ATS MongoDB 2025: Sistema de Inspecciones de Restaurantes

TAREAS OBLIGATORIAS (Hasta 4 puntos)

1. Diseño del esquema de la base de datos

- **Analizar la estructura de los datos y determinar el tipo de relación entre restaurantes e inspecciones (One-to-Few, One-to-Many, One-to-Millions).**

Analizando la estructura de ambos datasets y con la pequeña consulta a continuación:

```
db.getCollection('inspections').aggregate(
    [
      {
        $group: {
          _id: '$restaurant_id',
          count: { $sum: 1 }
        }
      },
      { $sort: { count: -1 } }
    ],
    { maxTimeMS: 60000, allowDiskUse: true }
);
```

```
{
  _id: '55f14312c7447c3da7051cfc',
  count: 3
}
{
  _id: '55f14312c7447c3da7051ff8',
  count: 3
}
{
  _id: '55f14312c7447c3da7051c7c',
  count: 3
}
{
  _id: '55f14312c7447c3da7051f55',
  count: 3
}
```

Observamos que los datos tienen una relación entre sí del tipo '*One-to-Few*' ya que cada restaurante, con la consulta ordenada de máximo a mínimo, tiene 3 inspecciones asociadas como mucho.

Arnau Muñoz Barrera (**1665982**)
Biel Ramon Polo (**1600209**)
ATS - G13_10_30

A pesar de esta relación, la estructura de los datos es realmente del tipo de '*One-to-Millions*' ya que podemos observar que los datos se encuentran referenciados para ser almacenados en datasets distintos, controlando así el crecimiento y optimizando el tiempo de espera al realizar consultas.

- **Justificar la elección de referencias (restaurant_id) en lugar de documentos embebidos. También puedes decidir crear una nueva colección que no utilice las referencias e incorpore los documentos embebidos. Justifica la decisión.**

La principal diferencia que encontramos entre las dos metodologías de trabajo es que, si referenciamos documentos, evitamos la duplicidad de datos y mejoramos la posible inserción futura de nueva información mientras que, si hacemos *'embeded'*, al encontrarse toda la información en un mismo lugar mejoramos el tiempo de respuesta de las consultas.

En nuestro caso, no vamos a hacer uso de nueva información a almacenar, haciendo que los datasets no crezcan y, a su vez, habiendo visto en el anterior apartado que cada restaurante tiene información asociada de como máximo 3 inspecciones, decidimos que al no tener una gran cantidad de información a analizar y el constante uso de las consultas, hacer uso del método de embeber para optimizar nuestras consultas es más beneficioso.

- **Definir un esquema de validación para ambas colecciones.**

* Código de los esquemas en el folder *'/scripts'* → *'validació ex1.3 inspections & validació ex1.3 restaurants'.*
En la creación de los esquemas de validación hemos seguido el criterio de comprobar todos los campos pero no hemos considerado todos relevantes como para considerarlos estrictamente necesarios para añadir los datos a la tabla.

Arnau Muñoz Barrera (**1665982**)
Biel Ramon Polo (**1600209**)
ATS - G13_10_30

Hemos ido haciendo las comprobaciones de cada una de las columnas con su tipo de valor esperado y de si el parámetro era opcional u obligatorio, añadiendo un mensaje de salida para mostrar el uso esperado en caso de error.

2. Implementación de consultas en MongoDB

- **Buscar todos los restaurantes de un tipo de comida específico (ej. "Chinese").**

  En esta consulta hacemos el filtrado de los restaurantes por tipo de comida mediante el campo *'type_of_food'* donde su valor entre comillas *type_of_food*: 'Chinese' (usado como ejemplo) puede ser reemplazado por el valor deseado para filtrar la consulta.

```
db.getCollection('restaurants').aggregate(
  [{ $match: { type_of_food: 'Chinese' } }],
  { maxTimeMS: 60000, allowDiskUse: true }
);
```

```
{
  _id: ObjectId('55f14312c7447c3da7051b35'),
  URL: 'http://www.just-eat.co.uk/restaurants-113-fish-bar-wallasey/menu',
  address: '113 Poulton Road',
  'address line 2': 'Merseyside',
  name: '113 Fish Bar',
  outcode: 'CH44',
  postcode: '9DE',
  rating: 5.5,
  type_of_food: 'Chinese'
}
```

- **Listar las inspecciones con violaciones, ordenadas por fecha.**

  En esta consulta tenemos que filtrar por el campo *'result'* el cual tiene opciones flexibles que contienen la palabra *'violation'*, por lo que tenemos que hacer uso de los símbolos **^** i **$** para indicar el inicio y final de la cadena y así solo consultar las inspecciones que hayan cometido alguna violación.

  Por otra parte, la ordenación por el campo de fecha, se realiza de manera descendente, sin embargo, ésta puede entrar en conflicto ya que, inicialmente, todos los componentes de la fecha *'MM - DD - YYYY'* eran valores enteros, y ahora són cadenas de string de longitud 3 formando una abreviación del mes en inglés.

Arnau Muñoz Barrera (**1665982**)
Biel Ramon Polo (**1600209**)
ATS - G13_10_30

```
db.getCollection('inspections').aggregate(
  [
    {
      $match: {
        result: {
          $regex: '^Violation Issued$',
          $options: ''
        }
      }
    },
    { $sort: { date: -1 } }
  ],
  { maxTimeMS: 60000, allowDiskUse: true }
);
```


```
{
  _id: ObjectId('56d61033a378eccde8a88a83'),
  id: '26253-2015-ENFO',
  certificate_number: 3021207,
  business_name: "AMEERA'S",
  date: 'Sep 30 2024',
  result: 'Violation Issued',
  sector: 'Grocery-Retail - 808',
  address: {
    city: 'TYNE AND WEAR',
    zip: '7AF',
    street: 'HYLTON ROAD',
    number: '61'
  },
  restaurant_id: '55f14312c7447c3da7051e48'
}
```


- **Encontrar restaurantes con una calificación superior a 4.**
  En esta consulta al dataset de restaurantes, filtramos el campo de rating, que contiene la valoración numérica asociada a cada restaurante, con el operador '*greater*' para encontrar las notas superiores a 4.

```
db.getCollection('restaurants').aggregate(
  [{ $match: { rating: { $gt: 4 } } }],
  { maxTimeMS: 60000, allowDiskUse: true }
);
```

```
{
  _id: ObjectId('55f14312c7447c3da7051b26'),
  URL: 'http://www.just-eat.co.uk/restaurants-cn-chinese-cardiff/menu',
  address: '228 City Road',
  'address line 2': 'Cardiff',
  name: '.CN Chinese',
  outcode: 'CF24',
  postcode: '3JH',
  rating: 5,
  type_of_food: 'Chinese'
}
```

3. Uso de agregaciones

- **Agrupar restaurantes por tipo de comida y calcular la calificación promedio.**
  En esta consulta al dataset de restaurantes, para acceder a la calificación en promedio, se hace uso de la operación de *$avg* calculando automáticamente el valor final con el conjunto de valores del campo *$rating*. Previo a este cálculo, se agrupan los valores a consultar según el tipo de comida, extraído del campo *$type_of_food*.

```
db.getCollection('restaurants').aggregate(
  [
    {
      $group: {
        _id: '$type_of_food',
        averageRating: { $avg: '$rating' }
      }
    },
    { $sort: { averageRating: 1 } }
  ],
  { maxTimeMS: 60000, allowDiskUse: true }
);
```

```
{
  _id: '*NEW*',
  averageRating: null
}
{
  _id: 'African',
  averageRating: 3.64
}
{
  _id: 'Vietnamese',
  averageRating: 4
}
{
  _id: 'Vegetarian',
  averageRating: 4
}
{
  _id: 'Healthy',
  averageRating: 4.166666666666667
}
```

- **Contar el número de inspecciones por resultado y mostrar los porcentajes.**
  En esta consulta al dataset de inspecciones,se realizan varias acciones, primeramente, se hace el recuento de la cantidad de inspecciones por resultado, mediante este primer valor, luego situamos el número total de inspecciones de todas las consultas. Para finalizar, mostrar el resultado en porcentaje del número de inspecciones por resultado, además de sus valores por separado, permitiéndonos comprobar que el resultado es el correcto.

Arnau Muñoz Barrera (**1665982**)
Biel Ramon Polo (**1600209**)
ATS - G13_10_30

```
db.getCollection('inspections').aggregate(
  [
    {
      $group: {
        _id: '$result',
        nInspectionsPerResult: { $sum: 1 }
      }
    },
    {
      $group: {
        _id: null,
        totalInspections: {
          $sum: '$nInspectionsPerResult'
        },
        data: { $push: '$$ROOT' }
      }
    },
```

```
    { $unwind: '$data' },
    {
      $project: {
        _id: '$data._id',
        nInspectionsPerResult:
          '$data.nInspectionsPerResult',
        inspectionsPercentage: {
          $multiply: [
            {
              $divide: [
                '$data.nInspectionsPerResult',
                '$totalInspections'
              ]
            },
            100
          ]
        }
      }
    },
    { $sort: { inspectionsPercentage: -1 } }
  ],
  { maxTimeMS: 60000, allowDiskUse: true }
);
```

```
{
  _id: 'Violation Issued',
  nInspectionsPerResult: 1291,
  inspectionsPercentage: 20.266875981161693
}
{
  _id: 'Warning Issued',
  nInspectionsPerResult: 1280,
  inspectionsPercentage: 20.09419152276295
}
{
  _id: 'Fail',
  nInspectionsPerResult: 1280,
  inspectionsPercentage: 20.09419152276295
}
{
  _id: 'No Violation Issued',
  nInspectionsPerResult: 1260,
  inspectionsPercentage: 19.78021978021978
}
{
  _id: 'Pass',
  nInspectionsPerResult: 1259,
  inspectionsPercentage: 19.76452119309262
}
```

- **Unir restaurantes con sus inspecciones utilizando $lookup.**
  Para poder unir los dos datasets en cuestión, para tener que modificar completamente la información de estos, hemos creado un campo adicional en *restaurants*, que concuerda en el tipo del campo, para poder compararlo y hacer *$match* exacto con el campo local de *inspections.*

Arnau Muñoz Barrera (**1665982**)
Biel Ramon Polo (**1600209**)
ATS - G13_10_30

```
db.getCollection('inspections').aggregate(
  [
    {
      $addFields: {
        restaurant_id_obj: {
          $toObjectId: '$restaurant_id'
        }
      }
    },
    {
      $lookup: {
        from: 'restaurants',
        localField: 'restaurant_id_obj',
        foreignField: '_id',
        as: 'Info'
      }
    }
  ],
  { maxTimeMS: 60000, allowDiskUse: true }
);
```

Este *$lookup* nos permite observar al mismo tiempo cada inspección con toda la información completa del restaurante asociado. La vista de la consulta final es la siguiente:

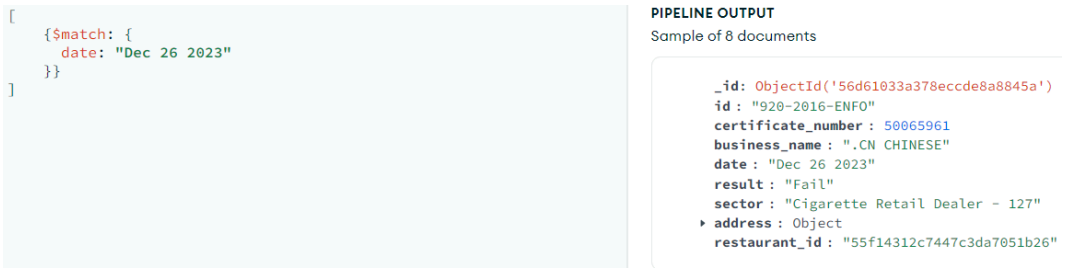
```
id : "920-2016-ENFO"
certificate_number : 50065961
business_name : ".CN CHINESE"
date : "Dec 26 2023"
result : "Fail"
sector : "Cigarette Retail Dealer - 127"
▾ address : Object
    city : "CARDIFF"
    zip : "3JH"
    street : "CITY ROAD"
    number : "228"
restaurant_id : "55f14312c7447c3da7051b26"
restaurant_id_obj : ObjectId('55f14312c7447c3da7051b26')
▾ Info : Array (1)
  ▾ 0: Object
      _id : ObjectId('55f14312c7447c3da7051b26')
      URL : "http://www.just-eat.co.uk/restaurants-cn-chinese-
            cardiff/menu"
      address : "228 City Road"
      address line 2 : "Cardiff"
      name : ".CN Chinese"
      outcode : "CF24"
      postcode : "3JH"
      rating : 5
      type_of_food : "Chinese"
```

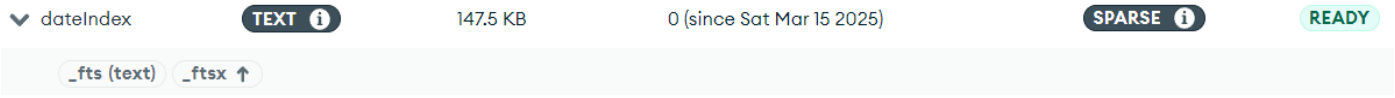
Arnau Muñoz Barrera (**1665982**)
Biel Ramon Polo (**1600209**)
ATS - G13_10_30

TAREAS AVANZADAS (Hasta 6 puntos)

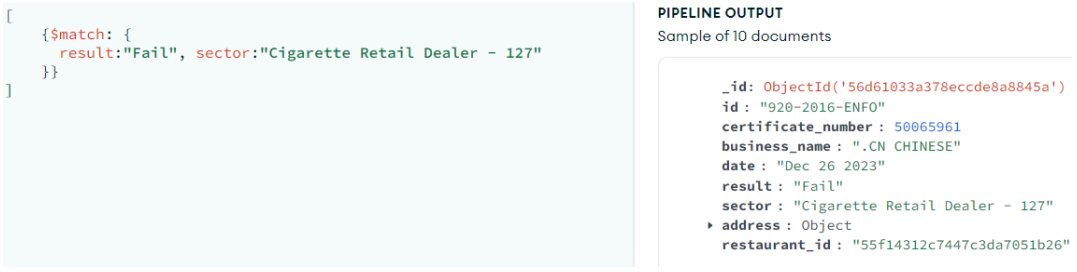
1. Optimización del rendimiento

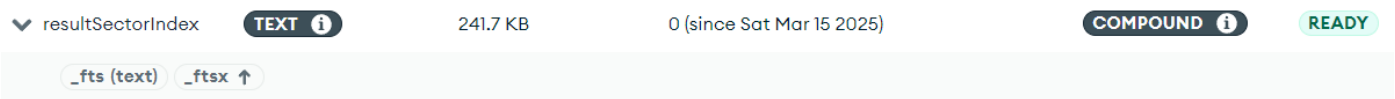
- **Identificar las posibles consultas más frecuentes.**
- **Implementar índices adecuados para esas consultas.**
  Las posibles consultas más frecuentes y por tanto, más útiles para tener índices y una consulta optimizada son:
  Datasets independientemente:
  - *Inspections:*
    1. Filtrado de las inspecciones según la fecha de realización.





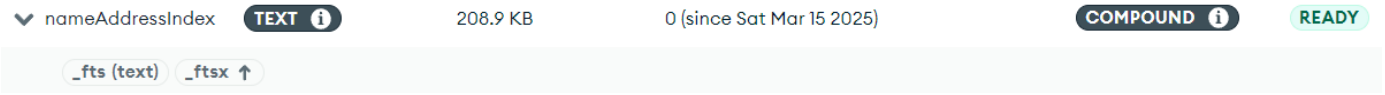
    2. Consultas de las inspecciones según el resultado de éstas y el sector al que pertenecen.



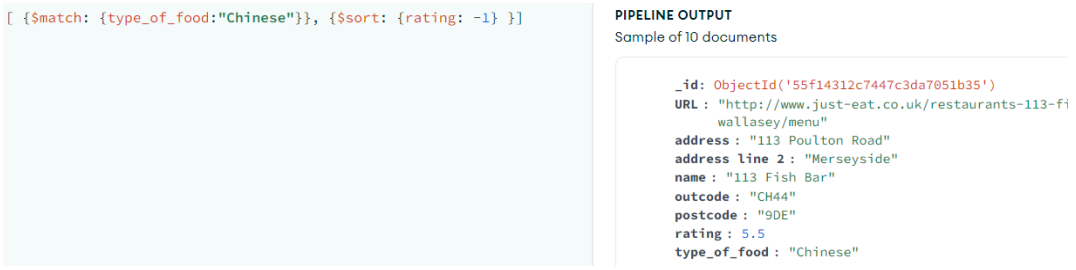


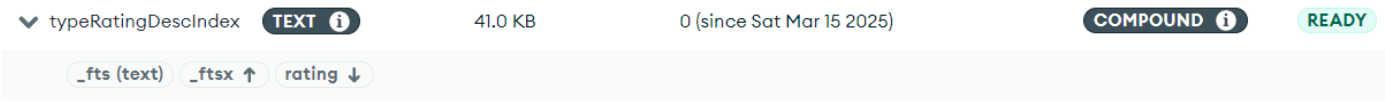
  - *Restaurants:*
    1. Consultas de los restaurantes según su nombre y la dirección principal en la que se encuentran.

Arnau Muñoz Barrera (**1665982**)
Biel Ramon Polo (**1600209**)
ATS - G13_10_30



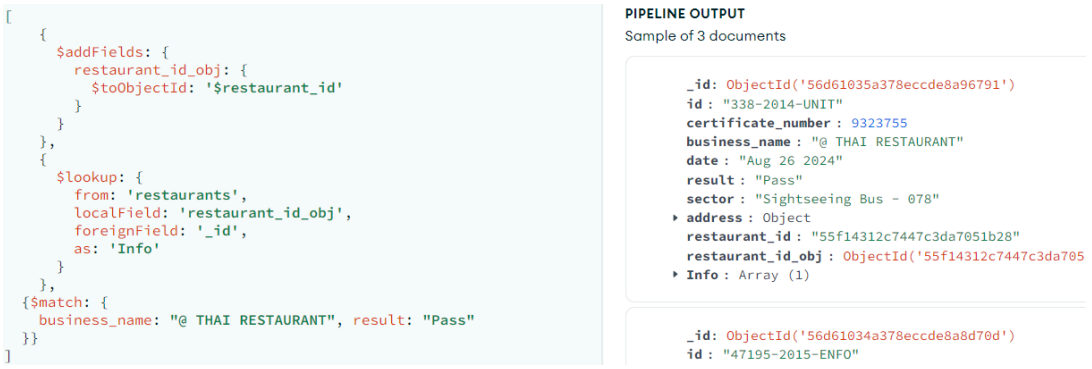
2. Consultas de los restaurantes según el tipo de comida y su puntuación asociada ordenada de mayor puntuación a menor (descendente).





Consultas conjuntas:

1. El resultado de las inspecciones realizadas a un restaurante según su nombre.



2. Fechas de inspecciones a restaurantes, filtrando las inspecciones con una fecha concreta y según el tipo de comida que ofrece el establecimiento.

Arnau Muñoz Barrera (**1665982**)
Biel Ramon Polo (**1600209**)
ATS - G13_10_30

```
    $addFields: {
      restaurant_id_obj: {
        $toObjectId: '$restaurant_id'
      }
    }
  },
  {
    $lookup: {
      from: 'restaurants',
      localField: 'restaurant_id_obj',
      foreignField: '_id',
      as: 'Info'
    }
  },
{$match: {
  date: "Dec 26 2023"
}},
{
      "$addFields": {
          "type_of_food": {
              "$arrayElemAt": ["$Info.type_of_food", 0]
          }
      }
  },
  {
      "$project": {
          "Info": 0
      }
  },
{$match: {type_of_food:"Chinese"} }
```

PIPELINE OUTPUT
Sample of 2 documents

```
_id: ObjectId('56d61033a378eccde8a8845a')
id : "920-2016-ENFO"
certificate_number : 50065961
business_name : ".CN CHINESE"
date : "Dec 26 2023"
result : "Fail"
sector : "Cigarette Retail Dealer - 127"
address : Object
restaurant_id : "55f14312c7447c3da7051b26"
restaurant_id_obj : ObjectId('55f14312c7447c3da7051
type_of_food : "Chinese"
```

```
_id: ObjectId('56d61033a378eccde8a87aec')
id : "55352-2015-ENFO"
certificate_number : 9285565
business_name : "BAMBOO GARDEN"
date : "Dec 26 2023"
result : "Violation Issued"
sector : "Misc Non-Food Retail - 817"
address : Object
restaurant_id : "55f14313c7447c3da7052195"
restaurant_id_obj : ObjectId('55f14313c7447c3da7052
type_of_food : "Chinese"
```

- **Comparar el rendimiento antes y después de crear los índices utilizando explain().**
  - *Inspections:*
    1. Con índice:

```
"executionStats": {
  "executionSuccess": true,
  "nReturned": 8,
  "executionTimeMillis": 0,
  "totalKeysExamined": 8,
  "totalDocsExamined": 8,
```

Sin índice:

```
"executionStats": {
  "executionSuccess": true,
  "nReturned": 8,
  "executionTimeMillis": 3,
  "totalKeysExamined": 0,
  "totalDocsExamined": 6370,
```

2. Con índice:


```
"executionStats": {
  "executionSuccess": true,
  "nReturned": 350,
  "executionTimeMillis": 0,
  "totalKeysExamined": 350,
  "totalDocsExamined": 350,
```


Sin índice:


```
"executionStats": {
  "executionSuccess": true,
  "nReturned": 350,
  "executionTimeMillis": 7,
  "totalKeysExamined": 0,
  "totalDocsExamined": 6370,
```


- *Restaurants:*

1. Con índice:


```
"executionStats": {
  "executionSuccess": true,
  "nReturned": 2,
  "executionTimeMillis": 0,
  "totalKeysExamined": 2,
  "totalDocsExamined": 2,
```


Sin índice:


```
"executionStats": {
  "executionSuccess": true,
  "nReturned": 2,
  "executionTimeMillis": 1,
  "totalKeysExamined": 0,
  "totalDocsExamined": 2548,
```

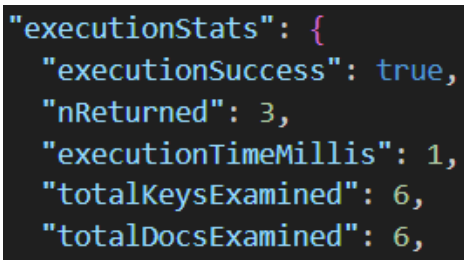

2. Con índice:


```
"executionStats": {
  "executionSuccess": true,
  "nReturned": 174,
  "executionTimeMillis": 0,
```

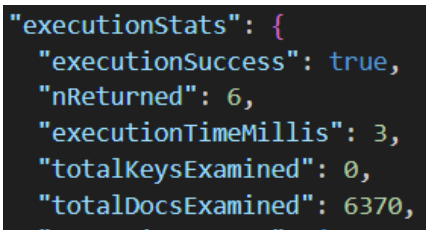

Sin índice:


```
"executionStats": {
  "executionSuccess": true,
  "nReturned": 174,
  "executionTimeMillis": 2,
  "totalKeysExamined": 0,
  "totalDocsExamined": 2548,
```

- *Conjuntas:*
    1. Con índice:

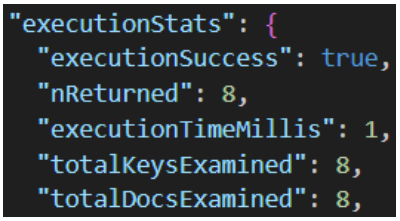


Sin índice:



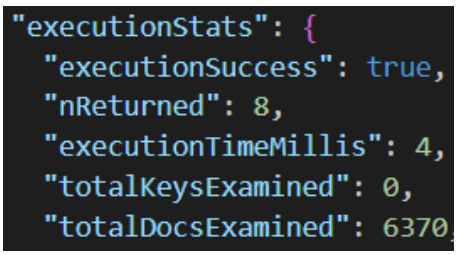
Para esta consulta, a pesar de que cada campo por separado tenga creado su índice por el cual filtrar la búsqueda de consultas, solo se hace uso de uno debido a que es la mejor opción en cuanto a optimización de los resultados.

2. Con índice:



Para esta consulta, a pesar de que cada campo por separado tenga creado su índice por el cual filtrar la búsqueda de consultas, solo se hace uso de uno debido a que es la mejor opción en cuanto a optimización de los resultados.

Sin índice:



Como podemos observar en la comparativa de las dos imágenes en cada caso, el tiempo de ejecución necesario para las dos consultas es diferente, en que haciendo uso de los índices, a pesar de ya ser un valor bajo, es aún mucho menor. Los índices nos permiten acceder al resultado de las consultas de una manera mucho más óptima y rápida.

2. Estrategias de escalabilidad

- **Proponer una estrategia de sharding adecuada para este dataset.**
  Para proponer una estrategia de sharding adecuada que ofrezca un gran rendimiento y escalabilidad se recomienda shardear por un campo con alta cardinalidad, es decir, un campo que contenga gran número de valores únicos y que estos estén distribuidos de manera uniforme. Proponemos:

```
sh.enableSharding("myDatabase");
sh.shardCollection("myDatabase.inspections", { date: 1 }); // Sharding por fecha de inspección
sh.shardCollection("myDatabase.restaurants", { postcode: "hashed" }); // Sharding por código postal del restaurante
```

  Para las inspecciones valoramos hacer agrupaciones según la fecha de realización de estas, mientras que para los restaurantes “agrupamos” según su valor en el capcom de código postal.

- **Diseñar un esquema de replicación para alta disponibilidad.**
  Nuestro esquema de replicación está formado por tres nodos secundarios donde, si alguno de los tres falla, los otros dos se propondrán como nodos primarios causando un empate las votaciones (cada nodo vota por sí mismo), haciendo que sea el árbitro el que tenga que resolver el conflicto aportando el voto clave:

```
rs.initiate({
    _id: "rs0",
    members: [
      { _id: 0, host: "mongo1:27017" },
      { _id: 1, host: "mongo2:27017" },
      { _id: 2, host: "mongo3:27017" },
      { _id: 3, host: "mongo4:27017", arbiterOnly: true }
    ]
});
```

- **Analizar posibles cuellos de botella y soluciones.**
  Los posibles *bottleneck* que pueden suceder son:

- Realización de consultas que requieren operaciones intensivas de lectura.
  Solución: implementar índices adecuados y balancear carga con sharding.
- Alta concurrencia de escritura en una única réplica primaria.
  Solución: usar sharding para distribuir carga de escritura y asegurar réplicas secundarias en diferentes regiones.
- Tiempos de latencia elevados en la replicación.
  Solución: configuración de garantías de éxito en las operaciones, tales como "write concern" y "read concern" adecuados según necesidades.