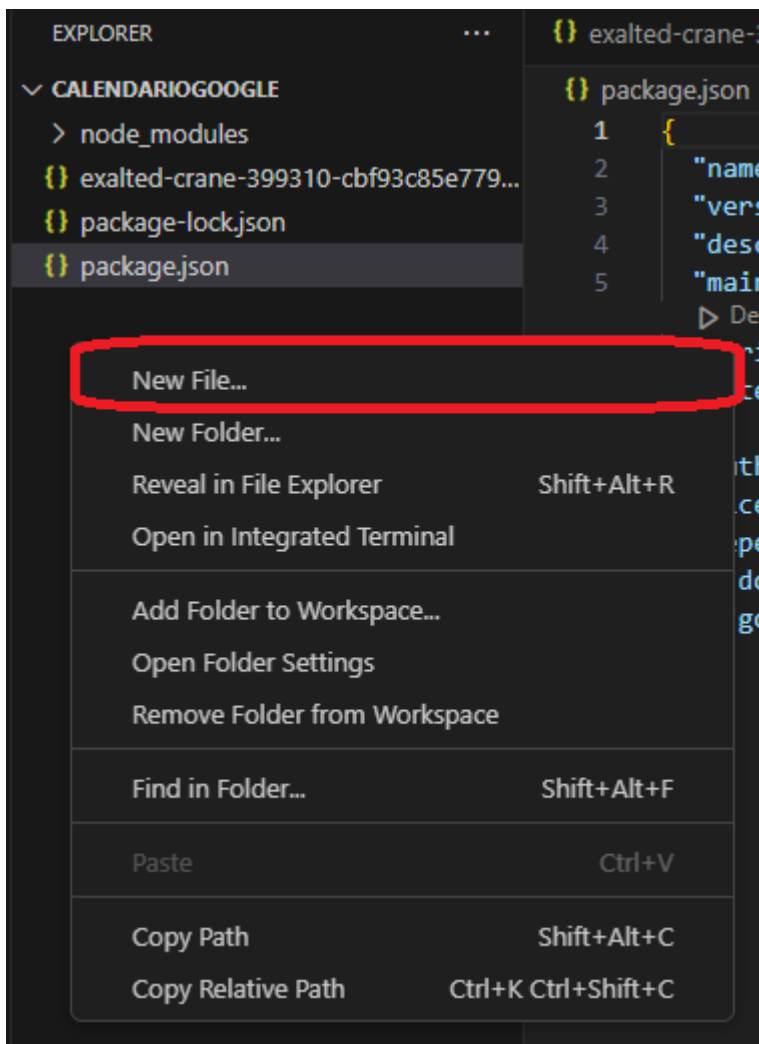
● PS S:\calendariogoogle> npm install googleapis dotenv --save

added 38 packages, and audited 39 packages in 37s

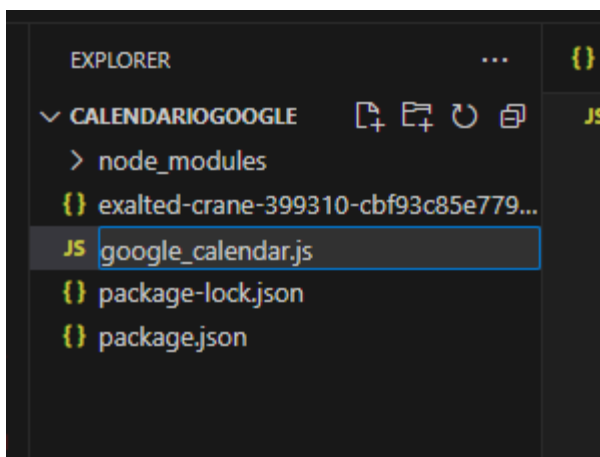
12 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS S:\calendariogoogle> 
```

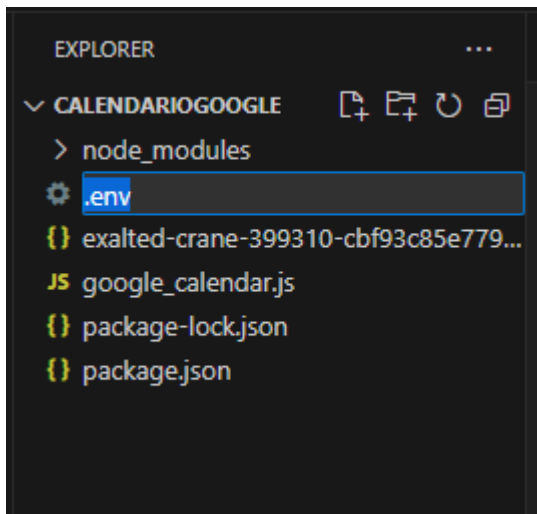
Paso 61: Ahora vamos a crear un nuevo archivo en nuestra carpeta raíz y lo vamos a llamar "google_calendar.js"



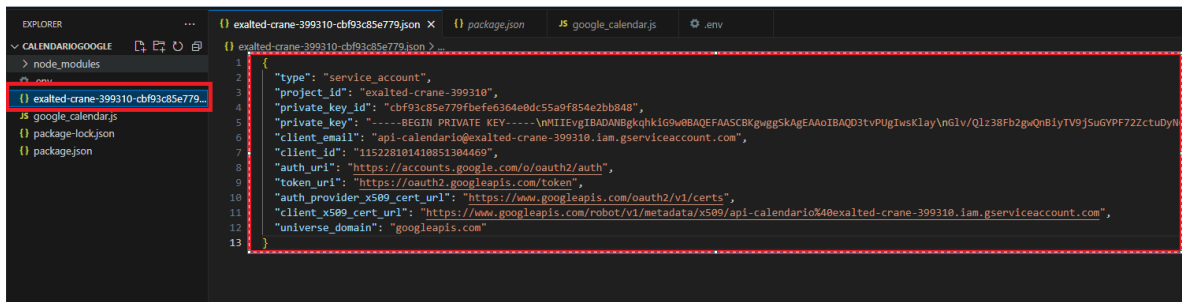
Paso 62: Así debe quedar nuestro archivo.



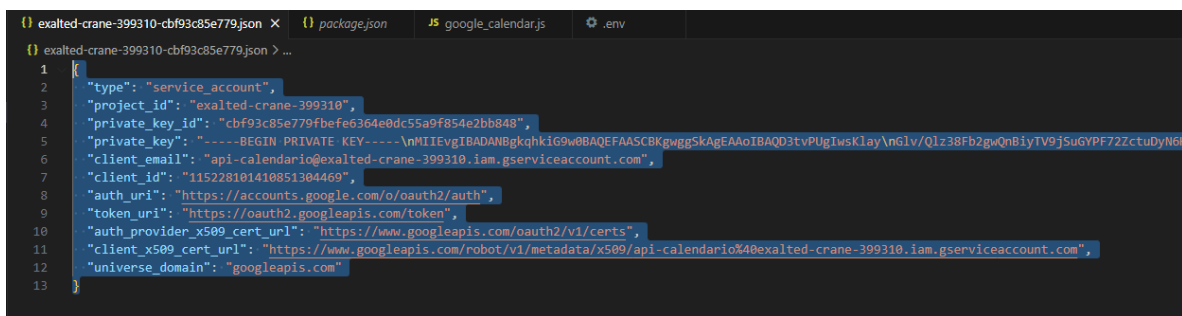
Paso 63: Ahora vamos a crear las variables de entorno. Creando un nuevo archivo llamado “.env”



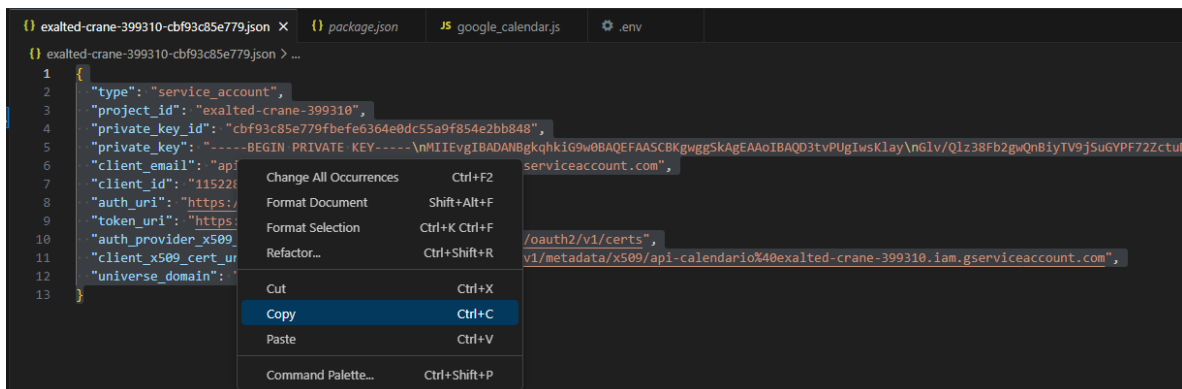
Paso 64: Iremos a primer archivo el cual descargamos de la Google. Y copiamos todo lo que hay dentro.



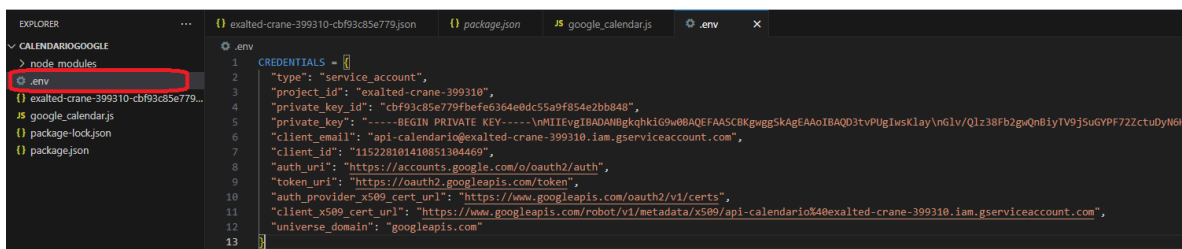
Paso 65: Lo seleccionamos todo y damos click derecho.



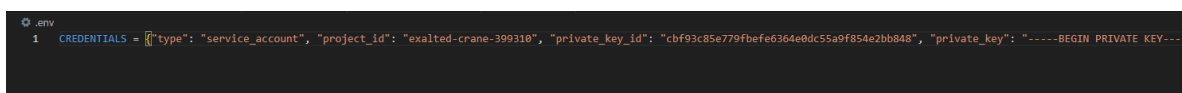
Paso 66: Le damos “copy” o “copiar”



Paso 67: Ahora iremos a nuestro archivo “.env” y vamos a crear una variable que contendrá todos los credenciales llamada “CREDENTIALS” y la haremos igual a lo que copiamos del archivo anterior. En este archivo de .env todas las variables deben ir todo en mayúscula



Paso 68: Ahora para que esté un poco más organizado vamos a dejarlo todo en una sola línea de texto, ten mucho cuidado de colocar todo bien.



The screenshot shows the Google Calendar settings interface. On the left, the 'Settings' menu is visible with categories like 'General', 'Settings for my calendars', and 'Settings for other calendars'. Under 'Settings for my calendars', 'Prueba Calendario' is selected. The main content area shows settings for this calendar, including Name, Description, Time zone, Organization, and an 'Export calendar' button. Below this, there are sections for 'Auto-accept invitations' and 'Access permissions for events'. Two red arrows are overlaid on the image: one pointing to the 'Export calendar' button and another pointing to the 'Learn more about auto-accept invitations' link.

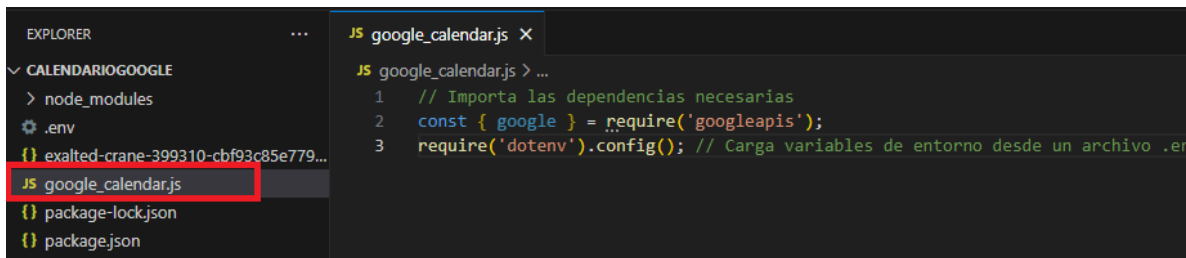
Integrate calendar

Use this URL to access this calendar from a web browser.

Use this code to embed this calendar in a web page

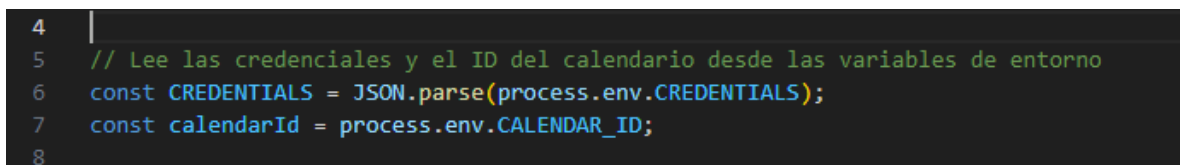
```
1 CREDENTIALS = {"type": "service_account", "project_id": "exalted-crane-399310", "private_key_id": "cbf93c85e779fbefe6364e0dc5"}
2 CALENDAR_ID=c_660fa07c93205be80cf31aac5a1789b75cb6f9f548d2737148118cd3cdd42e7@group.calendar.google.com
```

Paso 72: Ahora vamos a ir a nuestro archivo js “google_calendar.js” y vamos a importar las apis de google (línea 2) y también vamos a cargar las variables de entorno desde el archivo “.env” (línea 3)



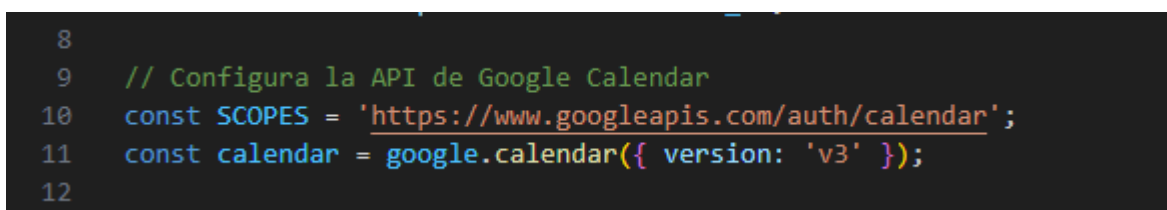
```
JS google_calendar.js X
JS google_calendar.js > ...
1 // Importa las dependencias necesarias
2 const { google } = require('googleapis');
3 require('dotenv').config(); // Carga variables de entorno desde un archivo .env
```

Paso 73: Ahora hacemos que lea las credenciales y el id del calendario almacenándolo (línea 6 y 7)



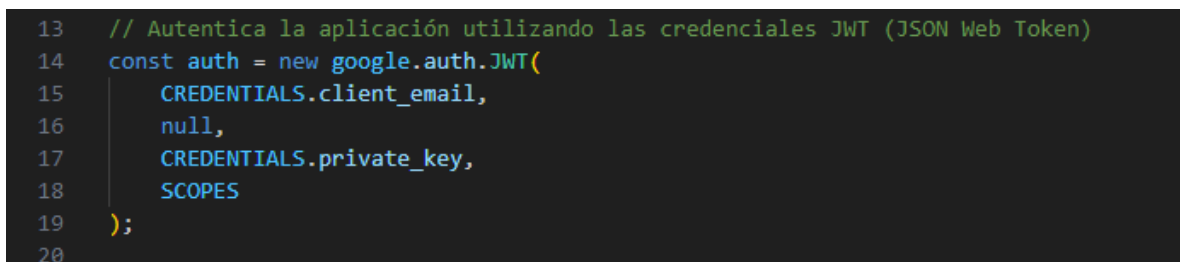
```
4
5 // Lee las credenciales y el ID del calendario desde las variables de entorno
6 const CREDENTIALS = JSON.parse(process.env.CREDENTIALS);
7 const calendarId = process.env.CALENDAR_ID;
8
```

Paso 74: Ahora vamos a configurar la API de Google Calendar con la (línea 10) vamos a solicitar acceso para la administración de calendarios del usuario mientras que en la (línea 11) vamos a crear un objeto “calendar” el cual va a interactuar con la API de Google Calendar usando la función de la biblioteca de “googleapis” llamada “calendar” y vamos a usar la versión 3 del api.



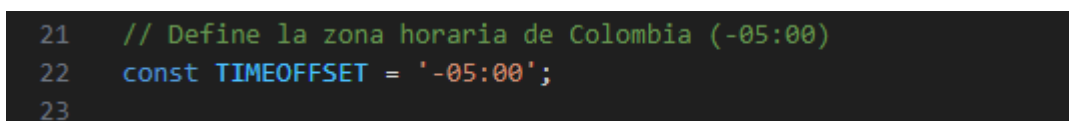
```
8
9 // Configura la API de Google Calendar
10 const SCOPES = 'https://www.googleapis.com/auth/calendar';
11 const calendar = google.calendar({ version: 'v3' });
12
```

Paso 75: Y vamos a autenticar la aplicación usando las credenciales a ver si están correctas. (línea desde la 14 hasta la 19)



```
13 // Autentica la aplicación utilizando las credenciales JWT (JSON Web Token)
14 const auth = new google.auth.JWT(
15   CREDENTIALS.client_email,
16   null,
17   CREDENTIALS.private_key,
18   SCOPES
19 );
20
```

Paso 76: Vamos a definir nuestra zona horaria que en caso de Colombia es “-05:00”



```
21 // Define la zona horaria de Colombia (-05:00)
22 const TIMEOFFSET = '-05:00';
23
```

Paso 77: Ahora vamos a crear una función llamada “dateTimeForCalendar” y le haremos una función flecha (línea 25 a 27)

```
23
24 // Función para obtener una cadena de fecha y hora para el calendario
25 const dateTimeForCalendar = () => {} // Inicio función
26
27 }; //Final función
```

Paso 78: Ahora vamos a crear una variable que obtiene la fecha y hora actual (línea 28) además de eso vamos a extraer los componentes de la fecha como el año (línea 31) el mes (línea 32) como dato curioso los meses comienzan desde 0 en javascript así que hacemos que se agregue un +1 para que comience en 1 luego haremos una condicional con los meses de que en caso de que el número del mes sea menor a 0 agregue un 0 al comienzo (línea 33 a 35) también haremos los días (línea 36) y haremos la misma condicional con los días (línea 37 a 39) luego vamos a capturar las horas (línea 40) y la misma condicional (línea 41 a 43) y por último los minutos (línea 44) y la misma condicional (línea 45 a 47)

```
25 const dateTimeForCalendar = () => {} // Inicio función
26
27 // Obtiene la fecha y hora actual
28 let date = new Date();
29
30 // Extrae los componentes de fecha y hora
31 let year = date.getFullYear(); // Obtiene el año actual
32 let month = date.getMonth() + 1; // Obtiene el mes actual (¡Nota! Los meses comienzan desde 0)
33 if (month < 10) {
34     month = `0${month}`; // Agrega un cero inicial si el mes es menor que 10
35 }
36 let day = date.getDate(); // Obtiene el día del mes actual
37 if (day < 10) {
38     day = `0${day}`; // Agrega un cero inicial si el día es menor que 10
39 }
40 let hour = date.getHours(); // Obtiene la hora actual
41 if (hour < 10) {
42     hour = `0${hour}`; // Agrega un cero inicial si la hora es menor que 10
43 }
44 let minute = date.getMinutes(); // Obtiene los minutos actuales
45 if (minute < 10) {
46     minute = `0${minute}`; // Agrega un cero inicial si los minutos son menores que 10
47 }
48
49 }; //Final función
```


Paso 79: Ahora dentro de la misma función vamos a construir una cadena de la fecha y la hora en un formato (ISO 8601) (Línea 50) también vamos a convertir esa cadena en un objeto date (línea 53) luego vamos a calcular la hora de inicio para agregar el evento al calendario (línea 56) y la hora de su finalización que será una hora después (línea 57) luego vamos a hacer que retorne las horas de cuando comenzó y cuando termina (líneas desde la 59 hasta la 62) , y dejamos ahí la función (línea 63)

```
46     minute = `0${minute}`; // Agrega un cero inicial si los minutos son menores que 10
47 }
48
49 // Construye la cadena de fecha y hora en formato ISO 8601
50 let newDateTime = `${year}-${month}-${day}T${hour}:${minute}:00.000${TIMEOFFSET}`;
51
52 // Convierte la cadena en un objeto Date
53 let event = new Date(Date.parse(newDateTime));
54
55 // Calcula la hora de inicio y finalización (1 hora después)
56 let startDate = event;
57 let endDate = new Date(new Date(startDate).setHours(startDate.getHours() + 1));
58
59 return {
60     'start': startDate, // Devuelve la hora de inicio calculada
61     'end': endDate // Devuelve la hora de finalización calculada (1 hora después)
62 }
63 }; //Final función
```

Paso 80: Ahora vamos a hacer que se imprima en la terminal para ver la hora para asegurarnos de que la función esta correcta (línea 66)

```
63 }; //Final función
64
65 // Imprime la fecha y hora para el calendario
66 console.log(dateTimeForCalendar());
67
```

Paso 81: Ahora vamos a la terminal y vamos a colocar (node google_calendar.js)

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS S:\calendariogoogle> node google_calendar.js
```

Paso 82: Y vemos como se agregó la hora correctamente, aunque no la comprendas si te sale algo a fechas vamos bien.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

● PS S:\calendariogoogle> node google_calendar.js
  { start: 2023-09-18T07:49:00.000Z, end: 2023-09-18T08:49:00.000Z }
○ PS S:\calendariogoogle> a
```

Paso 83: Ahora borramos donde imprimimos la hora (línea 66) y su comentario en caso de que lo hayas hecho (línea 65) y en su lugar vamos a crear una nueva función flecha llamada (insertEvent) y contendrá el argumento (event) que posteriormente vamos a crear.

```
63   }; //Final función
64
65   // Función para insertar un nuevo evento en Google Calendar
66   const insertEvent = async (event) => { //Inicio evento
67
68   }; //Final evento
```

Paso 84: Realizaremos un try catch (líneas desde las 68 a la 72)

```
66   const insertEvent = async (event) => { //Inicio evento
67
68       try {
69
70       } catch (error) {}
71
72   }; //Final evento
```

Paso 85: Dentro del try vamos a intentar insertar un evento en el calendario usando la api de Google Calendar (línea desde la 70 hasta la 74)

```
68       try {
69           // Intenta insertar un evento en el calendario usando la API de Google Calendar
70           let response = await calendar.events.insert({
71               auth: auth, // Autenticación previamente configurada
72               calendarId: calendarId, // ID del calendario previamente configurado
73               resource: event // Datos del evento a insertar
74           });
75
76       } catch (error) {
77
78       }
```

Paso 86: Dentro del try vamos a verificar que se haya insertado correctamente con una condicional (línea 77) en caso de que se haya insertado correctamente nos retorna 1 (línea 78) de lo contrario (línea 79) nos va a retornar 0 (línea 80)

```
69 // Intenta insertar un evento en el calendario usando la API de Google Calendar
70 let response = await calendar.events.insert({
71   auth: auth, // Autenticación previamente configurada
72   calendarId: calendarId, // ID del calendario previamente configurado
73   resource: event // Datos del evento a insertar
74 });
75
76 // Verifica si la inserción fue exitosa
77 if (response['status'] == 200 && response['statusText'] === 'OK') {
78   return 1; // Éxito: Se insertó el evento correctamente
79 } else {
80   return 0; // Falla: La inserción del evento no tuvo éxito
81 }
```

Paso 87: Ahora en el catch (línea 82) vamos a capturar un error en caso de que lo haya no envíe el error a la consola (línea 83) y además retorne 0 (línea 84) y vamos a terminar la función.

```
79   } else {
80     return 0; // Falla: La inserción del evento no tuvo éxito
81   }
82 } catch (error) {
83   console.log(`Error en insertEvent --> ${error}`); // Registra cualquier error en la consola
84   return 0; // Falla: Hubo un error durante la inserción del evento
85 }
86 }; //Final Evento
```

Paso 88: Ahora obtenemos la fecha y la hora para el evento (línea 89)

```
87
88 // Obtiene la fecha y hora para el evento
89 let dateTime = dateTimeForCalendar();
90
```

Paso 89: Ahora vamos a crear un evento para Google Calendar (línea 92) y lo llamaremos “event”

```
91 // Define un nuevo evento para Google Calendar
92 let event = {} //Inicio Evento
93
94 }; //Final Evento
95
```

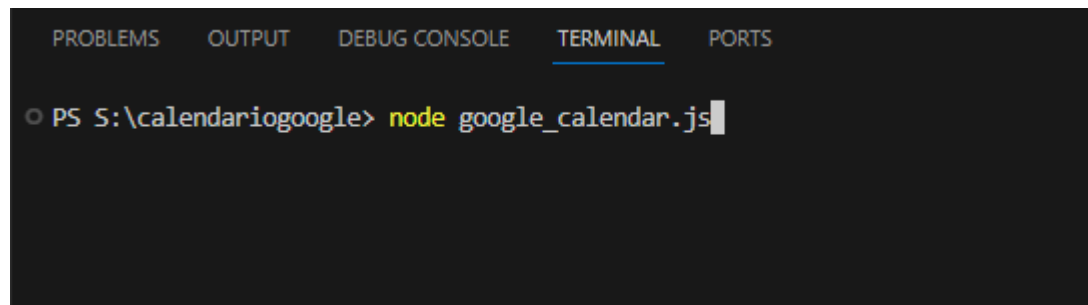
Paso 90: Dentro de este evento vamos a agregar las descripciones que nos pide crear un evento en google calendar, comenzaremos con por resumen (línea 93) es importante que mires las comillas ya que para seleccionar que agregaremos vamos a colocarla con comillas simple (' ') y lo que se agregará con otro tipo de comillas conocidas como “plantillas de cadenas” o “template literals” las cuales son (` `), también agregaremos la descripción (líneas 94) y vamos a agregar la hora del comienzo (líneas desde la 95 hasta la 98) y cuando terminará el evento (líneas desde la 99 hasta la 102) y por último cerramos el evento (línea 103)

```
92 let event = { //Inicio Evento
93   'summary': `Este es el resumen.`,
94   'description': `Esta es la descripción.`,
95   'start': {
96     'dateTime': dateTime['start'],
97     'timeZone': 'America/Bogota' // Configura la zona horaria de Bogotá
98   },
99   'end': {
100     'dateTime': dateTime['end'],
101     'timeZone': 'America/Bogota'
102   }
103 }; //Final Evento
104
```

Paso 91: Ahora vamos a llamar la función “insertEvent” y le vamos a insertar el evento (línea 106) y luego en caso de que haya funcionado vamos a imprimir la respuesta (línea 107 a 109) y vamos a capturar el error en caso de que no haya funcionado (líneas desde la 110 a la 112)

```
105 // Llama a la función para insertar el evento
106 insertEvent(event)
107   .then((res) => {
108     console.log(res); // Imprime la respuesta de la inserción del evento
109   })
110   .catch((err) => {
111     console.log(err); // Imprime cualquier error que ocurra durante la inserción
112   });
```

Paso 92: Ahora vamos a la terminal de nuevo y vamos a colocar “node google_calendar.js”



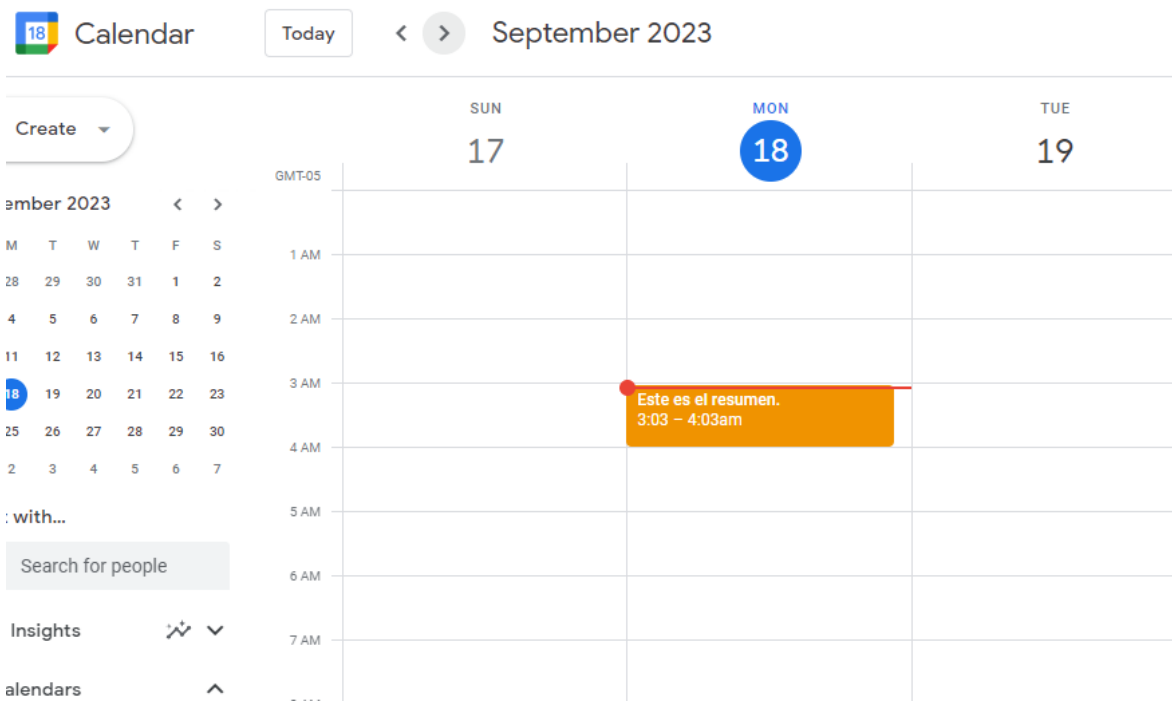
The screenshot shows a terminal window with a dark background. At the top, there are tabs labeled 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL', and 'PORTS'. The 'TERMINAL' tab is selected. Below the tabs, the command prompt shows 'PS S:\calendariogoogle>' followed by the command 'node google_calendar.js' which has been entered and is followed by a cursor.

Paso 93: Si nos retorna “1” significa que funciona.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

● PS S:\calendariogoogle> node google_calendar.js
1
○ PS S:\calendariogoogle> |
```

Paso 94: Vemos como efectivamente se agrego al calendario de google calendar.



Paso 95: Ahora que hemos agregado un evento al calendario vamos a comentar todo desde la (línea 88 a la 112), ya que no queremos que se sigan agregando cosas.

PD: Por si no sabías para hacerlo más sencillo puedes seleccionar todo el código y hacer “Ctrl + k” y luego “Ctrl + c” y si quieres descomentar mucho texto puedes hacer “Ctrl + k” y luego “Ctrl + u”

```
88 // // Obtiene la fecha y hora para el evento
89 // let dateTime = dateTimeForCalendar();
90
91 // // Define un nuevo evento para Google Calendar
92 // let event = { //Inicio Evento
93 //   'summary': `Este es el resumen.`,
94 //   'description': `Esta es la descripción.`,
95 //   'start': {
96 //     'dateTime': dateTime['start'],
97 //     'timeZone': 'America/Bogota' // Configura la zona horaria de Bogotá
98 //   },
99 //   'end': {
100 //     'dateTime': dateTime['end'],
101 //     'timeZone': 'America/Bogota'
102 //   }
103 // }; //Final Evento
104
105 // // Llama a la función para insertar el evento
106 // insertEvent(event)
107 //   .then((res) => {
108 //     console.log(res); // Imprime la respuesta de la inserción del evento
109 //   })
110 //   .catch((err) => {
111 //     console.log(err); // Imprime cualquier error que ocurra durante la inserción
112 //   });
```

Paso 96: Ahora vamos a crear una función para obtener los eventos entre 2 fechas. Y la llamaremos “getEvent” (línea 115)

```
113
114 // Función para obtener todos los eventos entre dos fechas
115 const getEvents = async (dateTimeStart, dateTimeEnd) => { //INICIO EVENTO
116
117 }; //FINAL EVENTO
118
```

Paso 97: Realizaremos un “try” y “catch” (líneas desde la 117 a la 121)

```
115 const getEvents = async (dateTimeStart, dateTimeEnd) => { //INICIO EVENTO
116
117   try {
118
119   } catch (error) {
120
121   }
122 }; //FINAL EVENTO
```

Paso 98: Dentro del try vamos a obtener una lista de eventos desde google calendar (línea 119) y vamos a verificar la autenticación (línea 120) y el Id del calendario (línea 121) también la fecha y hora del inicio (línea 122) y la fecha y hora del final (línea 123) y por último vamos a configurar la zona horaria de Colombia (línea 124)

```
117     try {  
118         // Intenta obtener una lista de eventos desde Google Calendar dentro del rango de fechas especificado  
119         let response = await calendar.events.list({  
120             auth: auth, // Autenticación previamente configurada  
121             calendarId: calendarId, // ID del calendario previamente configurado  
122             timeMin: dateTimeStart, // Fecha y hora de inicio del rango  
123             timeMax: dateTimeEnd, // Fecha y hora de fin del rango  
124             timeZone: 'America/Bogota' // Configura la zona horaria de Bogotá  
125         });  
126     }  
127     catch (error) {
```

Paso 99: Ahora dentro del try vamos a obtener los eventos de la respuesta y lo vamos a almacenar (línea 128) y por último lo retornamos (línea 129)

```
125         });  
126     }  
127     // Obtiene los eventos de la respuesta  
128     let items = response['data']['items'];  
129     return items; // Devuelve la lista de eventos encontrados  
130 } catch (error) {  
131     //  
132 }  
133 }; //FINAL EVENTO
```

Paso 100: Ahora por si sale mal el encontrar los eventos vamos a crear el catch donde se enviará un mensaje que contiene el error (línea 131) y también nos retornará 0 (línea 132)

```
130     } catch (error) {  
131         console.log(`Error en getEvents --> ${error}`); // Registra cualquier error en la consola  
132         return 0; // Falla: Hubo un error al intentar obtener los eventos  
133     }  
134 }; //FINAL EVENTO
```

Paso 101: Ahora vamos a definir la fecha de inicio (línea 137) y la fecha final (línea 138)

```
135  
136     // Define las fechas de inicio y fin para obtener eventos  
137     let start = '2023-08-10T00:00:00.000Z';  
138     let end = '2023-10-11T00:00:00.000Z';  
139
```

Paso 102: Ahora vamos a llamar la función “getEvent” para obtener el rango de las fechas especificado (línea 141) y va a imprimir en caso de que salga bien (líneas desde la 142 hasta la 144) y en caso de que envíe error lo imprimirá (líneas desde la 145 hasta la 147)

```
140 // Llama a la función para obtener eventos en el rango de fechas especificado
141 getEvents([start, end])
142   .then((res) => {
143     console.log(res); // Imprime los eventos obtenidos en la consola
144   })
145   .catch((err) => {
146     console.log(err); // Imprime cualquier error que ocurra durante la obtención de eventos en la consola
147   });
```

Paso 103: Sin embargo, de momento vamos a comentar todo desde la (línea 136 hasta la 147)

```
135
136 // // Define las fechas de inicio y fin para obtener eventos
137 // let start = '2023-08-10T00:00:00.000Z';
138 // let end = '2023-10-11T00:00:00.000Z';
139
140 // // Llama a la función para obtener eventos en el rango de fechas especificado
141 // getEvents(start, end)
142 //   .then((res) => {
143 //     console.log(res); // Imprime los eventos obtenidos en la consola
144 //   })
145 //   .catch((err) => {
146 //     console.log(err); // Imprime cualquier error que ocurra durante la obtención de eventos en la consola
147 //   });
```

Paso 104: Ahora vamos a crear una función para eliminar un evento por su id y la llamaremos (deleteEvent) (línea 150)

```
149 // Función para eliminar un evento por su ID
150 const deleteEvent = async (eventId) => { //INICIO FUNCIÓN
151
152   }; //FINAL FUNCIÓN
```

Paso 105: Dentro de la función vamos a crear un “try” “catch”

```
149 // Función para eliminar un evento por su ID
150 const deleteEvent = async (eventId) => { //INICIO FUNCION
151   try {
152
153   } catch (error) {
154
155   }
156 }; //FINAL FUNCION
```


Paso 106: Dentro del try vamos a enviar la solicitud para eliminar un evento según su id (líneas desde la 153 hasta la 157) en la (línea 156) se recibe el id del evento el cual por el momento no conocemos, sin embargo, colócalo tal cual está que luego lo vamos a conseguir.

```
150 const deleteEvent = async (eventId) => { //INICIO FUNCION
151   try {
152     // Envía una solicitud para eliminar un evento específico utilizando su ID
153     let response = await calendar.events.delete({
154       auth: auth,
155       calendarId: calendarId,
156       eventId: eventId
157     });
158   }
159 } catch (error) {}
```

Paso 107: Ahora dentro del mismo try vamos a verificar si la eliminación fue exitosa (línea 160) y en caso de que si imprima 1 (línea 161) de lo contrario (línea 162) vamos a retornar un 0 (línea 163)

```
157   });
158
159   // Verifica si la eliminación fue exitosa
160   if (response.data === '') {
161     return 1; // Éxito: el evento se eliminó correctamente
162   } else {
163     return 0; // Falla: no se pudo eliminar el evento
164   }
165 } catch (error) {}
```

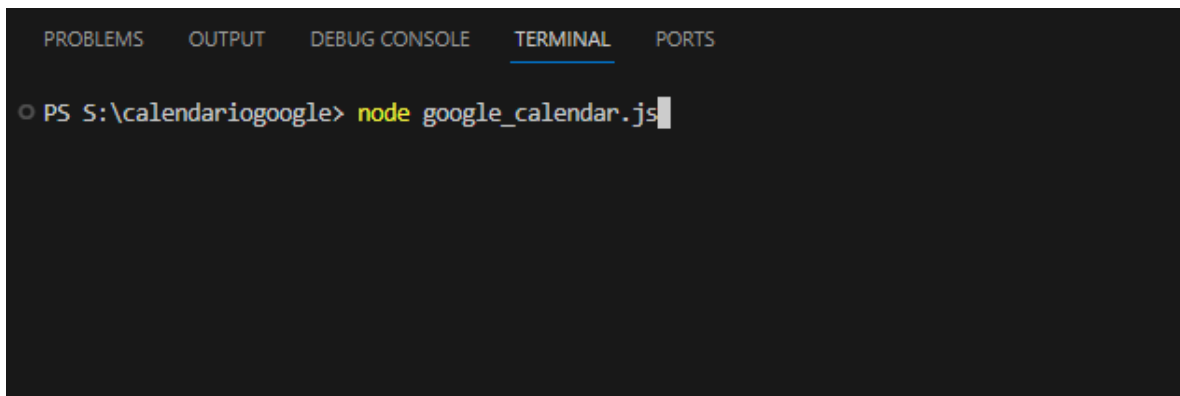
Paso 108: Ahora en el catch vamos (línea 165) vamos a capturar que hubo un error y lo va a imprimir (línea 166) y también va a retornar un 0 (línea 167)

```
165 } catch (error) {}
166   console.log(`Error en deleteEvent --> ${error}`);
167   return 0; // Falla: se produjo un error durante la eliminación del evento
168 }
169 }; //FINAL FUNCION
```

Paso 109: Ahora vamos a quitar todo el comentario desde la (línea 136 hasta la 147) es decir deshacer el paso 103

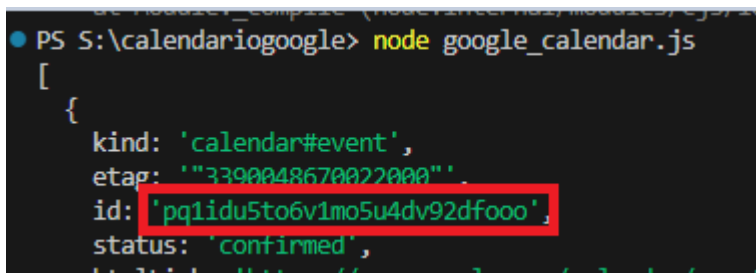
```
135
136 // Define las fechas de inicio y fin para obtener eventos
137 let start = '2023-08-10T00:00:00.000Z';
138 let end = '2023-10-11T00:00:00.000Z';
139
140 // Llama a la función para obtener eventos en el rango de fechas especificado
141 getEvents(start, end)
142   .then((res) => {
143     console.log(res); // Imprime los eventos obtenidos en la consola
144   })
145   .catch((err) => {
146     console.log(err); // Imprime cualquier error que ocurra durante la obtención de eventos en la consola
147   });
```

Paso 110: Ahora en la terminación vamos a depurar el node.



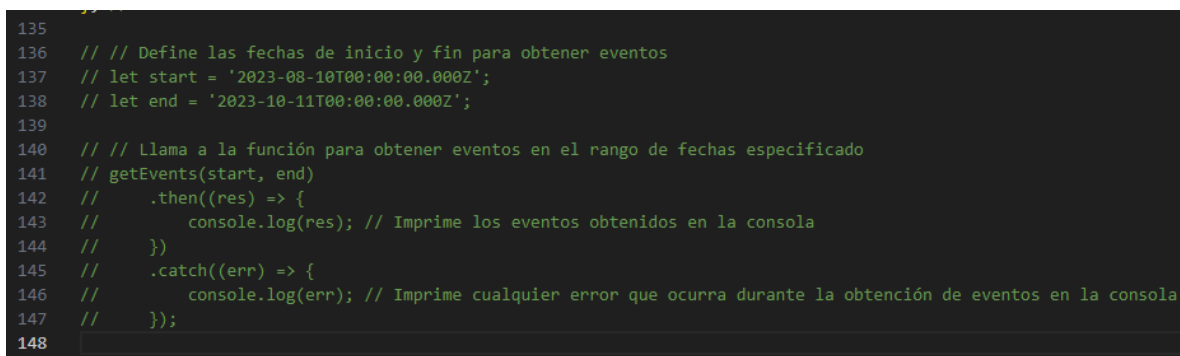
```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS S:\calendariogoogle> node google_calendar.js
```

Paso 111: De la información que nos envía vamos a buscar el “id”



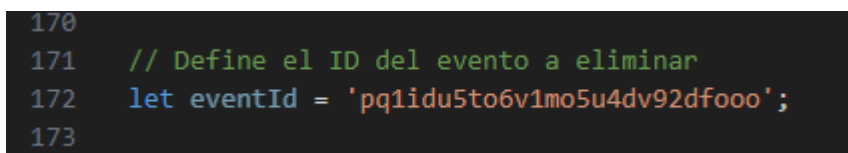
```
PS S:\calendariogoogle> node google_calendar.js
[
  {
    kind: 'calendar#event',
    etag: '"3390048670022000"',
    id: 'pq1idu5to6v1mo5u4dv92dfooo',
    status: 'confirmed',
    htmlLink: 'https://www.google.com/calendar/event...'
  }
]
```

Paso 112: Ahora que tenemos el id vamos de nuevo a comentar el código desde la (línea 136 hasta la 147)



```
135
136 // // Define las fechas de inicio y fin para obtener eventos
137 // let start = '2023-08-10T00:00:00.000Z';
138 // let end = '2023-10-11T00:00:00.000Z';
139
140 // // Llama a la función para obtener eventos en el rango de fechas especificado
141 // getEvents(start, end)
142 // .then((res) => {
143 //   console.log(res); // Imprime los eventos obtenidos en la consola
144 // })
145 // .catch((err) => {
146 //   console.log(err); // Imprime cualquier error que ocurra durante la obtención de eventos en la consola
147 // });
148
```

Paso 113: Ahora vamos a capturar el id y lo vamos a definir (línea 172)

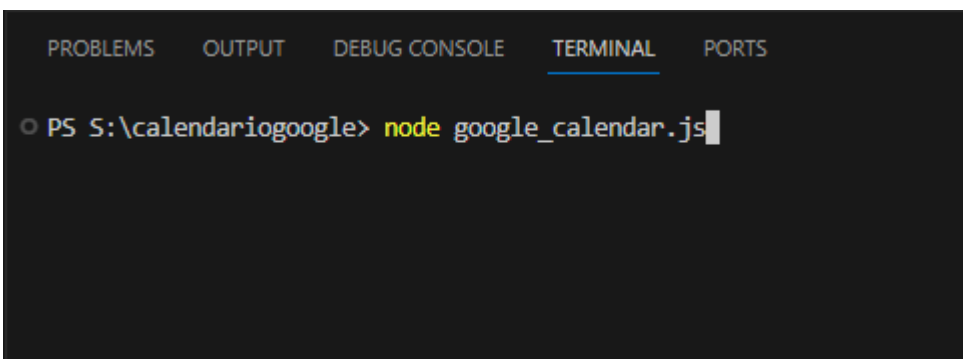


```
170
171 // Define el ID del evento a eliminar
172 let eventId = 'pq1idu5to6v1mo5u4dv92dfooo';
173
```

Paso 114: Ahora vamos a llamar a la función para eliminar el evento y le pasaremos el id (línea 175) y en caso de que se elimine correctamente imprimirá la respuesta en caso de que sea 1 se eliminó con éxito (línea 176 a 178) de lo contrario hará un catch que enviará un error (línea 179 a la línea 181)

```
172 let eventId = 'pq1idu5to6v1mo5u4dv92df0oo';
173
174 // Llama a la función para eliminar el evento
175 deleteEvent(eventId)
176   .then((res) => {
177     console.log(res); // Imprime la respuesta de la eliminación del evento (1 si se eliminó con éxito, 0 si falló)
178   })
179   .catch((err) => {
180     console.log(err); // Imprime cualquier error que ocurra durante la eliminación del evento
181   });
```

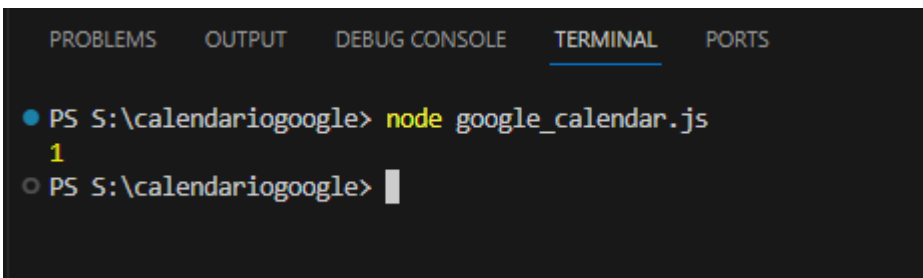
Paso 115: Ahora vamos a nuestro terminar y de nuevo vamos a ejecutar el node



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS S:\calendariogoogle> node google_calendar.js

Paso 116: Si la respuesta es 1 significa que se eliminó todo correctamente.



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS S:\calendariogoogle> node google_calendar.js
1
PS S:\calendariogoogle>

Paso 117: Podemos ver como se eliminó correctamente nuestro evento del calendario y con esto hemos finalizado nuestro proyecto donde aprendimos a crear eventos, ver la información de estos y eliminarlos usando una api.

