Sistemas de Gestión de Datos y de la Información
Máster en Ingeniería Informática, 2025–2026
Práctica 2

Fecha de entrega: domingo 2 de noviembre de 2025

Entrega de la práctica
La práctica se entregará en un único fichero **GrupoXX.zip** mediante el Campus Virtual de la asignatura. Este fichero **ZIP** contendrá tres ficheros, uno por cada apartado: `consultas.py`, `aggregation.py` y `geo.py`.

Lenguaje de programación
3.12 o superior

Calificación
Se valorará la corrección, la claridad y organización del código, además de la eficiencia de las consultas.

Declaración de autoría e integridad
Todos los ficheros entregados contendrán una cabecera en la que se indique la asignatura, la práctica y los autores. Esta cabecera también contendrá la siguiente declaración de integridad:

> *Declaramos que esta solución es fruto exclusivamente de nuestro trabajo personal. No hemos sido ayudados por ninguna otra persona o sistema automático ni hemos obtenido la solución de fuentes externas, y tampoco hemos compartido nuestra solución con otras personas de manera directa o indirecta. Declaramos además que no hemos realizado de manera deshonesta ninguna otra actividad que pueda mejorar nuestros resultados ni perjudicar los resultados de los demás.*

No se corregirá ningún fichero que no venga acompañado de dicha cabecera.

En los dos primeros apartados de estas práctica consideraremos una base de datos MongoDB sgdi_pr2 que almacena información sobre **usuarios** y **películas**. Para ello dispone de dos colecciones usuarios y peliculas con el siguiente aspecto (todos los documentos tienen el mismo número y tipo de campos):

```
COLECCION USUARIOS
{
  "_id" :"fernandonoguera",
  "nombre" :"Pedro",
  "apellido1" :"Cordero",
  "apellido2" :"Bustos",
  "sexo" :"M",
  "gustos" :["terror", "comedia", "tragedia" ],
  "email" :"hporcel@arregui-belmonte.com",
  "edad" :54,
  "password" :"c46526e4f34352111b9f98feaacf1338b59d8d15",
  "direccion" :{"cod_postal" :"73182",
                "numero" :90,
                "puerta" :"C",
                "pais" :"Alemania",
                "piso" :3,
                "nombre_via" :"Camino de Laura Rebollo",
                "tipo_via" :"Avenida" }
  "visualizaciones" :[
    { "_id" :ObjectId("583ef650323e9572e2813189"),
      "titulo" :"Ex delectus vel dicta delectus.",
      "fecha" :"1995-04-06" },
    { "_id" :ObjectId("583ef651323e9572e2813dd5"),
      "titulo" :"Ipsam repudiandae dolorem libero voluptatibus.",
      "fecha" :"1978-12-27" }
  ],
}
```

```
COLECCION PELICULAS
{
  "_id" :ObjectId("583ef650323e9572e2813189"),
  "director" :"Lourdes Sacristan Bernal",
  "titulo" :"Ex delectus vel dicta delectus.",
  "lanzamiento" :1967,
  "pais" :["Myanmar", "Azerbaiyan" ]
}
```

Podéis cargar datos de prueba para estas dos colecciones usando MongoDB Compass a partir de los ficheros usuarios.json y peliculas.json.

1. Consultas en MongoDB (3.5pt)

Completar el esqueleto consultas.py para realizar las operaciones de consulta que se presentan a continuación. Cada consulta está encapsulada en una función Python que acepta como primer parámetro mongoclient un objeto MongoClient de pymongo previamente conectado a la base de datos y

después recibe el resto de valores necesarios para realizar la consulta. El resultado de estas funciones debe ser directamente el **objeto pymongo.cursor.Cursor con los resultados de la consulta**.

Las consultas que se deben implementar son:

1. (0.5pt) Obtener el **email** y las `n` **primeras visualizaciones** de películas (según el orden en el que aparecen en el documento, no su fecha) visualizadas por el usuario `user_id`.
   Ej: `usuario_peliculas(mongoclient, 'fernandonoguera', 3)`


```
{
    "visualizaciones": [
      {
        "_id": ...,
        "titulo": "Ex delectus vel dicta delectus.",
        "fecha": "1995-04-06"
      },
      {
        "_id": ...,
        "titulo": "Ipsam repudiandae dolorem libero voluptatibus.",
        "fecha": "1978-12-27"
      },
      {
        "_id": ...,
        "titulo": "Facere impedit atque pariatur ratione occaecati.",
        "fecha": "2016-07-12"
      }
    ],
    "email": "hporcel@arregui-belmonte.com"
  }
```


2. (0.5pt) Devolver el **_id**, **nombre** y **apellidos** de los primeros `n` usuarios a los que les gusten una serie de tipos de película (todos a la vez).
   Ej: `usuarios_gustos(mongoclient, ['terror', 'comedia'], 5)`


```
{'_id': 'fernandonoguera',
 'apellido1': 'Cordero',
 'apellido2': 'Bustos',
 'nombre': 'Pedro'}
{'_id': 'wgutiérrez',
 'apellido1': 'Alberto',
 'apellido2': 'Aragón',
 'nombre': 'Alejandra'}
{'_id': 'tordóñez',
 'apellido1': 'Hernando',
 'apellido2': 'Hernando',
 'nombre': 'Jorge'}
```


3. (0.5pt) Obtener el **_id** de los usuarios de un determinado `sexo` y con una edad comprendida entre `edad_min` y `edad_max` (incluidas).
   Ej: `usuario_sexo_edad(mongoclient, 'M', 50, 80)`


```
{'_id': 'fernandonoguera'}
```

```
{'_id': 'wgutiérrez'}
{'_id': 'lidiaadán'}
...
```

4. (0.5pt) Devolver el **nombre** y **apellidos** de los usuarios cuyo primer y segundo apellido coinciden, ordenados por edad de manera ascendente.
   Ej: usuarios_apellidos (mongoclient)

```
{'apellido1': 'Parra', 'apellido2': 'Parra', 'nombre': 'Sara'}
{'apellido1': 'Valenzuela', 'apellido2': 'Valenzuela', 'nombre': 'Alberto'}
{'apellido1': 'Sobrino', 'apellido2': 'Sobrino', 'nombre': 'Santiago'}
 ...
```

5. (0.5pt) Recuperar el **título** y **director** de las películas cuyo director tenga un nombre que empiece por un determinado prefijo .
   Ej: pelicula_prefijo (mongoclient, 'Yol')

```
{'director': 'Yolanda Cámara Saldaña',
 'titulo': 'Ea deleniti nobis aliquid est rerum.'}
{'director': 'Yolanda Zabaleta Salvador',
 'titulo': 'Vero quidem eius adipisci aliquid sequi beatae.'}
{'director': 'Yolanda Valcárcel',
 'titulo': 'Minima odio error architecto fuga facilis vitae.'}
...
```

6. (0.5pt) Obtener el **_id**, **edad** y **gustos** de aquellos usuarios que tienen exactamente n gustos, ordenados por edad descendente.
   Ej: usuarios_gustos_numero(mongoclient, 6)

```
{'_id': 'lucíamurillo',
 'edad': 85,
 'gustos': ['terror', 'comedia', 'tragedia', ...]}
{'_id': 'hpujadas',
 'edad': 82,
 'gustos': ['terror', 'comedia', 'dibujos', ...]}
{'_id': 'solsonaaurora',
 'edad': 76,
 'gustos': ['terror', 'comedia', 'tragedia', ...]}
...
```

7. (0.5pt) Devolver el **_id** de los usuarios que vieron la película id_pelicula entre dos fechas inicio y fin .
   Ej: usuarios_vieron_pelicula (mongoclient, '583ef652323e9572e2814c48', '1999−01−01', '2002−12−31')

```
{'_id': 'eecheverría'}
{'_id': 'gabrielrecio'}
```

2. Tuberías de agregación (4pt)

En este apartado implementaremos consultas *avanzadas* utilizando tuberías de agregación. Todas las consultas se realizarán sobre las mismas colecciones del apartado anterior, dentro de la base de datos `sgdi_pr2`. Al igual que en el apartado anterior, se deberá implementar cada una de las consultas en una función Python que se conecte a MongoDB a través de la bibliteca `pymongo`, recibiendo un primer parámetro `mongoclient` con una conexión previamente abierta al servidor MongoDB. En este apartado se entregará un único fichero **`aggregation.py`** cuyo esqueleto se puede descargar del Campus Virtual. Cada consulta está encapsulada en una función (`agg1()`–`agg4()`) que **debe devolver los resultados de la consulta**, es decir, no debe almacenarlos en ninguna colección.

1. (1pt) Listado de *país-número de películas* rodadas en él, ordenado por número de películas descendente y en caso de empate por nombre país ascendente.
   Ejemplo de salida:

```
{'_id': 'Uruguay', 'num_peliculas': 136}
{'_id': 'Belice', 'num_peliculas': 130}
{'_id': 'India', 'num_peliculas': 129}
{'_id': 'República de Moldova', 'num_peliculas': 129}
...
```

2. (1pt) Listado de los 3 tipos de película más populares entre los usuarios de un determinado país, ordenados de mayor a menor número de usuarios que les gusta. En caso de empate a número de usuarios, se usa el tipo de película de manera ascendente.
   Ejemplo de `agg2(mongoclient, 'Emiratos Árabes Unidos')`:

```
{'total': 3, 'tipo': 'ciencia ficcion'}
{'total': 3, 'tipo': 'comedia'}
{'total': 3, 'tipo': 'musical'}
```

3. (1pt) Listado de *país-(edad mínima, edad-máxima, edad media)* teniendo en cuenta únicamente los usuarios mayores de edad, es decir, con más de 17 años. Los países con *menos de 3 usuarios mayores de edad* **no deben aparecer en el resultado**. Los resultados se deben **ordenar de manera ascendente por nombre de país**.
   Ejemplo de salida:

```
{'_id': 'Burkina Faso', 'min_edad': 22, 'max_edad': 76,
 'avg_edad': 41.666666666666664, 'num_usuarios': 3}
{'_id': 'Chad', 'min_edad': 20, 'max_edad': 90,
 'avg_edad': 55.333333333333336, 'num_usuarios': 3}
{'_id': 'Chile', 'min_edad': 47, 'max_edad': 96,
 'avg_edad': 74.66666666666667, 'num_usuarios': 3}
...
```

4. (1pt) Listado de *título película-número de visualizaciones* de las 10 películas más vistas, ordenado por número descencente de visualizaciones. En caso de empate, romper por título de película ascendente.
   Ejemplo de salida:

```
{'num_visualizaciones': 4,
 'titulo': 'Sunt sint sint sit in.'}
```

```
{'num_visualizaciones': 3,
 'titulo': 'Eveniet repellat at perferendis iure sunt.'}
{'num_visualizaciones': 3,
 'titulo': 'Nostrum quos explicabo quia earum minima.'}
...
```

3. Consultas geoespaciales en MongoDB (2.5pt)

En este apartado vamos a trabajar con el fichero 300356−0−monumentos−ciudad−madrid.json de monumentos de Madrid en formato JSON que se puede descargar de

`https://datos.gob.es/es/catalogo/l01280796-monumentos-de-la-ciudad-de-madrid1`

Este documento JSON contiene dos entradas: una entrada `@context` con metainformación y otra entrada `@graph` que contiene una lista de documentos con la información de los monumentos. Para realizar este apartado el primer paso será crear una colección monumentos dentro de la base de datos sgdi_pr2 y cargar en ella el contenido del fichero JSON. En este apartado se entregará un único fichero geo.py cuyo esqueleto se puede descargar del Campus Virtual, donde cada apartado se resuelve en una función Python.

1. (1pt) Diseña una tubería de agregación para limpieza que extraiga los documentos individuales de los monumentos de la entrada `@graph` y reemplace la entrada location de cada monumento por una entrada siguiendo el formato GeoJSON Point (**muy importante: primero la longitud y luego la latitud**):

   ```
   location: {
        type: "Point",
        coordinates: [<longitud>, <latitud>]
   }
   ```

   La última etapa de esta tubería de agregación debe almacenar los documentos de los monumentos en la nueva colección monumentos_clean dentro de la base de datos sgdi_pr2. Una vez creada la colección monumentos_clean, **se debe crear un índice de tipo 2dsphere sobre el campo location** recién creado.

   A modo de ejemplo, cada uno de los monumentos debe ser representado en monumentos_clean en documentos con este aspecto:

   ```
   {
     "_id": ...,
     "@id": "https://datos.madrid.es/egob/catalogo/tipo/monumento/424012-encuentro
        -playa.json",
     "id": "424012",
     "title": "Encuentro en la playa",
     "relation": "https://patrimonioypaisaje.madrid.es/sites/v/index.jsp...",
     "references": "https://patrimonioypaisaje.madrid.es/FrameWork/...",
     "address": {
       "district": {
         "@id":
         "https://datos.madrid.es/egob/kos/Provincia/Madrid/Municipio/Madrid/
            Distrito/PuenteDeVallecas"
   ```

```
    },
    "area": {
      "@id":
      "https://datos.madrid.es/egob/kos/Provincia/Madrid/Municipio/Madrid/
         Distrito/PuenteDeVallecas/Barrio/PalomerasBajas"
    },
    "locality": "MADRID",
    "postal-code": "",
    "street-address": "Trav Felipe de Diego 2"
  },
  "location": {
    "type": "Point",
    "coordinates": [
      -3.6633882416814347,
      40.37962183657809
    ]
  },
  "organization": {
    "organization-desc": "Esta obra es la última de una serie de esculturas y
       mosaicos
    decorativos...",
    "organization-name": "Encuentro en la playa"
  }
}
```

Tanto la ejecución de la tubería de agregación para crear la nueva colección monumentos_clean como la creación del índice de tipo 2dsphere debe realizarse con código Python dentro de la función agg_clean(mongoclient) que recibe una conexión abierta con MongoDB.

2. (0.75pt) Realiza una consulta que procese la colección monumentos_clean dentro de la base de datos sgdi_pr2 y devuelva el título y la calle (campo street−address) de los monumentos que están como mucho a n kilómetros de la Facultad de Informática de la UCM, ordenados por cercanía. Para ello debéis considerar que el edificio de la Facultad de Informática está en longitud -3.73336109904238 y latitud 40.45265933919037. Esta consulta debe estar encapsulada en una función Python geo_query1(mongoclient, n) que acepta una conexión abierta con MongoDB (mongoclient) y el parámetro n para devolver un **objeto pymongo.cursor.Cursor**, como en los apartados anteriores.

   Un ejemplo del resultado de geo_query1 sería:

```
{'title': 'José Ortega y Gasset',
  'address': {'street-address': 'PLAZA MENENDEZ PELAYO 4'}}
{'title': 'Camilo José Cela',
  'address': {'street-address': 'Pza Menéndez Pelayo '}}
{'title': 'Alfonso XIII', 'address': {'street-address': 'Avda Complutense '}}
{'title': 'Puerta de Hierro',
  'address': {'street-address': 'AVENIDA PADRE HUIDOBRO 1'}}
...
```

3. (0.75pt) Realiza una consulta que procese la colección monumentos_clean dentro de la base de datos sgdi_pr2 y devuelva el título y la calle (campo street−address) de los monumentos que están

en Ciudad Universitaria. Para esta consulta asumiremos que Ciudad Universitaria es una región cuadrada con esquina inferior izquierda en el punto

(-3.741817918089083, 40.434407294070)

y esquina superior derecha en el punto[1]

(-3.725071065360681, 40.45553987197873)

Esta consulta debe estar encapsulada en una función Python geo_query2(mongoclient) que acepta una conexión abierta con MongoDB y devuelve un **objeto pymongo.cursor.Cursor**, como en los apartados anteriores. Un ejemplo del resultado de geo_query2 sería:

```
{'title': 'Agricultor, agricultura y progreso',
  'address': {'street-address': 'Avda Complutense '}}
{'title': 'Alfonso XIII', 'address': {'street-address': 'Avda Complutense '}}
{'title': 'Blas Lázaro e Ibiza',
  'address': {'street-address': 'Pza Ramón y Cajal 2'}}
{'title': 'Camilo José Cela',
  'address': {'street-address': 'Pza Menéndez Pelayo '}}
...
```

[1]En ambos casos los puntos están representados como pares (longitud , latitud).