Georesearch 2022

Descripción breve

Documento guía para integrar las fases del proyecto

Proyecto CANIBALIZACIÓN

Modelo conceptual y automatización

Contenido

[Proceso de automatización 2022 2](#_Toc117157381)

[Modelo conceptual 2](#_Toc117157382)

[Bases de datos 2](#_Toc117157383)

[Tablas postgres 2](#_Toc117157384)

[Dns configurados [ ] 3](#_Toc117157385)

[Endpoints 3](#_Toc117157386)

[Api keys [ ] 3](#_Toc117157387)

[Repositorio [ ] 3](#_Toc117157388)

[Diagrama de arquitectura API DMS 4](#_Toc117157389)

[~~Migración con step Functions~~ Se incluye en el servicio replicate-tables 4](#_Toc117157390)

[Detalle de la solicitud: 5](#_Toc117157391)

[Diagrama de la arquitectura api estudios [ ] 6](#_Toc117157392)

[Lambda 7](#_Toc117157393)

[Endpoint 7](#_Toc117157394)

[Salida de resultados 7](#_Toc117157395)

[Solicitud 7](#_Toc117157396)

[~~Versión anterior~~ 7](#_Toc117157397)

[Ultima versión 8](#_Toc117157398)

[Ejemplo 2 10](#_Toc117157399)

[Lista de etls disponibles 10](#_Toc117157400)

[Diccionario de claves: 11](#_Toc117157401)

[customer\_list (requerido) [{},{}…{}] 11](#_Toc117157402)

[solicitud 11](#_Toc117157403)

[Filtro de regiones provincias y comunas 11](#_Toc117157404)

[Subclave parámetros 12](#_Toc117157405)

[~~Futuros cambios~~ ya incluidos 12](#_Toc117157406)

[Flujo de actualización de datos 13](#_Toc117157407)

[Paso 1 Migración 13](#_Toc117157408)

[Paso 2 Creación de buffers 15](#_Toc117157409)

[Paso 3 Replicación 16](#_Toc117157410)

[Postman 17](#_Toc117157411)

[Agregar API KEY 18](#_Toc117157412)

[Actualizaciones de de la api para inputs a medida 18](#_Toc117157413)

# Proceso de optimización/automatización 2022

Para este año se pretende mejorar el performance del reporte con algunas optimizaciones del código fuente y cambios de la arquitectura de los datos, de esta manera reducir los actuales tiempos de entrega que pueden ser de hasta 4 días dependiendo de la magnitud de los datos.

En este documento se mencionará particularmente el proceso de creación de la tabla de precálculo para el reporte de canibalización “local" que es un proceso extenso y costoso, además de incluir las etapas estándares de un ETL por medio de la generación de tablas en el datalake, extracción y procesamiento de datos, desarrollo de etls, desarrollo de apis y despliegue de información que tendrán que ver específicamente con la fase de creación del precálculo del reporte de canibalización.

# Modelo conceptual [modelo dominio]

El origen del reporte de canibalización nace en la plataforma Mastergeo, desde ahí su objetivo es responder a la pregunta: ¿En cuánto porcentaje afectan las ventas de una tienda o tiendas comerciales con respecto a la colocación de un nuevo local comercial en punto geográfico?

En respuesta a la pregunta se entregan 2 soluciones: la primera responde al comportamiento respecto a locales comerciales y la segunda sobre la modalidad “Delivery” en donde el área de impacto o canibalización es dinámica.

## Equipos de trabajo

Para dar comienzo a una solicitud del cliente, tendremos una previa coordinación de los equipos internos de analytics y desarrollo y datos, que se encargan de estimar y entregar algunos factores claves de acuerdo con la magnitud del reporte y sus datos.

En grandes rasgos tenemos las siguientes etapas previas que se deberán coordinar:

1. Recepción del requerimiento del cliente (solicitud de canibalización, ¿área de estudios?).
2. Coordinación del equipo datos y estudios (Estas áreas se preparan para la generación de precálculos necesarios para el reporte).
3. Coordinación del equipo de desarrollo para la integración del nuevo reporte/modulo a la plataforma Mastergeo.
4. Fase de validación de datos y o QA entre las áreas para dar paso a la entrega final en producción.
5. Liberación a producción.

## Área de datos

El objetivo inicial posterior a la llegada de la solicitud, es crear una tabla denominada “Precalculo” para dar origen a 2 nuevas tablas más livianas que puedan ser consultadas por el proceso general del reporte que está desarrollado en lambda Python y consulta directamente el datawarehouse , actualmente el cuello de botella se presenta acá pues esta tabla se crea en la base de datos y es demoroso por lo que se migrara a Amazon para que pueda ser procesada en el motor analítico Athena.

### Tabla de Precálculo

Esta tabla tiene 2 funciones, una es entregar el gasto acumulado por local comercial y el segundo es entregar una variable denominada num (numerador) a partir de 2 factores que son entregados por el área de Analytics, que se calcula con la siguiente formula:

Num = POWER(distancia del local al block , factor de distancia ) \* POWER(superficie del local , factor de superficie )

Nota: La distancia y superficie se expresan en metros y mts2, y los factores de distancia se entregan aparte por Analytics.

Para la confirmación de esta tabla también se utiliza una consulta previa (**gastos**) para obtener un factor de la suma de todos los gastos de los blocks, en este caso se indican **filtros** específicos (**canasta\_categoria\_id**) de acuerdo con el rubro del cliente/reporte.

Posteriormente se consulta por una **sub selección de comercios** respectivos que **intersecan** con la tabla de **gastos** y que estén **presentes en una área** de búsqueda que también es definida por el cliente, en este caso se define un buffer de n metros que usualmente borda entre los 2 km y 10 km , y como máximo 40km, aquí **es lo mas costoso** del calculo pues esto se genera dinámicamente.

(pois\_comercios\_servicios\_view cross join gastos , where category\_id , sales\_area > 0 y gasto > 0 and block in área de búsqueda respecto al local)

### Posibles mejoras

Para disminuir aún mas los tiempos de entrega seria necesario identificar cuanto tiempo toma realizar este calculo para evaluar su integración.

1. Automatización del calculo de los factores de distancia y superficie

### Mejoras realizadas

Actualmente a partir de esta tabla de precálculo se originarán 2 nuevas que optimizan el lambda que entrega los datos finales, la siguiente convención se usara para nombrarlas donde en rojo es el nombre definido a la solicitud, en azul el schema/pais.

1. customer\_**little\_caesar\_pizza**\_**mx**.precalculo\_**lcp**\_blocks
2. customer\_**little\_caesar\_pizza**\_**mx**.precalculo\_**lcp**\_locales

# Bases de datos

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Motor** | **Ambiente** | **Ip** | **Usuario** | **db** |
| PostgreSQL | PROD | 10.10.0.40 | postgres | mastergeo\_countries |
| PostgreSQL | QA | 10.10.0.52 | postgres | mastergeo\_countries |
| Athena | QA |  |  | qa\_countries |
| Athena | PROD |  |  | prod\_countries |
| Athena | QA |  |  | qa\_canibalizacion |
| Athena | PROD |  |  | prod\_ canibalizacion |

## Tablas postgres

|  |  |
| --- | --- |
| **Schema** | **Tabla** |
| country\_<pais> | blocks |
| country\_<pais> | pois\_comercios\_servicios |
| country\_<pais> | view\_blocks |
| country\_<pais> | pois |
| country\_<pais> | view\_blocks |
| country\_<pais> | view\_gastos |
| country\_<pais> | view\_blocks |
| country\_<pais> | categories |
| country\_<pais> | income\_levels |

# Dns configurados [ ]

* <https://canibalizacion.geo-research.com/>
* https://canibalizacion -dev.geo-research.com/
* https://canibalizacion -qa.geo-research.com/

# Endpoints

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| utilidad | api | name | order | endpoint | verb |
| Status stepfunctions |  | Creada por Francisco Mendoza para consultar genéricamente estos estados |  | https://5xvveqna8d.execute-api.us-east-1.amazonaws.com/dev/state | POST |

# Api keys [ ]

|  |  |
| --- | --- |
| **api key** | **clave** |
| canibalizacion-dev | HmtXfTvefC7QDpsSHQvvn2LFk4Yd19456mJc9Lei |
| canibalizacion -qa | bEfporJGeq88yRJy4vGcd3WdQzjwWzRaBEoYzzR1 |
| canibalizacion -prod | Fw60Wah3W15GtaF6yf7ok9XfoHWJT4642cDbbEUp |

# Repositorio [ ]

1. https://github.com/cvargas-geo/canibalizacin

# 

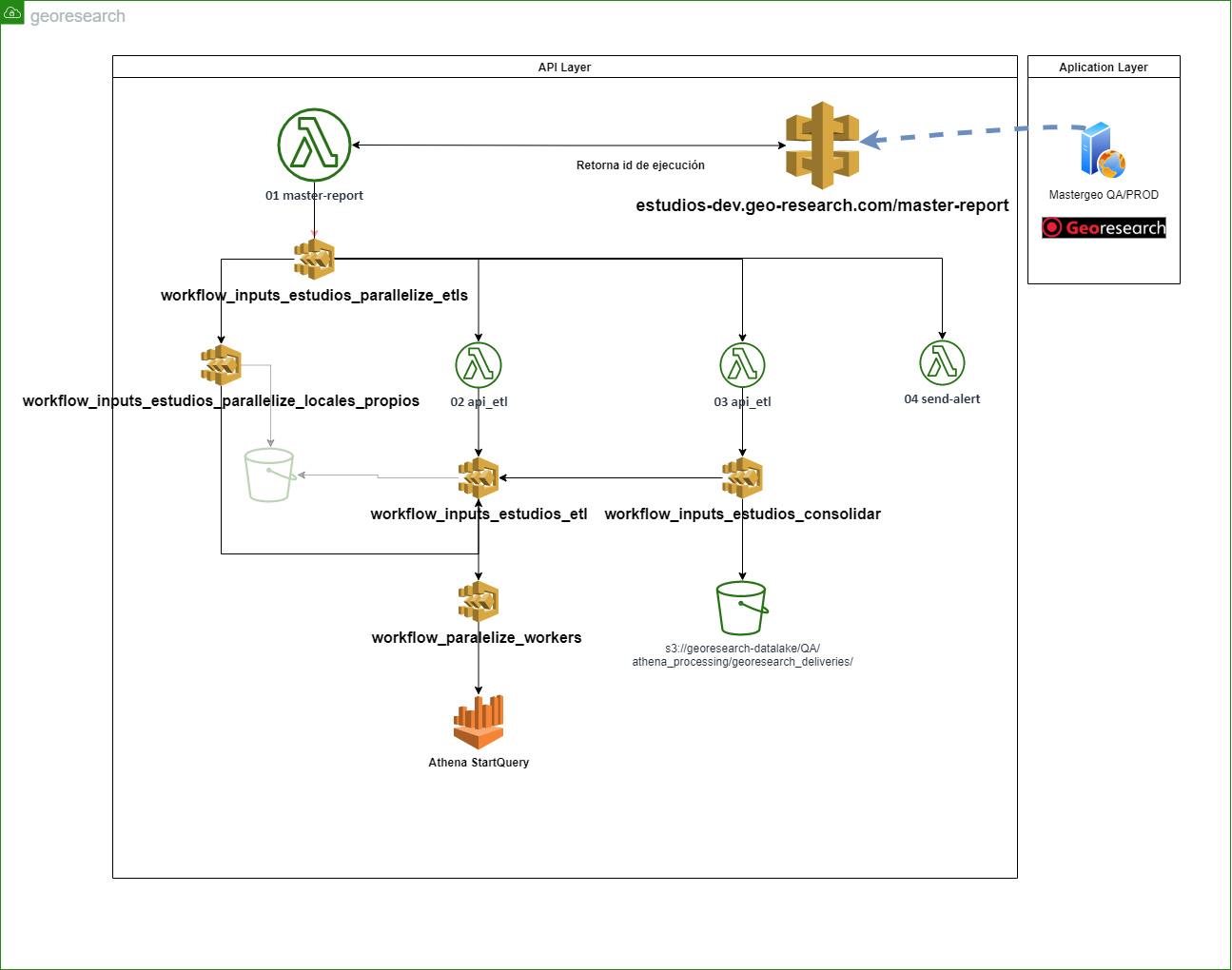
# Inputs del servicio

Casa solicitud de reporte debe proporcionar los siguientes valores para generar la tabla de precálculo inicial

* País de origen
* Coordenada inicial
* Polígono de busqueda
* Lista de canastas gastos
* Lista de categorias
* ¿Lista de subcategorías?
* Factor de superficie
* Factor de distancia
* Radio de búsqueda en metros (precalculo)

# Diagrama de la arquitectura api estudios [ ]

Toda la infraestructura planteada fue migrada desde un proceso en Airflow – gcp hacia AWS utilizando Lambda, Athena, S3, Step Functions y ApiGateway.



## Lambda

Actualmente las siguientes Lambdas están desplegadas en la cuenta productiva AWS, su acceso se debe solicitar al JP respectivo.

1. etl-estudios-2022-dev-master-report

## Endpoint

Para hacer consultas se habilito la siguiente url que puede ser consultada con Postman (www.postman.com) para hacer pruebas.

* POST - https://estudios-dev.geo-research.com/master-report

## Salida de resultados

Una vez terminado el proceso iniciado por el servicio master-report este creara los archivos final en la siguiente ubicación de S3 , en este caso **new\_customer** corresponde al nombre asignado para la solicitud de ejemplo.

Ejemplo para ambiente de QA

s3://georesearch-datalake/QA/athena\_processing/georesearch\_deliveries/**new\_customer\_inputs**

Ejemplo para ambiente de PROD

s3://georesearch-datalake/PROD/athena\_processing/georesearch\_deliveries/**new\_customer\_inputs**

## Solicitud

Para realizar pruebas utilizar el siguiente input, en rojo parámetros obligatorios y en azul opcionales.

### Versión anterior

{

"environment": "PROD",

"report\_name": "cani\_west",

"schema": "mx",

"report\_to": [

"cvargas@georesearch.cl"

],

"drop\_workflow": false,

"buffer\_search": 20000,

"pois\_state\_id": 1,

"surface\_factor": -1.95,

"distance\_factor": 1.1,

"start\_point": "POINT(X,X)",

"cannibalization\_shape": "POLYGON((X,X))",

"canasta\_categoria\_id": [

34

],

"substring\_id\_o\_subcadena": [

4,

5,

6

],

"pois\_category\_id": [

10008

],

"etl\_list": [

"local",

"delivery"

],

"local\_cannibalization\_reports": [

{

"drop\_workflow": false,

"buffer\_search": 2000,

"pois\_state\_id": 1,

"surface\_factor": -1.95,

"distance\_factor": 1.1,

"start\_point": "POINT(X,X)",

"cannibalization\_shape": "POLYGON((X,X))",

"canasta\_categoria\_id": [

34

],

"substring\_id\_o\_subcadena": [

4,

5,

6

],

"pois\_category\_id": [

10008

]

}

],

"delivery\_cannibalization\_reports": [

{

"drop\_workflow": false,

"canasta\_categoria\_id": [

34

],

"substring\_id\_o\_subcadena": [

4,

5,

6

],

"start\_point": "POINT(X,X)",

"cannibalization\_shape": "POLYGON((X,X))"

}

]

}

# Lista de etls disponibles

La salida de resultados contempla la siguiente información del negocio:

1. local
2. deliveryDiccionario de claves:

Definiciones previas

Objeto: {}

Lista: []

## customer\_list (requerido) [{},{}…{}]

Lista de solicitudes que se compone de una o más solicitudes.

Ejemplo:

{“customer\_list”:[{solicitud\_1}…{ solicitud\_2}]}

## solicitud

Objeto que contiene los destalles enviados en el Excel, va incorporado dentro de su clave padre:“customer\_list”

Parámetros de la solicitud, en rojo campos requeridos celeste opcionales:

* **customer\_name**: requerido, string, nombre de la solicitud que se asocia al cliente.
* **enviroment**:requerido,string indica el entorno de trabajo de los datos.
* **country\_list**: requerido, lista de países en formato ISO, contiene 1 o más. Ejemplo: [“cl”…”mx”] , actualmente solo permite los siguientes códigos isos: cl,mx,co,pe.
* **buffer\_list**: requerido, lista de números que contiene los buffers requeridos en metros, contiene 1 o más, ejemplo [1000,2000,3500], actualmente solo permite los siguientes: [500,600,1000,1500,2000,2500,3000,3500,4000,4500,5000]
* **id\_gastos**: requerido, lista de números que contiene los ids de los substring requeridos para los gastos, puede contener 1 o más, ejemplo [2,26].
* **etl\_list**: requerido, lista de strings que contiene los nombres de los etl a solicitar, puede contener 3 o más ejemplo [“demografico”,”gastos”,”competencias”].
* **parametros**: requerido, objeto que contiene los requerimientos de cada etl, se detalla más abajo. Ejemplo

{

        "competencias": {},

"locales\_propios": {}

}

* **drop\_workflow**: opcional, boleano, indicador para borrar los objetos internamente creados de una solicitud. Ejemplo true o false, (no acepta mayúsculas).
* **report\_to**: opcional, lista de correos a los que serán enviada una notificación cuando el proceso finalice ejemplo: [“customer@georesearch.com”]

### Filtro de regiones provincias y comunas

Este filtro por ahora contempla una sola regla la cual debe ser especificada en los 3 niveles para evitar colisiones. Ejemplo: para filtrar solo datos demográficos de la comuna de MAIPU de deben especificar además la provincia de SANTIAGO y la región Metropolitana. Esto solo aplica para la subclave demográfico.

Captura de pantalla de computadora

Descripción generada automáticamente

### Subclave parámetros

Esta clave está destinada a especificar al detalle cada parámetro necesario del etl.

Para identificarlas se agregan según el nombre etl a que hacen referencia.

Por ejemplo, en el caso de las competencias y locales propios se debe especificar la lista de sub strings y su estado.

Como locales propios es un etl opcional, la subclave dentro de parámetros también lo es, pero en el caso de competencias no, pues es un etl obligatorio.

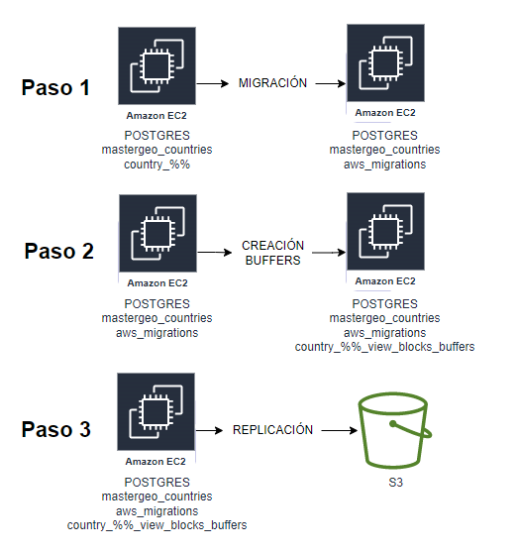
## ~~Futuros cambios~~ ya incluidos

Estos serán implementados para mejorar la actual solicitud de inputs.

1. Se incluirá la clave “country” dentro de “parametros” para especificar parámetros por países, esto también aplicará para id\_gastos que será movido dentro de parámetros, Esto se explica pues lo id gastos de un país pueden ser diferentes en otro.

# Flujo de actualización de datos

A continuación, se mencionan los pasos principales para actualizar las tablas en Athena para la generación de los inputs estudios.



1. Migración: Consiste en generar la solicitud para un conjunto de tablas/vistas y migrarlas al schema aws\_migrations, en este proceso asíncrono se convierten las columnas geometry a texto y las vistas a tablas por lo que se debe verificar en la bbdd los objetos creados. (postgres-postgres)
2. Creación de buffers: Generar la solicitud síncrona para la creación de los buffers para un conjunto de tablas, tiende a tardar menos de 15 min. (postgres-postgres)
3. Replicación: Consiste en tomar los objetos de aws\_migrations y llevarlos a S3 mediante una solicitud asíncrona, esta solicitud posee un conjunto de reglas de replicación que permitirán al servicio DMS migrar las tablas. ( postgres - aws ) sf : workflow\_dev\_dms\_replicacion

## Paso 1 Migración

Si por ejemplo tenemos una actualización para chile o cualquier otro país(mx,co,pe,ar,uy), las siguientes tablas serán necesarias para realizar el proceso:

1. country\_cl.pois\_comercios\_servicios\_view
2. country\_cl.view\_blocks
3. country\_cl.empresas
4. country\_cl.view\_gastos
5. country\_cl.categories
6. country\_cl.canasta\_categoria
7. country\_cl.canastas\_total
8. country\_cl.income\_levels

Con esto en mente se deberá preparar el request para la api, con el objetivo de mover las tablas al esquema de migración “**aws\_migrations**“, luego desde aquí otro servicio se encargara de copiar los datos a S3.

**Endpoint**: https://estudios-qa.geo-research.com/migrate-tables

A continuación, el siguiente request json será utilizado para completar el primer paso. La clave environment podrá ser PROD o QA.

{

"body": {

"environment": "PROD",

"tables": [

{

"schema": "country\_cl",

"table": "pois\_comercios\_servicios\_view"

},

{

"schema": "country\_cl",

"table": "view\_blocks"

},

{

"schema": "country\_cl",

"table": "empresas"

},

{

"schema": "country\_cl",

"table": "view\_gastos"

},

{

"schema": "country\_cl",

"table": "categories"

},

{

"schema": "country\_cl",

"table": "canasta\_categoria"

},

{

"schema": "country\_cl",

"table": "canastas\_total"

},

{

"schema": "country\_cl",

"table": "income\_levels"

}

]

}

}

Nota: Para abarcar más de un país, el siguiente script en Python ayudara a generar el json anterior, solo es necesario incluir los schemas necesarios mientras estén homologados. (MX,CL,PE,CO)(UR y AR en proceso)

Consulta para verificar el estado de las tablas

/\*PARA USAR EN DBEAVER: consulta por los registros que faltan por copiar \*/

WITH

SOURCE\_TABLES AS (

SELECT COUNT(\*) AS RECUENTO , 'country\_cl.pois\_comercios\_servicios\_view' AS TABLA FROM country\_cl.pois\_comercios\_servicios\_view UNION ALL SELECT COUNT(\*) AS RECUENTO , 'country\_cl.view\_blocks' AS TABLA FROM country\_cl.view\_blocks UNION ALL SELECT COUNT(\*) AS RECUENTO , 'country\_cl.empresas' AS TABLA FROM country\_cl.empresas UNION ALL SELECT COUNT(\*) AS RECUENTO , 'country\_cl.view\_gastos' AS TABLA FROM country\_cl.view\_gastos UNION ALL SELECT COUNT(\*) AS RECUENTO , 'country\_cl.categories' AS TABLA FROM country\_cl.categories UNION ALL SELECT COUNT(\*) AS RECUENTO , 'country\_cl.canasta\_categoria' AS TABLA FROM country\_cl.canasta\_categoria UNION ALL SELECT COUNT(\*) AS RECUENTO , 'country\_cl.canastas\_total' AS TABLA FROM country\_cl.canastas\_total UNION ALL SELECT COUNT(\*) AS RECUENTO , 'country\_cl.income\_levels' AS TABLA FROM country\_cl.income\_levels

)

,TARGET\_TABLES AS (

SELECT COUNT(\*) AS RECUENTO , 'country\_cl.pois\_comercios\_servicios\_view' AS TABLA FROM aws\_migrations.country\_cl\_pois\_comercios\_servicios\_view UNION ALL SELECT COUNT(\*) AS RECUENTO , 'country\_cl.view\_blocks' AS TABLA FROM aws\_migrations.country\_cl\_view\_blocks UNION ALL SELECT COUNT(\*) AS RECUENTO , 'country\_cl.empresas' AS TABLA FROM aws\_migrations.country\_cl\_empresas UNION ALL SELECT COUNT(\*) AS RECUENTO , 'country\_cl.view\_gastos' AS TABLA FROM aws\_migrations.country\_cl\_view\_gastos UNION ALL SELECT COUNT(\*) AS RECUENTO , 'country\_cl.categories' AS TABLA FROM aws\_migrations.country\_cl\_categories UNION ALL SELECT COUNT(\*) AS RECUENTO , 'country\_cl.canasta\_categoria' AS TABLA FROM aws\_migrations.country\_cl\_canasta\_categoria UNION ALL SELECT COUNT(\*) AS RECUENTO , 'country\_cl.canastas\_total' AS TABLA FROM aws\_migrations.country\_cl\_canastas\_total UNION ALL SELECT COUNT(\*) AS RECUENTO , 'country\_cl.income\_levels' AS TABLA FROM aws\_migrations.country\_cl\_income\_levels

)

SELECT

A.TABLA ,

CASE WHEN A.RECUENTO - B.RECUENTO >= 0 THEN A.RECUENTO - B.RECUENTO

ELSE -1 END AS POR\_COPIAR

FROM SOURCE\_TABLES A

LEFT JOIN TARGET\_TABLES B ON A.TABLA = B.TABLA

ORDER BY 2 DESC

Script de generación

import json

environment = 'PROD'

selected\_countries = [

'country\_cl' ,

# 'country\_mx' ,

# 'country\_co' ,

# 'country\_pe' ,

# 'country\_ar',

# 'country\_uy'

]

tables = [

"pois\_comercios\_servicios\_view",

"view\_blocks",

"empresas",

"view\_gastos",

"categories",

"canasta\_categoria",

"canastas\_total",

"income\_levels"

]

limit =''

migracion\_tablas = {

"environment": "PROD",

"tables": [

{

"schema": country,

"table": table,

} for table in tables for country in selected\_countries

]

}

request = {"body":migracion\_tablas}

print(json.dumps( request , indent=4))

Para ejecutar el código se puede ejecutar en los siguientes enlaces:

* <https://jupyter.org/try-jupyter/lab/>
* <https://replit.com/languages/python3>

## Paso 2 Creación de buffers

Dado que es los buffers se cran a partir de las tablas view\_blocks o pois , también será necesario actualizarlas por lo que se debe preparar la siguiente solicitud .

Endpoint: https://estudios-qa.geo-research.com/buffer

{

"body": {

"environment": "PROD",

"schemas": [

"country\_cl"

],

"tables": [

"view\_blocks"

],

"buffers": [

100,

500,

600,

800,

1000,

1500,

2000,

2500,

3000,

3500,

4000,

4500,

5000

]

}

}

Consulta de validación de datos:

/\*PARA USAR EN DBEAVER : consulta por los registros que faltan por copiar \*/

WITH

SOURCE\_TABLES AS (

SELECT COUNT(\*) AS RECUENTO , 'country\_cl.view\_blocks' AS TABLA FROM country\_cl.view\_blocks

)

,TARGET\_TABLES AS (

SELECT COUNT(\*) AS RECUENTO , 'country\_cl.view\_blocks' AS TABLA FROM aws\_migrations.country\_cl\_view\_blocks\_buffers

)

SELECT

A.TABLA ,

CASE WHEN A.RECUENTO - B.RECUENTO >= 0 THEN A.RECUENTO - B.RECUENTO

ELSE -1 END AS POR\_COPIAR

FROM SOURCE\_TABLES A

LEFT JOIN TARGET\_TABLES B ON A.TABLA = B.TABLA

ORDER BY 2 DESC

Consulta para agregar una columna buffer de ayuda

ALTER TABLE aws\_migrations.country\_mx\_view\_blocks\_buffers

ADD buffer\_5000 text NULL ;

UPDATE aws\_migrations.country\_mx\_view\_blocks\_buffers A

SET

buffer\_5000 = st\_astext(st\_buffer(st\_setsrid(st\_point(B.longitud, B.latitud), 4326)::geography, 5000)::geometry)

FROM country\_mx.view\_blocks B

WHERE A.id = B.id;

Consulta para crear la tabla de buffers

DROP TABLE IF EXISTS aws\_migrations.country\_mx\_view\_blocks\_buffers ;

CREATE TABLE aws\_migrations.country\_mx\_view\_blocks\_buffers AS

SELECT

id ,

block\_id,

latitud,

longitud,

administrative\_area\_level\_1,

administrative\_area\_level\_2

FROM country\_mx.view\_blocks ;

## Paso 3 Replicación

Los objetos de aws\_migrations son copiados a S3 y Athena, mientras se agreguen las reglas de mapeo necesarias en la solicitud.

**Endpoint**: https://estudios-qa.geo-research.com/replicate-tables

Nota: El servicio DMS permite el uso del wildcard “%” para realizar la búsqueda de los objetos.

{

"body": {

"environment": "PROD",

"rules": [

{

"schema-name": "aws\_migrations",

"table-name": "country\_cl\_% "

}

]

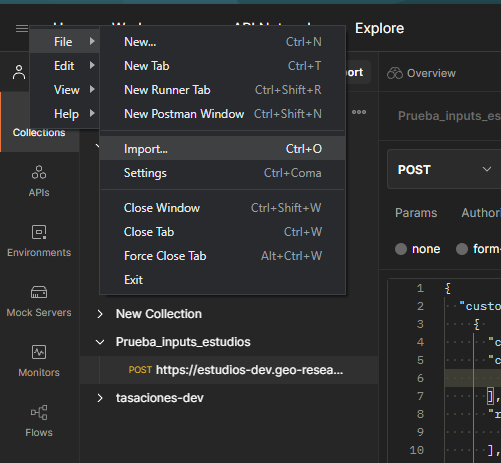
}

}

Para verificar el estado del proceso se deberán ver los logs de la stepfunction workflow\_dev\_dms\_replicacion

# Postman

Utilidades: https://jsoneditoronline.org/#/

1. Importar el json al programa File/Import
2. 
3. Interfaz de usuario gráfica, Aplicación

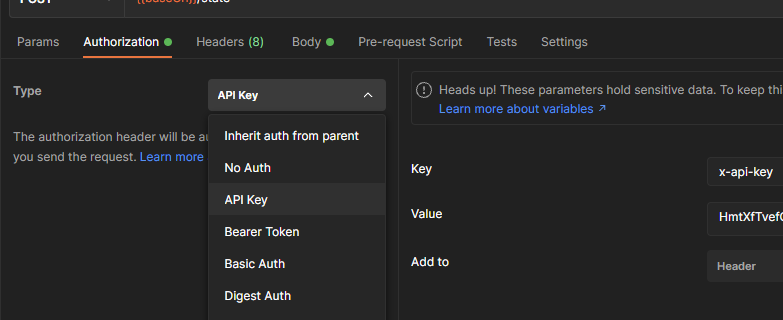
   Descripción generada automáticamente
4. Hay que asegurar que los headers estén activados.Interfaz de usuario gráfica, Aplicación

   Descripción generada automáticamente
5. Finalmente hacer una prueba, en la pestaña body + botón SEND
6. Interfaz de usuario gráfica, Texto

   Descripción generada automáticamente
7. Esto no indicara nada más, pero habilitare una opción para indicar fallos si la solicitud quedo malformada.

### Agregar API KEY

En este caso para incluirla en la request de postman agregarla en authorization / type API Key / y setar el campo value.



# Actualizaciones de de la api para inputs a medida

Dentro de los 2 casos de uso tenemos que cuando se requiere input con una modificación muy particular será necesario un desarrollo del flujo del etl , para ello solo será necesario crear una nueva consulta para el etl /stage particular que se necesite , pero deberá agregarse la solicitud dentro de siguiente archivo para que esta pueda ser reflejada en los resultados finales.

En este ejemplo se simula que la nueva solicitud cambia en todas sus consultas, lo que se va indicando dentro de la clave stages.

Texto

Descripción generada automáticamente