

GEOOs - ZRepo

Introducción

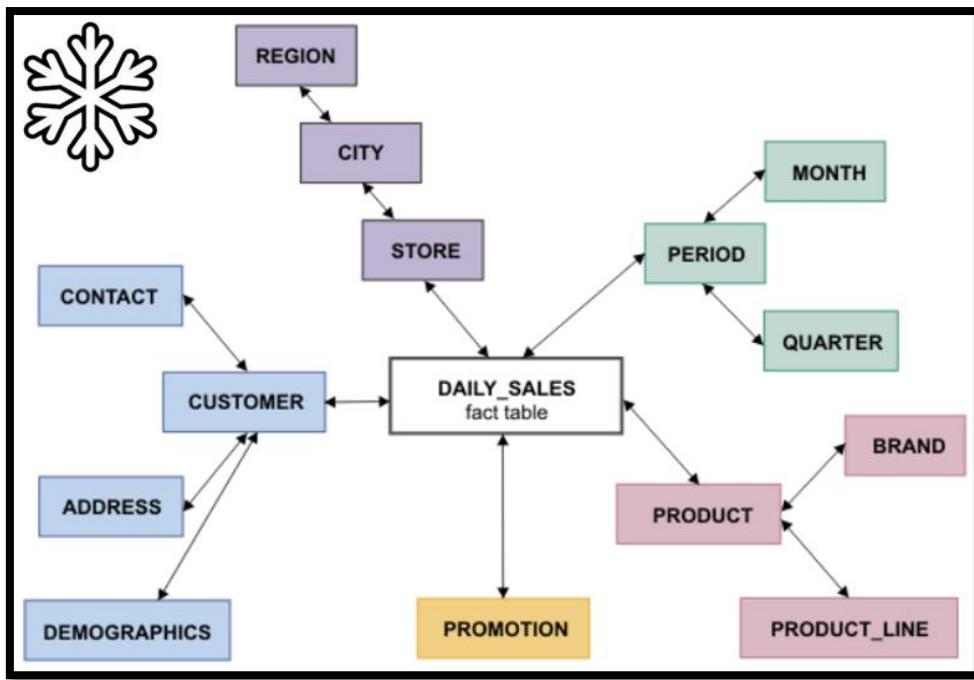
El sistema GEOOs se compone de tres módulos principales: Portal para la presentación e interacción con el usuario, GeoServer para la administración y consultas de capas georeferenciados y ZRepo para el manejo de **Cubos de Información**.

Debido a que GEOOs es un sistema para consulta, visualización y análisis de información georreferenciada, la información almacenada en los cubos de ZRepo debe tener algún tipo de asociación directa o indirecta a elementos que se puedan ubicar en el mapa. La selección de elementos en el mapa actúa como filtro de consulta a los datos agrupados en cubos de información. Al mismo tiempo, la información de los cubos es representada por el Portal dentro de los objetos vectoriales a los que están asociados, por ejemplo, rellenando de acuerdo con una escala de colores.

Al igual que el componente GeoServer, ZRepo ofrece una API REST que permite consultar la información de sus cubos y dimensiones de manera sencilla, facilitando así las integraciones desde otros sistemas, por ejemplo, para el análisis de información. A diferencia de GeoServer, con las credenciales de seguridad adecuadas, el API REST permite la acumulación (agregar) nueva información en los cubos.

Modelo de Copo de Nieve

El Data Warehousing corresponde al proceso de recopilación e integración de información desde varias fuentes, su almacenamiento en un repositorio común y mecanismos de consulta sobre esa información. El modelo de **Copo de Nieve** que implementa ZRepo formaliza una técnica de modelamiento y estructuración de los datos dentro de un data warehouse.



Según este esquema o modelo “Copo de Nieve”, se reconocen **hechos** que ocurren en el tiempo y que se clasifican por **dimensiones**, las que, a su vez, podrían estar clasificadas por otras **dimensiones**. Este tipo de modelos o estructura de representación de los datos se puede implementar como un motor de búsqueda y acumulación usando diferentes tecnologías y enfoques de diseño. ZRepo incluye una implementación de un modelo Copo de Nieve basado en un servidor de base de datos No-SQL (Mongo DB), una API REST para poblar y consultar y un lenguaje propio (basado en JSON) para las consultas.

Implementación ZRepo

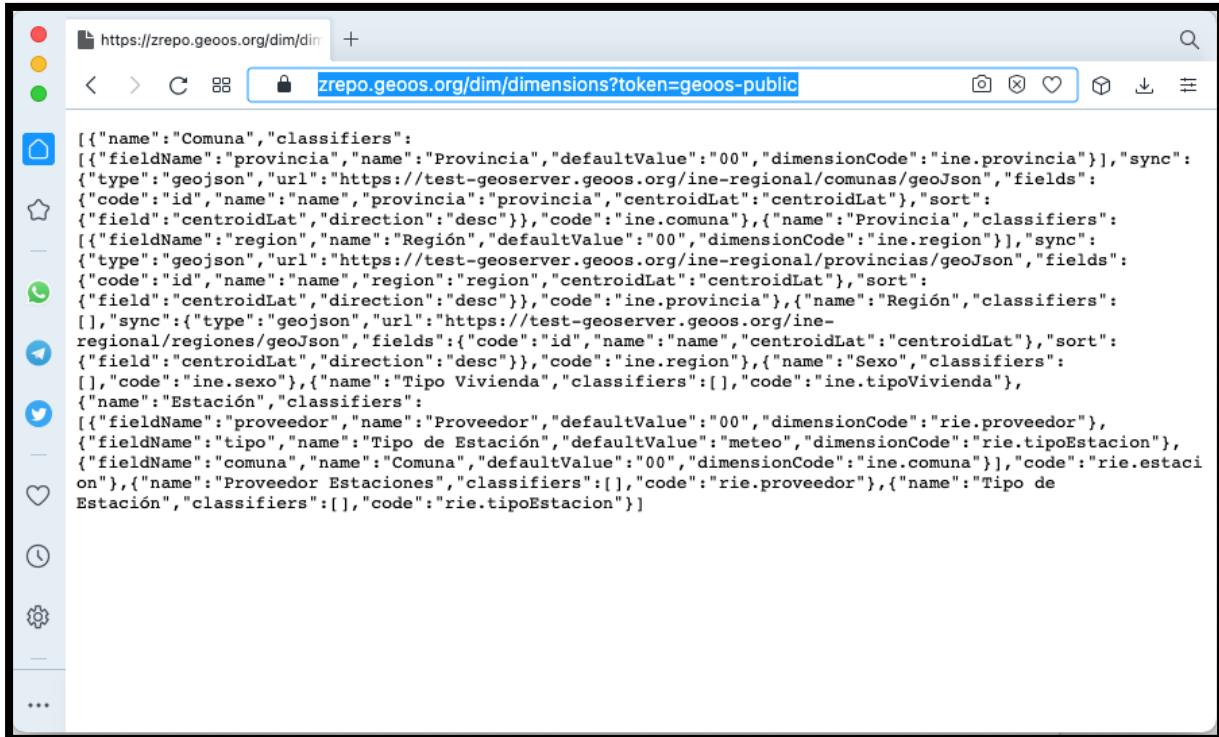
Los hechos en ZRepo corresponden a **Variables**, donde la dimensión temporal ya está considerada dentro de su definición. Las posibles temporalidades de una variable ZRepo son: 5, 15, 30 minutos; 1, 6, 12 horas; 1 día, 1, 3, 4, 6 meses y 1 año.

Una variable ZRepo se define por un código, un nombre, su temporalidad y un vector de dimensiones que la clasifican. Una dimensión es una tabla de datos con código y nombre en cada fila, cuyo contenido se mantiene relativamente constante en el tiempo, en comparación a las variables.

Al igual que el componente GeoServer, ZRepo requiere de un archivo “hjson” de configuración. En este archivo se declaran las dimensiones y las variables que se administran en el repositorio. La instalación en geoos.org ya tiene configuradas las dimensiones representativas de la división geopolítica de Chile; en particular, las regiones, provincias y comunas. Estas dimensiones están relacionadas dentro de una estructura jerárquica: **comuna → provincia → región**.

Utilizando la API REST de ZRepo, es posible consultar los datos y los metadatos del servidor. Por ejemplo, para obtener la lista de las dimensiones configuradas en el servidor, podemos hacer el siguiente request GET:

<https://zrepo.geoos.org/dim/dimensions?token=geoos-public>



Todas las invocaciones al API REST de ZRepo necesitan un token de autenticación, el que se configura en el mismo archivo base de configuración de ZRepo. De preferencia el token se debe enviar usando un header “**Authorization**”, comenzando con “**Bearer**” + token, según el estándar OAuth 2.

Por cada dimensión configurada en el servidor se retorna un objeto json, como, por ejemplo:

```
        "centroidLat": "centroidLat"
    },
    "sort": {
        "field": "centroidLat",
        "direction": "desc"
    }
},
"code": "ine.comuna"
}
```

Una dimensión se define por: un código, un nombre, sus dimensiones de clasificación y posiblemente información para su sincronización. El código de la dimensión se divide, a su vez, en: un dominio y un identificador final (separados por un punto). El dominio de una dimensión y una variable en GEOOs, debe corresponder a alguno de los proveedores definidos en GeoServer.

Para obtener el metadato de una dimensión usando el API REST, basta con usar la URL **/dim/codigoDimension** y el token de seguridad. Por ejemplo, la siguiente url, retorna el mismo objeto Json antes mostrado:

<https://zrepo.geoos.org/dim/ine.provincia?token=geoos-public>

Los datos o filas de una dimensión se pueden consultar usando el API REST mediante la url **/dim/codigoDimension/rows** y el token.

zrepo.geoos.org/dim/ine.provincia/rows?token=geoos-public	
<pre>[{"_id": "152", "centroidLat": -18.39444898872864, "code": "152", "name": "PARINACOTA", "order": 1, "region": "15"}, {"_id": "151", "centroidLat": -18.743936843070927, "code": "151", "name": "ARICA", "order": 2, "region": "15"}, {"_id": "14", "centroidLat": -19.984370289003703, "code": "14", "name": "TAMARUGAL", "order": 3, "region": "1"}, {"_id": "11", "centroidLat": -20.739792295546888, "code": "11", "name": "IQUIQUE", "order": 4, "region": "1"}, {"_id": "23", "centroidLat": -22.040287493181154, "code": "23", "name": "TOCOPILLA", "order": 5, "region": "2"}, {"_id": "22", "centroidLat": -22.718630914887544, "code": "22", "name": "EL LOA", "order": 6, "region": "2"}, {"_id": "21", "centroidLat": -24.440523319855238, "code": "21", "name": "ANTOFAGASTA", "order": 7, "region": "2"}, {"_id": "32", "centroidLat": -26.216636144838535, "code": "32", "name": "CHAÑARAL", "order": 8, "region": "3"}, {"_id": "52", "centroidLat": -27.052026466877276, "code": "52", "name": "ISLA DE PASCUA", "order": 9, "region": "5"}, {"_id": "31", "centroidLat": -27.417044730061086, "code": "31", "name": "COPIAPÓ", "order": 10, "region": "3"}, {"_id": "33", "centroidLat": -28.67102305509828, "code": "33", "name": "HUASCO", "order": 11, "region": "3"}, {"_id": "41", "centroidLat": -29.850248278257627, "code": "41", "name": "EL QUI", "order": 12, "region": "4"}, {"_id": "43", "centroidLat": -30.819090075796584, "code": "43", "name": "LIMARÍ", "order": 13, "region": "4"}, {"_id": "42", "centroidLat": -31.78544106001676, "code": "42", "name": "CHOAPA", "order": 14, "region": "4"}, {"_id": "54", "centroidLat": -32.36342622467841, "code": "54", "name": "PETORCA", "order": 15, "region": "5"}, {"_id": "57", "centroidLat": -32.60990228230332, "code": "57", "name": "SAN FELIPE DE ACONCAGUA", "order": 16, "region": "5"}, {"_id": "55", "centroidLat": -32.79927955067464, "code": "55", "name": "QUILLOTA", "order": 17, "region": "5"}, {"_id": "53", "centroidLat": -32.83716191206412, "code": "53", "name": "LOS ANDES", "order": 18, "region": "5"}, {"_id": "58", "centroidLat": -33.08152879451705, "code": "58", "name": "MARGA MARGA", "order": 19, "region": "5"}, {"_id": "133", "centroidLat": -33.13119335874739, "code": "133", "name": "CHACABUCO", "order": 20, "region": "13"}, {"_id": "51", "centroidLat": -33.24697181419046, "code": "51", "name": "VALPARAÍSO", "order": 21, "region": "5"}, {"_id": "131", "centroidLat": -33.36334075356036, "code": "131", "name": "SANTIAGO", "order": 22, "region": "13"}, {"_id": "56", "centroidLat": -33.562352039428454, "code": "56", "name": "SAN ANTONIO", "order": 23, "region": "5"}, {"_id": "136", "centroidLat": -33.66400435615692, "code": "136", "name": "TALAGANTE", "order": 24, "region": "13"}, {"_id": "132", "centroidLat": -33.70190234980547, "code": "132", "name": "CORDILLERA", "order": 25, "region": "13"}, {"_id": "135", "centroidLat": -33.740042061285386, "code": "135", "name": "MELIPILLA", "order": 26, "region": "13"}, {"_id": "134", "centroidLat": -33.81349800321045, "code": "134", "name": "MAIPO", "order": 27, "region": "13"}, {"_id": "61", "centroidLat": -34.26779322890312, "code": "61", "name": "CACHAPOAL", "order": 28, "region": "6"}, {"_id": "62", "centroidLat": -34.30915574396944, "code": "62", "name": "CARDENAL CARO", "order": 29, "region": "6"}, {"_id": "63", "centroidLat": -34.70505446541464, "code": "63", "name": "COLCHAGUA", "order": 30, "region": "6"}, {"_id": "73", "centroidLat": -35.11184649993691, "code": "73", "name": "CURICÓ", "order": 31, "region": "7"}, {"_id": "71", "centroidLat": -35.55366514808097, "code": "71", "name": "TALCA", "order": 32, "region": "7"}, {"_id": "72", "centroidLat": -35.93448870080114, "code": "72", "name": "CAUQUENES", "order": 33, "region": "7"}, {"_id": "74", "centroidLat": -36.09805596298197, "code": "74", "name": "LINARES", "order": 34, "region": "7"}, {"_id": "162", "centroidLat": -36.33119820836453, "code": "162", "name": "ITATA", "order": 35, "region": "16"}, {"_id": "163", "centroidLat": -36.53516331326021, "code": "163", "name": "PUNILLA", "order": 36, "region": "16"}, {"_id": "161", "centroidLat": -36.89713285157367, "code": "161", "name": "DIGUILLÍN", "order": 37, "region": "16"}, {"_id": "81", "centroidLat": -36.91985092458656, "code": "81", "name": "CONCEPCIÓN", "order": 38, "region": "8"}, {"_id": "83", "centroidLat": -37.519122955045, "code": "83", "name": "BIOBÍO", "order": 39, "region": "8"}]</pre>	

Como se observa en la imagen anterior, desde una fila de datos de una dimensión se refieren a los valores de las filas de otras dimensiones (a sus códigos). El nombre del atributo que contiene la referencia es el que se usó al configurar la dimensión, y que se mostró al consultar el metadato de la dimensión. En el caso de este ejemplo, el atributo “región” contiene el valor del código de la región (otra dimensión, como se indica en el metadato) asociada.

Al igual que con las dimensiones, para las variables existen servicios en el API REST para consultar su metadato y ejecutar las consultas a los cubos de información o variables. La lista completa de variables definidas en un servidor se puede obtener mediante la URL:

<https://zrepo.geoos.org/var/variables?token=geoos-public>

```

[{"name": "Casos Confirmados COVID", "temporality": "1d", "classifiers": [{"fieldName": "comuna", "name": "Comuna", "defaultValue": "00", "dimensionCode": "ine.comuna"}], "options": {"unit": "Nº", "provider": "cie", "decimals": 0, "defaults": {"accum": "sum"}, "code": "cie.covid_confirmados"}, {"name": "Densidad Poblacional", "temporality": "1y", "classifiers": [{"fieldName": "comuna", "name": "Comuna", "defaultValue": "00", "dimensionCode": "ine.comuna"}], "options": {"unit": "Hab/Km2", "provider": "ine", "subjects": ["senso"], "decimals": 2}, "code": "ine.densidad"}, {"name": "Dirección del Viento", "temporality": "5m", "classifiers": [{"fieldName": "estacion", "name": "Estación", "defaultValue": "00", "dimensionCode": "rie.estacion"}], "options": {"unit": "", "provider": "rie", "subjects": ["meteo"], "decimals": 2, "defaults": {"accum": "avg"}}, "code": "rie.dir_viento"}, {"name": "Electroconductividad del Suelo", "temporality": "5m", "classifiers": [{"fieldName": "estacion", "name": "Estación", "defaultValue": "00", "dimensionCode": "rie.estacion"}], "options": {"unit": "MS/cm", "provider": "rie", "subjects": ["meteo"], "decimals": 2, "defaults": {"accum": "avg"}}, "code": "rie.ec_suelo"}, {"name": "Humedad Relativa del Aire", "temporality": "5m", "classifiers": [{"fieldName": "estacion", "name": "Estación", "defaultValue": "00", "dimensionCode": "rie.estacion"}], "options": {"unit": "%", "provider": "rie", "subjects": ["meteo"], "decimals": 2, "defaults": {"accum": "avg"}}, "code": "rie.humedad"}, {"name": "Indice de Calor", "temporality": "5m", "classifiers": [{"fieldName": "estacion", "name": "Estación", "defaultValue": "00", "dimensionCode": "rie.estacion"}], "options": {"unit": "C", "provider": "rie", "subjects": ["meteo"], "decimals": 2, "defaults": {"accum": "avg"}}, "code": "rie.indice_calor"}, {"name": "Indice de Radicación UV", "temporality": "5m", "classifiers": [{"fieldName": "estacion", "name": "Estación", "defaultValue": "00", "dimensionCode": "rie.estacion"}], "options": {"unit": "UV", "provider": "rie", "subjects": ["meteo"], "decimals": 2, "defaults": {"accum": "avg"}}, "code": "rie.indice_uv"}, {"name": "Nivel del Mar - Sensor PRS", "temporality": "5m", "classifiers": [{"fieldName": "estacion", "name": "Mareógrafo", "defaultValue": "00", "dimensionCode": "rie.estacion"}], "options": {"unit": "m", "provider": "rie", "subjects": ["oce"], "decimals": 2, "defaults": {"accum": "avg"}}, "code": "rie.nivel_PRS"}, {"name": "Número Habitantes", "temporality": "ly", "classifiers": [{"fieldName": "comuna", "name": "Comuna", "defaultValue": "00", "dimensionCode": "ine.comuna"}, {"fieldName": "sexo", "name": "Sexo", "defaultValue": "I", "dimensionCode": "ine.sexo"}], "options": {"unit": "Nº", "provider": "ine", "subjects": ["senso"], "decimals": 0, "colorScale": {"name": "sst - NASA OceanColor", "auto": true, "clipOutOfRange": false}}, "code": "ine.poblacion"}]

```

Al igual que para las dimensiones, el metadato de sólo una variable se puede obtener con su código en la url del request:

<https://zrepo.geoos.org/var/ine.poblacion?token=geoos-public>

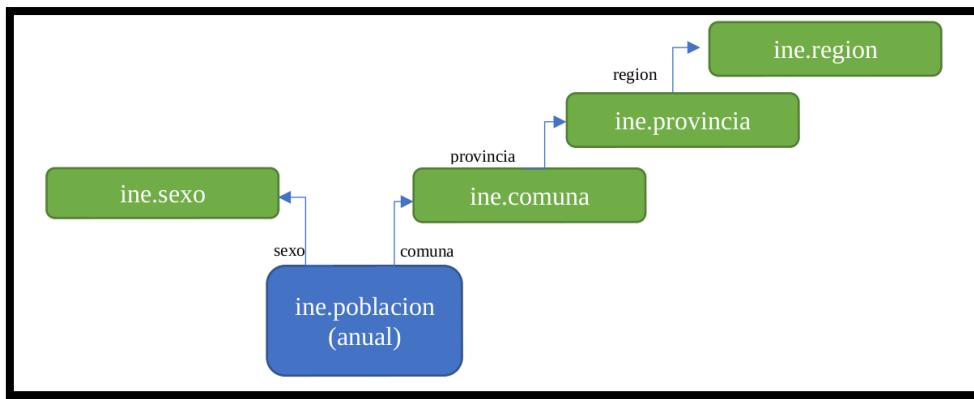
```

{"name": "Número Habitantes", "temporality": "ly", "classifiers": [{"fieldName": "comuna", "name": "Comuna", "defaultValue": "00", "dimensionCode": "ine.comuna"}, {"fieldName": "sexo", "name": "Sexo", "defaultValue": "I", "dimensionCode": "ine.sexo"}], "options": {"unit": "Nº", "provider": "ine", "subjects": ["senso"], "decimals": 0, "colorScale": {"name": "sst - NASA OceanColor", "auto": true, "clipOutOfRange": false}}, "code": "ine.poblacion"}

```

La variable del ejemplo anterior: “**ine.poblacion**” referencia a la dimensión “**ine.comuna**” a través del campo “**comuna**”, y a la dimensión “**ine.sexo**” mediante el campo “**sexo**”.

Usando el API de consulta de metadato de dimensiones, podemos determinar que la dimensión “**ine.comuna**” referencia a la dimensión “**ine.provincia**”, la que, a su vez, referencia a la dimensión “**ine.region**”. De la misma forma, obtenemos que la dimensión “**ine.sexo**” no referencia más dimensiones. De acuerdo con estas relaciones, podemos representar el modelo como el siguiente copo de nieve:



La variable **ine.poblacion** define un **Cubo** de información que acumulará valores de acuerdo con una clave dada por el siguiente vector: (año, sexo, comuna)

El API REST de ZRepo permite ejecutar consultas sobre los valores de las dimensiones y sobre las variables, usando la temporalidad y las dimensiones referenciadas como filtros.

Lenguaje de Consultas

Las consultas sobre los datos o filas de las dimensiones y las variables se realizan mediante el API REST que provee ZRepo. Ambos tipos de consulta tienen en común la forma de especificar los filtros, a partir de los **clasificadores** de una variable o dimensión. La forma más simple de consulta de filas de datos de una dimensión es mediante un request tipo GET a la URL **dim/codigoDimension/rows** usando el token de acceso a la información. Por ejemplo, para consultar las regiones de Chile, podemos usar:

<https://zrepo.geoos.org/dim/ine.region/rows?token=geoos-public>

```

[{"_id": "15", "centroidLat": -18.567386631806347, "code": "15", "name": "REGIÓN DE ARICA Y PARINACOTA", "order": 1}, {"_id": "1", "centroidLat": -20.230142141799863, "code": "1", "name": "REGIÓN DE TARAPACÁ", "order": 2}, {"_id": "2", "centroidLat": -23.79194129640924, "code": "2", "name": "REGIÓN DE ANTOFAGASTA", "order": 3}, {"_id": "3", "centroidLat": -27.600480947792473, "code": "3", "name": "REGIÓN DE ATACAMA", "order": 4}, {"_id": "4", "centroidLat": -30.850048857369462, "code": "4", "name": "REGIÓN DE COQUIMBO", "order": 5}, {"_id": "5", "centroidLat": -32.78008916957445, "code": "5", "name": "REGIÓN DE VALPARAÍSO", "order": 6}, {"_id": "13", "centroidLat": -33.65157178756864, "code": "13", "name": "REGIÓN METROPOLITANA DE SANTIAGO", "order": 7}, {"_id": "6", "centroidLat": -34.4137492065069, "code": "6", "name": "REGIÓN DEL LIBERTADOR GENERAL BERNARDO O'HIGGINS", "order": 8}, {"_id": "7", "centroidLat": -35.676578160948246, "code": "7", "name": "REGIÓN DEL MAULE", "order": 9}, {"_id": "16", "centroidLat": -36.5997289111397094, "code": "16", "name": "REGIÓN DEL NUBLE", "order": 10}, {"_id": "8", "centroidLat": -37.43359283300686, "code": "8", "name": "REGIÓN DEL BIOBÍO", "order": 11}, {"_id": "9", "centroidLat": -38.55373032014877, "code": "9", "name": "REGIÓN DE LA ARAUCANÍA", "order": 12}, {"_id": "14", "centroidLat": -39.887804600354144, "code": "14", "name": "REGIÓN DE LOS RÍOS", "order": 13}, {"_id": "10", "centroidLat": -42.37882869919311, "code": "10", "name": "REGIÓN DE LOS LAGOS", "order": 14}, {"_id": "11", "centroidLat": -46.04582449089817, "code": "11", "name": "REGIÓN DE AYSÉN DEL GENERAL CARLOS IBÁÑEZ DEL CAMPO", "order": 15}, {"_id": "12", "centroidLat": -52.84944271365398, "code": "12", "name": "REGIÓN DE MAGALLANES Y DE LA ANTÁRTICA CHILENA", "order": 16}]

```

Los únicos campos obligatorios para las filas de las dimensiones son **"code"** y **"name"**. Si la dimensión tiene clasificadores (otras dimensiones que clasifican cada fila) los códigos son retornados con el valor del campo clasificador. Por ejemplo, las provincias se clasifican por región.

```
[{"_id": "152", "centroidLat": -18.39444898872864, "code": "152", "name": "PARINACOTA", "order": 1, "region": "15"}, {"_id": "151", "centroidLat": -18.743936843070927, "code": "151", "name": "ARICA", "order": 2, "region": "15"}, {"_id": "14", "centroidLat": -19.984370289003703, "code": "14", "name": "TAMARUGAL", "order": 3, "region": "1"}, {"_id": "11", "centroidLat": -20.739792295546888, "code": "11", "name": "IQUIQUE", "order": 4, "region": "1"}, {"_id": "23", "centroidLat": -22.040287493181154, "code": "23", "name": "TOCOPILLA", "order": 5, "region": "2"}, {"_id": "22", "centroidLat": -22.718630914887544, "code": "22", "name": "EL LOA", "order": 6, "region": "2"}, {"_id": "21", "centroidLat": -24.440523319855238, "code": "21", "name": "ANTOFAGASTA", "order": 7, "region": "2"}, {"_id": "32", "centroidLat": -26.216636144838535, "code": "32", "name": "CHANARAL", "order": 8, "region": "3"}, {"_id": "52", "centroidLat": -27.052026466877276, "code": "52", "name": "ISLA DE PASCUA", "order": 9, "region": "5"}, {"_id": "31", "centroidLat": -27.417044730061086, "code": "31", "name": "COPIAPÓ", "order": 10, "region": "3"}, {"_id": "33", "centroidLat": -28.67102305509828, "code": "33", "name": "HUASCO", "order": 11, "region": "3"}, {"_id": "41", "centroidLat": -29.850248278257627, "code": "41", "name": "EL QUI", "order": 12, "region": "4"}, {"_id": "43", "centroidLat": -30.819090075796584, "code": "43", "name": "LIMARÍ", "order": 13, "region": "4"}, {"_id": "42", "centroidLat": -31.78544106001676, "code": "42", "name": "CHOAPA", "order": 14, "region": "4"}, {"_id": "54", "centroidLat": -32.36342622467841, "code": "54", "name": "PETORCA", "order": 15, "region": "5"}, {"_id": "57", "centroidLat": -32.60990228230332, "code": "57", "name": "SAN FELIPE DE ACONCAGUA", "order": 16, "region": "5"}, {"_id": "55", "centroidLat": -32.79927955067464, "code": "55", "name": "QUILLOTA", "order": 17, "region": "5"}, {"_id": "53", "centroidLat": -32.83716191206412, "code": "53", "name": "LOS ANDES", "order": 18, "region": "5"}, {"_id": "58", "centroidLat": -33.08152879451705, "code": "58", "name": "MARGA MARGA", "order": 19, "region": "5"}, {"_id": "133", "centroidLat": -33.13119335874739, "code": "133", "name": "CHACABUCO", "order": 20, "region": "13"}, {"_id": "51", "centroidLat": -33.24697181419046, "code": "51", "name": "VALPARAISO", "order": 21, "region": "5"}, {"_id": "131", "centroidLat": -33.36334075356036, "code": "131", "name": "SANTIAGO", "order": 22, "region": "13"}, {"_id": "56", "centroidLat": -33.562352039428454, "code": "56", "name": "SAN ANTONIO", "order": 23, "region": "5"}, {"_id": "136", "centroidLat": -33.66400435615692, "code": "136", "name": "TALAGANTE", "order": 24, "region": "13"}, {"_id": "132", "centroidLat": -33.70190234980547, "code": "132", "name": "CORDILLERA", "order": 25, "region": "13"}, {"_id": "135", "centroidLat": -33.740042061285386, "code": "135", "name": "MELIPILLA", "order": 26, "region": "13"}, {"_id": "134", "centroidLat": -33.81349800321045, "code": "134", "name": "MAIPO", "order": 27, "region": "13"}, {"_id": "61", "centroidLat": -34.26779322890312, "code": "61", "name": "CACHAPOAL", "order": 28, "region": "6"}, {"_id": "62", "centroidLat": -34.30915574396944, "code": "62", "name": "CARDENAL CARO", "order": 29, "region": "6"}, {"_id": "63", "centroidLat": -34.70505446541464, "code": "63", "name": "COLCHAGUA", "order": 30, "region": "6"}, {"_id": "73", "centroidLat": -35.1184649993691, "code": "73", "name": "CURICÓ", "order": 31, "region": "7"}, {"_id": "71", "centroidLat": -35.55366514808097, "code": "71", "name": "TALCA", "order": 32, "region": "7"}, {"_id": "72", "centroidLat": -35.93448870080114, "code": "72", "name": "CAUQUENES", "order": 33, "region": "7"}, {"_id": "74", "centroidLat": -36.09805596298197, "code": "74", "name": "LINARES", "order": 34, "region": "7"}, ]
```

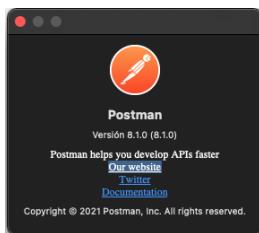
En el ejemplo anterior, el código de región es retornado como el campo “**region**” de cada fila, debido a que ese nombre de campo se usó en la declaración (archivo `zrepo.hjson`) de esa dimensión, dentro de “**classifiers**”, tal como lo muestra el metadato de la dimensión provincia:

```
{"name": "Provincia", "classifiers": [{"fieldName": "region", "name": "Region", "defaultValue": "00", "dimensionCode": "ine.region"}], "sync": {"type": "geojson", "url": "https://test-geoserver.geoos.org/ine-regional/provincias/geoJson", "fields": {"code": "id", "name": "name", "region": "region", "centroidLat": "centroidLat"}, "sort": {"field": "centroidLat", "direction": "desc"}}, "code": "ine.provincia"}
```

Una forma simple de filtrar las filas de una dimensión es por el contenido del campo “**name**”. Como parámetro de la consulta REST se puede usar “**textFilter**” con el texto que deseamos buscar dentro del campo “**name**” de cada fila.

```
[{"_id": "57", "centroidLat": -32.60990228230332, "code": "57", "name": "SAN FELIPE DE ACONCAGUA", "order": 16, "region": "5"}, {"_id": "131", "centroidLat": -33.36334075356036, "code": "131", "name": "SANTIAGO", "order": 22, "region": "13"}, {"_id": "56", "centroidLat": -33.562352039428454, "code": "56", "name": "SAN ANTONIO", "order": 23, "region": "5"}]
```

En el ejemplo anterior, se retornan todas las filas de la dimensión provincia que contengan el texto “**san**” como parte de su nombre.



Los siguientes ejemplos requieren de parámetros más complejos en la URL. En este manual se utiliza una herramienta llamada **Postman** para la ejecución de los request tipo GET y más adelante para los POST (agregar datos).

<https://www.postman.com/>

Además de la herramienta propuesta acá, es posible hacer estas pruebas desde el mismo browser o desde la consola usando comandos como **curl** o **wget**.

ZRepo soporta el manejo de tokens de seguridad como parte de la URL, como ya se usó en los ejemplos, o como un encabezado HTTP, de acuerdo con el estándar OAuth 2. Postman permite crear un conjunto de “**request**” como un grupo. Dentro de este grupo es posible configurar las credenciales de seguridad para todas las consultas (todos los requests) que se creen dentro de él.

The screenshot shows the Postman interface with the 'Authorization' tab selected for a collection named 'ZRepo - GEOOs REST API'. The 'Type' dropdown is set to 'OAuth 2.0'. A note at the bottom left of the authorization section cautions about sensitive data and recommends using variables. The 'Access Token' dropdown shows 'geoos-public' selected, and the 'Header Prefix' input field contains 'Bearer'.

Como se observa en la figura anterior, el token “**geoos-public**” se agregó en la sección Authorization del grupo “**ZRepo – GEOOs REST API**” de tipo “OAuth 2.0” con el prefijo por omisión “**Bearer**”.

Bajo el grupo, usando el botón derecho, agregamos un request de tipo “GET” con la URL básica de consulta a provincias y presionamos “Send”:

<https://zrepo.geoos.org/dim/ine.provincia/rows>

Usando la sección de parámetros, agregamos “**textFilter**” y “**san**” como valor. Esto es equivalente a haberlo agregado a la url, como los ejemplos anteriores. El resultado deberían ser las mismas 3 filas del ejemplo anterior.

- GET: <https://zrepo.geoos.org/dim/ine.provincia/rows>
- textFilter: "san"

The screenshot shows the Postman application interface. On the left, there's a sidebar with options like Home, Workspaces, Reports, Explore, Scratch Pad, Collections, APIs, Environments, Mock Servers, Monitors, and History. The main area is titled 'Working locally in Scratch Pad. Switch to a Workspace'. It shows a collection named 'ZRepo - GEOOs REST API' with a single item 'GET GET Provincias'. The request details show a GET method to 'https://zrepo.geoos.org/dim/ine.provincia/rows?textFilter=san'. Under 'Params', there is a single entry 'textFilter: san'. The 'Body' tab shows the response as a JSON array:

```

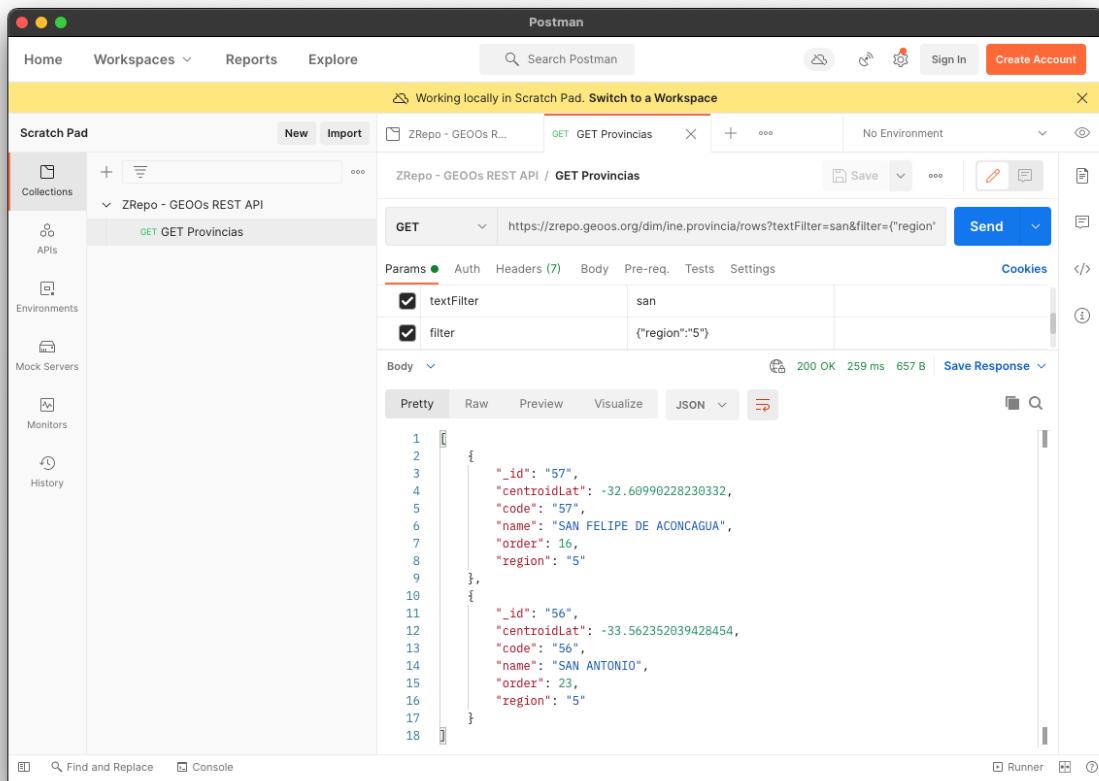
1  [
2   {
3     "_id": "57",
4     "centroidLat": -32.60990228230332,
5     "code": "57",
6     "name": "SAN FELIPE DE ACONCAGUA",
7     "order": 16,
8     "region": "5"
9   },
10  {
11    "_id": "131",
12    "centroidLat": -33.36334075356036,
13    "code": "131",
14    "name": "SANTIAGO",
15    "order": 22,
16    "region": "13"
17  },
18  {
19    "_id": "56",
20    "centroidLat": -33.562352039428454,
21    "code": "56",
22    "name": "SAN ANTONIO",
23  }

```

Sobre los clasificadores de una dimensión (y de una variable, como se verá más adelante) es posible aplicar filtros. Los filtros son objetos JSON que operan sobre los campos clasificadores. En el ejemplo anterior, las provincias se clasifican por “region” (el nombre de campo es “region” que referencia a la dimensión “**ine.region**” como se desplegó en el metadato).

El filtro más simple sería retornar sólo las filas dentro de una región. Para ello, agregamos el siguiente filtro como un parámetro más a la consulta GET anterior:

- GET: <https://zrepo.geoos.org/dim/ine.provincia/rows>
- textFilter: "san"
- filter: {"region": "5"}



En este caso, se retornan sólo las filas (provincias) cuyo campo “**region**” referencia a la fila de la dimensión “**ine.region**” cuyo código es “5”. El filtro recién usado es simple, de primer nivel, en donde directamente indicamos un valor de la dimensión asociada. La cantidad de campos con dimensiones asociadas puede ser mayor a uno (en este ejemplo tenemos sólo uno que es “**region**”) y se separan por comas “,” como otros atributos JSON normales.

El valor usado en el ejemplo anterior, para filtrar fue el código de la dimensión asociada (región “5” en el ejemplo). Este valor de comparación podría ser un objeto JSON, lo que equivale a tener un nuevo filtro, pero ahora sobre la dimensión referenciada, el que a su vez tiene campos que la filtran por sus clasificadores dentro del modelo de copo de nieve. La definición de filtro en ZRepo es recursiva. Un objeto “filtro” es una serie de campos clasificadores con su valor deseado. El valor puede ser el código de la dimensión a filtrar u otro objeto filtro sobre la dimensión filtrada.

Según el modelo de copo de nieve antes mostrado, las referencias entre las dimensiones que existen en GEOOs son:

Comuna → provincia → region.

Usando esa ruta de dependencias, podemos, por ejemplo, consultar todas las comunas que pertenecen a una provincia que está dentro de la quinta región. El filtro en este caso sería de la forma:

- GET: <https://zrepo.geoos.org/dim/ine.comuna/rows>
- filter: {"provincia": {"region": "5"}}

The screenshot shows the Postman application interface. In the left sidebar, under 'Scratch Pad', there is a collection named 'ZRepo - GEOOs REST API' containing two items: 'GET Provincias' and 'GET Comunas'. The 'GET Comunas' item is selected. The main workspace displays a 'GET' request to the URL [https://zrepo.geoos.org/dim/ine.comuna/rows?filter={"provincia":{"region":"5"}}](https://zrepo.geoos.org/dim/ine.comuna/rows?filter={). The 'Params' tab shows a single parameter 'filter' with the value '{"provincia":{"region":"5"}}'. The 'Body' tab shows the JSON response in 'Pretty' format:

```

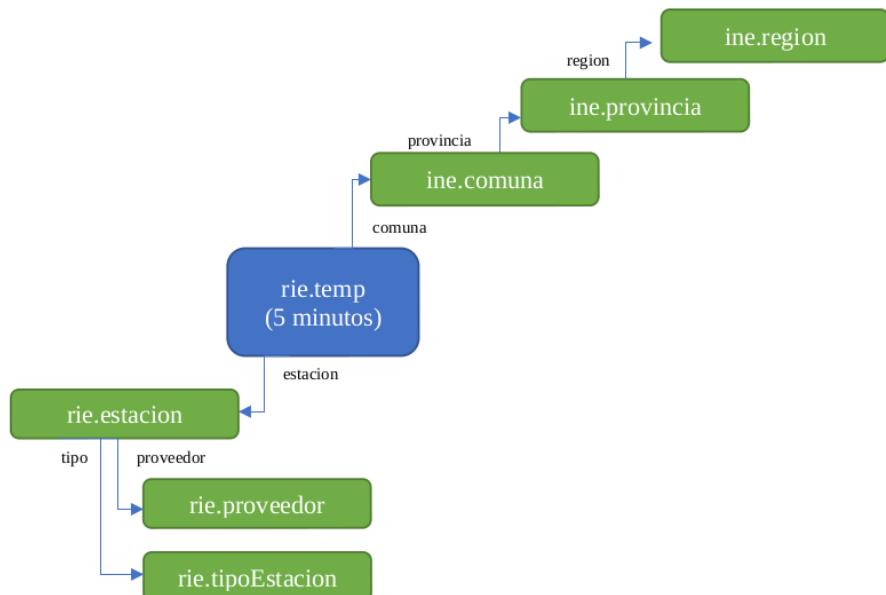
1  {
2    "_id": "5201",
3    "centroidLat": -27.05202646687835,
4    "code": "5201",
5    "name": "ISLA DE PASCUA",
6    "order": 23,
7    "provincia": "52"
8  },
9  {
10   "_id": "5101",
11   "centroidLat": -31.768237014496226,
12 }

```

The response status is 200 OK, with a time of 110 ms and a size of 4.49 KB. A 'Save Response' button is also visible.

RIE: Red Integrada de Estaciones

El Portal GEOOs permite consultar información de sensores o variables medidas por estaciones (meteorológicas, oceanográficas, mareógrafos, etc.). La información de las estaciones que se despliegan en GEOOs es administrada por ZRepo. Cada variable que se muestra asociada a las estaciones corresponde a una variable (Cubo) definido en ZRepo, con una temporalidad de 5 minutos, y de acuerdo con el siguiente modelo de Copo de Nieve.



Los proveedores de estaciones y los tipos pueden ser consultados usando el API REST de datos y metadatos ya mencionados. El modelo de estaciones permite consultar las variables, además de temporalmente, por su ubicación dentro de la división geopolítica de Chile (comuna, provincia, región) y su proveedor o tipo, como se verá más adelante en los ejemplos.

Los rangos de tiempo indicados como parámetros en las consultas se pueden enviar como milisegundos UTC (valor numérico de milisegundos transcurridos desde el 01-01-1970 00:00 UTC) o como un texto de fecha según formato ISO-8601 que incluye el corrimiento según zona horaria (por ejemplo, `2021-06-11T00:00:00-04:00`).

Consultas sobre Variables

Según el modelo de Copo de Nieve aplicado en ZRepo, las variables representan a cubos de información que acumulan datos de acuerdo con una temporalidad definida y una tupla de valores que dependen de sus clasificadores (dimensiones de clasificación).

El API REST de consultas sobre valores de variables de ZRepo utiliza el mismo lenguaje de especificación de filtros de las dimensiones, agregando el rango de fechas de consulta y algunos parámetros más que dependen del tipo de consulta que se desea realizar.

Para los siguientes ejemplos usaremos las variables de la Red Integrada de Estaciones de GEOOs, según el modelo de copo de nieve antes mostrado.

Consideraciones Generales:

Todas las consultas a variables ZRepo deben indicar una temporalidad, en la forma de un período de tiempo (rango).

Las temporalidades posibles de las variables ZRepo se identifican por un código, el que debe ser indicado en algunos tipos de consultas (las que retornan datos agrupados por esas temporalidades). Los códigos para cada temporalidad permitida son:

- 5m: 5 minutos, 15m: 15 minutos, 30m: 30 minutos
- 1h: 1 hora, 6h: 6 horas, 12h: 12 horas
- 1d: 1 día
- 1M: 1 mes, 3M: 3 meses, 4M: 4 meses, 6M: 6 meses
- 1y: 1 año.

Las consultas que requieren un rango de fechas esperan los parámetros “`startDate`” o “`endDate`”, según la especificación antes mencionada (milisegundos UTC o ISO-8601). Para estos casos el rango considerado incluye a la fecha de inicio y excluye a la de término: `[inicio-término[`

Dentro del metadato de la dimensión `rie.estacion` de RIE, se incluye la lista de las variables que cada estación mide. No todas las estaciones miden todas las variables, ya que dependerá de su tipo y los sensores que ellas posean. Por ejemplo, para consultar las variables que mide la estación con código “`quintero`” del servimet, podemos usar el API REST con la consulta GET:

- GET: <https://zrepo.geoos.org/dim/ine.estacion/rows>
- textFilter: quintero
- filter: {"proveedor": "servimet"}

The screenshot shows the Postman interface with a successful API call. The request URL is <https://zrepo.geoos.org/dim/rie.estacion/rows?textFilter=quintero&filter=%22proveedor%22%3D%22servimet%22>. The response body is a JSON object:

```

1  {
2   "_id": "quintero",
3   "code": "quintero",
4   "comuna": "5107",
5   "lat": -32.7761111111,
6   "lng": -71.52666666667,
7   "name": "Capitanía de Puerto - Quintero",
8   "order": 79,
9   "proveedor": "servimet",
10  "tipo": "meteo",
11  "variables": [
12    "rie.temp",
13    "rie.humedad",
14    "rie.punto_rcio",
15    "rie.vel_media_viento",
16    "rie.dir_viento",
17    "rie.presion_atm",
18    "rie.sens_termica",
19    "rie.indice_calor",
20    "rie.indice_uv"
21  ]
22}
23
24

```

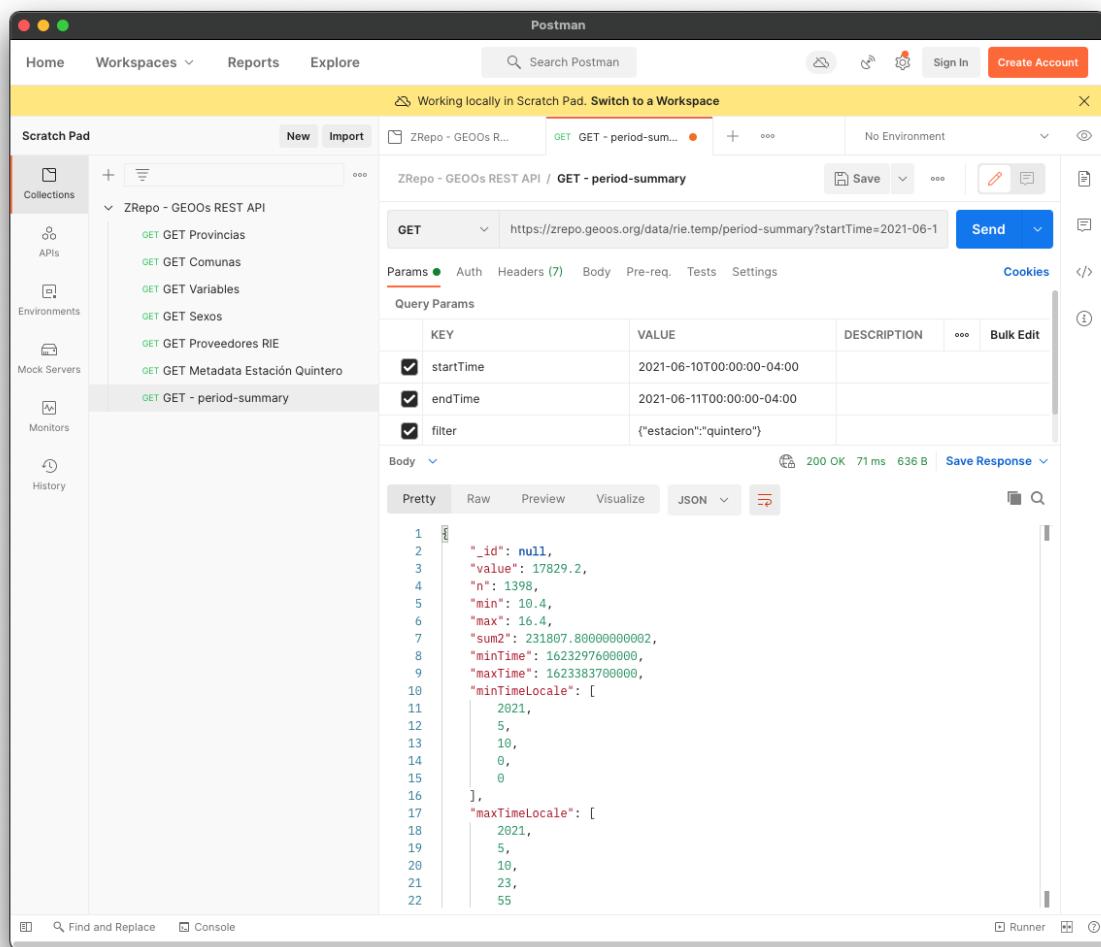
Valor Acumulado en un período:

Esta consulta permite obtener los valores acumulados para un período de tiempo, de acuerdo con los filtros y el rango de fechas. El formato de los filtros es un objeto JSON que aplica sobre los clasificadores en forma recursiva, según lo anteriormente explicado.

Se debe tener en cuenta que la acumulación de valores puede tener sentido para algunas variables y para otras no. Por ejemplo, para la temperatura en un período medida por las estaciones RIE, la suma no tiene sentido, pero sí su promedio y valores máximos y mínimos.

Para conocer la temperatura promedio de la estación de Quintero del Servimet (código "quintero") ejecutamos el siguiente request:

- GET: <https://zrepo.geoos.org/data/rie.temp/period-summary>
- startTime: 2021-06-10T00:00:00-04:00
- endTime: 2021-06-11T00:00:00-04:00
- filter: {"estacion": "quintero"}



Los datos retornados por la consulta anterior se interpretan de la siguiente forma:

- Para el día 10 de junio las temperaturas mínimas y máximas medidas por la estación de Quintero fueron 10.4 y 16.4 grados respectivamente.
- La temperatura media el 10 de junio fue de 12.7533 grado. Esto se obtiene de dividir la suma (campo “value”) por el total de muestras (campo “n”)
- Se agruparon 1398 muestras de temperatura para el día 10 de junio en la estación de quintero.

El campo **sum2** puede ser usado en conjunto con “n” para obtener la desviación estándar de los datos medidos.

Usando el campo filtro es posible acumular valores de acuerdo con lógicas más complejas. Por ejemplo, podemos obtener el resumen de período de la temperatura para todas las estaciones del servimet (proveedor servimet) dentro de alguna comuna que pertenezca a una provincia que esté dentro de la quinta región (Es decir las temperaturas de la 5ta región). El filtro que se usa en ese caso es:

`filter:{"estacion":{"proveedor":"servimet", "comuna":{"provincia":{"region":"5"}}}}`

con el que se obtienen los siguientes valores:

```

    "value": 70318.6,
    "n": 4200,
    "min": 10.4,
    "max": 23.7,
    "sum2": 1265067.52,
    "minTime": 1623297600000,
    "maxTime": 1623383700000,

```

Serie de Tiempo:

Cada fila que se retorna en las consultas de serie de tiempo corresponde a un agrupado temporal de los datos de la variable consultada, según los filtros indicados.

Además del período de inicio y término para la consulta, se debe indicar la agrupación temporal que deseamos para cada fila. Esta agrupación debe ser de nivel igual o superior al de la variable que se consulta; por ejemplo, para variables que se agrupan cada 1 hora, sólo puedo consultar por agrupaciones temporales de 1h, 6h, 1 día, un mes, etc. pero no por agrupaciones de 5 minutos.

- GET: <https://zrepo.geoos.org/data/rie.temp/time-serie>
- startTime: 2021-06-10T00:00:00-04:00
- endTime: 2021-06-11T00:00:00-04:00
- filter: {"estacion":"quintero"}
- temporality: 1h

```

3: {
    "_id": {
        "year": 2021,
        "month": 6,
        "day": 10,
        "hour": 0,
        "minute": 0
    },
    "value": 682.1,
    "n": 57,
    "min": 11.7,
    "max": 12.4,
    "sum2": 8164.67,
    "localTime": {
        "year": 2021,
        "month": 6,
        "day": 10,
        "hour": 0,
        "minute": 0
    },
    "time": 1623297600000
},

```

Cada fila retornada contiene los acumuladores de suma (value), mínimo, máximo, n (cantidad de muestras) y suma de cuadrados para cada período.

Datos agrupados por una Dimensión:

Para un período de tiempo es posible consultar los valores acumulados retornando grupos que pertenezcan a una dimensión clasificadora de la variable en un primer del copo de nieve, o en algún otro nivel de profundidad (por ejemplo, valores agrupados por región para consultas sobre estaciones RIE).

Los parámetros de la consulta deben incluir “**groupDimension**”. El valor de este parámetro es la ruta de la dimensión a agrupar, desde la variable hacia el exterior del copo de nieve, separando por punto cada nombre de campo. Por ejemplo, para consultar las temperaturas agrupadas por estación (sólo un nivel) se usa “**groupDimension=estacion**”, mientras que, para agruparlas por región, se usaría:

“groupDimension=estacion.comuna.provincia.region”

Cada fila retornada incluye el valor de la dimensión de agrupamiento como un atributo “**dim**” que corresponde a un objeto JSON que contiene el código, nombre y orden definido de la fila correspondiente a la dimensión. Por ejemplo, para obtener los valores acumulados por provincia de temperatura, para todas las provincias de la quinta región, se usa la consulta:

- GET: <https://zrepo.geoos.org/data/rie.temp/dim-serie>
- startTime: **2021-06-10T00:00:00-04:00**
- endTime: **2021-06-11T00:00:00-04:00**
- filter: **{"estacion":{"comuna":{"provincia":{"region ":"5"}}}}**
- GroupDimension: estacion.comuna.provincia

```
1 {
2   "_id": "52",
3   "value": 31152.4,
4   "n": 1399,
5   "min": 21.2,
6   "max": 23.7,
7   "sum2": 694185.28,
8   "dim": {
9     "code": "52",
10    "name": "ISLA DE PASCUA",
11    "order": 9
12  },
13 },
14 },
15 },
16 },
17 },
18 },
19 },
20 },
21 },
22 }
```

Tabla: Tiempo v/s Dimensión

Permite obtener una tabla de datos que cruza agrupaciones temporales con una dimensión que agrupe los valores consultados. Cada celda corresponde a los valores acumulados para una agrupación temporal y una de dimensión.

Se debe indicar el parámetro “**temporalidad**”, al igual que la serie de tiempo para indicar las agrupaciones temporales. Debe ser de orden igual o superior a la temporalidad de la variable que se consulta. Se debe indicar el parámetro “**groupDimension**” para especificar por qué dimensión (ruta separada por puntos) se usará como agrupador.

El siguiente ejemplo permite obtener una tabla que compara para cada región los valores de temperatura en cada hora de un día.

- GET: <https://zrepo.geoos.org/data/rie.temp/time-dim>
- startTime: **2021-06-10T00:00:00-04:00**
- endTime: **2021-06-11T00:00:00-04:00**
- filter: 0
- temporality: 1h
- groupDimension: estacion.comuna.provincia.region

The screenshot shows the Postman interface with a collection named "ZRepo - GEOOS REST API". A specific POST request is selected under the "Time-Dim" category. The "Params" tab shows the following values:

- startTime: 2021-06-10T00:00:00-04:00
- endTime: 2021-06-11T00:00:00-04:00
- filter: 0
- temporality: 1h
- groupDimension: estacion.comuna.provincia.region

The "Body" tab displays the JSON response:

```
1 {
2   "value": 2783.3,
3   "n": 171,
4   "min": 15.4,
5   "max": 16.7,
6   "sum2": 45313.67000000006,
7   "dim": {
8     "code": "1",
9     "name": "REGIÓN DE TARAPACÁ",
10    "order": 2
11  },
12  "localTime": {
13    "year": 2021,
14    "month": 6,
15    "day": 10,
16    "hour": 0,
17    "minute": 0
18  }
19 },
20 }
```

Cada fila retornada por la consulta corresponde a una celda de la matriz tiempo-dimensión. El valor temporal se retorna dentro del objeto “**localTime**”, mientras que el valor de la dimensión, dentro del objeto “**dim**” e incluye su código, nombre y orden. Cada celda contiene los atributos min, max, n, value (suma) y sum2 (suma de cuadrados).

Tabla: Dimensión v/s Dimensión

Para un período de tiempo, esta consulta permite recuperar una tabla en donde las filas y columnas son diferentes dimensiones y las celdas son los valores acumulados de una variable para ese par de dimensiones de filtrado. Al igual que las consultas anteriores, las dimensiones de filtrado son rutas desde la variable hacia dimensiones superiores (separadas por puntos) en el modelo de copo de nieve.

Los parámetros para indicar las dimensiones para filas y columnas son “**hGroupDimension**” y “**vGroupDimension**” y siguen la regla de rutas de nombres de clasificadores separados por puntos.

La siguiente consulta permite recuperar los datos para una tabla que compara los valores de temperatura medidos por estaciones según su proveedor (servimet, redmeteo) por cada región de Chile.

- GET: <https://zrepo.geoos.org/data/rie.temp/dim-dim>
- startTime: **2021-06-10T00:00:00-04:00**
- endTime: **2021-06-11T00:00:00-04:00**
- filter: 0
- temporality: 1h
- hGroupDimension: estacion.comuna.provincia.region
- vGroupDimension: estacion.proveedor

The screenshot shows the Postman interface with the following details:

- Scratch Pad** tab selected.
- Collections** section shows a collection named "ZRepo - GEOOs REST API" containing several API endpoints like "GET Provincias", "GET Comunas", etc.
- APIs**, **Environments**, **Mock Servers**, **Monitors**, and **History** sections are also visible on the left.
- Request Details**:
 - Method: GET
 - URL: https://zrepo.geoos.org/data/rie.temp/dim-dim?startTime=2021-06-10T00:00:00-04:00&endTime=2021-06-11T00:00:00-04:00
 - Params:
 - endTime
 - filter
 - temporality
 - hGroupDimension: estacion.comuna.provincia.region
 - vGroupDimension: estacion.proveedor
- Response Preview**:
 - Status: 200 OK
 - Time: 119 ms
 - Size: 3.86 KB
 - Save Response

```
1  {
2   "value": 70318.6,
3   "n": 4200,
4   "min": 10.4,
5   "max": 23.7,
6   "sum2": 1265067.52,
7   "hDim": [
8     {
9       "code": "5",
10      "name": "REGIÓN DE VALPARAISO",
11      "order": 6
12    }
13  ],
14  "vDim": [
15    {
16      "code": "servimet",
17      "name": "Servimet",
18      "order": 1
19    }
20  ],
21  {
22    "value": 222.2,
23    "n": 24
24  }
```

Instalación

La forma más sencilla de instalar ZRepo dentro de GEOOs es agregarlo como un servicio más dentro del stack instalado en Docker Swarm que se creó en el documento “[geoos-componentes-instalación](#)”. En ese documento sólo se instala el componente Portal y una instancia de un servidor Mongo DB (usado por el Portal). ZRepo utiliza la misma instancia del servidor MongoDB ya configurado, por lo que basta sólo referenciarlo.

Para poder iniciarse el servicio ZRepo requiere que exista su archivo de configuración “**zrepo.json**”. De acuerdo con la instalación de ejemplo iniciada en el documento referenciado, el directorio “**/opt/geoos/config**” está mapeado como volumen Docker a las configuraciones de cada servicio del stack Docker. Creamos entonces el archivo **/opt/geoos/config/zrepo.json** con el siguiente contenido inicial:

```
{  
    # Security - Comment to disable, after local users have been created  
    adminLogin: demo  
    adminPassword: test  
    timeZone:"America/Santiago"  
    masterToken:"demo-token"  
    tokens: {}  
    # Domains  
    domains: {}  
    # Dimensions  
    dimensions:{}  
    # Variables  
    variables: {}  
    # dataSets  
    dataSets:{}  
}
```

El componente Portal utiliza GeoServer para obtener la configuración de los proveedores de información (de capas). Los proveedores de estaciones (RIE) de ZRepo referencian a los mismos proveedores, por lo que es necesario ejecutar además una instancia de GeoServer (a menos que se refiera a una remota desde el Portal que defina los proveedores que se usarán).

De acuerdo con lo anterior, un stack de servicios mínimo para ejecutar GEOOs con ZRepo y GeoServer sería el definido por el siguiente archivo (**/opt/geoos/geoos.yml**):

```
#  
version: '3.6'  
services:  
  db:  
    image: mongo:4.4.1-bionic  
    environment:
```

```

MONGO_INITDB_ROOT_USERNAME: admin-user
MONGO_INITDB_ROOT_PASSWORD: admin-password
volumes:
- ./mongo-data:/data/db

portal:
image: docker.homejota.net/geoos/portal
ports:
- 8091:8090/tcp
environment:
MONGO_URL: "mongodb://admin-user:admin-password@db:27017"
MONGO_DATABASE: geoos
volumes:
- ./config:/home/config

geoserver:
image: docker.homejota.net/geoos/geoserver
ports:
- 8180:8080/tcp
environment:
LOG_LEVEL: debug
volumes:
- ./data:/home/data
- ./config:/home/config
- ./log:/home/log
- ./www:/home/www

zrepo:
image: docker.homejota.net/geoos/zrepo
deploy:
restart_policy:
condition: on-failure
ports:
- 8096:8096/tcp
environment:
MONGO_URL: "mongodb://admin-user:admin-password@db:27017"
MONGO_DATABASE: zrepo
volumes:
- ./config:/home/config
- ./log:/home/log

```

Para instalar o actualizar el stack de servicios, desde el directorio /opt/geoos se ejecuta el comando:

“docker stack deploy -c geoos.yml geoos”.

Para verificar que el servicio ha iniciado correctamente, podemos monitorear el log del servicio usando el comando:

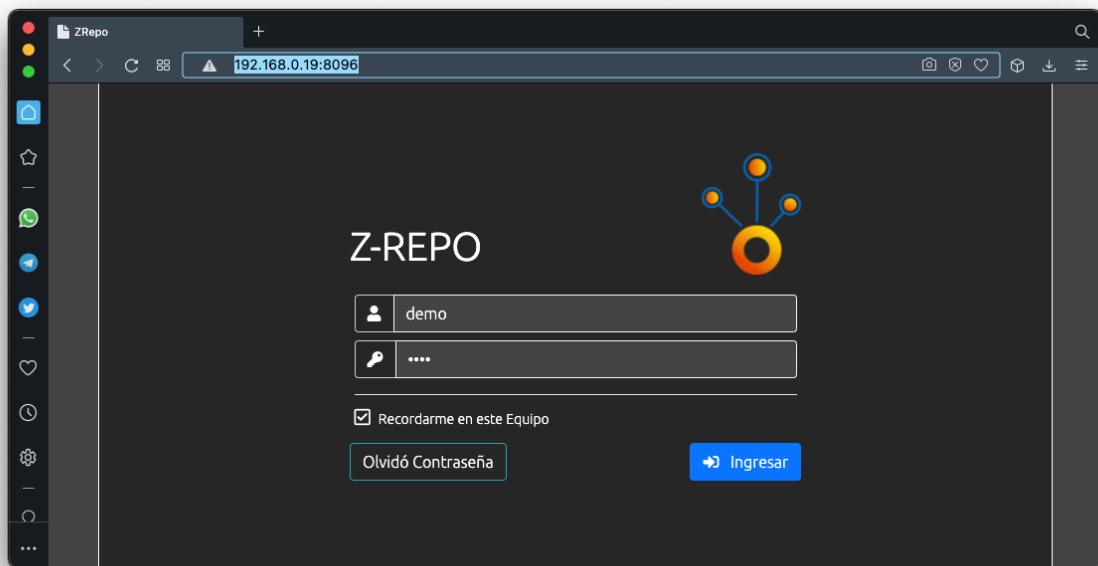
“docker service logs geoos_zrepo -f”

El que debería entregar los detalles del inicio, terminando con la línea:

```
geoos_zrepo.1.sxf0pw9zoo7q@jotanuc | Initializing ...  
geoos_zrepo.1.sxf0pw9zoo7q@jotanuc | [ZRepo HTTP Server] Listenning at Port 8096
```

ZRepo levanta un portal web que permite administrar los datos de las dimensiones y algunas operaciones básicas sobre los dataSets. Este portal está en proceso de mejoras, de tal forma de poder filtrar los datos de las dimensiones, de los dataSets y configurar dashboards con resúmenes de valores de las variables. Por el momento, las funcionalidades de administración básica de datos de las dimensiones están disponibles, por lo que se puede utilizar para editar sus filas, por ejemplo, las estaciones de la red RIE.

El portal se accede desde el puerto configurado (8096) y usando el usuario y contraseña usado en el archivo de configuración ([user:demo/pass:test](#))



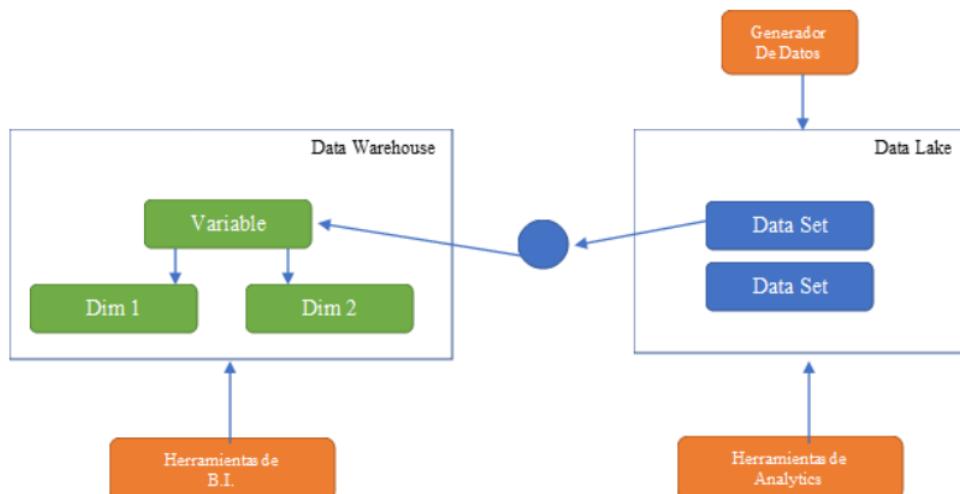
DataSets

Las dimensiones y variables mostradas hasta el momento entregan a ZRepo las capacidades de un DataWarehousing, orientado a Business Intelligence; es decir, a ofrecer capacidades de consulta de información clasificada temporal y multidimensionalmente. Ese tipo de consultas son muy útiles para la toma de decisiones, cálculos de indicadores, comparaciones temporales, tendencias, etc. sin embargo, debido al agrupamiento de la información (en cubos) los datos originales que acumularon sus valores se pierden. Por ejemplo, si algún dispositivo genera e inyecta datos periódicamente a ZRepo (en cualquier momento, al ocurrir los eventos, sin una temporalidad definida) las variables (cubos) normalizarán el tiempo del evento dentro de acuerdo con la temporalidad de la variable. Esto significa que el valor de la dimensión temporal se pasará a un valor “discreto”, cayendo dentro de un grupo determinado por la temporalidad. Si la temporalidad es horaria, por ejemplo, es imposible determinar el comportamiento temporal dentro de cada hora, usando los datos ya

agrupados. Lo mismo aplica a otras dimensiones, las que podrían clasificar datos continuos según una escala discreta.

Las técnicas actuales de Data Analytics requieren los datos lo más “crudos” posibles, de tal forma de poder aplicar algoritmos de reconocimiento de patrones, predicciones, etc. A diferencia de las Variables en ZRepo, los DataSets permiten almacenar y poner a disposición los datos crudos, antes de ser acumulados dentro de los cubos de información.

Un DataSet en ZRepo es una tabla de datos con una serie de columnas predefinidas, pero no obligatorias. Los DataSets son la forma preferida de agregar nueva información a las variables, ya que conservan los datos originales para ser luego consultados desde herramientas de analytics externas. Los DataSets de ZRepo equivalen a las unidades básicas que conforman un Data Lake, desde donde se alimenta el DataWarehouse, según el siguiente esquema:



ZRepo ofrece, a través de su API REST, operaciones para agregar datos directamente a las variables (acumular en los cubos); sin embargo, la forma preferida es configurar DataSets asociados a las cargas de datos, y usar los mecanismos de actualización de esos DataSets.

Los DataSets se actualizan usando también un API REST o cargando archivos (CSV) desde el portal ZRepo. Adicionalmente, se irán agregando a ZRepo otros formatos y mecanismos de carga de DataSets, como sincronizaciones con servidores remotos, etc. según las necesidades que se tengan a futuro.

Actualización de Datos en ZRepo

Tal como se mencionó antes, se usará la actualización de Variables de ZRepo pasando a través de un DataSet que permita conservar y consultar los datos originales agregados al repositorio (Data Lake).

Si bien ZRepo soporta datos sin asociación geográfica, GEOOs lo utiliza a través de su API REST para invocar consultas que comienzan desde la selección de un objeto georreferenciado, ya sea un polígono o una estación. Como ejemplo de importación de datos de una variable en ZRepo utilizaremos la publicación diaria que realiza el Ministerio de Ciencias de Chile de los nuevos casos de COVID-19 por región.

El detalle de esta publicación está en la siguiente dirección:

https://github.com/MinCiencia/Datos-COVID19/blob/master/output/producto13/CasosNuevosCumulativo_T.csv

Los datos son publicados y actualizados diariamente como un archivo CSV en la url:

https://raw.githubusercontent.com/MinCiencia/Datos-COVID19/master/output/producto13/CasosNuevosCumulativo_T.csv

El formato de los datos publicados es el siguiente:

A screenshot of a GitHub raw file showing a CSV table. The table has 16 columns representing different regions: Arica y Parinacota, Tarapacá, Antofagasta, Atacama, Coquimbo, Valparaíso, Metropolitana, O'Higgins, Maule, Ñuble, Biobío, Araucanía, Los Ríos, Los Lagos, Aysén, and Magallanes. The first column is labeled 'Region' and the second column is labeled with dates from 2020-03-03 to 2020-03-06. Each cell contains a numerical value representing the number of new cases for that specific date and region.

Region	2020-03-03	2020-03-04	2020-03-05	2020-03-06	Arica y Parinacota	Tarapacá	Antofagasta	Atacama	Coquimbo	Valparaíso	Metropolitana	O'Higgins	Maule	Ñuble	Biobío	Araucanía	Los Ríos	Los Lagos	Aysén	Magallanes
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Como se observa, cada fila corresponde a un día y las regiones están dadas por las columnas. Cada celda contiene la cantidad de casos nuevos de COVID para ese día y región. En este caso, cada fila del archivo original y del dataSet que se definirá, generará 16 muestras que se acumularán en la variable “**cie-covid.casos_nuevos**”.

Para poder asociar los datos a las regiones de Chile, debemos definirlas e importarlas en nuestro servidor. Para ello, en nuestro archivo **zrepo.json** (directorio config) agregamos primero el dominio “**ine**” que será desde donde GEOOs obtendrá la información de capas geopolíticas de Chile. Aprovechamos además de agregar el dominio “**cie-covid**” para representar el origen de datos de COVID-19 publicado por el Ministerio de Ciencias de Chile.

La sección de dominios del archivo **zrepo.json** queda entonces así:

```
# Domains
domains: {
    cie-covid:"Ministerio de Ciencia y Tecnología"
    ine:"Instituto Nacional de Estadísticas"
}
```

En el mismo archivo, ahora debemos definir la dimensión **ine.region**, de la siguiente forma:

```
# Dimensions
dimensions: {
    ine.region:{name:"Region", classifiers:[]}
}
```

Luego definimos la nueva variable que almacenará los nuevos casos de covid por cada región. La sección de variables queda entonces como:

```
# Variables
variables:{
    cie-covid.casos_nuevos:{
        name:"Caso Nuevos COVID"
        temporality:"1d"
        classifiers:[{fieldName:"region", name:"Region", defaultValue:"00", dimensionCode:"ine.region"}]
        options:{unit:"Nº"}
    }
}
```

A continuación, se debe definir el dataSet que usaremos para importar y almacenar los datos originales. Como parte de la definición del dataSet debemos indicar su temporalidad (puede ser igual a la de las variables o “**free**” para indicar que no hay restricciones).

La definición de un dataSet contiene una sección obligatoria con las columnas y dos secciones opcionales: una para indicar sus mecanismos de importación o sincronización, y la otra con los “**triggers**” que activan automáticamente la carga de variables. La definición de las columnas del nuevo dataSet es la siguiente:

```
# dataSets
dataSets:{
    cie-covid.reporte_casos_nuevos:{
        name:"COVID - Casos Nuevos por Region"
        temporality:"1d"
        columns:[
            {code:"fecha", name:"Fecha", shortName:"Fecha", type:"date", time:true, key:true}
            {code:"reg15", name:"Arica y Parinacota", shortName:"R15", type:"number"}
            {code:"reg1", name:"Tarapaca", shortName:"R1", type:"number"}
            {code:"reg2", name:"Antofagasta", shortName:"R2", type:"number"}
            {code:"reg3", name:"Atacama", shortName:"R3", type:"number"}
            {code:"reg4", name:"Coquimbo", shortName:"R4", type:"number"}
            {code:"reg5", name:"Valparaiso", shortName:"R5", type:"number"}
            {code:"reg13", name:"Metropolitana", shortName:"R13", type:"number"}
            {code:"reg6", name:"O Higgins", shortName:"R6", type:"number"}
            {code:"reg7", name:"Maule", shortName:"R7", type:"number"}
            {code:"reg16", name:"Nuble", shortName:"R16", type:"number"}
            {code:"reg8", name:"Biobio", shortName:"R8", type:"number"}
            {code:"reg9", name:"Araucania", shortName:"R9", type:"number"}
            {code:"reg14", name:"Los Rios", shortName:"R14", type:"number"}
            {code:"reg10", name:"Los Lagos", shortName:"R10", type:"number"}
            {code:"reg11", name:"Aysen", shortName:"R11", type:"number"}
            {code:"reg12", name:"Magallanes", shortName:"R12", type:"number"}
        ]
    }
}
```

La primera columna representa la fecha (**time:true, type:date**) y se le indica que es la clave que identifica los registros (**key:true**). Esto significa que, si se importa nuevamente el mismo archivo, los datos de la fila serán actualizados y no se agrega al final. Se debe tener en cuenta que esto **NO** sucede con las variables, las que siempre acumulan nuevos valores y no existe forma de eliminar una muestra específica (al acumularse se pierde la información detallada de todos sus campos para deshacer la acumulación). En el caso de las variables, se pone a disposición una operación del API REST para eliminar un período completo de datos de una variable.

La sección de importación de un dataSet puede ser (por el momento) de tipo “**upload**” o “**sync**”. El tipo “**upload**” permite que los usuarios puedan cargar desde la página del servidor ZRepo los archivos con las columnas definidas antes, mientras que la opción “**sync**” asume que los archivos están publicados en alguna URL vis:ble desde el servidor.

Para el caso del ejemplo, contamos con una URL pública (GitHub) en donde el Ministerio de Ciencias publica periódicamente los datos.

```

imports:[{
    type:"sync"
    url:"https://raw.githubusercontent.com/MinCiencia/Datos-COVID19/master/output/producto13/CasosNuevosCumulativo_T.csv"
    label:"Buscar Nuevos Datos"
    askForTime:false
    batchSize:100
    format:"csv", skipFirstLine:true, separator:","
    incremental:true, # Solo fechas mayores a la ultima
    mapFrom:{ 
        fecha:{columnIndex:0, timeFormat:"YYYY-MM-DD"}, 
        reg15:1, reg1:2, reg2:3, reg3:4, reg4:5, reg5:6, reg13:7, reg6:8, reg7:9, reg16:10, reg8:11, reg9:12, reg14:13, reg10:14, reg11:15, reg12:16
    }
}]

```

Por cada columna del dataSet, debemos indicar su ubicación en las columnas del archivo CSV con un número que corresponde a su índice, comenzando desde cero.

Finalmente completamos la sección de “**triggers**” del dataSet, en donde indicamos la forma en que se cargarán datos a las variables. Como los datos del dataSet se proveen en columnas (en este caso), cada fila del dataSet debe disparar la inserción de un dato por cada columna (región de Chile) en la variable **cie-covid.casos_nuevos**. Esta sección del archivo de configuración queda entonces como:

```

triggers:[
    type:"postVariable", variable:"cie-covid.casos_nuevos", value:"reg15", data:[{constant:"15", to:"region"}]
], {
    type:"postVariable", variable:"cie-covid.casos_nuevos", value:"reg1", data:[{constant:"1", to:"region"}]
}, {
    type:"postVariable", variable:"cie-covid.casos_nuevos", value:"reg2", data:[{constant:"2", to:"region"}]
}, {
    type:"postVariable", variable:"cie-covid.casos_nuevos", value:"reg3", data:[{constant:"3", to:"region"}]
}, {
    type:"postVariable", variable:"cie-covid.casos_nuevos", value:"reg4", data:[{constant:"4", to:"region"}]
}, {
    type:"postVariable", variable:"cie-covid.casos_nuevos", value:"reg5", data:[{constant:"5", to:"region"}]
}, {
    type:"postVariable", variable:"cie-covid.casos_nuevos", value:"reg13", data:[{constant:"13", to:"region"}]
}, {
    type:"postVariable", variable:"cie-covid.casos_nuevos", value:"reg6", data:[{constant:"6", to:"region"}]
}, {
    type:"postVariable", variable:"cie-covid.casos_nuevos", value:"reg7", data:[{constant:"7", to:"region"}]
}, {
    type:"postVariable", variable:"cie-covid.casos_nuevos", value:"reg16", data:[{constant:"16", to:"region"}]
}, {
    type:"postVariable", variable:"cie-covid.casos_nuevos", value:"reg8", data:[{constant:"8", to:"region"}]
}, {
    type:"postVariable", variable:"cie-covid.casos_nuevos", value:"reg9", data:[{constant:"9", to:"region"}]
}, {
    type:"postVariable", variable:"cie-covid.casos_nuevos", value:"reg14", data:[{constant:"14", to:"region"}]
}, {
    type:"postVariable", variable:"cie-covid.casos_nuevos", value:"reg10", data:[{constant:"10", to:"region"}]
}, {
    type:"postVariable", variable:"cie-covid.casos_nuevos", value:"reg11", data:[{constant:"11", to:"region"}]
}, {
    type:"postVariable", variable:"cie-covid.casos_nuevos", value:"reg12", data:[{constant:"12", to:"region"}]
}
]

```

Cada “**trigger**” indica desde qué columna del dataSet se obtiene el valor a agregar. En este caso, por cada trigger el campo “**value**” se obtiene desde la columna “**regNN**” asociada a la región. La sección “**data**” indica cómo obtener el resto de los datos a acumular en la variable, en particular debe indicar cómo obtener o calcular las referencias a dimensiones (en este caso “**region**”). Los valores pueden ser “**from**” o “**constant**”. Si es **from**, se obtienen desde una columna de este dataSet.

El contenido final del archivo **zrepo.json** es el siguiente:

```

{
# Security - Comment to disable, after local users have been created
adminLogin: demo
adminPassword: test
timeZone: "America/Santiago"
}

```

```
masterToken:"demo-token"
tokens: {}
# Domains
domains: {
cie-covid:"Ministerio de Ciencia y Tecnología"
ine:"Instituto Nacional de Estadística"
}
# Dimensions
dimensions: {
ine.region:{name:"Region", classifiers:[]}
}
# Variables
variables: {
cie.covid.casos_nuevos:{
name:"Casos nuevos COVID-19"
temporality:"1d"
classifiers:[{fieldName:"region", name:"Region", defaultValue:"00", dimensionCode:"ine.region" }]
options:{unit:"Nº"}
}
}
# dataSets
dataSets: {
cie.covid.reporte_casos_nuevos:{
name:"COVID - Casos nuevos por región"
temporality:"1d"
columns:[
{code:"fecha", name:"Fecha", shortName:"Fecha", type:"date", time:true, key:true}
{code:"reg15", name:"Arica y Parinacota", shortName:"R15", type:"number"}
{code:"reg1", name:"Tarapacá", shortName:"R1", type:"number"}
{code:"reg2", name:"Antofagasta", shortName:"R2", type:"number"}
{code:"reg3", name:"Atacama", shortName:"R3", type:"number"}
{code:"reg4", name:"Coquimbo", shortName:"R4", type:"number"}
{code:"reg5", name:"Valparaíso", shortName:"R5", type:"number"}
{code:"reg13", name:"Metropolitana", shortName:"R13", type:"number"}
{code:"reg6", name:"O'Higgins", shortName:"R6", type:"number"}
{code:"reg7", name:"Maule", shortName:"R7", type:"number"}
{code:"reg16", name:"Ñuble", shortName:"R16", type:"number"}
{code:"reg8", name:"Biobío", shortName:"R8", type:"number"}
{code:"reg9", name:"Araucanía", shortName:"R9", type:"number"}
{code:"reg14", name:"Los Ríos", shortName:"R14", type:"number"}
{code:"reg10", name:"Los Lagos", shortName:"R10", type:"number"}
{code:"reg11", name:"Aysén", shortName:"R11", type:"number"}
{code:"reg12", name:"Magallanes", shortName:"R12", type:"number"}
]
}
}
imports:[{
type:"sync"
}]
```

```
url: "https://raw.githubusercontent.com/MinCiencia/Datos-COVID19/master/output/producto13/CasosNuevosCumulativo\_T.csv"  
label: "Buscar nuevos datos"  
askForTime: false  
batchSize:100  
format:"csv", skipFirstLine:true, separator:",",  
incremental:true, #Solo fechas mayores a la última  
mapFrom:{  
fecha:{columnIndex:0, timeFormat:"YYYY-MM-DD"},  
reg15:1, reg1:2, reg2:3, reg3:4, reg4:5, reg5:6, reg13:7, reg6:8, reg7:9, reg16:10, reg8:11, reg9:12,  
reg14:13, reg10:14, reg11:15, reg12:16  
}  
}  
]  
  
triggers: [{  
type:"postVariable", variable:"cie-covid.casos_nuevos", value:"reg15", data: [{constant:"15",  
to:"region"}]  
, {  
type:"postVariable", variable:"cie-covid.casos_nuevos", value:"reg1", data: [{constant:"1",  
to:"region"}]  
, {  
type:"postVariable", variable:"cie-covid.casos_nuevos", value:"reg2", data: [{constant:"2",  
to:"region"}]  
, {  
type:"postVariable", variable:"cie-covid.casos_nuevos", value:"reg3", data: [{constant:"3",  
to:"region"}]  
, {  
type:"postVariable", variable:"cie-covid.casos_nuevos", value:"reg4", data: [{constant:"4",  
to:"region"}]  
, {  
type:"postVariable", variable:"cie-covid.casos_nuevos", value:"reg5", data: [{constant:"5",  
to:"region"}]  
, {  
type:"postVariable", variable:"cie-covid.casos_nuevos", value:"reg13", data: [{constant:"13",  
to:"region"}]  
, {  
type:"postVariable", variable:"cie-covid.casos_nuevos", value:"reg6", data: [{constant:"6",  
to:"region"}]  
, {  
type:"postVariable", variable:"cie-covid.casos_nuevos", value:"reg7", data: [{constant:"7",  
to:"region"}]  
, {  
type:"postVariable", variable:"cie-covid.casos_nuevos", value:"reg16", data: [{constant:"16",  
to:"region"}]  
, {  
type:"postVariable", variable:"cie-covid.casos_nuevos", value:"reg8", data: [{constant:"8",  
to:"region"}]
```

```

}, {
  type:"postVariable", variable:"cie-covid.casos_nuevos", value:"reg9", data: [{constant:"9",
  to:"region"}]
}, {
  type:"postVariable", variable:"cie-covid.casos_nuevos", value:"reg14", data: [{constant:"14",
  to:"region"}]
}, {
  type:"postVariable", variable:"cie-covid.casos_nuevos", value:"reg10", data: [{constant:"10",
  to:"region"}]
}, {
  type:"postVariable", variable:"cie-covid.casos_nuevos", value:"reg11", data: [{constant:"11",
  to:"region"}]
}, {
  type:"postVariable", variable:"cie-covid.casos_nuevos", value:"reg12", data: [{constant:"12",
  to:"region"}]
}
}
}

```

Antes de poder importar los datos desde la URL configurada, debemos completar las filas de datos de la dimensión **ine.region**. Para ello, podemos usar el archivo “**ine.region.json**” que se acompaña o se puede descargar desde:

<https://github.com/geoos/documentacion/tree/main/datos>

Desde la página del servidor ZRepo, seleccionamos “**Administrar Datos**” en el menú superior y “**Dimensiones**” a la izquierda. Nos aseguramos de que esté seleccionada la dimensión “**Region**” y usamos el botón “**Importar**” a la derecha. Seleccionamos el archivo **json** y deberíamos tener los nuevos datos importados.

The screenshot shows the Z-Repo application interface. On the left, there's a sidebar with icons for 'Dimensions' (selected), 'Data Sets', and 'Z-Repo'. The main area has a header with 'Region' selected in a dropdown, and buttons for 'Exportar', 'Importar', and 'Nueva Fila'. A table lists 16 regions with columns for 'Editar', 'Código', and 'Nombre / Etiqueta'. The row for 'REGIÓN DE TARAPACÁ' (Código 1) is highlighted with a blue background. To the right, a modal window titled '[1] REGIÓN DE TARAPACÁ' displays a JSON object under 'Datos Adicionales (JSON)':

```
{
  "centroidLat": -20.230142141799863
}
```

At the bottom of the modal is a 'Grabar' button with a save icon.

Una vez que los datos se han importado, podemos seleccionar “**DataSets**” a la izquierda y el botón “**Buscar Nuevos Datos**” arriba a la derecha. La importación de los datos ocurre como un proceso en segundo plano. Para monitorear los posibles errores, se puede consultar el archivo de log (directorio **/opt/geoos/log**) de zrepo (con el día como parte del nombre) o el log del servicio docker, usando:

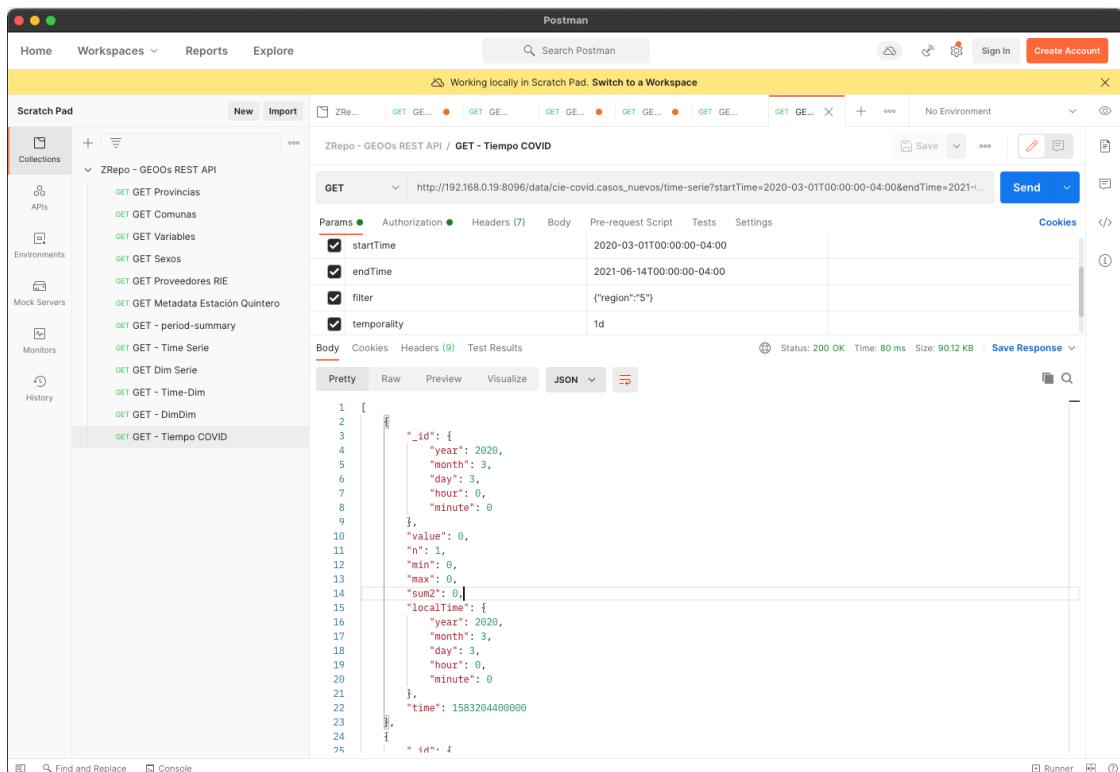
“docker service logs geoos_zrepo -f”

Los resultados en la variable pueden obtenerse usando el API REST contra el servidor local, por ejemplo, apuntando a la URL:

[http://192.168.0.19:8096/data/cie-covid.casos_nuevos/time-serie?startTime=2020-03-01T00:00:00-04:00&endTime=2021-06-14T00:00:00-04:00&filter={"region":"5"}&temporality=1d&token=demo-token](http://192.168.0.19:8096/data/cie-covid.casos_nuevos/time-serie?startTime=2020-03-01T00:00:00-04:00&endTime=2021-06-14T00:00:00-04:00&filter={)

También se puede usar la herramienta postman antes descrita:

- GET: http://192.168.0.19:8096/data/cie-covid.casos_nuevos/time-serie
 - startTime: 2021-03-10T00:00:00-04:00
 - endTime: 2021-06-14T00:00:00-04:00
 - filter: {"region":"5"}
 - temporality: 1d
 - token: demo-token

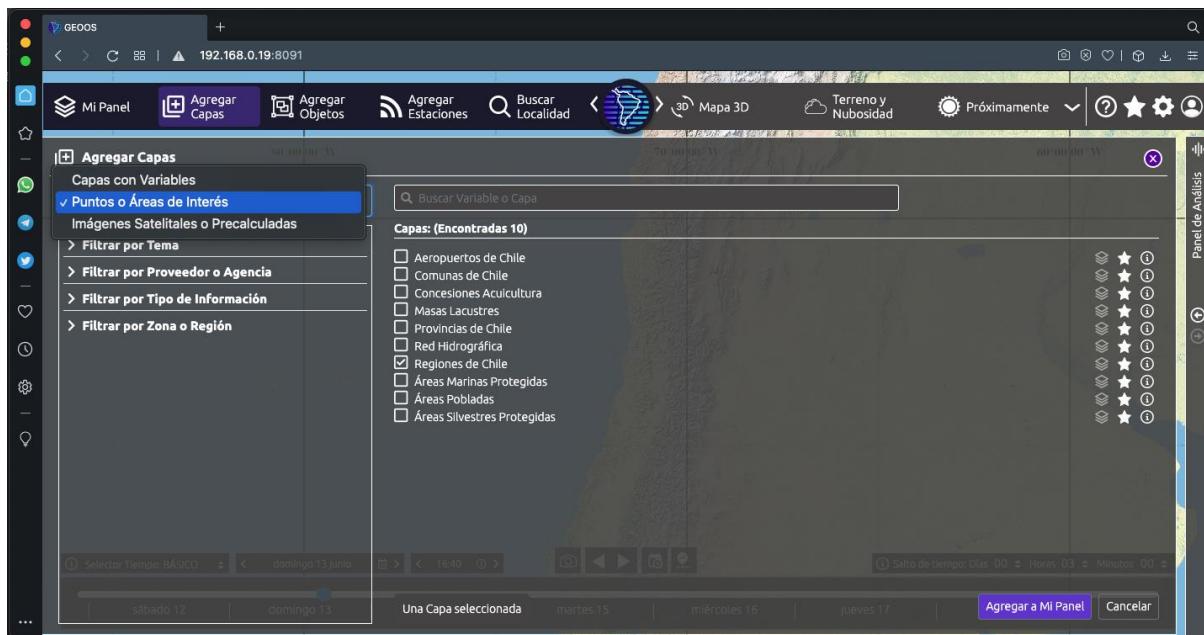


Para visualizar los datos dentro de GEOOs primero debemos modificar el archivo “**portal.hjson**” en la configuración para incorporar nuestro servidor ZRepo. Además, agregamos a los servidores en **geos.org** que ya contienen las capas de regiones (GeoServer) y el resto de las dimensiones. Para ello editamos el archivo **portal.hjson** y nos aseguramos de que los servidores sean:

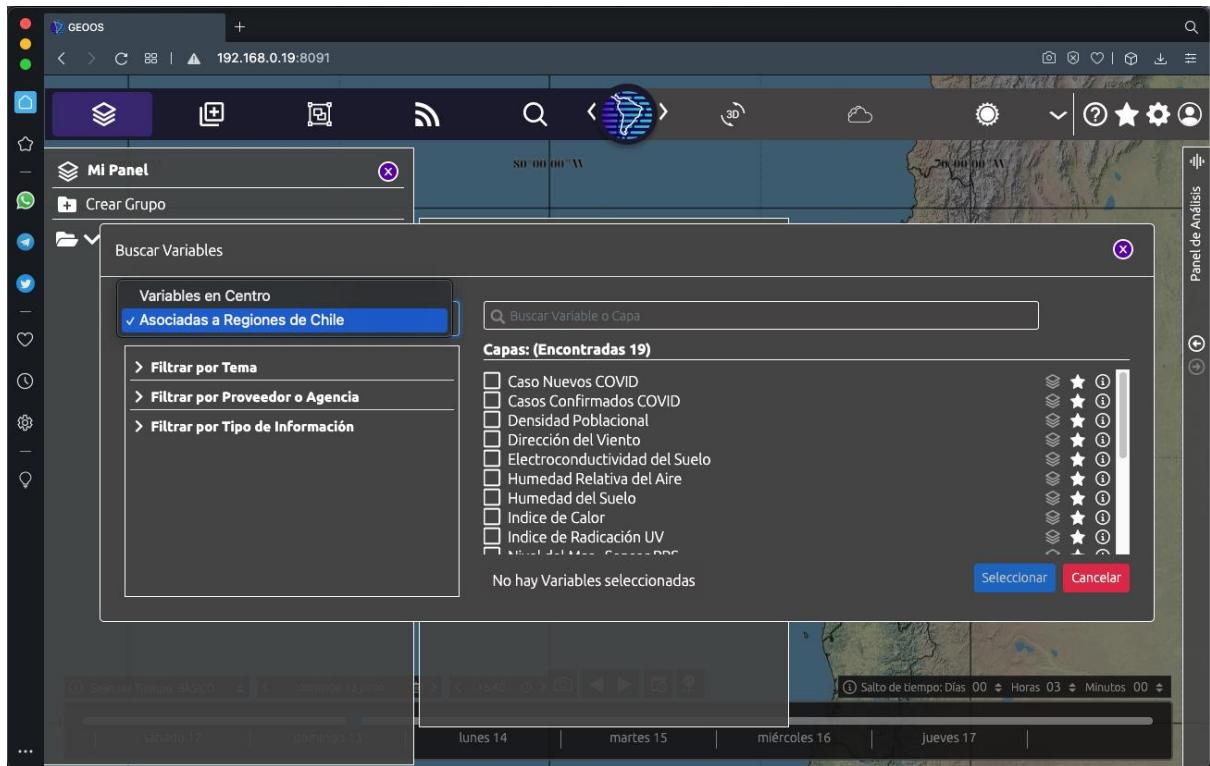
```
# Servers
geoServers:["https://geoserver.geoos.org", "http://192.168.0.19:8180"]
zRepoServers:[{
    url:"https://zrepo.geoos.org", token:"geoos-public"
}, {
    url:"http://192.168.0.19:8096", token:"demo-token"
}]
```

Se debe modificar el servidor “**192.168.0.19**” por el servidor local en donde se están ejecutando las pruebas. Luego se grabar los cambios (no es necesario reiniciar los servicios, pero si recargar la página en el browser si se estaba mostrando el portal) se carga la URL del portal, por ejemplo: <http://192.168.0.19:8091>

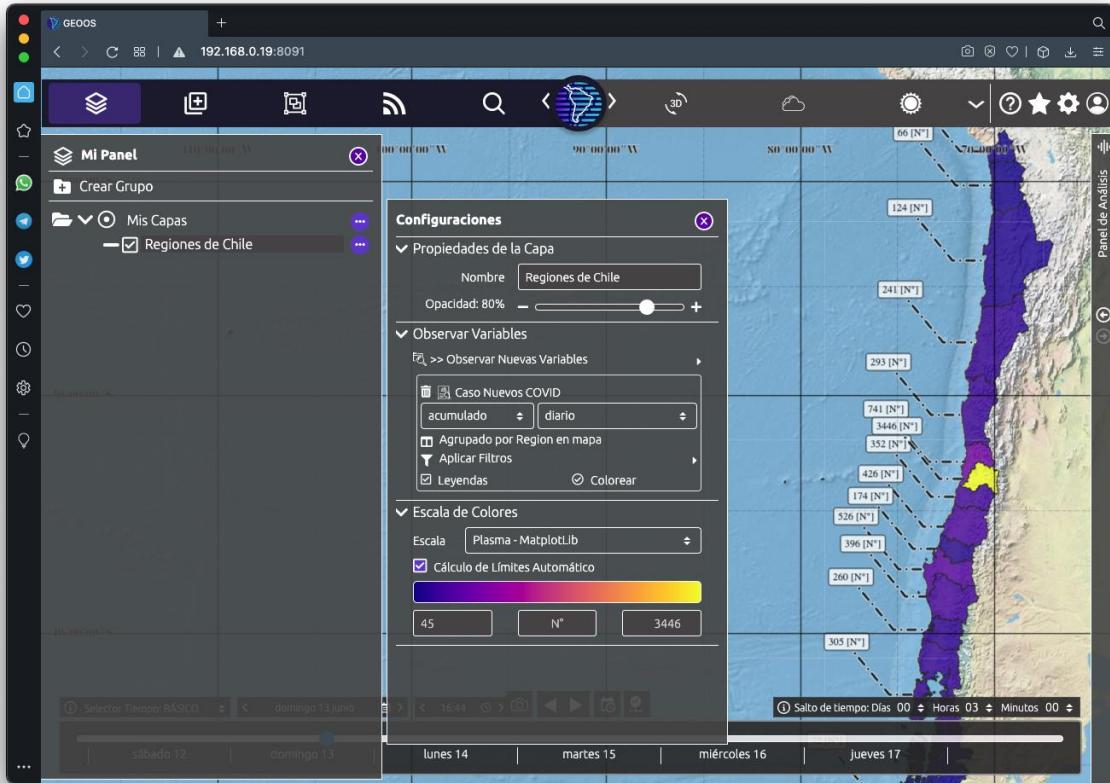
Usando la opción “agregar capa”. Y luego seleccionando “**Puntos o Áreas de Interés**” seleccionamos la capa “**Regiones de Chile**”



El mapa debería mostrar la capa vectorial de regiones. En el panel de capas seleccionamos esa capa y usamos “**Observar Variable**”. Luego seleccionamos “**Asociadas a Regiones de Chile**”.



Desde la lista seleccionamos “**Casos Nuevos COVID**”. Los polígonos de las regiones se pintan de acuerdo con una escala de colores definida (y modificable en las propiedades) y se muestra una etiqueta con la cantidad de casos para la fecha seleccionada. Dependiendo de la hora del día de la consulta, es posible que sea necesario modificar el tiempo en el mapa y navegar uno o dos días hacia atrás para poder ver datos (esto depende de las actualizaciones del Ministerio en el repositorio público).



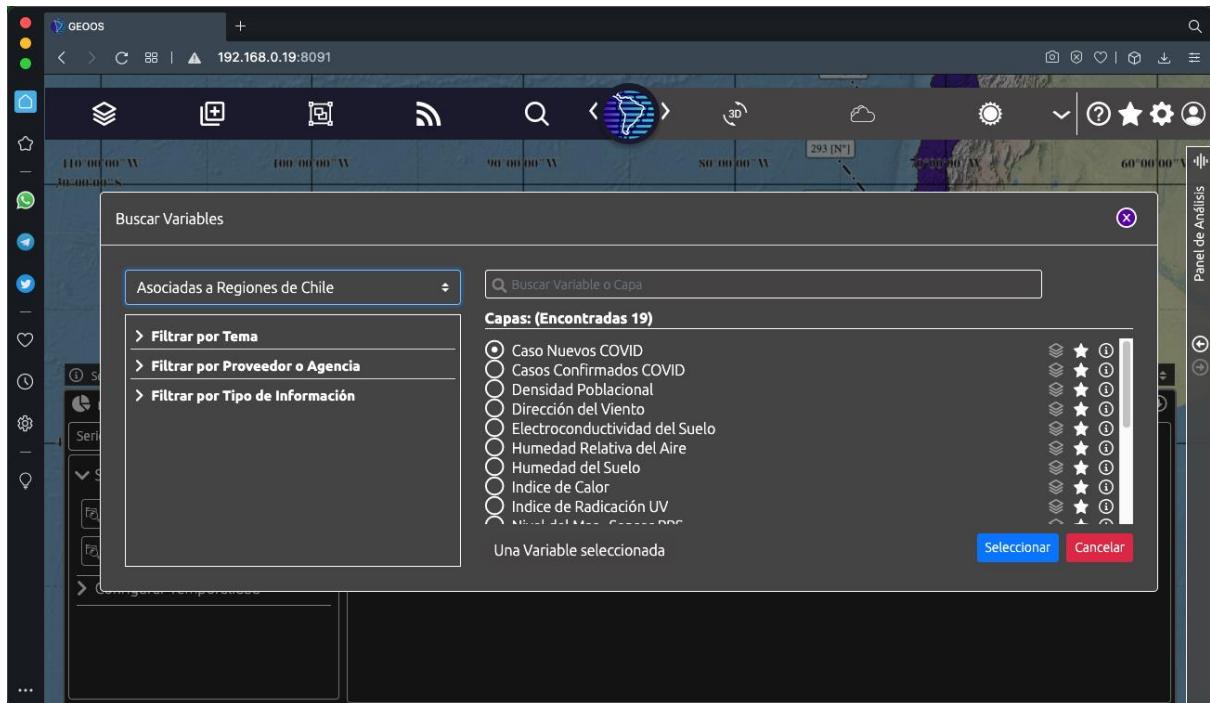
```

regiones: {
    cache:true
    tiledCache:true
    commonName:"Regiones de Chile"
    metadata:{ 
        idProperty:"REGION"
        nameProperty:"NOM_REGION"
        centroid:true
        center:true
        copyProperties:{ 
            vivi_particulares:"PARTICULAR"
            vivi_colectivas:"COLECTIVAS"
            num_hombres:"HOMBRES"
            num_mujeres:"MUJERES"
            densidad:"DENSIDAD"
            indice_mas:"INDICE_MAS"
            indice_dep:"INDICE_DEP"
            indice_dep_ju:"INDICE_DEP_JU"
            indice_dep_ve:"INDICE_DEP_VE"
        }
    }
    options:{ 
        minZDimension:"ine.region"
    }
}

```

GEOOs necesita relacionar los objetos vectoriales en la capa mostrada (Regiones de Chile en este caso) con las filas de una dimensión, para poder mostrar datos de variables asociados a esa dimensión (directa o indirectamente, saltando varios pasos en el modelo de copo de nieve). Esta asociación se hace en la configuración del componente “GeoServer”, en particular en el archivo de configuración de las capas asociadas (**ine-regional** en el servidor geoos.org). Ahí existe una declaración como la que se muestra a la izquierda. Dentro de las “options” se declara el atributo “minZDimension” como “**ine.region**” que es el código de la dimensión declarada en **zrepo.hjson** (minZ es el subcomponente encargado del DataWarehousing dentro del componente ZRepo).

Al seleccionar una región del mapa, el Portal muestra en la zona de análisis (zona inferior de la pantalla) un panel con las propiedades del objeto vectorial seleccionado. A la izquierda y arriba en ese panel inferior, es posible seleccionar el panel de análisis que se desea visualizar. En particular, se puede seleccionar “**Serie de Tiempo**”. Como variable principal se debe seleccionar “**Asociada a Regiones de Chile**” y “**Casos Nuevos COVID**”



Usando la sección “**Configurar Temporalidad**” a la izquierda abajo, es posible indicar la cantidad de días que se desea mostrar en el gráfico, por ejemplo, 450 días:



Al seleccionar una nueva región en el mapa, el gráfico se actualiza para mostrar los datos de esa región.

API REST para actualizaciones

Hasta el momento se ha mostrado acá cómo actualizar los datos de las variables desde archivos, ya sea cargados por los usuarios en el portal de ZRepo o sincronizados desde una URL visible por el servidor.

Existen muchos casos de actualización (automática principalmente) en donde los datos de los dataSets se obtienen o capturan uno a uno (o pocas filas) y se desean actualizar en línea en ZRepo (para visualizarlos en GEOOs). Este es el caso de las estaciones de la red RIE, por ejemplo. En estos casos es conveniente la utilización del API REST de actualización o carga de datos. Existen APIs para cargar directamente a las variables y otras para los dataSets. Al igual que el caso anterior (carga de archivos) se recomienda el uso de dataSets para representar los datos originales y conservar los detalles para futuros usos (herramientas de Analytics).

Al insertarse una nueva fila a un dataSet mediante el API REST, se activan los “triggers” definidos para la acumulación de valores de variables. En estos casos, no es necesaria la definición de la sección de “**imports**”, ya que el “**body**” esperado en el request “**POST**” del API REST debe contener los mismos códigos de columnas que define el dataSet al que se le agregan los datos.

Las actualizaciones de dataSets mediante el API REST requieren de una autorización especial en los tokens de seguridad. La sección “**tokens**” del archivo **zrepo.hjson** permite definir varios tokens, cada uno con diferentes autorizaciones sobre los datos. Las autorizaciones se aplican para cada dataSet y pueden ser de lectura y/o escritura por separado. Esto permite definir tokens “más seguros” para que alguna institución externa pueda “postear” los datos de alguna estación RIE, por ejemplo, sin entregar públicamente el mismo token a quienes sólo pueden consultar los datos.

El formato de asignación de privilegios a tokens es el siguiente:

```
masterToken: "demo-token"
tokens: {
    TOKEN-uno: {dataSet1-read:true, dataSet1-write:true, dataSet2:read}
}
```

El “**masterToken**” tiene autorización a consultar y modificar todos los dataSets y variables de ZRepo, por lo que debería ser utilizado sólo en modo de desarrollo y pruebas del software. Por cada dataset (“**TOKEN-uno**”, por ejemplo) se debe indicar su lista de autorizaciones.

En las llamadas POST para agregar datos a un dataSet (y disparar sus triggers que acumulan en variables) se puede suministrar el token como un Header HTTP (Authorization: Bearer \${token}) según el estándar OAuth 2.0, o como un parámetro más llamado “token” dentro del objeto JSON que es parte del “body” del request HTTP.

Para agregar una fila a un dataSet, basta con realizar una invocación (request) de tipo “**POST**” a la URL del servidor, seguida por /dataSet/codigoDelDataSet con un objeto json que

contenga atributos que correspondan a los códigos de las columnas definidas en el archivo zrepo.hjson para ese dataSet.

Para acumular valores directamente sobre una variable, sin pasar por los dataSets, se debe invocar un request POST a la URL: /data/codigoVariable con un objeto JSON en donde se debe incluir un atributo “time” (opcional, numérico con milisegundos UTC – se asume “ahora” si no se indica explícitamente), un atributo “value” de tipo numérico, que representa al valor que se acumula y un atributo por cada referencia a dimensión de la variable. El nombre de esos atributos de referencia debe ser el mismo definido como “fieldName” en la sección “classifiers” de las variables.

Carga de datos a RIE: Red Integrada de Estaciones

Como antes se mencionó, los datos de las estaciones en GEOOs se almacenan como variables ZRepo, asociadas a una “comuna” (dimensión “ine.comuna”) y a una “estacion” (dimensión “rie.estacion”). El servidor ZRepo que es parte de la instalación de GEOOs en el sitio geoos.org tiene definidas ya una serie de variables posibles de ser usadas por las estaciones de la red (dependiendo de su tipo). Los datos extra de cada estación (como fila de la dimensión “rie.estacion” identifican las coordenadas (latitud, longitud) de la estación y las variables que ella mide.

Editar	Código	Nombre / Etiqueta
<input checked="" type="checkbox"/>	yy:00:00:00:00:06	Chiguayante
<input checked="" type="checkbox"/>	yy:00:00:00:30	Pitrufquén - Colegio Madres Dominicas
<input checked="" type="checkbox"/>	pisagua	Alcaldía de Mar - Pisagua - Modificada
<input checked="" type="checkbox"/>	iquique	Capitanía de Puerto - Iquique
<input checked="" type="checkbox"/>	patache	Capitanía de Puerto - Patache
<input checked="" type="checkbox"/>	mejillones	Capitanía de Puerto - Mejillones
<input checked="" type="checkbox"/>	antofagasta	Faro Punta Molo - Antofagasta
<input checked="" type="checkbox"/>	taltal	Capitanía de Puerto - Taltal
<input checked="" type="checkbox"/>	chanaral	Capitanía de Puerto - Chañaral
<input checked="" type="checkbox"/>	caldera	Capitanía de Puerto - Caldera
<input checked="" type="checkbox"/>	pascua	Capitanía de Puerto - Hanga Roa
<input checked="" type="checkbox"/>	huasco	Capitanía de Puerto - Huasco
<input checked="" type="checkbox"/>	tortuga	Faro Punta Tortuga - Coquimbo
<input checked="" type="checkbox"/>	losvilos	Capitanía de Puerto - Los Vilos
<input checked="" type="checkbox"/>	quintero	Capitanía de Puerto - Quintero
<input checked="" type="checkbox"/>	valparaiso	Faro Punta Molo - Valparaíso
<input checked="" type="checkbox"/>	pichilemu	Capitanía de Puerto - Pichilemu
<input checked="" type="checkbox"/>	talcahuano	Gobernación Marítima - Talcahuano
<input checked="" type="checkbox"/>	lota	Capitanía de Puerto - Lota
<input checked="" type="checkbox"/>	carahue	Capitanía de Puerto - Carahue (Puerto Saavedra)
<input checked="" type="checkbox"/>	cenmeteopmo	Gobernación Marítima - Puerto Montt

El proceso para agregar una nueva estación es agregar la fila a la lista de estaciones (filas de datos de la dimensión “rie.estacion” mostrada en la imagen anterior, definir un dataSet en **zrepo.hjson** e indicar los triggers que llenan las variables desde el dataSet. Para el llenado de los datos en el dataSet, existen dos posibilidades.

1. Que la plataforma en donde se almacenan los datos de la estación invoque periódicamente al POST de ZRepo para agregar una fila al dataSet o que sea la misma estación quien lo haga.
2. Construir un proceso automático que consulte periódicamente el repositorio (normalmente en la nube o sitio del proveedor de la estación) y por cada nuevo muestreo, invocar al POST de ZRepo que agrega los valores al dataSet.

