

# API DEL SERVICIO DE NORMALIZACIÓN DE DATOS GEOGRÁFICOS DE ARGENTINA

**Versión :** 0.3.3

La API del Servicio de Normalización de Datos Geográficos, permite normalizar y codificar los nombres de unidades territoriales de la Argentina (provincias, departamentos, municipios y localidades) y de sus calles, así como ubicar coordenadas dentro de ellas.

En la Guía para la identificación y uso de entidades interoperables se explica cómo funcionan las unidades territoriales internas de la Argentina y la relación entre ellas.

## NORMALIZAR

Las unidades territoriales tienen nombres y códigos oficiales. Cuando no se usan, los datos son difíciles de cruzar entre sí y hay que normalizarlos antes.

provincia
Santiago del Estero
Stgo. del Estero
S. del Estero
Sgo. del Estero

GET <https://apis.datos.gob.ar/georef/api/provincias?nombre=Sgo.%20del%20Estero>

```
{
  "provincias": [
    {
      "nombre": "Santiago del Estero",
      "id": "86",
      "centroide": {
        "lat": -27.782412,
        "lon": -63.252387
      }
    }
  ],
  "cantidad": 1,
  "total": 1,
  "inicio": 0
}
```

## ENRIQUECER

Cuando un conjunto de datos tiene puntos de coordenadas dentro de Argentina, puede cruzarse con muchos datos más, relacionados a las unidades territoriales que lo contienen. Para esto hay que agregarlas a los datos originales.

lat	lon
-27.2741	-66.7529
-34.603633	-58.3837587

GET <https://apis.datos.gob.ar/georef/api/ubicacion?lat=-27.2741&lon=-66.7529>

```
{
  "ubicacion": {
```

```

    "departamento": {
      "id": "10035",
      "nombre": "Belén"
    },
    "lat": -27.2741,
    "lon": -66.7529,
    "municipio": {
      "id": "100077",
      "nombre": "Hualfín"
    },
    "provincia": {
      "id": "10",
      "nombre": "Catamarca"
    }
  }
}

```

## REFERENCIA

Finalmente, se puede utilizar la API como punto de referencia al momento de crear datos que estén vinculados a datos geográficos. Por ejemplo, si se cuenta con un formulario en el que se debe mostrar a un usuario un listado de provincias, y luego un listado de municipios a partir de la provincia seleccionada, se podrían ejecutar las siguientes consultas:

Listar las provincias de la República Argentina:

```
GET https://apis.datos.gob.ar/georef/api/provincias?campos=id,nombre
```

```

{
  "provincias": [
    {
      "nombre": "Chaco",
      "id": "22"
    },
    { ... } // 23 resultados omitidos
  ],
  "cantidad": 24,
  "total": 24,
  "inicio": 0
}

```

Asumiendo que el usuario selecciona **Chaco** (ID: **22**), se ejecutaría la siguiente consulta para obtener el listado de municipios:

```
GET https://apis.datos.gob.ar/georef/api/municipios?
provincia=22&campos=id,nombre&max=100
```

```

{
  "municipios": [
    {
      "nombre": "Makallé",
      "id": "220161"
    },
    { ... } // 67 resultados omitidos
  ],
  "cantidad": 68,
  "total": 68,
  "inicio": 0
}

```

Notar que al ser datos que no son modificados regularmente, es posible retener copias de los mismos para ser reutilizados en el futuro.

---

*Si sos usuario de la API de Normalización de Datos Geográficos y querés estar al tanto de los cambios y novedades, inscribite en la base de contactos de Datos Argentina y elegí sobre qué temas querés que te escribamos.*

*¿Tenés algo que contarnos? Nos encantaría que nos cuentes por Twitter o por mail qué mejoró usar la API en tu trabajo y qué le falta para mejorar aún más.*

# EJEMPLOS DE USO

## EJEMPLOS RÁPIDOS

A continuación, se muestran algunos ejemplos de uso de la API, utilizando los recursos `GET` :

### BÚSQUEDA DE PROVINCIAS

`GET https://apis.datos.gob.ar/georef/api/provincias?nombre=cordoba`

```
{
  "provincias": [
    {
      "id": "14",
      "centroide": {
        "lat": -32.142933,
        "lon": -63.801753
      },
      "nombre": "CÓRDOBA"
    }
  ],
  "cantidad": 1,
  "total": 1,
  "inicio": 0
}
```

### BÚSQUEDA DE DEPARTAMENTOS

`GET https://apis.datos.gob.ar/georef/api/departamentos?provincia=jujuy&max=16`

```
{
  "departamentos": [
    {
      "id": "38042",
      "centroide": {
        "lat": -24.194923,
        "lon": -65.12645
      },
      "nombre": "PALPALÁ",
      "provincia": {
        "id": "38",
        "nombre": "JUJUY"
      }
    },
    { ... } // 15 departamentos omitidos
  ],
  "cantidad": 16,
  "total": 16,
  "inicio": 0
}
```

### BÚSQUEDA DE MUNICIPIOS

`GET https://apis.datos.gob.ar/georef/api/municipios?provincia=tucuman&aplanar`

```
{
  "municipios": [
    {
      "centroide_lat": -27.816619,
      "centroide_lon": -65.199594,
```

```

        "id": "908210",
        "nombre": "Taco Ralo",
        "provincia_id": "90",
        "provincia_nombre": "Tucumán"
    },
    { ... } // 9 municipios omitidos
],
"cantidad": 10,
"total": 112,
"inicio": 0
}

```

## BÚSQUEDA DE LOCALIDADES

GET <https://apis.datos.gob.ar/georef/api/localidades?provincia=chubut&campos=nombre>

```

{
  "localidades": [
    {
      "id": "26007030000",
      "nombre": "PUERTO PIRAMIDE"
    },
    { ... } // 9 resultados omitidos
  ],
  "cantidad": 10,
  "total": 90,
  "inicio": 0
}

```

## NORMALIZACIÓN DE DIRECCIONES

GET <https://apis.datos.gob.ar/georef/api/direcciones?departamento=merlo&direccion=Florida al 1801>

```

{
  "direcciones": [
    {
      "altura": {
        "unidad": null,
        "valor": "1801"
      },
      "calle": {
        "categoria": "CALLE",
        "id": "0653901003005",
        "nombre": "FLORIDA"
      },
      "calle_cruce_1": {
        "categoria": null,
        "id": null,
        "nombre": null
      },
      "calle_cruce_2": {
        "categoria": null,
        "id": null,
        "nombre": null
      },
      "departamento": {
        "id": "06539",
        "nombre": "Merlo"
      },
      "nomenclatura": "FLORIDA 1801, Merlo, Buenos Aires",
      "piso": null,
      "provincia": {
        "id": "06",
        "nombre": "Buenos Aires"
      }
    },
    { ... } // 9 direcciones omitidas
  ]
}

```

```

        "ubicacion": {
            "lat": null,
            "lon": null
        }
    },
    "cantidad": 1,
    "total": 1,
    "inicio": 0
}

```

## LISTADO DE CALLES

GET <https://apis.datos.gob.ar/georef/api/calles?departamento=rio-chico&categoria=avenida>

```

{
  "calles": [
    {
      "altura": {
        "fin": {
          "derecha": 0,
          "izquierda": 0
        },
        "inicio": {
          "derecha": 0,
          "izquierda": 0
        }
      },
      "departamento": {
        "id": "90077",
        "nombre": "Río Chico"
      },
      "id": "9007701000050",
      "nombre": "AV GRL SAVIO",
      "nomenclatura": "AV GRL SAVIO, Río Chico, Tucumán",
      "provincia": {
        "id": "90",
        "nombre": "Tucumán"
      },
      "categoria": "AV"
    },
    { ... } // 2 resultados omitidos
  ],
  "cantidad": 3,
  "total": 3,
  "inicio": 0
}

```

# CONSULTAS POR LOTES

Todos los recursos de la API tienen una variante `POST` que permite realizar varias consultas en una misma petición. De esta forma, se pueden enviar más consultas en menos tiempo. Las versiones de los recursos `POST` aceptan los mismos parámetros que las `GET`, con la excepción del parámetro `formato`, que obligatoriamente toma el valor `json`. Adicionalmente, todos los parámetros se envían a través del cuerpo de la consulta HTTP, y no como parte del *query string*.

Por ejemplo, las siguientes tres consultas:

GET <https://apis.datos.gob.ar/georef/api/provincias?nombre=cordoba&campos=nombre>

GET <https://apis.datos.gob.ar/georef/api/provincias?nombre=chaco&campos=nombre>

GET <https://apis.datos.gob.ar/georef/api/provincias?nombre=san-luis&campos=nombre>

Son equivalentes a la siguiente consulta `POST` por lotes:

`POST https://apis.datos.gob.ar/georef/api/provincias`

```
{
  "provincias": [
    {
      "nombre": "cordoba",
      "campos": "nombre"
    },
    {
      "nombre": "chaco",
      "campos": "nombre"
    },
    {
      "nombre": "san luis",
      "campos": "nombre"
    }
  ]
}
```

Que resultaría en la siguiente respuesta JSON:

```
{
  "resultados": [
    {
      "cantidad": 1,
      "inicio": 0,
      "provincias": [
        {
          "id": "14",
          "nombre": "Córdoba"
        }
      ],
      "total": 1
    },
    {
      "cantidad": 1,
      "inicio": 0,
      "provincias": [
        {
          "id": "22",
          "nombre": "Chaco"
        }
      ],
      "total": 1
    },
    {
      "cantidad": 1,
      "inicio": 0,
      "provincias": [
        {
          "id": "74",
          "nombre": "San Luis"
        }
      ],
      "total": 1
    }
  ]
}
```

Como se muestra en el ejemplo, la respuesta contiene una lista `resultados`, con los resultados de cada consulta individual adentro. Las estructuras de las respuestas se mantienen idénticas que los recursos `GET`.

Cantidad máxima de consultas

La cantidad de consultas en una misma petición no debe superar las 1000.

Adicionalmente, el total de los parámetros `max` sumados de todas las consultas no debe superar los 5000. Por ejemplo, se permite enviar 1000 consultas con `max=5`, o 100 consultas con `max=50`, pero no 1000 consultas con `max=10`.

Utilizando los recursos por lotes, se pueden normalizar mayores cantidades de datos en menos tiempo. Por ejemplo, si se cuenta con 50000 direcciones, tan solo se necesitan 10 consultas para normalizar el activo de datos entero. Utilizando los recursos `GET`, se necesitarían 50000 (una por dato).

## EJEMPLOS DE USO

A diferencia de los recursos `GET`, los ejemplos de operaciones por lotes se muestran utilizando comandos contruïdos sobre la herramienta `curl`. La sección de ejemplos con Python también contiene ejemplos de uso de los recursos `POST`.

### BÚSQUEDA DE MUNICIPIOS EN LOTES

```
curl -X POST "https://apis.datos.gob.ar/georef/api/municipios" \
-H 'Content-Type: application/json' -d'
{
  "municipios": [
    {
      "nombre": "belgrano",
      "max": 1,
      "campos": "id, nombre"
    },
    {
      "nombre": "martin",
      "max": 1,
      "provincia": "la pampa",
      "aplanar": true
    }
  ]
},
'
```

Resultados:

```
{
  "resultados": [
    {
      "municipios": [
        {
          "id": "060301",
          "nombre": "General Belgrano"
        }
      ],
      "cantidad": 1,
      "total": 8,
      "inicio": 0
    },
    {
      "municipios": [
        {
          "centroide_lat": -35.361211,
          "centroide_lon": -64.294073,
          "id": "420126",
          "nombre": "Embajador Martini",
          "provincia_id": "42",
          "provincia_nombre": "La Pampa"
        }
      ],
      "cantidad": 1,
      "total": 2,
    }
  ]
}
```

```

        "inicio": 0
    }
]
}

```

## NORMALIZACIÓN DE DIRECCIONES EN LOTES

```

curl -X POST "https://apis.datos.gob.ar/georef/api/direcciones" \
-H 'Content-Type: application/json' -d'
{
  "direcciones": [
    {
      "direccion": "santa fe 3100",
      "max": 1,
      "campos": "basico"
    },
    {
      "direccion": "corrientes 4010",
      "max": 1,
      "campos": "basico",
      "departamento": "General López"
    }
  ]
}
'

```

### Resultados:

```

{
  "resultados": [
    {
      "cantidad": 1,
      "direcciones": [
        {
          "altura": {
            "valor": "3100"
          },
          "calle": {
            "id": "0642701011435",
            "nombre": "SANTA FE"
          },
          "calle_cruce_1": {
            "id": null,
            "nombre": null
          },
          "calle_cruce_2": {
            "id": null,
            "nombre": null
          },
          "nomenclatura": "SANTA FE 3100, La Matanza, Buenos
Aires"
        }
      ],
      "inicio": 0,
      "total": 29
    },
    {
      "cantidad": 1,
      "direcciones": [
        {
          "altura": {
            "valor": "4010"
          },
          "calle": {
            "id": "8204229000610",

```



```

        "nombre": "CORRIENTES"
    },
    "calle_cruce_1": {
        "id": null,
        "nombre": null
    },
    "calle_cruce_2": {
        "id": null,
        "nombre": null
    },
    "nomenclatura": "CORRIENTES 4010, General López, Sa
nta Fe"
    }
],
"inicio": 0,
"total": 1
}
]
}

```

## ENTIDADES GEOGRÁFICAS EN VARIOS PUNTOS

```

curl -X POST "https://apis.datos.gob.ar/georef/api/ubicacion" \
-H 'Content-Type: application/json' -d'
{
    "ubicaciones": [
        {
            "lat": -27.274161,
            "lon": -66.752929,
            "campos": "completo"
        },
        {
            "lat": -31.480693,
            "lon": -59.092813,
            "aplanar": true,
            "campos": "completo"
        }
    ]
}
'

```

Resultados:

```

{
    "resultados": [
        {
            "ubicacion": {
                "departamento": {
                    "fuente": "Adm. Grl. de Catastro",
                    "id": "10035",
                    "nombre": "Belén"
                },
                "lat": -27.274161,
                "lon": -66.752929,
                "municipio": {
                    "fuente": "Adm. Grl. de Catastro",
                    "id": "100077",
                    "nombre": "Hualfín"
                },
                "provincia": {
                    "fuente": "IGN",
                    "id": "10",
                    "nombre": "Catamarca"
                }
            }
        }
    ]
}

```

```

    },
    {
      "ubicacion": {
        "departamento_fuente": "ATER - Direc. de Catastro",
        "departamento_id": "30113",
        "departamento_nombre": "Villaguay",
        "lat": -31.480693,
        "lon": -59.092813,
        "municipio_fuente": null,
        "municipio_id": null,
        "municipio_nombre": null,
        "provincia_fuente": "IGN",
        "provincia_id": "30",
        "provincia_nombre": "Entre Ríos"
      }
    }
  ]
}

```

# NORMALIZACIÓN DE DIRECCIONES

El recurso `/direcciones` de la API permite normalizar y georreferenciar direcciones de calles. Como todos los otros recursos, cuenta con varios filtros y opciones que permiten controlar los resultados obtenidos.

## PARÁMETROS Y FILTROS

El único parámetro obligatorio del recurso `/direcciones` es `direccion`. El mismo debe tomar el valor de una dirección: es decir, una combinación de nombres de calles y una altura preferiblemente numérica. La API tolera direcciones con distintas estructuras, y se hace un esfuerzo en intentar interpretar qué información representa cada parte del valor recibido, teniendo en cuenta errores de escritura comunes. Para lograr esto, se utiliza la librería `georef-ar-address`. En algunos casos, la estructura de la dirección no puede ser interpretada correctamente; para evitar estos casos se recomienda utilizar direcciones con el siguiente formato aproximado:

- **calle** [altura]
- **calle 1** [altura] *esquina/y* **calle 2**
- **calle 1** *esquina/y* **calle 2** [altura]
- **calle 1** [altura] *entre* **calle 2** y **calle 3**
- **calle 1** *entre* **calle 2** y **calle 3** [altura]

En todos los casos, el valor [altura] es opcional, y de estar presente puede ser seguido de un piso/número de departamento.

El resto de los parámetros aceptados por el recurso `/direccion` están listados en la referencia completa de la API. Se recomienda utilizar los parámetros `provincia` y/o `departamento` para obtener resultados más precisos.

## CAMPOS DE RESPUESTA

Al normalizar una dirección, la API devuelve varios campos de datos. Para entender el significado de cada uno, es conveniente utilizar un ejemplo de uso:

GET <https://apis.datos.gob.ar/georef/api/direcciones?direccion=Av. Santa Fe nro 260 2ndo C, entre Santa Rosa y Colón&departamento=capital&provincia=cordoba>

```

{
  "cantidad": 1,
  "direcciones": [

```

```

{
  "altura": {
    "unidad": "nro",
    "valor": "260"
  },
  "calle": {
    "categoria": "AV",
    "id": "1401401002460",
    "nombre": "AV SANTA FE"
  },
  "calle_cruce_1": {
    "categoria": "CALLE",
    "id": "1401401038100",
    "nombre": "SANTA ROSA"
  },
  "calle_cruce_2": {
    "categoria": "AV",
    "id": "1401401002060",
    "nombre": "AV COLON"
  },
  "departamento": {
    "id": "14014",
    "nombre": "Capital"
  },
  "nomenclatura": "AV SANTA FE 260 (ENTRE SANTA ROSA Y AV COL
ON), Capital, Córdoba",
  "piso": null,
  "provincia": {
    "id": "14",
    "nombre": "Córdoba"
  },
  "ubicacion": {
    "lat": -31.4080674840673,
    "lon": -64.20062417513701
  }
},
"inicio": 0,
"total": 1
}

```

Como se puede observar, campos de respuesta estándar son:

- `altura`
  - `unidad` : Unidad de la altura, o prefijo del valor numérico de la misma.
  - `valor` : Valor numérico de la altura.
- `calle` : Propiedades de la primera calle presente en la dirección.
  - `nombre` : Nombre normalizado de la **calle 1** .
  - `id` : ID de la **calle 1** .
  - `categoria` : Tipo de la **calle 1** .
- `calle_cruce_1` : Propiedades de la segunda calle presente en la dirección (*valores opcionales*) .
  - `nombre` : Nombre normalizado de la **calle 2** .
  - `id` : ID de la **calle 2** .
  - `categoria` : Tipo de la **calle 2** .
- `calle_cruce_1` : Propiedades de la tercera calle presente en la dirección (*valores opcionales*) .
  - `nombre` : Nombre normalizado de la **calle 3** .
  - `id` : ID de la **calle 3** .

- `categoria` : Tipo de la **calle 3**.
- `departamento` : Departamento de la **calle 1**.
- `provincia` : Provincia de la **calle 1**.
- `piso` : Piso extraído de la dirección.
- `nomenclatura` : Versión normalizada de la dirección.
- `ubicacion` : Resultados de la georreferenciación de la dirección ( `lon` y `lat` ). Cuando los valores están presentes, representan una **aproximación** de la ubicación de la dirección. Cuando no están presentes, se debe a que los datos indexados en la API no fueron suficientes para obtener un resultado estimativo. La efectividad de la georreferenciación varía de acuerdo a cada región del país.

## NORMALIZACIÓN DE DIRECCIONES POR LOTES

Para normalizar grandes cantidades de direcciones, se recomienda utilizar los recursos de consultas por lotes.

# OPERACIONES CON GEOMETRÍAS

La API permite a los usuarios operar con las geometrías de distintas entidades geográficas. A continuación, se detallan los recursos y parámetros que permiten a los usuarios realizar estas operaciones.

## PARÁMETRO INTERSECCION

Los recursos `/provincias`, `/departamentos`, `/municipios` y `/calles` cuentan con el parámetro `interseccion`. El parámetro permite buscar entidades utilizando intersección de geometrías como filtro. El parámetro debe tomar valores con el siguiente formato:

```
interseccion=<tipo de entidad>:<id 1>[:<id 2>:...]
```

Al aplicar el filtro `interseccion`, se buscan entidades que compartan área con cualquiera de las entidades listadas en la lista de IDs. Entonces, utilizar (por ejemplo) `/municipios?`

`interseccion=departamento:18105` buscaría todos los municipios que interseccionen con el departamento con ID 18105, mientras que utilizar `/departamentos?`

`interseccion=municipio:620133:540378` buscaría todos los departamentos que interseccionen con el municipio con ID 620133 o el municipio con ID 540378.

De la misma forma, utilizar `/calles?interseccion=municipio:620133` buscaría todas las calles que estén contenidas en el municipio con ID 620133. También es posible buscar municipios a partir de una calle: `/municipios?interseccion=calle:0638503000235` buscaría el municipio que contiene a la calle con ID 0638503000235.

IDs inválidos

Todos los IDs listados que no correspondan a una entidad geográfica existente serán ignorados.

Ejemplo completo de llamado a la API:

```
GET https://apis.datos.gob.ar/georef/api/municipios?interseccion=departamento:18105
```

```
{
  "municipios": [
    {
      "centroide": {
        "lat": -28.508559,
        "lon": -58.031593
      },
      "id": "180042",
      "nombre": "Concepción",
      "provincia": {
```

```

        "id": "18",
        "nombre": "Corrientes"
      }
    },
    { ... } // 9 resultados omitidos
  ],
  "cantidad": 10,
  "total": 13,
  "inicio": 0
}

```

Aunque el recurso `/provincias` acepta el parámetro `interseccion`, se recomienda utilizar filtros por IDs y no por geometrías al momento de buscar provincias, ya que el filtrado por ID es más performante. Por ejemplo, la consulta `/provincias?interseccion=departamento:18105` es equivalente a la consulta `/provincias?id=18`, ya que los primeros dos dígitos de los códigos de los departamentos siempre corresponden a su provincia.

## RECURSO /UBICACION

En la sección de inicio, se dió un ejemplo de uso del recurso `/ubicacion` para enriquecer datos existentes. El recurso utiliza las geometrías de las entidades geográficas para determinar cuáles contienen al punto especificado por el usuario a través de los parámetros `lat` y `lon`. Las entidades devueltas son las siguientes:

- Provincia
- Departamento
- Municipio (*opcional*)

Dependiendo del punto elegido, es posible no obtener un municipio como parte de la respuesta de la API. Como ejemplo, se muestran dos llamados distintos al recurso `/ubicacion`.

Con municipio:

```
GET https://apis.datos.gob.ar/georef/api/ubicacion?lat=-27.2741&lon=-66.7529
```

```

{
  "ubicacion": {
    "departamento": {
      "id": "10035",
      "nombre": "Belén"
    },
    "lat": -27.2741,
    "lon": -66.7529,
    "municipio": {
      "id": "100077",
      "nombre": "Hualfín"
    },
    "provincia": {
      "id": "10",
      "nombre": "Catamarca"
    }
  }
}

```

Sin municipio:

```
GET https://apis.datos.gob.ar/georef/api/ubicacion?lat=-28.504&lon=-62.898
```

```

{
  "ubicacion": {
    "departamento": {
      "id": "86028",
      "nombre": "Aveellaneda"
    },
    "lat": -28.504,
    "lon": -62.898,
  }
}

```

```
{
  "municipio": {
    "id": null,
    "nombre": null
  },
  "provincia": {
    "id": "86",
    "nombre": "Santiago del Estero"
  }
}
```

# DESCARGA DE GEOMETRÍAS

La API permite la descarga de geometrías a través del formato ESRI Shapefile . Para utilizar el formato, se debe agregar `formato=shp` a la lista de parámetros especificados en la URL. El formato Shapefile está disponible en los siguientes recursos (versión GET):

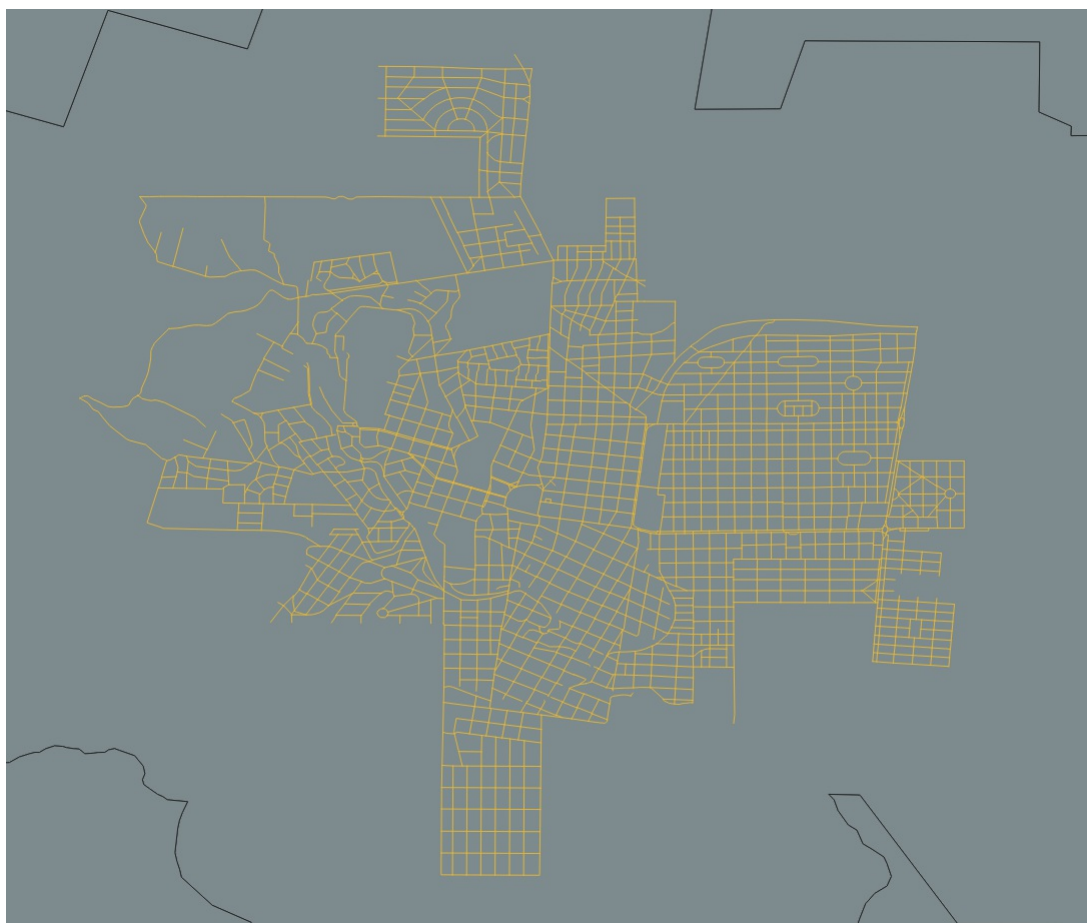
- `/provincias`
- `/departamentos`
- `/municipios`
- `/localidades`
- `/calles`

Cuando se especifica `formato=shp` , la respuesta de la API es un archivo ZIP que contiene los tres archivos requeridos por el estándar: `.shp` , `.shx` y `.dbf` . El archivo luego puede ser abierto con programas como QGIS .

Por ejemplo, si se desea obtener todas las calles del municipio Alta García (ID 141372), se puede utilizar la siguiente consulta:

`GET https://apis.datos.gob.ar/georef/api/calles?`  
`interseccion=municipio:141372&formato=shp&max=1000`

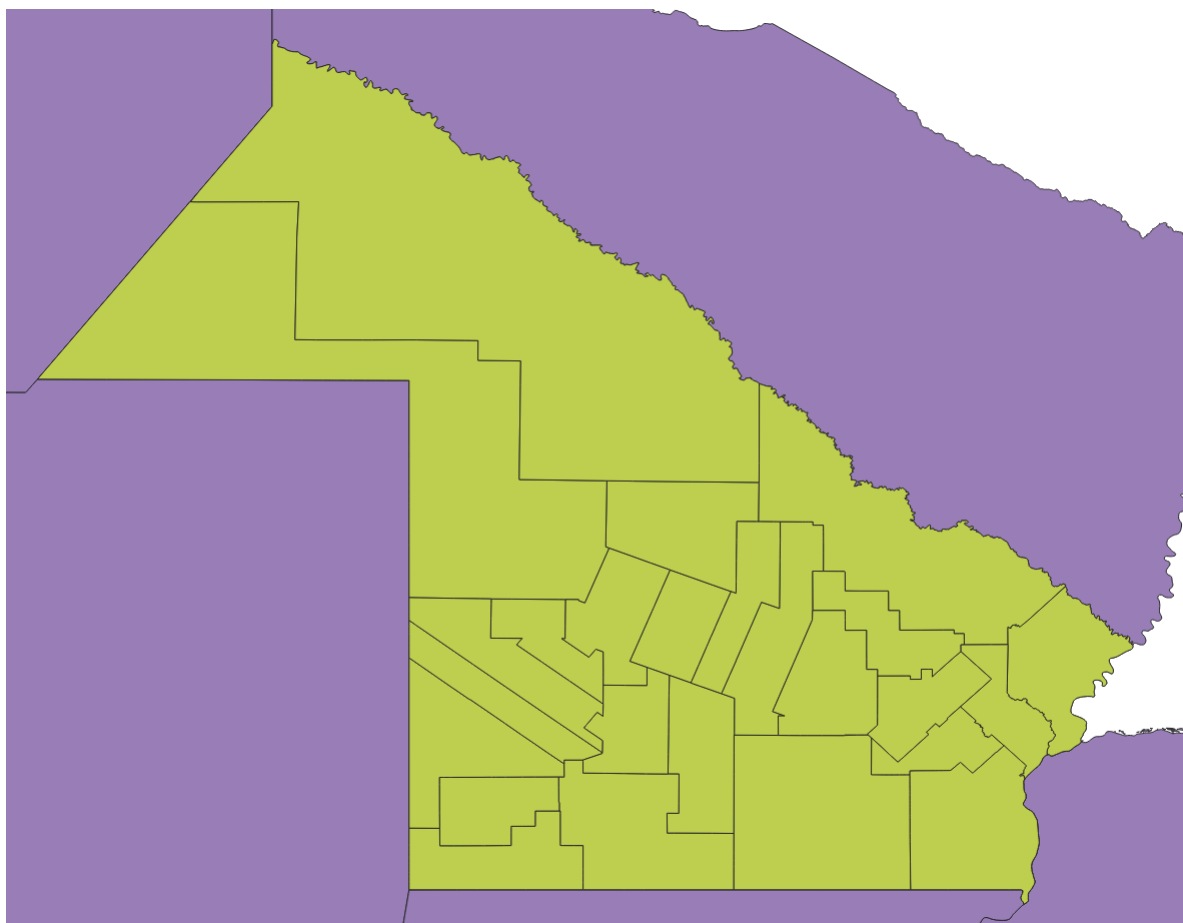
Que descargaría los siguientes datos:



Si se desean descargar todos los departamentos de la provincia de Chaco, se puede utilizar la siguiente consulta:

GET <https://apis.datos.gob.ar/georef/api/departamentos?provincia=chaco&formato=shp&max=1000>

Que descargaría los siguientes datos:

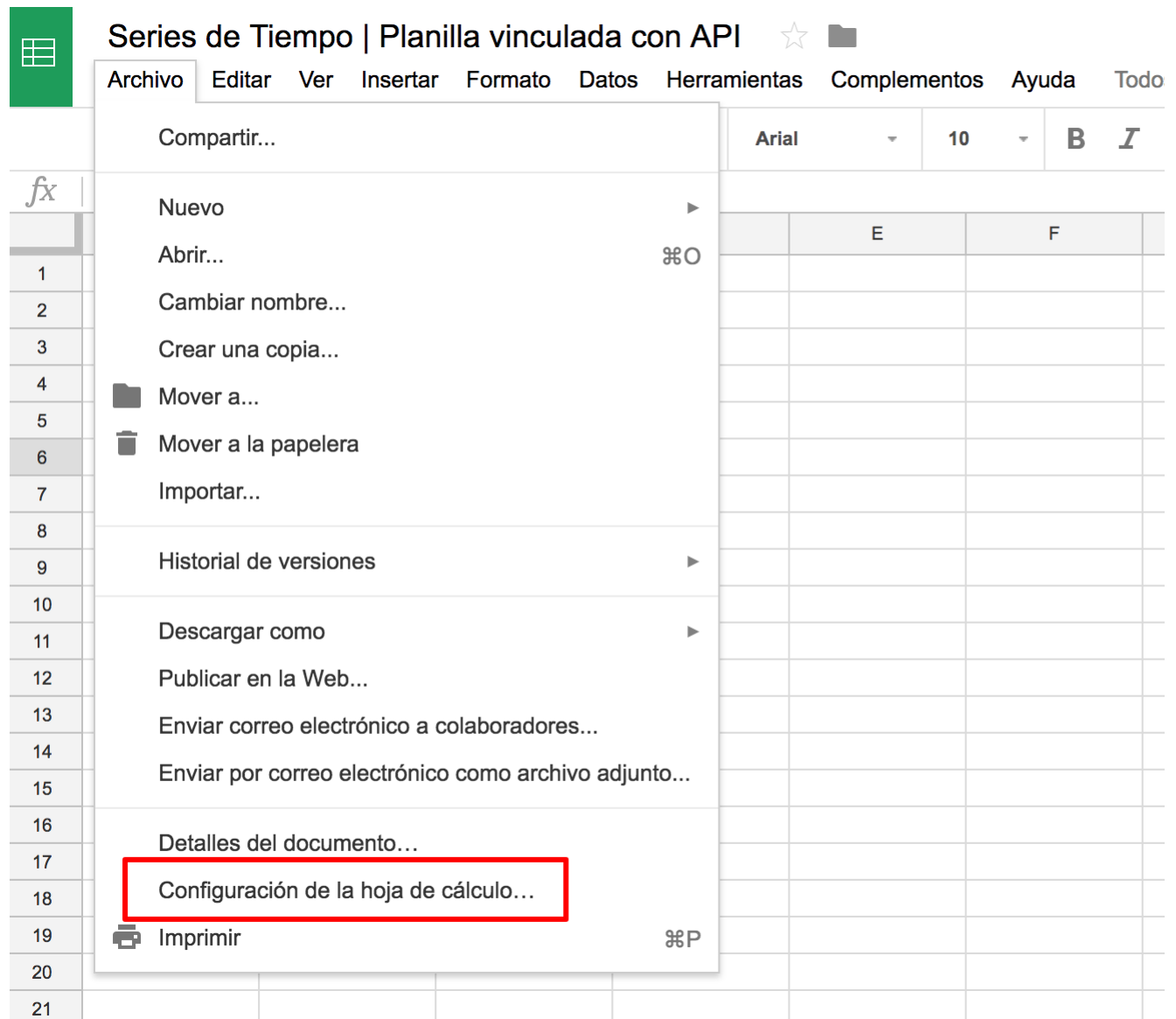


# INTEGRACIÓN CON PLANILLAS DE CÁLCULO

## GOOGLE DRIVE

### 1. MODIFICAR LA CONFIGURACIÓN REGIONAL

La API genera archivos CSV usando “.” como separador decimal. Para que Google Spreadsheet lea correctamente el archivo debe elegirse “Estados Unidos” o cualquier otra región compatible (esto sólo afecta a la lectura de coordenadas).



The screenshot shows the Google Sheets interface for a spreadsheet titled "Series de Tiempo | Planilla vinculada con API". The "Archivo" (File) menu is open, displaying various options. The option "Configuración de la hoja de cálculo..." (Configure spreadsheet...) is highlighted with a red rectangle. Other visible options include "Compartir...", "Nuevo", "Abrir...", "Cambiar nombre...", "Crear una copia...", "Mover a...", "Mover a la papelera", "Importar...", "Historial de versiones", "Descargar como", "Publicar en la Web...", "Enviar correo electrónico a colaboradores...", "Enviar por correo electrónico como archivo adjunto...", "Detalles del documento...", and "Imprimir". The spreadsheet grid is visible in the background, showing columns E and F.



Series de Tiempo | Planilla vinculada con API

ArchivoEditarVerInsertarFormatoDatosHerramientas

100%

€%

.0

.00

123

fx

	A	B	C	D
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				
21				
22				
23				
24				

- Chequia
- Chile
- China
- Colombia
- Corea del Sur
- Croacia
- Dinamarca
- Ecuador
- Egipto
- Eslovaquia
- Eslovenia
- España
- España (catalán)
- Estados Unidos
- Filipinas
- Finlandia
- Francia
- Georgia
- Grecia
- Hong Kong
- Hungría
- India
- India (bengalí)
- India (canarés)



=IMPORTDATA("http://apis.datos.gob.ar/georef/api/localidades?formato=csv&max=1000&provincia=santa fe")							
	A	B	C	D	E	F	G
1	localidad_id	localidad_nombre	localidad_centroid_x	localidad_centroid_y	provincia_id	provincia_nombre	departamento
2	82007020000	BOUQUET	-324248	-61890335	82	Santa Fe	82007020000
3	82007050000	MONTES DE OCA	-32567997	-61768076	82	Santa Fe	82007050000
4	82014080000	GODEKEN	-33402406	-61844931	82	Santa Fe	82014080000
5	82014130000	SANFORD	-33147773	-61277857	82	Santa Fe	82014130000
6	82021070000	BELLA ITALIA	-3128398	-61409343	82	Santa Fe	82021070000
7	82021120000	COLONIA RAQUIL	-30839224	-61489711	82	Santa Fe	82021120000
8	82021130000	CORONEL FRACASTRO	-31176159	-61919462	82	Santa Fe	82021130000
9	82021180000	EUSEBIA Y CARLOS	-30947614	-61857774	82	Santa Fe	82021180000
10	82021200000	FRONTERA	-31431399	-62063492	82	Santa Fe	82021200000
11	82021270000	PLAZA CLUCELI	-31454551	-61707476	82	Santa Fe	82021270000
12	82021280000	PLAZA SAGUIER	-31325105	-61677737	82	Santa Fe	82021280000
13	82021360000	SUNCHALES	-30946856	-6156125	82	Santa Fe	82021360000
14	82021390000	VILA	-31192371	-61833614	82	Santa Fe	82021390000
15	82021410000	VILLA SAN JOSÉ	-31339135	-61622588	82	Santa Fe	82021410000
16	82021430000	ZENON PEREYRA	-31564397	-61898425	82	Santa Fe	82021430000
17	82028010000	ALCORTA	-33540198	-61124634	82	Santa Fe	82028010000
18	82028040000	BOMBAL	-33460062	-61318932	82	Santa Fe	82028040000
19	82028070000	EMPALME VILLA	-33262863	-60380464	82	Santa Fe	82028070000
20	82028100000	GODOY	-3336971	-60509413	82	Santa Fe	82028100000
21	82028110000	JUAN B. MOLINA	-33496021	-60512213	82	Santa Fe	82028110000
22	82028120000	JUNCAL	-33717615	-61050018	82	Santa Fe	82028120000
23	82028210000	STEPHENSON	-33417857	-60557214	82	Santa Fe	82028210000
24	82042040000	CAÑADA DEL UDELAR	-33410344	-61607043	82	Santa Fe	82042040000
25	82042080000	CHOVET	-33600783	-61604665	82	Santa Fe	82042080000
26	82042140000	LA CHISPA	-3354467	-61973632	82	Santa Fe	82042140000
27	82042170000	MAGGIOLO	-33724424	-62247839	82	Santa Fe	82042170000
28	82042180000	MARIA TERESA	-33666665	-61666611	82	Santa Fe	82042180000

3. NORMALIZAR UN LISTADO DE UNIDADES TERRITORIALES

Si tenemos un listado de provincias que queremos normalizar, como el siguiente:

Servicio de normalización de datos geográficos

Archivo

Editar

Ver

Insertar

Formato

Datos

Herramientas

Formulario

100%

\$

%

.0

.00

123

Arial

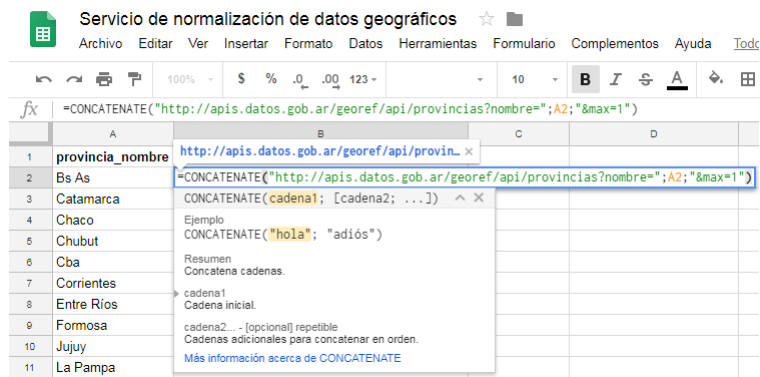
10

provincia\_nombre

	A	B	C
1	provincia_nombre		
2	Bs As		
3	Catamarca		
4	Chaco		
5	Chubut		
6	Cba		
7	Corrientes		
8	Entre Ríos		
9	Formosa		
10	Jujuy		
11	La Pampa		
12	La Rioja		
13	MZ		
14	Misiones		
15	Neuquén		

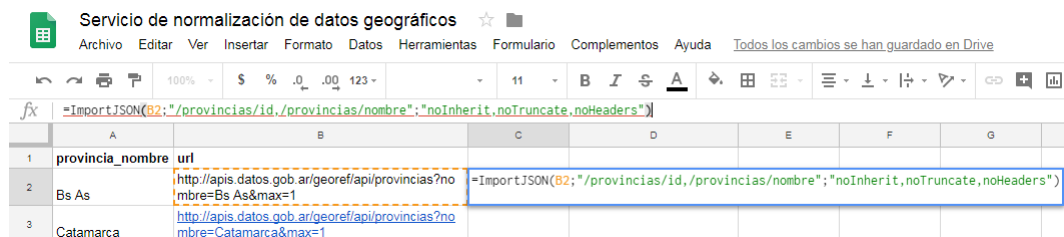
Podemos armar urls individuales para normalizar los nombres y traer alguno de sus atributos. Imaginemos que queremos el ID y el nombre normalizado.

Primero generamos la url para cada una de las provincias:



y luego necesitamos importar una nueva función en la hoja de cálculo. Para eso, desde el menú:

1. Herramientas → Editor de secuencia de comandos.
2. Borramos todo lo que hay en el editor y pegamos el siguiente script de Bradjasper .
3. Renombremos el script como ImportJson.gs y guardamos.
4. Ahora ya podemos usar la función `=importJSON()` en una celda.



`=ImportJSON(B2;"/provincias/id,/provincias/nombre";"noInherit,noTruncate,noHeaders")`

y obtendremos:

provincia_nombre	url	provincia_id	provincia_nombre
Buenos Aires	http://apis.datos.gob.ar/georef/api/provincias?nombre=Buenos Aires&max=1	09	Buenos Aires
Catamarca	http://apis.datos.gob.ar/georef/api/provincias?nombre=Catamarca&max=1	10	Catamarca
Chaco	http://apis.datos.gob.ar/georef/api/provincias?nombre=Chaco&max=1	22	Chaco
Chubut	http://apis.datos.gob.ar/georef/api/provincias?nombre=Chubut&max=1	26	Chubut
Cba	http://apis.datos.gob.ar/georef/api/provincias?nombre=Cba&max=1	14	Córdoba
Corrientes	http://apis.datos.gob.ar/georef/api/provincias?nombre=Corrientes&max=1	18	Corrientes
Entre Ríos	http://apis.datos.gob.ar/georef/api/provincias?nombre=Entre Rios&max=1	30	Entre Ríos
Formosa	http://apis.datos.gob.ar/georef/api/provincias?nombre=Formosa&max=1	34	Formosa

# USAR EN PYTHON

## CON REQUESTS

### NORMALIZAR UNA ENTIDAD

```
import requests
import urllib
```

```
API_BASE_URL = "https://apis.datos.gob.ar/georef/api/"
```

```
def get_similar(endpoint, nombre, **kwargs):
    kwargs["nombre"] = nombre
    url = "{}{}?{}".format(API_BASE_URL, endpoint, urllib.urlencode(kwargs))
    return requests.get(url).json()[endpoint]
```

```
provincias = get_similar("provincias", "San Juan")
```

```
[{
    u'centroide': {
        u'lat': -30.865368,
        u'lon': -68.889491
    },
    u'id': u'70',
    u'nombre': u'San Juan'
}]
```

## NORMALIZAR VARIAS ENTIDADES

```
def get_similar_bulk(endpoint, nombres):
    """Normaliza una lista de nombres de alguna de las entidades geográficas."""
```

```
    # realiza consulta a la API
    data = {
        endpoint: [
            {"nombre": nombre, "max": 1} for nombre in nombres
        ]
    }
    url = API_BASE_URL + endpoint
    results = requests.post(
        url, json=data, headers={"Content-Type": "application/json"}
    ).json()

    # convierte a una lista de "resultado más probable" o "vacío" cuando no hay
    parsed_results = [
        single_result[endpoint][0] if single_result[endpoint] else {}
        for single_result in results["resultados"]
    ]

    return parsed_results
```

```
provincias = get_similar_bulk("provincias", ["pxa", "sant fe"])
```

```
[
    {},
    {
        u'centroide': {
            u'lat': -30.706927,
            u'lon': -60.949837
        },
        u'id': u'82',
        u'nombre': u'Santa Fe'
    }
]
```

## ENRIQUECER COORDENADAS CON LAS UNIDADES TERRITORIALES QUE LAS CONTIENEN

```
def get_territorial_units(ubicaciones):
    """Pide las unidades territoriales que contienen a c/punto de una lista de coordenadas."""
```

```
    # realiza consulta a la API
    endpoint = "ubicacion"
    data = {
        "ubicaciones": [
            {"lat": ubicacion["lat"], "lon": ubicacion["lon"], "aplanar": True}
            for ubicacion in ubicaciones
        ]
    }
```

```

}}
url = API_BASE_URL + endpoint

results = requests.post(
    url, json=data, headers={"Content-Type": "application/json"})
.json()

# convierte a una lista de "resultado más probable" o "vacío" cuando
o no hay
parsed_results = [
    single_result[endpoint] if single_result[endpoint] else {}
    for single_result in results["resultados"]
]

return parsed_results

ubicaciones = get_territorial_units([
    {"lat": -32.9477132, "lon": -60.6304658},
    {"lat": -34.6037389, "lon": -58.3815704}
])

[
    {
        u'departamento_id': u'30105',
        u'departamento_nombre': u'Victoria',
        u'lat': -32.9477132,
        u'lon': -60.6304658,
        u'municipio_id': u'82210',
        u'municipio_nombre': u'Rosario',
        u'provincia_id': u'30',
        u'provincia_nombre': u'Entre Ríos'
    },
    {
        u'departamento_id': u'02007',
        u'departamento_nombre': u'Comuna 1',
        u'lat': -34.6037389,
        u'lon': -58.3815704,
        u'municipio_id': None,
        u'municipio_nombre': None,
        u'provincia_id': u'02',
        u'provincia_nombre': u'Ciudad Autónoma de Buenos Aires'
    }
]

```

## CON PANDAS

### CONSULTAR LISTAS DE REFERENCIA

Todas las consultas a la API en formato CSV, se pueden leer fácilmente a un `pandas.DataFrame`. De ahí se pueden tomar listas de referencia para distintas unidades territoriales.

```
import pandas as pd
```

```
provincias = pd.read_csv("https://apis.datos.gob.ar/georef/api/provincias?formato=csv")
```

provincia_id	provincia_nombre
14	Córdoba
22	Chaco
26	Chubut
6	Buenos Aires
10	Catamarca
30	Entre Ríos
34	Formosa
42	La Pampa

62	Río Negro
70	San Juan
78	Santa Cruz
82	Santa Fe
94	Tierra del Fuego, Antártida e Islas del Atlánt...
38	Jujuy
54	Misiones
2	Ciudad Autónoma de Buenos Aires
18	Corrientes
46	La Rioja
66	Salta
86	Santiago del Estero
50	Mendoza
58	Neuquén
74	San Luis
90	Tucumán

## ENRIQUECER COORDENADAS

```
def add_territorial_units(df, column_lat, column_lon):
    """Agrega unidades territoriales que contienen coordenadas a un DataFrame.
```

Args:

```
df (pandas.DataFrame): Un DataFrame que tiene coordenadas.
column_lat (str): Nombre de la columna que tiene latitud.
column_lon (str): Nombre de la columna que tiene longitud.
```

Returns:

```
pandas.DataFrame: DataFrame original aumentado con unidades territoriales que contienen a las coordenadas.
"""
```

```
# toma una lista de coordenadas únicas (no repetidas)
coordinates = df[[column_lon, column_lat]].rename(
    columns={column_lon: "lon", column_lat: "lat"}
).drop_duplicates().to_dict("records")
```

```
# crea DataFrame de unidades territoriales que contienen a las coordenadas
ubicaciones = pd.DataFrame(get_territorial_units(coordinates))
```

```
# agrega las unidades territoriales al DataFrame original
df_with_territorial_units = df.merge(
    ubicaciones, "left",
    left_on=[column_lon, column_lat],
    right_on=["lon", "lat"]
)
```

```
# elimina columnas de coordenadas repetidas, dejando las originales
return df_with_territorial_units.drop(["lon", "lat"], axis=1)
```

```
# descarga un CSV con coordenadas de aeropuertos
df = pd.read_csv("https://servicios.transporte.gob.ar/gobierno_abierto/
descargar.php?t=aeropuertos&d=detalle", sep=";")
```

```
# Agrega unidades territoriales que contienen coordenadas a un DataFrame
df_with_territorial_units = add_territorial_units(df, "longitud", "latitud")
```

tipo	denominacion	latitud	longitud	elev
Aeródromo	CORONEL BOGADO/AGROSERVICIOS	-60.57066	-33.27226	44.0
Aeródromo	GENERAL ACHA	-64.61351	-37.40164	277.0

Aeródromo	ARRECIFES/LA CURA MALAL	-60.14170	-34.07574	37.0
Aeródromo	PUERTO DESEADO	-65.90410	-47.73511	82.0
Aeródromo	BANDERA/AGROSERVICIOS DOÑA TERESA	-62.26462	-28.85541	75.0

departamento_id	departamento_nombre	municipio_id	municipio_nombre	\
82084	Rosario	823393	Coronel Bogado	
42154	Utracán	420133	General Acha	
06077	Arrecifes	060077	Arrecifes	
78014	Deseado	None	None	
86077	General Taboada	None	None	

provincia_id	provincia_nombre
82	Santa Fe
42	La Pampa
06	Buenos Aires
78	Santa Cruz
86	Santiago del Estero

# TOKENS JWT

Si pertenecés a un organismo de la Administración Pública Nacional y querés incrementar la cuota de uso de la API de Georef, podés pedir un token y autenticarte utilizando JWT .

Para generar un token JWT, se requieren dos elementos: una *key* y un *secret* generados para el uso con la API.

Una vez obtenidos ambos elementos, se puede generar un token JWT utilizando, por ejemplo, Python o Node.js. A continuación, se muestran ejemplos utilizando los siguientes valores demostrativos:

- `key = YXNkc2Rhc2RmYXNkZmFzZmRhc2RmYXNk`
- `secret = dnVvODY4Yzc2bzhzNzZqOG83czY4b2Nq`

El algoritmo de autenticación de mensajes con *hash* utilizado es HMAC-SHA256 ( `HS256` ).

## PYTHON

Utilizando la librería `pyjwt` :

```
$ pip install pyjwt
$ python

>>> import jwt
>>> key = 'YXNkc2Rhc2RmYXNkZmFzZmRhc2RmYXNk'
>>> message = { 'iss': key }
>>> secret = 'dnVvODY4Yzc2bzhzNzZqOG83czY4b2Nq'
>>> token_bytes = jwt.encode(message, secret, algorithm='HS256')
>>> token = token_bytes.decode()
>>> token
'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJZWWE5rYzJSaGMyUm1ZWWE5rWm1GelptUmhjMlJtWVhOayJ9.P4leoe9q_H3lmIlnpZuVFSt7ORgLhLfQ3JN_3FMexSo'
```

Finalmente, para consumir la API de Georef, adjuntar el token generado en las cabeceras HTTP. A continuación, se muestra un ejemplo utilizando la librería `requests` :

```
>>> import requests
>>> headers = { 'Authorization': 'Bearer {}'.format(token) }
>>> resp = requests.get('https://apis.datos.gob.ar/georef/api/provincias', headers=headers)
>>> resp.json()
{
  'provincias': [
    { ... }
  ]
}
```



```
}
```

## NODE.JS

Utilizando la librería `jsonwebtoken` :

```
$ npm install jsonwebtoken
$ node

> var jwt = require('jsonwebtoken')
> var payload = { 'iss': 'YXNkc2Rhc2RmYXNkZmFzZmRhc2RmYXNk' }
> var secret = 'dnVvODY4Yzc2bzhzNzZqOG83czY4b2Nq'
> var token = jwt.sign(payload, secret, { 'noTimestamp': true })
> token
'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJZWE5rYzJSaGMyUm1ZWE5rWm1GelptUmhjMlJtWVh0ayJ9.P4leoe9q_H3lmIlnpZuVFSt7ORgLhLfQ3JN_3FMexSo'
```

Finalmente, para consumir la API de Georef, adjuntar el token generado en las cabeceras HTTP:

```
> var http = require('http')
> http.get({
  'hostname': 'apis.datos.gob.ar',
  'path': '/georef/api/provincias',
  'headers': {
    'authorization': 'Bearer ' + token
  }
}, function(response) {
  ...
})
```

# CONDICIONES DE USO

## LEGALES

Los datos del Servicio de Normalización de Datos Geográficos de Argentina se disponibilizan bajo la licencia Creative Commons Attribution 4.0 y pueden ser usados para cualquier fin, incluyendo fines comerciales.

## TÉCNICAS

La API del servicio de normalización se encuentra en estado *alpha* de desarrollo y las cuotas de uso abierto sin autenticación no están definidas.

A medida que el servicio madure en su desarrollo y en el uso que se le da, se establecerán cuotas explícitas de uso sin autenticación por IP.

Los organismos de la APN que requieran cuotas de uso más elevadas que las del servicio abierto, pueden solicitar un token a [datos@modernizacion.gob.ar](mailto:datos@modernizacion.gob.ar) .

# APLICACIONES

Lista de proyectos y aplicaciones que utilizan, extienden o documentan la API del Servicio de Normalización de Datos Geográficos.

## DE TERCEROS

- [pdelboca/georefar](#) : Interface en R para usar la API por [@pdelboca](#) .
- [juancarlospaco/nim-georefar](#) : Interface en Nim para usar la API por [juancarlospaco](#) (GitHub) .

## CONTACTO

¿Desarrollaste una librería o tenés un proyecto que usa la API? Te invitamos a contarnos así la agregamos a esta lista.

Para todo lo demás, podés mandarnos tu comentario o consulta a [datos@modernizacion.gob.ar](mailto:datos@modernizacion.gob.ar).

# GUÍA DE INSTALACIÓN PARA DESARROLLADORES

En este documento se detallan los pasos a seguir si se desea configurar un servidor de API Georef propio.

## DEPENDENCIAS

- Elasticsearch >=6.4.2
- Python >=3.6.x
- Nginx (*para entornos productivos*)

## INSTALACIÓN

### 1. ELASTICSEARCH

Para instalar Elasticsearch, seguir las siguientes instrucciones en uno o más servidores (nodos).

#### 1.1 Instalar el entorno de ejecución para Java:

```
$ sudo apt install default-jre
```

#### 1.2 Instalar Elasticsearch

Instalar Elasticsearch e iniciar el servicio con `systemctl`:

```
$ wget https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-6.4.2.deb
$ sudo dpkg -i elasticsearch-6.4.2.deb
$ sudo systemctl enable elasticsearch
```

#### 1.3 Aplicar las configuraciones recomendadas

Editar el archivo `/etc/elasticsearch/elasticsearch.yml` (el valor de `node.name` debe ser único por nodo):

```
node.name: node-1
http.max_content_length: 100mb
```

Editar el archivo `/etc/elasticsearch/jvm.options` (siguiendo las recomendaciones de Elasticsearch):

```
# Tamaño del heap size de la JVM
# Se recomienda utilizar siempre el mismo valor como mínimo y máximo
# Ejemplo: 4 GB
-Xms4g
-Xmx4g
```

## 2. REPOSITORIO Y DEPENDENCIAS

### 2.1 Clonar el repositorio:

```
$ git clone https://github.com/datosgobar/georef-ar-api.git
$ cd georef-ar-api
```

## 2.2 Crear un entorno virtual y activarlo

```
$ python3 -m venv venv
$ source venv/bin/activate
```

## 2.3 Instalar dependencias con pip

```
(venv) $ pip3 install -r requirements.txt -r requirements-dev.txt
```

## 2.4 Copiar el archivo de configuración

```
(venv) $ cp config/georef.example.cfg config/georef.cfg
```

Luego, completar el archivo `config/georef.cfg` con los valores apropiados.

## 2.5 Copiar el archivo de configuración de logs

```
(venv) $ cp config/logging.example.ini config/logging.ini
```

Luego, completar el archivo `config/logging.ini` con los valores apropiados. Los valores por defecto son válidos y pueden ser utilizados en entornos productivos.

## 3. CREAR LOS ÍNDICES

Generar índices de entidades y calles:

```
(venv) $ make index
```

Listar los índices creados, y otros datos adicionales:

```
(venv) $ make print_index_stats
```

## 4. (OPCIONAL) RE-INDEXAR DATOS

Si se modifican los archivos de datos JSON, es posible re-indexarlos sin borrar los índices ya existentes. Dependiendo del comportamiento que se desee, se debe tomar una opción:

### Indexar datos nuevos

Si se desea actualizar los índices con los nuevos datos, solo si los datos entrantes son más recientes, se puede utilizar nuevamente:

```
(venv) $ make index
```

### Forzar re-indexado

Si se desea forzar un re-indexado, es decir, si se desea indexar los datos nuevamente sin importar la fecha de creación, se debe utilizar la siguiente receta:

```
(venv) $ make index_forced
```

La receta `index_forced` intenta utilizar un archivo de respaldo guardado anteriormente si no pudo acceder a los archivos especificados en `config/georef.cfg`. El uso de la receta es recomendado cuando se requiere re-indexar los datos incondicionalmente, algunas situaciones donde esto es necesario son:

- Modificación de la estructura de los archivos de datos
- Modificación de *mappeos* de tipos de Elasticsearch
- Modificación de analizadores de texto de Elasticsearch
- Modificación de listado de sinónimos

- Modificación de listado de términos excluyentes

Cualquiera de las dos opciones también permite indexar datos selectivamente: se debe especificar el nombre del índice a crear/re-indexar. Por ejemplo:

```
(venv) $ make index INDEX_NAME=localidades
(venv) $ make index_forced INDEX_NAME=calles
```

Los nombres de los índices disponibles son:

- provincias
- provincias-geometria
- departamentos
- departamentos-geometria
- municipios
- municipios-geometria
- localidades
- calles
- intersecciones

## 5. CORRER API

### Entornos de desarrollo

Correr la API de Georef utilizando un servidor de prueba (no apto para producción):

```
(venv) $ make start_dev_server
```

O También:

```
(venv) $ make start_gunicorn_dev_server
```

### Entornos productivos

#### 5.1 Configurar servicio georef-ar-api para systemd

Copiar el archivo `config/georef-ar-api.service` a `/etc/systemd/system/` y configurarlo. Notar los campos marcados entre '`<`' y '`>`', que deben ser reemplazados por el usuario.

#### 5.2 Activar y arrancar el servicio

```
$ sudo systemctl daemon-reload
$ sudo systemctl enable georef-ar-api.service
$ sudo systemctl start georef-ar-api.service
```

#### 5.3 Configurar nginx

Primero, crear `/etc/nginx/sites-available/georef-ar-api.nginx` tomando como base la configuración del archivo `georef-ar-api.nginx`.

#### 5.4 (Opcional) Crear cache para nginx

Si se desea activar el uso del cache de `nginx`, descomentar las líneas contentiendo las directivas `proxy_cache` y `proxy_cache_valid` del archivo `georef-ar-api.nginx` creado. Luego, activar el cache `georef` agregando la siguiente línea al archivo de configuración `nginx.conf` (sección `http`):

```
proxy_cache_path /data/nginx/cache levels=1:2 inactive=120m keys_zone=georef:10m use_temp_path=off;
```

Finalmente, crear el directorio `/data/nginx/cache`.

## 5.5 Activar y validar configuración `nginx`

Generar un link simbólico a la configuración del sitio:

```
$ sudo ln -s /etc/nginx/sites-available/georef-ar-api.nginx /etc/nginx/sites-enabled/georef-ar-api.nginx
```

Validar la configuración:

```
$ sudo nginx -T
```

Reiniciar Nginx:

```
$ systemctl restart nginx.service
```

## TESTS

Para ejecutar los tests unitarios (el servicio Elasticsearch debe estar activo y con los datos apropiados cargados):

```
(venv) $ make test
```

Para más información sobre los tests, ver el archivo `tests/README.md`.

Para comprobar que no existan errores comunes en el código, y que su estilo sea correcto:

```
(venv) $ make code_checks
```

## ARCHIVOS DE DATOS

- 

La estructura de los archivos de datos JSON utilizados por Georef está detallada en este documento.

- 

El archivo de configuración `config/georef.cfg` debe especificar una ruta local o una URL externa para cada archivo de datos JSON. Notar que los valores por defecto (en `georef.example.cfg`) utilizan el portal de descargas `infra.datos.gob.ar`, que siempre provee la última versión de los archivos JSON disponibles. La rama `master` de `georef-ar-api` siempre se mantiene compatible con la última versión de los datos disponibles en `infra.datos.gob.ar`.

- 

El archivo de configuración `config/georef.cfg` también debe especificar la URL del archivo de sinónimos para utilizar al momento de indexar campos de texto en Elasticsearch. El valor por defecto en `georef.example.cfg` puede ser utilizado, ya que utiliza la versión del archivo almacenado en `infra.datos.gob.ar`. El mismo criterio se aplica al archivo de términos excluyentes.

# MODELO DE DATOS PARA GEOREF API

Los archivos de datos de Georef consisten de seis (6) archivos en formato JSON, los cuales contienen provincias, departamentos, municipios, localidades, calles e intersecciones de calles.

## ETL

Los datos utilizados por Georef API son obtenidos a través de un proceso de ETL. El código del mismo se encuentra en el repositorio GitHub `georef-ar-etl`, así también como su guía de instalación y uso.

## FUENTES

Los orígenes de los datos procesados en el ETL son:

## UNIDADES TERRITORIALES

- Recursos: /provincias , /departamentos , /municipios , /ubicacion
- Fuente: **Instituto Geográfico Nacional (IGN)**
- Enlace: Datos Abiertos - Unidades Territoriales

## BAHRA

- Recursos: /localidades
- Fuente: **Base de Asentamientos Humanos de la República Argentina (BAHRA)**
- Enlace: BAHRA - Descargas

## VÍAS DE CIRCULACIÓN

- Recursos: /calles , /direcciones
- Fuente: **Instituto Nacional de Estadística y Censos de la República Argentina (INDEC)**
- Enlace: Portal de geoservicios de INDEC

## ARCHIVOS

A continuación se detallan, a través de ejemplos, los esquemas de los archivos para las entidades utilizadas. Notar que el campo `version` se utiliza al momento de indexar para determinar si los datos son compatibles con la versión de la API siendo utilizada; la versión detallada en este documento es la `9.0.0`.

## PROVINCIAS

El archivo de datos de provincias debe tener formato JSON. Su esquema de datos debe ser el siguiente:

```
{
  "timestamp": "1532435389", // Timestamp de creación
  "fecha_creacion": "2018-07-24 12:29:49.813835+00:00", // Fecha de c
reación
  "version": "9.0.0", // Versión de archivo
  "datos": [ // Lista de entidades
    {
      "id": "90", // ID de provincia
      "nombre": "Tucumán", // Nombre de provincia,
      "nombre_completo": "Provincia de Tucumán", // Nombre comple
to
      "iso_id": "AR-T", // Identificador ISO 3166-2
      "iso_nombre": "Tucumán", // Nombre ISO
      "categoria": "Provincia", // Tipo de entidad
      "centroide": {
        "lat": -26.9478, // Latitud de centroide
        "lon": -65.36475 // Longitud de centroide
      },
      "geometria": { // Geometría en formato GeoJSON
        "type": "MultiPolygon",
        "coordinates": [[[-58.4549, -34.5351], [-58.4545, -34.
5353], ...]]]
      },
      "fuente": "IGN" // Fuente del dato
    },
    { ... },
  ]
}
```

## DEPARTAMENTOS

El archivo de datos de departamentos debe tener formato JSON. Su esquema de datos debe ser el siguiente:

```
{
  "timestamp": "1532435389", // Timestamp de creación
  "fecha_creacion": "2018-07-24 12:29:49.813835+00:00", // Fecha de c
reación
  "version": "9.0.0", // Versión de archivo
  "datos": [ // Lista de entidades
    {
      "id": "06427", // ID del departamento
      "nombre": "La Matanza", // Nombre del departamento
      "nombre_completo": "Partido de la Matanza", // Nombre compl
eto
      "categoria": "Partido", // Tipo de entidad
      "centroide": {
        "lat": -34.770165, // Latitud de centroide
        "lon": -58.625449 // Longitud de centroide
      },
      "geometria": { // Geometría en formato GeoJSON
        "type": "MultiPolygon",
        "coordinates": [[[-58.4549, -34.5351], [-58.4545, -34.
5353], ...]]]
      },
      "provincia": { // Provincia que contiene al departamento
        "id": "06",
        "nombre": "Buenos Aires",
        "interseccion": "0.0412936" // Porcentaje del área de l
a provincia que ocupa el depto.
      },
      "fuente": "ARBA - Gerencia de Servicios Catastrales" // Fue
nte del dato
    },
    { ... },
  ]
}
```

## MUNICIPIOS

El archivo de datos de municipios debe tener formato JSON. Su esquema de datos debe ser el siguiente:

```
{
  "timestamp": "1532435389", // Timestamp de creación
  "fecha_creacion": "2018-07-24 12:29:49.813835+00:00", // Fecha de c
reación
  "version": "9.0.0", // Versión de archivo
  "datos": [ // Lista de entidades
    {
      "id": "060105", // ID del municipio
      "nombre": "Bolívar", // Nombre del municipio
      "nombre_completo": "Municipio Bolívar", // Nombre completo
      "categoria": "Municipio", // Tipo de entidad
      "centroide": {
        "lat": -36.298222, // Latitud de centroide
        "lon": -61.149648 // Longitud de centroide
      },
      "geometria": { // Geometría en formato GeoJSON
        "type": "MultiPolygon",
        "coordinates": [[[-58.4453, -34.4324], [-58.6463, -34.
6841], ...]]]
      },
      "provincia": { // Provincia que contiene al municipio
        "id": "06",
        "nombre": "Buenos Aires",
        "interseccion": "0.0100845" // Porcentaje del área de l

```

```

a provincia que ocupa el municipio
    },
    "fuente": "ARBA - Gerencia de Servicios Catastrales" // Fuente del dato
    },
    { ... },
]
}

```

## LOCALIDADES

El archivo de datos de localidades debe tener formato JSON. Su esquema de datos debe ser el siguiente:

```

{
  "timestamp": "1532435389", // Timestamp de creación
  "fecha_creacion": "2018-07-24 12:29:49.813835+00:00", // Fecha de creación
  "version": "9.0.0", // Versión de archivo
  "datos": [ // Lista de entidades
    {
      "id": "06189080000", // ID de la localidad
      "nombre": "San Roman", // Nombre de la localidad
      "categoria": "Localidad simple (LS)", // Tipo de asentamiento
      "centroide": {
        "lat": -38.741555, // Latitud de centroide
        "lon": -61.537720 // Longitud de centroide
      },
      "geometria": { // Geometría en formato GeoJSON
        "type": "MultiPoint",
        "coordinates": [[-61.5377, -38.7415], ...]
      },
      "municipio": { // Municipio que contiene a la localidad
        "id": "060189", // Puede ser nulo
        "nombre": "Coronel Dorrego" // Puede ser nulo
      },
      "departamento": { // Departamento que contiene a la localidad
        "id": "06189",
        "nombre": "Coronel Dorrego"
      },
      "provincia": { // Provincia que contiene a la localidad
        "id": "06",
        "nombre": "Buenos Aires"
      },
      "fuente": "INDEC" // Fuente del dato
    },
    { ... },
  ]
}

```

## CALLES

El archivo de datos de calles debe tener formato JSON. Su esquema de datos debe ser el siguiente:

```

{
  "timestamp": "1532435389", // Timestamp de creación
  "fecha_creacion": "2018-07-24 12:29:49.813835+00:00", // Fecha de creación
  "version": "9.0.0", // Versión de archivo
  "datos": [ // Lista de vías de circulación
    {
      "nomenclatura": "LARREA, Comuna 3, Ciudad Autónoma de Buenos Aires", // Nomenclatura: 'nombre, departamento, provincia'
      "id": "0202101007345", // ID de la vía de circulación
    }
  ]
}

```



```

        "nombre": "LARREA", // Nombre de vía de circulación
        "categoria": "CALLE", // Tipo de vía de circulación
        "altura": {
            "inicio": {
                "derecha": 1, // Número inicial de altura (lado der
echo)
                "izquierda": 2, // Número inicial de altura (lado i
zquierdo)
            },
            "fin": {
                "derecha": 799, // Número final de altura (lado der
echo)
                "izquierda": 800, // Número final de altura (lado i
zquierdo)
            }
        },
        "geometria": { // Geometría en formato GeoJSON
            "type": "MultiLineString",
            "coordinates": [[[-58.52815846522327, -34.6118003976374
24], ...]]
        },
        "departamento": { // Departamento
            "nombre": "Comuna 3",
            "id": "02021"
        },
        "provincia": { // Provincia
            "nombre": "Ciudad Autónoma de Buenos Aires",
            "id": "02"
        },
        "fuente": "INDEC" // Fuente del dato
    },
    { ... },
]
}

```

## INTERSECCIONES DE CALLES

El archivo de datos de intersecciones debe tener formato JSON, y no debe tener intersecciones repetidas. Dadas las calles con ID X y ID Z, solo debe estar presente la intersección X-Z o Z-X. Su esquema de datos debe ser el siguiente:

```

{
    "timestamp": "1532435389", // Timestamp de creación
    "fecha_creacion": "2018-07-24 12:29:49.813835+00:00", // Fecha de c
reación
    "version": "9.0.0", // Versión de archivo
    "datos": [ // Lista de intersecciones
        {
            "id": "0207001002300-0207001007975", // ID de la calle A, I
D de la calle B
            "calle_a": {
                "id": "0207001002300", // ID de la calle A
                "nombre": "BOSTON", // Nombre de la calle A
                "departamento": { // Departamento de la calle A
                    "id": "02070",
                    "nombre": "Comuna 10"
                },
                "provincia": { // Provincia de la calle A
                    "id": "02",
                    "nombre": "Ciudad Autónoma de Buenos Aires"
                },
                "categoria": "CALLE", // Tipo de la calle A
                "fuente": "INDEC" // Fuente del dato
            },
            "calle_b": {

```

```

    "id": "0207001007975", // ID de la calle B
    "nombre": "MARCOS SASTRE", // Nombre de la calle B
    "departamento": { // Departamento de la calle B
        "id": "02070",
        "nombre": "Comuna 10"
    },
    "provincia": { // Provincia de la calle B
        "id": "02",
        "nombre": "Ciudad Autónoma de Buenos Aires"
    },
    "categoria": "CALLE", // Tipo de la calle B
    "fuente": "INDEC" // Fuente del dato
},
"geometria": { // Geometría en formato GeoJSON
    "type": "Point",
    "coordinates": [
        -58.5077676091915,
        -34.6150993860767
    ]
}
}
{ ... },
]
}

```

# HISTORIAL DE VERSIONES

## 0.3.3 - 2019/02/11

- Corrige error HTTP 500 lanzado cuando se utilizaban valores como "NaN" o "Infinity" para parámetros `lat` y `lon` en `/ubicacion`.

## 0.3.2 - 2019/02/05

- El parámetro `interseccion` de los recursos `/provincias`, `/departamentos` y `/calles` ahora aceptan IDs de calles como parámetros.
- Modifica cantidad de máxima de consultas en una sola petición POST a 1000 (era 5000). El nuevo valor permite normalizar datos a una velocidad prácticamente igual a la anterior, pero con una menor carga a la infraestructura.

## 0.3.1 - 2019/01/29

- Mejora interpretación de direcciones ( `georef-ar-address` versión 0.0.7 ).
- Corrige error HTTP 500 lanzado en normalización de direcciones, en casos donde el comienzo y el fin de alturas de la calle comparten el mismo valor, y la dirección tiene ese valor exacto como altura.

## 0.3.0 - 2019/01/24

- Agrega XML como nuevo formato de respuesta de datos para todos los recursos. Para utilizarlo, agregar `formato=xml` a los parámetros de la URL.
- Agrega *Shapefile* como nuevo formato de respuesta de datos para todos los recursos (excepto `/direcciones` y `/ubicacion`). Para utilizarlo, agregar `formato=shp` a los parámetros de la URL. El archivo descargado contiene las geometrías e información de todas las entidades filtradas. Para más detalles, consultar la documentación de descarga de geometrías.
- El parámetro `id` ahora acepta listas de IDs separadas por comas. Otros parámetros que aceptaban un ID también aceptan ahora listas de IDs.
- Actualiza versión de datos de ETL a 9.0.0.

- Corrige mensajes de error equivocados.
- Cambios varios al recurso `/direcciones` :
  - Utilizando la librería `georef-ar-address` , se mejoró el proceso de interpretación de las direcciones recibidas. Ahora, se aceptan más tipos de direcciones, y la API es capaz de detectar errores comunes de escritura. Para más detalles, consultar la nueva documentación de normalización de direcciones .
  - Se removió el parámetro `tipo` .
  - Se modificó el campo de respuesta `altura` a un objeto `altura` que contiene los valores `valor` y `unidad` .
  - Se removieron los campos `nombre` y `id` , y se agregaron los nuevos campos objeto `calle` , `calle_cruce_1` y `calle_cruce_2` . Cada uno contiene los campos `nombre` , `id` y `categoria` , y representan las calles normalizadas que fueron detectadas en la dirección de entrada.
  - Se agregó el campo `piso` .
  - Se modificaron los nombres y el orden de los campos de respuesta en formato CSV.
- Cambios al recurso `/provincias` :
  - Se agregaron los campos `nombre_completo` , `iso_id` , `iso_nombre` y `categoria` .
- Cambios al recurso `/departamentos` :
  - Se agregaron los campos `nombre_completo` y `categoria` .
- Cambios al recurso `/municipios` :
  - Se agregaron los campos `nombre_completo` y `categoria` .
  - Parámetro `interseccion` : agrega `calle` como tipo de entidad posible a utilizar.
- Cambios al recurso `/localidades` :
  - Se modificó el nombre del campo `tipo` a `categoria` .
- Cambios al recurso `/calles` :
  - Se modificó el nombre del campo `tipo` a `categoria` .
  - Se modificó el nombre del parámetro `tipo` a `categoria` .
  - Agrega parámetro `interseccion` : permite buscar calles por intersección con geometrías de otras entidades.
- Cambios al recurso `/ubicacion` :
  - Se removió el campo `fuelle` , y se agregaron los campos `provincia.fuelle` , `departamento.fuelle` y `municipio.fuelle` .

### 0.2.3 - 2018/12/11

- Modifica formato de respuestas CSV: todos los campos no-numéricos ahora son devueltos entre comillas dobles. El cambio permite a algunas herramientas manejar mejor los valores de IDs, que consisten enteramente de dígitos, pero deberían ser tratados como texto.
- El parámetro 'direccion' del recurso `/direcciones` ahora acepta direcciones con altura 0.
- Mejora la fidelidad de resultados por en búsquedas por nombre: algunos términos de búsqueda ahora excluyen a otros, aunque sean textualmente similares. Por ejemplo, buscar 'salta' no incluye resultados con 'santa', ya que los dos términos refieren a entidades claramente distintas.

### 0.2.2 - 2018/11/05

- El parámetro 'direccion' del recurso `/direcciones` ahora acepta direcciones sin altura (por ejemplo, "Avenida Santa Fe"). Este cambio permite utilizar la versión `POST` del recurso con mayor facilidad, en caso de tener grandes cantidades de datos con y sin alturas en un mismo conjunto.
- Mejoras a mensajes de error para parámetros 'direccion' e 'interseccion'.

### 0.2.1 - 2018/10/25

- Actualiza proceso de indexación para utilizar datos de ETL versión `6.0.0`.
- Agrega mensaje de error descriptivo para errores HTTP 405.
- Mejora parámetro 'campos': se permite especificar un prefijo para referirse a varios campos (por ejemplo, `provincia` para `provincia.id` y `provincia.nombre`).
- Agrega parámetro `interseccion`. El mismo permite buscar provincias, departamentos y municipios utilizando intersección de geometrías con otras entidades geográficas.
  - Se pueden buscar provincias por intersección con municipios y departamentos.
  - Se pueden buscar departamentos por intersección con provincias y municipios.
  - Se pueden buscar municipios por intersección con provincias y departamentos.
- Agrega nuevo campo de datos `provincia.interseccion` a recursos `/municipios` y `/departamentos`. El campo especifica qué porcentaje del área de la provincia ocupa la entidad en sí.
- Agrega nuevos recursos de descarga de datos completos. Estos recursos permiten descargar la totalidad de los datos utilizados por cada recurso, en distintos formatos. Por ejemplo, para descargar los datos de provincias, se puede acceder a las siguientes URLs:
  - <https://apis.datos.gob.ar/georef/api/provincias.json>
  - <https://apis.datos.gob.ar/georef/api/provincias.csv>
  - <https://apis.datos.gob.ar/georef/api/provincias.geojson>

## 0.2.0 - 2018/09/21

- Remueve campo 'departamento' de la entidad municipio. Esto se debe a que los departamentos no son padres jerárquicos de los municipios.
- Agrega parámetro `orden` a recursos `/calles` y `/direcciones`.
- Agrega formato GeoJSON a recurso `/direcciones`.
- Agrega conjuntos de campos predefinidos al parámetro `campos`, los valores posibles son:
  - `basico`
  - `estandar` (utilizado por defecto)
  - `completo`
- Mueve campo de respuestas `fuelle` a conjunto `completo`.
- Permite el uso del parámetro `aplanar` en respuestas GeoJSON.

## 0.1.6 - 2018/09/07

- Actualiza proceso de indexación para utilizar datos de ETL versión `4.0.0`.
- Modifica manejo de altura en recurso `/direcciones`. La nueva versión del recurso intenta ubicar altura dentro de los extremos de la calle tomando en consideración que los datos pueden no siempre estar completos (o ser ideales). Este cambio también afecta la efectividad de la geolocalización de direcciones.

## 0.1.5 - 2018/09/04

- Mejora mensajes de errores para varios casos:
  - Acceso a recursos inexistentes (por ejemplo: `/provincia`).
  - Valores inválidos para parámetros con elección limitada de valores (por ejemplo, `orden`).
  - Listas de operaciones bulk inválidas.
  - Direcciones de calles malformadas.
- Agrega API de paginado.
  - Nuevo parámetro: `inicio`.

- Los resultados ahora incluyen tres metadatos: `cantidad`, `total` e `inicio`.

## 0.1.4 - 2018/08/23

- Se modificó la interpretación del parámetro `direccion` del recurso `/direcciones`:
  - Se ignoran ítems entre paréntesis y ocurrencias de "N°"
  - Se separa el texto utilizando "-", "," y "B°", y se intenta extraer una dirección (nombre + altura) de cada fragmento.

## 0.1.3 - 2018/08/21

- Se modificaron los siguientes campos:
  - `centroide_lat` y `centroide_lon` ahora están anidados.
  - Los campos `altura_inicio_derecha`, `altura_fin_derecha`, etc. ahora están anidados.
- Se agregó una validación de valores repetidos para parámetro `campos`.
- El recurso `/provincias` ahora acepta el parámetro `aplanar`.

## 0.1.2 - 2018/08/16

- Se removió `d` como stopword en Elasticsearch.

## 0.1.1 - 2018/08/15

- Se modificaron los siguientes campos:
  - `lat` ahora es `centroide_lat`.
  - `lon` ahora es `centroide_lon`.
  - Los campos `inicio_derecha`, `fin_derecha`, etc. ahora comienzan con `altura_`.

## 0.1.0 - 2018/08/14

- Versión inicial.

# INSTALACIÓN DE PYTHON 3.6

Para instalar Python 3.6 en entornos GNU/Linux, se puede utilizar la herramienta `pyenv` disponible en GitHub. `pyenv` permite al usuario instalar cualquier versión de Python existente, e incluso tener varias versiones instaladas simultáneamente.

A continuación, se detallan los pasos necesarios para instalar Python 3.6. Los mismos fueron creados utilizando Ubuntu 16.04.

## 1. DESCARGAR PYENV

Clonar el repositorio de `pyenv` en el directorio `~/pyenv`:

```
$ git clone https://github.com/pyenv/pyenv.git ~/.pyenv
```

## 2. AGREGAR CONFIGURACIÓN DE PYENV A ~/.BASHRC

```
$ echo 'export PYENV_ROOT="$HOME/.pyenv"' >> ~/.bashrc
$ echo 'export PATH="$PYENV_ROOT/bin:$PATH"' >> ~/.bashrc
$ echo -e 'if command -v pyenv 1>/dev/null 2>&1; then\n  eval "$(pyenv init -)"\nfi' >> ~/.bashrc
```

### 3. ACTIVAR LA NUEVA CONFIGURACIÓN

```
$ source ~/.bashrc
```

### 4. INSTALAR DEPENDENCIAS PARA COMPILAR PYTHON

```
$ sudo apt install make build-essential libssl-dev zlib1g-dev libbz2-dev  
libreadline-dev libsqlite3-dev wget curl llvm libncurses5-dev libncur  
sesw5-dev xz-utils tk-dev libffi-dev liblzma-dev
```

### 5. DESCARGAR, COMPILAR E INSTALAR PYTHON 3.6

```
$ pyenv install 3.6.5
```

### 6. ACTIVAR PYTHON 3.6

Una vez instalado Python 3.6, se debe activar su uso. `pyenv` permite establecer versiones de Python por directorio: de esta forma, es posible clonar el repositorio `georef-ar-api` en una ubicación, y activar el uso de Python 3.6 en la misma:

```
$ git clone https://github.com/datosgobar/georef-ar-api.git  
$ cd georef-ar-api  
$ pyenv version 3.6.5 # activar el uso de Python 3.6  
$ python --version    # el comando 'python' ahora utiliza Python 3.6, e  
n este directorio  
Python 3.6.5  
$ pip --version        # también se instala 'pip' automáticamente  
pip 9.0.1 (python 3.6.5)
```

Notar que `pyenv` crea un archivo llamado `.python-version`, donde se especifica la versión de Python que debería ser utilizada en el directorio.