



INFORME DE TALLER DE SHARDING

I. PORTADA

Tema:	Taller de sharding
Unidad de Organización Curricular:	PROFESIONAL
Nivel y Paralelo:	Quinto "A"
Alumnos participantes:	Aldas Jordan Wellington Ismael Caguasango Bayas Alex Patricio Gómez Llerena Luis Fernando Paredes Barrera Luis Enrique
Asignatura:	Sistemas de Base de Datos Distribuidas
Docente:	Ing. Jose Ruben Caiza Caizabuano, Mg.

II. INFORME DE TALLER DE SHARDING

2.1 Objetivos

2.1.1 Objetivo General:

Implementar un entorno de bases de datos distribuidas utilizando MongoDB y PostgreSQL, configurando la fragmentación, replicación y validaciones mediante triggers para garantizar la integridad y disponibilidad de la información.

2.1.2 Objetivos Específicos:

- Instalar y configurar MongoDB en entornos Windows y Linux, estableciendo los directorios necesarios para la fragmentación y replicación de datos.
- Diseñar y crear la base de datos "EmpresaViajes" en PostgreSQL con sus respectivas tablas relacionales, asegurando la correcta vinculación mediante claves foráneas.
- Implementar triggers en PostgreSQL que automaticen la verificación de licencias, la disponibilidad de vehículos y el cálculo del costo de viaje, fortaleciendo la integridad y consistencia de los datos.

2.2 Modalidad

Presencial

2.3 Tiempo de duración

Presenciales: 3

No presenciales: 0

2.4 Instrucciones

Descargar e instalar MongoDB en Windows usando el paquete MSI y configuren la opción de ejecutarlo como servicio. Luego, agreguen la carpeta bin de MongoDB al Path para que puedan usar el shell desde cualquier ubicación.



A continuación, descarguen e instalen el MongoDB Shell, agregando también su carpeta al Path. Después, creen los directorios shard1, shard2 y config para el sharding y verifiquen que se hayan generado correctamente.

Seguidamente, abran el Mongo Shell y prueben comandos básicos para confirmar que MongoDB funciona correctamente antes de configurar sharding y replicación. En Linux, instalen MongoDB con `sudo apt install mongodb-server-core` y creen los directorios equivalentes, verificando su existencia. Finalmente, en PostgreSQL, creen la base de datos EmpresaViajes y sus tablas, inserten datos de prueba y configuren los triggers para validar licencias, disponibilidad de vehículos y calcular el costo de los viajes.

Finalmente validar cada trigger para asegurarse de que funcionen correctamente y que los datos mantengan su integridad.

2.5 Listado de equipos, materiales y recursos

Listado de equipos y materiales generales empleados en la guía práctica:

- Máquina Virtual con Ubuntu
- MongoDB
- VirtualBox
- PostgreSQL

TAC (Tecnologías para el Aprendizaje y Conocimiento) empleados en la guía práctica:

- ☒ Plataformas educativas
- ☒ Simuladores y laboratorios virtuales
- ☐ Aplicaciones educativas
- ☒ Recursos audiovisuales
- ☐ Gamificación
- ☐ Inteligencia Artificial

Otros (Especifique): _____

2.6 Actividades desarrolladas

INSTALACIÓN DE MONGODB EN WINDOWS CON LA ACTIVACION DE VARIALES DE ENTORNO

Lo indispensable es descargar mongodb en su sitio oficial asegurando su confiabilidad:
<https://www.mongodb.com/try/download/community>

Dentro de la página se descargará el programa en si, es decir la aplicación:



UNIVERSIDAD TÉCNICA DE AMBATO
FACULTAD DE INGENIERÍA EN SISTEMAS ELECTRÓNICA E INDUSTRIAL
CARRERA DE Elige un elemento.
CICLO ACADÉMICO: AGOSTO 2025 - ENERO 2026



VersionVersión

8.0.6 (current)8.0.6 (actual)

▼

PlatformPlataforma

Windows x64Ventanas x64

▼

PackagePaquete

msimsi

▼

Download Descargar

Copy

linkCopiar enlace

More

Options

Más

opciones

...

Ilustración 1 Descarga de la aplicación de MongoDB en Windows

Posterior a esa descarga dentro de la misma página buscaremos el Shell para descargar el repositorio:

Versión

2.5.0

▼

Plataforma

Windows x64 (10+)

▼

Paquete

cremallera

▼

Descargar

Copiar enlace

Más opciones



PROCESO DE INSTALACION DE MONGODB EN WINDOWS

Ejecutar la descarga es decir el setup de instalación:

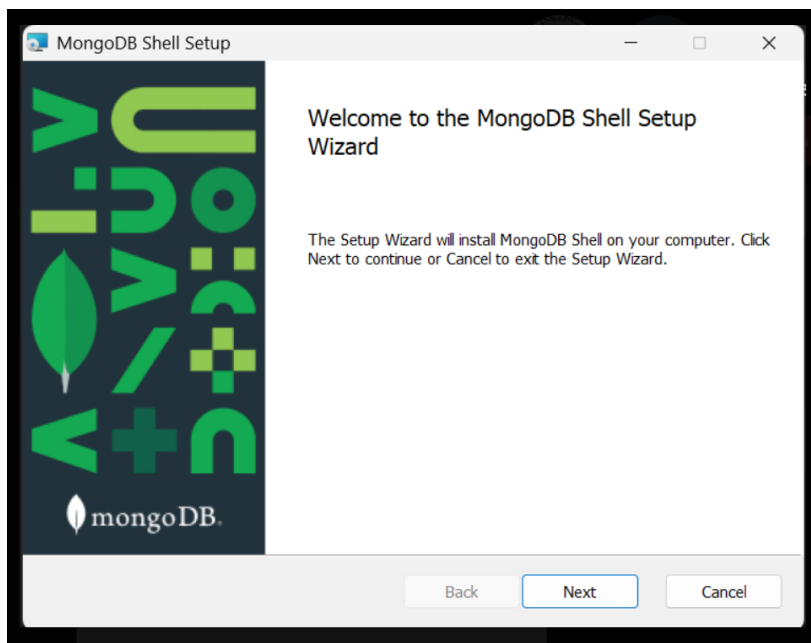


Ilustración 2 Setup de instalación (MongoDB)

Seleccionar el lugar en donde se realizará la descarga de los archivos y complementos necesarios para la ejecución y funcionamiento de Mongo dentro del equipo:

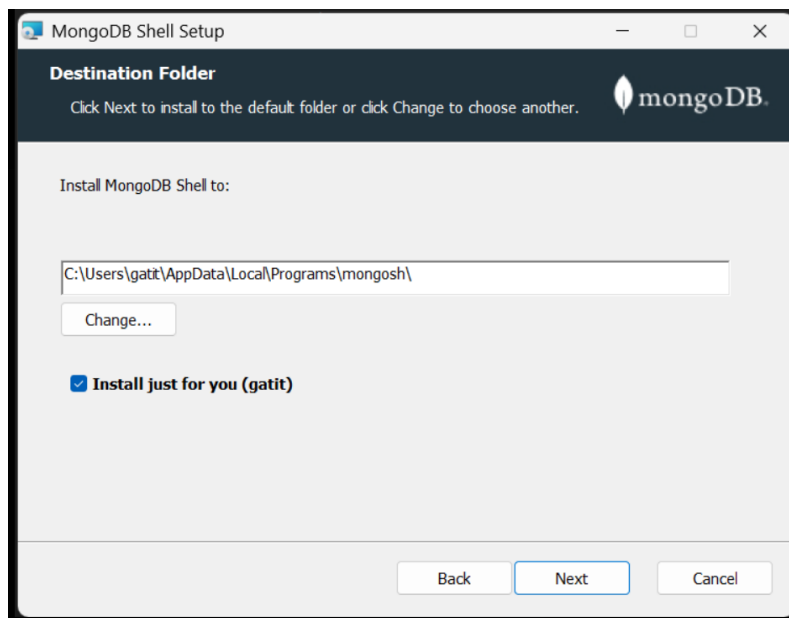


Ilustración 3 Selección de lugar de instalación

Al continuar solo se deberá aceptar el seguir con el proceso de instalación y se completará tras un tiempo de espera.

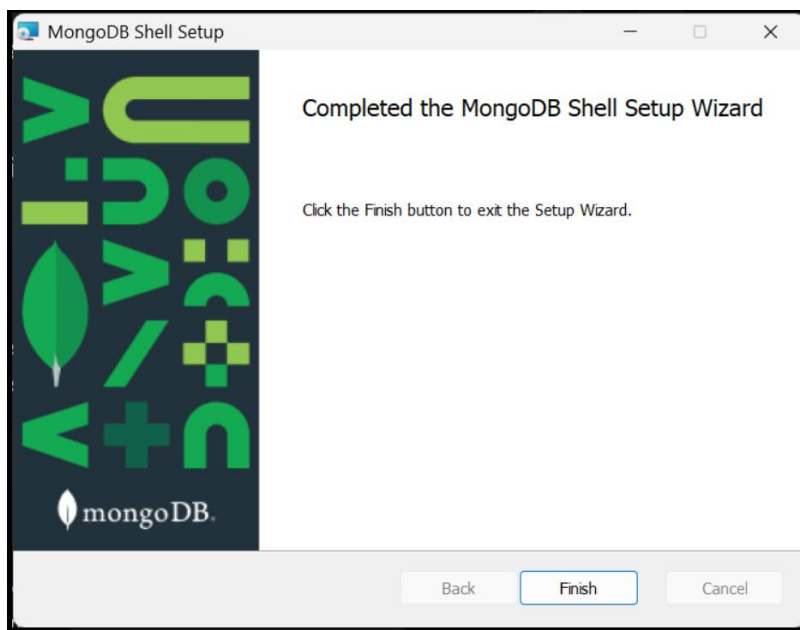


Ilustración 4 Instalación de MongoDB Completa

PROCESO PARA EL CAMBIO DE PATH DENTRO DE MONGO (DE SER NECESARIO)

Este proceso es útil ya que permite que los comandos del servidor y del shell se puedan ejecutar desde cualquier ubicación en la consola, sin necesidad de navegar hasta la carpeta de instalación cada vez. Esto facilita la administración de la base de datos, la ejecución de consultas y la configuración de herramientas adicionales.

Además, asegura que scripts o aplicaciones que dependan de MongoDB puedan encontrar los ejecutables automáticamente, evitando errores por rutas incorrectas y agilizando el trabajo tanto en Windows como en Linux.

Para realizar el proceso deberemos abrir las propiedades del sistema:

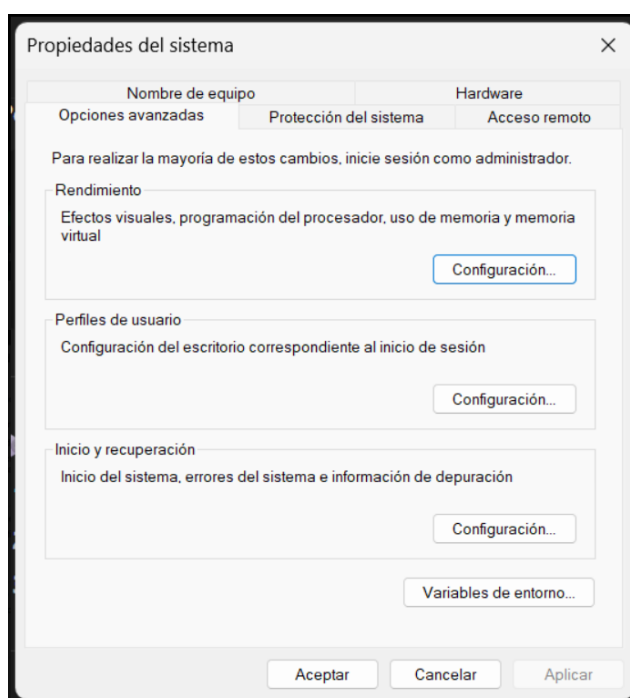


Ilustración 5 Propiedades del sistema



UNIVERSIDAD TÉCNICA DE AMBATO
FACULTAD DE INGENIERÍA EN SISTEMAS ELECTRÓNICA E INDUSTRIAL
CARRERA DE Elige un elemento.
CICLO ACADÉMICO: AGOSTO 2025 - ENERO 2026



Dentro de ellos nos dirigimos a variables de entorno:

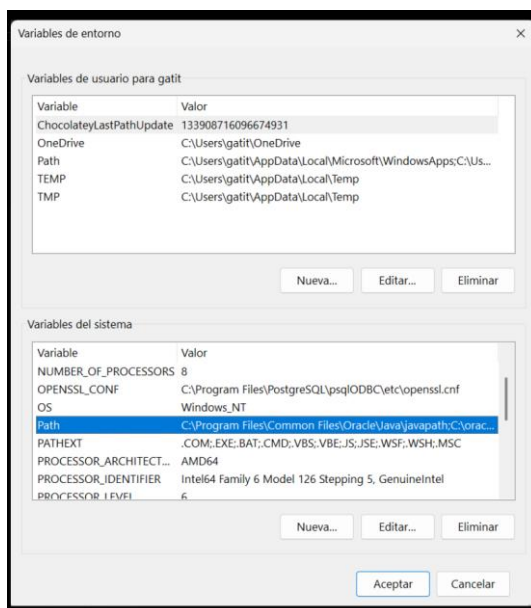


Ilustración 6 Configuración de Path

Seleccionamos Path y luego seleccionamos editar:

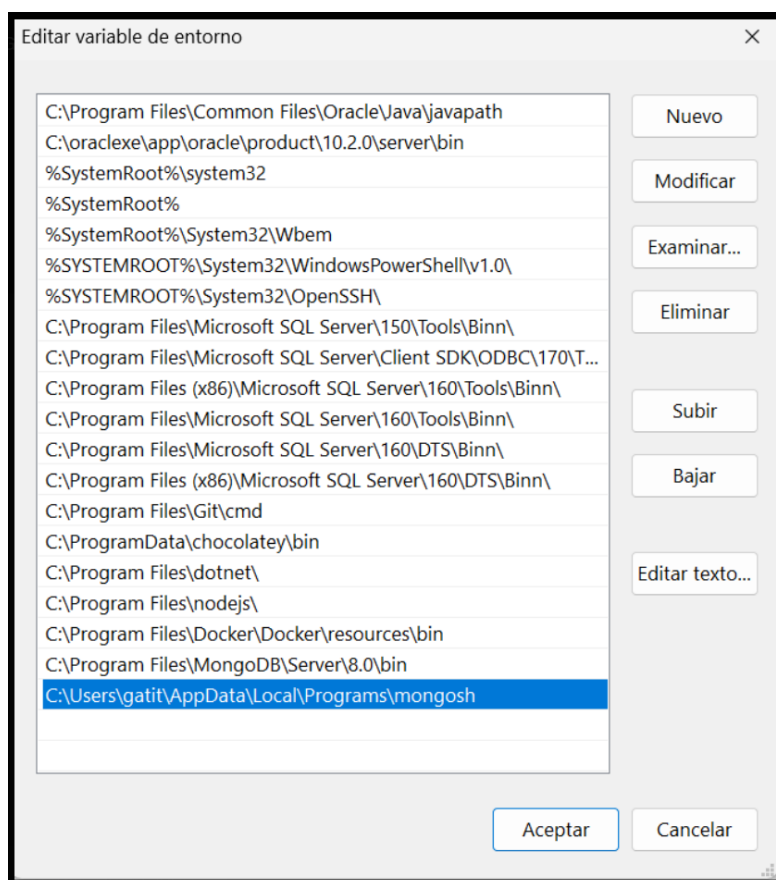


Ilustración 7 Cambio de Path

Una vez dentro bastara con seleccionar el path correcto y se finalizaría con la configuración.



AGREGAR LOS DIRECTORIOS A MONGODB EN WINDOWS

Es necesario para organizar los datos y la configuración del servidor, especialmente cuando se va a trabajar con sharding o múltiples instancias. Estos directorios (shard1, shard2, config) permiten almacenar los fragmentos de la base de datos y la información de configuración de manera ordenada, facilitando la gestión y replicación de los datos en el clúster.

Dentro del powershell de Windows se ejecutará el comando: **mongosh --version** para verificar si esta correctamente instalado:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Instale la versión más reciente de PowerShell para obtener nuevas características y mejoras. https://aka.ms/PSWindows

PS C:\Users\gatit> mongosh --version
2.5.8
PS C:\Users\gatit> mongod --version
db version v8.0.13
Build Info: {
  "version": "8.0.13",
  "gitVersion": "8dc5cd2a30c4524132e2d44bb314544dc477e611",
  "modules": [],
  "allocator": "tcmalloc-gperf",
  "environment": {
    "distmod": "windows"
  }
}
```

Ilustración 8 Verificación de la correcta instalación de MongoDB

Una vez comprobado la instalación procedemos agregar los directorios dentro del cmd:

```
Administrador: Símbolo del sistema
Microsoft Windows [Versión 10.0.26100.6584]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Windows\System32>mkdir "%USERPROFILE%\mongodb\shard1" "%USERPROFILE%\mongodb\shard2" "%USERPROFILE%\mongodb\config"

C:\Windows\System32>mkdir "%USERPROFILE%\mongodb\shard1\db" "%USERPROFILE%\mongodb\shard2\db" "%USERPROFILE%\mongodb\config\db"

C:\Windows\System32>
```

Ilustración 9 Agregar repositorios a mongodb

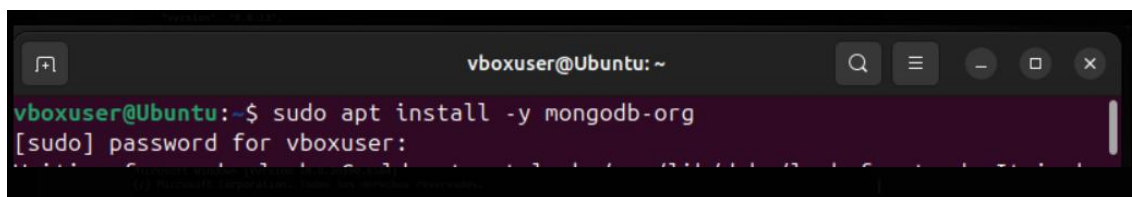
Una vez completa la instalación y configuración de MongoDB en Windows procedemos a realizar la instalación y configuración dentro de nuestra máquina virtual en VirtualBox con el sistema operativo Ubuntu.

INSTALACIÓN DE MONGODB EN UBUNTU

En este caso la instalación no se realizará mediante la página web sino simplemente con la ejecución de comandos.

Dentro del cmd de nuestro Ubuntu ejecutaremos el comando: **\$sudo apt install -y mongodb-org**

Dicho comando realizara la instalación.

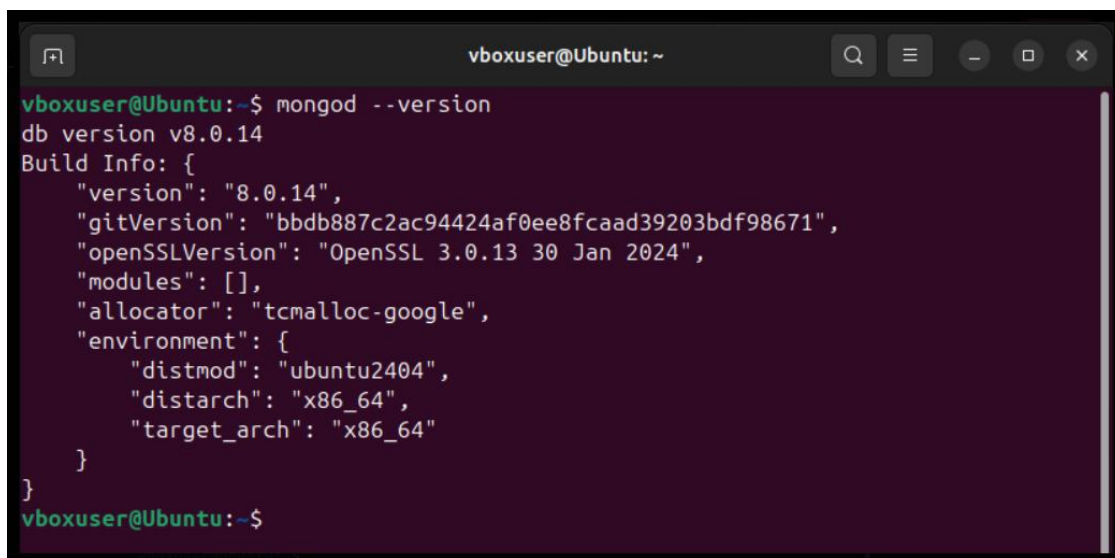


```
vboxuser@Ubuntu: ~  
vboxuser@Ubuntu:~$ sudo apt install -y mongodb-org  
[sudo] password for vboxuser:
```

Ilustración 10 Instalación de MongoDB en Ubuntu

De igual manera ejecutaremos el comando: `$mongo --version`

Este comando nos demostrara si la instalación se realizó correctamente.

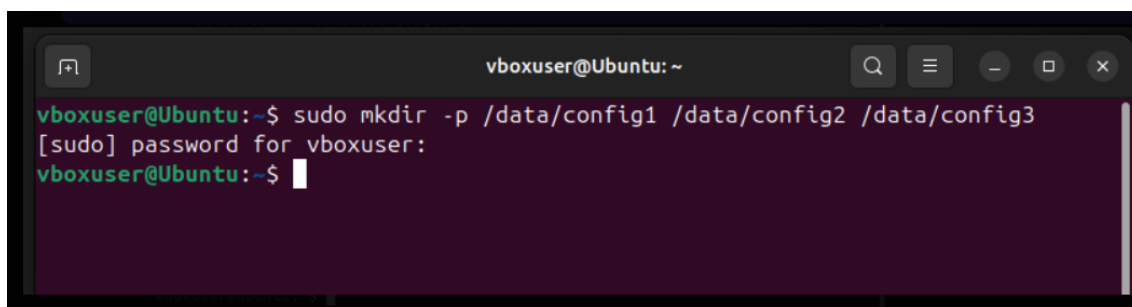


```
vboxuser@Ubuntu:~$ mongo --version  
db version v8.0.14  
Build Info: {  
  "version": "8.0.14",  
  "gitVersion": "bbdb887c2ac94424af0ee8fcaad39203bdf98671",  
  "opensslVersion": "OpenSSL 3.0.13 30 Jan 2024",  
  "modules": [],  
  "allocator": "tcmalloc-google",  
  "environment": {  
    "distmod": "ubuntu2404",  
    "distarch": "x86_64",  
    "target_arch": "x86_64"  
  }  
}  
vboxuser@Ubuntu:~$
```

Ilustración 11 Verificación de la Instalación correcta de mongodb en ubuntu

AGREGAR LOS DIRECTORIOS A MONGODB EN UBUNTU

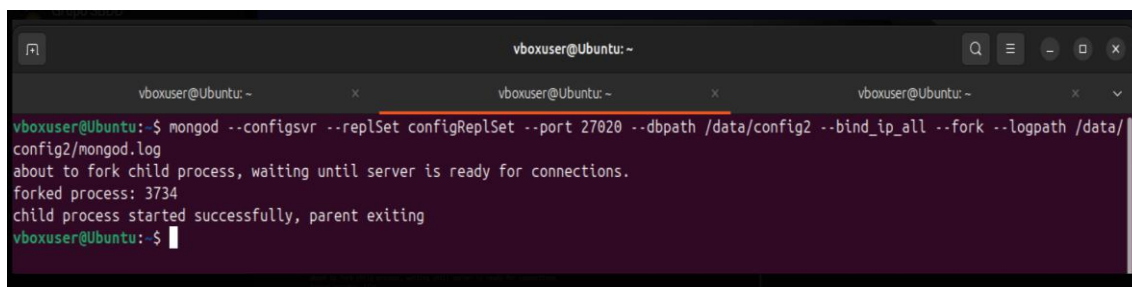
Es necesario para organizar los datos y la configuración del servidor, especialmente cuando se va a trabajar con sharding o múltiples instancias.



```
vboxuser@Ubuntu:~$ sudo mkdir -p /data/config1 /data/config2 /data/config3  
[sudo] password for vboxuser:  
vboxuser@Ubuntu:~$
```

Ilustración 12 Agregar directorios a mongodb en ubuntu

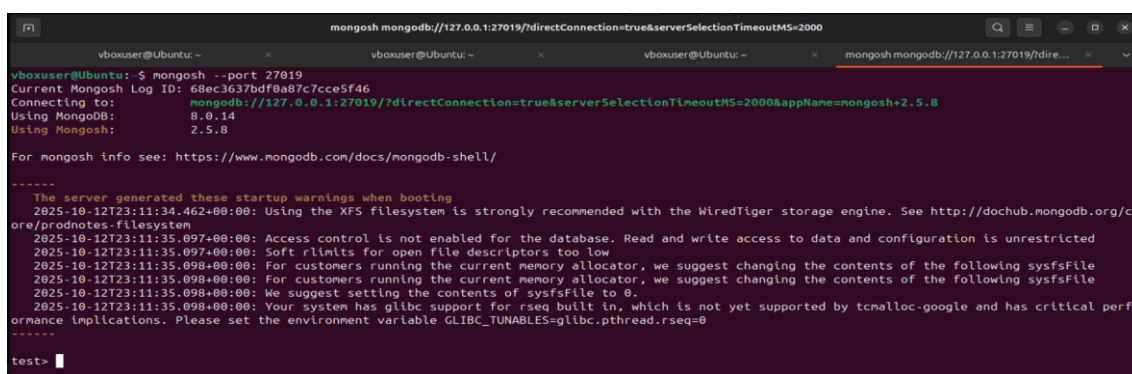
EJECUTAR 3 INTANCIAS CADA UNA EN UNA TERMINAL DIFERENTE



```
vboxuser@Ubuntu: ~  
vboxuser@Ubuntu: ~  
vboxuser@Ubuntu: ~  
vboxuser@Ubuntu: ~$ mongod --configsvr --replSet configReplSet --port 27021 --dbpath /data/config3 --bind_ip_all --fork --logpath /data/config3/mongod.log  
about to fork child process, waiting until server is ready for connections.  
forked process: 3825  
child process started successfully, parent exiting  
vboxuser@Ubuntu: ~$
```

INICIALIZACIÓN DE LA REPLICA SET

La inicialización de la réplica set en MongoDB permite que varias instancias del servidor trabajen juntas para replicar los datos, garantizando disponibilidad y tolerancia a fallos. Esto asegura que, si un nodo falla, los datos sigan accesibles desde otro miembro del conjunto, manteniendo la integridad y continuidad del sistema.



Verificación de la correcta inicialización:

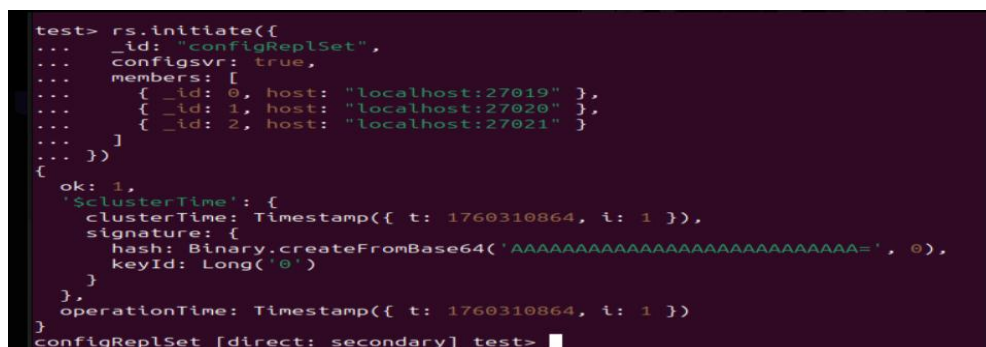


Ilustración 16 Verificación de la correcta inicialización



UNIVERSIDAD TÉCNICA DE AMBATO
FACULTAD DE INGENIERÍA EN SISTEMAS ELECTRÓNICA E INDUSTRIAL
CARRERA DE Elige un elemento.
CICLO ACADÉMICO: AGOSTO 2025 - ENERO 2026



```
mongosh mongodb://127.0.0.1:27019/?directConnection=true&serverSelectionTimeoutMS=2000
vboxuser@Ubuntu: ~
vboxuser@Ubuntu: ~
vboxuser@Ubuntu: ~
mongosh mongodb://127.0.0.1:27019/?dire...

configReplSet [direct: secondary] test> rs.status()
{
  set: 'configReplSet',
  date: ISODate('2025-10-12T23:14:39.229Z'),
  myState: 1,
  term: Long('1'),
  syncSourceHost: '',
  syncSourceId: -1,
  configsvr: true,
  heartbeatIntervalMillis: Long('2000'),
  majorityVoteCount: 2,
  writeMajorityCount: 2,
  votingMembersCount: 3,
  writableVotingMembersCount: 3,
  optimes: {
    lastCommittedOptime: { ts: Timestamp({ t: 1760310878, i: 1 }), t: Long('1') },
    lastCommittedWallTime: ISODate('2025-10-12T23:14:38.260Z'),
    readConcernMajorityOptime: { ts: Timestamp({ t: 1760310878, i: 1 }), t: Long('1') },
    appliedOptime: { ts: Timestamp({ t: 1760310878, i: 1 }), t: Long('1') },
    durableOptime: { ts: Timestamp({ t: 1760310878, i: 1 }), t: Long('1') },
    writtenOptime: { ts: Timestamp({ t: 1760310878, i: 1 }), t: Long('1') },
    lastAppliedWallTime: ISODate('2025-10-12T23:14:38.260Z'),
    lastDurableWallTime: ISODate('2025-10-12T23:14:38.260Z'),
    lastWrittenWallTime: ISODate('2025-10-12T23:14:38.260Z')
  },
  lastStableRecoveryTimestamp: Timestamp({ t: 1760310864, i: 1 }),
  electionCandidateMetrics: {
    lastElectionReason: 'electionTimeout',
    lastElectionDate: ISODate('2025-10-12T23:14:34.515Z'),
    electionTerm: Long('1'),
    lastCommittedOptimeAtElection: { ts: Timestamp({ t: 1760310864, i: 1 }), t: Long('-1') },
    lastSeenWrittenOptimeAtElection: { ts: Timestamp({ t: 1760310864, i: 1 }), t: Long('-1') },
    lastSeenOptimeAtElection: { ts: Timestamp({ t: 1760310864, i: 1 }), t: Long('-1') },
    numVotesNeeded: 2,

```

Ilustración 17 Verificación de la correcta inicialización

```
mongosh mongodb://127.0.0.1:27019/?directConnection=true&serverSelectionTimeoutMS=2000
vboxuser@Ubuntu: ~
vboxuser@Ubuntu: ~
vboxuser@Ubuntu: ~
mongosh mongodb://127.0.0.1:27019/?dire...

health: 1,
state: 2,
stateStr: 'SECONDARY',
uptime: 15,
optime: { ts: Timestamp({ t: 1760310878, i: 1 }), t: Long('1') },
optimeDurable: { ts: Timestamp({ t: 1760310878, i: 1 }), t: Long('1') },
optimeWritten: { ts: Timestamp({ t: 1760310878, i: 1 }), t: Long('1') },
optimeDate: ISODate('2025-10-12T23:14:38.000Z'),
optimeDurableDate: ISODate('2025-10-12T23:14:38.000Z'),
optimeWrittenDate: ISODate('2025-10-12T23:14:38.000Z'),
lastAppliedWallTime: ISODate('2025-10-12T23:14:38.260Z'),
lastDurableWallTime: ISODate('2025-10-12T23:14:38.260Z'),
lastWrittenWallTime: ISODate('2025-10-12T23:14:38.260Z'),
lastHeartbeat: ISODate('2025-10-12T23:14:38.570Z'),
lastHeartbeatRecv: ISODate('2025-10-12T23:14:37.503Z'),
pingMs: Long('0'),
lastHeartbeatMessage: '',
syncSourceHost: 'localhost:27019',
syncSourceId: 0,
infoMessage: '',
configVersion: 1,
configTerm: 1
},
{
  _id: 2,
  name: 'localhost:27021',
  health: 1,
  state: 2,
  stateStr: 'SECONDARY',
  uptime: 15,
  optime: { ts: Timestamp({ t: 1760310878, i: 1 }), t: Long('1') },
  optimeDurable: { ts: Timestamp({ t: 1760310878, i: 1 }), t: Long('1') },
  optimeWritten: { ts: Timestamp({ t: 1760310878, i: 1 }), t: Long('1') },
  optimeDate: ISODate('2025-10-12T23:14:38.000Z'),

```

Ilustración 18 Verificación de la correcta inicialización

```
optimeWrittenDate: ISODate('2025-10-12T23:14:38.000Z'),
lastAppliedWallTime: ISODate('2025-10-12T23:14:38.260Z'),
lastDurableWallTime: ISODate('2025-10-12T23:14:38.260Z'),
lastWrittenWallTime: ISODate('2025-10-12T23:14:38.260Z'),
lastHeartbeat: ISODate('2025-10-12T23:14:38.562Z'),
lastHeartbeatRecv: ISODate('2025-10-12T23:14:37.567Z'),
pingMs: Long('0'),
lastHeartbeatMessage: '',
syncSourceHost: 'localhost:27019',
syncSourceId: 0,
infoMessage: '',
configVersion: 1,
configTerm: 1
}
},
ok: 1,
'sclusterTime': {
  clusterTime: Timestamp({ t: 1760310878, i: 1 }),
  signature: {
    hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAAAA', 0),
    keyId: Long('0')
  }
},
operationTime: Timestamp({ t: 1760310878, i: 1 })
}
configReplSet [direct: primary] test> S
```

Ilustración 19 Verificación de la correcta inicialización



LANZAMIENTO DE LOS TRES NODOS



INICIALIZAMOS





UNIVERSIDAD TÉCNICA DE AMBATO
FACULTAD DE INGENIERÍA EN SISTEMAS ELECTRÓNICA E INDUSTRIAL
CARRERA DE Elige un elemento.
CICLO ACADÉMICO: AGOSTO 2025 - ENERO 2026



```
test> rs.initiate({
...   _id: "shard1RS",
...   members: [
...     { _id: 0, host: "localhost:27018" },
...     { _id: 1, host: "localhost:27028" },
...     { _id: 2, host: "localhost:27038" }
...   ]
... })
{
  ok: 1,
  'sclusterTime': {
    clusterTime: Timestamp({ t: 1760311274, i: 1 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAAAA', 0),
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1760311274, i: 1 })
}
shard1RS [direct: secondary] test>
```

Ilustración 25 Inicialización

VERIFICAMOS

```
mongosh mongodb://127.0.0.1:27018/?directConnection=true&serverSelectionTimeoutMS=2000
vboxuser@Ubuntu... x vboxuser@Ubuntu... x vboxuser@Ubuntu... x mongosh mong... x vboxuser@Ubuntu... x vboxuser@Ubuntu... x vboxuser@Ubuntu... x vboxuser@Ubuntu... x
shard1RS [direct: secondary] test> rs.status()
{
  set: 'shard1RS',
  date: ISODate('2025-10-12T23:21:27.987Z'),
  myState: 1,
  term: Long('1'),
  syncSourceHost: '',
  syncSourceId: -1,
  heartbeatIntervalMillis: Long('2000'),
  majorityVoteCount: 2,
  writeMajorityCount: 2,
  votingMembersCount: 3,
  writableVotingMembersCount: 3,
  optimes: {
    lastCommittedOpTime: { ts: Timestamp({ t: 1760311286, i: 1 }), t: Long('1') },
    lastCommittedWallTime: ISODate('2025-10-12T23:21:26.100Z'),
    readConcernMajorityOpTime: { ts: Timestamp({ t: 1760311286, i: 1 }), t: Long('1') },
    appliedOpTime: { ts: Timestamp({ t: 1760311286, i: 1 }), t: Long('1') },
    durableOpTime: { ts: Timestamp({ t: 1760311286, i: 1 }), t: Long('1') },
    writtenOpTime: { ts: Timestamp({ t: 1760311286, i: 1 }), t: Long('1') },
    lastAppliedWallTime: ISODate('2025-10-12T23:21:26.100Z'),
    lastDurableWallTime: ISODate('2025-10-12T23:21:26.100Z'),
    lastWrittenWallTime: ISODate('2025-10-12T23:21:26.100Z')
  },
  lastStableRecoveryTimestamp: Timestamp({ t: 1760311274, i: 1 }),
  electionCandidateMetrics: {
    lastElectionReason: 'electionTimeout',
    lastElectionDate: ISODate('2025-10-12T23:21:25.380Z'),
    electionTerm: Long('1'),
    lastCommittedOpTimeAtElection: { ts: Timestamp({ t: 1760311274, i: 1 }), t: Long('-1') },
    lastSeenWrittenOpTimeAtElection: { ts: Timestamp({ t: 1760311274, i: 1 }), t: Long('-1') },
    lastSeenOpTimeAtElection: { ts: Timestamp({ t: 1760311274, i: 1 }), t: Long('-1') },
    numVotesNeeded: 2,
    priorityAtElection: 1,

```

Ilustración 26 Verificación

```
mongosh mongodb://127.0.0.1:27018/?directConnection=true&serverSelectionTimeoutMS=2000
vboxuser@Ubuntu... x vboxuser@Ubuntu... x vboxuser@Ubuntu... x mongosh mong... x vboxuser@Ubuntu... x vboxuser@Ubuntu... x vboxuser@Ubuntu... x vboxuser@Ubuntu... x
electionTimeoutMillis: Long('10000'),
numCatchUpOps: Long('0'),
newTermStartDate: ISODate('2025-10-12T23:21:25.558Z'),
wMajorityWriteAvailabilityDate: ISODate('2025-10-12T23:21:25.993Z')
},
members: [
  {
    _id: 0,
    name: 'localhost:27018',
    health: 1,
    state: 1,
    stateStr: 'PRIMARY',
    optime: 109,
    optimeDate: ISODate('2025-10-12T23:21:26.000Z'),
    optimeTerm: { ts: Timestamp({ t: 1760311286, i: 1 }), t: Long('1') },
    optimeTermDate: ISODate('2025-10-12T23:21:26.000Z'),
    lastAppliedWallTime: ISODate('2025-10-12T23:21:26.100Z'),
    lastDurableWallTime: ISODate('2025-10-12T23:21:26.100Z'),
    lastWrittenWallTime: ISODate('2025-10-12T23:21:26.100Z'),
    syncSourceHost: '',
    syncSourceId: -1,
    infoMessage: 'Could not find member to sync from',
    electionTime: Timestamp({ t: 1760311285, i: 1 }),
    electionDate: ISODate('2025-10-12T23:21:25.000Z'),
    configVersion: 1,
    configTerm: 1,
    self: true,
    lastHeartbeatMessage: ''
  },
  {
    _id: 1,
    name: 'localhost:27028',
    health: 1,

```

Ilustración 27 Verificación



UNIVERSIDAD TÉCNICA DE AMBATO
FACULTAD DE INGENIERÍA EN SISTEMAS ELECTRÓNICA E INDUSTRIAL
CARRERA DE Elige un elemento.
CICLO ACADÉMICO: AGOSTO 2025 - ENERO 2026



```
mongosh mongodb://127.0.0.1:27018/?directConnection=true&serverSelectionTimeoutMS=2000
vboxuser@Ubu... x vboxuser@Ubu... x vboxuser@Ubu... x mongosh mong... x vboxuser@Ubu... x vboxuser@Ubu... x vboxuser@Ubu... x vboxuser@Ubu... x
state: 2,
stateStr: 'SECONDARY',
uptime: 13,
optime: { ts: Timestamp({ t: 1760311286, i: 1 }), t: Long('1') },
optimeDurable: { ts: Timestamp({ t: 1760311286, i: 1 }), t: Long('1') },
optimeWritten: { ts: Timestamp({ t: 1760311286, i: 1 }), t: Long('1') },
optimeDate: ISODate('2025-10-12T23:21:26.000Z'),
optimeDurableDate: ISODate('2025-10-12T23:21:26.000Z'),
optimeWrittenDate: ISODate('2025-10-12T23:21:26.000Z'),
lastAppliedWallTime: ISODate('2025-10-12T23:21:26.100Z'),
lastDurableWallTime: ISODate('2025-10-12T23:21:26.100Z'),
lastWrittenWallTime: ISODate('2025-10-12T23:21:26.100Z'),
lastHeartbeat: ISODate('2025-10-12T23:21:27.437Z'),
lastHeartbeatRecv: ISODate('2025-10-12T23:21:26.456Z'),
pingMs: Long('0'),
lastHeartbeatMessage: '',
syncSourceHost: 'localhost:27018',
syncSourceId: 0,
infoMessage: '',
configVersion: 1,
configTerm: 1
},
{
  _id: 2,
  name: 'localhost:27038',
  health: 1,
  state: 2,
  stateStr: 'SECONDARY',
  uptime: 13,
  optime: { ts: Timestamp({ t: 1760311286, i: 1 }), t: Long('1') },
  optimeDurable: { ts: Timestamp({ t: 1760311286, i: 1 }), t: Long('1') },
  optimeWritten: { ts: Timestamp({ t: 1760311286, i: 1 }), t: Long('1') },
  optimeDate: ISODate('2025-10-12T23:21:26.000Z'),
  optimeDurableDate: ISODate('2025-10-12T23:21:26.000Z'),
  optimeWrittenDate: ISODate('2025-10-12T23:21:26.000Z'),
  lastAppliedWallTime: ISODate('2025-10-12T23:21:26.100Z'),
  lastDurableWallTime: ISODate('2025-10-12T23:21:26.100Z'),
  lastWrittenWallTime: ISODate('2025-10-12T23:21:26.100Z'),
  lastHeartbeat: ISODate('2025-10-12T23:21:27.437Z'),
  lastHeartbeatRecv: ISODate('2025-10-12T23:21:26.456Z'),
  pingMs: Long('0'),
  lastHeartbeatMessage: '',
  syncSourceHost: 'localhost:27018',
  syncSourceId: 0,
  infoMessage: '',
  configVersion: 1,
  configTerm: 1
}
```

Ilustración 28 Verificación

LEVANTAR EL ROUTER (MONGOS)

```
vboxuser@Ubuntu:~$ sudo mongos --configdb configReplSet/localhost:27019,localhost:27020,localhost:27021 --bind_ip_all --port 27017 --logpath /data/mongos.log
{"t":{"$date":"2025-10-12T23:33:57.317Z"},"s":"I", "c":"CONTROL", "id":"20697", "ctx":"main","msg":"Renamed existing log file","attr":{"oldLogPath":"/data/mongos.log","newLogPath":"/data/mongos.log.2025-10-12T23-33-57"}}
vboxuser@Ubuntu:~$
```

Ilustración 29 Levantar el ROUTER (mongos)

CONEXION AL ROUTER

```
vboxuser@Ubuntu:~$ mongosh --port 27017
Current Mongosh Log ID: 68ec3b58e92fa43754ce5f46
Connecting to:
  mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.5.8
Using MongoDB:
  8.0.14
Using Mongosh:
  2.5.8

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

-----
The server generated these startup warnings when booting
2025-10-12T23:02:00.803+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See http://dochub.mongodb.org/core/prodnotes-filesystem
2025-10-12T23:02:04.839+00:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
2025-10-12T23:02:04.839+00:00: For customers running the current memory allocator, we suggest changing the contents of the following sysfsFile
2025-10-12T23:02:04.843+00:00: For customers running the current memory allocator, we suggest changing the contents of the following sysfsFile
2025-10-12T23:02:04.843+00:00: We suggest setting the contents of sysfsFile to 0.
-----

test>
```

Ilustración 30 Conéctate al router

AGREGA LOS SHARDS AL CLUSTER



UNIVERSIDAD TÉCNICA DE AMBATO
FACULTAD DE INGENIERÍA EN SISTEMAS ELECTRÓNICA E INDUSTRIAL
CARRERA DE Elige un elemento.
CICLO ACADÉMICO: AGOSTO 2025 - ENERO 2026



```
mongosh mongoDB://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
vboxuser@Ubuntu: ~
vboxuser@Ubuntu: ~
mongosh mongoDB://127.0.0.1:27017/?directConnection=true...

vboxuser@Ubuntu:~$ mongosh --port 27017
Current Mongosh Log ID: 68ec4358aeb338556bce5f46
Connecting to: mongosh://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.5.8
Using MongoDB: 8.0.14
Using Mongosh: 2.5.8

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

-----
The server generated these startup warnings when booting
2025-10-13T00:04:42.380+00:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
2025-10-13T00:04:42.380+00:00: You are running this process as the root user, which is not recommended
-----

[direct: mongos] test> sh.addShard("shard1RS/localhost:27018,localhost:27028,localhost:27038")
{
  shardAdded: 'shard1RS',
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp([ t: 1760314247, i: 3 ]),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAAAA=', 0),
      keyId: Long('0')
    }
  },
  operationTime: Timestamp([ t: 1760314247, i: 3 ])
}
[direct: mongos] test>
```

Ilustración 31 Agrega los shards al cluster

```
[direct: mongos] test> sh.addShard("shard2RS/localhost:27048,localhost:27058,localhost:27068")
{
  shardAdded: 'shard2RS',
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp([ t: 1760314264, i: 24 ]),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAAAA=', 0),
      keyId: Long('0')
    }
  },
  operationTime: Timestamp([ t: 1760314264, i: 18 ])
}
[direct: mongos] test>
```

Ilustración 32 Agrega los shards al cluster

VERIFICAMOS CONFIGURACIÓN GENERAL

```
[direct: mongos] test> sh.status()
shardingVersion
{ _id: 1, clusterId: ObjectId('68ec365b149cdd8197ad298c') }
---
shards
[
  {
    _id: 'shard1RS',
    host: 'shard1RS/localhost:27018,localhost:27028,localhost:27038',
    state: 1,
    topologyTime: Timestamp([ t: 1760314246, i: 10 ]),
    replSetConfigVersion: Long('-1')
  },
  {
    _id: 'shard2RS',
    host: 'shard2RS/localhost:27048,localhost:27058,localhost:27068',
    state: 1,
    topologyTime: Timestamp([ t: 1760314264, i: 9 ]),
    replSetConfigVersion: Long('-1')
  }
]
---
active mongoses
[ { '8.0.14': 1 } ]
---
autosplit
{ 'Currently enabled': 'yes' }
---
balancer
{
  'Currently enabled': 'yes',
  'Failed balancer rounds in last 5 attempts': 0,
  'Currently running': 'no',
  'Migration Results for the last 24 hours': 'No recent migrations'
}
```

Ilustración 33 Verificamos configuración general



CREAR BASE DE DATOS Y COLECCIONES (EQUIVALENTE AL MODELO POSTGRESQL)

```
[direct: mongos] test> use flota
switched to db flota
[direct: mongos] flota> 
```

Ilustración 34 Crear base de datos y colecciones (equivalente al modelo PostgreSQL)

CREAR COLECCIONES PRINCIPALES

```
[direct: mongos] flota> db.createCollection("vehiculos")
... db.createCollection("documentos")
... db.createCollection("conductores")
... db.createCollection("mantenimientos")
... db.createCollection("gasolina")
... db.createCollection("rutas")
{ ok: 1 }
[direct: mongos] flota>
```

Ilustración 35 Crear colecciones principales

HABILITAR SHARDING Y DEFINIR SHARD KEYS

HABILITAR SHARDING EN LA BASE

```
[direct: mongos] flota> sh.enableSharding("flota")
{
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1760314374, i: 1 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAAAA=', 0),
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1760314374, i: 1 })
}
[direct: mongos] flota>
```

Ilustración 36 HABILITAR SHARDING Y DEFINIR SHARD KEYS

CREAR ÍNDICES Y SHARDEAR LAS COLECCIONES



```
[direct: mongos] flota> db.vehiculos.createIndex({ id_veh: "hashed" })
... sh.shardCollection("flota.vehiculos", { id_veh: "hashed" })
...
... db.rutas.createIndex({ vehiculo_id: 1, fec_rut: 1 })
... sh.shardCollection("flota.rutas", { vehiculo_id: 1, fec_rut: 1 })
{
  collectionssharded: 'flota.rutas',
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1760314393, i: 13 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAAAA=', 0),
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1760314393, i: 13 })
}
[direct: mongos] flota> █
```

Ilustración 37 Crear índices y shardear las colecciones

VERIFICAMOS DISTRIBUCIÓN

```
[direct: mongos] flota> sh.status()
shardingVersion
{ _id: 1, clusterId: ObjectId('68ec365b149cdd8197ad298c') }
---
shards
[
  {
    _id: 'shard1RS',
    host: 'shard1RS/localhost:27018,localhost:27028,localhost:27038',
    state: 1,
    topologyTime: Timestamp({ t: 1760314246, i: 10 }),
    replSetConfigVersion: Long('-1')
  },
  {
    _id: 'shard2RS',
    host: 'shard2RS/localhost:27048,localhost:27058,localhost:27068',
    state: 1,
    topologyTime: Timestamp({ t: 1760314264, i: 9 }),
    replSetConfigVersion: Long('-1')
  }
]
---
active mongoses
[ { '8.0.14': 1 } ]
---
autosplit
{ 'Currently enabled': 'yes' }
---
balancer
{
  'Currently enabled': 'yes',
  'Currently running': 'no',
  'Failed balancer rounds in last 5 attempts': 0,
  'Migration Results for the last 24 hours': 'No recent migrations'
}
```

Ilustración 38 Verificamos distribución



```
}  
---  
shardedDataDistribution  
[  
  {  
    ns: 'flota.rutas',  
    shards: [  
      {  
        shardName: 'shard2RS',  
        numOrphanedDocs: 0,  
        numOwnedDocuments: 0,  
        ownedSizeBytes: 0,  
        orphanedSizeBytes: 0  
      }  
    ],  
  },  
  {  
    ns: 'flota.vehiculos',  
    shards: [  
      {  
        shardName: 'shard1RS',  
        numOrphanedDocs: 0,  
        numOwnedDocuments: 0,  
        ownedSizeBytes: 0,  
        orphanedSizeBytes: 0  
      },  
      {  
        shardName: 'shard2RS',  
        numOrphanedDocs: 0,  
        numOwnedDocuments: 0,  
        ownedSizeBytes: 0,  
        orphanedSizeBytes: 0  
      }  
    ]  
  }  
]
```

Ilustración 39 Verificamos distribución

```
},  
{  
  ns: 'config.system.sessions',  
  shards: [  
    {  
      shardName: 'shard1RS',  
      numOrphanedDocs: 0,  
      numOwnedDocuments: 7,  
      ownedSizeBytes: 693,  
      orphanedSizeBytes: 0  
    }  
  ]  
}  
---  
databases  
[  
  {  
    database: { _id: 'config', primary: 'config', partitioned: true },  
    collections: {  
      'config.system.sessions': {  
        shardKey: { _id: 1 },  
        unique: false,  
        balancing: true,  
        chunkMetadata: [ { shard: 'shard1RS', nChunks: 1 } ],  
        chunks: [  
          { min: { _id: MinKey() }, max: { _id: MaxKey() }, 'on shard': 'shard1RS', 'last modified': Timestamp({ t: 1, i: 0 }) }  
        ],  
        tags: []  
      }  
    }  
  },  
  {  
    database: { _id: 'config', primary: 'config', partitioned: true },  
    collections: {  
      'config.system.sessions': {  
        shardKey: { _id: 1 },  
        unique: false,  
        balancing: true,  
        chunkMetadata: [ { shard: 'shard1RS', nChunks: 1 } ],  
        chunks: [  
          { min: { _id: MinKey() }, max: { _id: MaxKey() }, 'on shard': 'shard1RS', 'last modified': Timestamp({ t: 1, i: 0 }) }  
        ],  
        tags: []  
      }  
    }  
  }  
]
```

Ilustración 40 Verificamos distribución



UNIVERSIDAD TÉCNICA DE AMBATO
FACULTAD DE INGENIERÍA EN SISTEMAS ELECTRÓNICA E INDUSTRIAL
CARRERA DE Elige un elemento.
CICLO ACADÉMICO: AGOSTO 2025 - ENERO 2026



```
database: {
  _id: 'flota',
  primary: 'shard2RS',
  version: {
    uuid: UUID('6065f0ff-aeb6-4917-92e7-f52aec4510d5'),
    timestamp: Timestamp({ t: 1760314354, i: 3 }),
    lastMod: 1
  }
},
collections: {
  'flota.rutas': {
    shardKey: { vehiculo_id: 1, fec_rut: 1 },
    unique: false,
    balancing: true,
    chunkMetadata: [ { shard: 'shard2RS', nChunks: 1 } ],
    chunks: [
      { min: { vehiculo_id: MinKey(), fec_rut: MinKey() }, max: { vehiculo_id: MaxKey(), fec_rut: MaxKey() }, 'on shard': 'shard2RS', 'last modified': Timestamp({ t: 1, i: 0 }) }
    ],
    tags: []
  },
  'flota.vehiculos': {
    shardKey: { id_veh: 'hashed' },
    unique: false,
    balancing: true,
    chunkMetadata: [
      { shard: 'shard1RS', nChunks: 1 },
      { shard: 'shard2RS', nChunks: 1 }
    ],
    chunks: [
      { min: { id_veh: MinKey() }, max: { id_veh: Long('0') }, 'on shard': 'shard2RS', 'last modified': Timestamp({ t: 1, i: 0 }) },
      { min: { id_veh: Long('0') }, max: { id_veh: MaxKey() }, 'on shard': 'shard1RS', 'last modified': Timestamp({ t: 1, i: 1 }) }
    ],
    tags: []
  }
}
```

Ilustración 41 Verificamos distribución

INGRESO EN PLSQL

```
postgres@Ubuntu:~$ psql
psql (17.6 (Ubuntu 17.6-1.pgdg24.04+1))
Type "help" for help.

postgres=# create database EmpresaViajes with owner postgres;
CREATE DATABASE
```

Ilustración 42 Ingreso en PLSQL

IMPLEMENTACIÓN DE TRIGGERS EN POSTGRESQL (REPLICACIÓN LÓGICA)

```
postgres=# CREATE TABLE RUTAS (
  ID_RUT SERIAL PRIMARY KEY,
  FEC_RUT DATE NOT NULL,
  ORI_RUT_VIA VARCHAR(10) NOT NULL,
  DES_RUT_VIA VARCHAR(10) NOT NULL,
  DIS_RRE_KM DECIMAL(10,2) NOT NULL,
  DUR_EST_VIA DECIMAL(10,2) NOT NULL,
  ID_VEH_PER INT NOT NULL,
  FOREIGN KEY (ID_VEH_PER) REFERENCES VEHICULO(ID_VEH)
);
CREATE TABLE
```

Ilustración 43 Implementación de triggers en PostgreSQL (replicación lógica)

VERIFICACIÓN DE SCHEMAS



```
postgres=# \dt
```

Schema	Name	Type	Owner
public	conductores	table	postgres
public	documentos	table	postgres
public	gasolina	table	postgres
public	mantenimientos	table	postgres
public	rutas	table	postgres
public	vehiculo	table	postgres

(6 rows)

Ilustración 44 Verificación de Schemas

CREACIÓN DE LOS TRIGGERS

```
postgres=# CREATE OR REPLACE FUNCTION verificar_licencia_vencida()  
RETURNS TRIGGER AS $$  
BEGIN  
    -- Si la fecha de vencimiento es menor que la fecha actual, lanzar un error  
    IF NEW.FEC_VEN_LIC < CURRENT_DATE THEN  
        RAISE EXCEPTION 'No se puede registrar un conductor con licencia vencida.';  
    END IF;  
    RETURN NEW;  
END;  
$$ LANGUAGE plpgsql;  
  
CREATE TRIGGER trg_verificar_licencia_vencida  
BEFORE INSERT OR UPDATE ON CONDUCTORES  
FOR EACH ROW  
EXECUTE FUNCTION verificar_licencia_vencida();
```

Ilustración 45 Creación de los triggers

CONTROL DE DISPONIBILIDAD DE VEHÍCULOS



```
postgres=# CREATE OR REPLACE FUNCTION calcular_costo_viaje()
RETURNS TRIGGER AS $$
DECLARE
    precio_litro DECIMAL(10,2);
BEGIN
    -- Obtener el precio por litro de gasolina del vehículo asociado a la ruta
    SELECT (PRECIO_X_GALON / 3.785) INTO precio_litro
    FROM GASOLINA g
    JOIN VEHICULO v ON g.ID_VEH_PER = v.ID_VEH
    WHERE v.ID_VEH = NEW.ID_VEH_PER
    LIMIT 1;

    -- Calcular el costo del viaje
    NEW.COSTO_VIAJE := precio_litro * NEW.DIS_RRE_KM;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_calcular_costo_viaje
BEFORE INSERT OR UPDATE ON RUTAS
FOR EACH ROW
EXECUTE FUNCTION calcular_costo_viaje();
```

Ilustración 46 Control de disponibilidad de vehículos

IMPORTAR A MONGODB

```
vboxuser@Ubuntu:~$ mongoimport --db flota --collection vehiculos --type csv --file /tmp/vehiculos.csv --headerline
2025-10-13T00:56:37.659+0000    connected to: mongodb://localhost/
2025-10-13T00:56:37.731+0000    5 document(s) imported successfully. 0 document(s) failed to import.
vboxuser@Ubuntu:~$
```

Ilustración 47 Importar a MongoDB

EXPORTAR CADA TABLA A CSV DESDE PSQL

```
postgres=# COPY (SELECT * FROM VEHICULO) TO '/tmp/vehiculos.csv' WITH CSV HEADER;
COPY 5
postgres=# ^C
postgres=# \copy DOCUMENTOS TO 'documentos.csv' CSV HEADER;
COPY 5
postgres=# \copy CONDUCTORES TO 'conductores.csv' CSV HEADER;
COPY 5
postgres=# \copy MANTENIMIENTOS TO 'mantenimientos.csv' CSV HEADER;
COPY 5
postgres=# \copy GASOLINA TO 'gasolina.csv' CSV HEADER;
COPY 5
postgres=# \copy RUTAS TO 'rutas.csv' CSV HEADER;
COPY 5
postgres=#
```

Ilustración 48 Exportar cada tabla a CSV desde psql



IMPORTAR A MONGODB

```
vboxuser@Ubuntu:~$ mongoimport --db flota --collection conductores --type csv --file /tmp/conductores.csv --headerline
2025-10-13T01:03:18.348+0000 connected to: mongodb://localhost/
2025-10-13T01:03:18.515+0000 5 document(s) imported successfully. 0 document(s) failed to import.
```

Ilustración 49 Importar a MongoDB

```
vboxuser@Ubuntu:~$ mongoimport --db flota --collection mantenimientos --type csv --file /tmp/mantenimientos.csv --headerline
2025-10-13T01:04:16.048+0000 connected to: mongodb://localhost/
2025-10-13T01:04:16.130+0000 5 document(s) imported successfully. 0 document(s) failed to import.
```

Ilustración 50 Importar a MongoDB

```
vboxuser@Ubuntu:~$ mongoimport --db flota --collection mantenimientos --type csv --file /tmp/mantenimientos.csv --headerline
2025-10-13T01:04:16.048+0000 connected to: mongodb://localhost/
2025-10-13T01:04:16.130+0000 5 document(s) imported successfully. 0 document(s) failed to import.
```

Ilustración 51 Importar a MongoDB

```
vboxuser@Ubuntu:~$ mongoimport --db flota --collection gasolina --type csv --file /tmp/gasolina.csv --headerline
2025-10-13T01:05:06.055+0000 connected to: mongodb://localhost/
2025-10-13T01:05:06.099+0000 5 document(s) imported successfully. 0 document(s) failed to import.
```

Ilustración 52 Importar a MongoDB

```
vboxuser@Ubuntu:~$ mongoimport --db flota --collection gasolina --type csv --file /tmp/gasolina.csv --headerline
2025-10-13T01:05:06.055+0000 connected to: mongodb://localhost/
2025-10-13T01:05:06.099+0000 5 document(s) imported successfully. 0 document(s) failed to import.
```

Ilustración 53 Importar a MongoDB

```
vboxuser@Ubuntu:~$ mongoimport --db flota --collection rutas --type csv --file /tmp/rutas.csv --headerline
2025-10-13T01:05:48.330+0000 connected to: mongodb://localhost/
2025-10-13T01:05:48.475+0000 5 document(s) imported successfully. 0 document(s) failed to import.
```

Ilustración 54 Importar a MongoDB

2.7 Resultados obtenidos

Se logró instalar y configurar correctamente MongoDB en el sistema, incluyendo la asignación de rutas en el Path, la creación de los directorios para el sharding y la inicialización del replica set. Se verificó el funcionamiento del Mongo Shell y la correcta comunicación entre las instancias configuradas. Además, se completó la instalación y configuración de PostgreSQL, donde se crearon las tablas, se insertaron datos de prueba y se implementaron



los triggers necesarios para automatizar validaciones y cálculos. En general, todos los procesos se ejecutaron de forma exitosa, comprobando el correcto desempeño de las bases de datos y su integración dentro del entorno de trabajo.

2.8 Habilidades blandas empleadas en la práctica

- ☐ Liderazgo
- ☐ Trabajo en equipo
- ☐ Comunicación asertiva
- ☐ La empatía
- ☒ Pensamiento crítico
- ☐ Flexibilidad
- ☒ La resolución de conflictos
- ☒ Adaptabilidad
- ☒ Responsabilidad

Durante el desarrollo de la práctica se aplicaron diversas habilidades blandas que contribuyeron al correcto proceso de instalación, configuración y verificación de MongoDB y PostgreSQL. El pensamiento crítico fue fundamental para analizar los pasos técnicos de configuración del sharding y la replicación, identificando posibles errores en las rutas o parámetros del sistema y aplicando las correcciones necesarias. La resolución de problemas permitió superar dificultades relacionadas con la creación de directorios, la inicialización del replica set y la ejecución de comandos en el shell, garantizando que cada componente funcionara correctamente. La adaptabilidad se evidenció al trabajar en distintos entornos operativos, como Windows y Linux, ajustando los comandos y procedimientos según las necesidades de cada sistema. Finalmente, la responsabilidad se reflejó en el cumplimiento ordenado de cada etapa, desde la instalación hasta la validación final, asegurando que los resultados obtenidos fueran precisos y funcionales.

2.9 Conclusiones

- La práctica permitió comprender el proceso completo de instalación y configuración de MongoDB tanto en Windows como en Linux, destacando la importancia de definir correctamente las rutas y directorios para garantizar el funcionamiento del sistema.
- Se comprobó el correcto uso del sharding y la replicación, logrando distribuir los datos de manera eficiente y asegurando la disponibilidad y consistencia de la información en las distintas instancias configuradas.
- La integración con PostgreSQL reforzó el manejo de bases de datos relacionales mediante la creación de tablas, inserción de datos y aplicación de triggers, demostrando la utilidad de las validaciones automáticas para mantener la integridad de los registros.

2.10 Recomendaciones

- Mantener siempre el uso de transacciones en operaciones críticas, ya que permiten garantizar la atomicidad y evitar inconsistencias en la base de datos.
- Documentar claramente las relaciones entre tablas y las reglas de integridad referencial, lo que facilitará el mantenimiento y la detección de errores en futuros desarrollos.