Proyecto Integrador Redis

Angel Fabrizio Franyutti Pulido
Manuel Antonio Hernández Maruri

06/06/2024
Miguel Ángel Ortigoza Clement

Universidad Veracruzana
Tecnologías computacionales
Bases de datos convencionales

Contenido

Introducción

En el día de hoy se busca una rápida respuesta en los sistemas, sistemas donde se pueda almacenar y recuperar los datos lo más rápido posible. Los programas de hoy utilizan un sistema de almacenamiento local llamado cache, “software que guarda datos para que las solicitudes futuras de esos datos se puedan atender con mayor rapidez” (Wikipedia, 2024). El caché se genera a partir de los datos consultados de una base de datos al ser procesados por un servicio.

Uno de los principales inconvenientes es que es local, no es posible tener el mismo caché en distintas computadoras por lo que al querer acceder a un sistema desde otro dispositivo, las credenciales no se pueden reutilizar; al acceder al sistema en otro dispositivo se tiene que almacenar los nuevos datos útiles para autenticación.

Tanto las consultas a una base de datos tanto relacional como no relacional, son relativamente lentas en comparación de hacer una consulta a datos en memoria.

Redis (Remote Directory Server) resuelve parte de este problema, redis almacena estructuras de datos de manera rápida y flexible. La estructura utilizada por redis son Hash Tables en los que se pueden almacenar datos no relacionados y son almacenados en memoria.

Desarrollo

Para el proyecto de la Experiencia Educativa de Bases De Datos No Convencionales decidimos realizar un pequeño reproductor de sonidos que utiliza redis para almacenar las sesiones y los usuarios.

Se utilizó Docker para crear las instancias principales como secundarias de redis y el framework Laravel para la lógica debido a que cuenta módulos para el manejo de distintas bases de datos, sesión y autenticación.

Creación de Web App

Como se mencionó anteriormente se utilizó el framework de Laravel en el lenguaje de programación de PHP, en el que se hizo la interfaz gráfica y la conexión a la base de datos utilizando el servidor de artisian.

El proceso fue:

En el desarrollo de esta aplicación web, comenzamos creando un archivo llamado "env" donde definimos las constantes necesarias para que el servidor se conecte correctamente, como el puerto, la dirección IP, el usuario y la contraseña para Redis.

Redis es una base de datos en memoria que elegimos por su alto rendimiento y capacidad de almacenar datos en caché. Luego, ejecutamos el servidor y verificamos que la conexión con Redis se estableció correctamente.

A continuación, creamos las vistas (interfaces gráficas) de la aplicación web, incluyendo formularios sencillos de inicio de sesión y registro de usuarios, así como una página principal donde se mostrarían y podrían reproducir, agregar o eliminar los audios registrados.

Desarrollamos los controladores necesarios para que las vistas funcionaran en conjunto con la base de datos. Primero, implementamos el registro de usuarios, donde cada usuario debía ingresar un nombre de usuario y una contraseña para acceder al sistema.

Una vez registrado, el usuario podría iniciar sesión con sus credenciales. Al hacerlo, su sesión se almacenaría en Redis. Luego, creamos el controlador de audios con funciones para agregar, visualizar y eliminar los audios almacenados en Redis en formato base 64 (una codificación que permite representar archivos binarios como texto).

Para agregar un audio, decidimos utilizar un modal y establecer un límite de tamaño de 1 MB debido a las limitaciones de Redis. La reproducción de los audios se realiza extrayendo los datos en milisegundos desde la base de datos para un acceso rápido.

Durante el desarrollo, enfrentamos el desafío de optimizar el almacenamiento y la recuperación de los archivos de audio en Redis, lo que nos llevó a implementar la codificación en base 64 y establecer el límite de tamaño mencionado.

En general, el proyecto involucró la integración de diferentes componentes, como la gestión de usuarios, el manejo de archivos multimedia y el uso de una base de datos en memoria para lograr un rendimiento óptimo.

Por último, se generaron las direcciones donde se encontrarán las views junto con sus controladores

Habilitar replicación

Para habilitar la replicación de redis utilizando Docker se deben hacer 3 cosas

- Crear archivos de configuración para cada instancia
- Actualizar las variables de ambiente para permitir el acceso la instancia principal
- Crear las imágenes y levantar el contenedor

Crear archivos de configuración:

El archivo de configuración para la instancia maestro:

bind 0.0.0.0

port 6379

requirepass redis_password1234

El archivo de configuración para la instancia esclavo:

bind 0.0.0.0

port 6379

replicaof redis-maestro 6379

masterauth redis_password1234

Actualizar variables de ambiente

En los archivos de configuración de la instancia maestro se especificó que se requiere una contraseña para acceder a los datos. Esta contraseña debe añadirse en las variables de ambiente en el archivo .env

REDIS_PASSWORD=redis_password1234

Crear y levantar las instancias

Es necesario crear una red de docker para que las instancias puedan comunicarse entre ellas.

docker network create redis-network

Una vez creada la red, se utiliza en el comando para crear y levantar la instancia maestro.

docker run -d --name redis-maestro --network redis-network -p 6379:6379 -v $pwd/redis/redis-maestro.conf:/usr/local/etc/redis/redis.conf redis:alpine redis-server /usr/local/etc/redis/redis.conf

Con este comando se crea un contenedor con el nombre redis-maestro, utilizando la red redis-network, exponiendo el puerto 6379 del contenedor en el puerto 6379 de la máquina local, montando el archivo redis-maestro.conf y lo ejecuta dentro del contenedor al iniciarse.

Docker solo acepta rutas absolutas para montar volúmenes, para facilitar y simular rutas relativas al proyecto, se utilizó el comando $pwd que en Windows imprime la ruta absoluta del directorio actual.

Se crea y levanta la instancia esclavo.

docker run -d --name redis-esclavo --network redis-network -v $pwd/redis/redis-esclavo.conf:/usr/local/etc/redis/redis.conf redis:alpine redis-server /usr/local/etc/redis/redis.conf

Con un funcionamiento similar, esta vez no expone los puertos porque es la instancia de replicación.

Conclusiones

Redis como sistema gestor de base de datos es potente e interesante de usar. Nos permitió mantener el registro de usuarios solamente en la memoria del servidor, también nos facilitó el almacenamiento de archivos de audio formateado en base64.

Es fácil de implementar con Laravel, cambiando el driver de conexión a base de datos y actualizando las variables de entorno.

No tener una estructura definida en el tipo de datos que deben tener todos los registros nos quitó una capa de dificultad al diseñar una base de datos, quedándonos solo con separar entre si era un usuario o un registro de archivo de audio.

Con esta experiencia quedamos abiertos a implementar no solo este tipo de motor de base de datos, sino bases de datos no relacionales.

Bibliografía

Wikipedia. (29 de Abril de 2024). *Wikipedia*. Obtenido de Caché (informática): https://es.wikipedia.org/wiki/Cach%C3%A9_(inform%C3%A1tica)