UNIVERSIDAD DE HUELVA

MÁSTER EN INGENIERÍA EN INFORMÁTICA

Almacenamiento y Gestión de la Almacenamiento y Gestión de la Información

Proyecto: MongoDB

Ciclo lectivo 2022-2023

Profesor: Jacinto Mata Vazquez

Alumnos: Monastyrski, Carlos Alberto

Faria, Juliao

Fecha de Entrega: 26/01/2023

Índice

Proyecto: MongoDB

En el presente documento se detalla el desarrollo del proyecto de Almacenamiento y Gestión de la Almacenamiento y Gestión de la Información, de la sección de MongoDB. Adjuntos con este archivo se encuentra un documento en el cual se encuentra publicado el link al repositorio de Github donde se encuentra publicado el código del sistema junto con las indicaciones para su instalación y configuración.

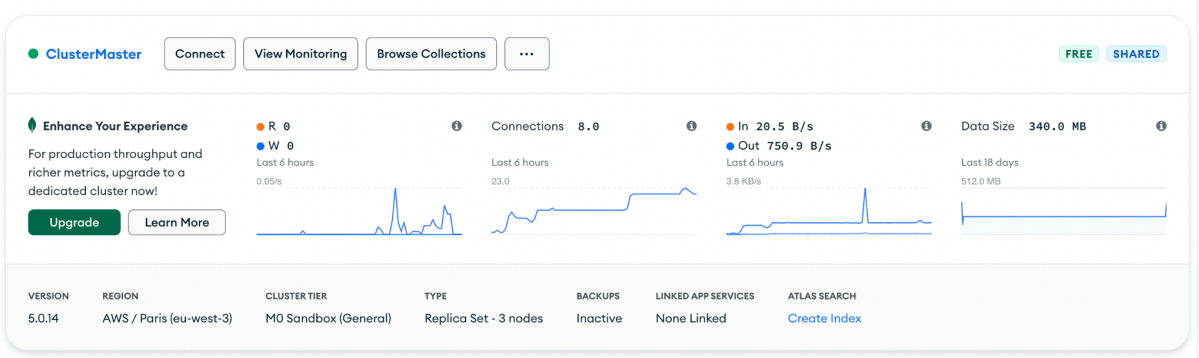
Tecnologías Utilizadas

- Script para captura de tweets: Python (el mismo es un archivo de extensión .py).
    - Propósito: es el script con el cual se obtuvieron los 50000 tweets y se enviaron a la base de datos mongo a ser guardados.
- Base de Datos: base de datos Mongo alojada en MongoDB Atlas, el cual es un servicio de Cloud Database.
    - Propósito: almacenar los 50000 tweets y recibir las distintas consultas que realizará el BackEnd. Es en pocas palabras nuestra base de datos, donde se almacenan los datos y se realizarán no solo las consultas sino también se gestionarán las colecciones (generando nuevas colecciones en algunos casos)
- Back End: NestJs, el cual es un framework de JavaScript para creación de APIs.
    - Propósito: es el encargado de gestionar los accesos a la base de datos, publicando distintos endpoints a los cuales se puede acceder mediante pedidos de tipo GET y POST para realizar las distintas consultas, así como la creación de nuevas colecciones y la autorización de los usuarios.
- React:

- Propósito: es la interfaz gráfica que el usuario puede ver y con la cual interactúa para realizar las consultas y creaciones de nuevas colecciones.

Descripción de la Base de Datos:

- La base de datos se aloja en un servidor de MongoDB Atlas de tier M0, alojado en un servidor AWS / Paris (eu-west-3).
- Se encuentra replicado en 3 nodos
- Conexiones máximas por nodo: 500
- Capacidad de almacenamiento: 512 MB
- El acceso es mediante usuario y contraseña, aunque recibe peticiones desde cualquier IP ya que fue necesario tener esa configuración para poder realizar peticiones desde la red de la Universidad.
- Se poseen dos colecciones principales: “users” y “tweets”. En “users” se crean los usuarios que pueden tener acceso a la aplicación creada y será utilizada para la autenticación de los mismos. Mientras que “tweets” es la colección donde almacenamos los tweets en el script mencionado anteriormente.
- La estructura de los documentos en la tabla “tweets” se adjuntan al final de esta memoria por cuestiones estéticas del mismo.



Se crearon 4 índices: uno de texto para la columna text de data, y tres para las columnas author_id, entities y tweets.

Para la creación del índice de texto se utilizó:

```
db.getCollection("tweets").createIndex( { "$**": "text" }, { name: "TextIndex" } )
```

Y para el resto de índices:

```
db.records.createIndex( { "nombre_campo": 1 } )
```

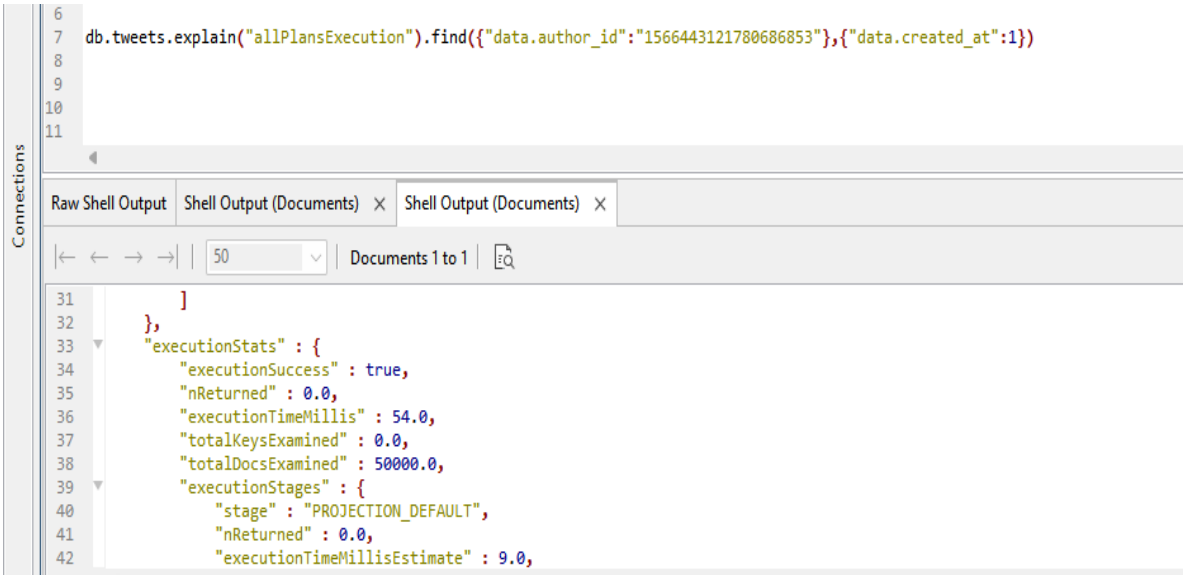
Name and Definition	Type	Size	Usage	Properties
> TextIndex	TEXT	50.3 MB	0	
> _id_	REGULAR	1.3 MB	0	UNIQUE
> data.author_id_1	REGULAR	880.6 KB	0	
> data.entities_1	REGULAR	11.9 MB	0	
> includes.tweets_1	REGULAR	88.3 MB	0	

Análisis Temporal de Consultas Mediante Índices

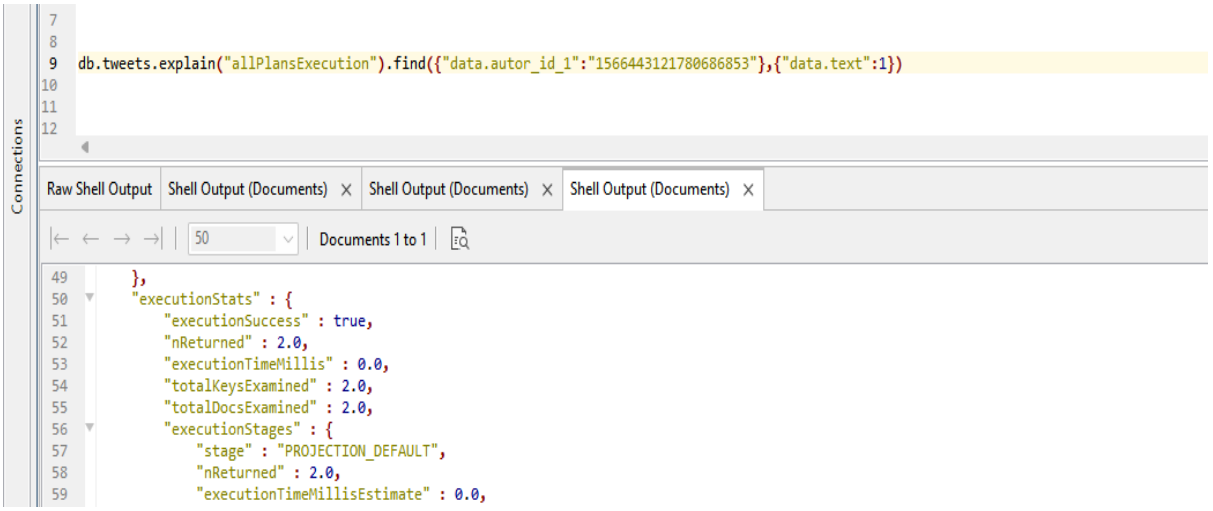
En los casos en que se almacenan grandes cantidades de información en las base de datos mongodb el uso de índices suelen ser muy útil cuando se pretende hacer consultas de lectura, en este apartado se hará una breve comparación de algunas consultas con y si índices.

Consulta de la fecha de creación del tweets sin indices

Resultado:



Consulta de la fecha de creación del tweets con indice



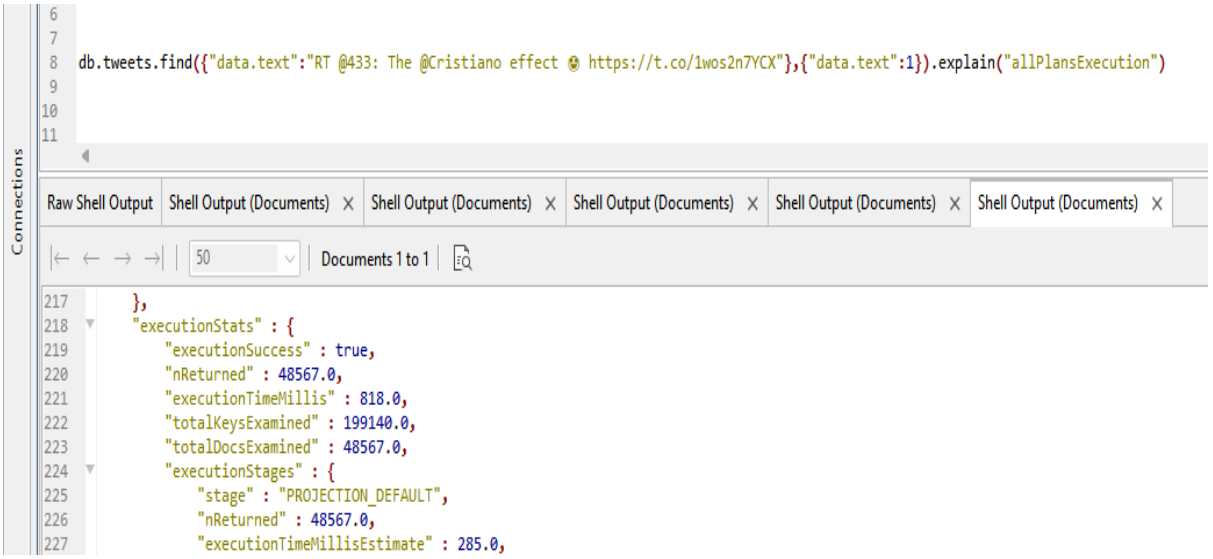
```
db.tweets.explain("allPlansExecution").find({"data.autor_id_1":"1566443121780686853"},{"data.text":1})
```

Raw Shell Output | Shell Output (Documents) | Shell Output (Documents) | Shell Output (Documents)

50 | Documents 1 to 1


```
    },
    "executionStats" : {
        "executionSuccess" : true,
        "nReturned" : 2.0,
        "executionTimeMillis" : 0.0,
        "totalKeysExamined" : 2.0,
        "totalDocsExamined" : 2.0,
        "executionStages" : {
            "stage" : "PROJECTION_DEFAULT",
            "nReturned" : 2.0,
            "executionTimeMillisEstimate" : 0.0,
```


Consulta sin índice y con índice de texto sobre el campo text.



```
db.tweets.find({"data.text":"RT @433: The @Cristiano effect 💀 https://t.co/1wos2n7YCX"},{"data.text":1}).explain("allPlansExecution")
```

Raw Shell Output | Shell Output (Documents) | Shell Output (Documents) | Shell Output (Documents) | Shell Output (Documents) | Shell Output (Documents)

50 | Documents 1 to 1


```
    },
    "executionStats" : {
        "executionSuccess" : true,
        "nReturned" : 48567.0,
        "executionTimeMillis" : 818.0,
        "totalKeysExamined" : 199140.0,
        "totalDocsExamined" : 48567.0,
        "executionStages" : {
            "stage" : "PROJECTION_DEFAULT",
            "nReturned" : 48567.0,
            "executionTimeMillisEstimate" : 285.0,
```

Como se puede analizar en casos de campos en los que solo se consultan y sin muchas escrituras vale usar los índices.

Consultas

Para la realización de las consultas se hizo uso del Framework Aggregate de mongo, y más precisamente utilizando pipelines.

- **Número de tweets en la colección:**

```
const pipeline = [{ $count: "totalRows" }];
```

Se puede observar que la consulta es bastante simple, se buscan todas las instancias en la colección y se devuelve un count de la cantidad de filas.

- **Fecha más antigua y reciente:**

```
const pipeline = [
  { $group: {
      _id: null,
      oldestDate: { $min: "$data.created_at" },
      newestDate: { $max: "$data.created_at" }
  } }
];
```

En esta consulta se busca la fecha más antigua del campo “created_at” del documento data de los tweets, así como el más reciente del mismo campo.

- **Texto del tweet más retwitteado:**

```
const pipeline = [
  {
    '$sort': {
      'data.public_metrics.retweet_count': -1
    }
  }, {
    '$limit': 1
  }, {
    '$project': {
      '_id': 0,
      'text': '$data.text'
    }
  }
]
```

En un primer paso se ordenan los documentos por cantidad de retweets de forma descendente, se obtiene el registro con el mayor número de retweets y finalmente se proyecta solamente el texto de dicho tweet.

- **10 Usuarios con más tweets:**

```
const pipeline = [
  {
    '$group': {
      '_id': '$data.author_id',
      'author_name': {
        '$first': '$includes.users'
      },
      'count': {
        '$sum': 1
      }
    }
  }, {
    '$sort': {
      'count': -1
    }
  }, {
    '$project': {
      '_id': 1,
      'count': 1,
      'author_name': {
        '$arrayElemAt': [
          '$author_name', 0
```

```
          ]
        }
      }
    }, {
      '$limit': 10
    }
  ]
```

Primero se agrupa por autor del tweet, obteniendo el nombre del mismo del primer registro del documento users, el cual está incluido en el documento includes. En este paso además se obtiene el número de tweets de cada uno. En el segundo paso se ordena descendentemente por número de tweets. Se procede a proyectar la información de los usuarios. Y finalmente se filtran solo los primeros 10 usuarios.

- **Texto del Tweet con más likes:**

```
const pipeline = [
  {
    '$sort': {
      'data.public_metrics.like_count': -1
    }
  }, {
    '$limit': 1
  }, {
    '$project': {
      '_id': 0,
      'text': '$data.text'
    }
  }
]
```

En esta consulta se comienza ordenando por número de likes de los tweets, se obtiene el que más likes posee y se proyecta su texto.

- **Cantidad de Tweets por lenguaje:**

```
const pipeline = [
  {
    '$group': {
      '_id': '$data.lang',
      'count': {
        '$sum': 1
      }
```

```
    }
  }, {
    '$project': {
      '_id': 0,
      'lang': '$_id',
      'tweets': '$count'
    }
  }
]
```

Como primer paso se agrupa por lenguaje, sumando 1 por cada tweet. Y se procede a proyectar la información del nombre del lenguaje y la cantidad de tweets que tiene registrado.

- **10 URLS con más apariciones:**

```
const pipeline = [
  {
    '$unwind': {
      'path': '$data.entities.urls'
    }
  }, {
    '$group': {
      '_id': '$data.entities.urls.url',
      'count': {
        '$sum': 1
      }
    }
  }, {
    '$sort': {
      'count': -1
    }
  }, {
    '$limit': 10
  }, {
    '$project': {
      '_id': 0,
      'cantidad': '$count',
      'url': '$_id'
    }
  }
]
```

Se comienza desglosando las distintas Urls que puede tener un tweet que están almacenadas en el documento entities del documento data.
Continuamos agrupando por la url, sumando 1 por cada aparición.
Se ordena el resultado por número de apariciones de mayor a menor.
Se obtienen sólo los primeros 10 registros. Y finalmente se proyectan los datos obtenidos.

- **10 usuarios más mencionados:**

```
const pipeline = [
  {
    '$unwind': {
      'path': '$data.entities.mentions'
    }
  }, {
    '$group': {
      '_id': '$data.entities.mentions.username',
      'count': {
        '$sum': 1
      }
    }
  }, {
    '$sort': {
      'count': -1
    }
  }, {
    '$limit': 10
  }, {
    '$project': {
      '_id': 0,
      'mentions': '$count',
      'username': '$_id'
    }
  }
]
```

Se comienza desglosando las distintas menciones que puede tener un tweet que están almacenadas en el documento entities del documento data.
Continuamos agrupando por el “username” del usuario mencionado, sumando 1 por cada aparición.
Se ordena el resultado por número de apariciones de mayor a menor.

Se obtienen sólo los primeros 10 registros. Y finalmente se proyectan los datos obtenidos.

- **Anotación con más apariciones por tipo:**

```
const pipeline = [
  {
    '$unwind': {
      'path': '$data.entities.annotations'
    }
  }, {
    '$group': {
      '_id': {
        'type': '$data.entities.annotations.type',
        'text': '$data.entities.annotations.normalized_text'
      },
      'count': {
        '$sum': 1
      }
    }
  }, {
    '$group': {
      '_id': '$_id.type',
      'elements': {
        '$push': {
          'text': '$_id.text',
          'count': '$count'
        }
      }
    }
  }, {
    '$project': {
      '_id': 0,
      'type': '$_id',
      'annotation': {
        '$arrayElemAt': [
          '$elements', {
            '$indexOfArray': [
              '$elements.count', {
                '$max': '$elements.count'
              }
            ]
          }
        ]
      }
```

```
      }
    }
  ]
```

Pasos:

1. Desglosar las anotaciones de los tweets
2. Agrupar por tipo de anotación y el texto normalizado de dicha anotación, contando la cantidad de apariciones de dicha combinación
3. Se agrupa por tipo, agregando en un array los distintos textos normalizados con sus apariciones
4. Se proyecta el nombre de cada tipo, con el texto normalizado y la cantidad de apariciones de aquel que más apareció en ese tipo.

- **10 países con más apariciones:**

```
const pipeline = [
  {
    '$unwind': {
      'path': '$includes.places'
    }
  }, {
    '$group': {
      '_id': '$includes.places.country',
      'count': {
        '$sum': 1
      }
    }
  }, {
    '$sort': {
      'count': -1
    }
  }, {
    '$limit': 10
  }
]
```

Se comienza desglosando los distintos lugares almacenados en el documento places del documento includes.
Se agrupa por país, contando sus apariciones.
Se ordena por número de apariciones de mayor a menor.

Se obtienen los primeros 10 registros.

- **Consulta parametrizada:**

```
const pipeline = []
if (body.startDate != undefined && body.endDate != undefined) {
  pipeline.push(
    {
      '$match': {
        '$expr': {
          '$and': [
            {
              '$gte': [
                {
                  '$dateFromString': {
                    'dateString': '$data.created_at'
                  }
                }, new Date(body.startDate)
              ]
            }, {
              '$lte': [
                {
                  '$dateFromString': {
                    'dateString': '$data.created_at'
                  }
                }, new Date(body.endDate)
              ]
            }
          ]
        }
      }
    }
  )
}
if (body.text != undefined) {
  pipeline.push(
    {
      '$match': {
        'data.text': {
          '$regex': `${body.text}`,
          '$options': 'im'
```

```
          }
        }
      }
    )
  }
  if (body.userName != undefined) {
    pipeline.push(
      {
        '$match': {
          'includes.users.0.username': {
            '$regex': `${body.userName}`,
            '$options': 'g'
          }
        }
      }
    )
  }
  if (body.retweet != undefined) {
    pipeline.push(
      {
        '$match': {
          'data.public_metrics.retweet_count': {
            '$gte': body.retweet
          }
        }
      }
    )
  }
  if (body.mentions != undefined) {
    pipeline.push(
      {
        '$match': {
          '$expr': {
            '$gte': [
              {
                '$size': {
                  '$ifNull': [
                    '$data.entities.mentions', []
                  ]
                }
              }, body.mentions
            ]
          }
        }
      }
```

```
    )
  }
  if (body.lang != undefined) {
    pipeline.push(
      {
        '$match': {
          'data.lang': `${body.lang}`
        }
      }
    )
  }
  pipeline.push(
    {
      "$out": `${body.newCollectionName}`
  }
  )
```

Se puede observar que solo se agrega el filtro si dentro del cuerpo de la petición se encuentran los valores para realizar dicho filtrado.

- Para filtrar por periodo de creación se hace un match en donde la fecha de creación sea mayor/igual al parámetro de inicio y menor igual al parámetro de fin. Se destaca el uso de dateFromString para convertir el string al formato de fecha.
- Para filtrar por texto del tweet se hace un match en donde se busca que el texto del mismo contenga la cadena de caracteres proporcionada. (No es sensible a mayúsculas/minúsculas)
- Para filtrar por el usuario que realizó el tweet se hace uso de un match entre el string con el nombre del usuario con el campo username del primer registro del documento users del documento includes.
- Para filtrar por número de retweets mayor a cierto número se hace un match entre el número proporcionado y el campo retweet_count del documento public_metrics de data.
- Para filtrar por número de menciones mayor a cierto número se hace un match entre el número proporcionado y la cantidad de documentos del array mentions del documento entities del documento data.
- Para filtrar por lenguaje del tweet se hace un match del código del lenguaje proporcionado con el campo lang de data.

- ❖ Por último el resultado de esta consulta se almacena en una nueva colección con nombre igual al proporcionado en el cuerpo de la petición.

Se hizo uso de esta consulta de soporte para este sistema:

- Obtener todos los nombres de los usuarios que son autores de al menos un tweet:

```
const pipeline = [
  {
    '$unwind': {
      'path': '$includes.users'
    }
  }, {
    '$match': {
      '$expr': {
        '$eq': [
          '$data.author_id', '$includes.users.id'
        ]
      }
    }
  }, {
    '$group': {
      '_id': '$includes.users.username'
    }
  }, {
    '$project': {
      '_id': 0,
      'nombre': '$_id'
    }
  }
]
```

El funcionamiento de esta consulta es:

1. Se desglosa el array con los usuarios
2. Se hace un match entre el usuario que generó el tweet y el usuario que quedó desglosado en el paso anterior para solo quedarnos con los autores
3. Se agrupa por nombre de usuario

4. Se proyecta la información con el campo llamándose “nombre”

Indicaciones de Instalación

Los archivos pueden ser descargados desde el siguiente repositorio de Github: https://github.com/juliao-faria/UHU-Twitter.git

Son necesarios:

- Python 3.9.15
- Npm: 16.5

Ahora en lo que respecta a cada componente:

- Para el archivo de recopilación de tweets es necesario tener instalado Python, junto con las librerías tweepy, json, pymongo y keys. Además de incorporar en el código la url de conexión a la base de datos y token a la api de twitter.
- Para el back end, abrimos una terminal y nos dirigimos a la carpeta back-end y ejecutamos los siguientes comandos:
    - npm install
    - npm run build
    - npm run start

  El primero instala las dependencias, el segundo construye la aplicación y con el tercero lo inicializamos y comienza a recibir peticiones en el puerto 9876.
- Para el front end es bastante similar, situándonos en la terminal en la carpeta front-end ejecutamos:
    - npm install
    - npm run start

Al igual que antes, el primero instala las dependencias, mientras que el segundo inicializa la aplicación corriendo en el puerto 3000

Conclusiones

Primeramente nos gustaría destacar lo enriquecedor de esta práctica, ya que durante el cursado de la asignatura tuvimos mucho tiempo para realizar consultas aisladas y probar distintos tipos de queries, pero fue en este proyecto que pudimos visualizar de forma concreta cómo se incorpora Mongo a un contexto real de una aplicación, sus potencialidades como sus limitaciones en algunos casos (ya que por ejemplo para lenguajes fuertemente tipados existe un desaprovechamiento de la potencialidad de mongo).

Fue muy interesante tener un contexto como la información de un tweet, para posteriormente generar en base a este un conjunto de funcionalidades que puedan describir su contenido y den información al usuario.

En cuanto a propuestas de mejoras sólo nos hubiera gustado tener un poco más de tiempo para practicar un poco más en los aspectos distribuidos de mongo, ahondando en sus detalles y cómo aprovecharlos al máximo.

Anexo

Estructura de los documentos de la tabla “tweets”:


```
{
   "_id" : ObjectId(string),
   "data" : {
      "author_id" : string,
      "created_at" : string,
      "edit_history_tweet_ids" : string[],
      "entities" : {
         "annotations" : [
            {
               "start" : number,
               "end" : number,
               "probability" : number,
               "type" : string,
               "normalized_text" : string
            },
            {
               "start" : number,
               "end" : number,
               "probability" : number,
               "type" : string,
               "normalized_text" : string
            },
            {
               "start" : number,
               "end" : number,
               "probability" : number,
               "type" : "Other",
               "normalized_text" : string,
```

```
        }
    ],
    "mentions" : [
        {
            "start" : number,
            "end" : number,
            "username" : string,
            "id" : string
        },
        {
            "start" : number,
            "end" : number,
            "username" : string,
            "id" : string
        }
    ],
    "urls" : [
        {
            "start" : number,
            "end" : number,
            "url" : string,
            "expanded_url" : string,
            "display_url" : string,
            "media_key" : string
        }
    ]
},
"geo" : {
            “place_id”: string
},
"id" : string,
"lang" : string,
"public_metrics" : {
    "retweet_count" : number,
```

```
        "reply_count" : number,
        "like_count" : number,
        "quote_count" : number
    },
    "referenced_tweets" : [
        {
            "type" : string,
            "id" : string
        }
    ],
    "text" : string
},
"includes" : {
    "users" : [
        {
            "created_at" : string,
            "description" : string,
            "entities" : {
                "url" : {
                    "urls" : [
                        {
                            "start" : number,
                            "end" : number,
                            "url" : string,
                            "expanded_url" : string,
                            "display_url" : string
                        }
                    ]
                }
            },
            "id" : string,
            "name" : string,
            "public_metrics" : {
                "followers_count" : number,
```

```
        "following_count" : number,
        "tweet_count" : number,
        "listed_count" : number
    },
    "username" : string,
    "verified" : boolean
},
{
    "created_at" : string,
    "description" : string,
    "entities" : {
        "url" : {
            "urls" : [
                {
                    "start" : number,
                    "end" : number,
                    "url" : string,
                    "expanded_url" : string,
                    "display_url" : string
                }
            ]
        }
    },
    "id" : string,
    "name" : "farqaleit",
    "public_metrics" : {
        "followers_count" : number,
        "following_count" : number,
        "tweet_count" : number,
        "listed_count" : number
    },
    "username" : string,
    "verified" : boolean
},
```

```
    {
        "created_at" : string,
        "description" : string,
        "entities" : {
            "url" : {
                "urls" : [
                    {
                        "start" : number,
                        "end" : number,
                        "url" : string,
                        "expanded_url" : string,
                        "display_url" : string
                    }
                ]
            }
        },
        "id" : string,
        "location" : string,
        "name" : string,
        "public_metrics" : {
            "followers_count" : number,
            "following_count" : number,
            "tweet_count" : number,
            "listed_count" : number
        },
        "username" : string,
        "verified" : boolean
    }
],
"tweets" : [
    {
        "author_id" : string,
        "created_at" : string
        "edit_history_tweet_ids" : string[],
```

```
"entities" : {
   "annotations" : [
      {
         "start" : number,
         "end" : number,
         "probability" : number,
         "type" : string,
         "normalized_text" : string
      },
      {
         "start" : number,
         "end" : number,
         "probability" : number,
         "type" : string,
         "normalized_text" : string
      },
      {
         "start" : number,
         "end" : number,
         "probability" : number,
         "type" : string,
         "normalized_text" : string
      }
   ],
   "mentions" : [
      {
         "start" : number,
         "end" : number,
         "username" : string,
         "id" : string
      },
      {
         "start" : number,
         "end" : number,
```

```
            "username" : string,
            "id" : string
        }
      ],
      "urls" : [
        {
          "start" : number,
          "end" : number,
          "url" : string,
          "expanded_url" : string,
          "display_url" : string,
          "media_key" : string
        }
      ]
    },
    "geo" : {
        “place_id”: string
    },
    "id" : string,
    "lang" : string,
    "public_metrics" : {
      "retweet_count" : number,
      "reply_count" : number,
      "like_count" : number,
      "quote_count" : number
    },
    "referenced_tweets" : [
      {
        "type" : string,
        "id" : string
      }
    ],
    "text" : string
},
```

```
{
    "author_id" : string,
    "created_at" : string,
    "edit_history_tweet_ids" : string[],
    "entities" : {
        "mentions" : [
            {
                "start" : number
                "end" : number,
                "username" : string,
                "id" : string
            }
        ],
        "urls" : [
            {
                "start" : number,
                "end" : number,
                "url" : string,
                "expanded_url" : string,
                "display_url" : string,
                "media_key" : string
            }
        ]
    },
    "geo" : {
         “place_id”: string
    },
    "id" : "1611012548526215168",
    "lang" : "en",
    "public_metrics" : {
        "retweet_count" : number,
        "reply_count" : number,
        "like_count" : number,
        "quote_count" : number
```

```
            },
            "text" : string
        }
    ]
},
"matching_rules" : [
    {
        "id" : string,
        "tag" : string
    }
]
}
```