UNIVERSIDAD TECNOLÓGICA DE PUEBLA DIVISIÓN DE TECNOLOGÍAS DE LA INFORMACIÓN - INGENIERÍA EN DESARROLLO Y GESTIÓN DE SOFTWARE

MATERIA:

ADMINISTRACIÓN DE BASE DE DATOS

Producto 2

Plan de Administración de una Base de Datos No Relacional

DOCENTE:

JOSÉ FRANCISCO ESPINOSA GARITA

ALUMNOS:

ANDRADE REYES JUSTINO JUAN CARLOS - UTP0002150

DOMINGUEZ CASTILLO SALVADOR ESTEBAN – UTP0155077

8 ° A

FECHA: 10/03/2025

Índice del contenido

Índice de imágenes

Introducción

El presente reporte documenta el proceso de diseño, creación y configuración y administración de una base de datos no relacional destinada a centralizar y gestionar la información asociada con un sistema de tickets. Este sistema tiene como objetivo facilitar la organización, seguimiento y resolución de incidencias o solicitudes de clientes mediante la interacción con representantes y administradores. El proyecto se desarrolla como parte de un enfoque práctico en la materia de administración de bases de datos, con el fin de aplicar lo aprendido en la gestión estructurada de datos y la elaboración de un plan integral de administración de bases de datos relacional.

La base de datos fue implementada utilizando MongoDB, un sistema de gestión de bases de datos NoSQL orientado a documentos, ampliamente utilizado por su flexibilidad y escalabilidad. Se gestionó a través de la terminal, utilizando un contenedor Docker para su despliegue y administración, lo que permitió una configuración ágil y un entorno controlado para su ejecución. Durante el proceso de desarrollo, se adoptaron principios de modelado de datos en NoSQL, asegurando una estructura eficiente y escalable, optimizada para manejar grandes volúmenes de datos y permitir consultas ágiles sin la rigidez de los esquemas relacionales.

Con base en esta estructura, se ha diseñado un plan de administración de la base de datos no relacional que abarca prácticas de seguridad, optimización de consultas, respaldo de datos, control de acceso y mantenimiento preventivo. Este plan asegura el rendimiento constante de la base de datos, minimizando el riesgo de pérdida de información y asegurando la integridad y disponibilidad de los datos en todo momento.

La base de datos cuenta con una única colección denominada "tickets", la cual almacena información clave sobre las solicitudes gestionadas en el sistema. El diseño de la colección ha sido optimizado para garantizar un acceso eficiente a los datos, facilitando la generación de reportes y análisis detallados. Además, se ha estructurado de manera flexible para adaptarse a los distintos tipos de solicitudes y estados de los tickets. Aunque no se cuenta con un sistema de autenticación dentro de la base de datos, la gestión de permisos y acceso a la información se maneja a nivel de la aplicación.

Este reporte detalla las decisiones técnicas tomadas durante la creación del esquema, los procesos de carga de datos y la implementación del plan de administración. También se abordan los retos enfrentados durante el desarrollo, así como las soluciones aplicadas. El resultado es una base de datos confiable y funcional que respalda el funcionamiento del sistema de tickets, proporcionando una herramienta robusta para la gestión eficiente de incidencias y la mejora de la atención al cliente, todo ello dentro de un marco de administración adecuado y sustentable.

Contenido

1. Esquema de la base de datos no relacional

{

_id: ObjectId,

ID: INT,

FECHA: STRING,

SERIE: STRING,

ESTADO: STRING,

DEPARTAMENTO: STRING,

ASUNTO: STRING,

MENSAJE: STRING

SOLUCION: STRING,

CLIENTE: STRING,

REPRESENTANTE: STRING

}

2. Planeación de respaldos de la base de datos

Mes	Respaldo	Fecha	Hora	Responsable	Datos a Respaldar	Justificación
Febrero	1°	Lunes, 12 de febrero	02:00 AM	Administrador del sistema	Base de Datos Completa: Incluye las tablas Administrador, Cliente, Representante y Ticket.	Primer respaldo del mes, realizado temprano para evitar interferencias con la carga de trabajo del sistema. Respaldo completo.
	2°	Lunes, 26 de febrero	02:00 AM	Administrador del sistema	Base de Datos Completa	Segundo respaldo del mes, realizado fuera de las horas de operación para evitar impacto en el sistema. Respaldo completo.
Marzo	1°	Lunes, 5 de marzo	02:00 AM	Administrador del sistema	Base de Datos Completa	Primer respaldo mensual después de verificar actividades recientes. Asegura que todos los cambios estén respaldados.

	2°	Lunes, 19 de marzo	02:00 AM	Administrador del sistema	Base de Datos Completa	Segundo respaldo del mes para mantener la seguridad y disponibilidad. Captura cambios recientes.
Abril	1°	Lunes, 2 de abril	02:00 AM	Administrador del sistema	Base de Datos Completa	Respaldo inicial del mes para asegurar que cualquier cambio desde el mes anterior esté respaldado.
	2°	Lunes, 16 de abril	02:00 AM	Administrador del sistema	Base de Datos Completa	Segundo respaldo de abril, enfocado en las semanas previas al cambio de mes. Asegura la integridad y disponibilidad.

2.1 Procedimiento de copias de seguridad

mongodump --db ticket --out /tmp/mongobackups/febrero_$(date +"%m-%d-%y")

mongodump --db ticket --out /tmp/mongobackups/marzo_$(date +"%m-%d-%y")

mongodump --db ticket --out /tmp/mongobackups/abril_$(date +"%m-%d-%y")

```
mongodb@f06fe198e33a:/tmp/mongobackups$ mongodump --db ticket --out /tmp/mongobackups/marzo_$(date +"%m-%d-%y")
2025-03-10T20:09:49.639+0000    writing ticket.tickets to /tmp/mongobackups/marzo_03-10-25/ticket/tickets.bson
2025-03-10T20:09:49.663+0000    done dumping ticket.tickets (1000 documents)
mongodb@f06fe198e33a:/tmp/mongobackups$ mongodump --db ticket --out /tmp/mongobackups/abril_$(date +"%m-%d-%y")
2025-03-10T20:09:58.826+0000    writing ticket.tickets to /tmp/mongobackups/abril_03-10-25/ticket/tickets.bson
2025-03-10T20:09:58.848+0000    done dumping ticket.tickets (1000 documents)
mongodb@f06fe198e33a:/tmp/mongobackups$ ls
abril_03-10-25  febrero_03-10-25  marzo_03-10-25
```

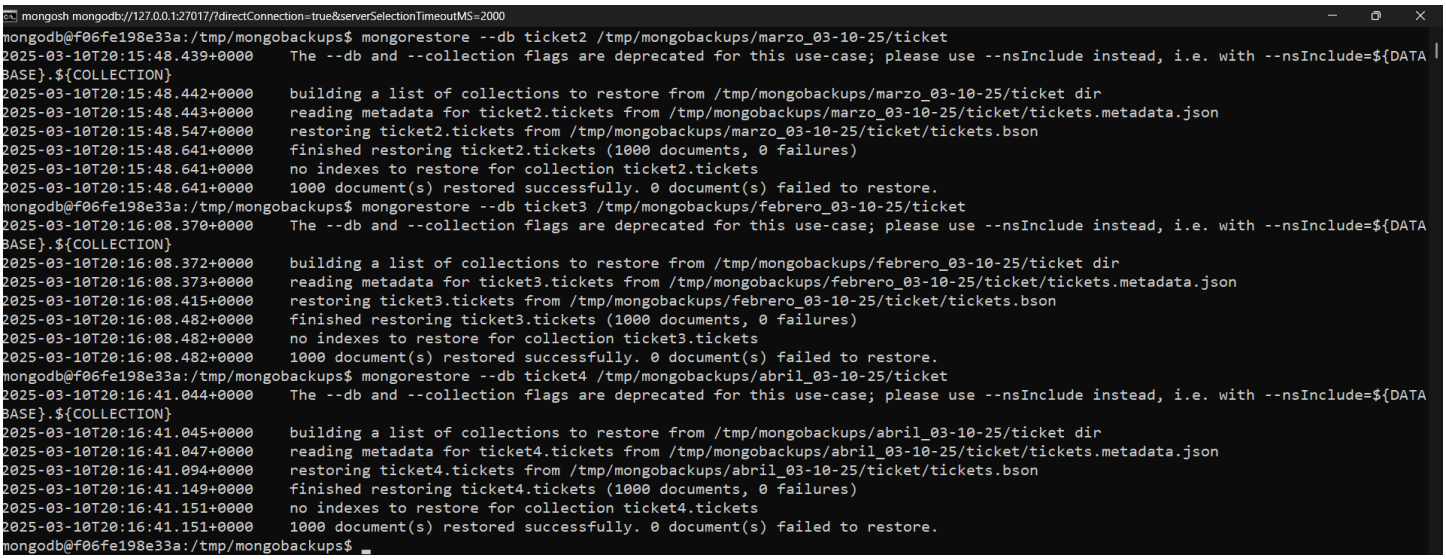
1. Img: Respaldo de datos de mongo en Docker para febrero, marzo y abril

2.2 Protocolos de restauración

mongorestore --db ticket2 /tmp/mongobackups/ferbrero_03-10-25/ticket

mongorestore --db ticket3 /tmp/mongobackups/marzo_03-10-25/ticket

mongorestore --db ticket4 /tmp/mongobackups/abril_03-10-25/ticket



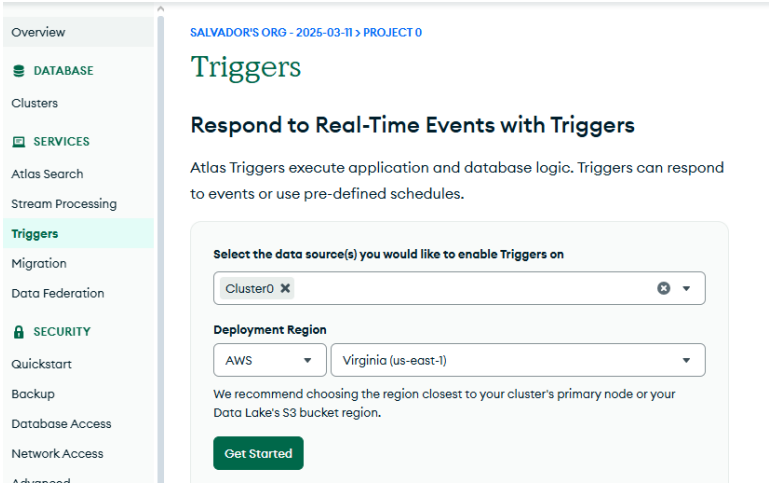
2. Img: Restauración de respaldos de mongo en Docker

3. Automatización de tareas

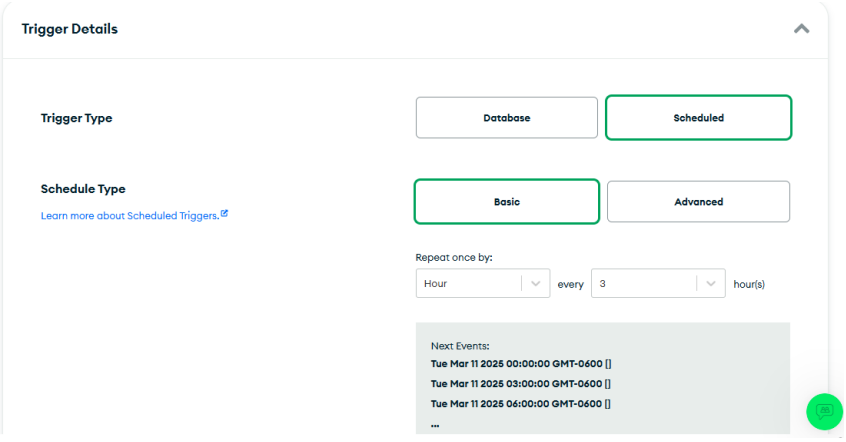
Para este proceso de automatización de tareas usaremos mongo atlas,a través de mongo Shell

```
PS C:\Users\sedcc> mongosh --version
2.3.9
PS C:\Users\sedcc> mongosh "mongodb+srv://cluster0.np21i.mongodb.net/" --apiVersion 1 --username sedccastillaa
Enter password: ****************
Current Mongosh Log ID: 67cfc1f1292121b2974d7941
Connecting to:          mongodb+srv://<credentials>@cluster0.np21i.mongodb.net/?appName=mongosh+2.3.9
Using MongoDB:          8.0.5 (API Version 1)
Using Mongosh:          2.3.9
mongosh 2.4.2 is available for download: https://www.mongodb.com/try/download/shell

For mongosh info see: https://www.mongodb.com/docs/mongodb-shell/

Atlas atlas-g7q2ns-shard-0 [primary] test>
```

Configurar Triggers en MongoDB Atlas

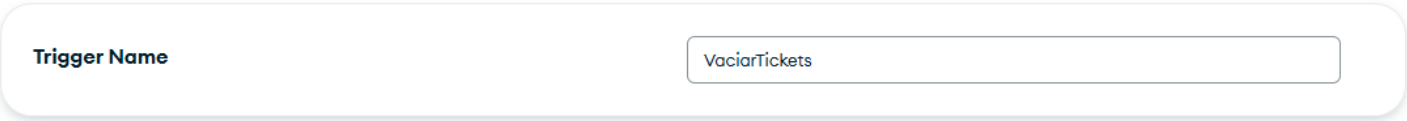


Después lo configuramos como tipo scheduled



Ponemos la función

- Se implementaron dos automatizaciones en MongoDB Atlas usando Triggers:
    - Un Scheduled Trigger que vacía la colección **tickets** cada día.
    - Un Database Trigger que muestra una alerta cuando un ticket tiene prioridad alta.



- Se configuró MongoDB Atlas y se creó un cluster.
- Se activaron los Triggers en la sección "Triggers" de Atlas.
- Se usaron los siguientes scripts en "Function":
- Vaciar la colección automáticamente:

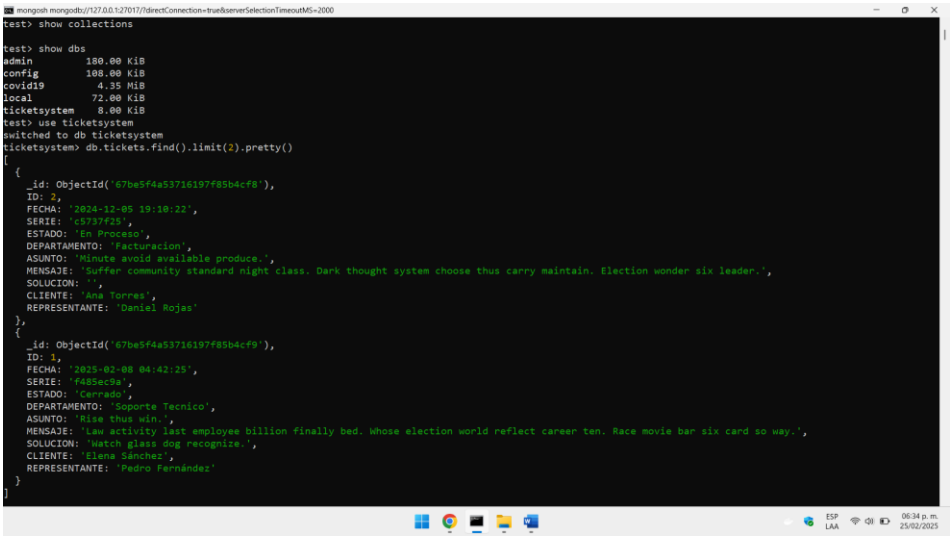
4. Métodos de importación y exportación de datos

4.1 Importación

mongoimport --db ticketsystem --collection tickets --type csv --headerline --file /tmp/ticketsdb.csv

```
mongodb@0fcc77a39573:/$ mongoimport --db ticketsystem --collection tickets --type csv --headerline --file /tmp/ticketsdb.csv
2025-02-26T00:24:42.312+0000    connected to: mongodb://localhost/
2025-02-26T00:24:42.550+0000    1000 document(s) imported successfully. 0 document(s) failed to import.
```

3. Img: Carga de csv a mongo db desde Docker



4. Img: Comprobando que se insertó con éxito la bd y se visualizan de manera correcta los datos

4.2 Exportación

mongoexport --db ticket --collection tickets --out /tmp/mongobackups/tickets.json

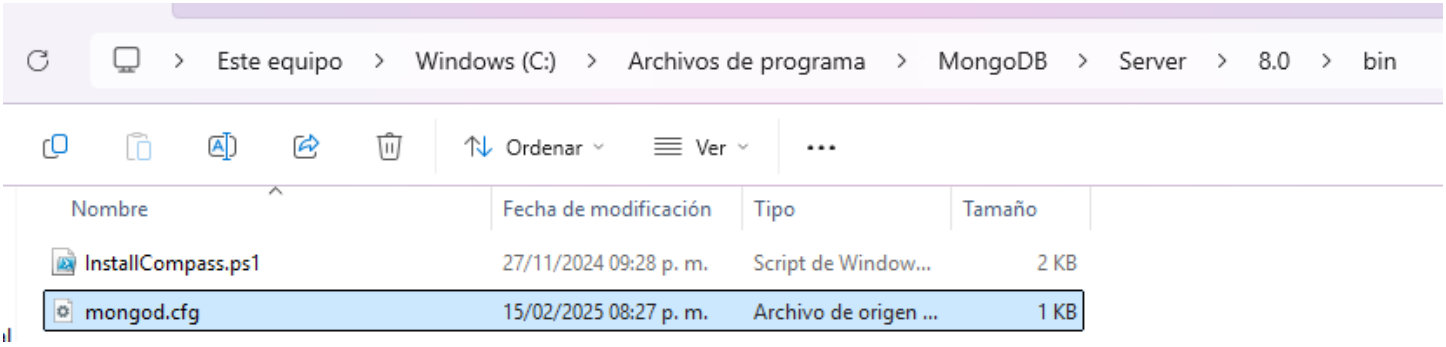
```
mongodb@f06fe198e33a:/tmp/mongobackups$ mongoexport --db ticket --collection tickets --out /tmp/mongobackups/tickets.json
2025-03-10T20:22:58.409+0000    connected to: mongodb://localhost/
2025-03-10T20:22:58.575+0000    exported 1000 records
mongodb@f06fe198e33a:/tmp/mongobackups$
```

5. Img: Exportando bd

5. Lineamientos de seguridad en la base de datos no relacional

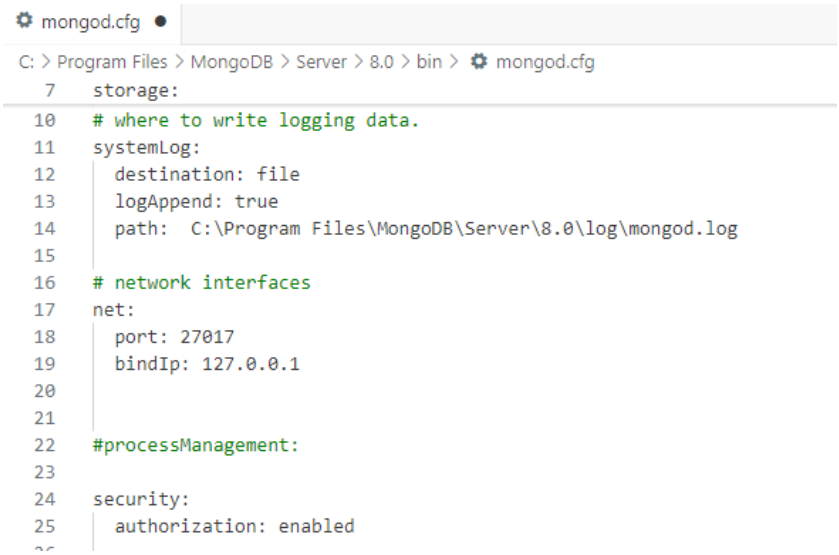
5.1 Configuración

Ubicación del Archivo de Configuración:

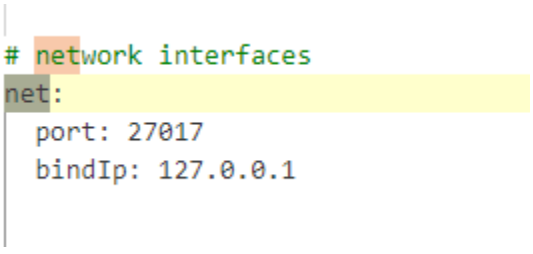


Windows: Crear en C:\Program Files\MongoDB\Server\<version>\bin\mongod.conf

Habilitar Autenticación:



Editamos el archivo y configuramos security



En net configuramos que MongoDB escuche solo en direcciones IP de confianza

Otra buena recomendación de seguridad es crear usuarios Administrativos

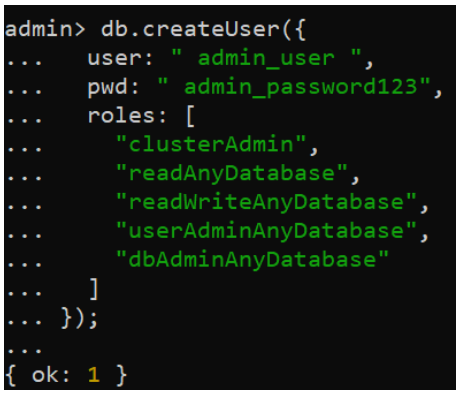
Crear Usuarios Administrativos:

```
test> use admin
switched to db admin
admin> db.createUser({
...   user: "admin_user",
...   pwd: "admin_password123",
...   roles: [
...     { role: "userAdminAnyDatabase", db: "admin" },
...     { role: "dbAdminAnyDatabase", db: "admin" },
...     { role: "readWriteAnyDatabase", db: "admin" }
...   ]
... })
{ ok: 1 }
admin>
```

5.2 Creación de usuarios

Usuario	Nombre de Usuario	Contraseña	Base de Datos	Objetos que Puede Manipular	Permisos
admin_user	admin_user	admin_password123	ticketsystem	Tablas: ticket, representante, cliente, administrador	SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, ALTER, GRANT en las tablas mencionadas. Acceso total.
support_user	support_user	support_password123	ticketsystem	Tablas: ticket, representante, cliente	SELECT, INSERT, UPDATE, DELETE solo en la tabla ticket. No tiene acceso a la tabla administrador ni a objetos.
readonly_user	readonly_user	readonly_password123	ticketsystem	Tablas: ticket, representante, cliente, administrador	Solo SELECT en todas las tablas. No puede modificar ni eliminar datos.

```
db.createUser({
  user: " admin_user ",
  pwd: " admin_password123",
  roles: [
   "clusterAdmin",
   "readAnyDatabase",
   "readWriteAnyDatabase",
   "userAdminAnyDatabase",
   "dbAdminAnyDatabase"
  ]
});
```

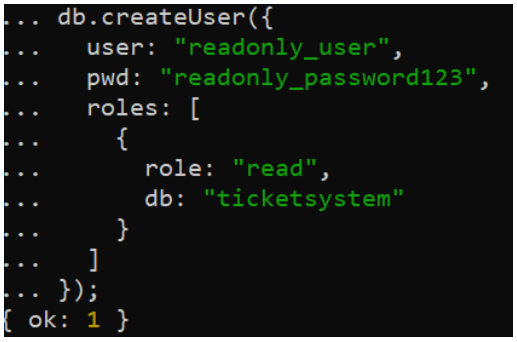


5. Img Creación de usuario 1

```
db.createUser({
  user: "support_user",
  pwd: "support_password123",
  roles: [
    {
      role: "readWrite",
      db: "ticketsystem"
    }
  ]
});
```

6. Img Creación de usuario 2

```
db.createUser({
  user: "readonly_user",
  pwd: "readonly_password123",
  roles: [
    {
      role: "read",
      db: "ticketsystem"
    }
  ]
});
```



7. Img Creación de usuario 3

5.3 Comprobación de acceso

Verificar y validar los mecanismos de autenticación y autorización en el sistema de base de datos MongoDB.

Paso 1: Conexión inicial

```
PS C:\Users\D5211> mongosh
Current Mongosh Log ID: 67cf7e943bbb0e2038cb0ce1
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2
.3.8
Using MongoDB:          8.0.4
Using Mongosh:          2.3.8
mongosh 2.4.2 is available for download: https://www.mongodb.com/try/download/shell

For mongosh info see: https://www.mongodb.com/docs/mongodb-shell/

------
```

Paso 2: Verificar usuarios existentes

```
test> use admin
switched to db admin
admin>
```

***use admin*:** Cambia al contexto de la base de datos administrativa, donde se almacenan los usuarios y roles.

```
admin> show users
[
  {
    _id: 'admin.adminUser',
    userId: UUID('2e4835b9-9d51-450c-ae7c-92bda7d06c2e'),
    user: 'adminUser',
    db: 'admin',
    roles: [
      { role: 'userAdminAnyDatabase', db: 'admin' },
      { role: 'readWriteAnyDatabase', db: 'admin' }
    ],
    mechanisms: [ 'SCRAM-SHA-1', 'SCRAM-SHA-256' ]
  }
]
admin>
```

show users: Lista los usuarios configurados en el sistema.

```
admin> db.system.users.find()
[
  {
    _id: 'admin.adminUser',
    userId: UUID('2e4835b9-9d51-450c-ae7c-92bda7d06c2e'),
    user: 'adminUser',
    db: 'admin',
    credentials: {
      'SCRAM-SHA-1': {
        iterationCount: 10000,
        salt: 'cnhn2aMJhHZc7Ae5TCt2Yg==',
        storedKey: 'wj0zwBKKQlnVQtWtSaxizM5ygck=',
        serverKey: '0lO0KjtxiYpLWCSS7lTgJkSaDl4='
      },
      'SCRAM-SHA-256': {
        iterationCount: 15000,
        salt: 'AgPom4Wp8boWepiR5aHyVaNkkdjxVF/NSNG6Vg==',
        storedKey: 'NwMDrb2KNevsZ/DAGm6lPsn5CbcB/UdnpfRNL7LoAVA=',
        serverKey: 'li9OSMaP9poSwhGnwrmFcEh9VZ9ys3LSNVBS0d1Loog='
      }
    },
    roles: [
      { role: 'userAdminAnyDatabase', db: 'admin' },
      { role: 'readWriteAnyDatabase', db: 'admin' }
    ]
  },
  {
    _id: 'ticket.ticketUser',
    userId: UUID('0a1dc119-0142-4c14-97f2-48a79bd78ae6'),
    user: 'ticketUser',
    db: 'ticket',
    credentials: {
      'SCRAM-SHA-1': {
        iterationCount: 10000,
        salt: 'MFszI2PgxaWGSqq2TcQoSQ==',
        storedKey: 'KzM09LXw2www272pX+o5SaZg1E4=',
        serverKey: '4enF1V1qlLC61eY/H8GEuY1SUIM='
      },
      'SCRAM-SHA-256': {
        iterationCount: 15000,
        salt: 'Nu0dAsyYpf9pPefz1y9qrSlJVaAUHG2l2e/A6Q==',
        storedKey: 'yCGtHkkE+h3u8We7csPNAaiF0jjC6BnGUshPVflZqJU=',
        serverKey: 'jn2616TDJHAoLv84dEEyM8cG4wdFXNp+9y0C7RIXpD8='
      }
    },
    roles: [ { role: 'readWrite', db: 'ticket' } ]
  }
]
admin>
```

db.system.users.find(): Muestra información detallada de los usuarios, incluyendo roles y bases de datos asociadas.

Paso 3: Autenticación de usuario

```
PS C:\Users\D5211> mongosh -u adminUser -p SecurePassword123! --authenticationDatabase admin
Current Mongosh Log ID: 67cf82c935918b6da6cb0ce1
Connecting to:          mongodb://<credentials>@127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&authSource=admin&appName=mongosh+2.3.8
Using MongoDB:          8.0.4
Using Mongosh:          2.3.8
mongosh 2.4.2 is available for download: https://www.mongodb.com/try/download/shell

For mongosh info see: https://www.mongodb.com/docs/mongodb-shell/

------
   The server generated these startup warnings when booting
   2025-03-10T17:35:49.149-06:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
   2025-03-10T17:35:49.149-06:00: This server is bound to localhost. Remote systems will be unable to connect to this server. Start the server with --bind_ip <
erve responses from, or with --bind_ip_all to bind to all interfaces. If this behavior is desired, start the server with --bind_ip 127.0.0.1 to disable this wa
------

test>
```

Paso 4: Verificar permisos

```
test> use ticket
switched to db ticket
ticket> db.tickets.find().limit(5)
[
  {
    _id: ObjectId('67cf7aa64375a30f492c4870'),
    ID: 3,
    FECHA: '2024-07-21 10:57:19',
    SERIE: '1b35fa25',
    ESTADO: 'Cerrado',
    DEPARTAMENTO: 'Facturacion',
    ASUNTO: 'Another best reality big.',
    MENSAJE: 'Still despite project improve. Foreign decide everything question bad. Section know impact under.',
    SOLUCION: 'Information Mr financial clear including poor quality democratic.',
    CLIENTE: 'Luis Ramírez',
    REPRESENTANTE: 'Daniel Rojas'
  },
  {
    _id: ObjectId('67cf7aa64375a30f492c4871'),
    ID: 7,
    FECHA: '2025-02-09 15:40:15',
    SERIE: 'e137383c',
    ESTADO: 'Abierto',
    DEPARTAMENTO: 'Facturacion',
    ASUNTO: 'Yeah various take decade history.',
    MENSAJE: 'Budget write main system. Responsibility back policy. Perform take our act.',
    SOLUCION: '',
    CLIENTE: 'Elena Sánchez',
    REPRESENTANTE: 'Daniel Rojas'
  },
  {
    _id: ObjectId('67cf7aa64375a30f492c4872'),
    ID: 13,
    FECHA: '2025-01-17 22:09:45',
    SERIE: '6ece2c47',
    ESTADO: 'Cerrado',
    DEPARTAMENTO: 'Soporte Tecnico',
    ASUNTO: 'Parent vote spring house mind while.',
    MENSAJE: 'Guy his give base production.',
    SOLUCION: 'Blood change process idea firm thank save certainly. Capital simply reflect usually month.',
    CLIENTE: 'Elena Sánchez',
    REPRESENTANTE: 'Pedro Fernández'
```

Explicación:

- show dbs: Muestra las bases de datos a las que el usuario tiene acceso.
- use ticket: Cambia al contexto de la base de datos de tickets.
- db.tickets.find().limit(5): Intenta recuperar los primeros 5 documentos, verificando permisos de lectura.

5.4. Revocación de permisos de un usuario

Restringir o eliminar permisos específicos de un usuario para controlar el acceso a recursos de la base de datos.

Paso 1: Identificar permisos actuales

```
ticket> db.getUser("ticketUser")
{
  _id: 'ticket.ticketUser',
  userId: UUID('a146ee68-2dc2-4fbd-9125-ea65253095b0'),
  user: 'ticketUser',
  db: 'ticket',
  roles: [ { role: 'readWrite', db: 'ticket' } ],
  mechanisms: [ 'SCRAM-SHA-1', 'SCRAM-SHA-256' ]
}
ticket>
```

Comando db.getUser() permite inspeccionar los roles actuales

Explicación:

- Permite visualizar los roles y privilegios actuales del usuario.
- Ayuda a comprender el alcance de los permisos antes de la revocación.
- Fundamental para realizar cambios precisos en la configuración de seguridad.

Paso 2: Revocar roles específicos

```
ticket> db.revokeRolesFromUser(
...   "ticketUser",
...   [{ role: "readWrite", db: "ticket" }]
... )
{ ok: 1 }
ticket>
```

```
db.updateUser(
  "ticketUser",
  {
    roles: [] # Eliminar todos los roles
  }
)
```

Paso 3: Verificamos los cambios

```
ticket> db.getUser("ticketUser")
{
  _id: 'ticket.ticketUser',
  userId: UUID('a146ee68-2dc2-4fbd-9125-ea65253095b0'),
  user: 'ticketUser',
  db: 'ticket',
  roles: [],
  mechanisms: [ 'SCRAM-SHA-1', 'SCRAM-SHA-256' ]
}
ticket>
```

Podemos verificar que fue exitosa la revocación de permisos en la columna roles.

Tabla de Control de Cambios:

Componente	Descripción	Antes	Después	Impacto
Usuario	ticketUser	Activo	Activo	Sin cambios
Roles	readWrite en ticket	Presente	Removido	Restringido
Acceso	Lectura/Escritura	Habilitado	Bloqueado	Controlado

5.5. Borrado de 1 de los usuarios

Eliminar completamente un usuario del sistema de base de datos MongoDB, revocando todos sus accesos y permisos, como parte de un proceso de gestión de seguridad y control de acceso.

Paso 1: Verificación de Usuarios Existentes

```
test> use admin
switched to db admin
admin> show users
[
  {
    _id: 'admin.adminUser',
    userId: UUID('9ddbc031-60da-42ff-9a5a-41d14a25a2d9'),
    user: 'adminUser',
    db: 'admin',
    roles: [
      { role: 'readWriteAnyDatabase', db: 'admin' },
      { role: 'userAdminAnyDatabase', db: 'admin' }
    ],
    mechanisms: [ 'SCRAM-SHA-1', 'SCRAM-SHA-256' ]
  }
]
```

Usamos show users y después db.system.users.find() para ver la lista completa

Paso 2: Eliminar Usuario

```
  },
  {
    _id: 'ticket.ticketUser',
    userId: UUID('a146ee68-2dc2-4fbd-9125-ea65253095b0'),
    user: 'ticketUser',
    db: 'ticket',
    credentials: {
      'SCRAM-SHA-1': {
        iterationCount: 10000,
        salt: 'FOj64yq4lllYbfSi6pQ+DA==',
        storedKey: 'x1FLXEsWFasxFPqHGL5kRYuUT40=',
        serverKey: 'QB4HgiTDCOH+WZ332wkUL8Gs2z8='
      },
      'SCRAM-SHA-256': {
        iterationCount: 15000,
        salt: 'qLmvPRjiL7yHzOE7p4xRHRDUKtZYFU/PKm96SQ==',
        storedKey: 'nsHj5MY0VFtQf1ZibIn8FSKjAUb0RcOyag55miTZA24=',
        serverKey: 'ctYgsMNPN8Zr4NzhOQObLW8gYW4ZW9nGkvINVnzg1Ns='
      }
    },
    roles: []
  }
]
admin> db.dropUser("ticketUser")
```

En este caso eliminamos “ticket.ticketUser” con db.dropUser("ticketUser")

```
ticket> db.dropUser("ticketUser")
{ ok: 1 }
ticket>
```

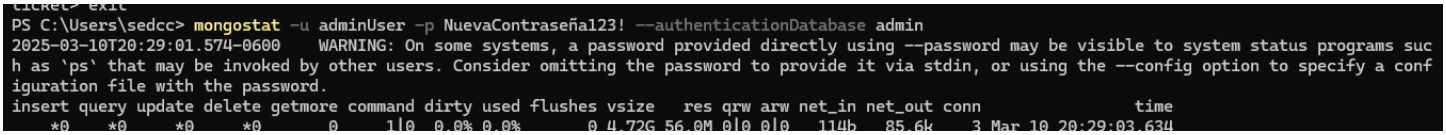
Podemos ver que el usuario fue eliminado y ya no esta disponible

```
ticket> show users
[]
ticket> db.system.users.find()

ticket>
```

6. Reporte del rendimiento de una base de datos relacional con herramienta de monitoreo

Realizar un análisis comprehensive del rendimiento de la base de datos MongoDB, utilizando herramientas de monitoreo para evaluar métricas críticas de desempeño, identificar cuellos de botella y optimizar el rendimiento del sistema.

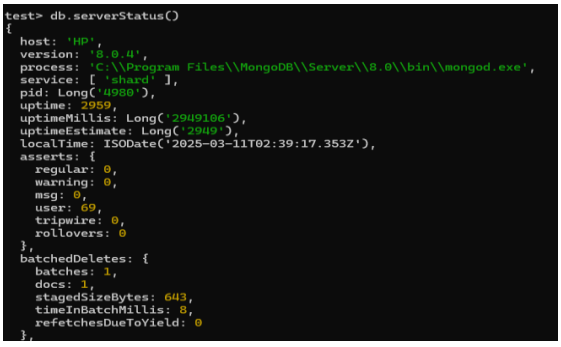
1.Usamos Mongostat- Metricas en tiempo real



Explicación de las columnas más importantes:

- **insert** → Número de inserciones por segundo.
- **query** → Consultas ejecutadas por segundo.
- **update** → Actualizaciones por segundo.
- **delete** → Eliminaciones por segundo.
- **getmore** → Solicitudes de más documentos en un cursor.
- **command** → Comandos ejecutados por segundo.
- **resident** → Memoria RAM utilizada.
- **virtual** → Memoria virtual usada.

2.Monitoreo Detallado con db.serverStatus()



Esto arroja un JSON con información detallada sobre:

- Opciones del servidor
- Manejo de conexiones
- Carga de CPU
- Uso de memoria
- Tiempo de respuesta

3.Puntos Claves de db.serverStatus()

1. Estado de las Conexiones

¿Por qué es importante? Muestra cuántas conexiones hay activas y disponibles, lo que indica si el servidor está saturado.

```
test> db.serverStatus().connections
{
  current: 5,
  available: 999995,
  totalCreated: 46,
  rejected: 0,
  active: 2,
  threaded: 5,
  exhaustIsMaster: Long('0'),
  exhaustHello: Long('1'),
  awaitingTopologyChanges: Long('1'),
  loadBalanced: Long('0')
}
test>
```

-Si current es muy alto en relación con available, la base podría estar sobrecargada.

2. Operaciones en la Base de Datos

¿Por qué es importante? Indica cuántas operaciones de lectura, escritura y eliminación se han realizado.

```
test> db.serverStatus().opcounters
{
  insert: Long('1006'),
  query: Long('22'),
  update: Long('33'),
  delete: Long('18'),
  getmore: Long('0'),
  command: Long('948')
}
test>
```

-Un número alto de query con un insert bajo puede indicar una base de datos más enfocada en consultas.

-Si delete es alto, tal vez haya muchos cambios o eliminación de datos innecesarios.

3. Uso de Memoria

¿Por qué es importante? Ayuda a detectar si la base de datos está consumiendo demasiada RAM.

```
test> db.serverStatus().mem
{
  bits: 64,
  resident: 42,
  virtual: 4836,
  supported: true,
  secureAllocByteCount: 0,
  secureAllocBytesInPages: 4096
}
test>
```

-Si resident es muy alto, puede haber problemas de rendimiento.

-Si virtual es mucho mayor que resident, significa que el sistema está usando "memoria swap", lo que lo hace más lento.

4. Uso de Índices

¿Por qué es importante? Si los índices son grandes, la base de datos puede volverse más lenta.

```
test> db.serverStatus().indexCounters

test> "indexCounters": {
...   "hits": 12000,
...   "misses": 50,
...   "resets": 0
... }
...
```

-Si misses es alto, significa que muchas consultas no están usando índices y eso hace que la base sea lenta.

5. Rendimiento de Escritura y Lectura

¿Por qué es importante? Indica cuántos datos se han leído y escrito.

```
test> db.serverStatus().metrics.document
{
  deleted: Long('9'),
  inserted: Long('1002'),
  returned: Long('7'),
  updated: Long('22')
}
test>
```

-Si inserted es mucho mayor que returned, se están escribiendo más datos de los que se leen.

Conclusiones

El plan de administración de la base de datos no relacional garantiza que los datos sean seguros, accesibles y eficientes. Para ello, se implementó un procedimiento de copias de seguridad que permite recuperar la información en caso de fallas o pérdidas. Además, los protocolos de restauración aseguran que el sistema pueda volver a su estado anterior de manera rápida y efectiva.

Se incluyó la automatización de tareas para reducir errores humanos y mejorar la eficiencia operativa, permitiendo programar respaldos, mantenimiento y limpieza de datos sin intervención manual. También se definieron métodos de importación y exportación que facilitan la migración de información entre sistemas sin afectar la integridad de los datos.

En cuanto a la seguridad, se establecieron lineamientos como el control de accesos, encriptación de datos y auditorías para prevenir accesos no autorizados y proteger la información almacenada.

Finalmente, el análisis del rendimiento de la base de datos con la herramienta de monitoreo permitió identificar posibles cuellos de botella, optimizar el uso de recursos y garantizar que las consultas se ejecuten de manera eficiente. La información obtenida sobre conexiones, uso de memoria, operaciones realizadas e índices ha sido clave para mejorar el desempeño del sistema.

Este plan proporciona una gestión estructurada de la base de datos no relacional, asegurando estabilidad, seguridad y un alto rendimiento para las operaciones del negocio.

Bibliografía

Administration. (s/f). Mongodb.com. Recuperado el 11 de marzo de 2025, de https://www.mongodb.com/docs/manual/administration/

Automatización de Datos: Beneficios, Uso y Softwares. (2024, enero 9). Klippa. https://www.klippa.com/es/blog/informativo/automatizacion-datos/

Seguridad de la base de datos de NoSQL. (s/f). Thales Cloud Security Products. Recuperado el 11 de marzo de 2025, de https://cpl.thalesgroup.com/es/encryption/database-security/nosql-encryption