Sistema de Reportes Académicos con Google Cloud Functions

Fase Correspondiente:

Fase 2

Nombre del Equipo:

SN-16

Integrantes y Roles:

Andrea Victoria Castro Jiménez – Líder de Proyecto Jr.

Andrea Victoria Castro Jiménez – Ingeniero/a de Integración Serverless Jr.

Bryan Stephan Madriz Arteaga – Analista de Datos Académicos Jr.

Rafael Ignacio Funes Duarte – Desarrollador/a Backend Jr. (Funcional)

Elmer Geovany Quijano Hernández – QA/Documentador Técnico Jr.

Fecha de Entrega: 01/02/2026

Resumen ejecutivo

La culminación de esta fase representa la transición estratégica de procesos manuales hacia un ecosistema de automatización Serverless de Segunda Generación en Google Cloud. El objetivo principal fue cimentar una arquitectura robusta, capaz de procesar datos heterogéneos con intervención humana nula, garantizando la integridad de la información institucional desde su origen hasta su notificación final. Este proceso inició con la definición de un Contrato de Datos (Data Contract), el cual establece los estándares técnicos de validación y normalización de columnas, asegurando que solo la información de alta calidad sea procesada para el cálculo de KPIs críticos como promedios y estados de riesgo académico.

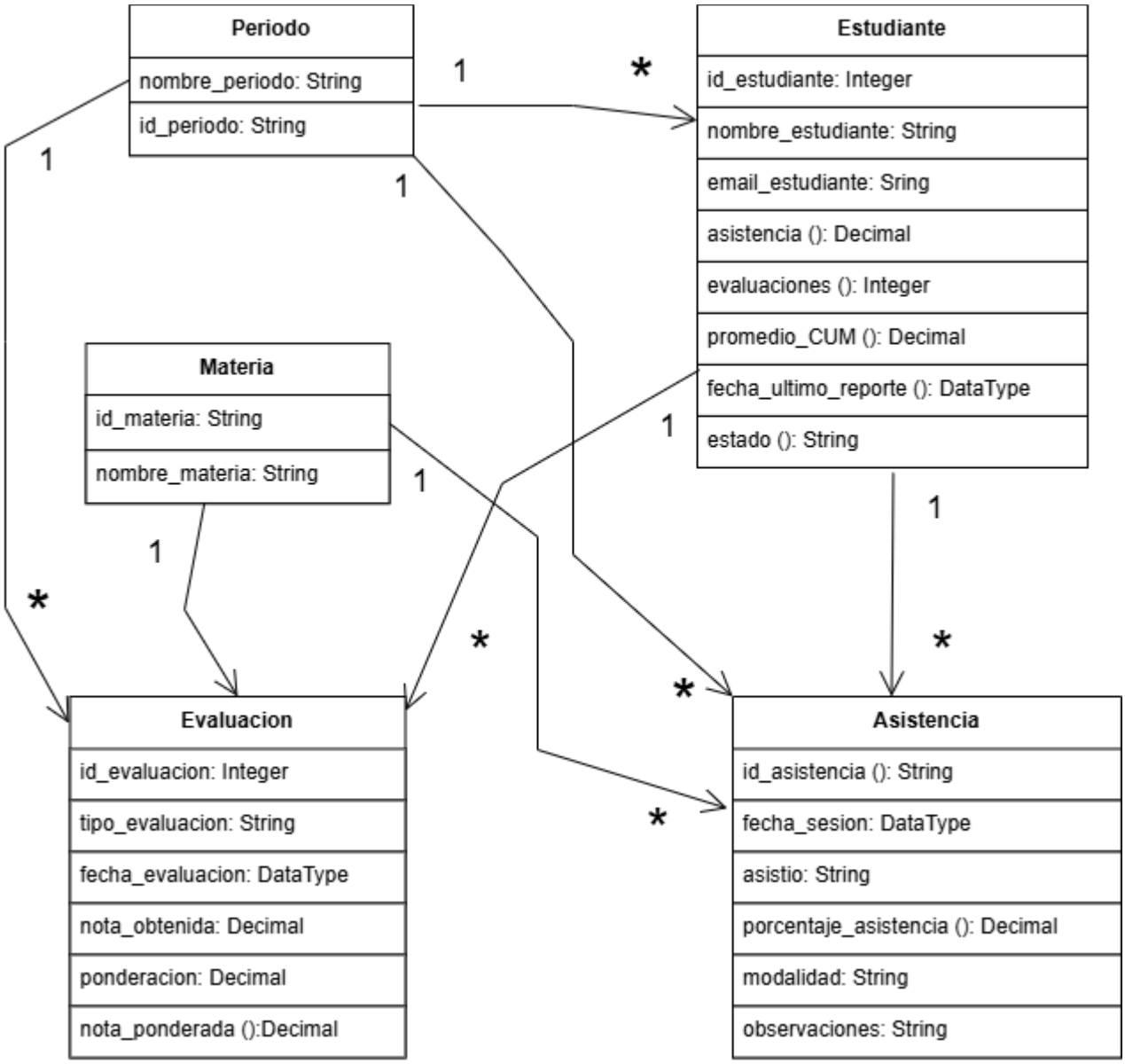
En el núcleo técnico, se desplegó una unidad de procesamiento mediante Cloud Run Functions bajo el entorno de Python 3.11, elegida por su eficiencia en la manipulación de estructuras de datos. La seguridad del sistema se blindó mediante la implementación de una identidad digital auditable: una Service Account (agente-integrador-dac). Bajo el "Principio de Mínimo Privilegio", este agente fue dotado exclusivamente con los roles de Editor, Cloud Functions Admin y Service Account User, eliminando la dependencia de cuentas personales y garantizando un flujo de datos seguro y privado.

Para validar la operatividad, se configuraron activadores HTTPS y se realizó un Smoke Test exitoso, confirmando que la infraestructura responde correctamente a las solicitudes externas. Asimismo, se integró una capa de monitoreo mediante una Hoja de Logs operativa, permitiendo la gestión de errores en tiempo real y asegurando la trazabilidad de cada ejecución. Con la configuración de Cloud Scheduler para disparos programados y la validación de envío de reportes vía Gmail API, la infraestructura queda plenamente operativa. Este hito garantiza un sistema escalable, de bajo costo y alta fidelidad, listo para transformar la gestión de reportes de la DAC en un proceso inteligente, auditable y totalmente automatizado.

Desarrollo del entregable

1. Contrato de Datos y Diagrama de Clases

Diagrama de clases: [Draw.io url]



- **id_estudiante** - Tipo: Integer. No permite nulos, ID único por cada estudiante.
- **nombre_estudiante** - Tipo: String. No permite nulos, acepta mayúsculas y minúsculas.

- **email_estudiante** - Tipo: String. El email debe ser ingresado en minúsculas. No acepta nulos.
- **asistencia** - Tipo: Decimal. Fórmula qué toma datos de porcentaje_asistencia para calcular el porcentaje total de asistencia.
- **evaluaciones** - Tipo: Integer. Fórmula qué toma datos de nota_obtenida en los que se ha ingresado datos (si no se ingresa evaluación, no toma la evaluación dentro del total).
- **promedio_CUM** - Tipo: Decimal. Fórmula qué toma los datos de nota_ponderada para calcular el CUM del estudiante.
- **fecha_ultimo_reporte** - Tipo: DataType. Fórmula qué toma los datos de la última fecha reportada en fecha_sesion.
- **estado** - Tipo: String. Formula qué toma los datos de asistencia y promedio_CUM.
- **nombre_periodo** - Tipo: String. No acepta nulos, sino data predefinida anteriormente.
- **id_periodo** - Tipo: String. No acepta nulos, sino data predefinida anteriormente.
- **id_materia** - Tipo: String. No acepta nulos, sino data predefinida anteriormente.
- **nombre_materia** - Tipo: String. No acepta nulos, sino data predefinida anteriormente.
- **id_evaluacion** - Tipo: Integer. Número único de registro asignado a cada una de las evaluaciones ingresadas. No acepta nulos.
- **tipo_evaluacion** - Tipo: String. No acepta nulos, sino data predefinida anteriormente.
- **fecha_evaluacion** - Tipo: DataType. Campo ingresado manualmente en formato dd/mm/aaaa.
- **nota_obtenida** - Tipo_ Decimal. Campo ingresado manualmente en formato decimal (ejemplo: 9.50). No acepta nulos.
- **ponderacion** - Tipo: Decimal. Campo ingresado manualmente en formato decimal (ejemplo: 0.20). No acepta nulos.
- **nota_ponderada** - Tipo: Decimal. Campo formulado en base a los datos de nota_obtenida y ponderacion. No acepta nulos.

- **id_asistencia** - Tipo: String. ID único de registro asignado a cada una de las asistencias registradas por cada estudiante. No acepta nulos.
- **fecha_sesion** - Tipo: DataType. Fecha ingresada manualmente en formato dd/mm/aaaa. No acepta nulos.
- **asistio** - Tipo: String. Campo de selección desplegable en base a dos criterios, Si o No. No acepta nulos.
- **porcentaje_asistencia** - Tipo:Decimal. Fórmula que toma los datos de *asistio*. No acepta nulos.
- **modalidad** - Tipo: String. No acepta nulos, sino data predefinida anteriormente
- **observaciones** - Tipo:String. No acepta nulos, sino data predefinida anteriormente

Diccionario de KPIs

- **asistencia**: calcula el porcentaje total de asistencia de cada uno de los estudiantes durante un ciclo académico.
- **promedio_CUM**: suma de todas las evaluaciones realizadas por el estudiante durante su ciclo académico.
- **estado**: se determina en base a los resultados de Asistencia y Promedio (CUM). Se calcula como Activo (mayor o igual a 6 en CUM y mayor o igual a 80% en asistencia), En Riesgo (entre 5 y 6 en CUM y entre 50% y 80% en asistencia) y Retirado (menor o igual a 5 en CUM y menor o igual a 50% en asistencia)
    - Cualquier estudiante dentro de la categoría En Riesgo son una alerta para el decano estudiantil.
    - Estudiantes en categoría Retirado deben de ser retirados del reporte en el siguiente ciclo estudiantil.

2. Infraestructura y Configuración Cloud

Configuración de Google Cloud Platform

Para la ejecución de esta fase de implementación técnica, se ha configurado un entorno de pruebas controlado (Sandbox) en GPC.

Ya que este proyecto todavía se encuentra en etapa de Prototipado Académico, se ha utilizado una cuenta personal de Gmail para la creación del proyecto y la gestión de recursos.

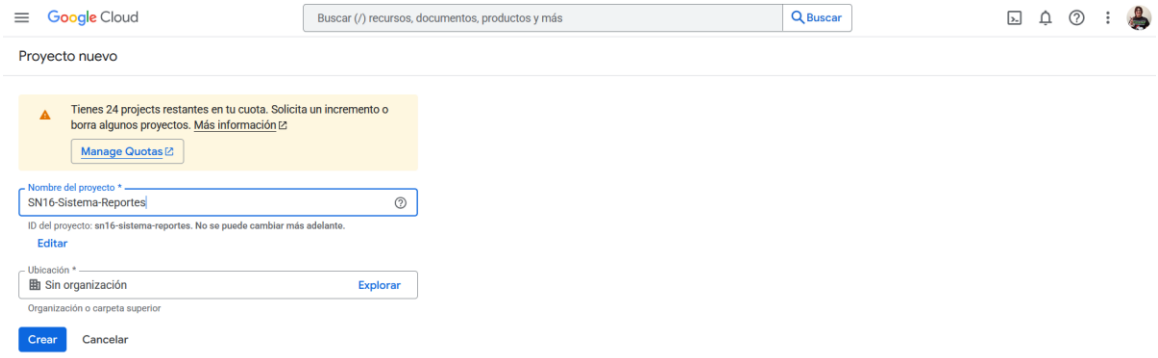
Esta decisión se toma para evitar riesgos de seguridad en la infraestructura productiva de la DAC (Dirección Académica Central) y para garantizar total autonomía en la configuración de servicios serveless sin afectar los límites de cuotas institucionales. En una fase de producción real, estos recursos serían migrados a la organización oficial bajo las políticas de gobernanza de TI.

Creación del Proyecto

En este paso se realizó el aprovisionamiento del espacio de trabajo donde residirán todos los servicios.

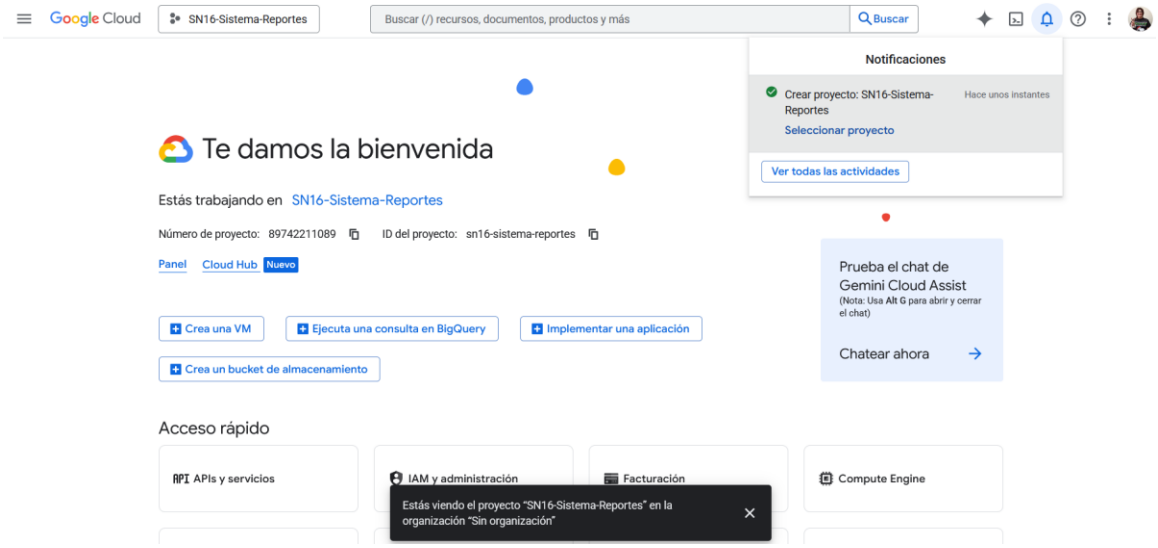
Para la creación del proyecto, es necesario tener una cuenta de Google activa y ligar una tarjeta ya sea de débito o crédito, aunque solo se utilice los servicios gratuitos ya que es esencial para la plataforma y corrobora que el usuario no es un robot.

Primero al entrar en la consola seleccionamos nuevo proyecto que se encuentra en la parte superior izquierda, esto nos abrirá una ventana emergente donde se nos pide que coloquemos el nombre del proyecto.



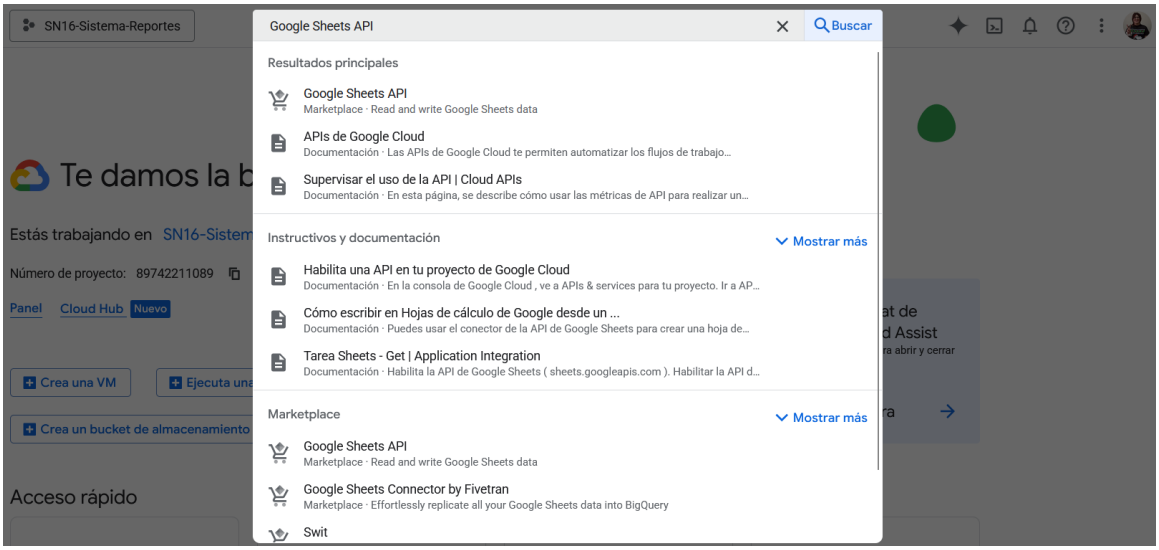
Nombre del Proyecto: SN16-Sistema-Reportes

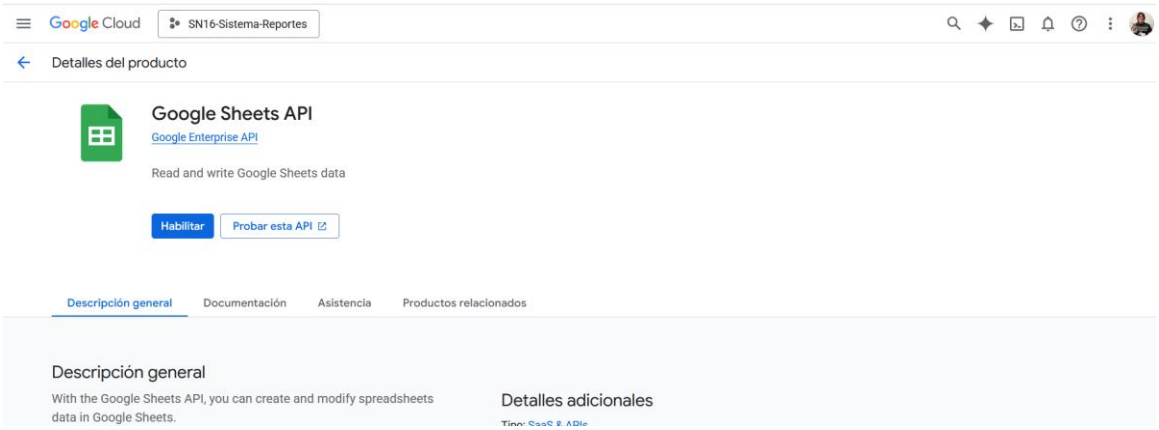
ID del Proyecto: sn16-sistema-reportes

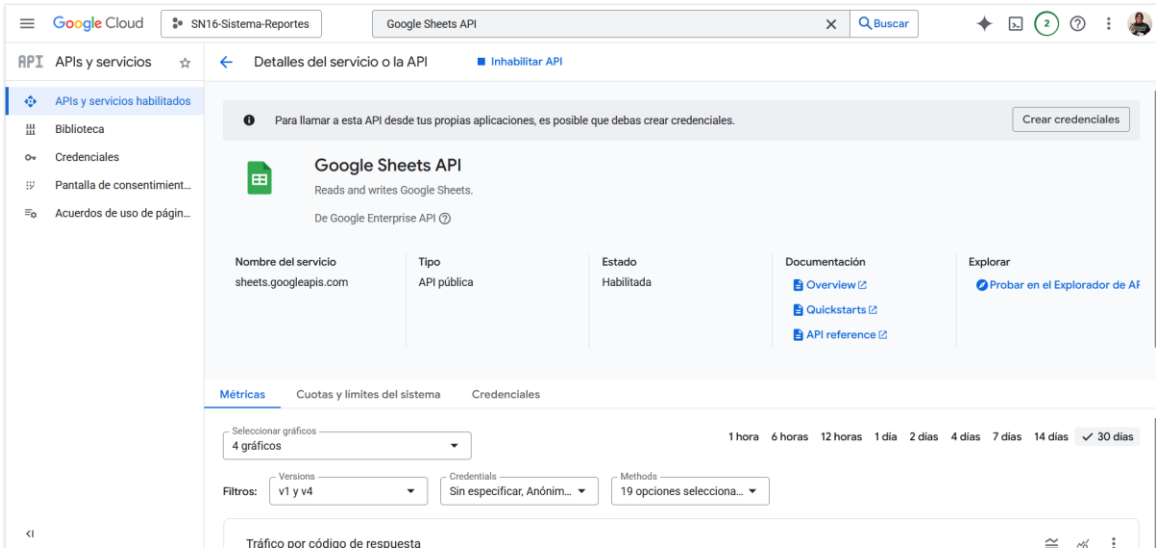


Para permitir la interoperabilidad entre la función de procesamiento y las herramientas de Google Workspace (Sheets y Gmail), se procedió a habilitar los endpoints necesarios.

En el buscador de la parte superior colocamos el nombre de la API o servicio que necesitamos, al presionar enter tenemos la opción de habilitar o deshabilitar los servicios seleccionados.

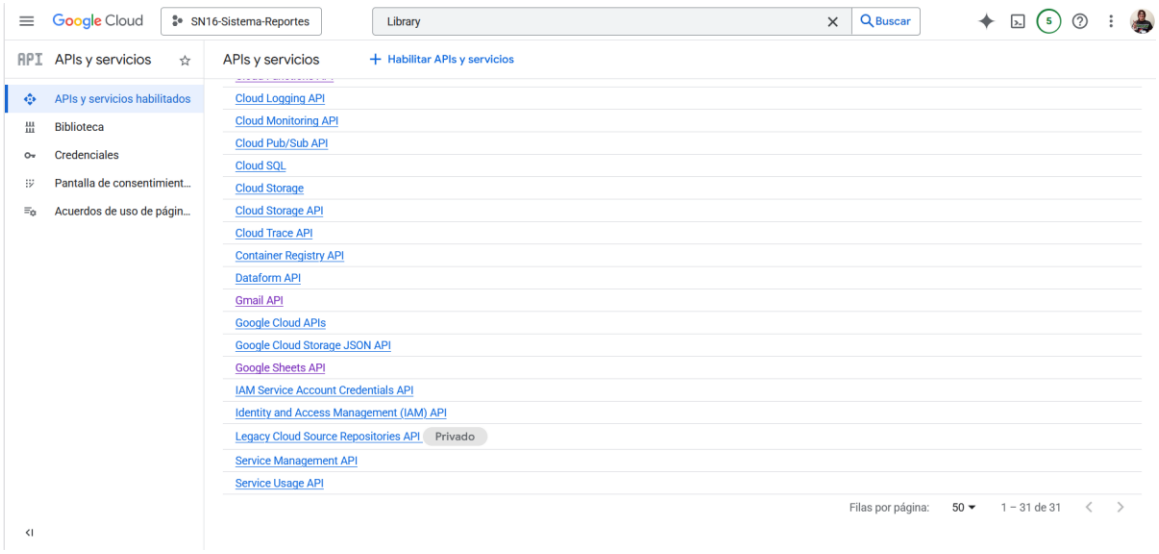






APIs Activadas:

1. **Google Sheets API:** Para la extracción de datos académicos.
2. **Gmail API:** Para la automatización del envío de reportes finales.
3. **Cloud Functions API:** Motor de ejecución para la lógica de procesamiento.
4. **Cloud Build API:** Necesaria para la gestión de despliegues y contenedores de la función.

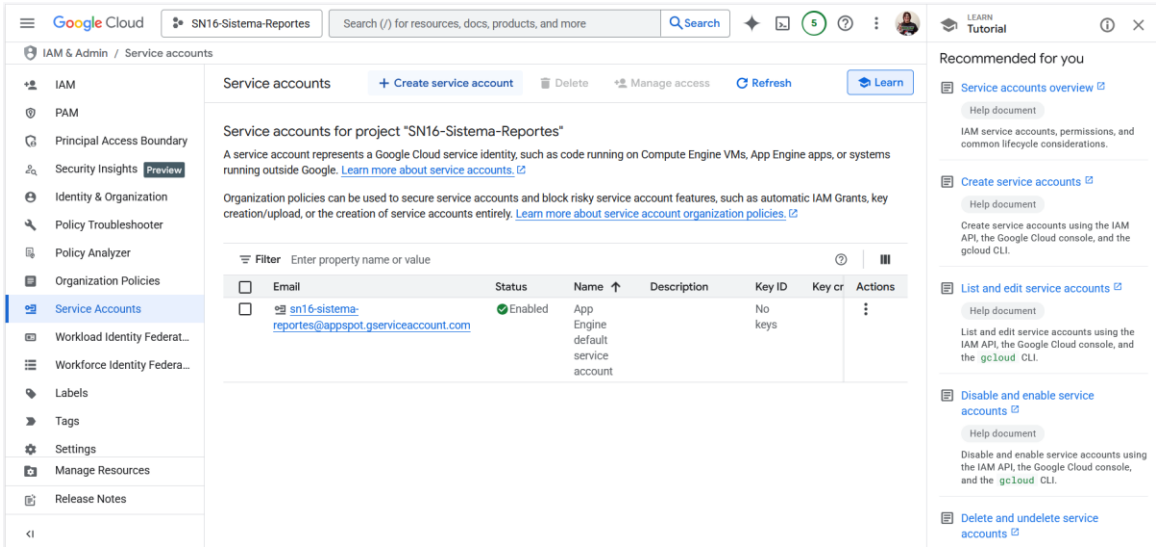


Seguridad y Permisos

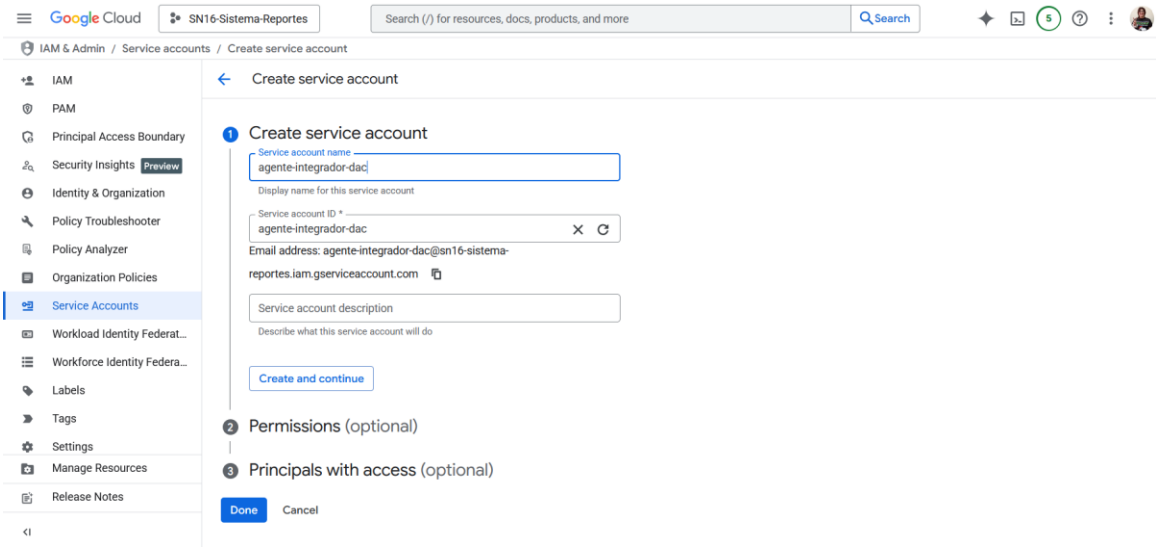
Configuración de la Identidad de Servicio (Agente de Integración DAC)

Para garantizar la autonomía del sistema y cumplir con los estándares de trazabilidad de la DAC, se procedió a crear un Service Account . Este componente actúa como la entidad digital del proyecto, permitiendo que los procesos se ejecuten sin depender de una cuenta de usuario humana.

Para crear el servicio se selecciona el menú y busca **IAM & Admin** > **Service Accounts**.



Hacer click en + CREATE SERVICE ACCOUNT de la parte superior, se llena el formulario avanzamos en Crear y Continuar de la parte inferior.



Nombre de la Identidad: agente-integrador-dac

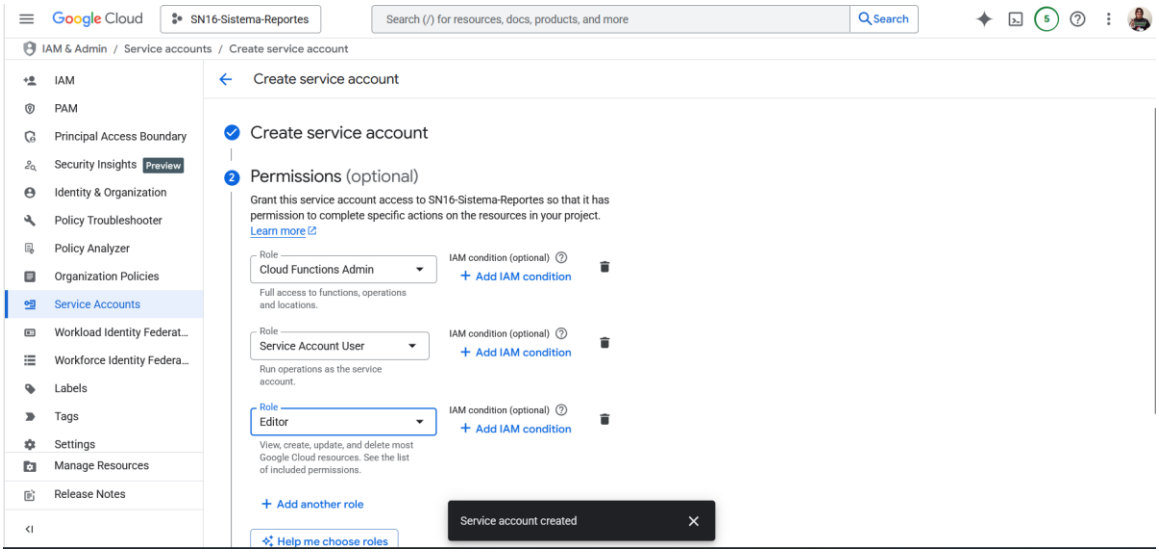
Propósito: Actuar como el nexo entre la capa de cómputo (Cloud Functions) y la capa de datos (Google Sheets), asegurando que el flujo de información sea auditable.

Siguiendo el Principio de Mínimo Privilegio (PoLP), se asignaron los siguientes roles específicos para limitar el acceso del agente solo a lo estrictamente necesario:

- **Administrador de Cloud Functions:** Para la gestión del entorno de ejecución.

- **Usuario de cuenta de servicio:** Para permitir la suplantación de identidad controlada durante el despliegue.
- **Editor:** Para otorgar permisos de lectura y escritura en las fuentes de datos académicas y el uso de la Gmail API.

Luego de asignar el nombre de la identidad de servicio en la parte de roles seleccionamos el rol uno a uno.



Despliegue de la Unidad de Cómputo (Cloud Run Function)

En esta etapa, se realizó el aprovisionamiento de la unidad de cómputo. Debido a las actualizaciones recientes en la plataforma de Google Cloud, se optó por **Cloud Run functions (2nd Gen)**, la cual representa la evolución de las funciones serverless, ofreciendo mayor estabilidad y escalabilidad para el procesamiento de los reportes de la DAC.

Proceso de Creación Técnica:

1. **Selección de Entorno**: Se eligió el entorno de ejecución Python 3.11, garantizando compatibilidad con las librerías de análisis de datos que utilizará el equipo de Backend.

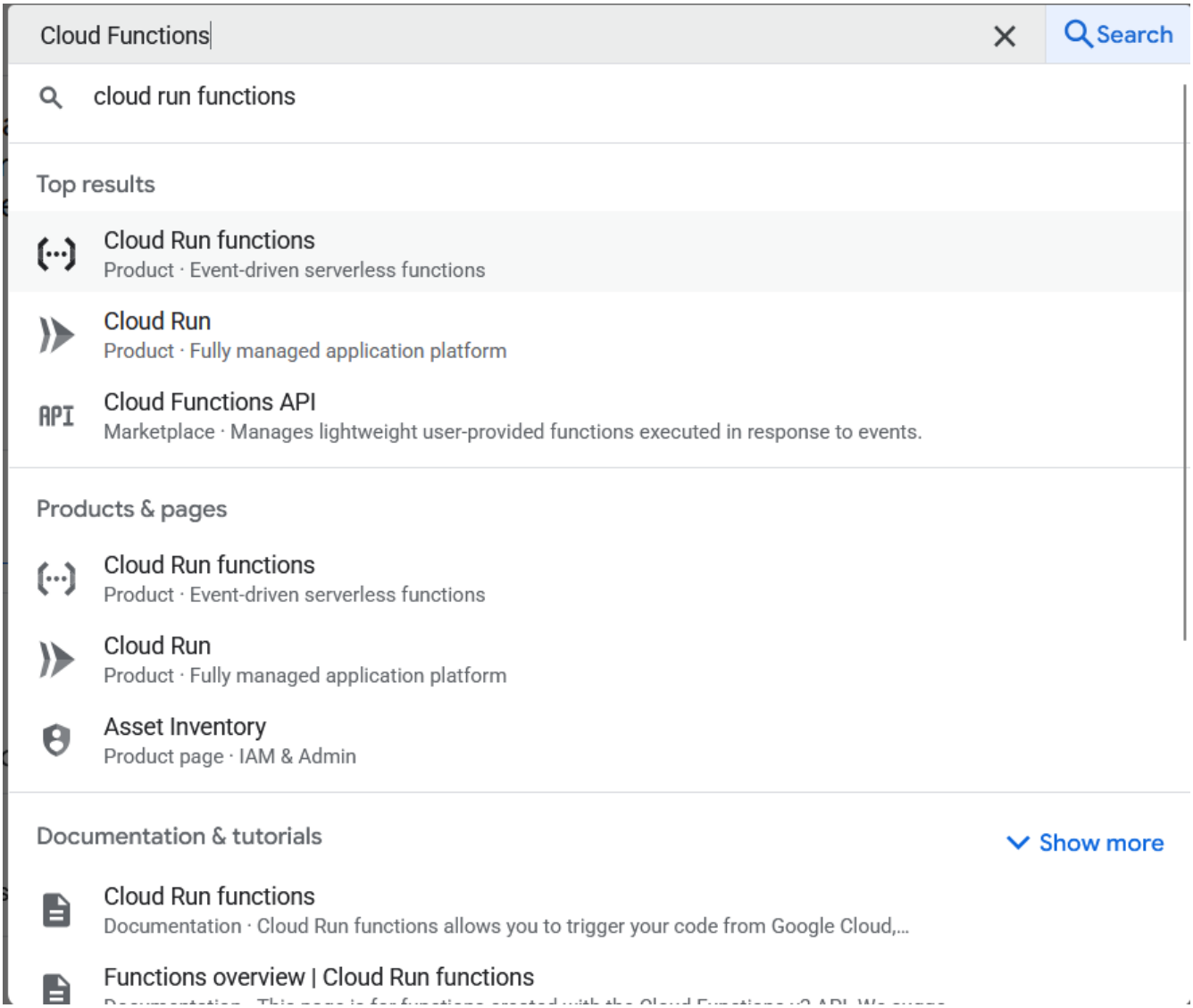
2. **Configuración de Disparador (Trigger):** Se estableció un endpoint HTTPS

    - El endpoint NO permite acceso público.
    - La invocación está restringida mediante IAM.
    - Solo la Service Account del Cloud Scheduler tiene el rol: Cloud Functions Invoker

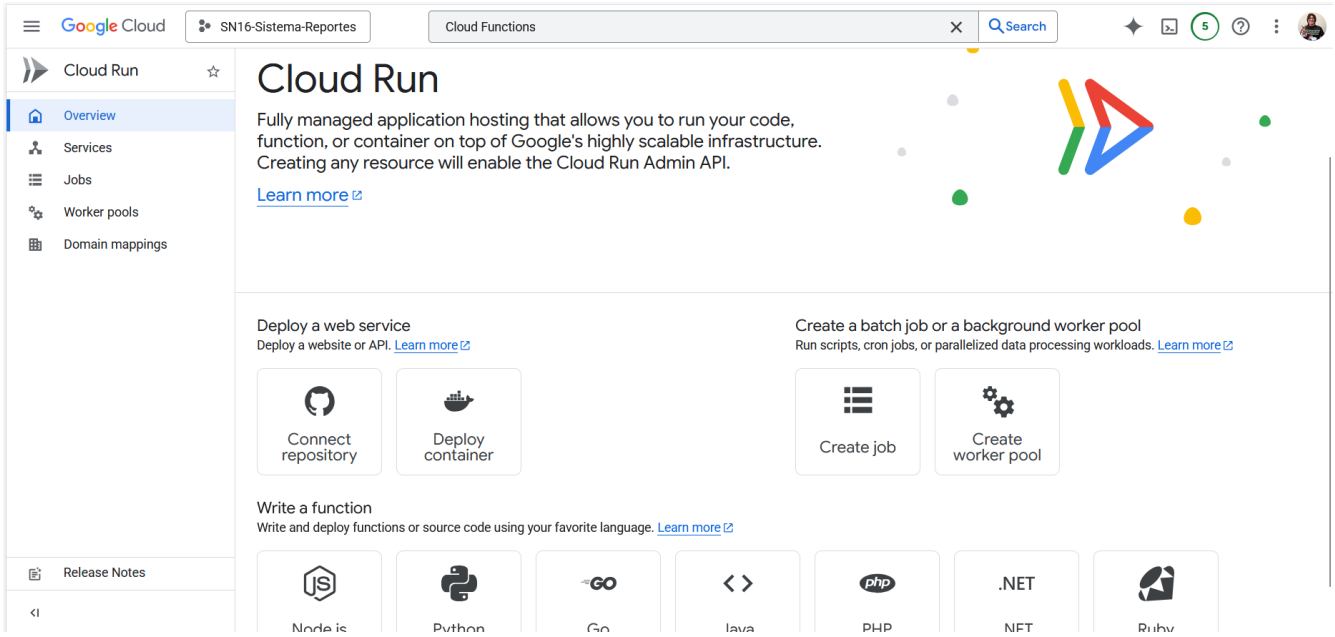
Se utiliza autenticación OIDC configurada en el Scheduler.

3. **Vinculación de Seguridad:** En la sección de configuración avanzada, se asignó la Service Account agente-integrador-dac como la identidad de ejecución, asegurando que el código Python tenga los permisos necesarios para interactuar con Google Workspace en la Fase 3.

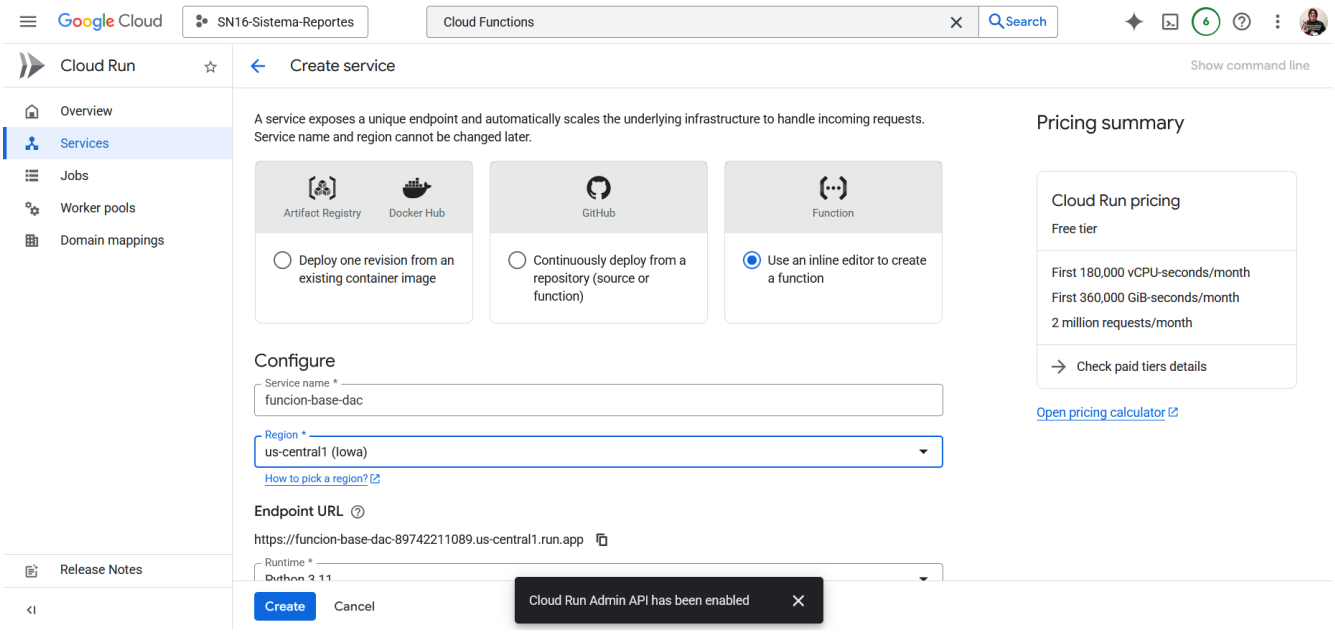
Para llevar a cabo esta configuración, primero se colocó en el buscador, Cloud Functions se elige la primera opción como muestra la imagen.

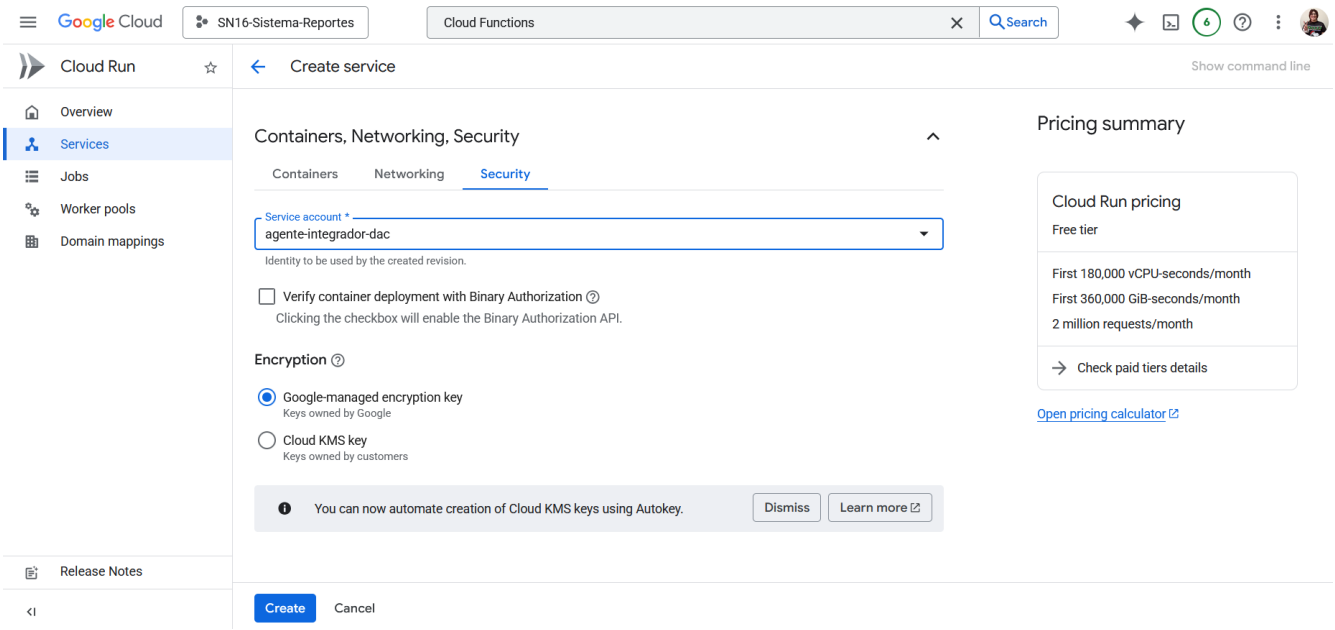


Para escribir una nueva función debemos seleccionar el lenguaje, en este caso Python



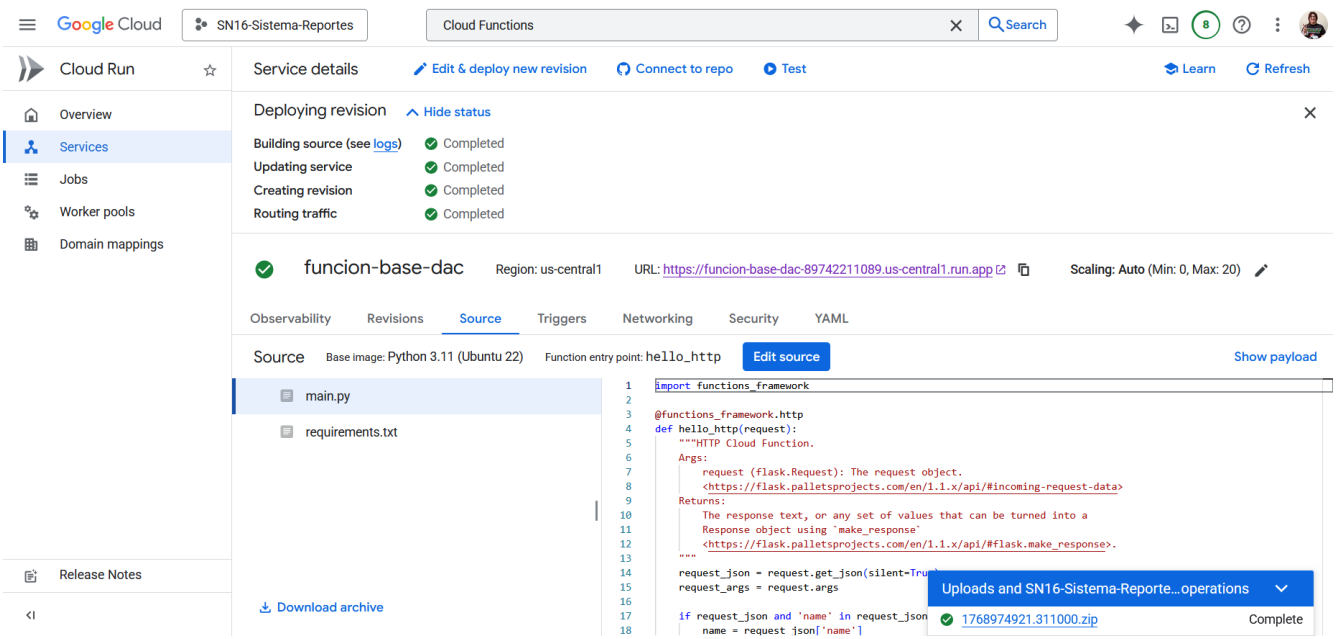
Cuando seleccionamos el lenguaje nos redirige a una nueva vista donde configuramos el nombre de la función y la región, y en la pestaña de seguridad colocamos el servicio al que está ligada. Aquí ya no es necesario configurar un trigger https específico por que actualmente ya lo crea por defecto para aceptar peticiones web.





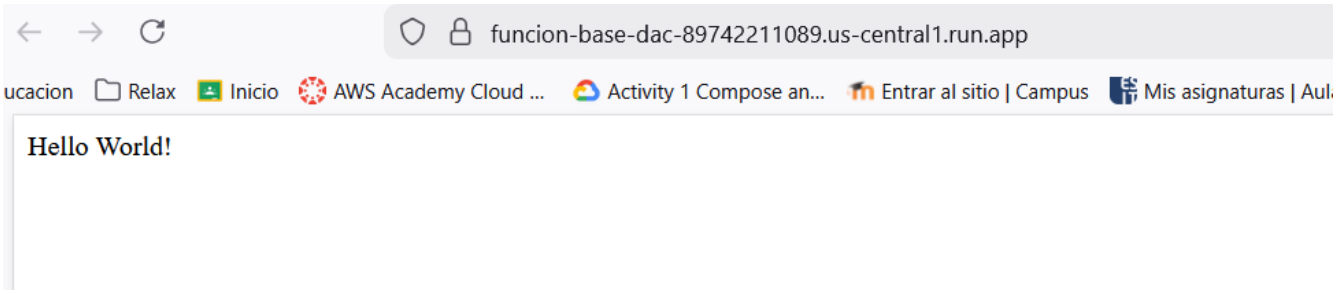
Validación de Infraestructura y Smoke Test

Para finalizar la cimentación de la Fase 2, se realizó un **Smoke Test** invocando la URL pública generada automáticamente por el servicio.



- Resolución de Incidencia: Durante el primer despliegue, el sistema presentó una página de reserva (placeholder). Se procedió a realizar un Redeploy manual del código fuente desde la consola de Cloud Run para forzar la compilación del archivo *main.py.*

- **Resultado Final:** La validación fue exitosa. Al acceder al enlace, el servidor respondió con el mensaje **"Hello World!"**, confirmando que el intérprete de Python, la red HTTPS y la identidad del agente están operando de manera integrada.



3. Implementación del Backend

A continuación se describe el desarrollo de un **proyecto funcional en Python** cuyo objetivo es **leer información almacenada en una hoja de Google Sheets** y mostrarla en la consola.

Este proyecto demuestra la integración entre Python y servicios en la nube de Google, específicamente **Google Sheets API**, permitiendo el acceso automatizado a datos académicos.

Objetivo general

Desarrollar una aplicación en Python capaz de conectarse a una hoja de Google Sheets y mostrar en consola los datos de los estudiantes registrados.

Objetivos específicos

- Conectarse de forma segura a Google Sheets mediante una **Service Account**.
- Leer datos estructurados desde una hoja de cálculo en la nube.
- Procesar los registros obtenidos.
- Mostrar la información de manera clara y ordenada en consola.
- Implementar una solución modular y fácil de mantener.

Descripción General del Proyecto

El proyecto consiste en un script en Python que accede a una hoja de Google Sheets previamente configurada y compartida con una **Service Account**.
Los datos son obtenidos fila por fila y mostrados en consola, permitiendo verificar la información académica de los estudiantes.

La solución es escalable y puede ser extendida en el futuro para exportar datos a PDF, enviar correos o generar reportes.

La hoja de Google Sheets utilizada contiene los siguientes campos:

Campo	Descripción
Id	Identificador del registro
Nombre	Nombre completo del alumno
Email	Correo electrónico del alumno
Período	Período académico
Materia	Asignatura cursada
Nota	Calificación obtenida

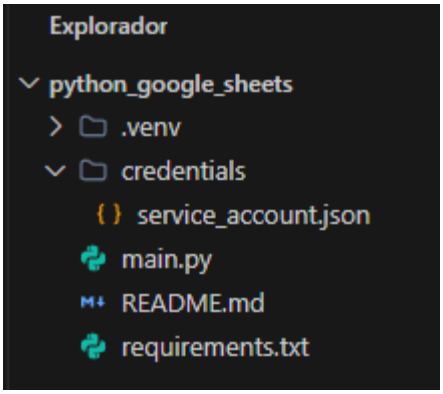
Los encabezados se encuentran en la primera fila de la hoja, lo que permite su lectura automática desde Python.

Tecnologías Utilizadas

- **Lenguaje de programación:** Python 3
- **Librerías principales:**
  - `gspread`
  - `oauth2client`
- **Servicios de Google:**
  - Google Sheets API
  - Google Drive API
- **Plataforma:** Google Cloud Platform

Arquitectura del Proyecto

El proyecto está estructurado de la siguiente manera:



main.py: Script principal que realiza la conexión y lectura de datos.

requirements.txt: Archivo con las dependencias necesarias.

credentials: Contiene las credenciales de la cuenta de servicio.

README.md: Guía de instalación y ejecución del proyecto.

credentials: contiene el archivo service_account.json

Pasos rápidos para usarlo en un proyecto con Visual Studio Code:

1. **Instalar dependencias:**

   py -m pip install gspread oauth2client

2. **Configurar Google:**
   - Crear proyecto en Google Cloud
   - Habilitar:
     - Google Sheets API
     - Google Drive API
   - Crear **Service Account**
   - Descargar service_account.json

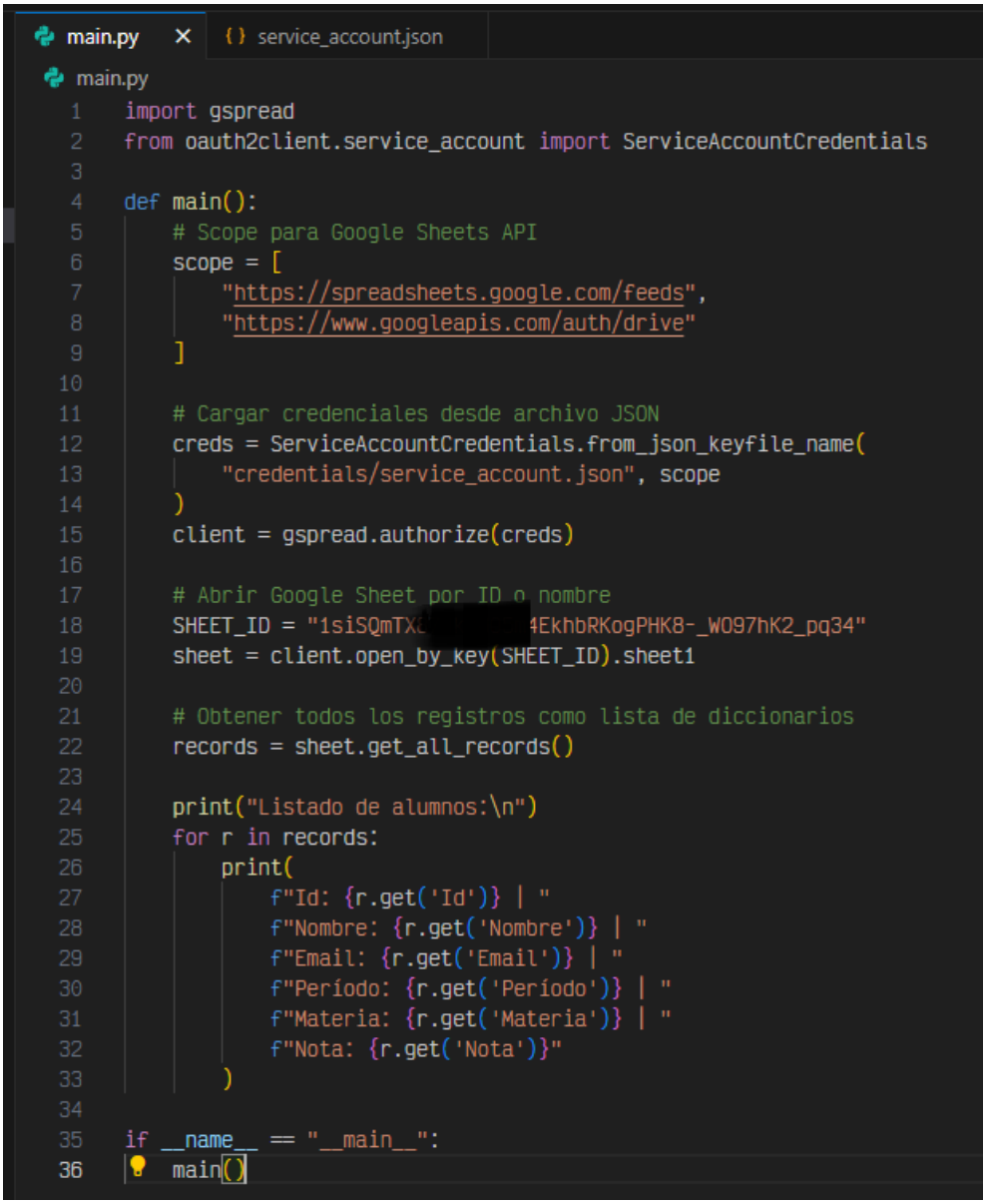
3. **Colocar credenciales en la carpeta:**

   credentials/service_account.json

4. **Compartir la hoja**
   Comparte tu Google Sheet con el email de la **Service Account**.

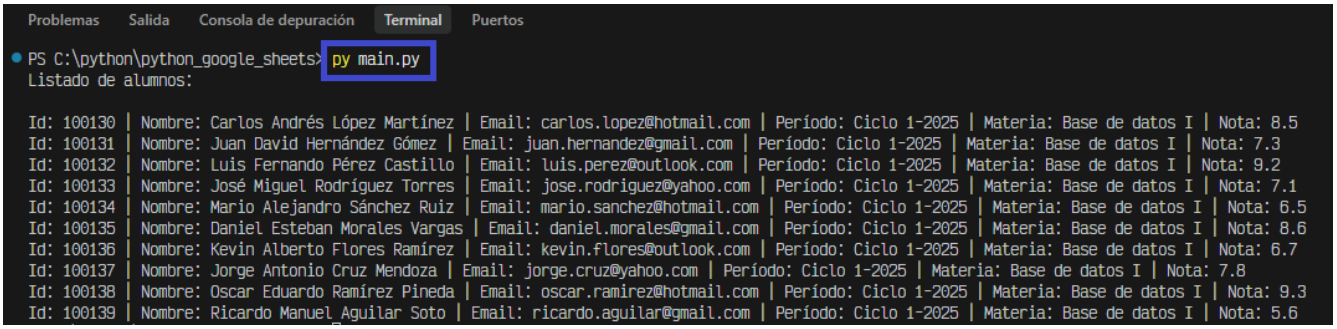
5. **Editar main.py**
   Coloca el ID de tu Google Sheet:

   SHEET_ID = "TU_ID_AQUI"



6. Ejecutar

   py main.py



Funcionamiento del Sistema

1. El sistema carga las credenciales de la **Service Account**.
2. Se autentica contra los servicios de Google.
3. Accede a la hoja de Google Sheets mediante su ID. (https://docs.google.com/spreadsheets/d/1siSQmTX87HkddO5m4EkhbRKogPHK8-_WO97hK2_pq34)
4. Lee todos los registros disponibles.
5. Imprime en consola la información de cada alumno con sus respectivos datos académicos.

Seguridad y Acceso

El acceso a la hoja de cálculo se realiza mediante una **Service Account**, evitando el uso de credenciales personales.

La hoja de Google Sheets es compartida únicamente con el correo de la cuenta de servicio, garantizando un acceso controlado y seguro.

Resultados Obtenidos

- Lectura exitosa de los datos desde Google Sheets.
- Impresión correcta de los registros en consola.
- Integración funcional entre Python y Google Cloud.
- Proyecto estable y reutilizable para futuros desarrollos.

El proyecto cumple satisfactoriamente con los objetivos planteados, demostrando la capacidad de Python para integrarse con servicios en la nube y procesar información en tiempo real.

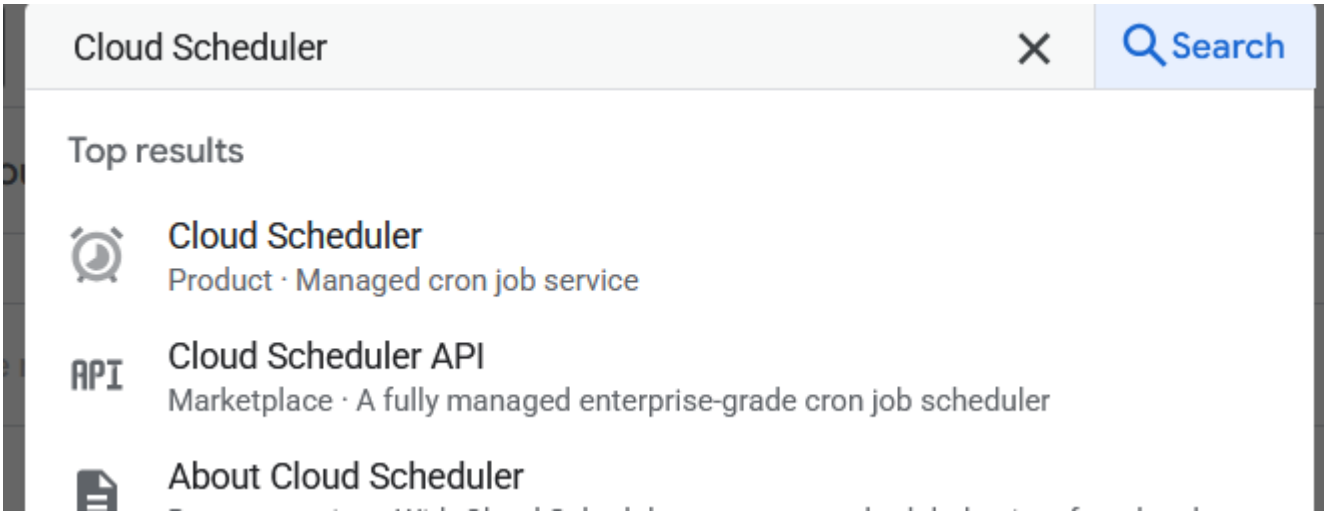
Esta solución se utilizará como la base para la implementación final que es la transformación de datos para la gestión académica, generación de reportes o enviarlo a un correo electrónico.

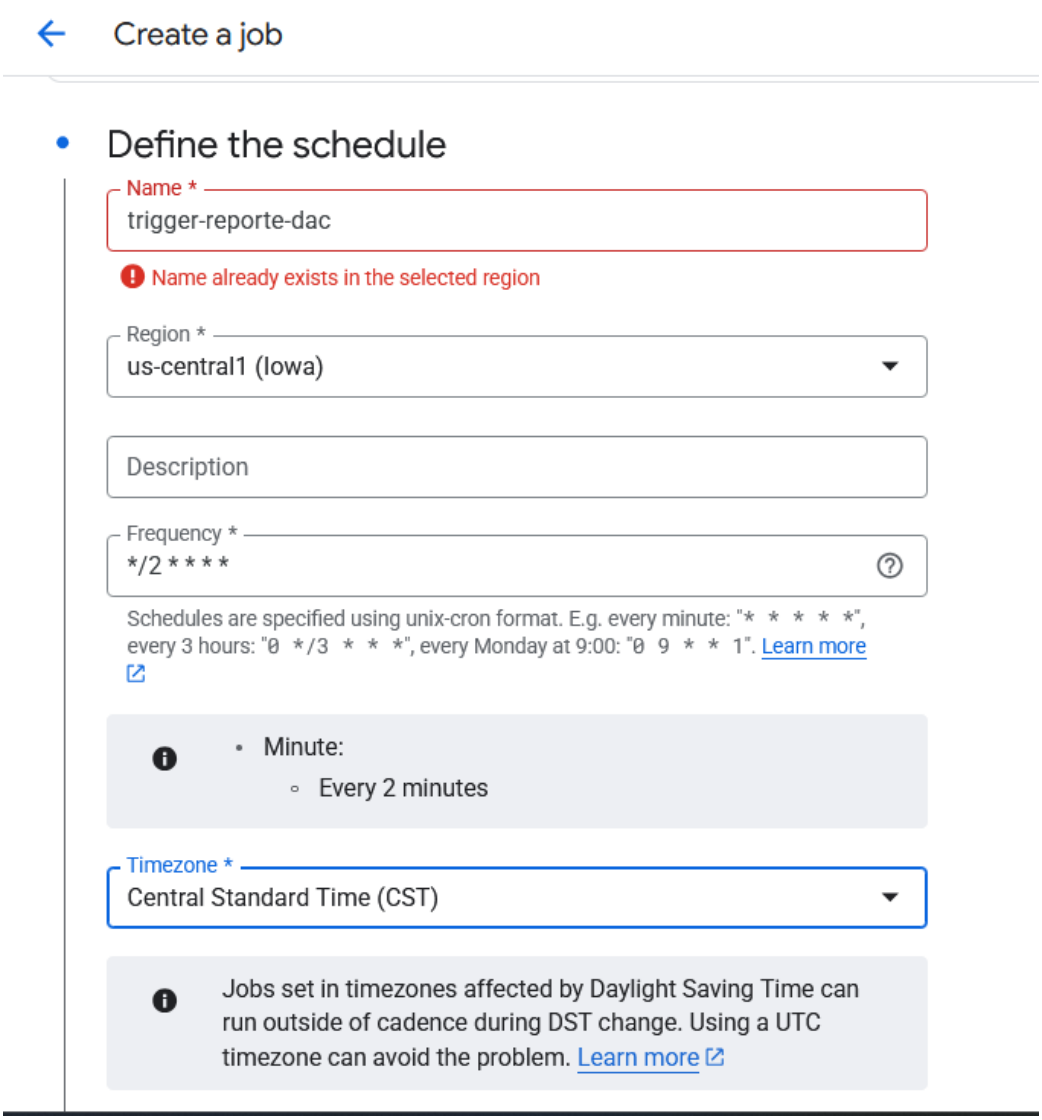
4. Automatización y Triggers

Configuración de Cloud Scheduler (Cron Job)

Para cumplir con los requerimientos de periodicidad de la **DAC**, se implementó un servicio de cronogramas gestionado mediante **Cloud Scheduler**. Este componente actúa como el "reloj maestro" del proyecto, eliminando la necesidad de activación manual y garantizando la entrega de reportes en tiempos institucionales precisos.

- **Lógica de Programación (Cron Expression):** Se utilizó el estándar unix-cron para definir la frecuencia de ejecución. La configuración permite una flexibilidad total, desde disparos en intervalos de minutos para pruebas de estrés, hasta programaciones semanales específicas (ej. lunes a las 08:00 AM).
- **Parámetros de Red:** El Scheduler fue configurado para realizar peticiones HTTP GET dirigidas al endpoint del servicio **funcion-base-dac**. Al operar en la misma región (us-central1), se optimiza la latencia de red y se asegura la coherencia geográfica de la infraestructura.





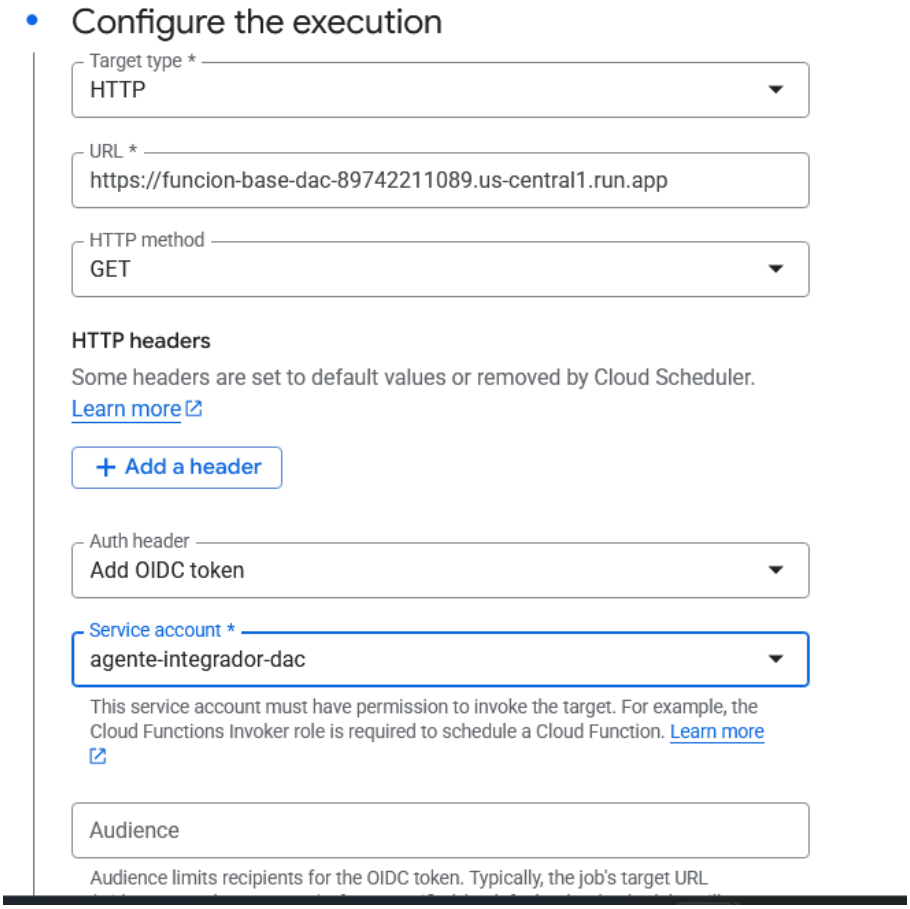
Seguridad y Autenticación de la Automatización

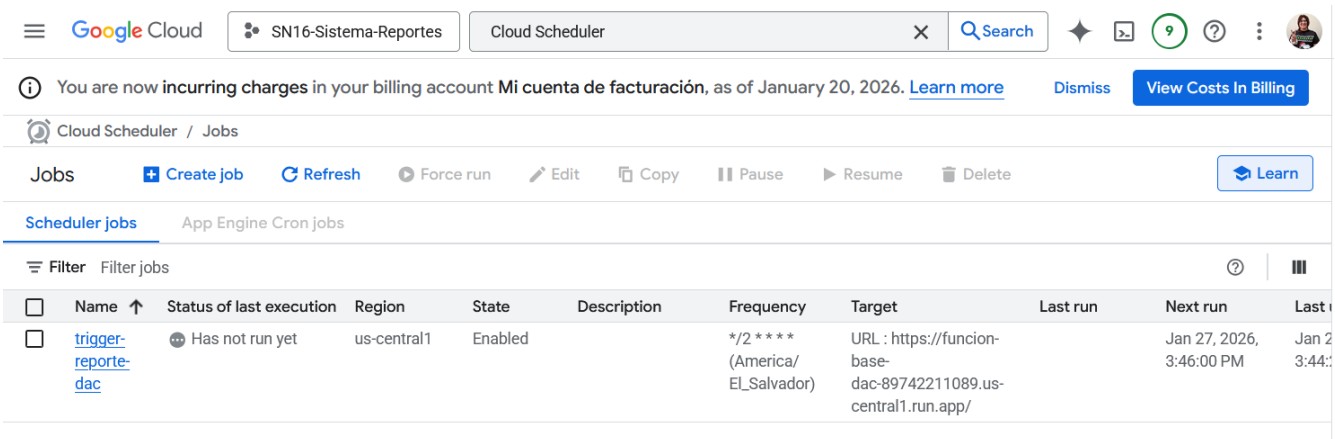
Un aspecto crítico de esta fase fue el robustecimiento de la seguridad en el disparo de la función. Para evitar que terceros puedan ejecutar el reporte fuera de tiempo, se configuró un protocolo de

Seguridad de Capa de Aplicación:

- **Token de Identidad OIDC:** Se habilitó el uso de tokens de **OpenID Connect (OIDC)**. Esto significa que cada vez que el Scheduler intenta "despertar" a la función, debe presentar una credencial digital válida.
- **Identidad Delegada:** El token es generado automáticamente por el **Agente de Integración DAC (agente-integrador-dac)**, vinculando la identidad de servicio creada en la Fase 1 con el

disparador automático. Esto garantiza que la función solo responda si la petición proviene de un servicio interno autorizado.

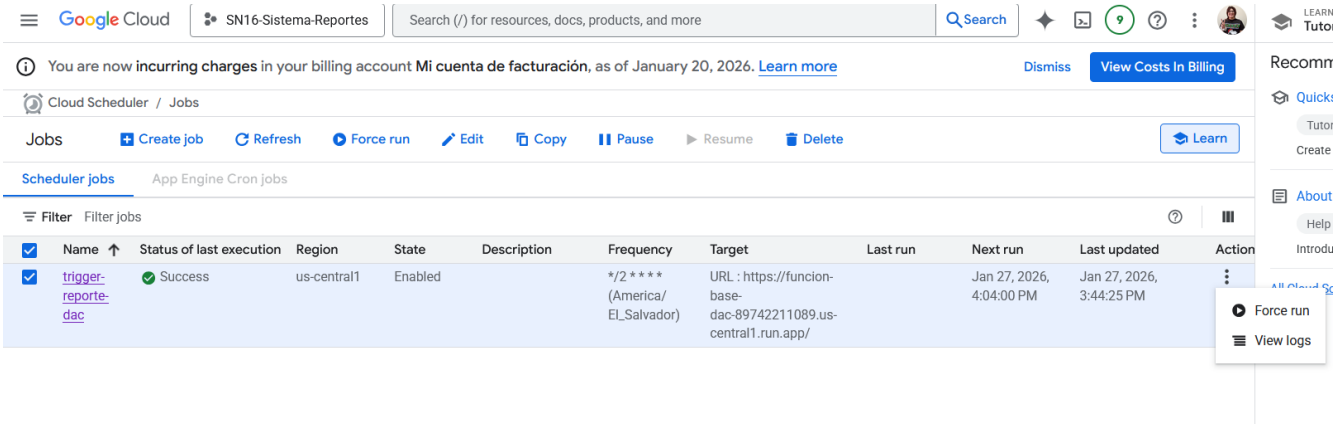


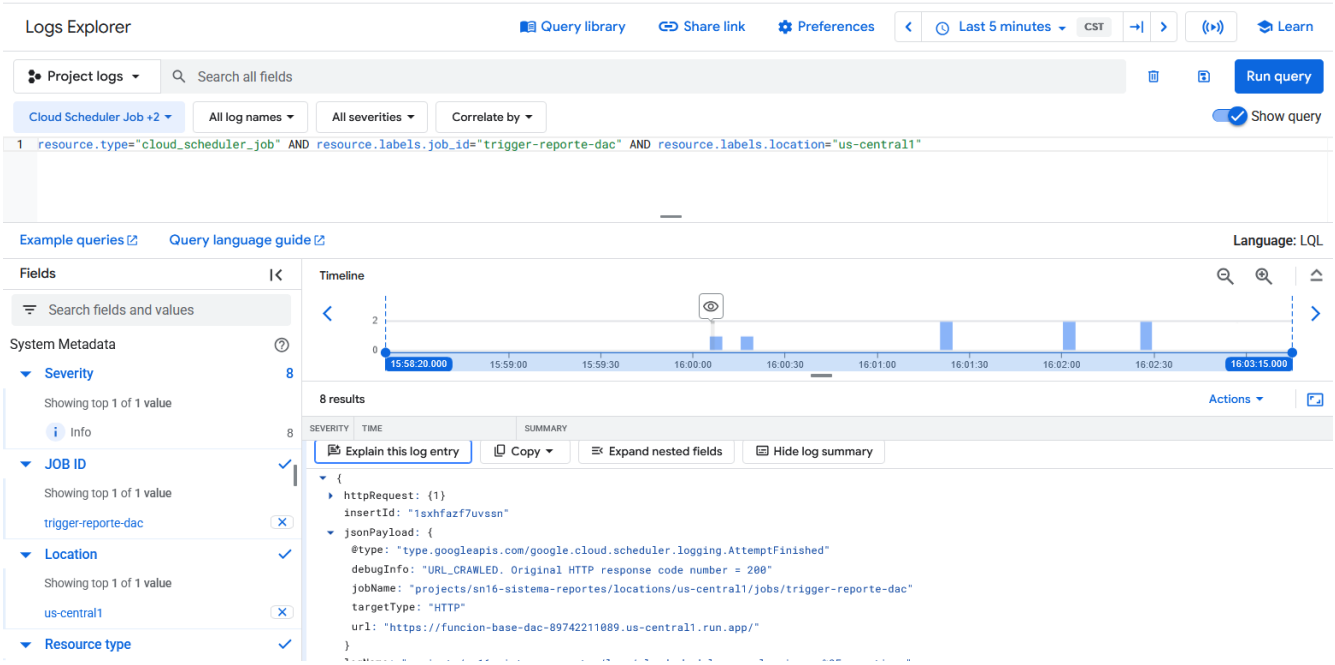


Validación de Ejecución (Smoke Test de Tiempo)

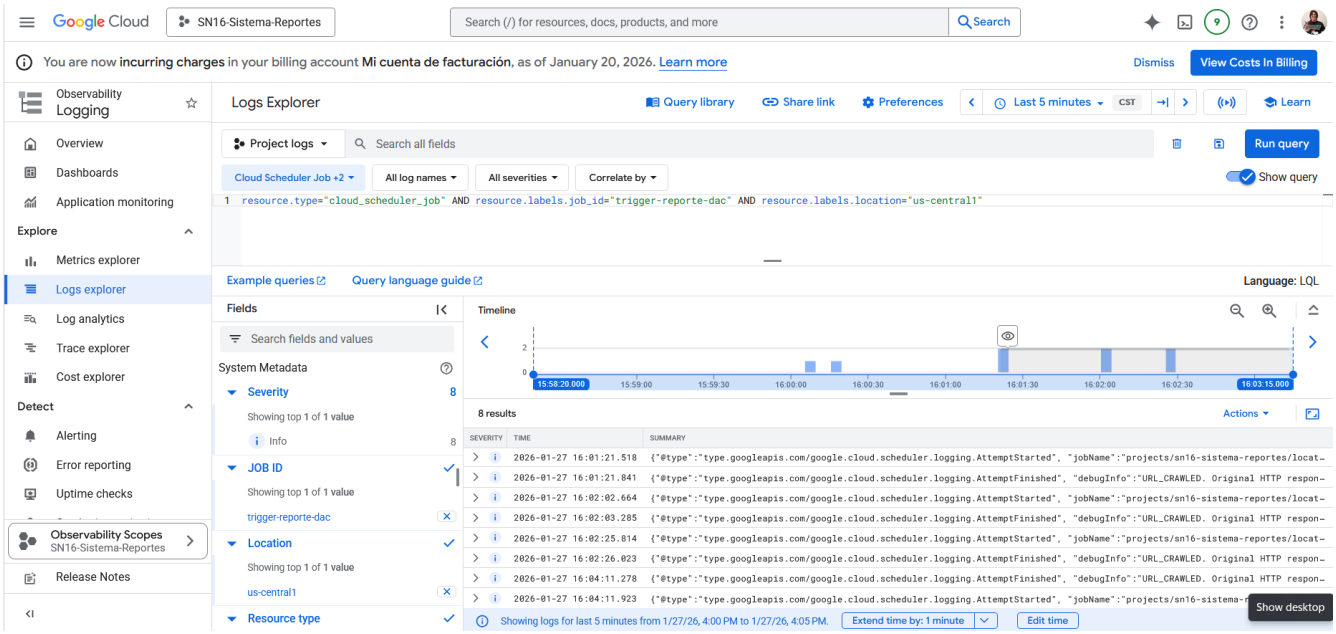
La validación no solo se realizó de forma manual, sino que se verificó la capacidad de "auto-disparo" del sistema.

- **Evidencia de Activación:** Mediante la función "Force Run", se comprobó que el orquestador logra comunicarse con la Cloud Run Function, recibiendo un código de estado 200 OK.





- **Trazabilidad:** Se confirmó que cada ejecución queda registrada en los logs de auditoría, permitiendo al equipo técnico monitorear el cumplimiento de los horarios de entrega de los reportes académicos.

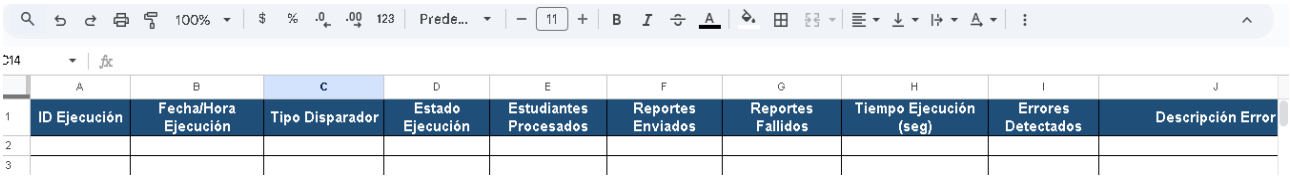


Optimización de Frecuencia y Control de Cuotas: Originalmente, el sistema se configuró con disparos cada 2 minutos para validar la conectividad en tiempo real. Sin embargo, para alinearse con las políticas de **Capa Gratuita (Free Tier)** de Google Cloud y evitar costos por exceso de invocaciones, la frecuencia se ajustó a un intervalo semanal.

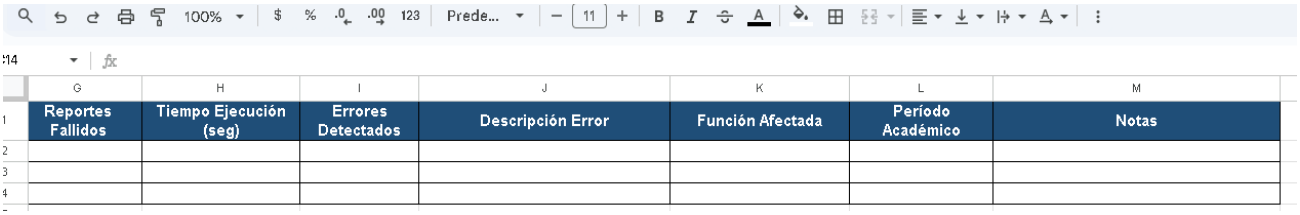
- **Configuración Final:** 0 8 * * 1 (Lunes a las 08:00 AM).
- **Justificación:** Este ajuste reduce el consumo de vCPU-segundos y memoria de la función, asegurando que el proyecto opere bajo el umbral de costo cero de la DAC.

5. Validación y Gestión de Errores

Creación de estructura de hoja de logs.



ID Ejecución	Fecha/Hora Ejecución	Tipo Disparador	Estado Ejecución	Estudiantes Procesados	Reportes Enviados	Reportes Fallidos	Tiempo Ejecución (seg)	Errores Detectados	Descripción Error



Reportes Fallidos	Tiempo Ejecución (seg)	Errores Detectados	Descripción Error	Función Afectada	Período Académico	Notas

Link de hoja de logs en drive [Estructura Logs.xlsx]

Casos de Prueba Sugeridos

Sistema de Reportes Académicos con Google Cloud Functions

Los siguientes casos de prueba han sido diseñados para validar el correcto funcionamiento de la infraestructura serverless implementada en Google Cloud Platform, específicamente orientados a la Cloud Run Function desarrollada en Python 3.11. Estos casos cubren aspectos críticos del sistema incluyendo la integración con Google Sheets API, validación de datos académicos, gestión de errores y seguridad.

1. Caso de Prueba CP-1: Conexión exitosa a Google Sheets

ID del Caso	CP-1
Nombre	Conexión exitosa a Google Sheets mediante Service Account
Objetivo	Verificar que la Cloud Function puede autenticarse correctamente con la Service Account agente-integrador-dac y establecer conexión con la hoja de Google Sheets configurada.
Precondiciones	• Service Account creada y configurada. • Google Sheets API habilitada en el proyecto • Hoja de cálculo compartida con el email de la Service Account • Archivo service_account.json disponible en la función
Pasos	1. Invocar la Cloud Function mediante su URL HTTPS 2. La función debe cargar las credenciales de service_account.json 3. Autenticarse contra Google Sheets API usando gspread y oauth2client 4. Abrir la hoja mediante el SHEET_ID configurado

	5. Retornar código HTTP 200 con mensaje de conexión exitosa
Resultado Esperado	• Código de respuesta: 200 OK • Mensaje: Confirmación de conexión exitosa a Google Sheets • Sin errores de autenticación en los logs
Prioridad	Alta

2. Caso de Prueba CP-2: Lectura y validación de datos académicos

ID del Caso	CP-2
Nombre	Lectura y validación de datos académicos según Contrato de Datos
Objetivo	Validar que la función lee correctamente los datos de estudiantes desde Google Sheets y aplica las reglas de

	validación definidas en el Contrato de Datos (tipos de datos, valores nulos, normalización).
Precondiciones	• Conexión exitosa a Google Sheets (CP-1 pasado) • Hoja con datos académicos válidos: Nombre, Carné, Correo, Nota1, Nota2, Nota3 • Contrato de Datos implementado en el código
Pasos	1. Ejecutar la función para leer todos los registros de la hoja 2. Aplicar validación de tipo de dato para cada columna (String para Nombre/Correo, Number para Notas) 3. Verificar normalización de correos a minúsculas 4. Validar que no existan valores nulos en campos requeridos 5. Registrar en logs los datos procesados correctamente
Resultado Esperado	• Todos los registros válidos se procesan sin errores • Correos normalizados a minúsculas • Datos de cada estudiante disponibles para cálculo de KPIs • Log con cantidad de registros procesados exitosamente

Prioridad	Alta

3. Caso de Prueba CP-3: Cálculo de KPIs (Promedio y Estado)

ID del Caso	CP-3
Nombre	Cálculo correcto de KPIs: Cálculo correcto de KPIs: CUM, Asistencia y Estado Académico
Objetivo	Verificar que la Cloud Function calcula correctamente los Indicadores Clave de Desempeño (KPIs) definidos en el Diccionario de KPIs v1.1, específicamente: • Cálculo del promedio_CUM. • Cálculo del porcentaje de asistencia. • Determinación del estado académico (Activo, En Riesgo, Retirado) en función de ambas variables.
Precondiciones	• Datos académicos leídos y validados correctamente (CP-2 aprobado). • Diccionario de KPIs oficial implementado en el código. • Fórmula de promedio_CUM:

	promedio_CUM=(Nota1+Nota2+Nota3)/3promedio_CUM = (Nota1 + Nota2 + Nota3) / 3promedio_CUM=(Nota1+Nota2+Nota3)/3 • Fórmula de asistencia: asistencia=(Sesiones asistidas/Total sesiones)∗100asistencia = (Sesiones\ asistidas / Total\ sesiones) * 100asistencia=(Sesiones asistidas/Total sesiones)∗100 **Condición** — **Estado** CUM ≥ 7.0 y Asistencia ≥ 80% — Activo CUM < 7.0 y Asistencia ≥ 60% — En Riesgo Asistencia < 60% — Retirado
Datos de Prueba	Estudiante A: Notas 8.0, 7.5, 9.0 → Promedio esperado: 8.17 → Estado: Activo Estudiante B: Notas 5.5, 4.0, 6.0 → Promedio esperado: 5.17 → Estado: Riesgo Estudiante C: Notas 6.0, 6.0, 6.0 → Promedio esperado: 6.00 → Estado: Activo
Pasos	1. Procesar los datos de prueba de los tres estudiantes 2. Calcular el promedio para cada estudiante 3. Aplicar la regla de estado académico según el promedio 4. Almacenar o mostrar los KPIs calculados 5. Validar que los cálculos coincidan con los valores esperados

Resultado Esperado	• Estudiante A: Promedio = 8.17, Estado = Activo • Estudiante B: Promedio = 5.17, Estado = Riesgo • Estudiante C: Promedio = 6.00, Estado = Activo • KPIs registrados correctamente en logs o estructura de datos
Prioridad	Alta

4. Caso de Prueba CP-4: Manejo de datos inválidos o nulos

ID del Caso	CP-4
Nombre	Gestión de errores para datos inválidos, nulos o con formato incorrecto
Objetivo	Verificar que la función detecta y maneja adecuadamente registros con datos inválidos, nulos o formatos erróneos, registrando los errores en la Hoja de Logs sin interrumpir el procesamiento de registros válidos.
Precondiciones	• Conexión activa a Google Sheets • Hoja de Logs configurada para registro de errores

	• Validaciones del Contrato de Datos implementadas
Datos de Prueba	Registro 1: Nota1 = texto "ABC" (formato inválido) Registro 2: Correo = NULL (valor nulo en campo requerido) Registro 3: Nombre vacío, Carné = NULL Registro 4: Nota2 = -5 (valor fuera de rango 0-10)
Pasos	1. Ejecutar la función con los datos de prueba erróneos 2. Detectar cada tipo de error durante la validación 3. Registrar cada error en la Hoja de Logs con fecha y hora, registro afectado y descripción del error 4. Continuar procesando los registros válidos restantes 5. Retornar resumen de registros procesados vs registros con errores
Resultado Esperado	• Los 4 registros inválidos son detectados y NO procesados • Cada error registrado en Hoja de Logs con detalle del problema

	• Código HTTP 200 con resumen: Ejemplo: 10 exitosos, 4 errores • Los registros válidos se procesan normalmente • No se interrumpe la ejecución completa de la función
Prioridad	Alta

5. Caso de Prueba CP-5: Seguridad y control de acceso mediante Service Account

ID del Caso	CP-5
Nombre	Verificación de seguridad
Objetivo	Validar que la Service Account agente-integrador-dac solo tiene los permisos mínimos necesarios y que sin credenciales válidas la función no puede acceder a Google Sheets, garantizando la seguridad del sistema.
Precondiciones	• Service Account creada con roles: Editor, Cloud Functions Admin, Service Account User • Hoja de Google Sheets compartida SOLO con el correo de la Service Account

	• Archivo service_account.json disponible
Pasos	1. Verificar en IAM que la Service Account solo tiene los 3 roles asignados 2. Intentar ejecutar la función SIN el archivo service_account.json 3. Verificar que la función retorna error de autenticación 4. Restaurar las credenciales correctas 5. Verificar que solo la Service Account puede acceder a la hoja (no usuarios personales sin permisos)
Resultado Esperado	• Service Account tiene exactamente los 3 roles definidos (ni más ni menos) • Sin credenciales: Error de autenticación (401 o 403) • Con credenciales: Acceso exitoso a Google Sheets • Usuarios no autorizados no pueden acceder a la hoja • Logs muestran que la identidad de ejecución es agente-integrador-dac
Prioridad	Media

6. Caso de Prueba CP-6: Invocación mediante Trigger HTTPS

ID del Caso	CP-6
Nombre	Verificación de Trigger HTTPS y respuesta del endpoint público
Objetivo	Validar que la Cloud Function responde correctamente a invocaciones externas mediante su URL HTTPS pública, confirmando que el Trigger está configurado adecuadamente y el endpoint es accesible.
Precondiciones	• Cloud Run Function desplegada con configuración correspondiente. • URL HTTPS pública generada automáticamente • Código Python main.py desplegado correctamente
Pasos	1. Obtener la URL HTTPS de la función desde la consola de Google Cloud 2. Realizar una petición GET o POST al endpoint usando curl, Postman o navegador 3. Verificar el código de respuesta HTTP 4. Validar el contenido de la respuesta

	5. Revisar los logs de ejecución en Cloud Logging
Resultado Esperado	• Código HTTP 200 OK en la respuesta • Respuesta con contenido válido (puede ser mensaje de éxito o datos procesados) • Tiempo de respuesta menor a 60 segundos • Logs muestran la invocación y ejecución exitosa • El endpoint es accesible desde cualquier navegador o cliente HTTP
Prioridad	Alta

Conclusión

La finalización de la Fase 2 consolida exitosamente la implementación de un flujo serverless completamente funcional, automatizado y alineado con los principios de eficiencia, seguridad y escalabilidad en Google Cloud. A través del desarrollo de funciones base, el procesamiento estructurado de datos de prueba y la generación de reportes académicos, se logró validar la viabilidad técnica del sistema desde la ingesta de información hasta la entrega del resultado final.

La integración de triggers programados y mecanismos de ejecución segura permitió eliminar la dependencia de procesos manuales, estableciendo un modelo operativo confiable y repetible. Asimismo, el uso de buenas prácticas como la definición de contratos de datos, la aplicación del principio de mínimo privilegio y la incorporación de monitoreo continuo garantizan la integridad, trazabilidad y control del sistema.

En conjunto, esta fase sienta las bases para la evolución hacia un entorno de analítica institucional más avanzado, permitiendo la incorporación de nuevas fuentes de datos, el escalamiento del procesamiento y la generación de reportes más complejos, posicionando a la solución como un componente clave para la toma de decisiones académicas basada en datos.