

Universidad Tecnológica de Xicoteppec de Juárez

Ing. En Desarrollo y Gestión de Software

Periodo Enero - Abril - 2024



Administración de Base de Datos

Dirección General de Hospital

Plan de Seguridad MySQL

8º A

Crespo Alvarado Carlos Iván

210237

Prof. M.T.I. MARCO A. RAMÍREZ HERNÁNDEZ

Introducción

Un plan de seguridad en MySQL es un conjunto de medidas, políticas y procedimientos diseñados para proteger una base de datos MySQL contra amenazas de seguridad, como accesos no autorizados, pérdida de datos, corrupción de datos, y otros riesgos relacionados con la seguridad de la información.

Algunos elementos comunes que podrían incluirse en un plan de seguridad de MySQL son:

- **Autenticación y autorización:** Establecer políticas sólidas para autenticar a los usuarios y controlar su acceso a la base de datos. Esto puede incluir el uso de contraseñas fuertes, autenticación de dos factores, y asignación adecuada de privilegios de usuario.
- **Cifrado de datos:** Implementar cifrado para proteger datos sensibles mientras están en tránsito (por ejemplo, mediante conexiones SSL/TLS) y mientras están en reposo en la base de datos (usando técnicas como el cifrado de columnas o el cifrado de tablas).
- **Auditoría y registro de eventos:** Configurar la base de datos para registrar eventos de seguridad relevantes, como intentos de acceso fallidos, cambios en la estructura de la base de datos o actividades de usuarios privilegiados. Luego, revisar periódicamente estos registros para detectar actividades sospechosas.
- **Actualizaciones y parches:** Mantener la base de datos MySQL actualizada instalando parches de seguridad y actualizaciones proporcionadas por el proveedor de MySQL. Esto ayuda a proteger la base de datos contra vulnerabilidades conocidas y exploits.
- **Respaldo y recuperación de datos:** Implementar políticas de respaldo regulares para garantizar que los datos críticos estén protegidos contra la pérdida y puedan recuperarse en caso de un incidente de seguridad o un fallo del sistema.
- **Firewalls y restricciones de red:** Utilizar firewalls y otras medidas de seguridad de red para controlar el tráfico que llega a la base de datos MySQL y restringir el acceso solo a las direcciones IP y los puertos necesarios.
- **Pruebas de seguridad:** Realizar pruebas de penetración y evaluaciones de vulnerabilidades de forma periódica para identificar posibles debilidades en la seguridad de la base de datos y tomar medidas correctivas.

En resumen, un plan de seguridad en MySQL es esencial para proteger la integridad, confidencialidad y disponibilidad de los datos almacenados en la base de datos MySQL, y para mitigar los riesgos asociados con las amenazas de seguridad.

Objetivo

Una de las principales preocupaciones en la administración de bases de datos es saber quién será el responsable de gestionar el correcto funcionamiento de la base de datos. Por lo que el objetivo principal es establecer normas y privilegios en la lectura, edición, inserción y eliminación a usuarios específicos de los departamentos del Hospital General PrivilegeCare, en función a las solicitudes de sus departamentos, por lo que solo contarán con los privilegios de Inserción y lectura, mientras el administrador del área de Dirección General podrá manipular cada una de las solicitudes con la aprobación, negación y cancelación de la solicitud del servicio que solicita.

Objetivos Específicos

La atención de las solicitudes estarán atendidas de acuerdo al listado de las solicitudes más longevas a las más recientes, el objetivo es que dentro de la gestión en la aprobación de los servicios el responsable se comunicara con las áreas de altos mandos enviando la información de la solicitud en un lapso pronto de respuesta para su aprobación o negación del servicio, del igual manera podrá realizar anotaciones relevantes en las solicitudes de las áreas solicitantes, esto permitirá que una vez solicitada, y autorizada se refleje en la bitácora de movimientos de área de Dirección General registrando la información en los datos históricos del hospital estando listos para su consulta.

Las solicitudes sin antecedentes relevantes o enviadas por error humano podrán ser eliminadas de la tabla registrándose en la tabla bitácora de la Dirección General, en ella se establece que departamento realizó la solicitud y los motivos por lo que la solicitud fue cancelada.

Los usuarios de los departamentos que se coordinan con la Dirección General solo tendrán los privilegios de Inserción y lectura, por lo que su restricción a los campos de Edición y Eliminación no estará asignada en el rol establecido.

El administrador del área de Dirección General tendrá la obligación de enviar la respuesta de las solicitudes aprobadas con su respectiva correspondencia en la sección de notas y fecha de autorización para su procedimiento.

Cada movimiento realizado en la gestión de Solicitudes de la Dirección se registrará en la bitácora, permitiendo realizar las observaciones en los periodos de Auditoría de Hospital, además de proyectar un gráfico de las solicitudes en proceso, autorizadas, y canceladas.

La organización de los privilegios permitirá brindar seguridad en los movimientos y los registros en la base de datos.

La base de datos se auto guardará realizando una copia de seguridad de la información, siendo importante y relevante para el resguardo de los datos.

Usuarios y privilegios

El acceso al servicio MySQL está controlado por usuarios y privilegios, los usuarios del servidor MySQL no tienen ninguna correspondencia con los usuarios del sistema operativo, aunque en la práctica común que algún administrador de MySQL asigne los mismos nombres que los usuarios tienen en el sistema, son mecanismos totalmente independientes y salen ser aconsejable en general.

El usuario administrador del sistema MySQL se llama **root**. Igual que el super usuario de los sistemas de UNIX.

Además del usuario **root** las instalaciones nuevas de MySQL incluyen el usuario anónimo, que tiene permisos sobre la base de datos test. Si queremos, también podemos restringirlos asignándole una contraseña. El usuario anónimo de MySQL se representa por una cadena vacía.

Para asignar contraseña a un usuario desde el cliente de MySQL se realiza de la siguiente manera:

```
ALTER USER ''@'localhost' IDENTIFIED WITH mysql_native_password BY 'nuevopassword';
```

La sentencia SQL que proporcioné es un comando **ALTER USER**, que se utiliza para modificar las propiedades de un usuario en una base de datos MySQL.

''@'localhost': Esto especifica el nombre de usuario y la ubicación desde donde se está accediendo a la base de datos. En este caso, '' representa un nombre de usuario vacío, lo que significa que se está modificando un usuario sin nombre (posiblemente un usuario anónimo o un usuario con un nombre de usuario en blanco). 'localhost' indica que esta modificación se aplica solo para conexiones que provienen del mismo servidor donde se encuentra la base de datos MySQL.

IDENTIFIED WITH mysql_native_password: Esta parte de la sentencia indica el método de autenticación que se utilizará para este **usuario**. **mysql_native_password** es uno de los métodos de autenticación disponibles en MySQL y es el método tradicional de autenticación de contraseñas en MySQL.

BY 'nuevopassword': Aquí especificamos la nueva contraseña que queremos asignar al usuario. **'nuevopassword'** debe reemplazarse con la nueva contraseña que desees establecer.

La administración de privilegios y usuarios en MySQL se realiza a través de las sentencias:

- **Grant**: Otorga privilegios a un usuario, en caso de no existir, se creará el usuario.
- **Revoke**: Elimina los privilegios de un usuario existente.
- **Set Password**: Asigna una contraseña.
- **Drop user**: Elimina un usuario.

La sintaxis simplificada de **Grant** conta de tres secciones, no puede omitirse ninguna. Y es importante el orden de la misma.

```
GRANT UPDATE, INSERT, SELECT ON api_aprobacionesservicios TO 'hospital_backend'@'localhost';
```

En la primera línea se especifica los privilegios que se otorgaran, en este caso se permite actualizar (**update**), insertar (**insert**) y consultar (**select**), la siguiente expresión especifica que los privilegios se aplican a la tabla **api_aprobacionesservicios** de la base de datos **hospital_backend**, en la última línea se encuentra el nombre del usuario y el equipo desde el que se va a permitir la conexión, siendo la correspondencia al **usuario** root y la **conexión local**.

El comando **Grant** crea la cuenta si no existe y, si existe, agrega los privilegios especificados. Es posible asignar una contraseña a la cuenta al mismo tiempo que se crea y le otorga privilegios.

Crear el usuario con la contraseña: Utilizamos la sentencia **CREATE USER** para crear un nuevo usuario en la base de datos. En este caso, estamos creando un usuario llamado '**Dr_Hector_Alvarado_Moreno**' que se conectará desde **localhost** y le asignamos la contraseña '**1234**'. La sintaxis es la siguiente:

```
CREATE USER 'Dr_Hector_Alvarado_Moreno'@'localhost' IDENTIFIED BY '1234';
```

Esta sentencia crea un nuevo usuario con el nombre '**Dr_Hector_Alvarado_Moreno**' y establece su contraseña como '**1234**'.

Otorgar permisos al usuario: Utilizamos la sentencia **GRANT** para otorgar permisos al usuario recién creado. En este caso, estamos otorgando permisos de **UPDATE, INSERT, y SELECT** en todas las tablas dentro de la base de datos **api_aprobacionesservicios**. La sintaxis es la siguiente:

```
GRANT UPDATE, INSERT, SELECT ON api_aprobacionesservicios.* TO 'Dr_Hector_Alvarado_Moreno'@'localhost';
```

Esta sentencia otorga permisos de **UPDATE, INSERT, y SELECT** al usuario '**Dr_Hector_Alvarado_Moreno**' para todas las tablas dentro de la base de datos **api_aprobacionesservicios**.

Especificaciones de lugares origen de la conexión

MySQL proporciona mecanismo para permitir que el usuario realice su conexión desde diferentes equipos dentro de una red específica, solo desde un, o únicamente desde el proveedor del propio servicio.

```
GRANT UPDATE, INSERT, SELECT ON hospital_backend.api_aprobacionesservicios TO 'Dr_Hector_Alvarado_Moreno'@'integradora-hospital-2024.online';
```

GRANT UPDATE, INSERT, SELECT: Esta parte de la sentencia especifica los permisos que se están otorgando al usuario. En este caso, se están otorgando los permisos de **UPDATE, INSERT y SELECT**, lo que permite al usuario realizar operaciones de actualización, inserción y selección de datos en la tabla especificada.

ON hospital_backend.api_aprobacionesservicios: Aquí se especifica la base de datos y la tabla a la que se están otorgando los permisos. **hospital_backend** es el nombre de la base de datos y **api_aprobacionesservicios** es el nombre de la tabla dentro de esa base de datos. Esto asegura que

los permisos se apliquen específicamente a la tabla **api_aprobacionesservicios** dentro de la base de datos **hospital_backend**.

TO 'Dr_Hector_Alvarado_Moreno'@'integradora-hospital-2024.online': Esta parte indica el usuario al que se le están otorgando los permisos y desde dónde puede conectarse. **'Dr_Hector_Alvarado_Moreno'** es el nombre de usuario y **integradora-hospital-2024.online** es el host desde donde se permite la conexión. Esto asegura que los permisos se apliquen solo cuando el usuario **Dr_Hector_Alvarado_Moreno** se conecte desde el host **integradora-hospital-2024.online**.

Especificaciones de base de datos y tablas

En la siguiente sentencia se otorga privilegios sobre todas las tablas de la base de datos **hospital_backend**.

```
grant all on hospital_backend.* to 'Dr_Hector_Alvarado_Moreno'@'integradora-hospital-2024.online'
```

De igual modo al especificar el nombre de una tabla se interpretará que pertenece a la base de datos en uso:

```
use hospital_backend;  
grant all on api_aprobacionesservicios to 'Dr_Hector_Alvarado_Moreno'@'integradora-hospital-2024.online';
```

Para otorgar permisos UPDATE, INSERT, SELEC y DROP para las tablas que se utilizaran en la base de datos se ejemplifica de la siguiente manera:

```
GRANT UPDATE, INSERT, SELECT, DROP ON hospital_backend.api_aprobacionesservicios TO 'Dr_Hector_Alvarado_Moreno'@'integradora-hospital-2024.online';  
GRANT SELECT ON hospital_backend.api_bitacora TO 'Dr_Hector_Alvarado_Moreno'@'integradora-hospital-2024.online';  
GRANT UPDATE, INSERT, SELECT ON hospital_backend.api_servicioshospitalarios TO 'Dr_Hector_Alvarado_Moreno'@'integradora-hospital-2024.online';  
GRANT UPDATE, INSERT, SELECT ON hospital_backend.api_serviciosmedicos TO 'Dr_Hector_Alvarado_Moreno'@'integradora-hospital-2024.online';  
GRANT SELECT ON hospital_backend.areas_medicas TO 'Dr_Hector_Alvarado_Moreno'@'integradora-hospital-2024.online';  
GRANT SELECT ON hospital_backend.personal_medico TO 'Dr_Hector_Alvarado_Moreno'@'integradora-hospital-2024.online';  
GRANT SELECT ON hospital_backend.personas TO 'Dr_Hector_Alvarado_Moreno'@'integradora-hospital-2024.online';
```

Especificaciones de columnas

Si se desea se puede otorgar privilegios de inserción, actualización, eliminación a las tablas de la siguiente manera.

```
GRANT  
    UPDATE, INSERT, SELECT (estatus),  
    UPDATE, INSERT, SELECT (comentarios),  
    UPDATE(fecha_aprobacion)  
ON hospital_backend.api_aprobacionesservicios  
TO 'Dr_Hector_Alvarado_Moreno'@'integradora-hospital-2024.online';
```

- Se están otorgando permisos de **UPDATE, INSERT y SELECT** en la columna **estatus** de la tabla **api_aprobacionesservicios**. Esto permite al usuario especificado actualizar, insertar y seleccionar datos en la columna **estatus**.

- Se están otorgando permisos de **UPDATE, INSERT y SELECT** en la columna **comentarios** de la tabla **api_aprobacionesservicios**. Esto permite al usuario especificado actualizar, insertar y seleccionar datos en la columna **comentarios**.
- Se está otorgando permiso solo de **UPDATE** en la columna **fecha_aprobacion** de la tabla **api_aprobacionesservicios**. Esto significa que el usuario especificado puede actualizar los valores de la columna **fecha_aprobacion**, pero no puede insertar ni seleccionar datos en esta columna.

Privilegios a otros usuarios y permisos

En entornos grandes, es frecuente encontrarse en necesidad de delegar el trabajo de administrar un servicio de bases de datos para que otros usuarios, además del administrador, pueda responsabilizarse de otorgar privilegios sobre una base de datos particular. Esto se puede hacer en MySQL con privilegio **gran all** :

```
GRANT ALL ON hospital_backend.api_aprobacionesservicios TO "Departamento_transplantes"@'integradora-hospital-2024.online';
```

GRANT ALL: Esta parte de la sentencia otorga todos los permisos disponibles en la tabla especificada.

ON hospital_backend.api_aprobacionesservicios: Indica la base de datos y la tabla a la que se otorgan los permisos.

TO "Departamento_transplantes"@'integradora-hospital-2024.online': Especifica el usuario y la ubicación desde donde se está accediendo a la base de datos. Aquí, se otorgan los permisos al usuario **"Departamento_transplantes"** cuando se conecta desde el host **integradora-hospital-2024.online**.

Esta sentencia **GRANT** permite al usuario **"Departamento_transplantes"** realizar todas las operaciones disponibles en la tabla **api_aprobacionesservicios** dentro de la base de datos **hospital_backend**, cuando se conecta desde el host **integradora-hospital-2024.online**.

Para permisos específicos a los médicos responsables del área correspondiente para realizar solicitudes para su aprobación del área de Dirección General se estructura de la siguiente manera

```
GRANT INSERT, SELECT ON hospital_backend.api_aprobacionesservicios TO "Medico_Encargado_transplantes"@'integradora-hospital-2024.online';
GRANT INSERT, SELECT ON hospital_backend.api_aprobacionesservicios TO "Medico_Encargado_farmacia"@'integradora-hospital-2024.online';
GRANT INSERT, SELECT ON hospital_backend.api_aprobacionesservicios TO "Medico_Encargado_pediatria"@'integradora-hospital-2024.online';
GRANT INSERT, SELECT ON hospital_backend.api_aprobacionesservicios TO "Medico_Encargado_farmacia"@'Programacion-hospital-2024.online';
GRANT INSERT, SELECT ON hospital_backend.api_aprobacionesservicios TO "Medico_Encargado_Programacion"@'integradora-hospital-2024.online';
GRANT INSERT, SELECT ON hospital_backend.api_aprobacionesservicios TO "Medico_Encargado_Radiologia"@'integradora-hospital-2024.online';
GRANT INSERT, SELECT ON hospital_backend.api_aprobacionesservicios TO "Medico_Encargado_RecursosHumanos"@'integradora-hospital-2024.online';
GRANT INSERT, SELECT ON hospital_backend.api_aprobacionesservicios TO "Medico_Encargado_RegistrosMedicos"@'integradora-hospital-2024.online';
GRANT INSERT, SELECT ON hospital_backend.api_aprobacionesservicios TO "Medico_Encargado_Transplantes"@'integradora-hospital-2024.online';
```

1. Esta sentencia otorga permisos de inserción (**INSERT**) y selección (**SELECT**) en la tabla **api_aprobacionesservicios** dentro de la base de datos **hospital_backend** al usuario **"Medico_Encargado_transplantes"** cuando se conecta desde el host **integradora-hospital-2024.online**.

2. Similar al anterior, esta sentencia otorga permisos de inserción y selección en la misma tabla a otro usuario, "**Medico_Encargado_farmacia**", desde el mismo host integradora-hospital-2024.online.
3. Aquí, se otorgan los mismos permisos a otro usuario, "**Medico_Encargado_farmacia**", pero desde un host diferente, Programacion-hospital-2024.online
4. Similar a la anterior, pero para el usuario "**Medico_Encargado_Programacion**" y desde el host integradora-hospital-2024.online.
5. Otro usuario, "**Medico_Encargado_Radiologia**", con los mismos permisos, desde el mismo host.
6. Para el usuario "**Medico_Encargado_RecursosHumanos**" desde el host integradora-hospital-2024.online.
7. Para el usuario "**Medico_Encargado_RegistrosMedicos**" desde el host integradora-hospital-2024.online.
8. Y finalmente, para el usuario "**Medico_Encargado_Transplantes**" desde el mismo host.

Cada una de estas sentencias otorga permisos de inserción y selección en la tabla **api_aprobacionesservicios** dentro de la base de datos **hospital_backend** a usuarios específicos desde hosts específicos. Esto proporciona un control granular sobre quién puede acceder y realizar qué tipo de operaciones en esa tabla.

Eliminar privilegios

El comando **revoke** permite eliminara privilegios otorgados con **grant** a los usuarios

```
REVOKE INSERT, SELECT FROM "Medico_Encargado_Temporal_transplantes"@"integradora-hospital-2024.online";
```

La sentencia **REVOKE** revoca los privilegios de inserción (**INSERT**) y selección (**SELECT**) para el usuario "**Medico_Encargado_Temporal_transplantes**" cuando se conecta desde el host '**integradora-hospital-2024.online**'.

REVOKE INSERT, SELECT: Esta parte de la sentencia indica los privilegios que se están revocando. En este caso, se están revocando los privilegios de inserción y selección.

FROM "Medico_Encargado_Temporal_transplantes"@"integradora-hospital-2024.online": Aquí se especifica el usuario y el host desde los cuales se están revocando los privilegios. El usuario "**Medico_Encargado_Temporal_transplantes**" se conecta desde el host '**integradora-hospital-2024.online**'.

Después de ejecutar esta sentencia, el usuario "**Medico_Encargado_Temporal_transplantes**" ya no tendrá los privilegios de inserción y selección en las bases de datos y tablas correspondientes cuando se conecte desde el host especificado.

Elimina usuarios

Antes de proceder a la eliminación de usuarios, es necesario asegurarse de que se le han quitado primero todos sus privilegios. Una vez asegurado este detalle, se procede a eliminar mediante el comando **drop user**;

```
drop user 'Medico_Encargado_Temporal_transplantes';
```

La sentencia **DROP USER** elimina un usuario de MySQL, lo que significa que ya no podrá iniciar sesión en el servidor MySQL. Aquí está la explicación línea por línea:

DROP USER: Esta parte de la sentencia indica que se está eliminando un usuario.

'Medico_Encargado_Temporal_transplantes': Este es el nombre del usuario que se desea eliminar. Se especifica entre comillas simples.

Después de ejecutar esta sentencia, el usuario **'Medico_Encargado_Temporal_transplantes'** será eliminado del servidor MySQL y no podrá iniciar sesión en él. Todos los privilegios y configuraciones asociados con este usuario también serán eliminados. Es importante tener cuidado al usar esta sentencia, ya que no se puede deshacer y la eliminación de un usuario podría tener consecuencias no deseadas.

Copias de Seguridad

Usando mysqldump: mysqldump es una herramienta de línea de comandos proporcionada por MySQL que te permite realizar copias de seguridad de bases de datos MySQL. Puedes usarla de la siguiente manera para realizar una copia de seguridad de una base de datos:

```
mysqldump -u usuario -p hospital_backend > Respaldo_Seguridad.sql
```

Esto creará un archivo **.sql** que contendrá todas las instrucciones SQL necesarias para recrear la base de datos y sus datos. Al ejecutar este comando, se te pedirá la contraseña del usuario especificado. Una vez ingresada, la copia de seguridad se creará.

Usando herramientas de administración: Muchas herramientas de administración de bases de datos, como phpMyAdmin y MySQL Workbench, ofrecen formas intuitivas de realizar copias de seguridad de bases de datos a través de interfaces gráficas de usuario. En general, estas herramientas te permiten seleccionar la base de datos que deseas respaldar y elegir opciones como la ubicación del archivo de copia de seguridad y el formato del archivo.

Usando almacenamiento de instantáneas (snapshot): Algunos sistemas de almacenamiento, como Amazon RDS para MySQL, ofrecen la capacidad de crear instantáneas de almacenamiento de bases de datos. Esto es particularmente útil en entornos en la nube, donde puedes crear instantáneas de tu base de datos sin detenerla, lo que garantiza la consistencia de los datos.

Copias de Seguridad Periódica

Para automatizar la creación periódica de copias de seguridad en MySQL, puedes utilizar herramientas de programación de tareas como cron en sistemas basados en Unix/Linux o Programador de tareas en sistemas Windows.

En sistemas basados en Unix/Linux (usando cron):

Abre una terminal y ejecuta el comando **crontab -e** para editar las tareas cron.

Agrega una nueva línea al archivo cron para programar la ejecución del comando **mysqldump** para crear la copia de seguridad en el intervalo deseado. Por ejemplo, para realizar una copia de seguridad todos los días a las 2:00 a.m., la línea puede verse así:

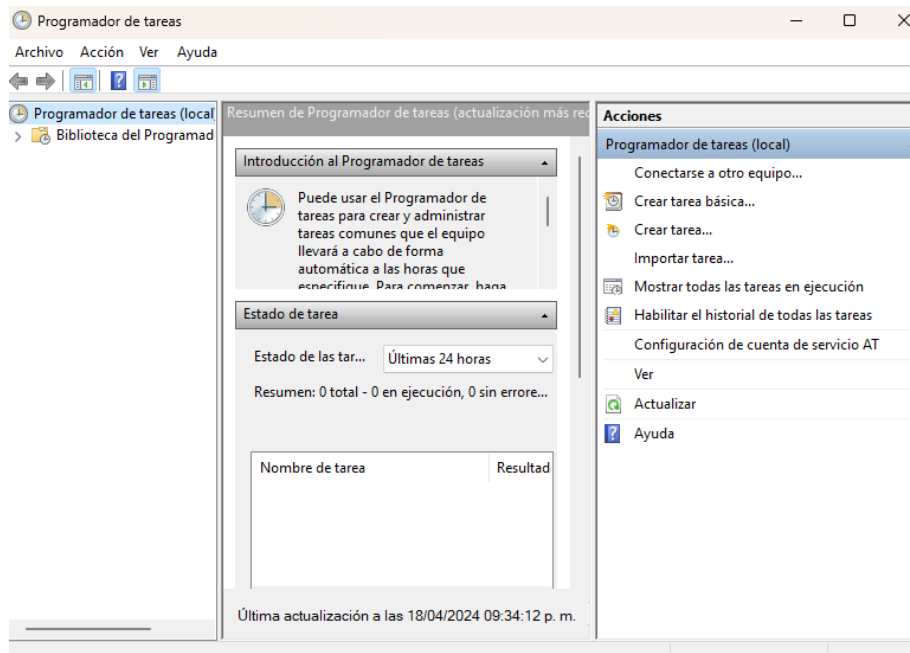
```
>0 2 * * * mysqldump -u root -p 1234 hospital_backend > /Administrador/Registros/BD/Respaldo_Direccion_General_$(date +%Y%m%d_%H%M%S).sql
```

Esto creará una copia de seguridad diaria con una marca de tiempo en el nombre del archivo.

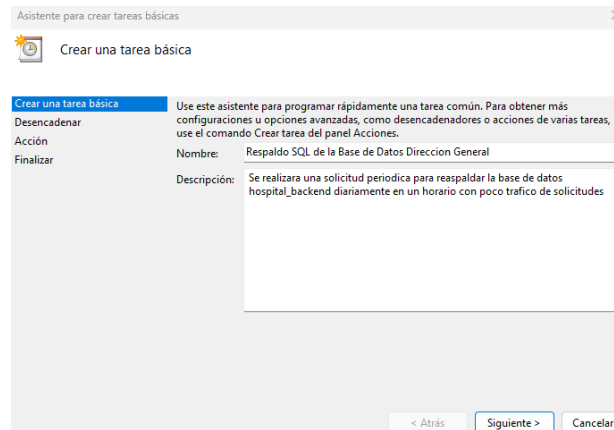
Guarda y cierra el archivo cron. Las copias de seguridad se generarán automáticamente según la programación especificada.

Programador de Respaldo en Windows

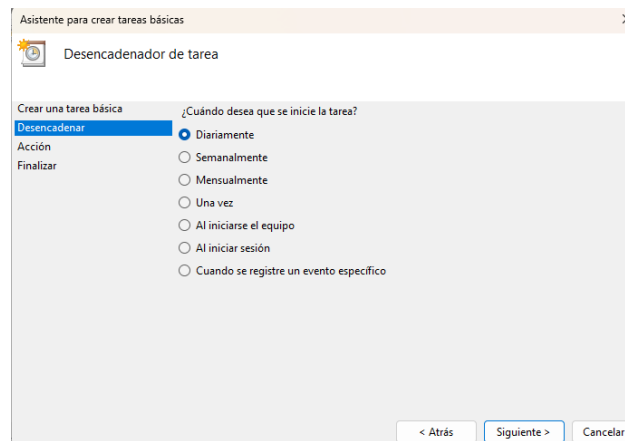
1. Abre el "Programador de tareas" desde el Panel de control o buscándolo en el menú de inicio.



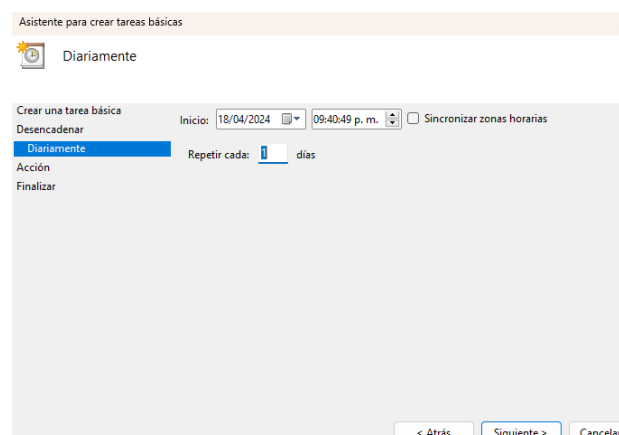
- Haz clic en "Crear tarea básica" en el panel derecho y sigue el asistente para crear una nueva tarea.



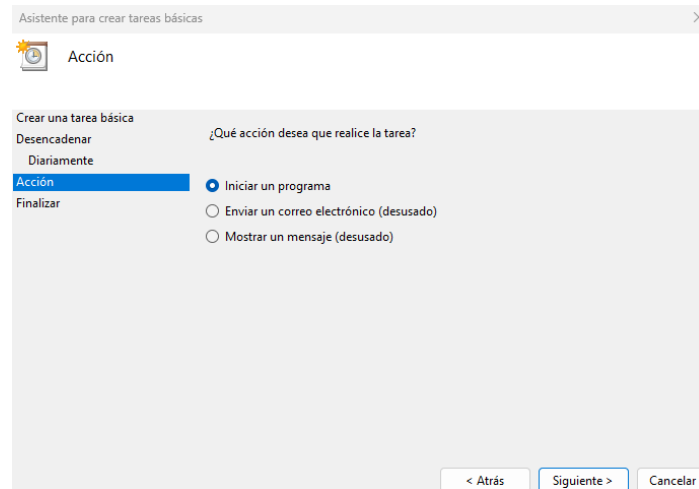
- Para realizar respaldo periódico se selecciona Diariamente:



- Se selecciona el horario de activación de la tarea que permitirá ejecutar el respaldo de la base de datos:



- Se Activa la Acción para que inicie un programa:



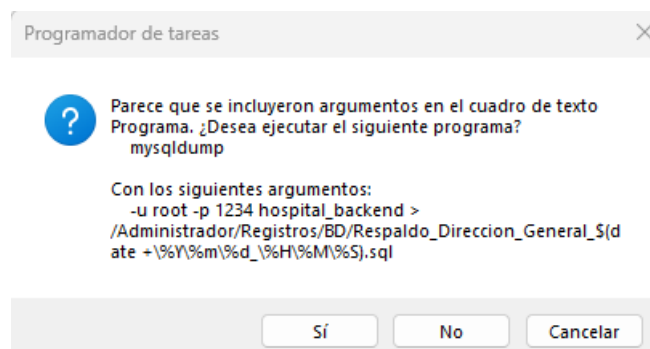
Se asigna la ruta donde se guardara la tarea, para iniciar `mysqldump` y realizar una copia de seguridad de tu base de datos `hospital_backend` con una contraseña específica desde la línea de comandos en Unix/Linux, puedes ejecutar el siguiente comando:

```
mysqldump -u root -p 1234 hospital_backend > /Administrador/Registros/BD/Respaldo_Direccion_General_$(date +%Y%m%d_%H%M%S).sql
```

Este comando realizará lo siguiente:

- mysqldump:** Inicia el comando **mysqldump** para realizar la copia de seguridad.
- u root:** Especifica el nombre de usuario de MySQL (**root** en este caso).
- p1234:** Especifica la contraseña de MySQL (**1234** en este caso). Nota que no hay espacio entre **-p** y la contraseña.
- hospital_backend:** Especifica el nombre de la base de datos que deseas respaldar.
- /Administrador/Registros/BD/Respaldo_Direccion_General_\$(date +%Y%m%d_%H%M%S).sql:** Redirige la salida de **mysqldump** a un archivo **.sql** en la ubicación especificada. El nombre del archivo incluye una marca de tiempo para que cada copia de seguridad tenga un nombre único y se guarde en el directorio **/Administrador/Registros/BD/**.

- Se acepta la automatización:



- Se Finaliza la Automatización en el programador de tareas de Windows:

Asistente para crear tareas básicas

Resumen

Crear una tarea básica

Desencadenar: **Diariamente**

Nombre: **Respaldo SQL de la Base de Datos Direccion General**

Descripción: **Se realizara una solicitud periodica para respaldar la base de datos hospital_backend diariamente en un horario con poco trafico de solicitudes**

Acción: **Iniciar un programa**

Finalizar

Desencadenador: **Diariamente; A las 09:40 p. m. todos los días**

Acción: **Iniciar un programa; mysqldump -u root -p 1234 hospital_backend > /A**

☐ Abrir el diálogo Propiedades para esta tarea al hacer clic en Finalizar

Al hacer clic en Finalizar, la nueva tarea se creará y se agregará a su programación de Windows.

< Atrás Finalizar Cancelar

- Para verificar que se agrego la tareas consultamos:

Tareas activas

Tareas activas son tareas habilitadas en este momento y que no expiraron.

Resumen: 154 en total

Nombre de tarea	Hora próxima ejecución	Desencadenadores	Ubicación
NvTmRep_CrashReport2_{B2FE1952-0186-46C3-BAEC-A80AA3...}	19/04/2024 06:25:59 p. ...	A las 06:25 p. m. todos los días	\
ASUSUpdateTaskMachineCore	19/04/2024 07:05:31 p. ...	Se definieron varios desencadenadores	ASUS
Respaldo SQL de la Base de Datos Direccion General	19/04/2024 09:40:49 p. ...	A las 09:40 p. m. todos los días	\
MicrosoftEdgeUpdateTaskMachineCore	19/04/2024 10:16:09 p. ...	Se definieron varios desencadenadores	\
BackupNonMaintenance	20/04/2024 12:44:31 a. ...	A las 12:00 a. m. cada 14 días	Microsoft\Windows\Ap...
ScanForUpdates	20/04/2024 12:51:29 a. ...	Se definieron varios desencadenadores	Microsoft\Windows\Inst...
IntegrityCheck	19/04/2024 01:51:40 p. ...	A las 12:00 p. m. el 01/01/2015 - Tras des...	Microsoft\Windows\De...
Backup	21/04/2024 06:50:57 a. ...	Se definieron varios desencadenadores	Microsoft\Windows\Clo...
PerformRemediation	21/04/2024 05:55:12 a. ...	A las 03:00 a. m. el 15/10/2000 - Tras des...	Microsoft\Windows\Wa...

Última actualización a las 18/04/2024 09:34:12 p. m.

Actualizar

Base de Datos No SQL Mongo DB

Administrar usuarios y roles es una parte crucial de la administración de cualquier base de datos, y MongoDB no es una excepción. Este capítulo de nuestro curso cubrirá cómo crear, administrar y asignar roles a usuarios en MongoDB.

En MongoDB, la autenticación y la autorización son las dos formas principales de gestionar la seguridad. La autenticación verifica la identidad de un usuario, mientras que la autorización determina qué acciones puede realizar un usuario. En otras palabras, la autenticación es el proceso de verificar quién es usted, mientras que la autorización es el proceso de verificar lo que tiene permitido hacer.

Para administrar usuarios y roles en MongoDB, debe comprender cómo MongoDB maneja la seguridad. MongoDB utiliza un modelo de seguridad basado en roles, lo que significa que puede asignar roles específicos a usuarios específicos. Cada rol tiene un conjunto específico de privilegios que determinan lo que el usuario puede hacer.

Para crear un nuevo usuario en MongoDB, puede utilizar el comando **db.createUser()**. Este comando crea un nuevo usuario y le asigna un rol. Por ejemplo, el siguiente comando crea un nuevo usuario llamado "Dr.Hector_Alvarado" con la contraseña "30711" y el rol "readWrite":

```
test> db.createUser(  
  {  
    usuario: "Dr.Hector_Alvarado",  
    pwd: "30711",  
    roles: [ "readWrite" ]  
  }  
)
```

Este comando crea un nuevo usuario que tiene permiso para leer y escribir datos. Sin embargo, este usuario no tiene permiso para realizar tareas administrativas como crear nuevas bases de datos o administrar usuarios.

Si desea cambiar la función de un usuario, puede utilizar el comando **db.updateUser()**. Este comando actualiza la función de un usuario existente. Por ejemplo, el siguiente comando actualiza la función de usuario "myUser" a "dbAdmin":

```
test> db.updateUser(  
  "Dr.Hector_Alvarado",  
  {  
    roles: ["dbAdmin"]  
  }  
)
```

Este comando actualiza la función del usuario "**Dr.Hector_Alvarado**" a "**dbAdmin**", lo que significa que este usuario ahora tiene permiso para realizar tareas administrativas.

Además, MongoDB también le permite crear sus propias funciones personalizadas. Para crear una nueva función, puede utilizar el comando **db.createRole()**. Por ejemplo, el siguiente comando crea una nueva función llamada "**readWriteAndAdmin**":

```
test> db.createRole(
  {
    rol: "readWriteAndAdmin",
    privilegios: [
      {
        recurso: { db: "DireGeneral", colección: "pacientesareas" },
        acciones: [ "buscar", "actualizar", "insertar", "eliminar", "createCollection", "dropCollection", "createIndex", "dropIndex", "viewOn", "collStats", "dbStats", "dbHash", "adminbd" ]
      }
    ],
    roles: []
  }
)
```

Este comando crea una nueva función que tiene permisos para leer, escribir y administrar la base de datos "**DireGeneral**".

En resumen, administrar usuarios y roles en MongoDB es una tarea importante que ayuda a garantizar la seguridad de su base de datos. Al comprender cómo crear, administrar y asignar roles a los usuarios, puede asegurarse de que cada usuario tenga los privilegios adecuados para realizar sus tareas.

Copia de Seguridad

mongodumpOperaciones básicas

La mongodump utilidad realiza una copia de seguridad de los datos conectándose a una instancia mongod en ejecución mongos.

La utilidad puede crear una copia de seguridad para un servidor, base de datos o colección completos, o puede usar una consulta para hacer una copia de seguridad solo de una parte de una colección.

Cuando se ejecuta mongodump sin ningún argumento, el comando se conecta a la instancia de MongoDB en el sistema local (por ejemplo, 127.0.0.1 o localhost) en el puerto 27017 y crea una copia de seguridad de la base de datos denominada dump/ en el directorio actual.

Para hacer una copia de seguridad de los datos de una instancia mongod mongos que se ejecuta en la misma máquina y en el puerto predeterminado de 27017, use el siguiente comando

mongodump

El formato de datos utilizado por mongodump desde la versión 2.2 o posterior es incompatible con versiones anteriores de mongod. No utilice versiones recientes de mongodump para realizar copias de seguridad de almacenes de datos más antiguos.

También puede especificar el --host y --port de la instancia de MongoDB a la que mongodump debe conectarse.

Por ejemplo:

mongodump --host mongodb.example.net --puerto 27017

mongodump describirá archivos BSON que contienen una copia de los datos accesibles a través del puerto de escucha 27017 del mongodb.example.net host. Consulte Crear copias de seguridad de instancias mongod no locales para obtener más información.

Para especificar un directorio de salida diferente, puede usar la opción: --out or -o

Por ejemplo:

mongodump --collection miColección --db prueba

Esta operación crea un volcado de la colección nombrada myCollection de la base de datos test en un dump/subdirectorio del directorio de trabajo actual.

mongodump sobrescribe los archivos de salida si existen en la carpeta de datos de respaldo. Antes de ejecutar el mongodump comando varias veces, asegúrese de que ya no necesita los archivos en la carpeta de salida (el valor predeterminado es la dump/carpeta) o cambie el nombre de las carpetas o archivos.

Operación en un momento dado utilizando Oplogs

Utilice la --oplog opción con mongodump para recopilar las entradas del registro de operaciones para crear una instantánea de un momento dado de una base de datos dentro de un conjunto de réplicas. Con --oplog, mongodump copia todos los datos de la base de datos de origen, así como todas las entradas del registro de operaciones desde el principio hasta el final del procedimiento de copia de seguridad. Esta operación, junto con , le permite restaurar una copia de seguridad que refleja el momento específico que corresponde al momento en que se completó la creación del archivo de volcado. mongorestore --oplogReplay mongodump.

mongod Crear copias de seguridad desde instancias no locales

Las opciones --host le permiten conectarse y realizar copias de seguridad desde un host remoto. Considere el siguiente ejemplo: --port mongodump

mongodump --host mongodb1.example.net --puerto 3017 --nombre de usuario usuario -- contraseña pasar --out /opt/backup/mongodump-2013-10-24

En cualquier mongodump comando, como se indicó anteriormente, puede especificar las credenciales de nombre de usuario y contraseña para especificar la autenticación de la base de datos.