# Generación archivo JSON proyecto GSL

**Función:** Genera un archivo JSON con datos asociados al run de personas pertenecientes al registro social de hogares(RSH) dispuesto por MIDESO, y que cruzan con el run de vacunas programáticas y egresos hospitalarios. Incluye los datos de riesgo asociados a EH, alertas por falta de vacunas e información asociada a la inscripción en Fonasa.

El proceso se compone de la ejecución de distintos SP que pueblan diversas tablas para finalmente ser utilizadas por un script Python que genera el archivo respectivo en el SFTP de MIDESO.

**Monitoreo:** Se debe supervisar que al finalizar el proceso se encuentre cargado en la carpeta SFTP de MIDESO el archivo AAAA-MM-DD\_MinsalData.json.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

**Requisito:** Requiere que se ejecuten los siguientes procedimientos almacenados desde la BD EgresosHospitalarios y S0\_RNI , y en el orden que se indica:

|  |  |
| --- | --- |
| Orden | NOMBRE SP |
| 1 | **S0\_RNI.dbo.LOAD\_DATAMART\_GSL\_PersonasEstadoVac** |
| 2 | **EgresosHospitalarios .dbo.Sp\_Carga\_DATAMART\_TBL\_GSL\_MultiEH** |
| 3 | **EgresosHospitalarios .dbo.Sp\_Carga\_DATAMART\_TBL\_EH\_Riesgo** |
| 4 | **EgresosHospitalarios .dbo.Sp\_Carga\_DATAMART\_TBL\_GSL\_PersonasEhRiesgo** |
| 5 | **EgresosHospitalarios .dbo.Sp\_Carga\_DATAMART\_TBL\_GSL\_PersonasEstadoVac** |
| 6 | **EgresosHospitalarios .dbo.Sp\_Carga\_DATAMART\_TBL\_GSL\_RunInscrip** |

Los procedimientos dejarán cargados los datos en las tablas:

|  |
| --- |
| * S0\_RNI.dbo.DATAMART\_GSL\_PersonasEstadoVac * Egresoshospitalarios.dbo.DATAMART\_TBL\_GSL\_MultiEH * Egresoshospitalarios.dbo.DATAMART\_TBL\_GSL\_EH\_Riesgo * Egresoshospitalarios.dbo.TBL\_GSL\_PersonasEhRiesgo |
| * Egresoshospitalarios.dbo.TBL\_GSL\_PersonasEstadoVac |
| * Egresoshospitalarios.dbo.TBL\_GSL\_RunInscrip |
|  |

Los sp y tablas involucradas en el proceso se encuentran descritas en el archivo **FlujoDeDatosFuentesDEIS\_v3.xlsx** de la biblioteca de documentos de Alfresco.

Posteriormente, se debe ejecutar el script de Json **EHInscripMideso\_Json\_v4.json**, adjunto en el Anexo 2 de este documento.

**Ubicación de los SP:** Los procedimientos almacenados se encuentran en la BD del servidor 172.16.1.218.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

**Salida:** Al finalizar el proceso correctamente debe haberse generado el documento json con el formato de la imagen siguiente:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

**Reproceso:** En caso de requerir reprocesar, se debe ejecutar el sp correspondiente al proceso que falló y los siguientes sp que continúan a continuación de ese. Posteriormente, ejecutar nuevamente el script Json.

# Credenciales utilizadas en los procesos:

SFTP:

Archivos MIDESO

Host: 10.64.170.133 Puerto: 22

Usuario : patricio.aguilera\_MIDESO Contraseña : 4RaEI4UcCU

# Anexo 1: Archivo ini\_EH.txt

username=[***completar***]

password=[***completar***]

server=172.16.1.218,5656

database=egresoshospitalarios

SFTP\_Password=4RaEI4UcCU

SFTP\_User=patricio.aguilera\_MIDESO

SFTP\_Server=10.64.170.133

SFTP\_Port=22

# Anexo 2: Archivo EHInscripMideso\_Json\_v4.json

import pyodbc

import json

import paramiko

import datetime

# Lee datos conexion

with open('D:/Doc\_JSON/Config/ini\_EH.txt', 'r') as archivo:

    for linea in archivo:

        variable, valor = linea.strip().split('=')

        globals()[variable]=valor

# Conecta a SQL SERVER

conn = pyodbc.connect('driver={%s};server=%s;database=%s;uid=%s;pwd=%s' % ('SQL Server', server, database, username, password ) )

cursor = conn.cursor()

cursor2 = conn.cursor()

current\_datetime = datetime.datetime.now()

formatted\_datetime = current\_datetime.strftime("%Y-%m-%d %H:%M:%S")

fecha\_ejecucion=current\_datetime.strftime("%Y-%m-%d")

print("Comienza la creación de archivo Json GSL. ", formatted\_datetime)

# Ejecutar la consulta SQL Principal

consulta\_1 = f"SELECT R.RUN, R.DV, \

    R.NombreCentroAtencion as Establecimiento, \

    R.NombreComuna as Comuna, \

    R.NombreRegion as Region, \

    R.FechaInscripcion as FechaInscripcion, \

    Riesgo\_EH as NivelDeRiesgo, \

    DescripcionRiesgoEH as Descripcion, \

    R.Origen as Origen \

  FROM [dbo].[TBL\_GSL\_RunInscrip] R WITH(INDEX(IDX\_RUN\_TBL\_GSL\_RunInscrip)) \

  LEFT JOIN [egresoshospitalarios].[dbo].[TBL\_GSL\_PersonasEhRiesgo] EH WITH(INDEX(IDX\_RUN\_TBL\_GSL\_PersonasEhRiesgo)) on EH.RUN=R.RUN \

  ORDER BY R.RUN"

cursor.execute(consulta\_1)

# Obtener los resultados de la consulta

results = cursor.fetchall()

# Crear una lista para almacenar los resultados

data = []

contador=0

# Recorrer los resultados y agregarlos al diccionario

for row in results:

    contador=contador + 1

    diccionario = {

        'RUN': row.RUN,

        'DV': row.DV,

        'Inscripcion': [],

        'RiesgoPorHospitalizacion': [],

        'Inmunizacion': [],

    }

    # Los datos anidados vienen de la misma consulta SQL principal

    nested\_item1 = {

            'Establecimiento': row.Establecimiento,

            'Comuna': row.Comuna,

            'Region': row.Region,

            'FechaInscripcion':row.FechaInscripcion

        }

    nested\_item2 = {

            'NivelDeRiesgo': row.NivelDeRiesgo,

            'Descripcion': row.Descripcion,

        }

    diccionario['Inscripcion'].append(nested\_item1)

    diccionario['RiesgoPorHospitalizacion'].append(nested\_item2)

    if row.Origen=="V":

        # Ejecutar una consulta SQL para obtener los elementos de la lista de vacunas

        cursor2.execute('SELECT [RUN],[ALERTA] \

        FROM [EgresosHospitalarios].[dbo].[TBL\_GSL\_PersonasEstadoVac]  WITH(INDEX(IDX\_RUN\_ALERTA\_TBL\_GSL\_PersonasEstadoVac)) \

        WHERE ALERTA IS NOT NULL AND RUN= ?', row.RUN)

        nested\_results = cursor2.fetchall()

        # Recorrer los resultados de datos anidados y agregarlos al diccionario

        for nested\_row in nested\_results:

            nested\_item3 = {

                'Estado': 'Alerta',

                'Descripcion': nested\_row.ALERTA,

            }

            diccionario['Inmunizacion'].append(nested\_item3)

    # Agregar registros de riesgo al resultado principal

    data.append(diccionario)

# Cerrar el cursor y la conexión a la base de datos

cursor.close()

cursor2.close()

conn.close()

# Convertir los datos a formato JSON

json\_data = json.dumps(data, indent=4, ensure\_ascii=False,  separators=(", ", ": "))

nombre\_archivo=fecha\_ejecucion +'\_MinsalData.json'

# Ruta local donde se genera el archivo

ruta\_local\_archivo='D:/Doc\_JSON/'+ nombre\_archivo

# Guardar los datos JSON en un archivo

with open(ruta\_local\_archivo, 'w', encoding='utf-8') as file:

    file.write(json\_data)

print("El archivo JSON se ha creado localmente.")

# Traspasa archivo a SFTP

# Datos de conexión SFTP

hostname = SFTP\_Server

port = SFTP\_Port

username = SFTP\_User

password = SFTP\_Password

# Ruta remota donde se subirá el archivo

ruta\_remota = '/EntregasMINSAL/OtrasExtracciones/' + nombre\_archivo

# Crear una conexión SFTP

try:

    transport = paramiko.Transport(hostname, int(SFTP\_Port))

    transport.connect(username=username, password=password)

    sftp = paramiko.SFTPClient.from\_transport(transport)

    # Subir el archivo

    sftp.put(ruta\_local\_archivo, ruta\_remota)

    # Cerrar la conexión SFTP

    sftp.close()

    transport.close()

    print("Archivo subido a SFTP con éxito.")

    current\_datetime = datetime.datetime.now()

    formatted\_datetime = current\_datetime.strftime("%Y-%m-%d %H:%M:%S")

    print("Finalizada la creación y entrega SFTP de archivo Json GSL. ", formatted\_datetime)

except Exception as e:

    print("Error:", str(e))