Índice de contenidos

1. Objetivos.

- Descarga de datos
- Explorar repositorios Open Data
- Lectura y escritura de JSON/XML
- Inserción y consulta en MongoDB

2. Actividades a desarrollar

2.1. Búsqueda de datos

- Hacer una descarga de un conjunto de datos del Open Data de: Valencia, Madrid o Barcelona. Hablar sobre la fuente de los datos y la calidad de los datos. (Mínimo del conjunto de datos 100 registros o filas).

2.2. Fichero e inserción

- Generar un JSON de los datos en un fichero en local.
- Insertar en MongoDB todos los datos sin utilizar un bucle. Definir colección y código.

2.3. Consultas

- Mediante una consulta a Mongo obtener unos campos concretos de una consulta buscando en un atributo un valor concreto. El valor tiene que ser un valor numérico.
- Hacer una consulta de Mongo de forma que nos devuelva, en función de un atributo, el top 3 ordenado. ¿Existe el atributo “coordinates” en el conjunto de datos? ¿Cómo se consulta?
- Recorrer todos los registros de la colección y realizar una media en un atributo.

3. Desarrollo de las actividades

3.1. Definición del conjunto de datos y repositorio

El *dataset* se obtiene del repositorio abierto del Ayuntamiento de Valencia. En concreto se usará en esta actividad un *dataset* en formato JSON que contiene 113 recursos sociales para personas con discapacidad cuya URL es http://gobiernoabierto.valencia.es/es/resource/?ds=recursos-sociales-personas-discapacidad&id=dadecca3-85de-4867-92d6-99a335689528

Cada uno de los 113 documentos de la colección sigue la estructura siguiente:


```
_id: ObjectId("5e64bfd6f71df81be49fa164")
type: "Feature"
properties: Object
    descripcion: "CENTRO OCUPACIONAL MUNICIPAL JUAN DE GARAY"
    titularidad: "Ayuntamiento de Valencia. Concejalía de Servicios Sociales"
    idnotes: "F970178132AAB7C5C12572D6002D2F23"
geometry: Object
    type: "Point"
    coordinates: Array
        0: 724111.861
        1: 4370509.307
numerico_inventado: 34
```


A destacar el último atributo llamado *numerico_inventado*, pues se trata de un campo que se añade con posterioridad a la creación del JSON basado en el *dataset* originario, ya que hará falta un campo numérico para las consultas (*queries*) requeridas en esta actividad.

3.2. Estrategia de programación

Librerías usadas en el script

`urllib.request` para acceder a la URL origen de los datos.

`json` para crear el fichero JSON y cargar los datos en él.

`pprint` para mostrar de forma elegante los documentos de la colección.

`pymongo` para conectar e interactuar con MongoDB.

`random` para dar valores aleatorios al nuevo campo numérico añadido.

Código importante

Se abre un archivo y se vuelca en el mismo los datos del diccionario:

```
def create_json(out_file, data):
    file_data = open(out_file, 'w')
    with open(out_file, 'w') as file:
        json.dump(data, file, indent=4)
    file_data.close()
```

Se exportan los datos a un objeto diccionario:

```
def download_data(url_data):
    with urllib.request.urlopen(url_data) as url:
```

```
    data = json.loads(url.read().decode())
  return data
```

Conexión a MongoDB:

```
def mongo_connect():
  print("Connecting to MongoDB...")
  uri_mongo =
    "mongodb+srv://adria:adria@cluster0-1yjkn.mongodb.net/test?retryWrites=tru
    e&w=majority"
  client = pymongo.MongoClient(uri_mongo)
  client.test
  return client
```

Lee los datos desde el fichero JSON:

```
def read_json(json_path):
  with open(json_path) as f:
    file_data = json.load(f)
  return file_data
```

Inserta los documentos en la colección e informa cuántos se han insertado:

```
def insert_mongo_list(list_data, mongo_client, database, collection):
  # first define the collection to insert
  db = mongo_client[database]
  col = db[collection]
```

```
    # insert array of Features into collection
    col.insert_many(list_data["features"])
    pprint.pprint(list_data["features"])
    longitud_array = len(list_data["features"])
    print('\n' + repr(longitud_array) + ' documentos insertados en la Colección ' +
        collection)
```

Modifica todos los documentos para añadir un nuevo campo numérico. En este caso inserta un número aleatorio entre 1 y 100 pero será el mismo para los 113 documentos.

```
def add_random_numeric(your_collection):
    print("Updating collection in order to add a random numeric...")
    your_collection.update_many(
        {},
        [
            {"$set": {"numerico_inventado": random.randint(1, 100)}}
        ]
    )
```

Consulta 1. Devuelve la *Descripción* del Recurso Social cuya *coordenada Y* sea 4371138.878:

```
def get_query1(collection):
```

```
    return
        collection.find_one({"geometry.coordinates.1":4371138.878},{"properties.desc
        ripcion":1})
```

Consulta 2. Devuelve los 3 primeros Recursos Sociales sin Titularidad, ordenados ascendentemente por Descripción:

```
def get_query2(collection):
    return collection.find({"properties.titularidad":
        ""}).sort("properties.descripcion",1).limit(3)
```

Consulta 2b. Devuelve el documento cuya coordenada X coincida con la dada. Muestra solo las Coordenadas:

```
def get_query2b(collection, coordenada):
    # devuelve el documento cuya coordenada x coincida con la dada. Muestra
        solo las Coordenadas.
    return collection.find_one({"geometry.coordinates.0":coordenada},
        {"_id":0,"geometry.coordinates":1})
```

Consulta 3. Devuelve la media aritmética del atributo “numerico_inventado” de todos los documentos de la colección:

```
def get_query3(collection):
    return collection.aggregate(
        [
```

```
        {"$group": {"_id": {}, "Promedio numerico_inventado": {"$avg":
    "$numerico_inventado"}}}
    ])
```

Referencias usadas

1. https://www.analyticslane.com/2018/07/16/archivos-json-con-python/ [consulta: 07/03/2020]
2. https://stackoverflow.com/questions/49510049/how-to-import-json-file-to-mongodb-using-python/49510257 [consulta: 07/03/2020]
3. https://www.journaldev.com/23642/python-concatenate-string-and-int [consulta: 07/03/2020]
4. https://docs.mongodb.com/manual/reference/ [consulta: 08/03/2020]
5. https://charlascylon.com/2013-07-10-tutorial-mongodb-operaciones-de-consulta-avanzadas [consulta: 08/03/2020]
6. https://stackoverflow.com/questions/10242149/using-sort-with-pymongo [consulta: 08/03/2020]
7. https://docs.mongodb.com/manual/aggregation/ [consulta: 08/03/2020]

3.3. Resultados de las actividades

```
*********Fichero JSON creado
Connecting to MongoDB…
113 documentos insertados en la Colección rrss_discapacidad
Updating collection...


QUERY 1:
```

{'_id': ObjectId('5e64bfd6f71df81be49fa186'),
 'properties': {'descripcion': 'ASOCIACIÓN SORDOS 2000 VALENCIA'}}

QUERY 2:

{'_id': ObjectId('5e64bfd6f71df81be49fa16b'),
 'geometry': {'coordinates': [729220.581, 4371944.379], 'type': 'Point'},
 'numerico_inventado': 34,
 'properties': {'descripcion': 'ACVEM. ASOCIACIÓN DE LA COMUNIDAD
VALENCIANA '
                               'DE ESCLEROSIS MÚLTIPLE',
            'idnotes': 'E464B84E489A539CC125746A0034C101',
            'titularidad': ''},
 'type': 'Feature'}
{'_id': ObjectId('5e6bd450b5e254c449e7a6c3'),
 'geometry': {'coordinates': [729220.581, 4371944.379], 'type': 'Point'},
 'numerico_inventado': 34,
 'properties': {'descripcion': 'ACVEM. ASOCIACIÓN DE LA COMUNIDAD
VALENCIANA '
                               'DE ESCLEROSIS MÚLTIPLE',
            'idnotes': 'E464B84E489A539CC125746A0034C101',
            'titularidad': ''},
 'type': 'Feature'}
{'_id': ObjectId('5e64bfd6f71df81be49fa16a'),
 'geometry': {'coordinates': [723640.434, 4371985.158], 'type': 'Point'},
 'numerico_inventado': 34,

```
 'properties': {'descripcion': 'ADELA. ASOCIACIÓN VALENCIANA DE ESCLEROSIS
'
                               'LATERAL AMIOTRÓFICA',
            'idnotes': '102EA7D9A552475BC125746A0035675E',
            'titularidad': ''},
 'type': 'Feature'}
```


QUERY 2b:


```
{'geometry': {'coordinates': [724111.861, 4370509.307]}}
```


QUERY 3:


```
{'Promedio numerico_inventado': 34.0, '_id': {}}
```


```
Finishing script...

Process finished with exit code 0
```

8. Conclusiones

En esta actividad hemos visto cómo beber de una fuente abierta para crear un fichero JSON e insertar los datos en **MongoDB** para luego poder consultarlos, todo ello mediante el uso del lenguaje **Python**. El script .py unifica todo el código para hacer posible lo anterior.

La naturalidad del lenguaje Python ha sido de agradecer, no tanto la sintaxis tan específica que se debe usar con MongoDB para insertar, actualizar y consultar datos; en este punto, bajo mi punto de vista, es menos flexible, entendible y amigable la sintaxis de MongoDB. Como neófito en este último tipo de bases de datos, siento que aún le falta para llegar al nivel de estandarización del modelo de bases de datos relacional.

El script Python y el presente documento se encuentran publicados en GitHub:

https://github.com/adrimarmol/06MIOT_adrimarmol.git