Renato Benjamín Ignacio Hernández Loyola
DOCUMENTO TÉCNICO
Integración de Google Sheets API como Base de Datos

---

1. CARACTERÍSTICAS Y MODO DE USO DE GOOGLE SHEETS API

1.1 Características Principales

Google Sheets API es un servicio RESTful que permite leer, escribir y formatear datos en hojas de cálculo de Google. Las características más relevantes son:

- **Acceso programático**: Permite manipular hojas de cálculo sin intervención manual del usu ario.
- **Operaciones CRUD**: Soporta creación, lectura, actualización y eliminación de datos.
- **Autenticación OAuth 2.0**: Garantiza seguridad mediante tokens de acceso.
- **Límites de cuota**: 300 solicitudes por minuto por proyecto (pueden ajustarse).
- **Formato estructurado**: Los datos se organizan en filas y columnas, ideal para bases de datos simples.

1.2 Preparación de Hoja de Cálculo como Base de Datos

Para utilizar una hoja de Google Sheets como base de datos, se deben seguir estos pasos:

Paso 1: Crear y estructurar la hoja de cálculo
- Crear una nueva hoja en Google Sheets.
- Definir la primera fila como encabezados de columna (ejemplo: ID, Usuario, Contraseña, Email, Fecha_Registro).
- Establecer formato consistente en cada columna (texto, número, fecha).
- Asignar permisos de acceso adecuados (compartir con la cuenta de servicio).

Paso 2: Habilitar Google Sheets API
- Acceder a Google Cloud Console (console.cloud.google.com).
- Crear un nuevo proyecto o seleccionar uno existente.
- Navegar a "API y servicios" > "Biblioteca".
- Buscar "Google Sheets API" y habilitarla.

Paso 3: Crear credenciales de cuenta de servicio
- En "API y servicios" > "Credenciales", hacer clic en "Crear credenciales".
- Seleccionar "Cuenta de servicio".
- Completar el formulario con nombre y descripción.
- Descargar el archivo JSON con las credenciales.
- Copiar el email de la cuenta de servicio.

Paso 4: Compartir la hoja de cálculo
- Abrir la hoja de Google Sheets.
- Hacer clic en "Compartir".

- Pegar el email de la cuenta de servicio.
- Asignar permisos de "Editor".

Paso 5: Obtener ID de la hoja
- El ID se encuentra en la URL de la hoja:
`https://docs.google.com/spreadsheets/d/[ID_AQUI]/edit`
- Copiar este ID para utilizarlo en el código.

1.3 Identificación de Líneas de Código para Implementación

Instalación de bibliotecas necesarias (Python):
```python
pip install google-auth google-auth-oauthlib google-auth-httplib2 google-api-python-client
```

Código de autenticación y conexión:
```python
from google.oauth2.service_account import Credentials
from googleapiclient.discovery import build

# Definir alcances de permisos
SCOPES = ['https://www.googleapis.com/auth/spreadsheets']

# Cargar credenciales desde archivo JSON
credentials = Credentials.from_service_account_file(
    'ruta/al/archivo/credenciales.json',
    scopes=SCOPES
)

# Crear servicio de Google Sheets
service = build('sheets', 'v4', credentials=credentials)

# ID de la hoja de cálculo
SPREADSHEET_ID = 'tu_spreadsheet_id_aqui'
```

Código para leer datos:
```python
# Leer rango específico
range_name = 'Hoja1!A1:E100'
result = service.spreadsheets().values().get(
    spreadsheetId=SPREADSHEET_ID,
    range=range_name
).execute()
values = result.get('values', [])
```

Código para escribir datos:

```python
# Escribir datos en la hoja
values = [
    ['ID', 'Usuario', 'Contraseña', 'Email', 'Fecha'],
    ['1', 'usuario1', 'pass123', 'user@mail.com', '2025-10-22']
]
body = {'values': values}
result = service.spreadsheets().values().update(
    spreadsheetId=SPREADSHEET_ID,
    range='Hoja1!A1',
    valueInputOption='RAW',
    body=body
).execute()
```

Código para agregar datos (append):
```python
# Agregar nueva fila al final
values = [['2', 'usuario2', 'pass456', 'user2@mail.com', '2025-10-22']]
body = {'values': values}
result = service.spreadsheets().values().append(
    spreadsheetId=SPREADSHEET_ID,
    range='Hoja1!A1',
    valueInputOption='RAW',
    body=body
).execute()
```

---

2. ANÁLISIS Y DEPURACIÓN DEL CÓDIGO "LOGEO Y REGISTRO 4"

2.1 Análisis del Código Original

Sin acceso al código específico proporcionado por el profesor, se presenta un análisis general de un sistema típico de logeo y registro:

Componentes esperados:
- Función de registro de usuarios
- Función de validación de credenciales
- Almacenamiento de datos de usuario
- Interfaz de usuario (GUI o CLI)

2.2 Procedimiento de Integración para Creación de Nuevos Usuarios

Función propuesta para agregar usuarios a Google Sheets:

```python
```

```
import hashlib
from datetime import datetime

def crear_nuevo_usuario(service, spreadsheet_id, usuario, contrasena, email):
    """
    Crea un nuevo usuario en la hoja de Google Sheets

    Args:
        service: Servicio de Google Sheets API
        spreadsheet_id: ID de la hoja de cálculo
        usuario: Nombre de usuario
        contrasena: Contraseña del usuario
        email: Correo electrónico del usuario

    Returns:
        bool: True si el registro fue exitoso, False en caso contrario
    """
    try:
        # 1. Verificar si el usuario ya existe
        range_usuarios = 'Usuarios!B:B'
        result = service.spreadsheets().values().get(
            spreadsheetId=spreadsheet_id,
            range=range_usuarios
        ).execute()
        usuarios_existentes = result.get('values', [])

        # Buscar usuario en la lista (omitir encabezado)
        for fila in usuarios_existentes[1:]:
            if fila and fila[0] == usuario:
                print(f"Error: El usuario '{usuario}' ya existe")
                return False

        # 2. Generar hash de contraseña (seguridad)
        contrasena_hash = hashlib.sha256(contrasena.encode()).hexdigest()

        # 3. Obtener ID único (último ID + 1)
        range_ids = 'Usuarios!A:A'
        result = service.spreadsheets().values().get(
            spreadsheetId=spreadsheet_id,
            range=range_ids
        ).execute()
        ids_existentes = result.get('values', [])

        if len(ids_existentes) > 1:
            ultimo_id = int(ids_existentes[-1][0])
            nuevo_id = ultimo_id + 1
        else:
            nuevo_id = 1
```

```
        # 4. Preparar datos del nuevo usuario
        fecha_registro = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
        nuevo_usuario = [[
            nuevo_id,
            usuario,
            contrasena_hash,
            email,
            fecha_registro
        ]]

        # 5. Agregar usuario a la hoja
        body = {'values': nuevo_usuario}
        service.spreadsheets().values().append(
            spreadsheetId=spreadsheet_id,
            range='Usuarios!A:E',
            valueInputOption='RAW',
            body=body
        ).execute()

        print(f"Usuario '{usuario}' registrado exitosamente con ID {nuevo_id}")
        return True

    except Exception as e:
        print(f"Error al crear usuario: {str(e)}")
        return False


def validar_credenciales(service, spreadsheet_id, usuario, contrasena):
    """
    Valida las credenciales de un usuario contra la base de datos

    Args:
        service: Servicio de Google Sheets API
        spreadsheet_id: ID de la hoja de cálculo
        usuario: Nombre de usuario
        contrasena: Contraseña a validar

    Returns:
        bool: True si las credenciales son válidas, False en caso contrario
    """
    try:
        # Obtener todos los usuarios
        range_datos = 'Usuarios!A:E'
        result = service.spreadsheets().values().get(
            spreadsheetId=spreadsheet_id,
            range=range_datos
        ).execute()
```

```
        usuarios = result.get('values', [])

        # Hash de la contraseña ingresada
        contrasena_hash = hashlib.sha256(contrasena.encode()).hexdigest()

        # Buscar usuario y validar contraseña (omitir encabezado)
        for fila in usuarios[1:]:
            if len(fila) >= 3 and fila[1] == usuario:
                if fila[2] == contrasena_hash:
                    print(f"Login exitoso para usuario '{usuario}'")
                    return True
                else:
                    print("Contraseña incorrecta")
                    return False

        print(f"Usuario '{usuario}' no encontrado")
        return False

    except Exception as e:
        print(f"Error al validar credenciales: {str(e)}")
        return False
```

2.3 Pasos de Depuración en Visual Studio Code

1. **Configurar entorno de depuración:**
   - Crear archivo `.vscode/launch.json`
   - Configurar Python debugger

2. **Establecer puntos de interrupción:**
   - En líneas críticas de autenticación
   - En funciones de lectura/escritura de Google Sheets
   - En validaciones de entrada de usuario

3. **Verificar variables:**
   - Inspeccionar valores de credenciales
   - Revisar respuestas de API
   - Validar formato de datos

4. **Pruebas unitarias:**
   - Crear casos de prueba para registro exitoso
   - Probar registro con usuario duplicado
   - Validar login con credenciales correctas e incorrectas

---

3. CHECK LIST PARA IMPLEMENTACIÓN PILOTO

3.1 Fase de Preparación

Configuración del Entorno:
- [ ] Crear proyecto en Google Cloud Console
- [ ] Habilitar Google Sheets API
- [ ] Generar credenciales de cuenta de servicio
- [ ] Descargar archivo JSON de credenciales
- [ ] Instalar bibliotecas necesarias (google-api-python-client, etc.)
- [ ] Configurar variables de entorno para credenciales

Preparación de Base de Datos:
- [ ] Crear hoja de Google Sheets
- [ ] Definir estructura de columnas (ID, Usuario, Contraseña, Email, Fecha)
- [ ] Agregar encabezados en primera fila
- [ ] Compartir hoja con cuenta de servicio
- [ ] Verificar permisos de edición
- [ ] Documentar ID de la hoja de cálculo

3.2 Fase de Desarrollo

Desarrollo del Código:
- [ ] Implementar función de autenticación con Google Sheets API
- [ ] Desarrollar función de registro de usuarios
- [ ] Desarrollar función de validación de credenciales
- [ ] Implementar hash de contraseñas (seguridad)
- [ ] Crear validaciones de entrada de datos
- [ ] Implementar manejo de errores y excepciones
- [ ] Agregar logging para seguimiento de operaciones

Pruebas de Funcionalidad:
- [ ] Probar conexión exitosa con Google Sheets
- [ ] Verificar registro de nuevo usuario
- [ ] Probar registro con usuario duplicado
- [ ] Validar login con credenciales correctas
- [ ] Validar login con credenciales incorrectas
- [ ] Probar manejo de errores de red
- [ ] Verificar límites de tasa de API

3.3 Fase de Implementación Piloto

Preparación del Entorno Piloto:
- [ ] Configurar servidor o máquina de prueba
- [ ] Instalar dependencias en entorno piloto
- [ ] Transferir credenciales de forma segura
- [ ] Configurar respaldos de la hoja de cálculo
- [ ] Establecer protocolo de rollback

Recolección de Información:

- [ ] Definir métricas de éxito (tasa de registro, tiempo de respuesta, etc.)
- [ ] Configurar sistema de logs
- [ ] Preparar formularios de retroalimentación de usuarios
- [ ] Establecer canales de comunicación para reportar problemas

Monitoreo y Seguimiento:
- [ ] Documentar horario de inicio del piloto
- [ ] Registrar número de usuarios participantes
- [ ] Monitorear consumo de cuota de API
- [ ] Registrar incidentes y errores
- [ ] Medir tiempos de respuesta de operaciones
- [ ] Recopilar feedback de usuarios piloto

3.4 Fase de Evaluación

Análisis de Resultados:
- [ ] Revisar logs de errores y excepciones
- [ ] Analizar métricas de rendimiento
- [ ] Evaluar experiencia de usuario
- [ ] Identificar cuellos de botella
- [ ] Documentar problemas encontrados
- [ ] Proponer mejoras y optimizaciones

Documentación Final:
- [ ] Elaborar informe técnico con resultados
- [ ] Documentar lecciones aprendidas
- [ ] Actualizar manual de usuario
- [ ] Crear guía de solución de problemas
- [ ] Definir plan de acción para producción

3.5 Información Requerida para Informe Técnico

Datos Técnicos:
- Versión de bibliotecas utilizadas
- Configuración del servidor/entorno
- Especificaciones de hardware
- Estructura de la base de datos
- Diagrama de flujo del sistema

Métricas Operacionales:
- Número total de usuarios registrados durante piloto
- Número de intentos de login (exitosos y fallidos)
- Tiempo promedio de registro de usuario
- Tiempo promedio de validación de credenciales
- Número de errores de API registrados
- Porcentaje de disponibilidad del sistema

Feedback Cualitativo:

- Comentarios de usuarios piloto
- Problemas reportados
- Sugerencias de mejora
- Nivel de satisfacción general

Análisis de Seguridad:
- Validación de hash de contraseñas
- Revisión de permisos de acceso
- Auditoría de logs de acceso
- Identificación de vulnerabilidades potenciales

CONCLUSIONES

La integración de Google Sheets API como base de datos para un sistema de logeo y registro ofrece una solución práctica y accesible para proyectos de pequeña y mediana escala. Las ventajas incluyen facilidad de configuración, acceso programático robusto y capacidad de visualización directa de datos.

Sin embargo, es importante considerar las limitaciones en términos de escalabilidad, cuotas de API y latencia, especialmente para aplicaciones con alto volumen de usuarios concurrentes. Para un piloto controlado, esta solución es viable y permite validar la funcionalidad antes de migrar a una base de datos tradicional si es necesario.

El checklist propuesto garantiza una implementación ordenada y permite recopilar información relevante para la toma de decisiones en la fase de producción.

Documento elaborado en: Arial, 12pt
Fecha: 22 de octubre de 2025

—----------