pydatajson Documentation

Versión 0.4.47

Datos Argentina

24 de septiembre de 2019

Índice general

Documentación de pydatajson: librería con funcionalidades para gestionar los metadatos de catálogos de datos abiertos que cumplan con el Perfil Nacional de Metadatos. Pydatajson es parte del Paquete de Apertura de Datos.

Podés colaborar cargando un nuevo issue, o respondiendo a un issue ya existente. Lo mismo te invitamos a hacer en el Paquete de Apertura de Datos.

CAPÍTULO 1

Indice

1.1 pydatajson

Paquete en python con herramientas para manipular y validar metadatos de catálogos de datos.

- Versión python: 2 y 3
- Licencia: MIT license
- Documentación: https://pydatajson.readthedocs.io/es/stable
- *Instalación*
- *Usos*
  - *Setup*
  - *Validación de metadatos de catálogos*
    - *Archivo data.json local*
    - *Otros formatos*
  - *Generación de reportes y configuraciones del Harvester*
    - *Crear un archivo de configuración eligiendo manualmente los datasets a federar*
    - *Crear un archivo de configuración que incluya únicamente los datasets con metadata válida*
  - *Transformación de un archivo de metados XLSX al estándar JSON*
  - *Generación de indicadores de monitoreo de catálogos*
- *Tests*
- *Recursos de interés*
- *Créditos*

Este README cubre los casos de uso más comunes para la librería, junto con ejemplos de código, pero sin mayores explicaciones. Para una versión más detallada de los comportamientos, revise la documentación oficial o el Manual de Uso de la librería.

1.1.1 Instalación

- **Producción:** Desde cualquier parte

```
$ pip install pydatajson
```

- **Desarrollo:** Clonar este repositorio, y desde su raíz, ejecutar:

```
$ pip install -e .
```

A partir de la versión 0.2.x (Febrero 2017), la funcionalidad del paquete se mantendrá fundamentalmente estable hasta futuro aviso. De todas maneras, si piensa utilizar esta librería en producción, le sugerimos fijar la versión que emplea en un archivo `requirements.txt`.

1.1.2 Usos

La librería cuenta con funciones para cuatro objetivos principales:

- **validación de metadatos de catálogos** y los *datasets*,
- **generación de reportes** sobre el contenido y la validez de los metadatos de catálogos y *datasets*,
- **transformación de archivos de metadatos** al formato estándar (JSON), y
- **generación de indicadores de monitoreo de catálogos** y sus *datasets*.

A continuación se proveen ejemplos de cada uno de estas acciones. Si desea analizar un flujo de trabajo más completo, refiérase a los Jupyter Notebook de `samples/`

Setup

`DataJson` utiliza un esquema default que cumple con el perfil de metadatos recomendado en la Guía para el uso y la publicación de metadatos (v0.1) del Paquete de Apertura de Datos.

```
from pydatajson import DataJson

dj = DataJson()
```

Si se desea utilizar un esquema alternativo, por favor, consulte la sección «Uso > Setup» del manual oficial, o la documentación oficial.

Validación de metadatos de catálogos

- Si se desea un **resultado sencillo (V o F)** sobre la validez de la estructura del catálogo, se utilizará **`is_valid_catalog(catalog)`**.
- Si se desea un **mensaje de error detallado**, se utilizará **`validate_catalog(catalog)`**.

Por conveniencia, la carpeta `tests/samples/` contiene varios ejemplos de `data.json` bien y mal formados con distintos tipos de errores.

Archivo data.json local

```
from pydatajson import DataJson

dj = DataJson()
catalog = "tests/samples/full_data.json"
validation_result = dj.is_valid_catalog(catalog)
validation_report = dj.validate_catalog(catalog)

print validation_result
True

print validation_report
{
    "status": "OK",
    "error": {
        "catalog": {
            "status": "OK",
            "errors": [],
            "title": "Datos Argentina"
        },
        "dataset": [
            {
                "status": "OK",
                "errors": [],
                "title": "Sistema de contrataciones electrónicas"
            }
        ]
    }
}
```

Otros formatos

`pydatajson` puede interpretar catálogos en formatos:

- JSON
- XLSX (ver plantilla de catálogo en XLSX)

Los catálogos pueden estar almacenados localmente o remotamente a través de URLs de descarga directa. También es capaz de interpretar diccionarios de Python con metadatos de catálogos.

```
from pydatajson import DataJson

dj = DataJson()
catalogs = [
    "tests/samples/full_data.json", # archivo JSON local
    "http://181.209.63.71/data.json", # archivo JSON remoto
    "tests/samples/catalogo_justicia.xlsx", # archivo XLSX local
    "https://raw.githubusercontent.com/datosgobar/pydatajson/master/tests/samples/
↪catalogo_justicia.xlsx", # archivo XLSX remoto
    {
        "title": "Catálogo del Portal Nacional",
    "description" "Datasets abiertos para el ciudadano."
        "dataset": [...],
    (...)
```

(continues on next page)

(proviene de la página anterior)

```
    } # diccionario de Python
]

for catalog in catalogs:
    validation_result = dj.is_valid_catalog(catalog)
    validation_report = dj.validate_catalog(catalog)
```

Generación de reportes y configuraciones del Harvester

Si ya se sabe que se desean cosechar todos los *datasets* [válidos] de uno o varios catálogos, se pueden utilizar directamente el método `generate_harvester_config()`, proveyendo `harvest='all'` o `harvest='valid'` respectivamente. Si se desea revisar manualmente la lista de *datasets* contenidos, se puede invocar primero `generate_datasets_report()`, editar el reporte generado y luego proveérselo a `generate_harvester_config()`, junto con la opción `harvest='report'`.

Crear un archivo de configuración eligiendo manualmente los datasets a federar

```
catalogs = ["tests/samples/full_data.json", "http://181.209.63.71/data.json"]
report_path = "path/to/report.xlsx"
dj.generate_datasets_report(
    catalogs=catalogs,
    harvest='none', # El reporte tendrá `harvest==0` para todos los datasets
    export_path=report_path
)

# A continuación, se debe editar el archivo de Excel 'path/to/report.xlsx',
# cambiando a '1' el campo 'harvest' en los datasets que se quieran cosechar.

config_path = 'path/to/config.csv'
dj.generate_harvester_config(
    harvest='report',
    report=report_path,
    export_path=config_path
)
```

El archivo `config_path` puede ser provisto a Harvester para federar los *datasets* elegidos al editar el reporte intermedio `report_path`.

Por omisión, en la salida de `generate_harvester_config` la frecuencia de actualización deseada para cada *dataset* será «R/P1D», para intentar cosecharlos diariamente. De preferir otra frecuencia (siempre y cuando sea válida según ISO 8601), se la puede especificar a través del parámetro opcional `frequency`. Si especifica explícitamente `frequency=None`, se conservarán las frecuencias de actualización indicadas en el campo `accrualPeriodicity` de cada *dataset*.

Crear un archivo de configuración que incluya únicamente los datasets con metadata válida

Conservando las variables anteriores:

```
dj.generate_harvester_config(
    catalogs=catalogs,
    harvest='valid'
```

(continues on next page)

(proviene de la página anterior)

```
    export_path='path/to/config.csv'
)
```

Transformación de un archivo de metados XLSX al estándar JSON

```
from pydatajson.readers import read_catalog
from pydatajson.writers import write_json
from pydatajson import DataJson

dj = DataJson()
catalogo_xlsx = "tests/samples/catalogo_justicia.xlsx"

catalogo = read_catalog(catalogo_xlsx)
write_json(obj=catalogo, path="tests/temp/catalogo_justicia.json")
```

Generación de indicadores de monitoreo de catálogos

`pydatajson` puede calcular indicadores sobre uno o más catálogos. Estos indicadores recopilan información de interés sobre los *datasets* de cada uno, tales como:

- el estado de validez de los catálogos;
- el número de días desde su última actualización;
- el formato de sus distribuciones;
- frecuencia de actualización de los *datasets*;
- estado de federación de los *datasets*, comparándolo con el catálogo central

La función usada es `generate_catalogs_indicators`, que acepta los catálogos como parámetros. Devuelve dos valores:

- una lista con tantos valores como catálogos, con cada elemento siendo un diccionario con los indicadores del catálogo respectivo;
- un diccionario con los indicadores de la red entera (una suma de los individuales)

```
catalogs = ["tests/samples/full_data.json", "http://181.209.63.71/data.json"]
indicators, network_indicators = dj.generate_catalogs_indicators(catalogs)

# Opcionalmente podemos pasar como segundo argumento un catálogo central,
# para poder calcular indicadores sobre la federación de los datasets en 'catalogs'

central_catalog = "http://datos.gob.ar/data.json"
indicators, network_indicators = dj.generate_catalogs_indicators(catalogs, central_
↪catalog)
```

1.1.3 Tests

Los tests se corren con `nose`. Desde la raíz del repositorio:

Configuración inicial:

```
$ pip install -r requirements_dev.txt
$ mkdir tests/temp
```

Correr la suite de tests:

```
$ nosetests
```

1.1.4 Recursos de interés

- Estándar ISO 8601 - Wikipedia
- JSON SChema - Sitio oficial del estándar
- Documentación completa de `pydatajson` - Read the Docs
- Guía para el uso y la publicación de metafatos

1.1.5 Créditos

El validador de archivos `data.json` desarrollado es mayormente un envoltorio (*wrapper*) alrededor de la librería `jsonschema`, que implementa el vocabulario definido por JSONSchema.org para anotar y validar archivos JSON.

1.2 Manual de uso

- *validate_catalog()*
- *generate_datasets_report()*
- *generate_harvester_config()*
- *generate_datasets_summary()*
- *generate_catalog_readme()*

1.2.1 Contexto

La política de Datos Abiertos de la República Argentina que nace con el Decreto 117/2016 (*«Plan de Apertura de Datos"*) se basa en un esquema descentralizado donde se conforma una red de nodos publicadores de datos y un nodo central o indexador.

El pilar fundamental de este esquema es el cumplimiento de un Perfil Nacional de Metadatos común a todos los nodos, en el que cada organismo de la APN que publique un archivo `data.json` o formato alternativo compatible.

Esto posibilita que todos los conjuntos de datos (*datasets*) publicados por organismos de la Administración Pública Nacional se puedan encontrar en el Portal Nacional de Datos: http://datos.gob.ar/.

1.2.2 Glosario

Un *catálogo* de datos abiertos está compuesto por *datasets*, que a su vez son cada uno un conjunto de *distribuciones* (archivos descargables). Ver la Guía para el uso y la publicación de metadatos para más información.

- **Catálogo de datos**: Directorio de conjuntos de datos que recopila y organiza metadatos descriptivos de los datos que produce una organización. Un portal de datos es una implementación posible de un catálogo. También lo es un archivo Excel, un JSON u otras.
- **Dataset**: También llamado conjunto de datos. Pieza principal en todo catálogo. Se trata de un activo de datos que agrupa recursos referidos a un mismo tema, que respetan una estructura de la información. Los recursos que lo componen pueden diferir en el formato en que se los presenta (por ejemplo: .csv, .json, .xls, etc.), la fecha a la que se refieren, el área geográfica cubierta o estar separados bajo algún otro criterio.
- **Distribución o recurso**: Es la unidad mínima de un catálogo de datos. Se trata de los activos de datos que se publican allí y que pueden ser descargados y re-utilizados por un usuario como archivos. Los recursos pueden tener diversos formatos (.csv, .shp, etc.). Están acompañados de información contextual asociada ("metadata") que describe el tipo de información que se publica, el proceso por el cual se obtiene, la descripción de los campos del recurso y cualquier información extra que facilite su interpretación, procesamiento y lectura.
- **data.json y catalog.xlsx**: Son las dos *representaciones externas* de los metadatos de un catálogo que `pydatajson` comprende. Para poder ser analizados programáticamente, los metadatos de un catálogo deben estar representados en un formato estandarizado: el PAD establece el archivo `data.json` para tal fin, y `pydatajson` permite leer una versión en XLSX equivalente.
- **diccionario de metadatos**: Es la *representación interna* que la librería tiene de los metadatos de un catálogo. Todas las rutinas de la librería `pydatajson` que manipulan catálogos, toman como entrada una *representación externa* (`data.json` o `catalog.xlsx`) del catálogo, y lo primero que hacen es «leerla» y generar una *representación interna* de la información que la rutina sea capaz de manipular.

1.2.3 Uso

.. autofunction:: pydatajson.backup.main

«ñalksdj»

Setup

`DataJson` valida catálogos contra un esquema default que cumple con el perfil de metadatos recomendado en la Guía para el uso y la publicación de metadatos del Paquete de Apertura de Datos.

```
from pydatajson import DataJson

catalog = DataJson("http://datos.gob.ar/data.json")
```

Si se desea utilizar un esquema alternativo, se debe especificar un **directorio absoluto** donde se almacenan los esquemas (`schema_dir`) y un nombre de esquema de validación (`schema_filename`), relativo al directorio de los esquemas. Por ejemplo, si nuestro esquema alternativo se encuentra en `/home/datosgobar/metadatos-portal/esquema_de_validacion.json`, especificaremos:

```
from pydatajson import DataJson

catalog = DataJson("http://datos.gob.ar/data.json",
                   schema_filename="esquema_de_validacion.json",
                   schema_dir="/home/datosgobar/metadatos-portal")
```

Lectura

`pydatajson` puede leer un catálogo en JSON, XLSX, CKAN o `dict` de python:

```
from pydatajson.ckan_reader import read_ckan_catalog
import requests

# data.json
catalog = DataJson("http://datos.gob.ar/data.json")
catalog = DataJson("local/path/data.json")

# catalog.xlsx
catalog = DataJson("http://datos.gob.ar/catalog.xlsx")
catalog = DataJson("local/path/catalog.xlsx")

# CKAN
catalog = DataJson(read_ckan_catalog("http://datos.gob.ar"))

# diccionario de python
catalog_dict = requests.get("http://datos.gob.ar/data.json").json()
catalog = DataJson(catalog_dict)
```

Escritura

Validación

Validar los metadatos de un catálogo y corregir errores.

```
from pydatajson import DataJson

catalog = DataJson("tests/samples/full_data.json")

# es falso si existe por lo menos UN error / verdadero si no hay ningún error
validation_result = catalog.is_valid_catalog(catalog)
```

(continues on next page)

(proviene de la página anterior)

```
# objeto con los errores encontrados
validation_report = catalog.validate_catalog(catalog, only_errors=True)

# se puede tener el reporte en distintos formatos para transformar más fácilmente en␣
↪un informe en CSV o Excel
validation_report = catalog.validate_catalog(catalog, only_errors=True, fmt="list")
```

También se puede correr desde la línea de comandos para ver un resultado rápido.

```
pydatajson validation "tests/samples/full_data.json"
pydatajson validation http://datos.gob.ar/data.json
```

Un ejemplo del resultado completo de `validate_catalog()` se puede consultar en el **Anexo I: Estructura de respuestas**.

Federación y restauración

`pydatajson` permite federar o restaurar fácilmente un dataset de un catálogo hacia un Portal Andino (usa todo el perfil de metadatos) o CKAN (sólo usa campos de metadatos de CKAN), utilizando la API de CKAN.

Para esto hace falta un *apikey* que se puede sacar de la API de CKAN `/api/action/user_list` ingresando con un usuario administrador.

Federar un dataset

Incluye la transformación de algunos metadatos, para adaptar un dataset de un nodo original a cómo debe documentarse en un nodo indexador.

```
catalog_origin = DataJson("https://datos.agroindustria.gob.ar/data.json")

catalog_origin.harvest_dataset_to_ckan(
    owner_org="ministerio-de-agroindustria",
    dataset_origin_identifier="8109e9e8-f8e9-41d1-978a-d20fcd2fe5f5",
    portal_url="http://datos.gob.ar",
    apikey="apikey",
    catalog_id="agroindustria"
)
```

La organización del nodo de destino debe estar previamente creada.

Restaurar un dataset

Los metadatos no sufren transformaciones: se escribe el dataset en el nodo de destino tal cual está en el nodo original.

```
catalog_origin = DataJson("datosgobar/backup/2018-01-01/data.json")

catalog_origin.restore_dataset_to_ckan(
    owner_org="ministerio-de-agroindustria",
    dataset_origin_identifier="8109e9e8-f8e9-41d1-978a-d20fcd2fe5f5",
    portal_url="http://datos.gob.ar",
    apikey="apikey"
)
```

La organización del nodo de destino debe estar previamente creada. En este caso no hace falta `catalog_id` porque el `dataset_identifier` no sufre ninguna transformación.

Transformación de `catalog.xlsx` a `data.json`

La lectura de un archivo de metadatos por parte de `pydatajson.readers.read_catalog` **no realiza ningún tipo de verificación sobre la validez de los metadatos leídos**. Por ende, si se quiere generar un archivo en formato JSON estándar únicamente en caso de que los metadatos de archivo XLSX sean válidos, se deberá realizar la validación por separado.

El siguiente código, por ejemplo, escribe a disco un catálogos de metadatos en formato JSONO sí y sólo sí los metadatos del XLSX leído son válidos:

```
from pydatajson.readers import read_catalog
from pydatajson.writers import write_json
from pydatajson import DataJson

catalog = DataJson()
catalogo_xlsx = "tests/samples/catalogo_justicia.xlsx"

catalogo = read_catalog(catalogo_xlsx)
if catalog.is_valid_catalog(catalogo):
    write_json(obj=catalogo, path="tests/temp/catalogo_justicia.json")
else:
    print "Se encontraron metadatos inválidos. Operación de escritura cancelada."
```

Para más información y una versión más detallada de esta rutina en Jupyter Notebook, dirigirse aquí (metadatos válidos) y aquí (metadatos inválidos).

Generación de reportes

El objetivo final de los métodos `generate_datasets_report`, `generate_harvester_config` y `generate_harvestable_catalogs`, es proveer la configuración que Harvester necesita para cosechar datasets. Todos ellos devuelven una «tabla», que consiste en una lista de diccionarios que comparten las mismas claves (consultar ejemplos en el **Anexo I: Estructura de respuestas**). A continuación, se proveen algunos ejemplos de uso comunes:

Crear un archivo de configuración eligiendo manualmente los datasets a federar

```
catalogs = ["tests/samples/full_data.json", "http://181.209.63.71/data.json"]
report_path = "path/to/report.xlsx"
catalog.generate_datasets_report(
    catalogs=catalogs,
    harvest='none', # El reporte generado tendrá `harvest==0` para todos los datasets
    export_path=report_path
)
# A continuación, se debe editar el archivo de Excel 'path/to/report.xlsx', cambiando␣
↪a '1' el campo 'harvest' para aquellos datasets que se quieran cosechar.

config_path = 'path/to/config.csv'
catalog.generate_harvester_config(
    harvest='report',
    report=report_path,
```

(continues on next page)

(proviene de la página anterior)

```
    export_path=config_path
)
```

El archivo `config_path` puede ser provisto a Harvester para federar los datasets elegidos al editar el reporte intermedio `report_path`.

Alternativamente, el output de `generate_datasets_report()` se puede editar en un intérprete de python:

```
# Asigno el resultado a una variable en lugar de exportarlo
datasets_report = catalog.generate_datasets_report(
    catalogs=catalogs,
    harvest='none', # El reporte generado tendrá `harvest==0` para todos los datasets
)
# Imaginemos que sólo se desea federar el primer dataset del reporte:
datasets_report[0]["harvest"] = 1

config_path = 'path/to/config.csv'
catalog.generate_harvester_config(
    harvest='report',
    report=datasets_report,
    export_path=config_path
)
```

Crear un archivo de configuración que incluya únicamente los datasets con metadata válida

Conservando las variables anteriores:

```
catalog.generate_harvester_config(
    catalogs=catalogs,
    harvest='valid'
    export_path='path/to/config.csv'
)
```

Para fines ilustrativos, se incluye el siguiente bloque de código que produce los mismos resultados, pero genera el reporte intermedio sobre datasets:

```
datasets_report = catalog.generate_datasets_report(
    catalogs=catalogs,
    harvest='valid'
)

# Como el reporte ya contiene la información necesaria sobre los datasets que se␣
↪pretende cosechar, el argumento `catalogs` es innecesario.
catalog.generate_harvester_config(
    harvest='report'
    report=datasets_report
    export_path='path/to/config.csv'
)
```

Modificar catálogos para conservar únicamente los datasets válidos

```
# Creamos un directorio donde guardar los catálogos
output_dir = "catalogos_limpios"
```

(continues on next page)

(proviene de la página anterior)

```
import os; os.mkdir(output_dir)

catalog.generate_harvestable_catalogs(
    catalogs,
    harvest='valid',
    export_path=output_dir
)
```

1.2.4 Funcionalidades

La librería cuenta con funciones para tres objetivos principales:

- **validación de metadatos de catálogos** y los datasets,
- **generación de reportes** sobre el contenido y la validez de los metadatos de catálogos y datasets,
- **transformación de archivos de metadatos** al formato estándar (JSON),
- **federación de datasets** a portales de destino.

Como se menciona en el Glosario estos métodos no tienen acceso *directo* a ningún catálogo, dataset ni distribución, sino únicamente a sus *representaciones externas*: archivos o partes de archivos en formato JSON que describen ciertas propiedades. Por conveniencia, en este documento se usan frases como «validar el dataset X», cuando una versión más precisa sería «validar la fracción del archivo `data.json` que consiste en una representación del dataset X en forma de diccionario». La diferencia es sutil, pero conviene mantenerla presente.

Todos los métodos públicos de la librería toman como primer parámetro `catalog`:

- o bien un diccionario de metadatos (una *representación interna*),
- o la ruta (local o remota) a un archivo de metadatos en formato legible (idealmente JSON, alternativamente XLSX).

Cuando el parámetro esperado es `catalogs`, en plural, se le puede pasar o un único catálogo, o una lista de ellos.

Todos los métodos comienzan por convertir `catalog(s)` en una **representación interna** unívoca: un diccionario cuyas claves son las definidas en el Perfil de Metadatos. La conversión se realiza a través de `pydatajson.readers.read_catalog(catalog)`: éste es la función que todos ellos invocan para obtener un diccionario de metadatos estándar.

Métodos de validación de metadatos

- **pydatajson.DataJson.is_valid_catalog(catalog) -> bool**: Responde `True` únicamente si el catálogo no contiene ningún error.
- **pydatajson.DataJson.validate_catalog(catalog) -> dict**: Responde un diccionario con información detallada sobre la validez «global» de los metadatos, junto con detalles sobre la validez de los metadatos a nivel catálogo y cada uno de sus datasets. De haberlos, incluye una lista con información sobre los errores encontrados.

Métodos de transformación de formatos de metadatos

Transformar un archivo de metadatos de un formato a otro implica un primer paso de lectura de un formato, y un segundo paso de escritura a un formato distinto. Para respetar las disposiciones del PAD, sólo se pueden escribir catálogos en formato JSON.

- **pydatajson.readers.read_catalog()**: Método que todas las funciones de DataJson llaman en primer lugar para interpretar cualquier tipo de representación externa de un catálogo.

- **pydatajson.writers.write_json_catalog()**: Fina capa de abstracción sobre `pydatajson.writers.write_json`, que simplemente vuelca un objeto de Python a un archivo en formato JSON.

Métodos de generación de reportes

Para federación de datasets

Los siguientes métodos toman una o varias representaciones externas de catálogos, y las procesan para generar reportes específicos sobre su contenido:

- **pydatajson.DataJson.generate_datasets_report()**: Devuelve un reporte con información clave sobre cada dataset incluido en un catálogo, junto con variables indicando la validez de sus metadatos.
- **pydatajson.DataJson.generate_harvester_config()**: Devuelve un reporte con los campos mínimos que requiere el Harvester para federar un conjunto de datasets.
- **pydatajson.DataJson.generate_harvestable_catalogs()**: Devuelve la lista de catálogos ingresada, filtrada de forma que cada uno incluya únicamente los datasets que se pretende que el Harvester federe.

Los tres métodos toman los mismos cuatro parámetros, que se interpretan de manera muy similar:

- **catalogs**: Representación externa de un catálogo, o una lista compuesta por varias de ellas.
- **harvest**: Criterio de decisión utilizado para marcar los datasets a ser federados/cosechados. Acepta los siguientes valores:
  - `'all'`: Cosechar todos los datasets presentes en **catalogs**.
  - `'none'`: No cosechar ninguno de los datasets presentes en **catalogs**.
  - `'valid'`: Cosechar únicamente los datasets que no contengan errores, ni en su propia metadata ni en la metadata global del catálogo.
  - `'report'`: Cosechar únicamente los datasets indicados por el reporte provisto en `report`.
- **report**: En caso de que se pretenda cosechar un conjunto específico de catálogos, esta variable debe recibir la representación externa (path a un archivo) o interna (lista de diccionarios) de un reporte que identifique los datasets a cosechar.
- **export_path**: Esta variable controla el valor de retorno de los métodos de generación. Si es `None`, el método devolverá la representación interna del reporte generado. Si especifica el path a un archivo, el método devolverá `None`, pero escribirá a `export_path` la representación externa del reporte generado, en formato CSV o XLSX.

generate_harvester_config() puede tomar un parámetro extra, `frequency`, que permitirá indicarle a la rutina de cosecha de con qué frecuencia debe intentar actualizar su versión de cierto dataset. Por omisión, lo hará diariamente.

Para presentación de catálogos y datasets

Existen dos métodos, cuyos reportes se incluyen diariamente entre los archivos que disponibiliza el repositorio `libreria-catalogos`:

- **pydatajson.DataJson.generate_datasets_summary()**: Devuelve un informe tabular (en formato CSV o XLSX) sobre los datasets de un catálogo, detallando cuántas distribuciones tiene y el estado de sus propios metadatos.
- **pydatajson.DataJson.generate_catalog_readme()**: Genera un archivo de texto plano en formato Markdown para ser utilizado como «README», es decir, como texto introductorio al contenido del catálogo.

Métodos para federación de datasets

- **pydatajson.DataJson.push_dataset_to_ckan()**: Copia la metadata de un dataset y la escribe en un portal de CKAN. Toma los siguientes parámetros:
  - **owner_org**: La organización a la que pertence el dataset. Debe encontrarse en el portal de destino.
  - **dataset_origin_identifier**: Identificador del dataset en el catálogo de origen.
  - **portal_url**: URL del portal de CKAN de destino.
  - **apikey**: La apikey de un usuario del portal de destino con los permisos para crear el dataset bajo la organización pasada como parámetro.
  - **catalog_id** (opcional, default: None): El prefijo que va a preceder el id y name del dataset en el portal destino, separado por un guión.
  - **demote_superThemes** (opcional, default: True):Si está en true, los ids de los themes del dataset, se escriben como groups de CKAN.
  - **demote_themes** (opcional, default: True): Si está en true, los labels de los themes del dataset, se escriben como tags de CKAN; sino,se pasan como grupo.
  - **download_strategy** (opcional, default None): La referencia a una función que toma (catalog, distribution) de entrada y devuelve un booleano. Esta función se aplica sobre todas las distribuciones del dataset. Si devuelve `True`, se descarga el archivo indicado en el `downloadURL` de la distribución y se lo sube al portal de destino. Si es None, se omite esta operación.
  - **generate_new_access_url** (opcional, default None): Se pasan los ids de las distribuciones cuyo accessURL se regenerar en el portal de destino. Para el resto, el portal debe mantiene el valor pasado en el DataJson.

Retorna el id en el nodo de destino del dataset federado.

Advertencia: La función `push_dataset_to_ckan()` sólo garantiza consistencia con los estándares de CKAN. Para mantener una consistencia más estricta dentro del catálogo a federar, es necesario validar los datos antes de pasarlos a la función.

- **pydatajson.federation.remove_dataset_from_ckan()**: Hace un borrado físico de un dataset en un portal de CKAN. Toma los siguientes parámetros:
  - **portal_url**: La URL del portal CKAN. Debe implementar la funcionalidad de `/data.json`.
  - **apikey**: La apikey de un usuario con los permisos que le permitan borrar el dataset.
  - **filter_in**: Define el diccionario de filtro en el campo `dataset`. El filtro acepta los datasets cuyos campos coincidan con todos los del diccionario `filter_in['dataset']`.
  - **filter_out**: Define el diccionario de filtro en el campo `dataset`. El filtro acepta los datasets cuyos campos coincidan con alguno de los del diccionario `filter_out['dataset']`.
  - **only_time_series**: Borrar los datasets que tengan recursos con series de tiempo.
  - **organization**: Borrar los datasets que pertenezcan a cierta organizacion.

  En caso de pasar más de un parámetro opcional, la función `remove_dataset_from_ckan()` borra aquellos datasets que cumplan con todas las condiciones.
- **pydatajson.DataJson.push_theme_to_ckan()**: Crea un tema en el portal de destino Toma los siguientes parámetros:
  - **portal_url**: La URL del portal CKAN.
  - **apikey**: La apikey de un usuario con los permisos que le permitan crear un grupo.

- **identifier** (opcional, default: None): Id del `theme` que se quiere federar, en el catálogo de origen.
- **label** (opcional, default: None): label del `theme` que se quiere federar, en el catálogo de origen.

Debe pasarse por lo menos uno de los 2 parámetros opcionales. En caso de que se provean los 2, se prioriza el identifier sobre el label.

- **pydatajson.DataJson.push_new_themes()**: Toma los temas de la taxonomía de un DataJson y los crea en el catálogo de destino si no existen. Toma los siguientes parámetros:
  - **portal_url**: La URL del portal CKAN adonde se escribiran los temas.
  - **apikey**: La apikey de un usuario con los permisos que le permitan crear los grupos.

Hay también funciones que facilitan el uso de `push_dataset_to_ckan()`:

- **pydatajson.DataJson.harvest_dataset_to_ckan()**: Federa la metadata de un dataset en un portal de CKAN. Toma los siguientes parámetros:
  - **owner_org**: La organización a la que pertence el dataset. Debe encontrarse en el portal de destino.
  - **dataset_origin_identifier**: Identificador del dataset en el catálogo de origen.
  - **portal_url**: URL del portal de CKAN de destino.
  - **apikey**: La apikey de un usuario del portal de destino con los permisos para crear el dataset bajo la organización pasada como parámetro.
  - **catalog_id**: El prefijo que va a preceder el id y name del dataset en el portal destino, separado por un guión.
  - **download_strategy** (opcional, default None): La referencia a una función que toma (catalog, distribution) de entrada y devuelve un booleano. Esta función se aplica sobre todas las distribuciones del dataset. Si devuelve `True`, se descarga el archivo indicado en el `downloadURL` de la distribución y se lo sube al portal de destino. Si es None, se omite esta operación.

Retorna el id en el nodo de destino del dataset federado.

- **pydatajson.DataJson.restore_dataset_to_ckan()**: Restaura la metadata de un dataset en un portal de CKAN. Toma los siguientes parámetros:
  - **owner_org**: La organización a la que pertence el dataset. Debe encontrarse en el portal de destino.
  - **dataset_origin_identifier**: Identificador del dataset en el catálogo de origen.
  - **portal_url**: URL del portal de CKAN de destino.
  - **apikey**: La apikey de un usuario del portal de destino con los permisos para crear el dataset bajo la organización pasada como parámetro.
  - **download_strategy** (opcional, default None): La referencia a una función que toma (catalog, distribution) de entrada y devuelve un booleano. Esta función se aplica sobre todas las distribuciones del dataset. Si devuelve `True`, se descarga el archivo indicado en el `downloadURL` de la distribución y se lo sube al portal de destino. Si es None, se omite esta operación.
  - **generate_new_access_url** (opcional, default None): Se pasan los ids de las distribuciones cuyo accessURL se regenerar en el portal de destino. Para el resto, el portal debe mantiene el valor pasado en el DataJson.

Retorna el id del dataset restaurado.

- **pydatajson.DataJson.harvest_catalog_to_ckan()**: Federa los datasets de un catálogo al portal pasado por parámetro. Toma los siguientes parámetros:
  - **dataset_origin_identifier**: Identificador del dataset en el catálogo de origen.

- **portal_url**: URL del portal de CKAN de destino.
- **apikey**: La apikey de un usuario del portal de destino con los permisos para crear el dataset.
- **catalog_id**: El prefijo que va a preceder el id y name del dataset en el portal destino, separado por un guión.
- **dataset_list** (opcional, default: None): Lista de ids de los datasets a federar. Si no se pasa, se federan todos los datasets del catálogo.
- **owner_org** (opcional, default: None): La organización a la que pertence el dataset. Debe encontrarse en el portal de destino. Si no se pasa, se toma como organización el catalog_id.
- **download_strategy** (opcional, default None): La referencia a una función que toma (catalog, distribution) de entrada y devuelve un booleano. Esta función se aplica sobre todas las distribuciones del catálogo. Si devuelve `True`, se descarga el archivo indicado en el `downloadURL` de la distribución y se lo sube al portal de destino. Si es None, se omite esta operación.

Retorna el id en el nodo de destino de los datasets federados.

- **pydatajson.DataJson.restore_organization_to_ckan()**: Restaura los datasets de una organización al portal pasado por parámetro. Toma los siguientes parámetros:
  - **catalog**: El catálogo de origen que se restaura.
  - **portal_url**: La URL del portal CKAN de destino.
  - **apikey**: La apikey de un usuario con los permisos que le permitan crear o actualizar los dataset.
  - **dataset_list**: Los ids de los datasets a restaurar. Si no se pasa una lista, todos los datasests se restauran.
  - **owner_org**: La organización a la cual pertencen los datasets.
  - **download_strategy**: Una función (catálogo, distribución)->bool. Sobre las distribuciones que evalúa True, descarga el recurso en el downloadURL y lo sube al portal de destino. Por default no sube ninguna distribución.
  - **generate_new_access_url** (opcional, default None): Se pasan los ids de las distribuciones cuyo accessURL se regenerar en el portal de destino. Para el resto, el portal debe mantiene el valor pasado en el DataJson.

  Retorna la lista de ids de datasets subidos.

- **pydatajson.DataJson.restore_catalog_to_ckan()**: Restaura los datasets de un catálogo al portal pasado por parámetro. Toma los siguientes parámetros:
  - **catalog**: El catálogo de origen que se restaura.
  - **origin_portal_url**: La URL del portal CKAN de origen.
  - **destination_portal_url**: La URL del portal CKAN de destino.
  - **apikey**: La apikey de un usuario con los permisos que le permitan crear o actualizar los dataset.
  - **download_strategy**: Una función (catálogo, distribución)-> bool. Sobre las distribuciones que evalúa True, descarga el recurso en el downloadURL y lo sube al portal de destino. Por default no sube ninguna distribución.
  - **generate_new_access_url** (opcional, default None): Se pasan los ids de las distribuciones cuyo accessURL se regenerar en el portal de destino. Para el resto, el portal debe mantiene el valor pasado en el DataJson.

  Retorna un diccionario con key organización y value la lista de ids de datasets subidos a esa organización

- **pydatajson.federation.resources_update()**: Sube archivos de recursos a las distribuciones indicadas y regenera los accessURL en las distribuciones indicadas. Toma los siguientes parámetros:

- **portal_url**: URL del portal de CKAN de destino.
- **apikey**: La apikey de un usuario del portal de destino con los permisos para modificar la distribución.
- **distributions**: Lista de distribuciones posibles para actualizar.
- **resource_files** Diccionario con el id de las distribuciones y un path al recurso correspondiente a subir.
- **generate_new_access_url** (opcional, default None): Lista de ids de distribuciones a las cuales se actualizará el accessURL con los valores generados por el portal de destino.
- **catalog_id** (opcional, default None): prependea el id al id del recurso para encontrarlo antes de subirlo si es necesario.

Retorna una lista con los ids de las distribuciones modificadas exitosamente.

Advertencia: La función `resources_update()` cambia el `resource_type` de las distribuciones a `file.upload`.

Métodos para manejo de organizaciones

- **pydatajson.federation.get_organizations_from_ckan()**: Devuelve el árbol de organizaciones del portal pasado por parámetro. Toma los siguientes parámetros:
  - **portal_url**: URL del portal de CKAN. Debe implementar el endpoint `/group_tree`.

  Retorna una lista de diccionarios con la información de las organizaciones. Recursivamente, dentro del campo `children`, se encuentran las organizaciones dependientes en la jerarquía.
- **pydatajson.federation.get_organization_from_ckan()**: Devuelve un diccionario con la información de una organización en base a un id y un portal pasados por parámetro. Toma los siguientes parámetros:
  - **portal_url**: URL del portal de CKAN. Debe implementar el endpoint `/organization_show`.
  - **org_id**: Identificador o name de la organización a buscar.

  Retorna un diccionario con la información de la organización correspondiente al identificador obtenido. *No* incluye su jerarquía, por lo cual ésta deberá ser conseguida mediante el uso de la función `get_organizations_from_ckan`.
- **pydatajson.federation.push_organization_tree_to_ckan()**: Tomando un árbol de organizaciones como el creado por `get_organizations_from_ckan()` crea en el portal de destino las organizaciones dentro de su jerarquía. Toma los siguientes parámetros:
  - **portal_url**: La URL del portal CKAN de destino.
  - **apikey**: La apikey de un usuario con los permisos que le permitan crear las organizaciones.
  - **org_tree**: lista de diccionarios con la data de organizaciones a crear.
  - **parent** (opcional, default: None): Si se pasa, el árbol de organizaciones pasado en `org_tree` se crea bajo la organización con `name` pasado en `parent`. Si no se pasa un parámetro, las organizaciones son creadas como primer nivel.

  Retorna el árbol de organizaciones creadas. Cada nodo tiene un campo `success` que indica si fue creado exitosamente o no. En caso de que `success` sea False, los hijos de esa organización no son creados.
- **pydatajson.federation.push_organization_to_ckan()**: Tomando en un diccionario la data de una organización; la crea en el portal de destino. Toma los siguientes parámetros:
  - **portal_url**: La URL del portal CKAN de destino.
  - **apikey**: La apikey de un usuario con los permisos que le permitan crear las organizaciones.

- **organization**: Diccionario con la información a crear, el único campo obligatorio es `name`. Para más información sobre los campos posibles, ver la documentación de CKAN
- **parent** (opcional, default: None): Si se define, la organización pasada en `organization` se crea bajo la organización con `name` pasado en `parent`. Si no se pasa un parámetro, las organizaciones son creadas como primer nivel.

Retorna el diccionario de la organización creada. El resultado tiene un campo `success` que indica si fue creado exitosamente o no.

- **pydatajson.federation.remove_organization_from_ckan()**: Tomando el id o name de una organización; la borra en el portal de destino. Toma los siguientes parámetros:
  - **portal_url**: La URL del portal CKAN de destino.
  - **apikey**: La apikey de un usuario con los permisos que le permitan borrar la organización.
  - **organization_id**: Id o name de la organización a borrar.

  Retorna None.

  Advertencia: En caso de que la organización tenga hijos en la jerarquía, estos pasan a ser de primer nivel.
- **pydatajson.federation.remove_organizations_from_ckan()**: Tomando una lista de ids o names de organizaciones, las borra en el portal de destino. Toma los siguientes parámetros:
  - **portal_url**: La URL del portal CKAN de destino.
  - **apikey**: La apikey de un usuario con los permisos que le permitan borrar organizaciones.
  - **organization_list**: Lista de id o names de las organizaciones a borrar.

  Retorna None.

1.2.5 Anexo I: Estructura de respuestas

validate_catalog()

El resultado de la validación completa de un catálogo, es un diccionario con la siguiente estructura:

```
{
    "status": "OK",  # resultado de la validación global
    "error": {
    "catalog": {
            # validez de la metadata propia del catálogo, ignorando los
            # datasets particulares
        "status": "OK",
        "errors": []
        "title": "Título Catalog"},
    "dataset": [
        {
        # Validez de la metadata propia de cada dataset
                "status": "OK",
        "errors": [],
        "title": "Titulo Dataset 1"
        },
        {
        "status": "ERROR",
        "errors": [
                    {
```

(continues on next page)

(proviene de la página anterior)

```
                        "error_code": 2,
                        "instance": "",
                        "message": "'' is not a 'email'",
                        "path": ["publisher", "mbox"],
                        "validator": "format",
                        "validator_value": "email"
                    },
                    {
                        "error_code": 2,
                        "instance": "",
                        "message": """" is too short",
                        "path": ["publisher", "name"],
                        "validator": "minLength",
                        "validator_value": 1
                    }
                ],
                "title": "Titulo Dataset 2"
        }
    ]
    }
}
```

Si `validate_catalog()` encuentra algún error, éste se reportará en la lista `errors` del nivel correspondiente, a través de un diccionario con las siguientes claves:

- **path**: Posición en el diccionario de metadata del catálogo donde se encontró el error.
- **instance**: Valor concreto que no pasó la validación. Es el valor de la clave `path` en la metadata del catálogo.
- **message**: Descripción humanamente legible explicando el error.
- **validator**: Nombre del validador violado, («type» para errores de tipo, «minLength» para errores de cadenas vacías, et cétera).
- **validator_value**: Valor esperado por el validador `validator`, que no fue respetado.
- **error_code**: Código describiendo genéricamente el error. Puede ser:
  - **1**: Valor obligatorio faltante: Un campo obligatorio no se encuentra presente.
  - **2**: Error de tipo y formato: se esperaba un `array` y se encontró un `dict`, se esperaba un `string` en formato `email` y se encontró una `string` que no cumple con el formato, et cétera.

generate_datasets_report()

El reporte resultante tendrá tantas filas como datasets contenga el conjunto de catálogos ingresado, y contará con los siguientes campos, casi todos autodescriptivos:

- **catalog_metadata_url**: En caso de que se haya provisto una representación externa de un catálogo, la string de su ubicación; sino `None`.
- **catalog_title**
- **catalog_description**
- **valid_catalog_metadata**: Validez de la metadata «global» del catálogo, es decir, ignorando la metadata de datasets particulares.
- **dataset_title**
- **dataset_description**

- **dataset_index**: Posición (comenzando desde cero) en la que aparece el dataset en cuestión en lista del campo `catalog["dataset"]`.
- **valid_dataset_metadata**: Validez de la metadata *específica a este dataset* que figura en el catálogo (`catalog["dataset"][dataset_index]`).
- **harvest**: "0" o "1", según se desee excluir o incluir, respectivamente, un dataset de cierto proceso de cosecha. El default es "0", pero se puede controlar a través del parámetro "harvest".
- **dataset_accrualPeriodicity**
- **dataset_publisher_name**
- **dataset_superTheme**: Lista los valores que aparecen en el campo dataset[«superTheme»], separados por comas.
- **dataset_theme**: Lista los valores que aparecen en el campo dataset[«theme»], separados por comas.
- **dataset_landingPage**
- **distributions_list**: Lista los títulos y direcciones de descarga de todas las distribuciones incluidas en un dataset, separadas por «newline».

La *representación interna* de este reporte es una lista compuesta en su totalidad de diccionarios con las claves mencionadas. La *representación externa* de este reporte, es un archivo con información tabular, en formato CSV o XLSX. A continuación, un ejemplo de la *lista de diccionarios* que devuelve `generate_datasets_report()`:

```
[
    {
        "catalog_metadata_url": "http://181.209.63.71/data.json",
        "catalog_title": "Andino",
        "catalog_description": "Portal Andino Demo",
        "valid_catalog_metadata": 0,
        "dataset_title": "Dataset Demo",
        "dataset_description": "Este es un dataset de ejemplo, se incluye como␣
↪material DEMO y no contiene ningun valor estadistico.",
        "dataset_index": 0,
        "valid_dataset_metadata": 1,
        "harvest": 0,
        "dataset_accrualPeriodicity": "eventual",
        "dataset_publisher_name": "Andino",
        "dataset_superThem"": "TECH",
        "dataset_theme": "Tema.demo",
        "dataset_landingPage": "https://github.com/datosgobar/portal-andino",
        "distributions_list": ""Recurso de Ejemplo": http://181.209.63.71/dataset/
↪6897d435-8084-4685-b8ce-304b190755e4/resource/6145bf1c-a2fb-4bb5-b090-bb25f8419198/
↪download/estructura-organica-3.csv"
    },
    {
        "catalog_metadata_url": "http://datos.gob.ar/data.json",
        "catalog_title": "Portal Nacional de Datos Abiertos",
        ( ... )
    }
]
```

generate_harvester_config()

Este reporte se puede generar a partir de un conjunto de catálogos, o a partir del resultado de `generate_datasets_report()`, pues no es más que un subconjunto del mismo. Incluye únicamente las claves necesarias para que el Harvester pueda federar un dataset, si `'harvest'==1`:

- **catalog_metadata_url**
- **dataset_title**
- **dataset_accrualPeriodicity**

La *representación interna* de este reporte es una lista compuesta en su totalidad de diccionarios con las claves mencionadas. La *representación externa* de este reporte, es un archivo con información tabular, en formato CSV o XLSX. A continuación, un ejemplo con la *lista de diccionarios* que devuelve `generate_harvester_config()`:

```
[
    {
        "catalog_metadata_url": "tests/samples/full_data.json",
        "dataset_title": "Sistema de contrataciones electrónicas",
        "dataset_accrualPeriodicity": "R/P1Y"
    },
    {
        "catalog_metadata_url": "tests/samples/several_datasets_for_harvest.json",
        "dataset_title": "Sistema de Alumbrado Público CABA",
        "dataset_accrualPeriodicity": "R/P1Y"
    },
    {
        "catalog_metadata_url": "tests/samples/several_datasets_for_harvest.json",
        "dataset_title": "Listado de Presidentes Argentinos",
        "dataset_accrualPeriodicity": "R/P1Y"
    }
]
```

generate_datasets_summary()

Se genera a partir de un único catálogo, y contiene, para cada uno de dus datasets:

- **Índice**: El índice, identificador posicional del dataset dentro de la lista `catalog["dataset"]`.
- **Título**: dataset[«title»], si lo tiene (es un campo obligatorio).
- **Identificador**: dataset[«identifier»], si lo tiene (es un campo recomendado).
- **Cantidad de Errores**: Cuántos errores de validación contiene el dataset, según figure en el detalle de `validate_catalog`
- **Cantidad de Distribuiones**: El largo de la lista `dataset["distribution"]`

A continuación, un fragmento del resultado de este método al aplicarlo sobre el Catálogo del Ministerio de Justicia:

```
[OrderedDict([(u'indice', 0),
              (u'titulo', u'Base de datos legislativos Infoleg'),
              (u'identificador', u'd9a963ea-8b1d-4ca3-9dd9-07a4773e8c23'),
              (u'estado_metadatos', u'OK'),
              (u'cant_errores', 0),
              (u'cant_distribuciones', 3)]),
 OrderedDict([(u'indice', 1),
              (u'titulo', u'Centros de Acceso a la Justicia -CAJ-'),
              (u'identificador', u'9775fcdf-99b9-47f6-87ae-6d46cfd15b40'),
              (u'estado_metadatos', u'OK'),
              (u'cant_errores', 0),
              (u'cant_distribuciones', 1)]),
 OrderedDict([(u'indice', 2),
              (u'titulo',
```

(continues on next page)

(proviene de la página anterior)

```
                u'Sistema de Consulta Nacional de Rebeld\xedas y Capturas - Co.Na.R.C.
↪'),
               (u'identificador', u'e042c362-ff39-476f-9328-056a9de753f0'),
               (u'estado_metadatos', u'OK'),
               (u'cant_errores', 0),
               (u'cant_distribuciones', 1)]),

( ... 13 datasets más ...)

 OrderedDict([(u'indice', 15),
               (u'titulo',
                u'Registro, Sistematizaci\xf3n y Seguimiento de Hechos de Violencia␣
↪Institucional'),
               (u'identificador', u'c64b3899-65df-4024-afe8-bdf971f30dd8'),
               (u'estado_metadatos', u'OK'),
               (u'cant_errores', 0),
               (u'cant_distribuciones', 1)])]
```

generate_catalog_readme()

Este reporte en texto plano se pretende como primera introducción somera al contenido de un catálogo, como figurarán en la Librería de Catálogos. Incluye datos clave sobre el editor responsable del catálogo, junto con:

- estado de los metadatos a nivel catálogo,
- estado global de los metadatos, y
- cantidad de datasets y distribuciones incluidas.

A continuación, el resultado de este método al aplicarlo sobre el Catálogo del Ministerio de Justicia:

```
# Catálogo: Datos Justicia Argentina

## Información General

- **Autor**: Ministerio de Justicia y Derechos Humanos
- **Correo Electrónico**: justiciaabierta@jus.gov.ar
- **Nombre del catálogo**: Datos Justicia Argentina
- **Descripción**:

> Portal de Datos de Justicia de la República Argentina. El Portal publica datos del␣
↪sistema de justicia de modo que pueda ser reutilizada para efectuar visualizaciones␣
↪o desarrollo de aplicaciones. Esta herramienta se propone como un punto de␣
↪encuentro entre las organizaciones de justicia y la ciudadanía.

## Estado de los metadatos y cantidad de recursos

Estado metadatos globales | Estado metadatos catálogo | # de Datasets | # de␣
↪Distribuciones
----------------|---------------------------|---------------|--------------------
OK | OK | 16 | 56

## Datasets incluidos

Por favor, consulte el informe [`datasets.csv`](datasets.csv).
```

1.2.6 Anexo II: Restaurar un catálogo

El primer paso es replicar la estructura de organizaciones del catálogo original al catálogo destino. Asumiendo que los nombres e ids de las organizaciones del original no se utilizan en el portal donde se replican:

```
from pydatajson.federation import get_organizations_from_ckan, push_organization_tree_
↪to_ckan
arbol_original = get_organizations_from_ckan('url_portal_original')
arbol_replicado = push_organization_tree_to_ckan('url_portal_destino', 'apikey',␣
↪arbol_original)
```

Para cada organización en `arbol_replicado`, el campo `success` tiene un booleano que marca si fue subida exitosamente. Con las organizaciones replicadas podemos restaurar la data y metadata del catálogo orginal:

```
from pydatajson.core import DataJson
from pydatajson.helpers import is_local_andino_resource

original = DataJson('portal-original/data.json')
pushed_datasets =original.restore_catalog_to_ckan('portal-original-url', 'portal-
↪destino-url', 'apikey',
    download_strategy=is_local_andino_resource)
```

Si pasamos `download_strategy=None`, tan solo se restaura la metadata. `is_local_andino_resource` es una función auxiliar que toma una distribución y un catálogo y realiza las siguientes validaciones:

-1: Chequea que el campo `type` sea `file.upload`

-2: Si la distribución no tiene campo `type`, chequea que el `downloadURL` comience con el `homepage` del catálogo

Si se cumple alguna de las condiciones, descarga el recurso y lo sube al portal de destino. Tambien es posible definir una función propia como estrategia para carga y descarga de archivos. Esta función debe tomar una distribución, un catálogo y devolver un booleano.

1.3 Referencia rápida

1.3.1 Lectura

class pydatajson.core.**DataJson**(*catalog=None*, *schema_filename=None*, *schema_dir=None*, *default_values=None*, *catalog_format=None*, *validator_class=<class 'pydatajson.validation.Validator'>*, *verify_ssl=False*, *requests_timeout=30*)

Objeto que representa un catálogo de activos de datos.

__init__(*catalog=None*, *schema_filename=None*, *schema_dir=None*, *default_values=None*, *catalog_format=None*, *validator_class=<class 'pydatajson.validation.Validator'>*, *verify_ssl=False*, *requests_timeout=30*)

Lee un catálogo y crea un objeto con funciones para manipularlo.

Salvo que se indique lo contrario, se utiliza como default el schema de la versión 1.1 del Perfil de Metadatos de Argentina.

Parámetros

- **catalog** (*dict or str*) – Representación externa/interna de un catálogo. Una representación _externa_ es un path local o una URL remota a un archivo con la metadata de un catálogo, en formato JSON o XLSX. La representación _interna_ de un catálogo es

un diccionario. Ejemplos: http://datos.gob.ar/data.json, http://www.ign.gob.ar/descargas/geodatos/catalog.xlsx, «/energia/catalog.xlsx».

- **schema_filename** (*str*) – Nombre del archivo que contiene el esquema validador.
- **schema_dir** (*str*) – Directorio (absoluto) donde se encuentra el esquema validador (y sus referencias, de tenerlas).
- **default_values** (*dict*) – Valores default para algunos de los campos del catálogo:

```
{
    "dataset_issued": "2017-06-22",
    "distribution_issued": "2017-06-22"
}
```

1.3.2 Escritura

class pydatajson.core.**DataJson**(*catalog=None*, *schema_filename=None*, *schema_dir=None*, *default_values=None*, *catalog_format=None*, *validator_class=<class 'pydatajson.validation.Validator'>*, *verify_ssl=False*, *requests_timeout=30*)

Objeto que representa un catálogo de activos de datos.

to_json(*path*)

Escribe el catálogo en JSON.

Parámetros

- **catalog** (DataJson) – Catálogo de datos.
- **path** (*str*) – Directorio absoluto donde se crea el archivo XLSX.

to_xlsx(*path*, *xlsx_fields=None*)

Escribe el catálogo en Excel.

Parámetros

- **catalog** (DataJson) – Catálogo de datos.
- **path** (*str*) – Directorio absoluto donde se crea el archivo XLSX.
- **xlsx_fields** (*dict*) – Orden en que los campos del perfil de metadatos se escriben en cada hoja del Excel.

1.3.3 Validación

class pydatajson.core.**DataJson**(*catalog=None*, *schema_filename=None*, *schema_dir=None*, *default_values=None*, *catalog_format=None*, *validator_class=<class 'pydatajson.validation.Validator'>*, *verify_ssl=False*, *requests_timeout=30*)

Objeto que representa un catálogo de activos de datos.

is_valid_catalog(*catalog=None*)

Valida que un archivo *data.json* cumpla con el schema definido.

Chequea que el data.json tiene todos los campos obligatorios y que tanto los campos obligatorios como los opcionales siguen la estructura definida en el schema.

Parámetros **catalog** (*str o dict*) – Catálogo (dict, JSON o XLSX) a ser validado. Si no se pasa, valida este catálogo.

Devuelve True si el data.json cumple con el schema, sino False.

Tipo del valor devuelto bool

validate_catalog(*catalog=None*, *only_errors=False*, *fmt=u'dict'*, *export_path=None*)

Analiza un data.json registrando los errores que encuentra.

Chequea que el data.json tiene todos los campos obligatorios y que tanto los campos obligatorios como los opcionales siguen la estructura definida en el schema.

Parámetros

- **catalog** (*str o dict*) – Catálogo (dict, JSON o XLSX) a ser validado. Si no se pasa, valida este catálogo.
- **only_errors** (*bool*) – Si es True sólo se reportan los errores.
- **fmt** (*str*) – Indica el formato en el que se desea el reporte. «dict» es el reporte más verborrágico respetando la

  estructura del data.json.

  »list» devuelve un dict con listas de errores formateados para generar tablas.

- **export_path** (*str*) – Path donde exportar el reporte generado (en formato XLSX o CSV). Si se especifica, el método no devolverá nada, a pesar de que se pase algún argumento en *fmt*.

Devuelve

Diccionario resumen de los errores encontrados:

```
{
    "status": "OK",  # resultado de la validación global
    "error": {
        "catalog": {
            "status": "OK",
            "errors": []
            "title": "Título Catalog"},
        "dataset": [
            {
                "status": "OK",
                "errors": [],
                "title": "Titulo Dataset 1"
            },
            {
                "status": "ERROR",
                "errors": [error1_info, error2_info, ...],
                "title": "Titulo Dataset 2"
            }
        ]
    }
}
```

Donde errorN_info es un dict con la información del N-ésimo error encontrado, con las siguientes claves: «path», «instance», «message», «validator», «validator_value», «error_code».

Tipo del valor devuelto dict

1.3.4 Búsqueda

class pydatajson.core.**DataJson**(*catalog=None*, *schema_filename=None*, *schema_dir=None*, *default_values=None*, *catalog_format=None*, *validator_class=<class 'pydatajson.validation.Validator'>*, *verify_ssl=False*, *requests_timeout=30*)

Objeto que representa un catálogo de activos de datos.

get_dataset(*identifier=None*, *title=None*)

Devuelve un Dataset del catálogo.

get_datasets(*filter_in=None*, *filter_out=None*, *meta_field=None*, *exclude_meta_fields=None*, *only_time_series=False*)

Devuelve una lista de datasets del catálogo o de uno de sus metadatos.

Parámetros

- **catalog** (*dict, str or DataJson*) – Representación externa/interna de un catálogo. Una representación _externa_ es un path local o una URL remota a un archivo con la metadata de un catálogo, en formato JSON o XLSX. La representación _interna_ de un catálogo es un diccionario. Ejemplos: http://datos.gob.ar/data.json, http://www.ign.gob.ar/descargas/geodatos/catalog.xlsx, «/energia/catalog.xlsx».
- **filter_in** (*dict*) – Devuelve los datasets cuyos atributos coinciden con los pasados en este diccionario. Ejemplo:

```
{
    "dataset": {
        "publisher": {"name": "Ministerio de Ambiente"}
    }
}
```

  Sólo se devolverán los datasets de ese publisher_name.
- **filter_out** (*dict*) – Devuelve los datasets cuyos atributos no coinciden con los pasados en este diccionario. Ejemplo:

```
{
    "dataset": {
        "publisher": {"name": "Ministerio de Ambiente"}
    }
}
```

  Sólo se devolverán los datasets que no sean de ese publisher_name.
- **meta_field** (*str*) – Nombre de un metadato de Dataset. En lugar de devolver los objetos completos «Dataset», devuelve una lista de valores para ese metadato presentes en el catálogo.
- **exclude_meta_fields** (*list*) – Metadatos de Dataset que se quieren excluir de los objetos Dataset devueltos.
- **only_time_series** (*bool*) – Si es verdadero, sólo devuelve datasets que tengan por lo menos una distribución de series de tiempo.

get_field(*identifier=None*, *title=None*, *distribution_identifier=None*)

Devuelve un Field del catálogo.

get_fields(*filter_in=None*, *filter_out=None*, *meta_field=None*, *only_time_series=False*, *distribution_identifier=None*)

Devuelve lista de campos del catálogo o de uno de sus metadatos.

Parámetros

- **catalog** (*dict, str or DataJson*) – Representación externa/interna de un catálogo. Una representación _externa_ es un path local o una URL remota a un archivo con la metadata de un catálogo, en formato JSON o XLSX. La representación _interna_ de un catálogo es un diccionario. Ejemplos: http://datos.gob.ar/data.json, http://www.ign.gob.ar/descargas/geodatos/catalog.xlsx, «/energia/catalog.xlsx».
- **filter_in** (*dict*) – Devuelve los campos cuyos atributos coinciden con los pasados en este diccionario. Ejemplo:

```
{
    "dataset": {
        "publisher": {"name": "Ministerio de Ambiente"}
    }
}
```

  Sólo se devolverán los campos que pertenezcan a un dataset de ese publisher_name.
- **filter_out** (*dict*) – Devuelve los campos cuyos atributos no coinciden con los pasados en este diccionario. Ejemplo:

```
{
    "dataset": {
        "publisher": {"name": "Ministerio de Ambiente"}
    }
}
```

  Sólo se devolverán los campos que no pertenezcan a un dataset de ese publisher_name.
- **meta_field** (*str*) – Nombre de un metadato de Field. En lugar de devolver los objetos completos «Field», devuelve una lista de valores para ese metadato presentes en el catálogo.
- **exclude_meta_fields** (*list*) – Metadatos de Field que se quieren excluir de los objetos Field devueltos.
- **only_time_series** (*bool*) – Si es verdadero, sólo devuelve campos que sean series de tiempo.

1.3.5 Indicadores

class pydatajson.core.**DataJson**(*catalog=None*, *schema_filename=None*, *schema_dir=None*, *default_values=None*, *catalog_format=None*, *validator_class=<class 'pydatajson.validation.Validator'>*, *verify_ssl=False*, *requests_timeout=30*)

Objeto que representa un catálogo de activos de datos.

1.3.6 Reportes

class pydatajson.core.**DataJson**(*catalog=None*, *schema_filename=None*, *schema_dir=None*, *default_values=None*, *catalog_format=None*, *validator_class=<class 'pydatajson.validation.Validator'>*, *verify_ssl=False*, *requests_timeout=30*)

Objeto que representa un catálogo de activos de datos.

generate_catalog_readme(*catalog*, *export_path=None*)
Este método está para mantener retrocompatibilidad con versiones anteriores. Se ignora el argumento _data_json.

generate_datasets_summary(*catalog*, *export_path=None*)
Genera un informe sobre los datasets presentes en un catálogo, indicando para cada uno:

- Índice en la lista catalog[«dataset»]
- Título
- Identificador
- Cantidad de distribuciones
- Estado de sus metadatos [«OK»|»ERROR»]

Es utilizada por la rutina diaria de *libreria-catalogos* para reportar sobre los datasets de los catálogos mantenidos.

Parámetros

- **catalog** (*str o dict*) – Path a un catálogo en cualquier formato, JSON, XLSX, o diccionario de python.
- **export_path** (*str*) – Path donde exportar el informe generado (en formato XLSX o CSV). Si se especifica, el método no devolverá nada.

Devuelve Contiene tantos dicts como datasets estén presentes en *catalogs*, con los datos antes mencionados.

Tipo del valor devuelto list

1.3.7 Federación

class pydatajson.core.**DataJson**(*catalog=None*, *schema_filename=None*, *schema_dir=None*, *default_values=None*, *catalog_format=None*, *validator_class=<class 'pydatajson.validation.Validator'>*, *verify_ssl=False*, *requests_timeout=30*)
Objeto que representa un catálogo de activos de datos.

harvest_catalog_to_ckan(*portal_url*, *apikey*, *catalog_id*, *dataset_list=None*, *owner_org=None*, *download_strategy=None*, *origin_tz='America/Buenos_Aires'*, *dst_tz='America/Buenos_Aires'*)
Federa los datasets de un catálogo al portal pasado por parámetro.

Parámetros

- **catalog** (DataJson) – El catálogo de origen que se federa.
- **portal_url** (*str*) – La URL del portal CKAN de destino.
- **apikey** (*str*) – La apikey de un usuario con los permisos que le permitan crear o actualizar el dataset.
- **catalog_id** (*str*) – El prefijo con el que va a preceder el id del dataset en catálogo destino.
- **dataset_list** (*list(str)*) – Los ids de los datasets a federar. Si no se pasa una lista, todos los datasests se federan.
- **owner_org** (*str*) – La organización a la cual pertencen los datasets. Si no se pasa, se utiliza el catalog_id.

- **download_strategy** (*callable*) – Una función (catálogo, distribución)-> bool. Sobre las distribuciones que evalúa True, descarga el recurso en el downloadURL y lo sube al portal de destino. Por default no sube ninguna distribución.
- **origin_tz** (*str*) – Timezone de origen, un string (EJ: Africa/Bamako) el cual identifica el timezone del emisor del DataJson.
- **dst_tz** (*str*) – Timezone de destino, un string (EJ: Antarctica/Palmer) el cual identifica el timezone del receptor del DataJson, comunmente el timezone del servidor.

Devuelve El id del dataset en el catálogo de destino.

Tipo del valor devuelto str

harvest_dataset_to_ckan(*owner_org*, *dataset_origin_identifier*, *portal_url*, *apikey*, *catalog_id*, *download_strategy=None*, *origin_tz='America/Buenos_Aires'*, *dst_tz='America/Buenos_Aires'*)

Federa la metadata de un dataset en el portal pasado por parámetro.

Parámetros

- **catalog** (DataJson) – El catálogo de origen que contiene el dataset.
- **owner_org** (*str*) – La organización a la cual pertence el dataset.
- **dataset_origin_identifier** (*str*) – El id del dataset que se va a restaurar.
- **portal_url** (*str*) – La URL del portal CKAN de destino.
- **apikey** (*str*) – La apikey de un usuario con los permisos que le permitan crear o actualizar el dataset.
- **catalog_id** (*str*) – El id que prependea al dataset y recursos
- **download_strategy** (*callable*) – Una función (catálogo, distribución)-> bool. Sobre las distribuciones que evalúa True, descarga el recurso en el downloadURL y lo sube al portal de destino. Por default no sube ninguna distribución.
- **origin_tz** (*str*) – Timezone de origen, un string (EJ: Africa/Bamako) el cual identifica el timezone del emisor del DataJson.
- **dst_tz** (*str*) – Timezone de destino, un string (EJ: Antarctica/Palmer) el cual identifica el timezone del receptor del DataJson, comunmente el timezone del servidor.

Devuelve El id del dataset restaurado.

Tipo del valor devuelto str

push_new_themes(*portal_url*, *apikey*)

Toma un catálogo y escribe los temas de la taxonomía que no están presentes.

Args:

catalog (DataJson): El catálogo de origen que contiene la taxonomía.

portal_url (str): La URL del portal CKAN de destino. apikey (str): La apikey de un usuario con los permisos que le

permitan crear o actualizar los temas.

Returns: str: Los ids de los temas creados.

push_theme_to_ckan(*portal_url*, *apikey*, *identifier=None*, *label=None*)

Escribe la metadata de un theme en el portal pasado por parámetro.

Parámetros

- **catalog** (DataJson) – El catálogo de origen que contiene el theme.
- **portal_url** (*str*) – La URL del portal CKAN de destino.
- **apikey** (*str*) – La apikey de un usuario con los permisos que le permitan crear o actualizar el dataset.
- **identifier** (*str*) – El identificador para buscar el theme en la taxonomia.
- **label** (*str*) – El label para buscar el theme en la taxonomia.

Devuelve El name del theme en el catálogo de destino.

Tipo del valor devuelto str

restore_catalog_to_ckan(*origin_portal_url*, *destination_portal_url*, *apikey*, *download_strategy=None*, *generate_new_access_url=None*, *origin_tz='America/Buenos_Aires'*, *dst_tz='America/Buenos_Aires'*)

Restaura los datasets de un catálogo original al portal pasado por parámetro. Si hay temas presentes en el DataJson que no están en el portal de CKAN, los genera.

Args: catalog (DataJson): El catálogo de origen que se restaura. origin_portal_url (str): La URL del portal CKAN de origen. destination_portal_url (str): La URL del portal CKAN de

destino.

apikey (str): La apikey de un usuario con los permisos que le permitan crear o actualizar el dataset.

download_strategy(callable): Una función (catálogo, distribución)-> bool. Sobre las distribuciones que evalúa True, descarga el recurso en el downloadURL y lo sube al portal de destino. Por default no sube ninguna distribución.

generate_new_access_url(list): Se pasan los ids de las distribuciones cuyo accessURL se regenerar en el portal de destino. Para el resto, el portal debe mantiene el valor pasado en el DataJson.

origin_tz(str): Timezone de origen, un string (EJ: Africa/Bamako) el cual identifica el timezone del emisor del DataJson.

dst_tz(str): Timezone de destino, un string (EJ: Antarctica/Palmer) el cual identifica el timezone del receptor del DataJson, comunmente el timezone del servidor.

Returns:

dict: Diccionario con key organización y value la lista de ids de datasets subidos a esa organización

restore_dataset_to_ckan(*owner_org*, *dataset_origin_identifier*, *portal_url*, *apikey*, *download_strategy=None*, *generate_new_access_url=None*, *origin_tz='America/Buenos_Aires'*, *dst_tz='America/Buenos_Aires'*)

Restaura la metadata de un dataset en el portal pasado por parámetro.

Parámetros

- **catalog** (DataJson) – El catálogo de origen que contiene el dataset.
- **owner_org** (*str*) – La organización a la cual pertence el dataset.
- **dataset_origin_identifier** (*str*) – El id del dataset que se va a restaurar.
- **portal_url** (*str*) – La URL del portal CKAN de destino.
- **apikey** (*str*) – La apikey de un usuario con los permisos que le permitan crear o actualizar el dataset.

- **download_strategy** (*callable*) – Una función (catálogo, distribución)-> bool. Sobre las distribuciones que evalúa True, descarga el recurso en el downloadURL y lo sube al portal de destino. Por default no sube ninguna distribución.
- **generate_new_access_url** (*list*) – Se pasan los ids de las distribuciones cuyo accessURL se regenerar en el portal de destino. Para el resto, el portal debe mantiene el valor pasado en el DataJson.
- **origin_tz** (*str*) – Timezone de origen, un string (EJ: Africa/Bamako) el cual identifica el timezone del emisor del DataJson.
- **dst_tz** (*str*) – Timezone de destino, un string (EJ: Antarctica/Palmer) el cual identifica el timezone del receptor del DataJson, comunmente el timezone del servidor.

Devuelve El id del dataset restaurado.

Tipo del valor devuelto str

pydatajson.federation.**remove_dataset_from_ckan**(*identifier*, *portal_url*, *apikey*, *verify_ssl=False*, *requests_timeout=30*)

CAPÍTULO 2

Referencia

2.1 Referencia completa

2.1.1 DataJson

Módulo principal de pydatajson

Contiene la clase DataJson que reúne los métodos públicos para trabajar con archivos data.json.

class pydatajson.core.**DataJson**(*catalog=None*, *schema_filename=None*, *schema_dir=None*, *default_values=None*, *catalog_format=None*, *validator_class=<class 'pydatajson.validation.Validator'>*, *verify_ssl=False*, *requests_timeout=30*)

Objeto que representa un catálogo de activos de datos.

catalog_report(*catalog*, *harvest=u'none'*, *report=None*, *catalog_id=None*, *catalog_homepage=None*, *catalog_org=None*)

Genera un reporte sobre los datasets de un único catálogo.

Parámetros

- **catalog** (*dict, str o unicode*) – Representación externa (path/URL) o interna (dict) de un catálogo.
- **harvest** (*str*) – Criterio de cosecha ("all", "none", "valid", "report" o "good").

Devuelve

Lista de diccionarios, con un elemento por cada dataset presente en *catalog*.

Tipo del valor devuelto list

datasets

Devuelve una lista de datasets del catálogo o de uno de sus metadatos.

Parámetros

- **catalog** (*dict, str or DataJson*) – Representación externa/interna de un catálogo. Una representación _externa_ es un path local o una URL remota a un archivo con la metadata de un catálogo, en formato JSON o XLSX. La representación _interna_ de un catálogo es un diccionario. Ejemplos: http://datos.gob.ar/data.json, http://www.ign.gob.ar/descargas/geodatos/catalog.xlsx, «/energia/catalog.xlsx».
- **filter_in** (*dict*) – Devuelve los datasets cuyos atributos coinciden con los pasados en este diccionario. Ejemplo:

```
{
    "dataset": {
        "publisher": {"name": "Ministerio de Ambiente"}
    }
}
```

  Sólo se devolverán los datasets de ese publisher_name.
- **filter_out** (*dict*) – Devuelve los datasets cuyos atributos no coinciden con los pasados en este diccionario. Ejemplo:

```
{
    "dataset": {
        "publisher": {"name": "Ministerio de Ambiente"}
    }
}
```

  Sólo se devolverán los datasets que no sean de ese publisher_name.
- **meta_field** (*str*) – Nombre de un metadato de Dataset. En lugar de devolver los objetos completos «Dataset», devuelve una lista de valores para ese metadato presentes en el catálogo.
- **exclude_meta_fields** (*list*) – Metadatos de Dataset que se quieren excluir de los objetos Dataset devueltos.
- **only_time_series** (*bool*) – Si es verdadero, sólo devuelve datasets que tengan por lo menos una distribución de series de tiempo.

distributions

Devuelve lista de distribuciones del catálogo o de uno de sus metadatos.

Parámetros

- **catalog** (*dict, str or DataJson*) – Representación externa/interna de un catálogo. Una representación _externa_ es un path local o una URL remota a un archivo con la metadata de un catálogo, en formato JSON o XLSX. La representación _interna_ de un catálogo es un diccionario. Ejemplos: http://datos.gob.ar/data.json, http://www.ign.gob.ar/descargas/geodatos/catalog.xlsx, «/energia/catalog.xlsx».
- **filter_in** (*dict*) – Devuelve los distribuciones cuyos atributos coinciden con los pasados en este diccionario. Ejemplo:

```
{
    "dataset": {
        "publisher": {"name": "Ministerio de Ambiente"}
    }
}
```

  Sólo se devolverán los distribuciones que pertenezcan a un dataset de ese publisher_name.

- **filter_out** (*dict*) – Devuelve los distribuciones cuyos atributos no coinciden con los pasados en este diccionario. Ejemplo:

```
{
    "dataset": {
        "publisher": {"name": "Ministerio de Ambiente"}
    }
}
```

  Sólo se devolverán los distribuciones que no pertenezcan a un dataset de ese publisher_name.

- **meta_field** (*str*) – Nombre de un metadato de Distribution. En lugar de devolver los objetos completos Distribution, devuelve una lista de valores para ese metadato presentes en el catálogo.
- **exclude_meta_fields** (*list*) – Metadatos de Distribution que se quieren excluir de los objetos Distribution devueltos.
- **only_time_series** (*bool*) – Si es verdadero, sólo devuelve distribuciones que sean distribuciones de series de tiempo.

fields

Devuelve lista de campos del catálogo o de uno de sus metadatos.

Parámetros

- **catalog** (*dict, str or DataJson*) – Representación externa/interna de un catálogo. Una representación _externa_ es un path local o una URL remota a un archivo con la metadata de un catálogo, en formato JSON o XLSX. La representación _interna_ de un catálogo es un diccionario. Ejemplos: http://datos.gob.ar/data.json, http://www.ign.gob.ar/descargas/geodatos/catalog.xlsx, «/energia/catalog.xlsx».
- **filter_in** (*dict*) – Devuelve los campos cuyos atributos coinciden con los pasados en este diccionario. Ejemplo:

```
{
    "dataset": {
        "publisher": {"name": "Ministerio de Ambiente"}
    }
}
```

  Sólo se devolverán los campos que pertenezcan a un dataset de ese publisher_name.

- **filter_out** (*dict*) – Devuelve los campos cuyos atributos no coinciden con los pasados en este diccionario. Ejemplo:

```
{
    "dataset": {
        "publisher": {"name": "Ministerio de Ambiente"}
    }
}
```

  Sólo se devolverán los campos que no pertenezcan a un dataset de ese publisher_name.

- **meta_field** (*str*) – Nombre de un metadato de Field. En lugar de devolver los objetos completos «Field», devuelve una lista de valores para ese metadato presentes en el catálogo.
- **exclude_meta_fields** (*list*) – Metadatos de Field que se quieren excluir de los objetos Field devueltos.

- **only_time_series** (*bool*) – Si es verdadero, sólo devuelve campos que sean series de tiempo.

generate_catalog_readme(*catalog*, *export_path=None*)
Este método está para mantener retrocompatibilidad con versiones anteriores. Se ignora el argumento _data_json.

generate_dataset_documentation(*dataset_identifier*, *export_path=None*, *catalog=None*)
Genera texto en markdown a partir de los metadatos de una *dataset*.

Parámetros

- **dataset_identifier** (*str*) – Identificador único de un dataset.
- **export_path** (*str*) – Path donde exportar el texto generado. Si se especifica, el método no devolverá nada.
- **catalog** (*dict, str o unicode*) – Representación externa (path/URL) o interna (dict) de un catálogo. Si no se especifica se usa el catálogo cargado en *self* (el propio objeto DataJson).

Devuelve Texto que describe una *dataset*.

Tipo del valor devuelto str

generate_datasets_report(*catalogs*, *harvest=u'valid'*, *report=None*, *export_path=None*, *catalog_ids=None*, *catalog_homepages=None*, *catalog_orgs=None*)
Genera un reporte sobre las condiciones de la metadata de los datasets contenidos en uno o varios catálogos.

Parámetros

- **catalogs** (*str, dict o list*) – Uno (str o dict) o varios (list de strs y/o dicts) catálogos.
- **harvest** (*str*) – Criterio a utilizar para determinar el valor del campo «harvest» en el reporte generado ("all", "none", "valid", "report" o "good").
- **report** (*str*) – Path a un reporte/config especificando qué datasets marcar con harvest=1 (sólo si harvest=="report").
- **export_path** (*str*) – Path donde exportar el reporte generado (en formato XLSX o CSV). Si se especifica, el método no devolverá nada.
- **catalog_id** (*str*) – Nombre identificador del catálogo para federación
- **catalog_homepage** (*str*) – URL del portal de datos donde está implementado el catálogo. Sólo se pasa si el portal es un CKAN o respeta la estructura:

  https://datos.{organismo}.gob.ar/dataset/{dataset_identifier}

Devuelve

Contiene tantos dicts como datasets estén presentes en *catalogs*, con la data del reporte generado.

Tipo del valor devuelto list

generate_datasets_summary(*catalog*, *export_path=None*)
Genera un informe sobre los datasets presentes en un catálogo, indicando para cada uno:

- Índice en la lista catalog[«dataset»]
- Título
- Identificador

- Cantidad de distribuciones
- Estado de sus metadatos [«OK»|»ERROR»]

Es utilizada por la rutina diaria de *libreria-catalogos* para reportar sobre los datasets de los catálogos mantenidos.

Parámetros

- **catalog** (*str o dict*) – Path a un catálogo en cualquier formato, JSON, XLSX, o diccionario de python.
- **export_path** (*str*) – Path donde exportar el informe generado (en formato XLSX o CSV). Si se especifica, el método no devolverá nada.

Devuelve Contiene tantos dicts como datasets estén presentes en *catalogs*, con los datos antes mencionados.

Tipo del valor devuelto list

generate_distribution_ids()

Genera identificadores para las distribuciones que no los tienen.

Los identificadores de distribuciones se generan concatenando el id del dataset al que pertenecen con el índice posicional de la distribución en el dataset: distribution_identifier = «{dataset_identifier}_{index}».

generate_harvestable_catalogs(*catalogs*, *harvest=u'all'*, *report=None*, *export_path=None*)

Filtra los catálogos provistos según el criterio determinado en *harvest*.

Parámetros

- **catalogs** (*str, dict o list*) – Uno (str o dict) o varios (list de strs y/o dicts) catálogos.
- **harvest** (*str*) – Criterio para determinar qué datasets conservar de cada catálogo ("all", "none", "valid" o "report").
- **report** (*list o str*) – Tabla de reporte generada por generate_datasets_report() como lista de diccionarios o archivo en formato XLSX o CSV. Sólo se usa cuando *harvest=="report"*.
- **export_path** (*str*) – Path a un archivo JSON o directorio donde exportar los catálogos filtrados. Si termina en «.json» se exportará la lista de catálogos a un único archivo. Si es un directorio, se guardará en él un JSON por catálogo. Si se especifica *export_path*, el método no devolverá nada.

Devuelve Lista de catálogos.

Tipo del valor devuelto list of dicts

generate_harvester_config(*catalogs=None*, *harvest=u'valid'*, *report=None*, *export_path=None*)

Genera un archivo de configuración del harvester a partir de un reporte, o de un conjunto de catálogos y un criterio de cosecha (*harvest*).

Parámetros

- **catalogs** (*str, dict o list*) – Uno (str o dict) o varios (list de strs y/o dicts) catálogos.
- **harvest** (*str*) – Criterio para determinar qué datasets incluir en el archivo de configuración generado ("all", "none", "valid", "report" o "good").

- **report** (*list o str*) – Tabla de reporte generada por generate_datasets_report() como lista de diccionarios o archivo en formato XLSX o CSV. Sólo se usa cuando *harvest=="report"*, en cuyo caso *catalogs* se ignora.
- **export_path** (*str*) – Path donde exportar el reporte generado (en formato XLSX o CSV). Si se especifica, el método no devolverá nada.

Devuelve

Un diccionario con variables de configuración por cada dataset a cosechar.

Tipo del valor devuelto list of dicts

get_catalog_metadata(*exclude_meta_fields=None*)
Devuelve sólo la metadata de nivel catálogo.

get_dataset(*identifier=None*, *title=None*)
Devuelve un Dataset del catálogo.

get_datasets(*filter_in=None*, *filter_out=None*, *meta_field=None*, *exclude_meta_fields=None*, *only_time_series=False*)
Devuelve una lista de datasets del catálogo o de uno de sus metadatos.

Parámetros

- **catalog** (*dict, str or DataJson*) – Representación externa/interna de un catálogo. Una representación _externa_ es un path local o una URL remota a un archivo con la metadata de un catálogo, en formato JSON o XLSX. La representación _interna_ de un catálogo es un diccionario. Ejemplos: http://datos.gob.ar/data.json, http://www.ign.gob.ar/descargas/geodatos/catalog.xlsx, «/energia/catalog.xlsx».
- **filter_in** (*dict*) – Devuelve los datasets cuyos atributos coinciden con los pasados en este diccionario. Ejemplo:

```
{
    "dataset": {
        "publisher": {"name": "Ministerio de Ambiente"}
    }
}
```

  Sólo se devolverán los datasets de ese publisher_name.
- **filter_out** (*dict*) – Devuelve los datasets cuyos atributos no coinciden con los pasados en este diccionario. Ejemplo:

```
{
    "dataset": {
        "publisher": {"name": "Ministerio de Ambiente"}
    }
}
```

  Sólo se devolverán los datasets que no sean de ese publisher_name.
- **meta_field** (*str*) – Nombre de un metadato de Dataset. En lugar de devolver los objetos completos «Dataset», devuelve una lista de valores para ese metadato presentes en el catálogo.
- **exclude_meta_fields** (*list*) – Metadatos de Dataset que se quieren excluir de los objetos Dataset devueltos.
- **only_time_series** (*bool*) – Si es verdadero, sólo devuelve datasets que tengan por lo menos una distribución de series de tiempo.

get_distribution(*identifier=None*, *title=None*, *dataset_identifier=None*)
Devuelve una Distribution del catálogo.

get_distributions(*filter_in=None*, *filter_out=None*, *meta_field=None*, *exclude_meta_fields=None*, *only_time_series=False*)
Devuelve lista de distribuciones del catálogo o de uno de sus metadatos.

Parámetros

- **catalog** (*dict, str or DataJson*) – Representación externa/interna de un catálogo. Una representación _externa_ es un path local o una URL remota a un archivo con la metadata de un catálogo, en formato JSON o XLSX. La representación _interna_ de un catálogo es un diccionario. Ejemplos: http://datos.gob.ar/data.json, http://www.ign.gob.ar/descargas/geodatos/catalog.xlsx, «/energia/catalog.xlsx».
- **filter_in** (*dict*) – Devuelve los distribuciones cuyos atributos coinciden con los pasados en este diccionario. Ejemplo:

```
{
    "dataset": {
        "publisher": {"name": "Ministerio de Ambiente"}
    }
}
```

  Sólo se devolverán los distribuciones que pertenezcan a un dataset de ese publisher_name.
- **filter_out** (*dict*) – Devuelve los distribuciones cuyos atributos no coinciden con los pasados en este diccionario. Ejemplo:

```
{
    "dataset": {
        "publisher": {"name": "Ministerio de Ambiente"}
    }
}
```

  Sólo se devolverán los distribuciones que no pertenezcan a un dataset de ese publisher_name.
- **meta_field** (*str*) – Nombre de un metadato de Distribution. En lugar de devolver los objetos completos Distribution, devuelve una lista de valores para ese metadato presentes en el catálogo.
- **exclude_meta_fields** (*list*) – Metadatos de Distribution que se quieren excluir de los objetos Distribution devueltos.
- **only_time_series** (*bool*) – Si es verdadero, sólo devuelve distribuciones que sean distribuciones de series de tiempo.

get_field(*identifier=None*, *title=None*, *distribution_identifier=None*)
Devuelve un Field del catálogo.

get_fields(*filter_in=None*, *filter_out=None*, *meta_field=None*, *only_time_series=False*, *distribution_identifier=None*)
Devuelve lista de campos del catálogo o de uno de sus metadatos.

Parámetros

- **catalog** (*dict, str or DataJson*) – Representación externa/interna de un catálogo. Una representación _externa_ es un path local o una URL remota a un archivo con la metadata de un catálogo, en formato JSON o XLSX. La representación _interna_ de un catálogo es un diccionario. Ejemplos: http://datos.gob.ar/data.json, http://www.ign.gob.ar/descargas/geodatos/catalog.xlsx, «/energia/catalog.xlsx».

- **filter_in** (*dict*) – Devuelve los campos cuyos atributos coinciden con los pasados en este diccionario. Ejemplo:

```
{
    "dataset": {
        "publisher": {"name": "Ministerio de Ambiente"}
    }
}
```

  Sólo se devolverán los campos que pertenezcan a un dataset de ese publisher_name.

- **filter_out** (*dict*) – Devuelve los campos cuyos atributos no coinciden con los pasados en este diccionario. Ejemplo:

```
{
    "dataset": {
        "publisher": {"name": "Ministerio de Ambiente"}
    }
}
```

  Sólo se devolverán los campos que no pertenezcan a un dataset de ese publisher_name.

- **meta_field** (*str*) – Nombre de un metadato de Field. En lugar de devolver los objetos completos «Field», devuelve una lista de valores para ese metadato presentes en el catálogo.
- **exclude_meta_fields** (*list*) – Metadatos de Field que se quieren excluir de los objetos Field devueltos.
- **only_time_series** (*bool*) – Si es verdadero, sólo devuelve campos que sean series de tiempo.

get_themes()
Devuelve la lista de temas del catálogo (taxonomía temática).

get_time_series(***kwargs*)
Devuelve lista de series de tiempo del catálogo o uno de sus metadatos.

Parámetros

- **catalog** (*dict, str or* DataJson) – Representación externa/interna de un catálogo. Una representación _externa_ es un path local o una URL remota a un archivo con la metadata de un catálogo, en formato JSON o XLSX. La representación _interna_ de un catálogo es un diccionario. Ejemplos: http://datos.gob.ar/data.json, http://www.ign.gob.ar/descargas/geodatos/catalog.xlsx, «/energia/catalog.xlsx».
- **filter_in** (*dict*) – Devuelve las series cuyos atributos coinciden con los pasados en este diccionario. Ejemplo:

```
{
    "dataset": {
        "publisher": {"name": "Ministerio de Ambiente"}
    }
}
```

  Sólo se devolverán las series que pertenezcan a un dataset de ese publisher_name.

- **filter_out** (*dict*) – Devuelve las series cuyos atributos no coinciden con los pasados en este diccionario. Ejemplo:

```
{
    "dataset": {
        "publisher": {"name": "Ministerio de Ambiente"}
    }
}
```

Sólo se devolverán las series que no pertenezcan a un dataset de ese publisher_name.

- **meta_field** (*str*) – Nombre de un metadato de Field. En lugar de devolver los objetos completos «Field», devuelve una lista de valores para ese metadato presentes en el catálogo.
- **exclude_meta_fields** (*list*) – Metadatos de Field que se quieren excluir de los objetos Field devueltos.

harvest_catalog_to_ckan(*portal_url*, *apikey*, *catalog_id*, *dataset_list=None*, *owner_org=None*, *download_strategy=None*, *origin_tz='America/Buenos_Aires'*, *dst_tz='America/Buenos_Aires'*)

Federa los datasets de un catálogo al portal pasado por parámetro.

Parámetros

- **catalog** (DataJson) – El catálogo de origen que se federa.
- **portal_url** (*str*) – La URL del portal CKAN de destino.
- **apikey** (*str*) – La apikey de un usuario con los permisos que le permitan crear o actualizar el dataset.
- **catalog_id** (*str*) – El prefijo con el que va a preceder el id del dataset en catálogo destino.
- **dataset_list** (*list(str)*) – Los ids de los datasets a federar. Si no se pasa una lista, todos los datasests se federan.
- **owner_org** (*str*) – La organización a la cual pertencen los datasets. Si no se pasa, se utiliza el catalog_id.
- **download_strategy** (*callable*) – Una función (catálogo, distribución)-> bool. Sobre las distribuciones que evalúa True, descarga el recurso en el downloadURL y lo sube al portal de destino. Por default no sube ninguna distribución.
- **origin_tz** (*str*) – Timezone de origen, un string (EJ: Africa/Bamako) el cual identifica el timezone del emisor del DataJson.
- **dst_tz** (*str*) – Timezone de destino, un string (EJ: Antarctica/Palmer) el cual identifica el timezone del receptor del DataJson, comunmente el timezone del servidor.

Devuelve El id del dataset en el catálogo de destino.

Tipo del valor devuelto str

harvest_dataset_to_ckan(*owner_org*, *dataset_origin_identifier*, *portal_url*, *apikey*, *catalog_id*, *download_strategy=None*, *origin_tz='America/Buenos_Aires'*, *dst_tz='America/Buenos_Aires'*)

Federa la metadata de un dataset en el portal pasado por parámetro.

Parámetros

- **catalog** (DataJson) – El catálogo de origen que contiene el dataset.
- **owner_org** (*str*) – La organización a la cual pertence el dataset.
- **dataset_origin_identifier** (*str*) – El id del dataset que se va a restaurar.

- **portal_url** (*str*) – La URL del portal CKAN de destino.
- **apikey** (*str*) – La apikey de un usuario con los permisos que le permitan crear o actualizar el dataset.
- **catalog_id** (*str*) – El id que prependea al dataset y recursos
- **download_strategy** (*callable*) – Una función (catálogo, distribución)-> bool. Sobre las distribuciones que evalúa True, descarga el recurso en el downloadURL y lo sube al portal de destino. Por default no sube ninguna distribución.
- **origin_tz** (*str*) – Timezone de origen, un string (EJ: Africa/Bamako) el cual identifica el timezone del emisor del DataJson.
- **dst_tz** (*str*) – Timezone de destino, un string (EJ: Antarctica/Palmer) el cual identifica el timezone del receptor del DataJson, comunmente el timezone del servidor.

Devuelve El id del dataset restaurado.

Tipo del valor devuelto str

is_valid_catalog(*catalog=None*)

Valida que un archivo *data.json* cumpla con el schema definido.

Chequea que el data.json tiene todos los campos obligatorios y que tanto los campos obligatorios como los opcionales siguen la estructura definida en el schema.

Parámetros **catalog** (*str o dict*) – Catálogo (dict, JSON o XLSX) a ser validado. Si no se pasa, valida este catálogo.

Devuelve True si el data.json cumple con el schema, sino False.

Tipo del valor devuelto bool

make_catalog_backup(*catalog_id=None, local_catalogs_dir=u", include_metadata=True, include_data=True, include_datasets=None, include_distribution_formats=[u'CSV', u'XLS'], include_metadata_xlsx=True, use_short_path=False*)

Realiza una copia local de los datos y metadatos de un catálogo.

Parámetros

- **catalog** (*dict or str*) – Representación externa/interna de un catálogo. Una representación _externa_ es un path local o una URL remota a un archivo con la metadata de un catálogo, en formato JSON o XLSX. La representación _interna_ de un catálogo es un diccionario.
- **catalog_id** (*str*) – Si se especifica, se usa este identificador para el backup. Si no se especifica, se usa catalog[«identifier»].
- **local_catalogs_dir** (*str*) – Directorio local en el cual se va a crear la carpeta «catalog/...» con todos los catálogos.
- **include_metadata** (*bool*) – Si es verdadero, se generan los archivos data.json y catalog.xlsx.
- **include_data** (*bool*) – Si es verdadero, se descargan todas las distribuciones de todos los catálogos.
- **include_datasets** (*list*) – Si se especifica, se descargan únicamente los datasets indicados. Si no, se descargan todos.
- **include_distribution_formats** (*list*) – Si se especifica, se descargan únicamente las distribuciones de los formatos indicados. Si no, se descargan todas.

- **use_short_path** (*bool*) – No implementado. Si es verdadero, se utiliza una jerarquía de directorios simplificada. Caso contrario, se replica la existente en infra.

Devuelve None

make_catalogs_backup(*catalogs=None*, *local_catalogs_dir=u'.'*, *copy_metadata=True*, *copy_data=True*)

Realiza copia de los datos y metadatos de uno o más catálogos.

Parámetros

- **catalogs** (*list or dict*) – Lista de catálogos (elementos que pueden ser interpretados por DataJson como catálogos) o diccionario donde las keys se interpretan como los catalog_identifier: { «modernizacion»: «http://infra.datos.gob.ar/catalog/modernizacion/data.json» } Cuando es una lista, los ids se toman de catalog_identifer, y se ignoran los catálogos que no tengan catalog_identifier. Cuando se pasa un diccionario, los keys reemplazan a los catalog_identifier (estos no se leeen).
- **local_catalogs_dir** (*str*) – Directorio local en el cual se va a crear la carpeta «catalog/...» con todos los catálogos.
- **copy_metadata** (*bool*) – Si es verdadero, se generan los archivos data.json y catalog.xlsx.
- **copy_data** (*bool*) – Si es verdadero, se descargan todas las distribuciones de todos los catálogos.

Devuelve None

push_dataset_to_ckan(*owner_org*, *dataset_origin_identifier*, *portal_url*, *apikey*, *catalog_id=None*, *demote_superThemes=True*, *demote_themes=True*, *download_strategy=None*, *generate_new_access_url=None*, *origin_tz='America/Buenos_Aires'*, *dst_tz='America/Buenos_Aires'*)

Escribe la metadata de un dataset en el portal pasado por parámetro.

Parámetros

- **catalog** (DataJson) – El catálogo de origen que contiene el dataset.
- **owner_org** (*str*) – La organización a la cual pertence el dataset.
- **dataset_origin_identifier** (*str*) – El id del dataset que se va a federar.
- **portal_url** (*str*) – La URL del portal CKAN de destino.
- **apikey** (*str*) – La apikey de un usuario con los permisos que le permitan crear o actualizar el dataset.
- **catalog_id** (*str or None*) – El prefijo con el que va a preceder el id del dataset en catálogo destino.
- **demote_superThemes** (*bool*) – Si está en true, los ids de los super themes del dataset, se propagan como grupo.
- **demote_themes** (*bool*) – Si está en true, los labels de los themes del dataset, pasan a ser tags. Sino, se pasan como grupo.
- **download_strategy** (*callable*) – Una función (catálogo, distribución)-> bool. Sobre las distribuciones que evalúa True, descarga el recurso en el downloadURL y lo sube al portal de destino. Por default no sube ninguna distribución.
- **generate_new_access_url** (*list*) – Se pasan los ids de las distribuciones cuyo accessURL se regenerar en el portal de destino. Para el resto, el portal debe mantiene el valor pasado en el DataJson.

- **origin_tz** (*str*) – Timezone de origen, un string (EJ: Africa/Bamako) el cual identifica el timezone del emisor del DataJson.
- **dst_tz** (*str*) – Timezone de destino, un string (EJ: Antarctica/Palmer) el cual identifica el timezone del receptor del DataJson, comunmente el timezone del servidor.

Devuelve El id del dataset en el catálogo de destino.

Tipo del valor devuelto str

push_new_themes(*portal_url*, *apikey*)
Toma un catálogo y escribe los temas de la taxonomía que no están presentes.

Args:

catalog (DataJson): El catálogo de origen que contiene la taxonomía.

portal_url (str): La URL del portal CKAN de destino. apikey (str): La apikey de un usuario con los permisos que le

permitan crear o actualizar los temas.

Returns: str: Los ids de los temas creados.

push_theme_to_ckan(*portal_url*, *apikey*, *identifier=None*, *label=None*)
Escribe la metadata de un theme en el portal pasado por parámetro.

Parámetros

- **catalog** (DataJson) – El catálogo de origen que contiene el theme.
- **portal_url** (*str*) – La URL del portal CKAN de destino.
- **apikey** (*str*) – La apikey de un usuario con los permisos que le permitan crear o actualizar el dataset.
- **identifier** (*str*) – El identificador para buscar el theme en la taxonomia.
- **label** (*str*) – El label para buscar el theme en la taxonomia.

Devuelve El name del theme en el catálogo de destino.

Tipo del valor devuelto str

restore_catalog_to_ckan(*origin_portal_url*, *destination_portal_url*, *apikey*, *download_strategy=None*, *generate_new_access_url=None*, *origin_tz='America/Buenos_Aires'*, *dst_tz='America/Buenos_Aires'*)
Restaura los datasets de un catálogo original al portal pasado por parámetro. Si hay temas presentes en el DataJson que no están en el portal de CKAN, los genera.

Args: catalog (DataJson): El catálogo de origen que se restaura. origin_portal_url (str): La URL del portal CKAN de origen. destination_portal_url (str): La URL del portal CKAN de

destino.

apikey (str): La apikey de un usuario con los permisos que le permitan crear o actualizar el dataset.

download_strategy(callable): Una función (catálogo, distribución)-> bool. Sobre las distribuciones que evalúa True, descarga el recurso en el downloadURL y lo sube al portal de destino. Por default no sube ninguna distribución.

generate_new_access_url(list): Se pasan los ids de las distribuciones cuyo accessURL se regenerar en el portal de destino. Para el resto, el portal debe mantiene el valor pasado en el DataJson.

origin_tz(str): Timezone de origen, un string (EJ: Africa/Bamako) el cual identifica el timezone del emisor del DataJson.

dst_tz(str): Timezone de destino, un string (EJ: Antarctica/Palmer) el cual identifica el timezone del receptor del DataJson, comunmente el timezone del servidor.

Returns:

dict: Diccionario con key organización y value la lista de ids de datasets subidos a esa organización

restore_dataset_to_ckan(*owner_org*, *dataset_origin_identifier*, *portal_url*, *apikey*, *download_strategy=None*, *generate_new_access_url=None*, *origin_tz='America/Buenos_Aires'*, *dst_tz='America/Buenos_Aires'*)

Restaura la metadata de un dataset en el portal pasado por parámetro.

Parámetros

- **catalog** (DataJson) – El catálogo de origen que contiene el dataset.
- **owner_org** (*str*) – La organización a la cual pertence el dataset.
- **dataset_origin_identifier** (*str*) – El id del dataset que se va a restaurar.
- **portal_url** (*str*) – La URL del portal CKAN de destino.
- **apikey** (*str*) – La apikey de un usuario con los permisos que le permitan crear o actualizar el dataset.
- **download_strategy** (*callable*) – Una función (catálogo, distribución)-> bool. Sobre las distribuciones que evalúa True, descarga el recurso en el downloadURL y lo sube al portal de destino. Por default no sube ninguna distribución.
- **generate_new_access_url** (*list*) – Se pasan los ids de las distribuciones cuyo accessURL se regenerar en el portal de destino. Para el resto, el portal debe mantiene el valor pasado en el DataJson.
- **origin_tz** (*str*) – Timezone de origen, un string (EJ: Africa/Bamako) el cual identifica el timezone del emisor del DataJson.
- **dst_tz** (*str*) – Timezone de destino, un string (EJ: Antarctica/Palmer) el cual identifica el timezone del receptor del DataJson, comunmente el timezone del servidor.

Devuelve El id del dataset restaurado.

Tipo del valor devuelto str

restore_organization_to_ckan(*owner_org*, *portal_url*, *apikey*, *dataset_list=None*, *download_strategy=None*, *generate_new_access_url=None*, *origin_tz='America/Buenos_Aires'*, *dst_tz='America/Buenos_Aires'*)

Restaura los datasets de la organización de un catálogo al portal pasado por parámetro. Si hay temas presentes en el DataJson que no están en el portal de CKAN, los genera.

Args: catalog (DataJson): El catálogo de origen que se restaura. portal_url (str): La URL del portal CKAN de destino. apikey (str): La apikey de un usuario con los permisos que le

permitan crear o actualizar el dataset.

dataset_list(list(str)): Los ids de los datasets a restaurar. Si no se pasa una lista, todos los datasests se restauran.

owner_org (str): La organización a la cual pertencen los datasets. download_strategy(callable): Una función (catálogo, distribución)->

bool. Sobre las distribuciones que evalúa True, descarga el recurso en el downloadURL y lo sube al portal de destino. Por default no sube ninguna distribución.

generate_new_access_url(list): Se pasan los ids de las distribuciones cuyo accessURL se regenerar en el portal de destino. Para el resto, el portal debe mantiene el valor pasado en el DataJson.

origin_tz(str): Timezone de origen, un string (EJ: Africa/Bamako) el cual identifica el timezone del emisor del DataJson.

dst_tz(str): Timezone de destino, un string (EJ: Antarctica/Palmer) el cual identifica el timezone del receptor del DataJson, comunmente el timezone del servidor.

Returns: list(str): La lista de ids de datasets subidos.

restore_organizations_to_ckan(*organizations*, *portal_url*, *apikey*, *download_strategy=None*, *generate_new_access_url=None*, *origin_tz='America/Buenos_Aires'*, *dst_tz='America/Buenos_Aires'*)

Restaura los datasets indicados para c/organización de un catálogo al portal pasado. Si hay temas presentes en el DataJson que no están en el portal de CKAN, los genera. Las organizaciones ya deben estar creadas.

Parámetros

- **catalog** (DataJson) – El catálogo de origen que se restaura.
- **organizations** (*dict*) – Datasets a restaurar por c/organización donde {«organizacion_id»: [dataset_id1, dataset_id2, . . .]}
- **portal_url** (*str*) – La URL del portal CKAN de destino.
- **apikey** (*str*) – La apikey de un usuario con los permisos que le permitan crear o actualizar el dataset.
- **download_strategy** (*callable*) – Una función (catálogo, distribución)-> bool. Sobre las distribuciones que evalúa True, descarga el recurso en el downloadURL y lo sube al portal de destino. Por default no sube ninguna distribución.
- **generate_new_access_url** (*list*) – Se pasan los ids de las distribuciones cuyo accessURL se regenerar en el portal de destino. Para el resto, el portal debe mantiene el valor pasado en el DataJson.
- **origin_tz** (*str*) – Timezone de origen, un string (EJ: Africa/Bamako) el cual identifica el timezone del emisor del DataJson.
- **dst_tz** (*str*) – Timezone de destino, un string (EJ: Antarctica/Palmer) el cual identifica el timezone del receptor del DataJson, comunmente el timezone del servidor.

Devuelve La lista de ids de datasets subidos.

Tipo del valor devuelto list(str)

themes

Devuelve la lista de temas del catálogo (taxonomía temática).

time_series

Devuelve lista de series de tiempo del catálogo o uno de sus metadatos.

Parámetros

- **catalog** (*dict, str or DataJson*) – Representación externa/interna de un catálogo. Una representación _externa_ es un path local o una URL remota a un archivo con la metadata de un catálogo, en formato JSON o XLSX. La representación _interna_ de un catálogo es un diccionario. Ejemplos: http://datos.gob.ar/data.json, http://www.ign.gob.ar/descargas/geodatos/catalog.xlsx, «/energia/catalog.xlsx».
- **filter_in** (*dict*) – Devuelve las series cuyos atributos coinciden con los pasados en este diccionario. Ejemplo:

```
{
    "dataset": {
        "publisher": {"name": "Ministerio de Ambiente"}
    }
}
```

  Sólo se devolverán las series que pertenezcan a un dataset de ese publisher_name.
- **filter_out** (*dict*) – Devuelve las series cuyos atributos no coinciden con los pasados en este diccionario. Ejemplo:

```
{
    "dataset": {
        "publisher": {"name": "Ministerio de Ambiente"}
    }
}
```

  Sólo se devolverán las series que no pertenezcan a un dataset de ese publisher_name.
- **meta_field** (*str*) – Nombre de un metadato de Field. En lugar de devolver los objetos completos «Field», devuelve una lista de valores para ese metadato presentes en el catálogo.
- **exclude_meta_fields** (*list*) – Metadatos de Field que se quieren excluir de los objetos Field devueltos.

to_json(*path*)
Escribe el catálogo en JSON.

Parámetros

- **catalog** (DataJson) – Catálogo de datos.
- **path** (*str*) – Directorio absoluto donde se crea el archivo XLSX.

to_xlsx(*path*, *xlsx_fields=None*)
Escribe el catálogo en Excel.

Parámetros

- **catalog** (DataJson) – Catálogo de datos.
- **path** (*str*) – Directorio absoluto donde se crea el archivo XLSX.
- **xlsx_fields** (*dict*) – Orden en que los campos del perfil de metadatos se escriben en cada hoja del Excel.

validate_catalog(*catalog=None*, *only_errors=False*, *fmt=u'dict'*, *export_path=None*)
Analiza un data.json registrando los errores que encuentra.

Chequea que el data.json tiene todos los campos obligatorios y que tanto los campos obligatorios como los opcionales siguen la estructura definida en el schema.

Parámetros

- **catalog** (*str o dict*) – Catálogo (dict, JSON o XLSX) a ser validado. Si no se pasa, valida este catálogo.
- **only_errors** (*bool*) – Si es True sólo se reportan los errores.
- **fmt** (*str*) – Indica el formato en el que se desea el reporte. «dict» es el reporte más verborrágico respetando la

  estructura del data.json.

  »list» devuelve un dict con listas de errores formateados para generar tablas.

- **export_path** (*str*) – Path donde exportar el reporte generado (en formato XLSX o CSV). Si se especifica, el método no devolverá nada, a pesar de que se pase algún argumento en *fmt*.

Devuelve

Diccionario resumen de los errores encontrados:

```
{
    "status": "OK",  # resultado de la validación global
    "error": {
        "catalog": {
            "status": "OK",
            "errors": []
            "title": "Título Catalog"},
        "dataset": [
            {
                "status": "OK",
                "errors": [],
                "title": "Titulo Dataset 1"
            },
            {
                "status": "ERROR",
                "errors": [error1_info, error2_info, ...],
                "title": "Titulo Dataset 2"
            }
        ]
    }
}
```

Donde errorN_info es un dict con la información del N-ésimo error encontrado, con las siguientes claves: «path», «instance», «message», «validator», «validator_value», «error_code».

Tipo del valor devuelto dict

pydatajson.core.**main**()

Permite ejecutar el módulo por línea de comandos.

Valida un path o url a un archivo data.json devolviendo True/False si es válido y luego el resultado completo.

Example

python pydatajson.py http://181.209.63.71/data.json python pydatajson.py ~/github/pydatajson/tests/samples/full_data.json

2.1.2 Lectura

Módulo "readers" de Pydatajson

Contiene los métodos auxiliares para leer archivos con información tabular y catálogos de metadatos, en distintos fomatos.

pydatajson.readers.**read_catalog**(*catalog*, *default_values=None*, *catalog_format=None*, *verify=False*, *timeout=30*)

Toma una representación cualquiera de un catálogo, y devuelve su representación interna (un diccionario de Python con su metadata.)

Si recibe una representación _interna_ (un diccionario), lo devuelve intacto. Si recibe una representación _externa_ (path/URL a un archivo JSON/XLSX), devuelve su represetación interna, es decir, un diccionario.

Parámetros **catalog** (*dict or str*) – Representación externa/interna de un catálogo. Una representación _externa_ es un path local o una URL remota a un archivo con la metadata de un catálogo, en formato JSON o XLSX. La representación _interna_ de un catálogo es un diccionario.

Devuelve Representación interna de un catálogo para uso en las funciones de esta librería.

Tipo del valor devuelto dict

pydatajson.readers.**read_json**(*json_path_or_url*, *verify=False*, *timeout=30*)

Toma el path a un JSON y devuelve el diccionario que representa.

Se asume que el parámetro es una URL si comienza con "http" o "https", o un path local de lo contrario.

Parámetros **json_path_or_url** (*str*) – Path local o URL remota a un archivo de texto plano en formato JSON.

Devuelve El diccionario que resulta de deserializar json_path_or_url.

Tipo del valor devuelto dict

pydatajson.readers.**read_local_xlsx_catalog**(*xlsx_path*, *logger=None*)

Genera un diccionario de metadatos de catálogo a partir de un XLSX bien formado.

Parámetros **xlsx_path** (*str*) – Path a un archivo XLSX «template» para describir la metadata de un catálogo.

Devuelve Diccionario con los metadatos de un catálogo.

Tipo del valor devuelto dict

pydatajson.readers.**read_table**(*path*)

Lee un archivo tabular (CSV o XLSX) a una lista de diccionarios.

La extensión del archivo debe ser «.csv» o «.xlsx». En función de ella se decidirá el método a usar para leerlo.

Si recibe una lista, comprueba que todos sus diccionarios tengan las mismas claves y de ser así, la devuelve intacta. Levanta una Excepción en caso contrario.

Parámetros **path** (*str o list*) – Como "str", path a un archivo CSV o XLSX.

Devuelve Lista de diccionarios con claves idénticas representando el archivo original.

Tipo del valor devuelto list

pydatajson.readers.**read_xlsx_catalog**(*xlsx_path_or_url*, *logger=None*, *verify=False*, *timeout=30*)

Toma el path a un catálogo en formato XLSX y devuelve el diccionario que representa.

Se asume que el parámetro es una URL si comienza con "http" o "https", o un path local de lo contrario.

Parámetros **xlsx_path_or_url** (*str*) – Path local o URL remota a un libro XLSX de formato específico para guardar los metadatos de un catálogo.

Devuelve El diccionario que resulta de procesar xlsx_path_or_url.

Tipo del valor devuelto dict

2.1.3 Escritura

Módulo "writers" de pydatajson

Contiene los métodos para escribir - diccionarios con metadatos de catálogos a formato JSON, así como - listas de diccionarios («tablas») en formato CSV o XLSX

pydatajson.writers.**write_json**(*obj*, *path*)
Escribo un objeto a un archivo JSON con codificación UTF-8.

pydatajson.writers.**write_json_catalog**(*catalog*, *path*)
Escribe el catálogo en JSON.

Parámetros

- **catalog** (DataJson) – Catálogo de datos.
- **path** (*str*) – Directorio absoluto donde se crea el archivo XLSX.

pydatajson.writers.**write_table**(*table*, *path*, *column_styles=None*, *cell_styles=None*)
Exporta una tabla en el formato deseado (CSV o XLSX).

La extensión del archivo debe ser «.csv» o «.xlsx», y en función de ella se decidirá qué método usar para escribirlo.

Parámetros

- **table** (*list of dicts*) – Tabla a ser exportada.
- **path** (*str*) – Path al archivo CSV o XLSX de exportación.

pydatajson.writers.**write_tables**(*tables*, *path*, *column_styles=None*, *cell_styles=None*, *tables_fields=None*, *tables_names=None*)
Exporta un reporte con varias tablas en CSV o XLSX.

Si la extensión es «.csv» se crean varias tablas agregando el nombre de la tabla al final del «path». Si la extensión es «.xlsx» todas las tablas se escriben en el mismo excel.

Parámetros

- **table** (*dict of (list of dicts)*) – Conjunto de tablas a ser exportadas donde {

  »table_name»: [{ «field_name1»: «field_value1», «field_name2»: «field_value2», «field_name3»: «field_value3»

  }]

  }
- **path** (*str*) – Path al archivo CSV o XLSX de exportación.

pydatajson.writers.**write_xlsx_catalog**(*catalog*, *path*, *xlsx_fields=None*)
Escribe el catálogo en Excel.

Parámetros

- **catalog** (DataJson) – Catálogo de datos.
- **path** (*str*) – Directorio absoluto donde se crea el archivo XLSX.

- **xlsx_fields** (*dict*) – Orden en que los campos del perfil de metadatos se escriben en cada hoja del Excel.

2.1.4 Validación

Módulo "validator" de Pydatajson

Contiene los métodos para validar el perfil de metadatos de un catálogo.

pydatajson.validation.**is_valid_catalog**(*catalog*, *validator=None*)

Valida que un archivo *data.json* cumpla con el schema definido.

Chequea que el data.json tiene todos los campos obligatorios y que tanto los campos obligatorios como los opcionales siguen la estructura definida en el schema.

Parámetros **catalog** (*str o dict*) – Catálogo (dict, JSON o XLSX) a ser validado.

Devuelve True si el data.json cumple con el schema, sino False.

Tipo del valor devuelto bool

pydatajson.validation.**validate_catalog**(*catalog*, *only_errors=False*, *fmt=u'dict'*, *export_path=None*, *validator=None*)

Analiza un data.json registrando los errores que encuentra.

Chequea que el data.json tiene todos los campos obligatorios y que tanto los campos obligatorios como los opcionales siguen la estructura definida en el schema.

Parámetros

- **catalog** (*str o dict*) – Catálogo (dict, JSON o XLSX) a ser validado.
- **only_errors** (*bool*) – Si es True sólo se reportan los errores.
- **fmt** (*str*) – Indica el formato en el que se desea el reporte. «dict» es el reporte más verborrágico respetando la

  estructura del data.json.

  »list» devuelve un dict con listas de errores formateados para generar tablas.

- **export_path** (*str*) – Path donde exportar el reporte generado (en formato XLSX o CSV). Si se especifica, el método no devolverá nada, a pesar de que se pase algún argumento en *fmt*.

Devuelve

Diccionario resumen de los errores encontrados:

```
{
    "status": "OK",  # resultado de la validación global
    "error": {
        "catalog": {
            "status": "OK",
            "errors": []
            "title": "Título Catalog"},
        "dataset": [
            {
                "status": "OK",
                "errors": [],
                "title": "Titulo Dataset 1"
```

(continues on next page)

(proviene de la página anterior)

```
            },
            {
                "status": "ERROR",
                "errors": [error1_info, error2_info, ...],
                "title": "Titulo Dataset 2"
            }
        ]
    }
}
```

Donde errorN_info es un dict con la información del N-ésimo error encontrado, con las siguientes claves: «path», «instance», «message», «validator», «validator_value», «error_code».

Tipo del valor devuelto dict

2.1.5 Búsqueda

Módulo "search" de Pydatajson

Contiene los métodos para navegar un data.json iterando y buscando entidades de un catálogo.

pydatajson.search.**get_catalog_metadata**(*catalog*, *exclude_meta_fields=None*)
Devuelve sólo la metadata de nivel catálogo.

pydatajson.search.**get_dataset**(*catalog*, *identifier=None*, *title=None*)
Devuelve un Dataset del catálogo.

pydatajson.search.**get_datasets**(*catalog*, *filter_in=None*, *filter_out=None*, *meta_field=None*, *exclude_meta_fields=None*, *only_time_series=False*)
Devuelve una lista de datasets del catálogo o de uno de sus metadatos.

Parámetros

- **catalog** (*dict, str or* DataJson) – Representación externa/interna de un catálogo. Una representación _externa_ es un path local o una URL remota a un archivo con la metadata de un catálogo, en formato JSON o XLSX. La representación _interna_ de un catálogo es un diccionario. Ejemplos: http://datos.gob.ar/data.json, http://www.ign.gob.ar/descargas/geodatos/catalog.xlsx, «/energia/catalog.xlsx».
- **filter_in** (*dict*) – Devuelve los datasets cuyos atributos coinciden con los pasados en este diccionario. Ejemplo:

```
{
    "dataset": {
        "publisher": {"name": "Ministerio de Ambiente"}
    }
}
```

Sólo se devolverán los datasets de ese publisher_name.

- **filter_out** (*dict*) – Devuelve los datasets cuyos atributos no coinciden con los pasados en este diccionario. Ejemplo:

```
{
    "dataset": {
        "publisher": {"name": "Ministerio de Ambiente"}
    }
}
```

Sólo se devolverán los datasets que no sean de ese publisher_name.

- **meta_field** (*str*) – Nombre de un metadato de Dataset. En lugar de devolver los objetos completos «Dataset», devuelve una lista de valores para ese metadato presentes en el catálogo.
- **exclude_meta_fields** (*list*) – Metadatos de Dataset que se quieren excluir de los objetos Dataset devueltos.
- **only_time_series** (*bool*) – Si es verdadero, sólo devuelve datasets que tengan por lo menos una distribución de series de tiempo.

pydatajson.search.**get_distribution**(*catalog*, *identifier=None*, *title=None*, *dataset_identifier=None*)

Devuelve una Distribution del catálogo.

pydatajson.search.**get_distributions**(*catalog*, *filter_in=None*, *filter_out=None*, *meta_field=None*, *exclude_meta_fields=None*, *only_time_series=False*)

Devuelve lista de distribuciones del catálogo o de uno de sus metadatos.

Parámetros

- **catalog** (*dict, str or DataJson*) – Representación externa/interna de un catálogo. Una representación _externa_ es un path local o una URL remota a un archivo con la metadata de un catálogo, en formato JSON o XLSX. La representación _interna_ de un catálogo es un diccionario. Ejemplos: http://datos.gob.ar/data.json, http://www.ign.gob.ar/descargas/geodatos/catalog.xlsx, «/energia/catalog.xlsx».
- **filter_in** (*dict*) – Devuelve los distribuciones cuyos atributos coinciden con los pasados en este diccionario. Ejemplo:

```
{
    "dataset": {
        "publisher": {"name": "Ministerio de Ambiente"}
    }
}
```

  Sólo se devolverán los distribuciones que pertenezcan a un dataset de ese publisher_name.
- **filter_out** (*dict*) – Devuelve los distribuciones cuyos atributos no coinciden con los pasados en este diccionario. Ejemplo:

```
{
    "dataset": {
        "publisher": {"name": "Ministerio de Ambiente"}
    }
}
```

  Sólo se devolverán los distribuciones que no pertenezcan a un dataset de ese publisher_name.
- **meta_field** (*str*) – Nombre de un metadato de Distribution. En lugar de devolver los objetos completos Distribution, devuelve una lista de valores para ese metadato presentes en el catálogo.
- **exclude_meta_fields** (*list*) – Metadatos de Distribution que se quieren excluir de los objetos Distribution devueltos.
- **only_time_series** (*bool*) – Si es verdadero, sólo devuelve distribuciones que sean distribuciones de series de tiempo.

pydatajson.search.**get_field**(*catalog*, *identifier=None*, *title=None*, *distribution_identifier=None*)
Devuelve un Field del catálogo.

pydatajson.search.**get_fields**(*catalog*, *filter_in=None*, *filter_out=None*, *meta_field=None*, *only_time_series=False*, *distribution_identifier=None*)
Devuelve lista de campos del catálogo o de uno de sus metadatos.

Parámetros

- **catalog** (*dict, str or DataJson*) – Representación externa/interna de un catálogo. Una representación _externa_ es un path local o una URL remota a un archivo con la metadata de un catálogo, en formato JSON o XLSX. La representación _interna_ de un catálogo es un diccionario. Ejemplos: http://datos.gob.ar/data.json, http://www.ign.gob.ar/descargas/geodatos/catalog.xlsx, «/energia/catalog.xlsx».
- **filter_in** (*dict*) – Devuelve los campos cuyos atributos coinciden con los pasados en este diccionario. Ejemplo:

```
{
    "dataset": {
        "publisher": {"name": "Ministerio de Ambiente"}
    }
}
```

  Sólo se devolverán los campos que pertenezcan a un dataset de ese publisher_name.
- **filter_out** (*dict*) – Devuelve los campos cuyos atributos no coinciden con los pasados en este diccionario. Ejemplo:

```
{
    "dataset": {
        "publisher": {"name": "Ministerio de Ambiente"}
    }
}
```

  Sólo se devolverán los campos que no pertenezcan a un dataset de ese publisher_name.
- **meta_field** (*str*) – Nombre de un metadato de Field. En lugar de devolver los objetos completos «Field», devuelve una lista de valores para ese metadato presentes en el catálogo.
- **exclude_meta_fields** (*list*) – Metadatos de Field que se quieren excluir de los objetos Field devueltos.
- **only_time_series** (*bool*) – Si es verdadero, sólo devuelve campos que sean series de tiempo.

pydatajson.search.**get_themes**(*catalog*)
Devuelve la lista de temas del catálogo (taxonomía temática).

pydatajson.search.**get_time_series**(*catalog*, ***kwargs*)
Devuelve lista de series de tiempo del catálogo o uno de sus metadatos.

Parámetros

- **catalog** (*dict, str or DataJson*) – Representación externa/interna de un catálogo. Una representación _externa_ es un path local o una URL remota a un archivo con la metadata de un catálogo, en formato JSON o XLSX. La representación _interna_ de un catálogo es un diccionario. Ejemplos: http://datos.gob.ar/data.json, http://www.ign.gob.ar/descargas/geodatos/catalog.xlsx, «/energia/catalog.xlsx».
- **filter_in** (*dict*) – Devuelve las series cuyos atributos coinciden con los pasados en este diccionario. Ejemplo:

```
{
    "dataset": {
        "publisher": {"name": "Ministerio de Ambiente"}
    }
}
```

Sólo se devolverán las series que pertenezcan a un dataset de ese publisher_name.

- **filter_out** (*dict*) – Devuelve las series cuyos atributos no coinciden con los pasados en este diccionario. Ejemplo:

```
{
    "dataset": {
        "publisher": {"name": "Ministerio de Ambiente"}
    }
}
```

Sólo se devolverán las series que no pertenezcan a un dataset de ese publisher_name.

- **meta_field** (*str*) – Nombre de un metadato de Field. En lugar de devolver los objetos completos «Field», devuelve una lista de valores para ese metadato presentes en el catálogo.
- **exclude_meta_fields** (*list*) – Metadatos de Field que se quieren excluir de los objetos Field devueltos.

2.1.6 Reportes

Módulo "reporting" de Pydatajson

Contiene los métodos para generar reportes sobre un catálogo.

pydatajson.reporting.**generate_datasets_summary**(*catalog*, *export_path=None*, *validator=None*)

Genera un informe sobre los datasets presentes en un catálogo, indicando para cada uno:

- Índice en la lista catalog[«dataset»]
- Título
- Identificador
- Cantidad de distribuciones
- Estado de sus metadatos [«OK»|»ERROR»]

Es utilizada por la rutina diaria de *libreria-catalogos* para reportar sobre los datasets de los catálogos mantenidos.

Parámetros

- **catalog** (*str o dict*) – Path a un catálogo en cualquier formato, JSON, XLSX, o diccionario de python.
- **export_path** (*str*) – Path donde exportar el informe generado (en formato XLSX o CSV). Si se especifica, el método no devolverá nada.

Devuelve Contiene tantos dicts como datasets estén presentes en *catalogs*, con los datos antes mencionados.

Tipo del valor devuelto list

2.1.7 Federación

Extensión de pydatajson para la federación de metadatos de datasets a través de la API de CKAN.

pydatajson.federation.**get_organization_from_ckan**(*portal_url*, *org_id*, *verify_ssl=False*, *requests_timeout=30*)

Toma la url de un portal y un id, y devuelve la organización a buscar.

Parámetros

- **portal_url** (*str*) – La URL del portal CKAN de origen.
- **org_id** (*str*) – El id de la organización a buscar.
- **verify_ssl** (*bool*) – Verificar certificados SSL
- **requests_timeout** (*int*) – cantidad en segundos para timeoutear un request al server.

Devuelve Diccionario con la información de la organización.

Tipo del valor devuelto dict

pydatajson.federation.**get_organizations_from_ckan**(*portal_url*, *verify_ssl=False*, *requests_timeout=30*)

Toma la url de un portal y devuelve su árbol de organizaciones.

Parámetros

- **portal_url** (*str*) – La URL del portal CKAN de origen.
- **verify_ssl** (*bool*) – Verificar certificados SSL
- **requests_timeout** (*int*) – cantidad en segundos para timeoutear un request al server.

Devuelve Lista de diccionarios anidados con la información de las organizaciones.

Tipo del valor devuelto list

pydatajson.federation.**harvest_catalog_to_ckan**(*catalog*, *portal_url*, *apikey*, *catalog_id*, *dataset_list=None*, *owner_org=None*, *download_strategy=None*, *origin_tz='America/Buenos_Aires'*, *dst_tz='America/Buenos_Aires'*)

Federa los datasets de un catálogo al portal pasado por parámetro.

Parámetros

- **catalog** (DataJson) – El catálogo de origen que se federa.
- **portal_url** (*str*) – La URL del portal CKAN de destino.
- **apikey** (*str*) – La apikey de un usuario con los permisos que le permitan crear o actualizar el dataset.
- **catalog_id** (*str*) – El prefijo con el que va a preceder el id del dataset en catálogo destino.
- **dataset_list** (*list(str)*) – Los ids de los datasets a federar. Si no se pasa una lista, todos los datasests se federan.
- **owner_org** (*str*) – La organización a la cual pertencen los datasets. Si no se pasa, se utiliza el catalog_id.
- **download_strategy** (*callable*) – Una función (catálogo, distribución)-> bool. Sobre las distribuciones que evalúa True, descarga el recurso en el downloadURL y lo sube al portal de destino. Por default no sube ninguna distribución.

- **origin_tz** (*str*) – Timezone de origen, un string (EJ: Africa/Bamako) el cual identifica el timezone del emisor del DataJson.
- **dst_tz** (*str*) – Timezone de destino, un string (EJ: Antarctica/Palmer) el cual identifica el timezone del receptor del DataJson, comunmente el timezone del servidor.

Devuelve El id del dataset en el catálogo de destino.

Tipo del valor devuelto str

pydatajson.federation.**harvest_dataset_to_ckan**(*catalog*, *owner_org*, *dataset_origin_identifier*, *portal_url*, *apikey*, *catalog_id*, *download_strategy=None*, *origin_tz='America/Buenos_Aires'*, *dst_tz='America/Buenos_Aires'*)

Federa la metadata de un dataset en el portal pasado por parámetro.

Parámetros

- **catalog** (DataJson) – El catálogo de origen que contiene el dataset.
- **owner_org** (*str*) – La organización a la cual pertence el dataset.
- **dataset_origin_identifier** (*str*) – El id del dataset que se va a restaurar.
- **portal_url** (*str*) – La URL del portal CKAN de destino.
- **apikey** (*str*) – La apikey de un usuario con los permisos que le permitan crear o actualizar el dataset.
- **catalog_id** (*str*) – El id que prependea al dataset y recursos
- **download_strategy** (*callable*) – Una función (catálogo, distribución)-> bool. Sobre las distribuciones que evalúa True, descarga el recurso en el downloadURL y lo sube al portal de destino. Por default no sube ninguna distribución.
- **origin_tz** (*str*) – Timezone de origen, un string (EJ: Africa/Bamako) el cual identifica el timezone del emisor del DataJson.
- **dst_tz** (*str*) – Timezone de destino, un string (EJ: Antarctica/Palmer) el cual identifica el timezone del receptor del DataJson, comunmente el timezone del servidor.

Devuelve El id del dataset restaurado.

Tipo del valor devuelto str

pydatajson.federation.**push_dataset_to_ckan**(*catalog*, *owner_org*, *dataset_origin_identifier*, *portal_url*, *apikey*, *catalog_id=None*, *demote_superThemes=True*, *demote_themes=True*, *download_strategy=None*, *generate_new_access_url=None*, *origin_tz='America/Buenos_Aires'*, *dst_tz='America/Buenos_Aires'*)

Escribe la metadata de un dataset en el portal pasado por parámetro.

Parámetros

- **catalog** (DataJson) – El catálogo de origen que contiene el dataset.
- **owner_org** (*str*) – La organización a la cual pertence el dataset.
- **dataset_origin_identifier** (*str*) – El id del dataset que se va a federar.
- **portal_url** (*str*) – La URL del portal CKAN de destino.

- **apikey** (*str*) – La apikey de un usuario con los permisos que le permitan crear o actualizar el dataset.
- **catalog_id** (*str or None*) – El prefijo con el que va a preceder el id del dataset en catálogo destino.
- **demote_superThemes** (*bool*) – Si está en true, los ids de los super themes del dataset, se propagan como grupo.
- **demote_themes** (*bool*) – Si está en true, los labels de los themes del dataset, pasan a ser tags. Sino, se pasan como grupo.
- **download_strategy** (*callable*) – Una función (catálogo, distribución)-> bool. Sobre las distribuciones que evalúa True, descarga el recurso en el downloadURL y lo sube al portal de destino. Por default no sube ninguna distribución.
- **generate_new_access_url** (*list*) – Se pasan los ids de las distribuciones cuyo accessURL se regenerar en el portal de destino. Para el resto, el portal debe mantiene el valor pasado en el DataJson.
- **origin_tz** (*str*) – Timezone de origen, un string (EJ: Africa/Bamako) el cual identifica el timezone del emisor del DataJson.
- **dst_tz** (*str*) – Timezone de destino, un string (EJ: Antarctica/Palmer) el cual identifica el timezone del receptor del DataJson, comunmente el timezone del servidor.

Devuelve El id del dataset en el catálogo de destino.

Tipo del valor devuelto str

pydatajson.federation.**push_new_themes**(*catalog*, *portal_url*, *apikey*)

Toma un catálogo y escribe los temas de la taxonomía que no están presentes.

Args:

catalog (DataJson): El catálogo de origen que contiene la taxonomía.

portal_url (str): La URL del portal CKAN de destino. apikey (str): La apikey de un usuario con los permisos que le

permitan crear o actualizar los temas.

Returns: str: Los ids de los temas creados.

pydatajson.federation.**push_organization_to_ckan**(*portal_url*, *apikey*, *organization*, *parent=None*, *verify_ssl=False*, *requests_timeout=30*)

Toma una organización y la crea en el portal de destino. :param portal_url: La URL del portal CKAN de destino. :type portal_url: str :param apikey: La apikey de un usuario con los permisos que le

permitan crear la organización.

Parámetros

- **organization** (*dict*) – Datos de la organización a crear.
- **parent** (*str*) – Campo name de la organización padre.
- **verify_ssl** (*bool*) – Verificar certificados SSL
- **requests_timeout** (*int*) – cantidad en segundos para timeoutear un request al server.

Devuelve

Devuelve el diccionario de la organizacion enviada, junto con el status detallando si la creación fue exitosa o no.

Tipo del valor devuelto (dict)

pydatajson.federation.**push_organization_tree_to_ckan**(*portal_url*, *apikey*, *org_tree*, *parent=None*)

Toma un árbol de organizaciones y lo replica en el portal de destino.

Args: portal_url (str): La URL del portal CKAN de destino. apikey (str): La apikey de un usuario con los permisos que le

permitan crear las organizaciones.

org_tree(list): lista de diccionarios con la data de las organizaciones a crear.

parent(str): campo name de la organizacion padre.

Returns:

(list): Devuelve el arbol de organizaciones recorridas, junto con el status detallando si la creación fue exitosa o no.

pydatajson.federation.**push_theme_to_ckan**(*catalog*, *portal_url*, *apikey*, *identifier=None*, *label=None*)

Escribe la metadata de un theme en el portal pasado por parámetro.

Parámetros

- **catalog** (DataJson) – El catálogo de origen que contiene el theme.
- **portal_url** (*str*) – La URL del portal CKAN de destino.
- **apikey** (*str*) – La apikey de un usuario con los permisos que le permitan crear o actualizar el dataset.
- **identifier** (*str*) – El identificador para buscar el theme en la taxonomia.
- **label** (*str*) – El label para buscar el theme en la taxonomia.

Devuelve El name del theme en el catálogo de destino.

Tipo del valor devuelto str

pydatajson.federation.**remove_datasets_from_ckan**(*portal_url*, *apikey*, *filter_in=None*, *filter_out=None*, *only_time_series=False*, *organization=None*, *verify_ssl=False*, *requests_timeout=30*)

Borra un dataset en el portal pasado por parámetro.

Parámetros

- **portal_url** (*str*) – La URL del portal CKAN de destino.
- **apikey** (*str*) – La apikey de un usuario con los permisos que le permitan borrar el dataset.
- **filter_in** (*dict*) – Diccionario de filtrado positivo, similar al de search.get_datasets.
- **filter_out** (*dict*) – Diccionario de filtrado negativo, similar al de search.get_datasets.
- **only_time_series** (*bool*) – Filtrar solo los datasets que tengan recursos con series de tiempo.
- **organization** (*str*) – Filtrar solo los datasets que pertenezcan a cierta organizacion.
- **verify_ssl** (*bool*) – Verificar certificados SSL
- **requests_timeout** (*int*) – cantidad en segundos para timeoutear un

- **al server.** (*request*) –

Devuelve None

pydatajson.federation.**remove_organization_from_ckan**(*portal_url*, *apikey*, *organization_id*, *verify_ssl=False*, *requests_timeout=30*)

Toma un id de organización y la purga del portal de destino. :param portal_url: La URL del portal CKAN de destino. :type portal_url: str :param apikey: La apikey de un usuario con los permisos que le

permitan borrar la organización.

Parámetros

- **organization_id** (*str*) – Id o name de la organización a borrar.
- **verify_ssl** (*bool*) – Verificar certificados SSL
- **requests_timeout** (*int*) – cantidad en segundos para timeoutear un request al server.

Devuelve None.

pydatajson.federation.**remove_organizations_from_ckan**(*portal_url*, *apikey*, *organization_list*)

Toma una lista de ids de organización y las purga del portal de destino. :param portal_url: La URL del portal CKAN de destino. :type portal_url: str :param apikey: La apikey de un usuario con los permisos que le

permitan borrar la organización.

Parámetros **organization_list** (*list*) – Id o name de las organizaciones a borrar.

Devuelve None.

pydatajson.federation.**resources_update**(*portal_url*, *apikey*, *distributions*, *resource_files*, *generate_new_access_url=None*, *catalog_id=None*, *verify_ssl=False*, *requests_timeout=30*)

Sube archivos locales a sus distribuciones correspondientes en el portal pasado por parámetro.

Args: portal_url (str): La URL del portal CKAN de destino. apikey (str): La apikey de un usuario con los permisos que le

permitan crear o actualizar el dataset.

distributions(list): Lista de distribuciones posibles para actualizar.

resource_files(dict): Diccionario con entradas id_de_distribucion:path_al_recurso a subir

generate_new_access_url(list): Lista de ids de distribuciones a las cuales se actualizará el accessURL con los valores generados por el portal de destino

catalog_id(str): prependea el id al id del recurso para encontrarlo antes de subirlo

verify_ssl(bool): Verificar certificados SSL requests_timeout(int): cantidad en segundos para timeoutear un request al server.

Returns: list: los ids de los recursos modificados

pydatajson.federation.**restore_catalog_to_ckan**(*catalog*, *origin_portal_url*, *destination_portal_url*, *apikey*, *download_strategy=None*, *generate_new_access_url=None*, *origin_tz='America/Buenos_Aires'*, *dst_tz='America/Buenos_Aires'*)

Restaura los datasets de un catálogo original al portal pasado por parámetro. Si hay temas presentes en el DataJson que no están en el portal de CKAN, los genera.

Args: catalog (DataJson): El catálogo de origen que se restaura. origin_portal_url (str): La URL del portal CKAN de origen. destination_portal_url (str): La URL del portal CKAN de

destino.

apikey (str): La apikey de un usuario con los permisos que le permitan crear o actualizar el dataset.

download_strategy(callable): Una función (catálogo, distribución)-> bool. Sobre las distribuciones que evalúa True, descarga el recurso en el downloadURL y lo sube al portal de destino. Por default no sube ninguna distribución.

generate_new_access_url(list): Se pasan los ids de las distribuciones cuyo accessURL se regenerar en el portal de destino. Para el resto, el portal debe mantiene el valor pasado en el DataJson.

origin_tz(str): Timezone de origen, un string (EJ: Africa/Bamako) el cual identifica el timezone del emisor del DataJson.

dst_tz(str): Timezone de destino, un string (EJ: Antarctica/Palmer) el cual identifica el timezone del receptor del DataJson, comunmente el timezone del servidor.

Returns:

dict: Diccionario con key organización y value la lista de ids de datasets subidos a esa organización

pydatajson.federation.**restore_dataset_to_ckan**(*catalog*, *owner_org*, *dataset_origin_identifier*, *portal_url*, *apikey*, *download_strategy=None*, *generate_new_access_url=None*, *origin_tz='America/Buenos_Aires'*, *dst_tz='America/Buenos_Aires'*)

Restaura la metadata de un dataset en el portal pasado por parámetro.

Parámetros

- **catalog** (DataJson) – El catálogo de origen que contiene el dataset.
- **owner_org** (*str*) – La organización a la cual pertence el dataset.
- **dataset_origin_identifier** (*str*) – El id del dataset que se va a restaurar.
- **portal_url** (*str*) – La URL del portal CKAN de destino.
- **apikey** (*str*) – La apikey de un usuario con los permisos que le permitan crear o actualizar el dataset.
- **download_strategy** (*callable*) – Una función (catálogo, distribución)-> bool. Sobre las distribuciones que evalúa True, descarga el recurso en el downloadURL y lo sube al portal de destino. Por default no sube ninguna distribución.
- **generate_new_access_url** (*list*) – Se pasan los ids de las distribuciones cuyo accessURL se regenerar en el portal de destino. Para el resto, el portal debe mantiene el valor pasado en el DataJson.
- **origin_tz** (*str*) – Timezone de origen, un string (EJ: Africa/Bamako) el cual identifica el timezone del emisor del DataJson.

- **dst_tz** (*str*) – Timezone de destino, un string (EJ: Antarctica/Palmer) el cual identifica el timezone del receptor del DataJson, comunmente el timezone del servidor.

Devuelve El id del dataset restaurado.

Tipo del valor devuelto str

pydatajson.federation.**restore_organization_to_ckan**(*catalog*, *owner_org*, *portal_url*, *apikey*, *dataset_list=None*, *download_strategy=None*, *generate_new_access_url=None*, *origin_tz='America/Buenos_Aires'*, *dst_tz='America/Buenos_Aires'*)

Restaura los datasets de la organización de un catálogo al portal pasado por parámetro. Si hay temas presentes en el DataJson que no están en el portal de CKAN, los genera.

Args: catalog (DataJson): El catálogo de origen que se restaura. portal_url (str): La URL del portal CKAN de destino. apikey (str): La apikey de un usuario con los permisos que le

permitan crear o actualizar el dataset.

dataset_list(list(str)): Los ids de los datasets a restaurar. Si no se pasa una lista, todos los datasests se restauran.

owner_org (str): La organización a la cual pertencen los datasets. download_strategy(callable): Una función (catálogo, distribución)->

bool. Sobre las distribuciones que evalúa True, descarga el recurso en el downloadURL y lo sube al portal de destino. Por default no sube ninguna distribución.

generate_new_access_url(list): Se pasan los ids de las distribuciones cuyo accessURL se regenerar en el portal de destino. Para el resto, el portal debe mantiene el valor pasado en el DataJson.

origin_tz(str): Timezone de origen, un string (EJ: Africa/Bamako) el cual identifica el timezone del emisor del DataJson.

dst_tz(str): Timezone de destino, un string (EJ: Antarctica/Palmer) el cual identifica el timezone del receptor del DataJson, comunmente el timezone del servidor.

Returns: list(str): La lista de ids de datasets subidos.

pydatajson.federation.**restore_organizations_to_ckan**(*catalog*, *organizations*, *portal_url*, *apikey*, *download_strategy=None*, *generate_new_access_url=None*, *origin_tz='America/Buenos_Aires'*, *dst_tz='America/Buenos_Aires'*)

Restaura los datasets indicados para c/organización de un catálogo al portal pasado. Si hay temas presentes en el DataJson que no están en el portal de CKAN, los genera. Las organizaciones ya deben estar creadas.

Parámetros

- **catalog** (DataJson) – El catálogo de origen que se restaura.
- **organizations** (*dict*) – Datasets a restaurar por c/organización donde {«organizacion_id»: [dataset_id1, dataset_id2, . . .]}
- **portal_url** (*str*) – La URL del portal CKAN de destino.

- **apikey** (*str*) – La apikey de un usuario con los permisos que le permitan crear o actualizar el dataset.
- **download_strategy** (*callable*) – Una función (catálogo, distribución)-> bool. Sobre las distribuciones que evalúa True, descarga el recurso en el downloadURL y lo sube al portal de destino. Por default no sube ninguna distribución.
- **generate_new_access_url** (*list*) – Se pasan los ids de las distribuciones cuyo accessURL se regenerar en el portal de destino. Para el resto, el portal debe mantiene el valor pasado en el DataJson.
- **origin_tz** (*str*) – Timezone de origen, un string (EJ: Africa/Bamako) el cual identifica el timezone del emisor del DataJson.
- **dst_tz** (*str*) – Timezone de destino, un string (EJ: Antarctica/Palmer) el cual identifica el timezone del receptor del DataJson, comunmente el timezone del servidor.

Devuelve La lista de ids de datasets subidos.

Tipo del valor devuelto list(str)

2.1.8 Backup

Módulo con funciones auxiliares para hacer backups de catálogos.

pydatajson.backup.**get_catalog_path**(*catalog_id*, *catalogs_dir=u''*, *fmt=u'json'*)
Genera el path estándar de un catálogo en un filesystem.

pydatajson.backup.**get_distribution_dir**(*catalog_id*, *dataset_id*, *distribution_id*, *catalogs_dir=u''*, *use_short_path=False*)
Genera el path estándar de un catálogo en un filesystem.

pydatajson.backup.**get_distribution_path**(*catalog_id*, *dataset_id*, *distribution_id*, *distribution_file_name*, *catalogs_dir=u''*, *use_short_path=False*)
Genera el path estándar de un catálogo en un filesystem.

pydatajson.backup.**main**(*catalogs*, *include_data=True*, *use_short_path=True*)
Permite hacer backups de uno o más catálogos por línea de comandos.

Parámetros **catalogs** (*str*) – Lista de catálogos separados por coma (URLs o paths locales) para hacer backups.

pydatajson.backup.**make_catalog_backup**(*catalog*, *catalog_id=None*, *local_catalogs_dir=u''*, *include_metadata=True*, *include_data=True*, *include_datasets=None*, *include_distribution_formats=[u'CSV', u'XLS']*, *include_metadata_xlsx=True*, *use_short_path=False*)
Realiza una copia local de los datos y metadatos de un catálogo.

Parámetros

- **catalog** (*dict or str*) – Representación externa/interna de un catálogo. Una representación _externa_ es un path local o una URL remota a un archivo con la metadata de un catálogo, en formato JSON o XLSX. La representación _interna_ de un catálogo es un diccionario.
- **catalog_id** (*str*) – Si se especifica, se usa este identificador para el backup. Si no se especifica, se usa catalog[«identifier»].
- **local_catalogs_dir** (*str*) – Directorio local en el cual se va a crear la carpeta «catalog/...» con todos los catálogos.

- **include_metadata** (*bool*) – Si es verdadero, se generan los archivos data.json y catalog.xlsx.
- **include_data** (*bool*) – Si es verdadero, se descargan todas las distribuciones de todos los catálogos.
- **include_datasets** (*list*) – Si se especifica, se descargan únicamente los datasets indicados. Si no, se descargan todos.
- **include_distribution_formats** (*list*) – Si se especifica, se descargan únicamente las distribuciones de los formatos indicados. Si no, se descargan todas.
- **use_short_path** (*bool*) – No implementado. Si es verdadero, se utiliza una jerarquía de directorios simplificada. Caso contrario, se replica la existente en infra.

Devuelve None

pydatajson.backup.**make_catalogs_backup**(*catalogs*, *local_catalogs_dir=u”*, *include_metadata=True*, *include_data=True*, *include_metadata_xlsx=False*, *use_short_path=False*)

Realiza una copia local de los datos y metadatos de un catálogo.

Parámetros

- **catalogs** (*list or dict*) – Lista de catálogos (elementos que pueden ser interpretados por DataJson como catálogos) o diccionario donde las keys se interpretan como los catalog_identifier:

  { «modernizacion»: «http://infra.datos.gob.ar/catalog/modernizacion/data.json» }

  Cuando es una lista, los ids se toman de catalog_identifer, y se ignoran los catálogos que no tengan catalog_identifier. Cuando se pasa un diccionario, los keys reemplazan a los catalog_identifier (estos no se leeen).
- **catalog_id** (*str*) – Si se especifica, se usa este identificador para el backup. Si no se espedifica, se usa catalog[«identifier»].
- **local_catalogs_dir** (*str*) – Directorio local en el cual se va a crear la carpeta «catalog/...» con todos los catálogos.
- **include_metadata** (*bool*) – Si es verdadero, se generan los archivos data.json y catalog.xlsx.
- **include_data** (*bool*) – Si es verdadero, se descargan todas las distribuciones de todos los catálogos.

Devuelve None

- modindex
- genindex

CAPÍTULO 3

Versiones

3.1 Versiones

3.1.1 0.4.47 (2019-09-17)

- Implementación de timezones para los nodos de origen y destino en una federación

3.1.2 0.4.46 (2019-08-16)

- Fix a lecturas de catálogos XLSX en contextos concurrentes

3.1.3 0.4.45 (2019-08-01)

- Subo versión de python-dateutil

3.1.4 0.4.44 (2019-07-30)

- Bugfix en validación de campos de mail
- subo versión de openpyxl

3.1.5 0.4.43 (2019-06-26)

- Validación para ids numéricos en la resturación de catálogos.

3.1.6 0.4.42 (2019-06-04)

- Agrega parámetros `verify_ssl` y `requests_timeout` a los métodos de federación.

3.1.7 0.4.41 (2019-05-28)

- Agrega parámetros `verify_ssl` y `requests_timeout` a `DataJson` que controla el comportamiento de descarga de catálogos remotos.

3.1.8 0.4.40 (2019-05-28)

- Refactor de validaciones. Ahora `DataJson` acepta un parámetro `validator_class` que corre validaciones sobre el catálogo.

3.1.9 0.4.39 (2019-05-07)

- Cambia el nivel del logging para los warnings que se logueaban bajo error

3.1.10 0.4.38 (2019-04-15)

- Bugfix en la escritura de catálogos sin themes
- Actualiza planilla xlsx de catálogo
- La validación de datasets devuelve una lista en vez de un generador

3.1.11 0.4.37 (2019-04-03)

- Permite el cálculo de indicadores de federación usando ids

3.1.12 0.4.36 (2019-03-01)

- Cambia el cálculo de indicadores porcentuales para que calculen de 0 a 1
- Bugfix para ciertos catálogos con sufijos no reconocidos

3.1.13 0.4.35 (2019-02-19)

- Actualiza la validación para aceptar el string vacío como valor válido
- Marca los identificadores de distribuciones y datasets como campos requeridos

3.1.14 0.4.34 (2019-02-01)

- Implementa método para tomar frecuencia de una serie de tiempo

3.1.15 0.4.33 (2019-01-10)

- Cambia el kwarg dj_format por catalog_format
- Pequeño fix para los catalogos remotos json incorrectamente leídos como xlsx

3.1.16 0.4.32 (2019-01-08)

- Fix al dj_format para las lecturas

3.1.17 0.4.31 (2019-01-08)

- Se aceptan catálogos sin formato para la lectura del DataJson
- Nuevo parámetro para forzar la lectura de un catálogo en cierto formato
- Actualización de pyyaml

3.1.18 0.4.30 (2018-12-28)

- No se validan URLs repetidas para datasets, hay casos válidos donde ocurren

3.1.19 0.4.29 (2018-12-21)

- Método `remove_organizations_from_ckan()`.
- Cambia la estrategia de lectura para json sin extensión.

3.1.20 0.4.28 (2018-12-11)

- Parametro opcional a `push_dataset_to_ckan()` para regenerar `accessURL` de recursos.
- Permite el cálculo de indicadores con catálogo central opcional.

3.1.21 0.4.27 (2018-11-23)

- Las funcionalidades que estaban en `restore_catalog_to_ckan()` pasan a ser de `restore_organization`. `restore_catalog` se compone de varias llamadas a `restore_organization`.
- Documentación de `restore_catalog_to_ckan`.

3.1.22 0.4.26 (2018-11-05)

- Agrega métodos de manejo de organizaciones para bajar la información o subir a un portal CKAN.
- Fix en indicador "datasets_con_datos_pct" al calcular el porcentaje.
- Cambio en los tests para que usen archivos temporales en lugar de crearlos en la carpeta results.

3.1.23 0.4.25 (2018-10-22)

- Agrega indicador "datasets_con_datos_cant" para identificar la cantidad de datasets que tienen alguna distribución potencialmente con datos y los que no.
- Expande la función `backup.make_catalogs_backup()` con argumentos opcionales para facilitar la generación de backups descargando las distribuciones.

3.1.24 0.4.24 (2018-10-16)

- Cambia el valor default en el indicador `datasets_frecuencias_cant`.

3.1.25 0.4.23 (2018-10-2)

- Se agregan HTML, PHP y RAR como formatos de datos posibles.
- Bugfix relacionado a los valores default en el cálculo de indicadores.

3.1.26 0.4.22 (2018-09-05)

- Agrega espacios a los caracteres permitidos en keyword.

3.1.27 0.4.21 (2018-08-21)

- Tests y pequeños bugfixes a ckan_reader.
- Adecua el codigo a pycodestyle.
- Fija piso de 80 % de coverage para CI.

3.1.28 0.4.20 (2018-08-09)

- Agrega tildes y ñ a los caracteres permitidos en keyword.
- Cuenta los campos faltantes como `None` en los indicadores.

3.1.29 0.4.19 (2018-08-07)

- Validación de caracteres permitidos en los keywords.
- Bugfix a la lectura de listas en xlsx con comas extras.
- Bugfix en el cual se repetían los errores de validación si se pedía formato lista.

3.1.30 0.4.18 (2018-07-30)

- Agrega interfaz por línea de comandos para validar rápidamente un catálogo: `pydatajson validation http://datos.gob.ar/data.json`.
- Validación de keywords, themes, y lenguajes vacíos.
- Bugfix en `distribution_has_time_index` para capturar excepciones en field inválidos.

3.1.31 0.4.17 (2018-07-10)

- Agregados 3 indicadores `distribuciones_federadas`, `datasets_licencias_cant` y `distribuciones_tipos_cant`.
- `harvest_catalog_to_ckan` devuelve el mensaje en lugar de las representaciones de las excepciones.

3.1.32 0.4.16 (2018-06-19)

- Bugfix en la escritura y lectura de catálogos xlsx.
- Federar campo `type` en distribuciones.
- Refactor del logging del módulo. Todos los eventos se escriben en el logger `pydatajson`.
- Reestructuración de la respuesta de `harvest_catalog_to_ckan()`, devuelve adicionalmente los datasets con errores de federación.

3.1.33 0.4.15 (2018-05-15)

- Cambios en los requerimientos y `setup.py` para definir los environment markers de manera que soporte setuptools.

3.1.34 0.4.14 (2018-05-11)

- `harvest_catalog_to_ckan()` atrapa todas las excepciones de un dataset y no detiene la ejecución.

3.1.35 0.4.13 (2018-05-06)

- Agrega una primer interfaz sencilla por línea de comandos. Cualquier módulo puede ser usado como `pydatajson module_name arg1 arg2 arg3` siempre que defina un método `main()` a nivel del módulo que procese los parámetros.

3.1.36 0.4.12 (2018-05-04)

- Agrega función `get_distribution_time_index()` que devuelve el `title` del `field` marcado como time_index en una distribución de series de tiempo, si este lo tiene.

3.1.37 0.4.11 (2018-04-25)

- Corrige bug de `harvest_catalog_ot_ckan` para manejar excepciones de validación de los datasets

3.1.38 0.4.10 (2018-04-24)

- Mejora manejo de errores de las funciones para federar catálogos completos.

3.1.39 0.4.9 (2018-04-24)

- Agrego función para generar ids de distribuciones en catálogos que nos los tienen (compatibilidad con perfil 1.0)
- Agrega función para eliminar todos los datastets federados de un catálogo que se encuentren en un CKAN
- Implemento fallback que busca un theme por identifier primero o por label después (si falla)
- Agrego excepciones a los chequeos de formato vs. extensión
- Agrego paramétros a la función title_to_name() para establecer una longitud máxima del resultado de la transformación en caracteres

3.1.40 0.4.8 (2018-04-18)

- Mejoro manejo de errores de los métodos optimizados de búsqueda

3.1.41 0.4.7 (2018-04-17)

- Flexibiliza métodos de búsqueda optimizados para aceptar data.json's versión 1.0
- Mejora la performance de los métodos de búsqueda optimizados

3.1.42 0.4.6 (2018-04-17)

- Re-estructura el archivo de configuración para federación (nueva versión simplificada)
- Agrega módulo para hacer backups de datos y metadatos de un catálogo
- Mejora la performance de guardar catálogos en Excel

3.1.43 0.4.4 (2018-04-09)

- Agrega wrappers para `push_dataset_to_ckan()`

3.1.44 0.4.3 (2018-03-20)

- Mejora el manejo de themes para recrear un catálogo

3.1.45 0.4.2 (2018-03-13)

- Agrega funciones auxiliares para la administración de un CKAN vía API para facilitar la administración de la federación de datasets
  - `remove_dataset_to_ckan()`
- Incorpora nuevas validaciones (formatos y fileNames)
- Agrega flags opcionales para que `push_dataset_to_ckan()` sea un método que transforma opcionalmente la metadata de un dataset

3.1.46 0.4.1 (2018-02-16)

- `datasets_equal()` permite especificar los campos a tener en cuenta para la comparación, como un parámetro.

3.1.47 0.4.0 (2018-02-08)

- Incorpora métodos para federar un dataset de un catálogo a un CKAN o un Andino: `push_dataset_to_ckan()`.
- Actualiza validaciones y esquema de metadatos al Perfil Nacional de Metadatos versión 1.1.

3.1.48 0.3.21 (2017-12-22)

- Agrega soporte para Python 3.6

3.1.49 0.3.20 (2017-11-16)

- Agrego método `get_theme()` para devolver un tema de la taxonomía específica del catálogo según su `id` o `label`.

3.1.50 0.3.19 (2017-10-31)

- Agrego métodos de búsqueda de series de tiempo en un catálogo (`get_time_series()`) y un parámetro `only_time_series=True or False` para filtrar datasets y distribuciones en sus métodos de búsqueda (`get_datasets(only_time_series=True)` devuelve sólo aquellos datasets que tengan alguna serie de tiempo).

3.1.51 0.3.18 (2017-10-19)

- Agrego posibilidad de pasar un logger desde afuera a la función de lectura de catálogos en Excel.

3.1.52 0.3.15 (2017-10-09)

- Agrega filtro por series de tiempo en `get_datasets()` y `get_distributions()`. Tienen un parámetro `only_time_series` que devuelve sólo aquellos que tengan o sean distribuciones con series de tiempo.

3.1.53 0.3.12 (2017-09-21)

- Agrega función para escribir un catálogo en Excel.
- Agrega funciones para remover datasets o distribuciones de un catálogo.

3.1.54 0.3.11 (2017-09-13)

- Incorpora parámetro para excluir campos de metadatos en la devolución de la búsqueda de datasets y distribuciones.

3.1.55 0.3.10 (2017-09-11)

- Agregar referencia interna a los ids de las entidades padre de otras (distribuciones y fields.)

3.1.56 0.3.9 (2017-09-05)

- Flexibiliza lectura de extras en ckan to datajson.
- Flexibiliza longitud mínima de campos para recomendar su federación o no.
- Agrega método para devolver los metadatos a nivel de catálogo.
- Resuelve la escritura de objetos python como texto en excel.

3.1.57 0.3.8 (2017-08-25)

- Agrega stop words a `helpers.title_to_name()`

3.1.58 0.3.4 (2017-08-21)

- Agrega método para buscar la localización de un `field` en un catálogo.

3.1.59 0.3.3 (2017-08-20)

- Agrega método para convertir el título de un dataset o distribución en un nombre normalizado para la creación de URLs.

3.1.60 0.3.2 (2017-08-16)

- Amplía reporte de federación en markdown.

3.1.61 0.3.0 (2017-08-14)

- Agrega métodos para navegar un catálogo desde el objeto DataJson.

3.1.62 0.2.27 (2017-08-11)

- Agrega validacion de que el campo `superTheme` sólo contenga ids en mayúsculas o minúsculas de alguno de los 13 temas de la taxonomía temática de datos.gob.ar.
- Agrega validación limitando a 60 caracteres los nombres de los campos `field_title`.
- Mejoras al reporte de asistencia a la federación.

3.1.63 0.2.26 (2017-08-04)

- Agrega validación de que no haya ids repetidos en la lista de temas de `themeTaxonomy`.
- Agrega traducción de ckan del campo extra `Cobertura temporal` a `temporal`.

3.1.64 0.2.24 (2017-08-03)

- Mejoras en los reportes de errores y análisis de datasets para federación
- Métodos `DataJson.validate_catalog()` y `DataJson.generate_datasets_report()` tienen nuevas opciones para mejorar los reportes, especialmente en excel.

3.1.65 0.2.23 (2017-08-02)

- Bug fixes

3.1.66 0.2.22 (2017-08-02)

- Agrega estilo y formato al reporte de datasets
- Agrega nuevos campos al reporte de datasets
- Agrega un campo identificador del catálogo en el archivo de configuración de federación

3.1.67 0.2.21 (2017-08-02)

- Tolera el caso de intentar escribir un reporte de datasets sobre un catálogo que no tiene datasets. Loggea un warning en lugar de levantar una excepción.

3.1.68 0.2.20 (2017-08-01)

- Elimina la verificación de SSL en las requests de ckan_reader.

3.1.69 0.2.19 (2017-08-01)

- Elimina la verificación de SSL en las requests.

3.1.70 0.2.18 (2017-07-25)

- Mejora la validación del campo `temporal`
- Agrega formas de reporte de errores para el método `DataJson.validate_catalog()`:
  - Devuelve sólo errores con `only_errors=True`
  - Devuelve una lista de errores lista para ser convertida en tabla con `fmt="list"`

3.1.71 0.2.17 (2017-07-18)

- Agrega un método para convertir un intervalo repetido (Ej.: R/P1Y) en su representación en prosa («Anualmente»).
- Agrego método que estima los datasets federados que fueron borrados de un catálogo específico. Se consideran datasets federados y borrados de un catálogo específico aquellos cuyo publisher.name existe dentro de algún otro dataset todavía presente en el catálogo específico.

3.1.72 0.2.16 (2017-07-13)

- Bug fix: convierte a unicode antes de escribir un objeto a JSON.

3.1.73 0.2.15 (2017-07-11)

- Modifica la definición de dataset actualizado usando el campo «modified» del perfil de metadatos. Si este campo no está presente en la metadata de un dataset, se lo considera desactualizado.

3.1.74 0.2.14 (2017-07-10)

- Modifica la definición de dataset usada para comparar limitándola a la comparación por «title» y «publisher_name».

3.1.75 0.2.13 (2017-06-22)

- Agrega método para verificar si un dataset individual está actualizado

3.1.76 0.2.12 (2017-06-22)

- Se modifica el template de CATALOG README
- Se agrega el indicador «datasets_no_federados» a generate_catalogs_indicators

3.1.77 0.2.11 (2017-05-23)

- Se agrega en `core` el método `DataJson.generate_catalogs_indicators`, que genera indicadores de monitoreo de catálogos, recopilando información sobre, entre otras cosas, su validez, actualidad y formato de sus contenidos.

3.1.78 0.2.10 (2017-05-11)

- Correción ortográfica del listado de frecuencias de actualización admisibles (`pydatajson/schemas/accrualPeriodicity.json`).

3.1.79 0.2.9 (2017-05-04)

- Hotfixes para que `pydatajson` sea deployable en nuevos entornos donde el `setup.py` estaba fallando.

3.1.80 0.2.5 (2017-02-16)

- Se agrega una nueva función a `readers`, `read_ckan_catalog`, que traduce los metadatos que disponibiliza la Action API v3 de CKAN al estándar `data.json`. Esta función *no* está integrada a `read_catalog`.
- Se modifican todos los esquemas de validación, de modo que los campos opcionales de cualquier tipo y nivel acepten strings vacías.

3.1.81 0.2.0 (2017-01-31)

- Se reestructura la librería en 4 módulos: `core`, `readers`, `writers` y `helpers`. Toda la funcionalidad se mantiene intacta, pero algunas funciones muy utilizadas cambian de módulo. En particular, `pydatajson.pydatajson.read_catalog` es ahora `pydatajson.readers.read_catalog`, y `pydatajson.xlsx_to_json.write_json_catalog` es ahora `pydatajson.writers.write_json_catalog` (o `pydatajson.writers.write_json`).
- Se agrega el parámetro `frequency` a `pydatajson.DataJson.generate_harvester_config`, que controla la frecuencia de cosecha que se pretende de los datasets a incluir en el archivo de configuración. Por omisión, se usa `'R/P1D'` (diariamente) para todos los datasets.

- Se agrega la carpeta `samples/`, con dos rutinas de transformación y reporte sobre catálogos de metadatos en formato XLSX.

3.1.82 0.1.7 (2017-01-10)

- Se agrega el módulo `xlsx_to_json`, con dos métodos para lectura de archivos locales o remotos, sean JSON genéricos (`xlsx_to_json.read_json()`) o metadatos de catálogos en formato XLSX (`read_local_xlsx_catalog()`).
- Se agrega el método `pydatajson.read_catalog()` que interpreta todos las representaciones externas o internas de catálogos conocidas, y devuelve un diccionario con sus metadatos.

3.1.83 0.1.6 (2017-01-04)

- Se incorpora el método `DataJson.generate_harvestable_catalogs()`, que filtra los datasets no deseados de un conjunto de catálogos.
- Se agrega el parámetro `harvest` a los métodos `DataJson.generate_harvestable_catalogs()`, `DataJson.generate_datasets_report()` y `DataJson.generate_harvester_config()`, para controlar el criterio de elección de los datasets a cosechar.
- Se agrega el parámetro `export_path` a los métodos `DataJson.generate_harvestable_catalogs()`, `DataJson.generate_datasets_report()` y `DataJson.generate_harvester_config()`, para controlar la exportación de sus resultados.

3.1.84 0.1.4 (2016-12-23)

- Se incorpora el método `DataJson.generate_datasets_report()`, que reporta sobre los datasets y la calidad de calidad de metadatos de un conjunto de catálogos.
- Se incorpora el método `DataJson.generate_harvester_config()`, que crea archivos de configuración para el Harvester a partir de los reportes de `generate_datasets_report()`.

3.1.85 0.1.3 (2016-12-19)

- Al resultado de `DataJson.validate_catalog()` se le incorpora una lista (`"errors"`) con información de los errores encontrados durante la validación en cada nivel de jerarquía («catalog» y cada elemento de «dataset»)

3.1.86 0.1.2 (2016-12-14)

- Se incorpora validación de tipo y formato de campo
- Los métodos `DataJson.is_valid_catalog()` y `DataJson.validate_catalog()` ahora aceptan un `dict` además de un `path/to/data.json` o una url a un data.json.

3.1.87 0.1.0 (2016-12-01)

Primera versión para uso productivo del paquete.

- La instalación via `pip install` debería reconocer correctamente la ubicación de los validadores por default.

- El manejo de data.json's ubicados remotamente se hace en función del resultado de `urlparse.urlparse`
- El formato de respuesta de `validate_catalog` se adecúa a la última especificación (ver `samples/validate_catalog_returns.json`.

3.1.88 0.0.13 (2016-11-25)

- Intentar que la instalación del paquete sepa donde están instalados los schemas por default

3.1.89 0.0.12 (2016-11-25)

- Primera versión propuesta para v0.1.0

Índice de Módulos Python

p

Índice

Symbols

C

D

F

G

H

I

M

P

R

T

V

W