Universidad Autónoma del Estado de Hidalgo

ICBI

REPORTE DE PRÁCTICA NO. 2.1

Base de Datos Distribuida

ALUMNO: Jesús Eduardo Conrnejo-Clavel
Dr. Eduardo Cornejo-Velazquez

1. Introducción

Los sistemas de bases de datos distribuidas son esenciales en la informática moderna para garantizar la disponibilidad de datos, la tolerancia a fallos y la escalabilidad horizontal. Este proyecto utiliza Podman para gestionar contenedores de MongoDB y configura un conjunto de réplicas para demostrar los principios de bases de datos distribuidas. Además, se proporciona una API RESTful para interactuar con la base de datos, mostrando una aplicación práctica de la configuración.

3. Marco teórico

Para construir el marco teórico se consultó la siguiente bibliografía:

Bases de Datos Distribuidas

Una base de datos distribuida es aquella que almacena datos en múltiples ubicaciones físicas y gestiona el acceso y modificación de estos datos a través de una red. Según Özsu y Valduriez [1], las bases de datos distribuidas mejoran la disponibilidad y el rendimiento en sistemas a gran escala.

Replicación de Datos

La replicación es una técnica que implica mantener copias de los mismos datos en diferentes ubicaciones. Según Bernstein y Newcomer [2], la replicación mejora la disponibilidad de datos y la tolerancia a fallos.

Contenedores y Virtualización

Los contenedores proporcionan un entorno ligero y portable para ejecutar aplicaciones. Según Pahl [3], los contenedores facilitan la implementación y gestión de sistemas distribuidos.

MongoDB

MongoDB es una base de datos NoSQL orientada a documentos que proporciona alta escalabilidad y flexibilidad. Según Chodorow [4], MongoDB es adecuado para aplicaciones distribuidas debido a su modelo de datos flexible y sus capacidades de replicación.

3. Herramientas empleadas

Las herramientas utilizadas para desarrollar la práctica fueron:

1. **Podman**: Para gestionar contenedores Linux sin necesidad de privilegios de root.
2. **MongoDB**: Como sistema de base de datos NoSQL orientado a documentos.
3. **Rust**: Para implementar la API RESTful.
4. **OpenSSL**: Para generar claves de autenticación seguras.
5. **Bash**: Para automatizar la configuración del entorno.
6. **Ghostty con Neovim**: Como entorno de desarrollo integrado (IDE).
7. **Postman**: Para realizar llamadas a las APIs.

4. Desarrollo

Planteamiento del Problema

La gestión de grandes volúmenes de datos y la necesidad de alta disponibilidad y tolerancia a fallos son desafíos comunes en la informática moderna. Las bases de datos distribuidas ofrecen una solución a estos problemas, pero su configuración y gestión pueden ser complejas. Este proyecto aborda la implementación de una base de datos distribuida utilizando herramientas modernas como Podman y MongoDB.

Objetivo General

Implementar y demostrar una configuración de base de datos distribuida utilizando Podman y MongoDB, proporcionando una API RESTful para la interacción con la base de datos.

Objetivos Específicos

1. Configurar un conjunto de réplicas de MongoDB utilizando contenedores gestionados por Podman.
2. Proporcionar una API RESTful para interactuar con la base de datos distribuida.
3. Evaluar la escalabilidad y tolerancia a fallos de la configuración implementada.
4. Documentar el proceso de configuración y los resultados obtenidos.

Arquitectura del Sistema

En la Figura 2 se presenta la arquitectura propuesta para el sistema de base de datos distribuida.

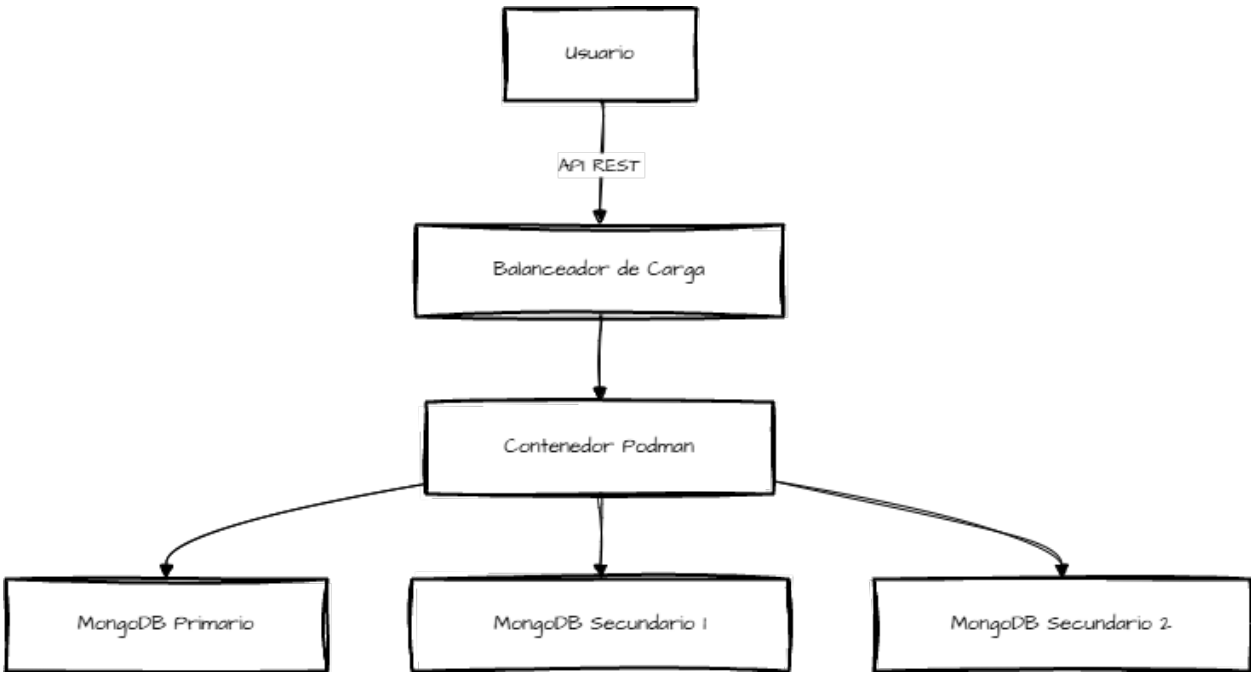

Figure 1: Arquitectura del sistema de base de datos distribuida.

Configuración del Entorno

En el Listado 1 se presentan las instrucciones para configurar el entorno.

Listing 1: Configuración del entorno

```
# Crea un archivo .env con la configuraci n de MongoDB
cat > .env << EOF
MONGODB_URI1=mongodb://mongo1:27017
MONGODB_URI2=mongodb://mongo2:27018
EOF

# Ejecuta el script de inicio
./start.sh
```

Implementación de la API RESTful

En el Listado 2 se presenta un ejemplo de implementación de la API RESTful en Rust.

Listing 2: Implementación de la API RESTful

```
// main.rs
use actix_web::{web, App, HttpServer, Responder};
use mongodb::{Client, options::ClientOptions};
use serde::{Deserialize, Serialize};
use std::env;

#[derive(Serialize, Deserialize)]
struct User {
    name: String,
    email: String,
}

async fn get_users() -> impl Responder {
    let client_uri = env::var("MONGODB_URI1").unwrap_or_else(|_| "mongodb://mongo1:27017".
    let mut client_options = ClientOptions::parse(&client_uri).await.unwrap();
    let client = Client::with_options(client_options).unwrap();
    let database = client.database("social_media");
    let collection = database.collection::<User>("users");
    let cursor = collection.find(None, None).await.unwrap();
    let users: Vec<_> = cursor.try_collect().await.unwrap();
    web::Json(users)
}

#[actix_web::main]
async fn main() -> std::io::Result<()> {
    HttpServer::new(|| {
        App::new()
            .route("/api/v1/users", web::get().to(get_users))
    })
    .bind("0.0.0.0:3000")?
    .run()
    .await
}
```

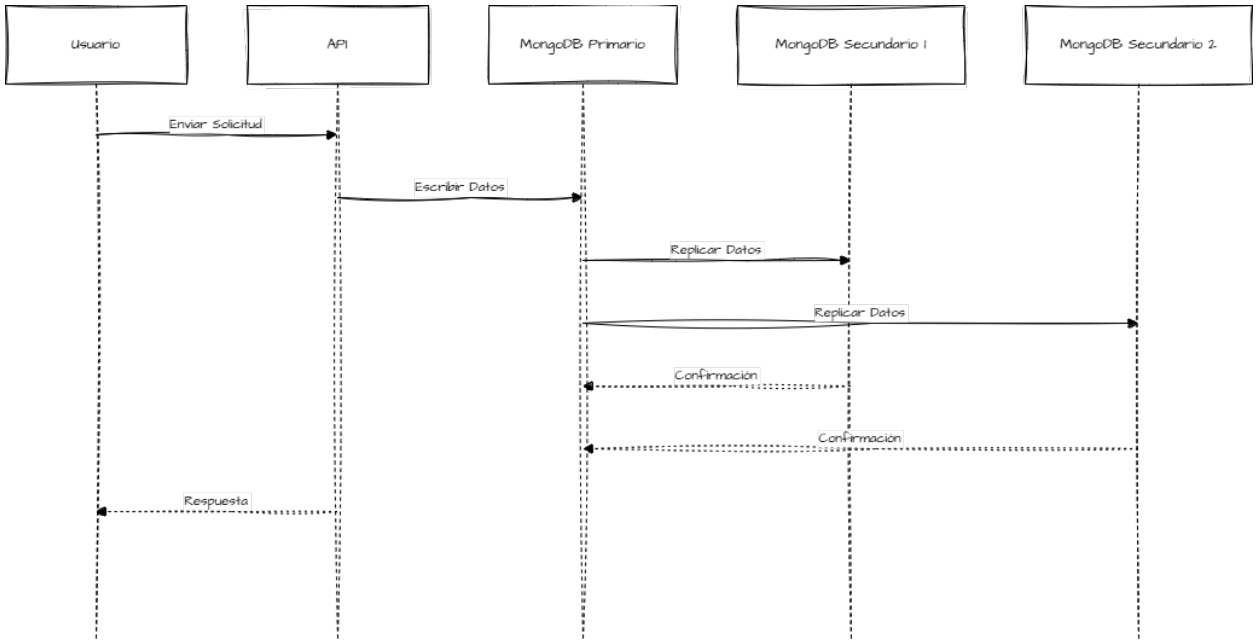

Figure 2: Diagrama de flujo del sistema.

5. Conclusiones

Este proyecto demuestra la aplicación práctica de los conceptos de bases de datos distribuidas utilizando Podman y MongoDB. Al configurar un conjunto de réplicas y proporcionar una API RESTful, se destacan los beneficios de escalabilidad, tolerancia a fallos y rendimiento. La implementación propuesta ofrece una solución robusta para la gestión de datos en entornos distribuidos.
Entre los principales logros del proyecto destacan:

- Configuración exitosa de un conjunto de réplicas de MongoDB en contenedores Podman.
- Implementación de una API RESTful funcional para interactuar con la base de datos distribuida.
- Demostración de la tolerancia a fallos mediante pruebas de desconexión de nodos.
- Documentación detallada del proceso de configuración y los resultados obtenidos.

El trabajo futuro podría implicar la implementación de particionamiento (sharding) para una mayor escalabilidad y la exploración de diferentes herramientas de orquestación de contenedores.

Referencias Bibliográficas

References

[1] Özsu, M. T.; Valduriez, P. (2011). *Principles of Distributed Database Systems*. Springer.

[2] Bernstein, P. A.; Newcomer, E. (2009). *Principles of Transaction Processing*. Morgan Kaufmann.

[3] Pahl, C. (2018). *Containerization and the PaaS Cloud*. IEEE Cloud Computing.

[4] Chodorow, K. (2013). *MongoDB: The Definitive Guide*. O'Reilly Media.