

# SIS313 Infraestructura, Plataformas Tecnológicas y Redes

## Proyecto de Segunda Instancia

---

Este proyecto documenta el diseño e implementación de una infraestructura de TI web escalable y funcional, siguiendo los requisitos del proyecto de segunda instancia. Se ha simulado un entorno de producción utilizando.

**cuatro máquinas virtuales** en Ubuntu Server 24.04 LTS.

La arquitectura se compone de un:

Rol del Servidor	Hostname (Ubuntu)	Software Instalado	Dirección IP Estática
<b>Servidor de Base de Datos</b>	db01	MySQL Server, mdadm	192.168.0.104
<b>Balanceador de Carga</b>	lb01	Nginx	192.168.0.101
<b>Servidor de Aplicación 1</b>	app01	Node.js, npm, express, mysql2	192.168.0.102
<b>Servidor de Aplicación 2</b>	app02	Node.js, npm, express, mysql2	192.168.0.103

balanceador de carga (Nginx) que distribuye el tráfico entre dos servidores de aplicación (Node.js), los cuales a su vez se comunican con un servidor de base de datos (MySQL) centralizado. El objetivo principal ha sido demostrar de forma práctica conceptos de balanceo de carga, la estructura de una aplicación distribuida, y la tolerancia a fallos a nivel de almacenamiento mediante la configuración de un **RAID 5**.

---

### Justificación de Tecnologías

**Plataforma de Virtualización (Oracle VirtualBox):** Se utilizó para simular la infraestructura de 4 servidores en un solo host físico, permitiendo un entorno de pruebas controlado y eficiente.

**Sistema Operativo (Ubuntu Server 24.04 LTS):** Se seleccionó esta versión por ser un requisito explícito del proyecto y por su estabilidad y amplio soporte como sistema operativo para servidores.

**Balanceador de Carga (Nginx):** Se eligió Nginx por su alto rendimiento y bajo consumo de recursos. Fue configurado con el algoritmo por defecto

**Round Robin** para distribuir equitativamente las peticiones entre los dos servidores de aplicación.

**Aplicación Web (Node.js + Express):** Se utilizó Node.js junto con el framework Express para desarrollar una API RESTful con funcionalidad CRUD, cumpliendo con los requisitos de la aplicación de prueba.

**Base de Datos (MySQL):** Se implementó MySQL como sistema gestor de base de datos relacional, una de las opciones recomendadas, para garantizar la persistencia de los datos de la aplicación.

**Esquema de Red (IPs Estáticas):** Se configuró una dirección IP estática en cada servidor para asegurar una comunicación estable y predecible entre los componentes (Balanceador -> Apps, Apps -> Base de Datos), lo cual es fundamental para el correcto funcionamiento del sistema.

**Almacenamiento (RAID 5):** Se implementó un arreglo RAID 5 en el servidor de base de datos para proveer tolerancia a fallos a nivel de disco. Esta configuración protege los datos contra la falla de una de las unidades de disco.

---

## **Hardening Aplicado**

Se implementaron las siguientes medidas de hardening básico para asegurar los servicios críticos y reducir la superficie de ataque de la infraestructura:

**Configuración de Firewall (UFW):** Se utilizó Uncomplicated Firewall (UFW) en los servidores para restringir el acceso únicamente a los puertos necesarios para la operación de cada servicio.

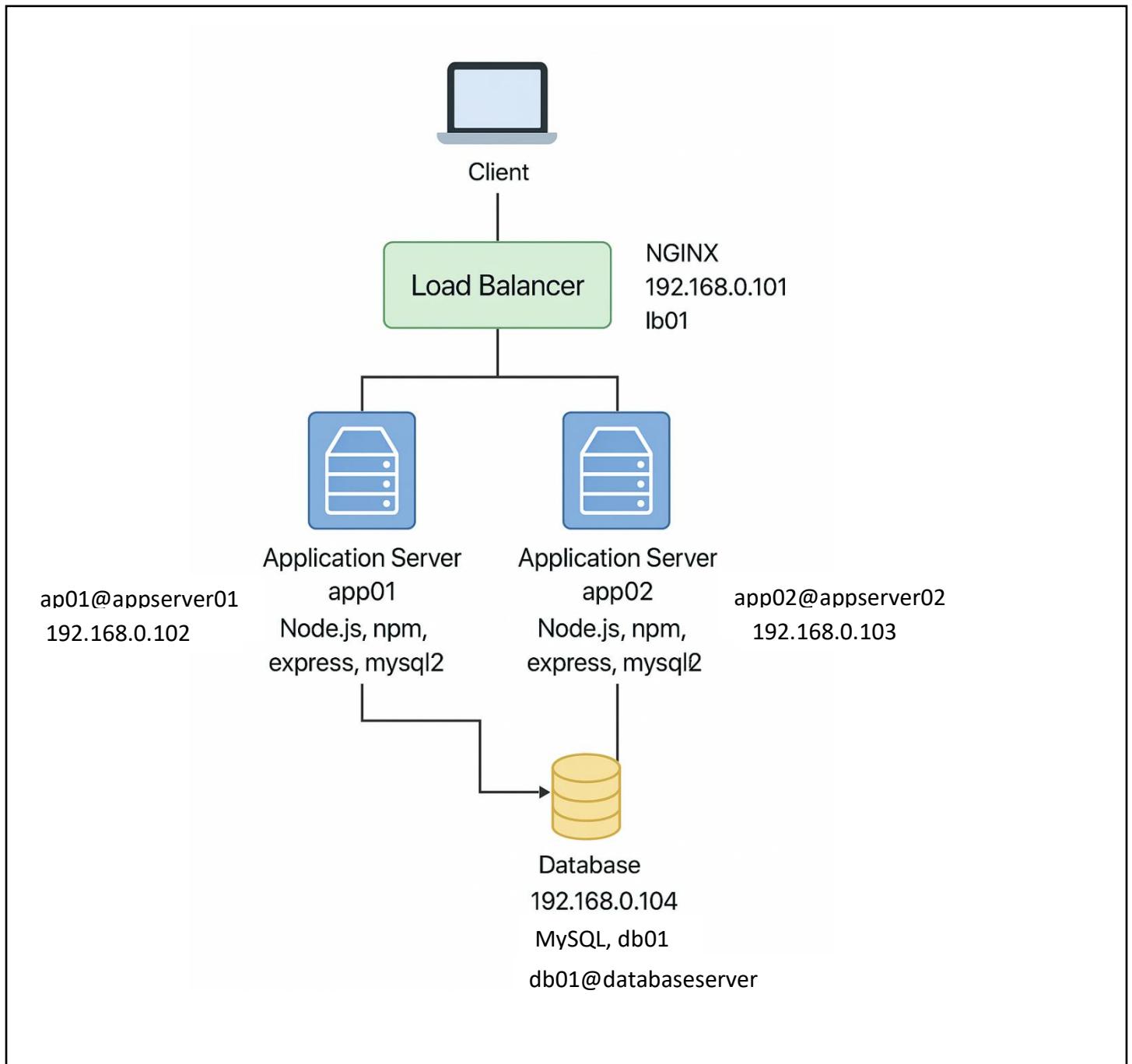
**En el Balanceador (lb01):** Se permitió el tráfico entrante en el puerto 80 (HTTP) para recibir las peticiones de los clientes.

**En los Servidores de Aplicación (app01, app02):** Se permitió el tráfico entrante en el puerto 3000, que es el puerto donde escucha la aplicación Node.js.

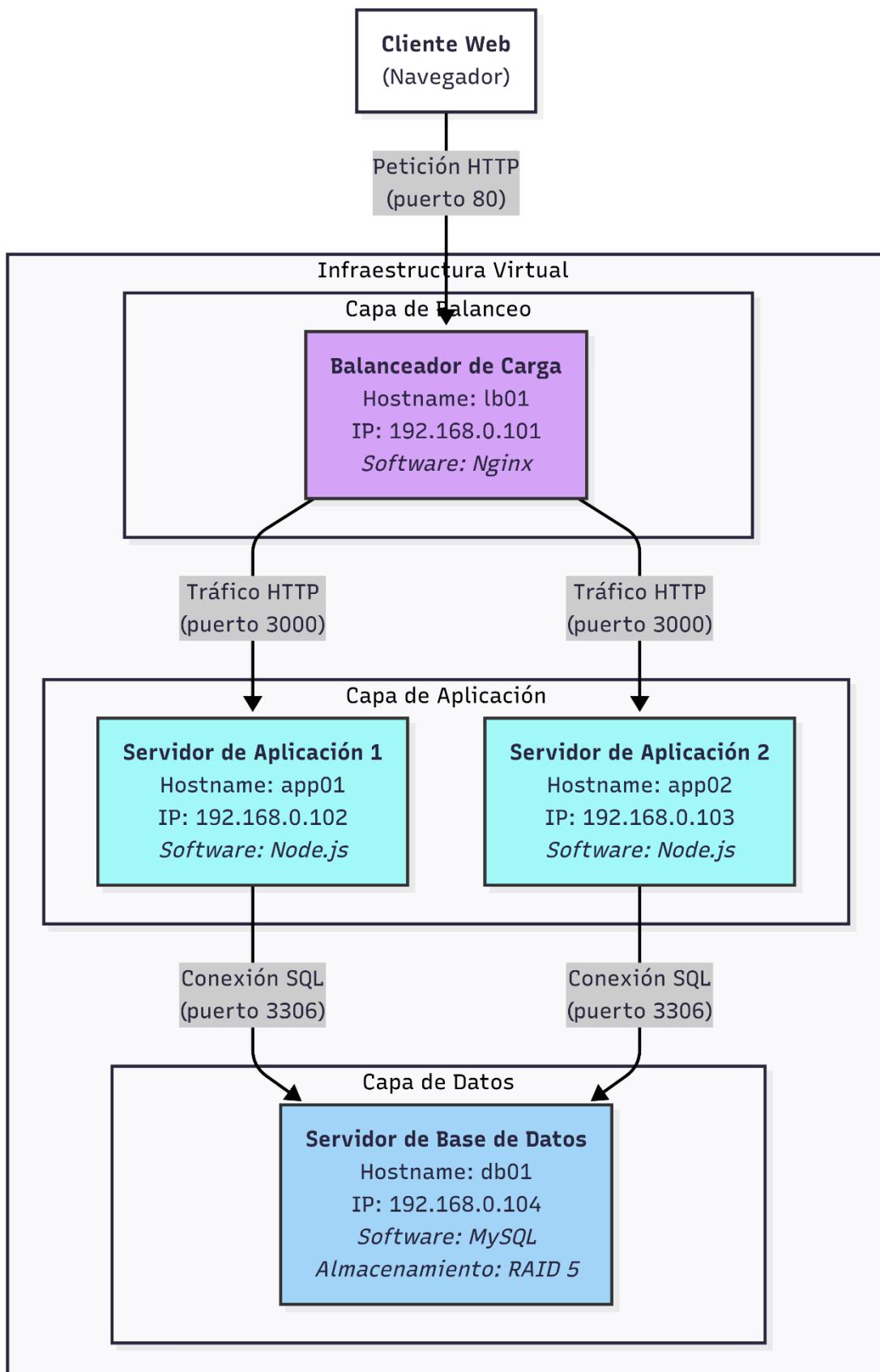
**En el Servidor de Base de Datos (db01):** Se permitió el tráfico entrante en el puerto 3306 (MySQL) para aceptar conexiones de los servidores de aplicación.

**Seguridad de la Base de Datos:** Se ejecutó el script mysql\_secure\_installation para eliminar usuarios anónimos, deshabilitar el acceso remoto del usuario root y establecer una contraseña segura, minimizando los riesgos de acceso no autorizado a la base de datos.

## Diagrama de la Implementación



## Diagrama más técnico:



## CONFIGURACIÓN EN SERVIDOR MySQL

### TOLERANCIA A FALLOS EN DISCOS (RAID5)

- Crear un Arreglo RAID 5

Listamos los discos con **sudo fdisk -l**

```
Device      Start      End  Sectors Size Type
/dev/sda1     2048    4095    2048   1M BIOS boot
/dev/sda2    4096  4198399  4194304   2G Linux filesystem
/dev/sda3  4198400 52426751 48228352  23G Linux filesystem

Disk /dev/sdb: 10,5 GiB, 11274289152 bytes, 22020096 sectors
Disk model: VBOX HARDDISK
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk /dev/sdc: 10 GiB, 10737418240 bytes, 20971520 sectors
Disk model: VBOX HARDDISK
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk /dev/sdd: 10 GiB, 10737418240 bytes, 20971520 sectors
Disk model: VBOX HARDDISK
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk /dev/mapper/ubuntu--vg-ubuntu--lv: 11,5 GiB, 12343836672 bytes, 24109056 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
db01@database:~$ _
```

Usaremos **mdadm** para crear el arreglo. Nos aseguramos de reemplazar `/dev/sdb` `/dev/sdc` `/dev/sdd`

- Crear RAID 5

```
sudo mdadm --create --verbose /dev/md0 --level=5 --raid-devices=3 /dev/sdb /dev/sdc /dev/sdd
```

```
db01@database:~$ sudo mdadm --create --verbose /dev/md0 --level=5 --raid-devices=3 /dev/sdb /dev/sdc /dev/sdd
mdadm: layout defaults to left-symmetric
mdadm: layout defaults to left-symmetric
mdadm: chunk size defaults to 512K
mdadm: size set to 10476544K
mdadm: largest drive (/dev/sdb) exceeds size (10476544K) by more than 1%
Continue creating array? y
mdadm: Defaulting to version 1.2 metadata
mdadm: array /dev/md0 started.
db01@database:~$
```

```
modadmin:~$ cat /proc/mdstat
db01@databaseserver:~$ cat /proc/mdstat
Personalities : [raid0] [raid1] [raid6] [raid5] [raid4] [raid10]
md0 : active raid5 sdd[3] sdc[1] sdb[0]
      20953088 blocks super 1.2 level 5, 512k chunk, algorithm 2 [3/3] [UUU]

unused devices: <none>
db01@databaseserver:~$
```

Formatear el disco creando un sistema de archivos ext4:

```
sudo mkfs.ext4 /dev/md0
```

Crear un punto de montaje:

```
sudo mkdir /mnt/data
```

Montar el Dispositivo:

```
sudo mount /dev/md0 /mnt/data
```

Verificamos que esté montado:

```
df -h
```

```
db01@databaseserver:~$ df -h
Filesystem           Size  Used Avail Use% Mounted on
tmpfs                 383M  1,1M  381M   1% /run
/dev/mapper/ubuntu--vg-ubuntu--lv  12G  5,3G  5,5G  50% /
tmpfs                 1,9G    0  1,9G   0% /dev/shm
tmpfs                 5,0M    0  5,0M   0% /run/lock
/dev/sda2              2,0G  100M  1,7G   6% /boot
tmpfs                 383M   12K  382M   1% /run/user/1000
/dev/md0                20G   24K   19G   1% /mnt/data
db01@databaseserver:~$ _
```

Hacer el Montaje Permanente

Obtenemos el Identificador Único del Disco (UUID)

```
sudo blkid /dev/md0 (Lo copiamos)
```

Editamos el archivo fstab

```
sudo nano /etc/fstab
```

```
UUID=a76a0da7-8ee7-4c39-9c4a-7e28d73d1cbb      /mnt/data      ext4      defaults      0      2
```

## Prueba del Montaje

Desmontar disco manualmente

**sudo umount /mnt/data**

Verificamos

**df -h**

```
db01@databaseserver:~$ sudo umount /mnt/data
db01@databaseserver:~$ df -h
Filesystem           Size  Used Avail Use% Mounted on
tmpfs                 383M  1,1M  381M   1% /run
/dev/mapper/ubuntu--vg-ubuntu--lv  12G  5,3G  5,5G  50% /
tmpfs                 1,9G     0  1,9G   0% /dev/shm
tmpfs                 5,0M     0  5,0M   0% /run/lock
/dev/sda2              2,0G  100M  1,7G   6% /boot
tmpfs                 383M   12K  382M   1% /run/user/1000
db01@databaseserver:~$
```

Ejecutamos el Montaje automático

**sudo mount -a**

Verificamos nuevamente

**df -h**

```
tmpfs                 383M   12K  382M   1% /run
db01@databaseserver:~$ sudo mount -a
db01@databaseserver:~$ df -h
Filesystem           Size  Used Avail Use% Mounted on
tmpfs                 383M  1,1M  381M   1% /run
/dev/mapper/ubuntu--vg-ubuntu--lv  12G  5,3G  5,5G  50% /
tmpfs                 1,9G     0  1,9G   0% /dev/shm
tmpfs                 5,0M     0  5,0M   0% /run/lock
/dev/sda2              2,0G  100M  1,7G   6% /boot
tmpfs                 383M   12K  382M   1% /run/user/1000
/dev/md0                20G   24K   19G   1% /mnt/data
db01@databaseserver:~$
```

RAID 5 100% terminado y funcionando.

---

## CONFIGURACIÓN DE LA BASE DE DATOS MySQL

### Y CREACIÓN DE TABLAS

- Servidor db01 de Bases de Datos (MySQL)

Para instalar MySQL y la herramienta para gestionar RAID, se ejecutó el siguiente comando:

```
sudo apt update && sudo apt install mysql-server mdadm -y
```

```
db01@database:~$ sudo apt update && sudo apt install mysql-server mdadm -y
Obj:1 http://bo.archive.ubuntu.com/ubuntu noble InRelease
Obj:2 http://security.ubuntu.com/ubuntu noble-security InRelease
Obj:3 http://bo.archive.ubuntu.com/ubuntu noble-updates InRelease
Obj:4 http://bo.archive.ubuntu.com/ubuntu noble-backports InRelease
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Se pueden actualizar 58 paquetes. Ejecute «apt list --upgradable» para verlos.
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
mysql-server ya está en su versión más reciente (8.0.42-0ubuntu0.24.04.1).
mdadm ya está en su versión más reciente (4.3-1ubuntu2.1).
0 actualizados, 0 nuevos se instalarán, 0 para eliminar y 58 no actualizados.
db01@database:~$
```

**Configuración de Red:** Se modificó el archivo de configuración mysqld.cnf para cambiar el bind-address de 127.0.0.1 a 0.0.0.0, permitiendo así conexiones remotas desde los servidores de aplicación.

- Necesitamos cambiar esto para que tus servidores de aplicación (**app01** y **app02**) puedan conectarse a esta base de datos.

**Sudo nano /etc/mysql/mysql.conf.d/mysqld.cnf**

La línea de bind-addres es cambiada para que escuche en todas las direcciones de red.

**Bind-address = 0.0.0.0**

```
# localhost which is more compatible and is not less secure.
bind-address            = 0.0.0.0_
mysqlx-bind-address     = 127.0.0.1
#
```

reiniciamos mysql con los cambios y luego vemos el estado.

**Sudo systemctl restart mysql**

**sudo systemctl status mysql**

```
db01@database:~$ sudo systemctl restart mysql
db01@database:~$ sudo systemctl status mysql
● mysql.service - MySQL Community Server
   Loaded: loaded (/usr/lib/systemd/system/mysql.service; enabled; preset: enabled)
   Active: active (running) since Tue 2025-07-01 18:24:24 UTC; 19s ago
     Process: 1800 ExecStartPre=/usr/share/mysql/mysql-systemd-start pre (code=exited, status=0/SUCCESS)
    Main PID: 1808 (mysqld)
      Status: "Server is operational"
     Tasks: 38 (limit: 4490)
    Memory: 363.9M (peak: 378.2M)
       CPU: 2.742s
      CGroup: /system.slice/mysql.service
              └─1808 /usr/sbin/mysqld

Jul 01 18:24:22 database systemd[1]: Starting MySQL Community Server...
Jul 01 18:24:24 database systemd[1]: Started MySQL Community Server.
db01@database:~$
```

## Creación de la Base de Datos y Usuario

Se creó una base de datos (sis313\_db) y un usuario específico (app\_user) con los privilegios necesarios para que las aplicaciones se conecten de forma segura, sin usar el usuario root

- Entrar a la Consola de MySQL

```
sudo mysql -u root -p
```

```
db01@databaseserver:~$ sudo mysql -u root -p
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 10
Server version: 8.0.42-0ubuntu0.24.04.1 (Ubuntu)

Copyright (c) 2000, 2025, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> _
```

- Creación de la Base de Datos:

```
CREATE DATABASE sis313_db;
```

```
mysql> CREATE DATABASE sis313_db;
Query OK, 1 row affected (0,03 sec)

mysql>
```

- Crear un nuevo usuario para tu aplicación:

```
CREATE USER 'app_user'@'%' IDENTIFIED BY 'Kevin123#';
```

```
mysql> CREATE USER 'app_user'@'%' IDENTIFIED BY 'Kevin123#';
Query OK, 0 rows affected (0,07 sec)

mysql>
```

- Darle todos los permisos a ese usuario sobre la nueva base de datos:

```
GRANT ALL PRIVILEGES ON sis313_db.* TO 'app_user'@'%';
```

```
mysql> GRANT ALL PRIVILEGES ON sis313_db.* TO 'app_user'@'%';
Query OK, 0 rows affected (0,04 sec)

mysql>
```

- Aplicar los cambios de privilegios:

```
FLUSH PRIVILEGES;
```

```
mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0,02 sec)

mysql>
```

## Creación de Tabla en La Base de Datos

- Seleccionar la Base de Datos para trabajar en ella:

```
USE sis313_db;
```

```
mysql> USE sis313_db;
Database changed
mysql>
```

- Crear la tabla **productos** para el CRUD:

```
mysql> CREATE TABLE productos (
    -> id INT AUTO_INCREMENT PRIMARY KEY,
    -> nombre VARCHAR(255) NOT NULL,
    -> descripcion TEXT,
    -> fecha_creacion TIMESTAMP DEFAULT CURRENT_TIMESTAMP
    -> );
Query OK, 0 rows affected (0,17 sec)
```

```
mysql>
```

- Ver la tabla

```
DESCRIBE productos;
```

```
mysql> DESCRIBE productos;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| id   | int  | NO   | PRI | NULL    | auto_increment |
| nombre | varchar(255) | NO | NULL | NULL |
| descripcion | text | YES | NULL | NULL |
| fecha_creacion | timestamp | YES | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
+-----+-----+-----+-----+-----+
4 rows in set (0,01 sec)
```

```
mysql>
```

- Salir de MySQL:

```
mysql> EXIT
Bye
db01@databaseserver:~$
```

- Por último, convertir la ip de **databaseserver** ah estática.

Esto es crucial para que los servidores de aplicación (**app01** y **app02**) siempre sepan exactamente dónde encontrar la base de datos.

Sudo nano /etc/netplan/50-cloud-init.yaml

```
network:
  version: 2
  ethernets:
    enp0s3:
      dhcp4: no
      addresses: [192.168.0.104/24]
      gateway4: 192.168.0.1
      nameservers:
        addresses: [8.8.8.8, 1.1.1.1]
```

## CONFIGURACIÓN DEL SERVIDOR DE APLICACIONES

### App01 y App02 con Node.js

En app01server - app01 - aplicación web 1

#### Instalar Software Básico

- Instala Node.js y NPM

```
sudo apt update && sudo apt install nodejs npm -y
```

```
app01@appserver01:~$ sudo apt update && sudo apt install nodejs npm -y
Obj:1 http://security.ubuntu.com/ubuntu noble-security InRelease
Obj:2 http://bo.archive.ubuntu.com/ubuntu noble InRelease
Obj:3 http://bo.archive.ubuntu.com/ubuntu noble-updates InRelease
Obj:4 http://bo.archive.ubuntu.com/ubuntu noble-backports InRelease
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Se pueden actualizar 58 paquetes. Ejecute «apt list --upgradable» para verlos.
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
nodejs ya está en su versión más reciente (18.19.1+dfsg-6ubuntu5).
npm ya está en su versión más reciente (9.2.0~ds1-2).
0 actualizados, 0 nuevos se instalarán, 0 para eliminar y 58 no actualizados.
app01@appserver01:~$
```

#### Instalar Paquetes de la Aplicación

- Usa NPM para instalar Express (el framework web) y el conector de MySQL.

```
npm install express mysql2
```

```
app01@appserver01:~$ npm install express mysql2
up to date, audited 79 packages in 2s

15 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
app01@appserver01:~$ _
```

- Configurar ip estatica

```
sudo nano /etc/netplan/50-cloud-init.yaml
```

```
network:
  version: 2
  ethernets:
    enp0s3:
      dhcp4: no
      addresses: [192.168.0.102/24]
      gateway4: 192.168.0.1
      nameservers:
        addresses: [8.8.8.8, 1.1.1.1]
```

- Se crea un archivo

nano app.js

```

const express = require('express');
const mysql = require('mysql2');
const app = express();
const port = 3000;

app.use(express.json());

// --- Configuracion de la Conexion a la Base de Datos ---
const db = mysql.createConnection({
  host: '192.168.0.104',          // La IP del servidor db01
  user: 'app_user',              // Usuario de la BD
  password: 'Kevin123#',         // La contraseña que se asigno para app_user
  database: 'sis313_db'          // La BD
});

// Conectar a la Base de Datos
db.connect(err => {
  if(err) {
    console.error('Error al conectar a la base de datos:', err);
    return;
  }
  console.log('Conectando exitosamente a la Base de Datos MySQL.');
});

// Ruta para la pagina principal (raiz) para pruebas
app.get('/', (req, res) => {
  res.send('API funcional en Servidor APP01. ');
});

// --- Definicion de Rutas del CRUD ---

// Ruta para OBTENER todos los productos (READ)
app.get('/productos', (req, res) => {
  const sql = 'SELECT * FROM productos';
  db.query(sql, (err, results) => {
    if (err) { return res.status(500).send('Error en el servidor al obtener productos'); }
    res.json(results);
  });
});

// Ruta para crear un nuevo producto (CREATE)
app.post('/productos', (req, res) => {
  const { nombre, descripcion } = req.body;

```

```
const sql = 'INSERT INTO productos (nombre, descripcion) VALUES (?, ?)';
db.query(sql, [nombre, descripcion], (err, result) => {
    if (err) { return res.status(500).send('Error en el servidor al crear producto'); }
    res.status(201).json({ id: result.insertId, nombre, descripcion, message: 'Producto creado!' });
});

// Ruta para actualizar un producto (UPDATE)
app.put('/productos/:id', (req, res) => {
    const { id } = req.params;
    const { nombre, descripcion } = req.body;
    const sql = 'UPDATE productos SET nombre = ?, descripcion = ? WHERE id = ?';
    db.query(sql, [nombre, descripcion, id], (err, result) => {
        if (err) { return res.status(500).send('Error en el servidor al actualizar producto'); }
    });
});

// Ruta para borrar un producto (DELETE)
app.delete('/productos/:id', (req, res) => {
    const { id } = req.params;
    const sql = 'DELETE FROM productos WHERE id = ?';
    db.query(sql, [id], (err, result) => {
        if (err) { return res.status(500).send('Error en el servidor al borrar producto'); }
        res.json({ message: 'Producto eliminado!' });
    });
});

// --- Iniciar Servidor ---
app.listen(port, () => {
    console.log(`API CRUD corriendo en el puesto ${port}`);
});
```

En app02server - app02 - aplicación web 2

## Instalar Software Básico

- Instala Node.js y NPM

```
sudo apt update && sudo apt install nodejs npm -y
```

```
app02@appserver02:~$ sudo apt update && sudo apt install nodejs npm -y
Obj:1 http://bo.archive.ubuntu.com/ubuntu noble InRelease
Obj:2 http://security.ubuntu.com/ubuntu noble-security InRelease
Obj:3 http://bo.archive.ubuntu.com/ubuntu noble-updates InRelease
Obj:4 http://bo.archive.ubuntu.com/ubuntu noble-backports InRelease
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Se pueden actualizar 58 paquetes. Ejecute «apt list --upgradable» para verlos.
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
nodejs ya está en su versión más reciente (18.19.1+dfsg-6ubuntu5).
npm ya está en su versión más reciente (9.2.0~ds1-2).
0 actualizados, 0 nuevos se instalarán, 0 para eliminar y 58 no actualizados.
app02@appserver02:~$
```

## Instalar Paquetes de la Aplicación

- Usa NPM para instalar Express (el framework web) y el conector de MySQL.

```
npm install express mysql2
```

```
app02@appserver02:~$ npm install express mysql2
up to date, audited 79 packages in 861ms

15 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
app02@appserver02:~$
```

- Configurar ip estatica

```
sudo nano /etc/netplan/50-cloud-init.yaml
```

```
network:
  version: 2
  ethernets:
    enp0s3:
      dhcp4: no
      addresses: [192.168.0.103/24]
      gateway4: 192.168.0.1
      nameservers:
        addresses: [8.8.8.8, 1.1.1.1]
```

- Se crea un archivo

nano app.js

```
const express = require('express');
const mysql = require('mysql2');
const app = express();
const port = 3000;

app.use(express.json());

// --- Configuracion de la Conexion a la Base de Datos ---
const db = mysql.createConnection({
  host: '192.168.0.104',          // La IP del servidor db01
  user: 'app_user',
  password: 'Kevini123#',
  database: 'sis318_db'
});

// Conectar a la base de datos
db.connect(err => {
  if(err) {
    console.error('Error al conectar a la base de datos:', err);
    return;
  }
  console.log('Conectando exitosamente a la Base de Datos MySQL.');
});

// Ruta para la pagina principal (raiz) para pruebas
app.get('/', (req, res) => {
  res.send('API funcional en Servidor APP02.');
});

// --- Definicion de Rutas del CRUD ---

// Ruta para obtener todos los productos (READ)
app.get('/productos', (req, res) => {
  const sql = 'SELECT * FROM productos';
  db.query(sql, (err, results) => {
    if (err) { return res.status(500).send('Error en el servidor al obtener productos'); }
    res.json(results);
  });
});

// Ruta para crear un nuevo producto (CREATE)
app.post('/productos', (req, res) => {
  const { nombre, descripcion } = req.body;
```

```

const sql = 'INSERT INTO productos (nombre, descripcion) VALUES (?, ?)';
db.query(sql, [nombre, descripcion], (err, result) => {
    if (err) { return res.status(500).send('Error en el servidor al crear producto'); }
    res.status(201).json({ id: result.insertId, nombre, descripcion, message: 'Producto creado!' });
});

// Ruta para actualizar un producto (UPDATE)
app.put('/productos/:id', (req, res) => {
    const { id } = req.params;
    const { nombre, descripcion } = req.body;
    const sql = 'UPDATE productos SET nombre = ?, descripcion = ? WHERE id = ?';
    db.query(sql, [nombre, descripcion, id], (err, result) => {
        if (err) { return res.status(500).send('Error en el servidor al actualizar producto'); }
    });
});

// Ruta para borrar un producto (DELETE)
app.delete('/productos/:id', (req, res) => {
    const { id } = req.params;
    const sql = 'DELETE FROM productos WHERE id = ?';
    db.query(sql, [id], (err, result) => {
        if (err) { return res.status(500).send('Error en el servidor al borrar producto'); }
        res.json({ message: 'Producto eliminado!' });
    });
});

// --- Iniciar Servidor ---
app.listen(port, () =>{
    console.log(`API CRUD corriendo en el puesto ${port}`);
});

```

- Probamos la conexión

#### node app.js

```

app02@appserver02:~$ node app.js
API CRUD corriendo en el puesto 3000
Conectando exitosamente a la Base de Datos MySQL.
-
```

```

app01@appserver01:~$ node app.js
API CRUD corriendo en el puesto 3000
Conectando exitosamente a la Base de Datos MySQL.
```

- Ver las conexiones desde la Base de Datos  
SHOW PROCESSLIST;

```

mysql> SHOW PROCESSLIST;
+-----+-----+-----+-----+-----+-----+-----+
| Id | User      | Host     | db   | Command | Time | State          | Info           |
+-----+-----+-----+-----+-----+-----+-----+
| 5  | event_scheduler | localhost | NULL | Daemon  | 6563 | Waiting on empty queue | NULL          |
| 16 | app_user    | 192.168.0.102:41754 | sis313_db | Sleep   | 175  |                  | NULL          |
| 17 | app_user    | 192.168.0.103:40072 | sis313_db | Sleep   | 153  |                  | NULL          |
| 18 | root       | localhost | NULL | Query   | 0    | init            | SHOW PROCESSLIST |
+-----+-----+-----+-----+-----+-----+-----+
4 rows in set, 1 warning (0,01 sec)
```

## CONFIGURACIÓN DE LAS APLICACIONES WEB TERMINADAS

# CONFIGURACIÓN DE BALANCEADOR CON NGINX

## ALOJADA EN LB01

Repartirá el tráfico de app01 y app02

- Instalar NGINX en lb01

```
sudo apt update && sudo apt install nginx -y
```

```
lb01@balancerserver:~$ sudo apt update && sudo apt install nginx -y
Obj:1 http://bo.archive.ubuntu.com/ubuntu noble InRelease
Obj:2 http://bo.archive.ubuntu.com/ubuntu noble-updates InRelease
Obj:3 http://bo.archive.ubuntu.com/ubuntu noble-backports InRelease
Obj:4 http://security.ubuntu.com/ubuntu noble-security InRelease
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Se pueden actualizar 58 paquetes. Ejecute «apt list --upgradable» para verlos.
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
nginx ya está en su versión más reciente (1.24.0-2ubuntu7.4).
0 actualizados, 0 nuevos se instalarán, 0 para eliminar y 58 no actualizados.
lb01@balancerserver:~$
```

- Ip estatica

```
network:
  version: 2
  ethernets:
    enp0s3:
      dhcp4: no
      addresses: [192.168.0.101/24]
      gateway4: 192.168.0.1
      nameservers:
        addresses: [8.8.8.8, 1.1.1.1]
```

- Editar el archivo de configuración por defecto de NGINX

```
sudo nano /etc/nginx/sites-available/default
```

```
# Se define el grupo de servidores de aplicacion
upstream servidores_app {
    server 192.168.0.102:3000;
    server 192.168.0.103:3000;
}

# Configuracion del servidor que recibe el trafico
server {
    listen 80 default_server;
    listen [::]:80 default_server;

    server_name _;

    location / {
        # Pasa la petición al grupo de servidores
        proxy_pass http://servidores_app;
    }
}
```

- Se verifica la sintaxis de la configuración

```
sudo nginx -t
```

```
lb01@balancerserver:~$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
lb01@balancerserver:~$
```

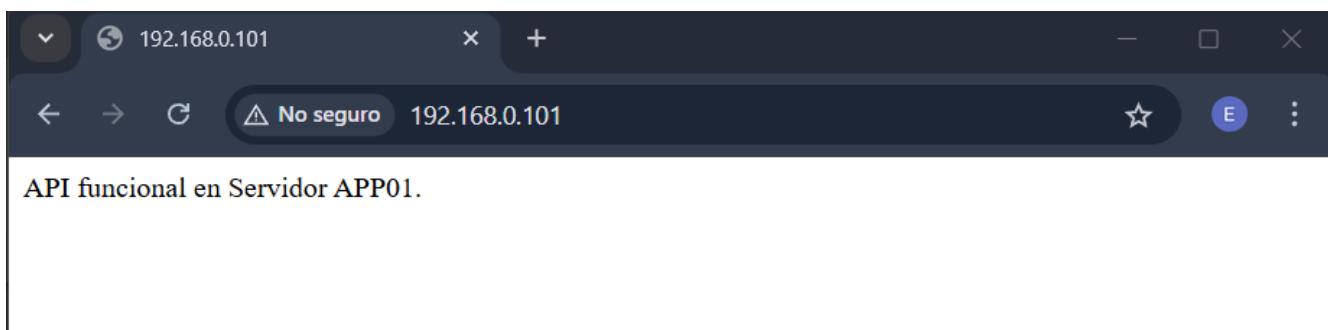
- Se aplica los cambios

```
sudo systemctl reload nginx
```

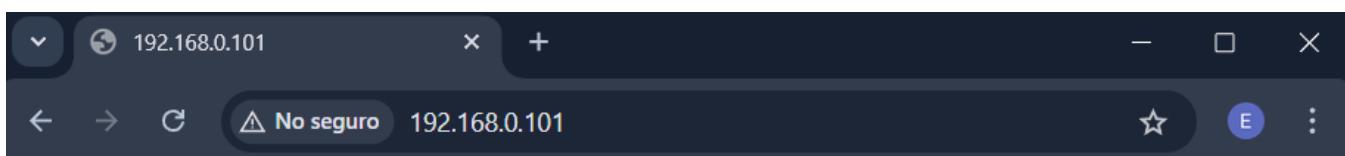
```
nginx: configuration file /etc/nginx/nginx.conf test is successful
lb01@balancerserver:~$ sudo systemctl reload nginx
lb01@balancerserver:~$ http://192.168.0.101/productos
```

- **Verificación:** Se recargó la configuración de NGINX y se realizaron pruebas de acceso desde un navegador web y mediante curl directamente en el servidor. Las peticiones a la IP.

192.168.0.101 mostraron respuestas alternadas de los servidores "APP01" y "APP02", confirmando que el balanceo de carga funcionaba correctamente.



- (refrescando la página) F5



- Probamos el CRUD. (corchetes vacíos, porque se agrego productos aún)

```
[+] lb01@balancerserver:~$ curl http://192.168.0.101/productos  
[] lb01@balancerserver:~$ curl http://127.0.0.1
```

- Prueba interna de Tolerancia a Fallos. (el balanceador distribuye el tráfico)

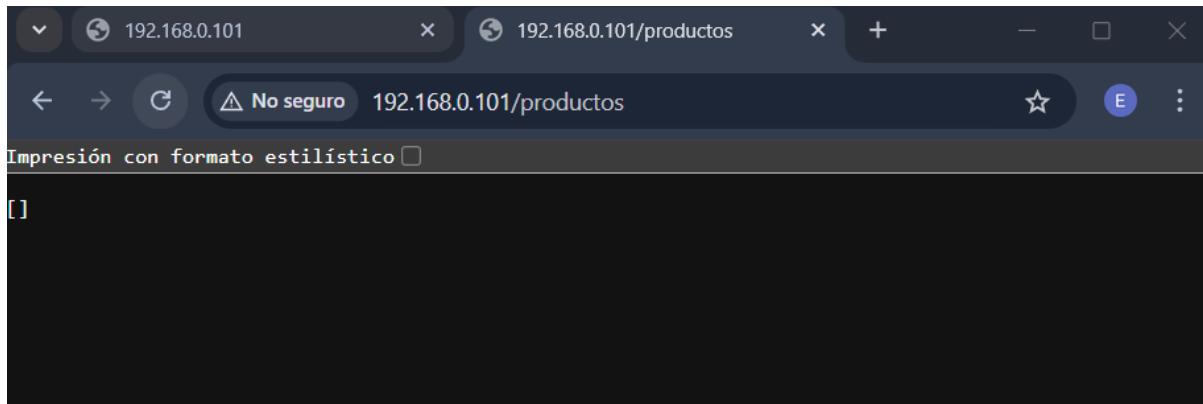
```
[+] lb01@balancerserver:~$ curl http://127.0.0.1  
API funcional en Servidor APP02. lb01@balancerserver:~$ curl http://127.0.0.1  
API funcional en Servidor APP01. lb01@balancerserver:~$ curl http://127.0.0.1  
API funcional en Servidor APP02. lb01@balancerserver:~$ curl http://127.0.0.1  
API funcional en Servidor APP01. lb01@balancerserver:~$ curl http://127.0.0.1  
API funcional en Servidor APP02. lb01@balancerserver:~$ curl http://127.0.0.1  
API funcional en Servidor APP01. lb01@balancerserver:~$ curl http://127.0.0.1  
API funcional en Servidor APP02. lb01@balancerserver:~$ curl http://127.0.0.1  
API funcional en Servidor APP01. lb01@balancerserver:~$ curl http://127.0.0.1  
API funcional en Servidor APP02. lb01@balancerserver:~$ curl http://127.0.0.1
```

## Comprobaciones

### 1. Probar “Leer” (READ)

- Se abre un navegador web y se visita la siguiente dirección para pedir lista de productos:

<http://192.168.0.101/productos>



Ver una página con unos corchetes vacíos [], lo que significa que el balanceador se comunicó con una de las aplicaciones y esta consultó la base de datos (que está vacía).

### 2. Probar el “Crear” (Create)

- Desde la terminal de **1b01**, usa curl para crear un producto:

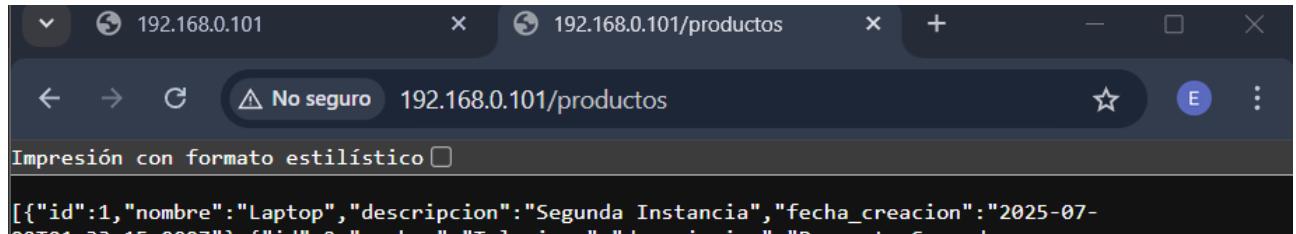
```
curl -X POST -H "Content-Type: application/json" -d '{"nombre": "Laptop", "descripcion": "segunda instancia"}' http://192.168.0.101/productos
```

```
[root@1b01 ~]# curl -X POST -H "Content-Type: application/json" -d '{"nombre": "Laptop", "descripcion": "Segunda Instancia"}' http://192.168.0.101/productos
{"id":1,"nombre":"Laptop","descripcion":"Segunda Instancia"}
```

- revisamos la base de datos

```
mysql> SELECT *FROM productos;
+----+-----+-----+-----+
| id | nombre | descripcion | fecha_creacion |
+----+-----+-----+-----+
| 1 | Laptop | Segunda Instancia | 2025-07-02 01:33:15 |
+----+-----+-----+-----+
1 row in set (0,00 sec)
```

- También en el navegador



- Prueba Final

A continuación, se desactiva manualmente una aplicación web app01

```
app01@appserver01:~$ node app.js
API CRUD corriendo en el puesto 3000
Conectando exitosamente a la Base de Datos MySQL.
^C
app01@appserver01:~$
```

- Con una aplicación apagada, se seguirá posteando o creando un nuevo producto.

(La infraestructura con la implementación completada a tolerancia a fallos, tendría que crear el producto sin problemas y el balanceador NGINX ve este error y pasa toda la petición del tráfico al otro servidor app02.)

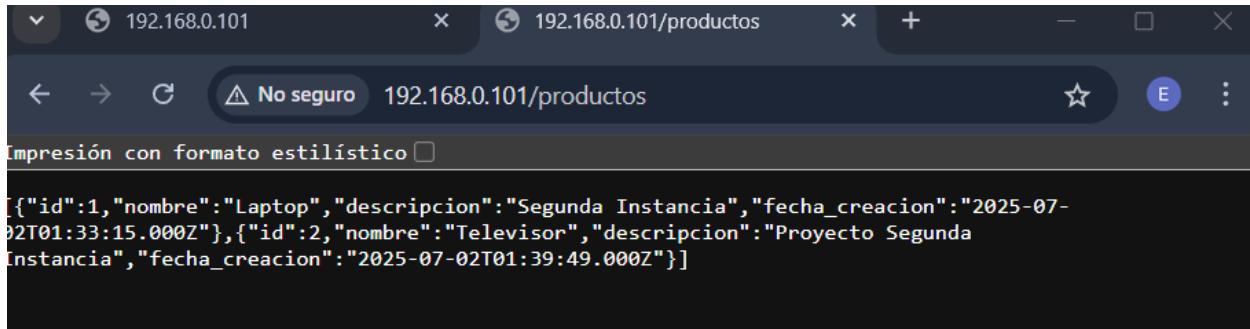
```
lb01@balancerserver:~$ curl -X POST -H "Content-Type: application/json" -d '{"nombre": "Televisor", "descripcion": "Proyecto Segunda Instancia"}' http://192.168.0.101/productos
{"id":2,"nombre":"Televisor","descripcion":"Proyecto Segunda Instancia","message":"Producto creado!"}lb01@balancerserver:~$
```

- Comprobamos la base de datos

```
mysql> SELECT *FROM productos;
+----+-----+-----+-----+
| id | nombre | descripcion | fecha_creacion |
+----+-----+-----+-----+
| 1 | Laptop | Segunda Instancia | 2025-07-02 01:38:15 |
| 2 | Televisor | Proyecto Segunda Instancia | 2025-07-02 01:39:49 |
+----+-----+-----+-----+
2 rows in set (0,00 sec)

mysql> _
```

- También comprobamos en el navegador web



The screenshot shows a web browser window with two tabs open. The left tab is titled '192.168.0.101' and the right tab is titled '192.168.0.101/productos'. The main content area displays a JSON array:

```
[{"id":1,"nombre":"Laptop","descripcion":"Segunda Instancia","fecha_creacion":"2025-07-02T01:33:15.000Z"}, {"id":2,"nombre":"Televisor","descripcion":"Proyecto Segunda Instancia","fecha_creacion":"2025-07-02T01:39:49.000Z"}]
```

---

## Resumen Final de Logros del Proyecto

Con esto se ha cumplido con todos los requisitos funcionales de la infraestructura:

- **Balanceador de Carga (Nginx):** Completado.
- **Servidores de Aplicaciones (Node.js):** Completado.
- **Aplicación CRUD:** Completado. La aplicación puede crear, leer datos, actualizar y eliminar datos.
- **Servidor de Base de Datos (MySQL):** Completado.
- **Tolerancia a Fallos en Discos (RAID 5):** Completado. El arreglo está creado y configurado.
- **Comunicación de Red (Networking):** Completado. Todas las instancias se comunican con sus IPs estáticas.