1. **Infraestructura de la base de datos**

La infraestructura levantada cuenta con un replica set en el que existe un nodo principal de lectura-escritura y dos nodos secundarios que aseguran la disponibilidad a la hora de efectuar lecturas, guardando copias de los documentos contenidos en la base de datos. En este caso no se está utilizando una estructura de sharding, dado el bajo volumen de datos guardados. Se consideraría usar esta estructura para un nivel mucho mayor de estos para ahorrar espacio en la base de datos, mientras que se asegura la disponibilidad mediante las copias en los nodos de los replica sets.

El repositorio cuenta con tres bases de datos. “news\_blog“ (ejercicio 2), que contiene tres colecciones distintas, webs\_db (ejercicio 3), que contiene sólo una y educacional (ejercicio 4), con dos colecciones.

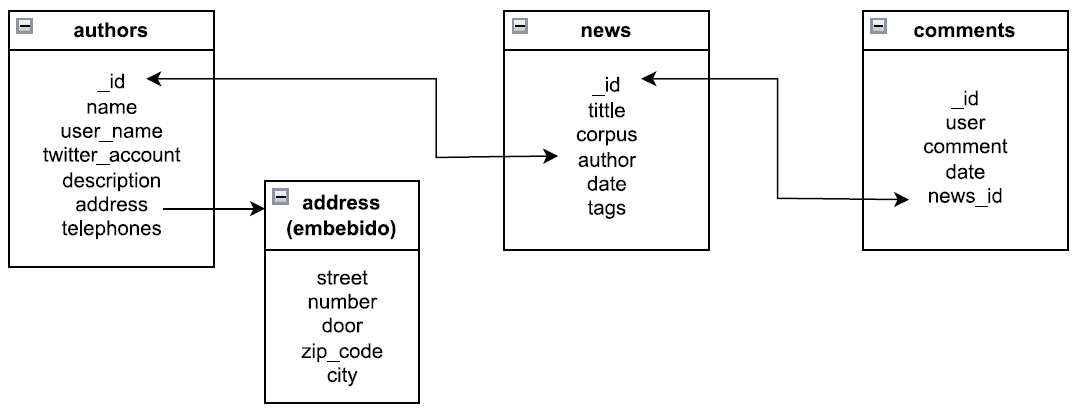
Cada una de estas colecciones se ha creado con diferentes esquemas de validación para asegurar la integridad y calidad de los datos y en cada una se han creado distintos índices que permitan por un lado efectuar búsquedas de manera eficiente y en algunos casos, asegurar la unicidad de documentos que presenten­­­ el mismo valor en alguno de sus campos.

**Cadena para la conexión de sólo-lectura:** mongodb+srv://Reader:1234@myfirstmongodbcluster.lbmtt5n.mongodb.net/

1. **Blog de noticias (news\_blog):**

Voy a crear un total de tres colecciones, una para los usuarios, otra para las noticias y otra para los comentarios. Los documentos de usuario pueden llevar embebidos documentos de dirección (address), de forma que se reduce el número de estos en la base de datos y permite realizar consultas más rápidas sobre los campos relacionados, al estar en una ubicación común.

Las colecciones creadas y los campos que contiene cada documento se muestran en la siguiente figura.



Para cada colección vamos a establecer unos criterios para la validación que van a hacer que al introducir un nuevo documento, este no se cree si no cumple con las reglas definidas.

* Colección autores (authors):

En esta colección los campos name, user\_name, twitter\_account y description son obligatorios. Además, defino el tipo de dato que guarda cada campo. Como ya se ha mencionado, cada autor lleva embebido un documento dirección, que contiene sus propios campos.

* Creación de índice sobre el campo nombre. Con la creación de este índice aseguro la unicidad de los usuarios



* Creación de la colección con validación

db.createCollection("authors", {

"validator": {

"$jsonSchema": {

"bsonType": "object",

"title": "Authors object validation",

"required": ["name", "user\_name", "twitter\_account", "description"],

"properties": {

"name": {

"bsonType": "string",

"description": "must be a string and is required"

},

"user\_name": {

"bsonType": "string",

"description": "must be a string and is required"

},

"twitter\_account": {

"bsonType": "string",

"description": "must be a string and is required"

},

"description": {

"bsonType": "string",

"description": "must be a string and is required"

},

"address": {

"bsonType": "object",

"description": "must be an object and is not required",

},

"address.street": {

"bsonType": "string",

"description": "must be a string and is not required"

},

"address.number": {

"bsonType": "int",

"description": "must be an integer and is not required"

},

"address.door": {

"bsonType": "string",

"description": "must be a string or an integer and is not required"

},

"address.zip\_code": {

"bsonType": "int",

"description": "must be an integer and is not required"

},

"address.city": {

"bsonType": "string",

"description": "must be a string and is not required"

},

"telephone": {

"bsonType": "array"}

}

}

}

})

* Población de la colección

db.authors.insertMany([

{

"name": "Johny Bravo" ,

"user\_name": "JohnyBravo",

"twitter\_account": "jb\_twitter",

"description": "JB, but cooler than Justin Biever",

"address": {

"street": "Main St",

"number": 1,

"door": "3",

"zip\_code": 34678,

"city": "NYC",

"telephones": [123456789, 987654321]

}

},

{

"name": "Justin Biever",

"user\_name": "JustinBiever",

"twitter\_account": "DaRealJustinBiever",

"description": "JB, but not so cool as JohnyBravo",

"address": {

"street": "Main St",

"number": 2,

"door": "A",

"zip\_code": 34678,

"city": "NYC",

"telephones": [345678925, 465858643]

}

},

Una vez poblada la colección, se puede probar a crear un usuario duplicado, lo cuál debería devolver un error.

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza baja

* Colección noticias (news):
* Creación

Los documentos de esta colección contienen los campos “título”, “cuerpo”, “fecha” (un objeto de tipo fecha), “autor” (que contiene el objeto ObjectId del usuario autor de la noticia) y “tags/etiquetas”, de los cuales “autor” y “etiquetas” no son obligatorios.

{

    "validator": {

        "$jsonSchema": {

            "bsonType": "object",

            "title": "News object validation",

            "required": ["title", "corpus", "date", "tags"],

            "properties": {

                "title": {

                    "bsonType": "string",

                    "description": "must be a string and is required"

                },

                "corpus": {

                    "bsonType": "string",

                    "description": "must be a string and is required"

                },

                "date": {

                    "bsonType": "date",

                    "description": "must be a date and is required"

                },

                "author": {

                    "bsonType": "string",

                    "description": "must be a string and is not required"

                },

                "tags": {

                    "bsonType": "array",

                    "description": "must be an array and is not required"

                }

            }

        }

    }

}

* Población de la colección

Como en los campos autor estoy almacenando los ObjectId de usuario, extraigo estos \_ids con find de la colección autores, usando el nombre de usuario como condición y proyectando sobre el \_id:



db.news.insertMany([

{

"title": "Justin Biever on concert 💖",

"corpus": "I'll be waiting for you to share the best music at my roof party tonight",

"author": "6618084527337aeb04dcd397",

"date": new ISODate("2022-01-01"),

"tags": ["music", "party", "JustinBiever"]

},

{

"title": "Snoop Dogg Happy-Flower tour",

"corpus": "🎤 Snoop Dogg and Wiz Khalifa are teaming up for their High School Reunion Tour!🎉\nThis multi-city jaunt, slated to kick off on July 7, will feature West Coast legends Warren G,\nToo $hort, and Berner, with special guest DJ Drama. ",

"author": "6618084527337aeb04dcd397",

"date": new ISODate("2022-01-01"),

"tags": ["music", "rap", "SnoopDogg"]

},

* Colección comentarios (comments):
* Creación

Esta colección contiene los campos “usuario” (autor del comentario), “comentario”, “fecha” y “id de la noticia”. Este último campo nos permite hacer consultas y agregaciones de comentarios por cada noticia” y es el único campo no obligatorio, acorde con lo que pide el ejercicio.

{

    "validator": {

        "$jsonSchema": {

            "bsonType": "object",

            "title": "Comments object validation",

            "required": ["user", "comment", "date"],

            "properties": {

                "user": {

                    "bsonType": "string",

                    "description": "must be a string and is required"

                },

                "comment": {

                    "bsonType": "string",

                    "description": "must be a string and is required"

                },

                "date": {

                    "bsonType": "date",

                    "description": "must be a date and is required"

                },

                "news\_id": {

                    "bsonType": "string",

                    "description": "must be a string and is required"

                }

            }

        }

    }

}

Texto

Descripción generada automáticamente

* Creación de otros índices



Consultas de noticias de un autor. Se debe crear un índice sobre el campo autor de la colección noticias



Número de comentarios por noticia, día y usuario:

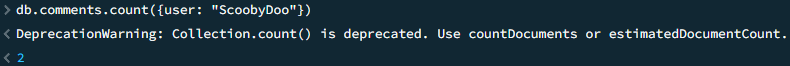


Ejemplos de consultas frecuentes:

2.1 Por nombre de usuario

Texto

Descripción generada automáticamente

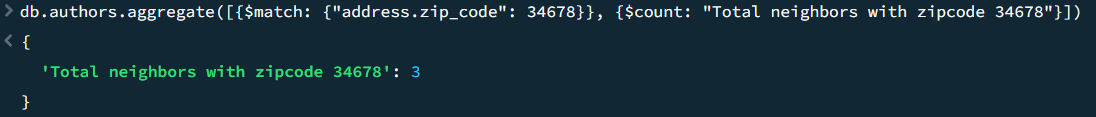


2.2 Por cuenta de twitter

Texto

Descripción generada automáticamente

2.3 Agregaciones por código postal consultando el número de usuarios que tienen el mismo C.P.



2.4 Noticias de un usuario, ordenadas por fecha (las "n" últimas noticias publicadas, de la más reciente a la más antigua)(con n=2, por ejemplo)

Texto

Descripción generada automáticamente

2.5 Número de comentarios por noticia, por día o por usuario.

Texto

Descripción generada automáticamente con confianza media

Imagen que contiene Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

1. **Webs\_db: Consultas y agregaciones**

Consulta1:



Consulta2:

Texto

Descripción generada automáticamente

Consulta3:

Texto

Descripción generada automáticamente

Da 20 salidas iguales. Para asegurarme de que no tuviera que salir algún otro explorador, hice esta otra consulta



Agregación1:

Texto

Descripción generada automáticamente con confianza baja

Agregación2:

Pantalla de computadora con letras

Descripción generada automáticamente con confianza media

Agregación3:

Texto

Descripción generada automáticamente

Agregación4:

Texto

Descripción generada automáticamente

Agregación5:

Texto

Descripción generada automáticamente

Agregación6:

Texto

Descripción generada automáticamente

1. **Base de datos: educacional**

La base de datos propuesta contiene cursos de una plataforma de educación, además de los usuarios registrados, que pueden tener el rol de estudiante, profesor o ambos.

La ventaja de usar una herramienta como mongodb es que no tiene un esquema definido y proporciona flexibilidad a la hora de cursos con diferentes estructuras y tipos de contenido.

La colección “cursos” tiene tres campos obligatorios "nombre", "descripcion" y "fecha\_inicio" y los documentos llevan embebidos documentos de los temas enseñados en cada curso, que a su vez embeben los documentos de recursos. Otros campos como fecha de fin son libres, ya que se da la flexibilidad de crear cursos, por ejemplo, que se vayan actualizando con el tiempo y a los que el usuario pueda acceder indefinidamente.

La colección usuarios tiene como campos obligatorios a "nombre y apellidos", "email" y "tipo\_usuario" y embebe los documentos cursos inscritos y cursos enseñados, en función de si el usuario es estudiante o profesor.

A screenshot of a computer

Description automatically generated

* **Índices**:

Antes de poblar la colección de cursos, se creó un índice sobre el nombre de los cursos para asegurar la unicidad de este campo.



Asimismo, se creó un índice para asegurar la unicidad de los correos, de forma que no se permitiera crear varias cuentas de usuario con un mismo email.



Población de las colecciones:

{

    "validator": {

        "$jsonSchema": {

            "bsonType": "object",

            "title": "cursos validator",

            "required": ["nombre", "descripcion", "fecha\_inicio"],

            "properties": {

                "nombre": {

                    "bsonType": "string",

                    "description": "must be a string and is required"

                },

                "descripcion": {

                    "bsonType": "string",

                    "description": "must be a string and is required"

                },

                "fecha\_inicio": {

                    "bsonType": "date",

                    "description": "must be a date and is required"

                },

                "fecha\_fin": {

                    "bsonType": "date",

                    "description": "must be a date and is not required"

                },

                "temas": {

                    "bsonType": "array",

                    "description": "must be an array and is not required",

                    "required": ["nombre"]

                },

                "temas.nombre": {

                    "bsonType": "string",

                    "description": "must be a string and is not required"

                },

                "temas.recursos": {

                    "bsonType": "array",

                    "description": "must be an array and is not required"

                },

                "temas.recursos.tipo": {

                    "bsonType": "string",

                    "description": "must be a string and is required"

                },

                "examenes": {

                    "bsonType": "array",

                    "description": "must be an array and is not required"

                },

                "profesor": {

                    "bsonType": "string",

                    "description": "must be a string and is not required"

                },

                "valoracion": {

                    "bsonType": "int",

                    "description": "must be an integer and is not required"

                }

            }

        }

    }

}

{

    "validator": {

        "$jsonSchema": {

            "bsonType": "object",

            "title": "usuarios validator",

            "required": ["nombre y apellidos", "email", "tipo\_usuario"],

            "properties": {

                "nombre y apellidos": {

                    "bsonType": "string",

                    "description": "must be a string and is required"

                },

                "email": {

                    "bsonType": "string",

                    "pattern": "^.+@.+$",

                    "description": "must be a string and is required. Must match the email format"

                },

                "tipo\_usuario": {

                    "bsonType": "array",

                    "items": {

                        "bsonType": "string",

                        "enum": ["alumno", "profesor"]

                    },

                    "description": "must be a string and is required. Can be one or both alumno and profesor"

                },

                "cursos\_inscritos": {

                    "bsonType": "array",

                    "description": "must be an array and is not required",

                    "items": {

                        "bsonType": "object",

                        "required": ["curso", "fecha\_inscripcion"],

                        "properties": {

                                "curso": {

                                    "bsonType": "string",

                                    "description": "must be a valid name and is required"

                                },

                                "fecha\_inscripcion": {

                                    "bsonType": "date",

                                    "description": "must be a date and is required"

                                }

                            }

                        }

                },

                "cursos\_enseñados": {

                    "bsonType": "array",

                    "items": {

                        "bsonType": "object",

                        "required": ["curso", "fecha\_inicio"],

                        "properties": {

                            "curso": {

                                "bsonType": "string",

                                "description": "must be a valid name and is required"

                            },

                            "fecha\_inicio": {

                                "bsonType": "date",

                                "description": "must be a date and is required"

                            }

                        }

                    },

                    "description": "must be an array and is not required"

                }

            }

        }

    }

}

Ejemplo de curso:

A screenshot of a computer

Description automatically generated

Ejemplo de usuario:

A computer screen shot of a computer code

Description automatically generated

Creación de índices para optimizar las queries:











Consultas más frecuentes:

Consulta 1: Por nombre del curso, de forma que se pueda buscar por temáticas.

Consulta 2: Cursos por valoración. Para buscar los mejores cursos.

Consulta 3: Por profesor. Ya que es muy posible que si a un alumno le ha gustado un profesor determinado, busquen más cursos impartidos por él.

Consulta 4: Por tipo de usuario (estudiante, profesor y ambos). Con fines estadísticos, se podría querer aplicar este tipo de filtro.

Consulta 5: Por nombre y apellidos. En general si se busca a un usuario, lo lógico es que se haga por su nombre.

Ejemplos y resultados:

Por nombre de curso, buscando un patrón de regular expression e ignorando mayúsculas (/i)

A computer screen shot of a program code

Description automatically generated

Por valoración (mayor o igual a 4)



Por profesor:

A computer code with green and blue text

Description automatically generated

Por tipo de usuario (agregación de número de usuarios por tipo):

A screen shot of a computer code

Description automatically generated

Usuario por nombre y apellidos:

A screen shot of a computer

Description automatically generated