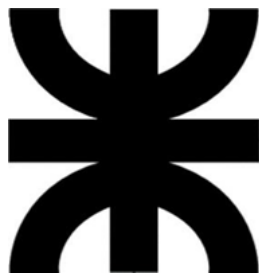


Universidad Tecnológica Nacional

Facultad Regional Tucumán



Cátedra Virtualización - Consolidación de Servidores

Trabajo Final Integrador

“Caso de Uso – Virtualización Parcial”

Alumno:

Maza, Sebastián Agustín - 50240

Docente:

Carriles, Luis Maria

Comisión: 5k3

Fecha de Presentación: 25/06/2024

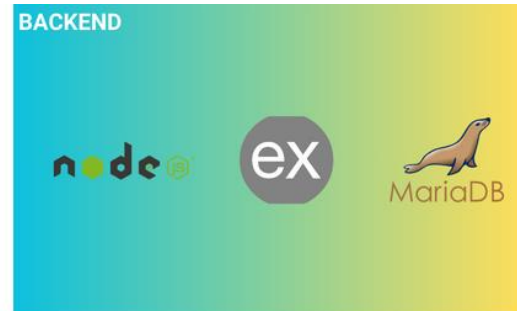
Contenido

Arquitectura de las aplicaciones	2
Acceso a la plataforma Proxmox.....	2
Doble Factor de Autenticación	3
Creación de contenedores	4
Contenido de los contenedores.....	7
Configuración del contenedor de Backend	7
Configuración del contenedor de Frontend	13
Comunicación entre contenedores	14
Redirección de puertos y pruebas	14

Arquitectura de las aplicaciones

Se desarrollaron dos aplicaciones: una de Backend y otra de Frontend. Para la aplicación de Backend, se optó por crear una API con Node.js que se conecta a una base de datos MariaDB. Además, se implementó seguridad en la ejecución de ciertas solicitudes HTTP mediante Json Web Token (JWT) y se encriptaron las contraseñas de los usuarios.

Para el Frontend, se desarrolló una aplicación con React que se comunica con la API mediante solicitudes HTTP para obtener o enviar datos. Se utilizó React Router DOM para la gestión de las rutas y su protección. También se realizaron las validaciones de los formularios con React Hook Form.

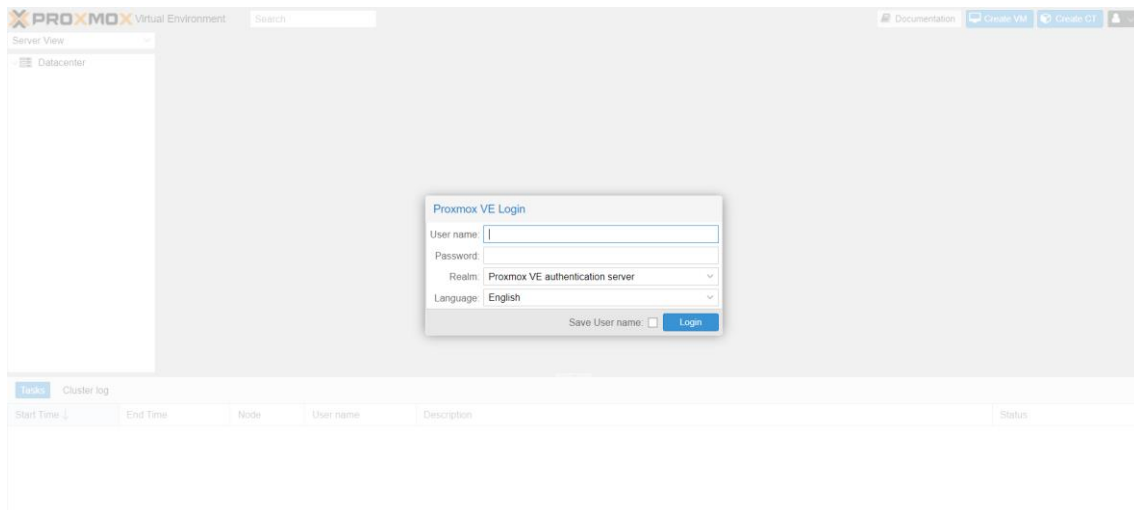


Acceso a la plataforma Proxmox

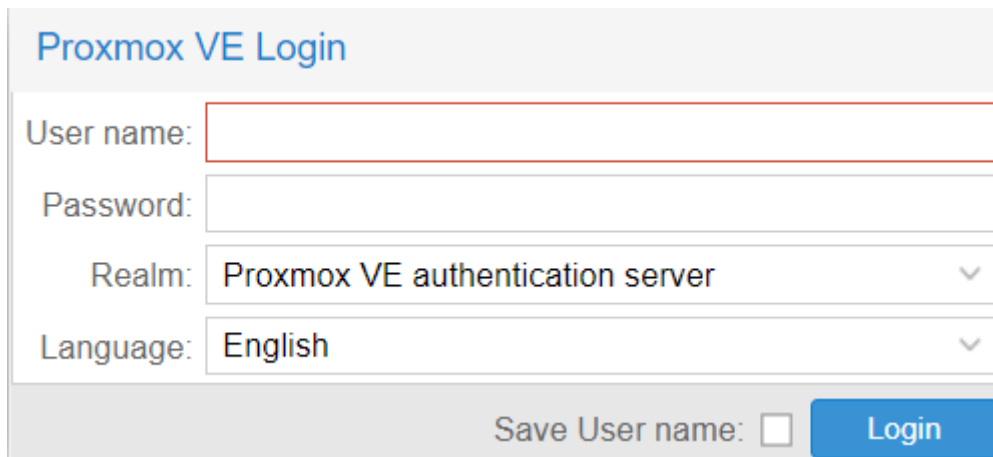
Para ingresar a la plataforma Proxmox, se debe enviar al administrador del clúster una solicitud de alta del usuario con el que vamos a trabajar.

Una vez creado el usuario, el administrador nos proporcionó la ip para acceder al clúster acompañado del usuario y contraseña para ingresar al mismo.

Al ingresar a la ip correspondiente al cluster, nos aparecía la siguiente pantalla:



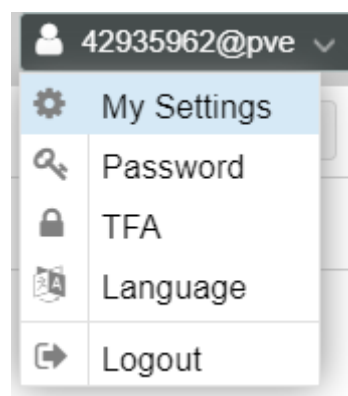
Ahí debemos iniciar sesión para poder ingresar a la plataforma, para ingresar correctamente, se debe ingresar el usuario la contraseña y además en el campo “Realm” seleccionar la opción que diga “Proxmox VE authentication server”.



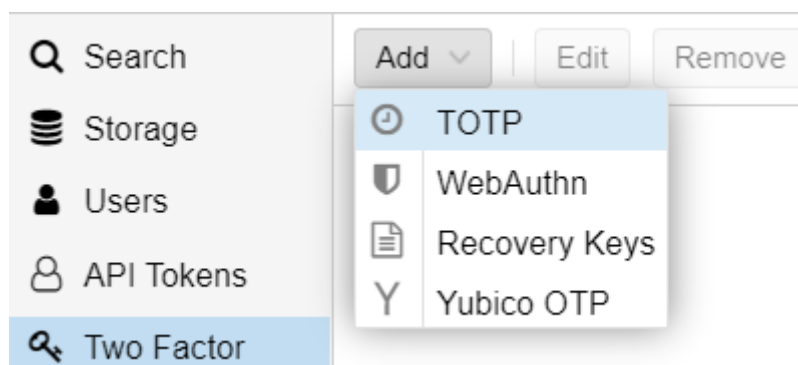
The image shows the Proxmox VE Login interface. It has a title 'Proxmox VE Login' in blue. Below the title are four input fields: 'User name:' (with a red border), 'Password:', 'Realm:' (set to 'Proxmox VE authentication server'), and 'Language:' (set to 'English'). At the bottom right, there is a checkbox for 'Save User name:' and a blue 'Login' button.

Doble Factor de Autenticación

Una vez ingresado en Proxmox, nos dirigiremos a la esquina superior derecha, en donde esta nuestro nombre de usuario y seleccionamos la opción “TFA”.

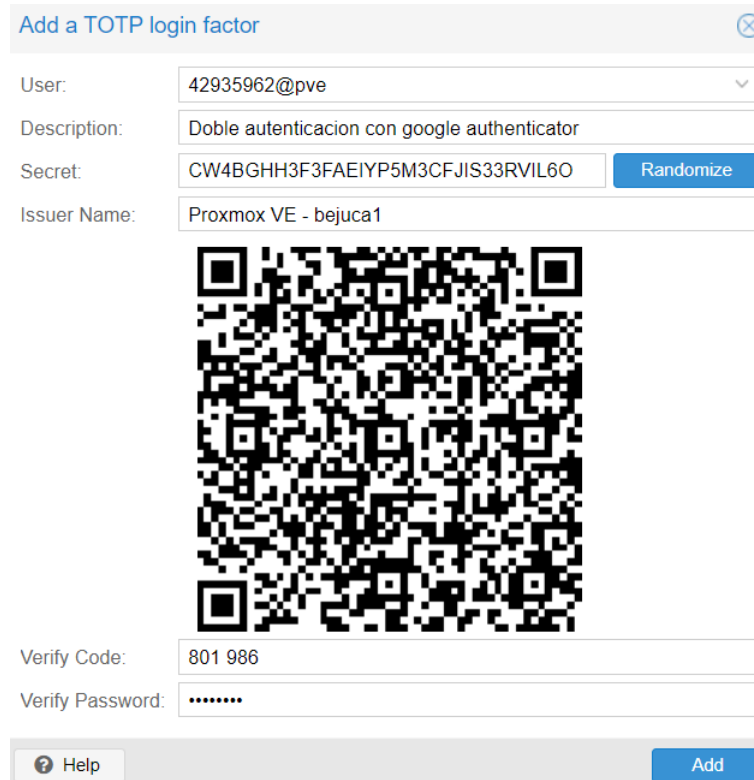


Llegaremos a una pantalla con diferentes opciones, seleccionamos la opción de “Two Factor” y luego presionamos en “Add” y damos click en “TOTP” (algoritmo de contraseña de un solo uso).



Se nos abre una ventana, en donde hay que completar algunos datos para poder agregar el doble factor exitosamente. En esta vista, se nos muestra un código qr que debe ser escaneado por alguna aplicación de autenticación.

Para esta práctica elegí la aplicación Google Authenticator para Android, con la cual escaneé el código qr y me proporciona una clave que dura aproximadamente 30 segundos y luego se genera otra clave, esta clave debe ser colocada en el campo “Verify Code”.



Add a TOTP login factor

User: 42935962@pve

Description: Doble autentificacion con google authenticator

Secret: CW4BGHH3F3FAEIYP5M3CFJIS33RVIL6O Randomize

Issuer Name: Proxmox VE - bejuca1

Verify Code: 801 986

Verify Password:

Help Add

Cuando completamos todos los datos, presionamos en el botón “Add” y ya tenemos configurado el doble factor de autenticación.

Creación de contenedores

Para la creación de los contenedores se nos dieron ciertas pautas, que indicaremos a continuación:

- Para los nombres de estos se debe utilizar las siguientes especificaciones:
 - o Para el contenedor de Backend el nombre debe ser DNI del alumno + DB
 - o Para el contenedor de Frontend el nombre debe ser DNI del alumno + A
- Ambos contenedores deben tener los mismos recursos los cuales son:
 - o Memoria RAM: 128 Mb.
 - o Almacenamiento: 8 Gb.
 - o Procesamiento: 1 núcleo.
 - o Ambos contenedores deben obtener ip por DHCP.

Una vez dentro de la plataforma de proxmox, debemos elegir el nodo en el cual queremos crear nuestros contenedores, en este caso se eligió el nodo “bejuca2”, nos posicionamos en él y presionamos la opción “CREATE CT” que se encuentra en la parte superior derecha.



Se nos abrirá una ventana en la cual vamos a tener que ir completando algunos datos, en la primera, deberemos colocar el nombre de nuestro contenedor y una contraseña para poder a la consola del mismo y luego presionar “next”.

Create: LXC Container

General

Template

Disks

CPU

Memory

Network

DNS

Confirm

Node:

bejuca2

CT ID:

146

Hostname:

42935962DB

Unprivileged container:

☒

Nesting:

☒

Resource Pool:

Password:

.....

Confirm password:

.....

SSH public key:

Load SSH Key File

Help

Advanced ☐

Back

Next

En la siguiente ventana seleccionaremos el sistema operativo, en nuestro caso elegimos Ubuntu.

Create: LXC Container

General

Template

Disks

CPU

Memory

Network

DNS

Confirm

Storage:

local

Template:

l4-standard_20.04-1_amd64.tar.gz

Help

Advanced ☐

Back

Next

Ahora, procederemos a colocar los recursos del contenedor, como ser almacenamiento, procesadores y memoria RAM, se hacen de a uno y se debe presionar “next”.

Create: LXC Container ⓧ

General Template **Disks** CPU Memory Network DNS Confirm

rootfs	Storage:	local-lvm
	Disk size (GiB):	8

Create: LXC Container ⓧ

General Template Disks **CPU** Memory Network DNS Confirm

Cores: 1

Create: LXC Container ⓧ

General Template Disks CPU **Memory** Network DNS Confirm

Memory (MiB): 128

Swap (MiB): 128

Lo ultimo que debemos configurar nosotros es la red, la cual por defecto aparece para asignar una dirección ip estática, a eso lo debemos cambiar para que reciba dicha dirección por DHCP.

Create: LXC Container ⓧ

General Template Disks CPU Memory **Network** DNS Confirm

Name:	eth0	IPv4:	<input type="radio"/> Static <input checked="" type="radio"/> DHCP
MAC address:	auto	IPv4/CIDR:	
Bridge:	vmbr0	Gateway (IPv4):	
VLAN Tag:	no VLAN	IPv6:	<input checked="" type="radio"/> Static <input type="radio"/> DHCP <input type="radio"/> SLAAC
Rate limit (MB/s):	unlimited	IPv6/CIDR:	None
Firewall:	<input checked="" type="checkbox"/>	Gateway (IPv6):	

Una vez hecho esto, presionamos “next”, en la pestaña DNS no configuramos nada y pasamos a confirmar la creación del contenedor, se nos mostrara una ventana con los datos del contenedor y deberemos presionar el botón “finish”.

Create: LXC Container

General
Template
Disks
CPU
Memory
Network
DNS
Confirm

Key ↑	Value
cores	1
features	nesting=1
hostname	42935962DB
memory	128
net0	name=eth0,bridge=vmbr0,firewall=1,ip=dhcp
nodename	bejuca2
ostemplate	local:vztmpl/ubuntu-20.04-standard_20.04-1_amd64.tar.gz
pool	
rootfs	local-lvm:8
swap	128
unprivileged	1
vmid	146

☐ Start after created

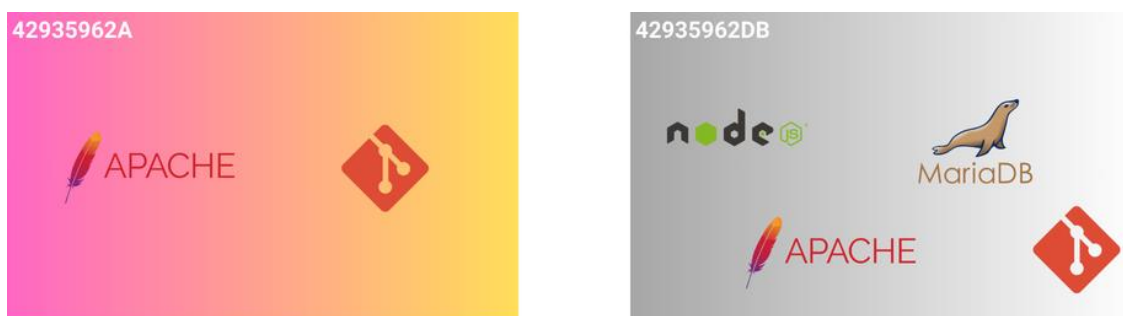
Advanced
Back
Finish

Una vez realizado esto, se debe esperar un momento hasta que se realiza la creación del contenedor.

Se deben seguir los mismos pasos para la creación de ambos contenedores.

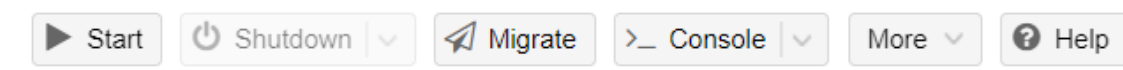
Contenido de los contenedores

En la siguiente imagen se muestra lo que contendrá cada contenedor, para este problema elegimos utilizar el servidor web apache en ambos contenedores y git para el manejo de las versiones de las aplicaciones.

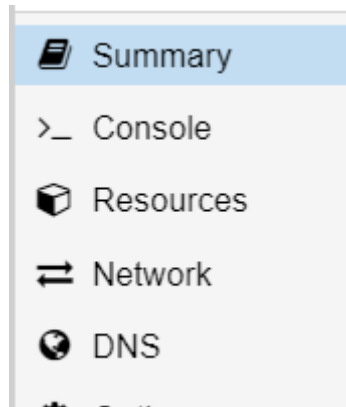


Configuración del contenedor de Backend

Comenzaremos con la configuración del contenedor de Backend, para ellos deberemos iniciar el contenedor presionando el botón “start”.



Una vez iniciado el contenedor, presionaremos en “console” para que podamos acceder a la terminal del contenedor.



Cuando se inicie la terminal, tendremos que iniciar sesión con el usuario “root” y la contraseña que definimos cuando creamos el contenedor.

```
Ubuntu 20.04 LTS 42935962DB tty1
42935962DB login: root
Password:
Welcome to Ubuntu 20.04 LTS (GNU/Linux 5.15.30-2-pve x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

Last login: Mon Jun 24 22:43:04 UTC 2024 on tty1
root@42935962DB:~#
```

Una vez iniciada la sesión, realizamos lo siguiente:

1. Como primer paso ejecutamos el comando “**apt update**” para actualizar la lista de paquetes.
2. Como utilizaremos la base de datos Maria DB, instalamos la misma en el contenedor mediante el comando “**apt-get install mariadb-server**”.
3. Una vez ejecutado el comando anterior, verificamos que maria db se haya instalado correctamente, para ello ejecutamos el comando “**mariadb -v**”, si accede a la base de datos significa que se instaló correctamente.

```
root@42935962DB:~# mariadb -v
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 36
Server version: 10.3.39-MariaDB-0ubuntu0.20.04.2 Ubuntu 20.04

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Reading history-file /root/.mysql_history
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

4. También es conveniente mirar el estado del servicio para verificar que este activo, en este caso, lo verificamos mediante el comando “**systemctl status mariadb**” para corroborar que el servicio de Maria DB este activo.

```
* mariadb.service - MariaDB 10.3.39 database server
   Loaded: loaded (/lib/systemd/system/mariadb.service; enabled; vendor prese
   Active: active (running) since Wed 2024-06-19 20:47:22 UTC; 1min 28s ago
     Docs: man:mysql(8)
           https://mariadb.com/kb/en/library/systemd/
   Main PID: 10094 (mysqld)
    Status: "Taking your SQL requests now..."
     Tasks: 31 (limit: 4465)
    Memory: 26.1M
       CPU: 902ms
    CGroup: /system.slice/mariadb.service
           └─10094 /usr/sbin/mysqld

Jun 19 20:47:16 42935962DB systemd[1]: Starting MariaDB 10.3.39 database server>
Jun 19 20:47:22 42935962DB systemd[1]: Started MariaDB 10.3.39 database server.
Jun 19 20:47:30 42935962DB /etc/mysql/debian-start[10132]: Looking for 'mysql' >
Jun 19 20:47:30 42935962DB /etc/mysql/debian-start[10132]: Looking for 'mysqlch>
Jun 19 20:47:30 42935962DB /etc/mysql/debian-start[10132]: This installation of>
Jun 19 20:47:30 42935962DB /etc/mysql/debian-start[10132]: There is no need to>
lines 1-19
```

5. Una vez realizado todo lo anterior, ingresamos a Maria DB con el comando “**mariadb**”, una vez dentro, procedemos a crear la base de datos que utilizaremos, en este caso virtualizaciondb, y luego la seleccionamos mediante el comando “**USE virtualizaciondb;**”

```
MariaDB [(none)]> CREATE DATABASE IF NOT EXISTS virtualizaciondb;
Query OK, 1 row affected (0.075 sec)

MariaDB [(none)]> USE virtualizaciondb;
Database changed
MariaDB [virtualizaciondb]> 
```

6. Ahora procedemos a crear las tablas de la base de datos, en nuestro caso son dos, la tabla estudiante y la tabla usuarios. Ejecutamos el script que se dejara a continuación para crear las tablas:

```
CREATE TABLE IF NOT EXISTS estudiante (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombrecompleto VARCHAR(255) NOT NULL,
  email VARCHAR(320) NOT NULL UNIQUE,
  edad INT NOT NULL,
  github VARCHAR(255),
  descripcion VARCHAR(500) NOT NULL,
  legajo INT NOT NULL UNIQUE
);
```

```
CREATE TABLE IF NOT EXISTS usuario (
  id INT AUTO_INCREMENT PRIMARY KEY,
  email VARCHAR(255) NOT NULL UNIQUE,
  password VARCHAR(255) NOT NULL,
  rol ENUM('admin', 'usuario') DEFAULT 'usuario'
);
```

);

```
MariaDB [virtualizaciondb]> CREATE TABLE IF NOT EXISTS estudiante (  
E    ->      id INT AUTO_INCREMENT PRIMARY KEY,  
      ->      nombrecompleto VARCHAR(255) NOT NULL,  
em   ->      email VARCHAR(320) NOT NULL UNIQUE,  
      ->      edad INT NOT NULL,  
b    ->      github VARCHAR(255),  
      ->      descripcion VARCHAR(500) NOT NULL,  
I    ->      legajo INT NOT NULL UNIQUE  
      -> );  
Query OK, 0 rows affected (1.082 sec)
```

Una vez que ejecutamos lo anterior, verificamos que se crearon las tablas correctamente con el comando show tables.

```
MariaDB [virtualizaciondb]> show tables;  
+-----+  
| Tables_in_virtualizaciondb |  
+-----+  
| estudiante                  |  
| usuario                    |  
+-----+  
2 rows in set (0.000 sec)
```

7. Es necesario crear un usuario para acceder a la base de datos, por ello, creamos un usuario "root". La creación del mismo se realizó mediante los siguientes comandos:

```
CREATE USER 'root'@'%' IDENTIFIED BY 'root';  
GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' WITH GRANT OPTION;  
FLUSH PRIVILEGES;
```

Luego de ejecutar lo anterior, verificamos que el usuario se creó correctamente por medio de la consulta:

```
SELECT User, Host FROM mysql.user;
```

```

MariaDB [(none)]> CREATE USER 'root'@'%' IDENTIFIED BY 'root';
Query OK, 0 rows affected (0.137 sec)

MariaDB [(none)]> GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' WITH GRANT OPTION;
Query OK, 0 rows affected (0.000 sec)

MariaDB [(none)]> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.048 sec)

MariaDB [(none)]> SELECT User, Host FROM mysql.user;
+-----+-----+
| User | Host |
+-----+-----+
| root | %    |
| root | localhost |
+-----+-----+
2 rows in set (0.000 sec)

```

8. Ya tenemos lista la configuración de la base de datos, ahora procederemos a instalar el servidor web. Elegimos instalar apache, para ello ejecutamos el comando **“apt-get install apache2 -y”** y luego **“systemctl status apache2”** para verificar que el servicio se instaló correctamente y este activo.
9. Como trabajamos con node, necesitamos instalar **“curl”** que es una herramienta para transferir datos desde o hacia un servidor, el mismo nos ayudara a instalar nodejs. Para ello ejecutamos el comando **“apt-get install curl -y”**
10. Una vez instalado curl, procederemos a ejecutar el siguiente comando **“curl -fsSL https://deb.nodesource.com/setup_20.x -o install.sh”** el cual descargara el instalador de nodejs desde la url proporcionada (obtenida de la página de node) y lo guardara en un archivo llamado **“install.sh”**.
11. Luego de realizado lo anterior, ejecutamos el comando **“ll”** y verificamos que se encuentre el archivo **install.sh** que es un script de instalación. Ejecutaremos ese script con el comando **“bash install.sh”**.
12. Cuando termina la instalación ejecutada anteriormente, ya podremos instalar nodejs mediante el comando **“apt-get install nodejs”**. Una vez finalizada la instalación de node, verificamos que se instaló correctamente mediante los comandos **“node -v”** y **“npm -v”**.

```

root@42935962DB:~# node -v
v20.14.0
root@42935962DB:~# npm -v
10.7.0

```

13. Ahora necesitamos instalar git, ya que nuestro proyecto de Backend se encuentra en un repositorio en GitHub, para ello ejecutamos el comando **“apt-get install git -y”**. Una vez instalado, nos movemos a la carpeta **/var/www** con el comando **“cd /var/www”** y clonamos nuestro proyecto con el comando **“git clone url_repositorio nombrecarpeta”**.

```

root@42935962DB:/var/www# git clone https://github.com/Agustin030s/backendblog.git backendblog_1
Cloning into 'backendblog_1'...
remote: Enumerating objects: 64, done.
remote: Counting objects: 100% (64/64), done.
remote: Compressing objects: 100% (40/40), done.
remote: Total 64 (delta 20), reused 64 (delta 20), pack-reused 0
Unpacking objects: 100% (64/64), 28.78 KiB | 1.20 MiB/s, done.

```

14. Ahora, posicionados en “**/var/www**” nos movemos hacia la carpeta donde se clono nuestro proyecto, en este caso “**backendblog**”, ejecutamos el comando “**cd backendblog**” y una vez dentro ejecutamos “**npm install**” para instalar las dependencias del proyecto.

```

root@42935962DB:/var/www# cd backendblog
root@42935962DB:/var/www/backendblog# npm install

```

15. Una vez realizado lo anterior, ejecutamos “**cd ..**” para volver hacia atrás, nos posicionaremos en la ubicación “**/var/www**”, estando allí, borramos la carpeta llamada “**html**” y renombramos la carpeta “**backendblog**” donde tenemos nuestro Backend con el nombre “**html**”, esto es para que apache reconozca nuestro proyecto.

```

root@42935962DB:/var/www# rm -r html
root@42935962DB:/var/www# mv backendblog html
root@42935962DB:/var/www# ll
total 12
drwxr-xr-x  3 root root 4096 Jun 19 20:44 ./
drwxr-xr-x 12 root root 4096 Jun 19 20:32 ../
drwxr-xr-x  6 root root 4096 Jun 19 20:39 html/
root@42935962DB:/var/www#

```

16. Ahora que ya tenemos todo instalado y configurado, ejecutamos el proyecto por primera vez, nos dirigimos hacia la carpeta “**html**” y lo ejecutamos mediante el comando “**node index.js**”. En nuestro caso, verificamos que estaba funcionando correctamente, ya que nos mostro el puerto donde se estaba ejecutando, como así también que pudo crear el usuario administrador y el estudiante por defecto.

```

root@42935962DB:/var/www/html# node index.js
Estoy en el puerto 3001
Usuario administrador inicializado correctamente
Estudiante por defecto inicializado correctamente

```

Para verificar que realmente se creó el usuario y el estudiante, ingresamos a María DB con el comando: **mariadb -u root -p**

Podemos observar que tanto el estudiante, como el usuario administrador, se crearon correctamente.

```

MariaDB [virtualizaciondb]> select * from estudiante;
+-----+-----+-----+-----+-----+-----+
| id | nombrecompleto | email | edad | github | descripcion |
+-----+-----+-----+-----+-----+-----+
| 1 | Sebastian Agustin Maza | sebas030.maza@gmail.com | 23 | Agustin030s | Soy estudiante de 5to año de Ingeniería en Sistemas de Información, apasionado por la tecnología y siempre en constante aprendizaje. Me dedico a innovar y desarrollar soluciones eficientes, manteniéndome al día con los últimos avances del sector. | 50240 |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.000 sec)

MariaDB [virtualizaciondb]> select * from usuario;
+-----+-----+-----+-----+
| id | email | password | rol |
+-----+-----+-----+-----+
| 1 | admin@admin.com | $2b$10$Q6izpFGi2MwbrDf3F1pXLuJ.PTs/HSayobtrLcornh6lk8gYtKiEO | admin |
+-----+-----+-----+-----+
1 row in set (0.000 sec)

```

Configuración del contenedor de Frontend

Para el contenedor de Frontend, seguiremos los mismos pasos que en el contenedor de Backend, con la excepción de que no instalaremos nodejs ni María DB, pero si instalaremos curl, git y apache2.

Una vez instalado lo especificado en el parrado anterior, nos dirigiremos a la carpeta **“/var/www”**, allí eliminaremos la carpeta **“html”** que viene por defecto con el comando **“rm -r html”**.

Ahora, clonaremos nuestro proyecto de Frontend en la dirección **“/var/www”** para ello ejecutamos el comando **“git clone url_repositorio nombrecarpeta”**, en nombre de la carpeta le pondremos **“html”** para que la misma sea reconocida por el servidor web apache.

Como el Frontend fue realizado en React, antes de subirlo a git, se ejecutó el comando **“npm run build”** para dejar el proyecto listo para producción y ser desplegado.

También, al estar manejando las rutas con React Router Dom, en necesario realizar unas configuraciones en un archivo que se debe llamar **“.htaccess”**. Para ello nos dirigimos a la ubicación **/var/www/html** con el comando que venimos viendo anteriormente.

Una vez allí, ejecutamos **“nano .htaccess”** el cual nos creará el archivo y se nos abrirá con el editor de texto nano, en el se coloca lo siguiente:

```

GNU nano 4.8                               .htaccess                               Modified
<IfModule mod_rewrite.c>
RewriteEngine On
RewriteBase /
RewriteRule ^index\.html$ - [L]
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule . /index.html [L]
</IfModule>

```

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^G Cur Pos M-U Undo M-A Mark Text
 ^X Exit ^R Read File ^\ Replace ^U Paste Text ^T To Spell ^_ Go To Line M-E Redo M-C Copy Text

Con esto, ya tendremos desplegada nuestra aplicación de Frontend en el contenedor.

Comunicación entre contenedores

Una vez configurado los dos contenedores, podemos probar la comunicación entre ambos y también el correcto funcionamiento de la api desarrollada en nodejs.

Para ello utilizaremos “curl” en el contenedor de Frontend, pero primero debemos conocer la ip del contenedor de Backend. Para ello, en el contenedor de Backend ejecutamos el comando “ip address” el cual nos mostrara la dirección ip de nuestro contenedor.

Obtenida la dirección ip, que en este caso fue la dirección “192.168.77.209”, en el contenedor de Backend procedemos a iniciar la api, nos dirigimos a la carpeta “/var/www/html” y ejecutamos “node index.js” como lo hicimos anteriormente.

Una vez hecho esto, en nuestro contenedor Frontend ejecutaremos el comando “curl 192.168.77.209:3001/api/students”, el cual realizará una solicitud GET a la ruta “/api/students/” y nos traerá los datos de los estudiantes que se encuentran en la base de datos.

```
root@42935962A:~# curl 192.168.77.209:3001/api/students/
[{"id":1,"nombrecompleto":"Sebastian Agustin Maza","email":"sebas030.maza@gmail.com","edad":23,"github":"Agustin030s","descripcion":"Soy estudiante de 5to año de Ingeniería en Sistemas de Información, apasionado por la tecnología y siempre en constante aprendizaje. Me dedico a innovar y desarrollar soluciones eficientes, manteniéndome al día con los últimos avances del sector.","legajo":50240}]root@42935962A:~#
```

En el contenedor de Backend, mediante la dependencia instalada en el proyecto llamada Morgan, podemos observar que nos muestra que se realizó la solicitud GET a la ruta “/api/students/” y que se devolvió el estado 200, lo que significa que todo salió bien.

```
root@42935962DB:/var/www/html# node index.js
Estoy en el puerto 3001
Usuario administrador inicializado correctamente
Estudiante por defecto inicializado correctamente
GET /ports 200 24.424 ms - 20
GET /api/students/ 200 42.901 ms - 404
```

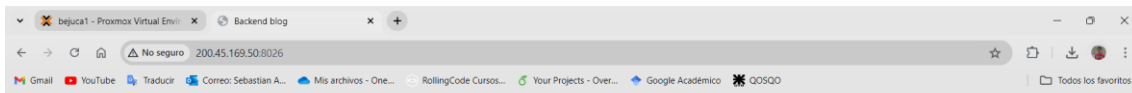
Redirección de puertos y pruebas

Para poder acceder a los contenedores desde el exterior, es decir desde la red, es necesario solicitar una redirección de puertos al administrador del clúster. Para ello necesitamos proporcionarle la ip del contenedor y el puerto que deseamos que sea redirigido.

En el caso del contenedor de Frontend, necesitamos pedir la redirección del puerto 80 que es donde se ejecuta por defecto apache. Para el contenedor de Backend, necesitamos el puerto 3001, ya que es donde tenemos configurado para que escuche nuestra aplicación en Node.js.

Cuando se nos proporcionan los puertos a los cuales se redirecciona, accedemos a los mismos mediante la siguiente dirección “200.42.169.50: puerto”.

Una vez obtenida la redirección de puertos, realizamos la prueba del Backend desde el navegador, ingresando a la ip mas el puerto asignado para la redirección.



Backend blog personal

Como podemos observar, ingresa correctamente. Ahora procedemos a encender el contenedor de Frontend e ingresar de la misma manera, como resultado obtenemos nuestra aplicación funcionando, obteniendo los datos desde el contenedor de Backend.

