

Universidad Nacional
ARTURO JAURETCHE

SISTEMAS DISTRIBUIDOS

LABORATORIO N° 3

Blockchain

Autor
Emiliano SALVATORI

5 de diciembre de 2020

Índice

1. Finalidad	2
2. Introduccion	2
3. Precedentes	2
4. Configuración para Entorno virtual	4
4.1. Instalación de curl	5
4.2. Instalación de flask y request	6
4.3. Modificación de los archivos fuente para el cliente	7
5. Prueba de la máquina virtual	8
6. Ejercicio nº 1	9
6.1. Implementación en dos máquinas	9
6.1.1. Configuración de la máquina Servidor	10
6.1.2. Configuración de la máquina Cliente	11
6.1.3. Minado mediante el Navegador Brave	12
7. Ejercicio nº 2	13
7.1. Configuración de los Nodos Distribuidos	13
7.2. Procedimiento	15
8. Conclusiones	21
Bibliografía	21

1. Finalidad

En el siguiente informe se detalla lo realizado como parte del laboratorio de la materia **Sistemas Distribuidos** para la **Comisión n° 1**.

La implementación realizada para este trabajo se basa en los siguientes objetivos:

- Comprender la arquitectura distribuida de tipo BlockChain.
- Consenso Distribuidos.
- Minado.
- Prueba de trabajo.

Para este propósito se utilizan las máquinas virtuales modificadas en el Laboratorio n° 1. La cual contiene las siguientes características:

- Versión de linux Debian 10 sin entorno grafico.
- Usuario: linuxtest
Password: 123456
- Usuario: root
Password: 123456

2. Introduccion

En el siguiente trabajo, se implementará de forma sencilla un sistema de Block-chain, escrito bajo el lenguaje de programación Python. Para ello será necesario contar con algunas herramientas instaladas previamente, que se explican en la sección 3. Asimismo el programa permite crear puntos de acceso para diferentes funciones, como puede ser: añadir una única transacción (explicada en la sección 6.1); o bien, poder replicar la base de datos mediante el programa *curl* y luego mediante el programa *flask* (explicado en la sección 7) pudiendo de esta manera soportar múltiples transacciones para que la base de datos se replique en múltiples máquinas para crear una red de tipo descentralizada.

El código permite abordar de manera fácil y comprensible, la manera que tienen de trabajar estos tipos de sistemas, permitiendo tener una interfaz accesible al usuario, permitiéndole almacenar información para cualquier uso, de forma segura y como se hizo mención anteriormente, de forma descentralizada.

3. Precedentes

Antes de abordar de lleno el Laboratorio, se procede a visualizar que todo lo necesario para llevarlo a cabo, esté funcionando correctamente. Para ello se comprueba que las configuraciones de las máquinas virtuales del Laboratorio n° 1. Para este propósito se debe tener acceso a internet para poder instalar los programas requeridos, para ello se modifica el archivo *interfaces*

```

linuxtest@NodoMaestro:~$ cat /etc/network/interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
#allow-hotplug enp0s3
#iface enp0s3 inet dhcp
auto enp0s3
iface enp0s3 inet dhcp
#address 192.168.2.156
#netmask 255.255.255.0

#Modificado por Emiliano
#iface enp0s3 inet static
#address 192.168.0.10
#netmask 255.255.255.0

linuxtest@NodoMaestro:~$

```

Se procede a instalar el programa *unzip* para poder desempaquetar los códigos necesarios con el siguiente comando:

```
$ sudo apt-get install unzip
```

```

linuxtest@NodoMaestro:~$ sudo apt-get install unzip
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Paquetes sugeridos:
  zip
Se instalarán los siguientes paquetes NUEVOS:
  unzip
0 actualizados, 1 nuevos se instalarán, 0 para eliminar y 41 no actualizados.
Se necesita descargar 172 kB de archivos.
Se utilizarán 580 kB de espacio de disco adicional después de esta operación.
Des:1 http://deb.debian.org/debian buster/main amd64 unzip amd64 6.0-23+deb10u1 [172 kB]
Descargados 172 kB en 1s (286 kB/s)
Seleccionando el paquete unzip previamente no seleccionado.
(Leyendo la base de datos ... 67986 ficheros o directorios instalados actualmente.)
Preparando para desempaquetar .../unzip_6.0-23+deb10u1_amd64.deb ...
Desempaquetando unzip (6.0-23+deb10u1) ...
Configurando unzip (6.0-23+deb10u1) ...
Procesando disparadores para mime-support (3.62) ...
linuxtest@NodoMaestro:~$

```

Luego se procede a instalar el paquete *python3-venv*, el cual permite crear entornos virtuales ligeros con sus propios directorios, aislando el entorno del sistema que lo hospeda. Se procede a ejecutar el siguiente comando:

```
$ sudo apt-get install python3-venv
```

```
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
ca-certificates libpython3.7-minimal libpython3.7-stdlib openssl python-pip-whl
python3-distutils python3-lib2to3 python3.7 python3.7-minimal python3.7-venv
Paquetes sugeridos:
python3.7-doc binfmt-support
Se instalarán los siguientes paquetes NUEVOS:
ca-certificates openssl python-pip-whl python3-distutils python3-lib2to3 python3-venv
python3.7-venv
Se actualizarán los siguientes paquetes:
libpython3.7-minimal libpython3.7-stdlib python3.7 python3.7-minimal
4 actualizados, 7 nuevos se instalarán, 0 para eliminar y 37 no actualizados.
Se necesita descargar 7,201 kB de archivos.
Se utilizarán 4,962 kB de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n] S
Des:1 http://deb.debian.org/debian buster/main amd64 python3.7 amd64 3.7.3-2+deb10u2 [330 kB]
Des:2 http://deb.debian.org/debian buster/main amd64 libpython3.7-stdlib amd64 3.7.3-2+deb10u2 [1.7
2 kB]
Des:3 http://deb.debian.org/debian buster/main amd64 python3.7-minimal amd64 3.7.3-2+deb10u2 [1.731
kB]
Des:4 http://deb.debian.org/debian buster/main amd64 libpython3.7-minimal amd64 3.7.3-2+deb10u2 [58
kB]
Des:5 http://deb.debian.org/debian buster/main amd64 openssl amd64 1.1.1d-0+deb10u3 [844 kB]
Des:6 http://deb.debian.org/debian buster-updates/main amd64 ca-certificates all 20200601~deb10u1 [
59 kB]
Des:7 http://deb.debian.org/debian buster/main amd64 python-pip-whl all 18.1-5 [1.591 kB]
Des:8 http://deb.debian.org/debian buster/main amd64 python3-lib2to3 all 3.7.3-1 [76,7 kB]
Des:9 http://deb.debian.org/debian buster/main amd64 python3-distutils all 3.7.3-1 [142 kB]
Des:10 http://deb.debian.org/debian buster/main amd64 python3.7-venv amd64 3.7.3-2+deb10u2 [6.148 B]
Des:11 http://deb.debian.org/debian buster/main amd64 python3-venv amd64 3.7.3-1 [1.180 B]
Descargados 7,201 kB en 2s (3,408 kB/s)
Preconfigurando paquetes ...
(Leyendo la base de datos.... 68004 ficheros o directorios instalados actualmente.)
Preparando para desempaquetar .../00-python3.7_3.7.3-2+deb10u2_amd64.deb ...
Desempaquetando python3.7 (3.7.3-2+deb10u2) sobre (3.7.3-2+deb10u1) ...
```

Se comprueba que todo finalizó de forma correcta:

```
Configurando libpython3.7-minimal:amd64 (3.7.3-2+deb10u2) ...
Configurando python3.7-minimal (3.7.3-2+deb10u2) ...
Configurando openssl (1.1.1d-0+deb10u3) ...
Configurando python3-lib2to3 (3.7.3-1) ...
Configurando python3-distutils (3.7.3-1) ...
Configurando libpython3.7-stdlib:amd64 (3.7.3-2+deb10u2) ...
Configurando ca-certificates (20200601~deb10u1) ...
Updating certificates in /etc/ssl/certs...
126 added, 0 removed; done.
Configurando python-pip-whl (18.1-5) ...
Configurando python3.7 (3.7.3-2+deb10u2) ...
Configurando python3.7-venv (3.7.3-2+deb10u2) ...
Configurando python3-venv (3.7.3-1) ...
Procesando disparadores para mime-support (3.62) ...
Procesando disparadores para ca-certificates (20200601~deb10u1) ...
Updating certificates in /etc/ssl/certs..
0 added, 0 removed; done.
Running hooks in /etc/ca-certificates/update.d...
done.
linuxtest@NodoMaestro:~$ _
```

4. Configuración para Entorno virtual

Como se puede apreciar a continuación, se crea un directorio para el Laboratorio nº 3 y se procede a descomprimir el paquete que contiene los archivos base:

```
linuxtest@NodoMaestro:~$  
linuxtest@NodoMaestro:~$ mkdir laboratorio3  
linuxtest@NodoMaestro:~$  
linuxtest@NodoMaestro:~$ ls -l  
total 24  
drwxr-xr-x 2 linuxtest linuxtest 4096 nov 15 15:36 laboratorio3  
-rw-r--r-- 1 linuxtest linuxtest 5806 oct 1 18:25 matrices.c  
drwxr-xr-x 1 linuxtest linuxtest 4096 nov 15 15:09 mnt  
-rw-r--r-- 1 linuxtest linuxtest 73 sep 27 21:19 mount.sh  
drwxr-xr-x 3 linuxtest linuxtest 4096 oct 26 20:55 share  
linuxtest@NodoMaestro:~$  
linuxtest@NodoMaestro:~$ cd mnt/Lab3/  
linuxtest@NodoMaestro:~/mnt/Lab3$  
linuxtest@NodoMaestro:~/mnt/Lab3$ ls -l  
total 184  
-rw-r--r-- 1 linuxtest linuxtest 185895 nov 15 10:25 python_blockchain_app-master.zip  
linuxtest@NodoMaestro:~/mnt/Lab3$  
linuxtest@NodoMaestro:~/mnt/Lab3$ cp python_blockchain_app-master.zip ../../laboratorio3/  
linuxtest@NodoMaestro:~/mnt/Lab3$  
linuxtest@NodoMaestro:~/mnt/Lab3$ cd -  
/home/linuxtest  
linuxtest@NodoMaestro:~$  
linuxtest@NodoMaestro:~$ ls -l laboratorio3/  
total 184  
-rw-r--r-- 1 linuxtest linuxtest 185895 nov 15 15:37 python_blockchain_app-master.zip  
linuxtest@NodoMaestro:~$
```

Se puede apreciar a continuación que los archivos fueron descomprimidos de forma correcta:

```
linuxtest@NodoMaestro:~/laboratorio3/python_blockchain_app-master$ ls -la  
total 52  
drwxr-xr-x 4 linuxtest linuxtest 4096 sep 17 08:58 .  
drwxr-xr-x 3 linuxtest linuxtest 4096 nov 15 15:40 ..  
drwxr-xr-x 3 linuxtest linuxtest 4096 sep 17 08:58 app  
-rw-r--r-- 1 linuxtest linuxtest 707 sep 17 08:58 CONTRIBUTING.md  
-rw-r--r-- 1 linuxtest linuxtest 135 sep 17 08:58 .deepsources.toml  
-rw-r--r-- 1 linuxtest linuxtest 15 sep 17 08:58 .gitignore  
-rw-r--r-- 1 linuxtest linuxtest 10057 sep 17 08:58 node_server.py  
-rw-r--r-- 1 linuxtest linuxtest 3226 sep 17 08:58 README.md  
-rw-r--r-- 1 linuxtest linuxtest 26 sep 17 08:58 requirements.txt  
-rw-r--r-- 1 linuxtest linuxtest 41 sep 17 08:58 run_app.py  
drwxr-xr-x 2 linuxtest linuxtest 4096 sep 17 08:58 screenshots  
linuxtest@NodoMaestro:~/laboratorio3/python_blockchain_app-master$ _
```

4.1. Instalación de curl

A continuación instalamos el paquete curl, el cual transfiere datos desde o hacia un servidor, utilizando alguno de los siguientes protocolos: HTTP, HTTPS, FTP, FTPS, SCP, SFTP, TFTP, DICT, TELNET, LDAP o FILE.

```
linuxtest@NodoMaestro:~$  
linuxtest@NodoMaestro:~$ sudo apt-get install curl  
[sudo] password for linuxtest:  
Leyendo lista de paquetes... Hecho  
Creando árbol de dependencias  
Leyendo la información de estado... Hecho  
Se instalarán los siguientes paquetes adicionales:  
  libcurl4 libnghttp2-14 libpsl5 librtmp1 libssh2-1 publicsuffix  
Se instalarán los siguientes paquetes NUEVOS:  
  curl libcurl4 libnghttp2-14 libpsl5 librtmp1 libssh2-1 publicsuffix  
0 actualizados, 7 nuevos se instalarán, 0 para eliminar y 37 no actualizados.  
Se necesita descargar 1.051 kB de archivos.  
Se utilizarán 2.163 kB de espacio de disco adicional después de esta operación.  
¿Desea continuar? [S/n] S_
```

```
Preparando para desempaquetar .../0-libnghttp2-14_1.36.0-2+deb10u1_amd64.deb ...
Desempaquetando libnghttp2-14:amd64 (1.36.0-2+deb10u1) ...
Seleccionando el paquete libps15:amd64 previamente no seleccionado.
Preparando para desempaquetar .../1-libps15_0.20.2-2_amd64.deb ...
Desempaquetando libps15:amd64 (0.20.2-2) ...
Seleccionando el paquete librtmp1:amd64 previamente no seleccionado.
Preparando para desempaquetar .../2-librtmp1_2.4+20151223.gitfa8646d.1-2_amd64.deb ...
Desempaquetando librtmp1:amd64 (2.4+20151223.gitfa8646d.1-2) ...
Seleccionando el paquete libssh2-1:amd64 previamente no seleccionado.
Preparando para desempaquetar .../3-libssh2-1_1.8.0-2.1_amd64.deb ...
Desempaquetando libssh2-1:amd64 (1.8.0-2.1) ...
Seleccionando el paquete libcurl4:amd64 previamente no seleccionado.
Preparando para desempaquetar .../4-libcurl4_7.64.0-4+deb10u1_amd64.deb ...
Desempaquetando libcurl4:amd64 (7.64.0-4+deb10u1) ...
Seleccionando el paquete curl previamente no seleccionado.
Preparando para desempaquetar .../5-curl_7.64.0-4+deb10u1_amd64.deb ...
Desempaquetando curl (7.64.0-4+deb10u1) ...
Seleccionando el paquete publicsuffix previamente no seleccionado.
Preparando para desempaquetar .../6-publicsuffix_20190415.1030-1_all.deb ...
Desempaquetando publicsuffix (20190415.1030-1) ...
Configurando libps15:amd64 (0.20.2-2) ...
Configurando libnghttp2-14:amd64 (1.36.0-2+deb10u1) ...
Configurando librtmp1:amd64 (2.4+20151223.gitfa8646d.1-2) ...
Configurando libssh2-1:amd64 (1.8.0-2.1) ...
Configurando publicsuffix (20190415.1030-1) ...
Configurando libcurl4:amd64 (7.64.0-4+deb10u1) ...
Configurando curl (7.64.0-4+deb10u1) ...
Procesando disparadores para libc-bin (2.28-10) ...
linuxtest@NodoMaestro:~$ _
```

A continuación invocamos la creación de los entornos virtuales en la carpeta donde se encuentran los códigos fuente, con el siguiente comando:

```
$ python3 -m venv venv
```

```
linuxtest@NodoMaestro:~/laboratorio3/python_blockchain_app-master$
linuxtest@NodoMaestro:~/laboratorio3/python_blockchain_app-master$ python3
python3      python3.7  python3.7m  python3m
linuxtest@NodoMaestro:~/laboratorio3/python_blockchain_app-master$ python3 -m venv venv
linuxtest@NodoMaestro:~/laboratorio3/python_blockchain_app-master$
linuxtest@NodoMaestro:~/laboratorio3/python_blockchain_app-master$ _
```

4.2. Instalación de flask y request

Una vez realizado esto, se ejecuta la instalación de *flask* y de *request* mediante la ejecución del siguiente comando:

```
$ pip install -r requirements.txt
```

Se visualiza la finalización en la instalación de los paquetes y sus dependencias:

```
Collecting urllib3<1.27,>=1.21.1 (from requests==2.22->-r requirements.txt (line 2))
  Downloading https://files.pythonhosted.org/packages/f5/71/45d36a8df68f3ebb098d6861b2c017f3d094530fb98fa61d4dc43e69b9/urllib3-1.26.2-py2.py3-none-any.whl (136kB)
  100% |#####| 143kB 4.7MB/s
Collecting chardet<4,>=3.0.2 (from requests==2.22->-r requirements.txt (line 2))
  Downloading https://files.pythonhosted.org/packages/bc/a9/01ffebfb562e4274b6487b4bb1dddec7ca55ec70b22e4c51f14098443b8/chardet-3.0.4-py2.py3-none-any.whl (133kB)
  100% |#####| 143kB 3.0MB/s
Collecting certifi<2017.4.17 (from requests==2.22->-r requirements.txt (line 2))
  Downloading https://files.pythonhosted.org/packages/c1/6f/3d85f0850962279a7e4c622695d7b3171e95ac0308a57d3b29738b27149/certifi-2020.11.8-py2.py3-none-any.whl (155kB)
  100% |#####| 163kB 5.4MB/s
Collecting idna<3,>=2.5 (from requests==2.22->-r requirements.txt (line 2))
  Downloading https://files.pythonhosted.org/packages/a2/38/928ddce2273eaa564f6f50de919327bf3a00f0b5baba8dfa9460f3a8a8/idna-2.10-py2.py3-none-any.whl (58kB)
  100% |#####| 61kB 1.9MB/s
Collecting MarkupSafe<0.23 (from Jinja2==2.10.1->Flask==1.1->-r requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/98/7b/ff284bd8c80654e471b76962a9b43cc5d03e715048d96f4619df8d420/MarkupSafe-1.1.1-cp37m-manylinux1_x86_64.whl
Installing collected packages: Werkzeug, MarkupSafe, Jinja2, itsdangerous, click, Flask, urllib3, chardet, certifi, idna, requests
Successfully installed Flask-1.1.2 Jinja2-2.11.2 MarkupSafe-1.1.1 Werkzeug-1.0.1 certifi-2020.11.8 chardet-3.0.4 click-7.1.2 idna-2.10 itsdangerous-1.1.0 requests-2.25.0 urllib3-1.26.2
(venv) linuxtest@NodoMaestro:~/laboratorio3/python_blockchain_app-master$
```

4.3. Modificación de los archivos fuente para el cliente

Para poder utilizar el front-end de la aplicación se requiere modificar el archivo *app/views.py* modificando el contenido de la variable *CONNECTED_NODE_ADDRESS* por la dirección IP que se tenga asignada en la máquina que se quiera ejecutar, junto con el puerto por el cual estará escuchando el servidor.

Para saber qué dirección IP tiene asignada la máquina se ingresa:

```
$ hostname -I
```

```
Debian GNU/Linux 10 NodoMaestro tty2
NodoMaestro login: linuxtest
Password:
Last login: Thu Nov 26 21:41:35 -03 2020 on tty3
Linux NodoMaestro 4.19.0-6-amd64 #1 SMP Debian 4.19.67-2+deb10u2 (2019-11-11) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
linuxtest@NodoMaestro:~$ hostname -I
192.168.100.22 2803:9800:a011:85d0:a00:27ff:fe18:2992
linuxtest@NodoMaestro:~$
```

Y a continuación se ingresa esa misma información dentro del archivo mencionado:


```

1 import datetime
2 import json
3
4 import requests
5 from flask import render_template, redirect, request
6
7 from app import app
8
9 # The node with which our application interacts, there can be multiple
10 # such nodes as well.
11 # CONNECTED_NODE_ADDRESS = "http://127.0.0.1:8000"
12 CONNECTED_NODE_ADDRESS = "http://192.168.100.22:8000"
13
14 posts = []
15
16
17 def fetch_posts():
18     """
19     Function to fetch the chain from a blockchain node, parse the

```

Luego también se debe modificar el archivo *run_app.py* ingresando lo siguiente:

```

1 from app import app
2
3 #app.run(debug=True)
4 app.run(debug=True, host='0.0.0.0')

```

5. Prueba de la máquina virtual

Para comprobar que todo lo anterior resultó de forma exitosa, se procede a abrir dos terminales en una misma máquina virtual mediante la combinación de teclas **Alt+F1** (tty1) y **Alt+F2** (tty2).

Se procede a invocar al módulo *venv* para la primer terminal (tty1) dentro del directorio donde se descomprimieron los archivos de código fuente y activarlo mediante el siguiente comando:

```
$ source venv/bin/activate
```

Y luego se da ejecución al cliente con el siguiente comando:

```
$ python run_app.py
```

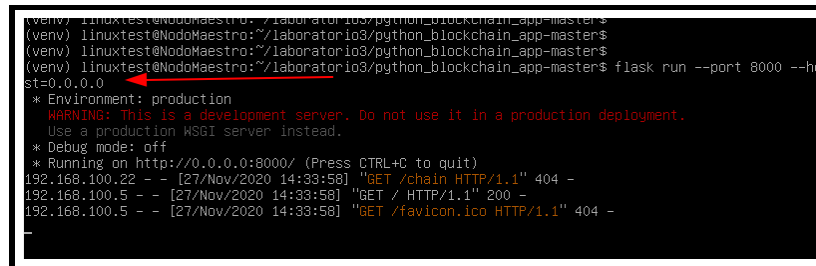
```

linuxtest@NodoMaestro:~/laboratorio3/python_blockchain_app-master$
linuxtest@NodoMaestro:~/laboratorio3/python_blockchain_app-master$ source venv/bin/activate
(venv) linuxtest@NodoMaestro:~/laboratorio3/python_blockchain_app-master$
(venv) linuxtest@NodoMaestro:~/laboratorio3/python_blockchain_app-master$
(venv) linuxtest@NodoMaestro:~/laboratorio3/python_blockchain_app-master$
(venv) linuxtest@NodoMaestro:~/laboratorio3/python_blockchain_app-master$ python run_app.py
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 543-936-497
^_

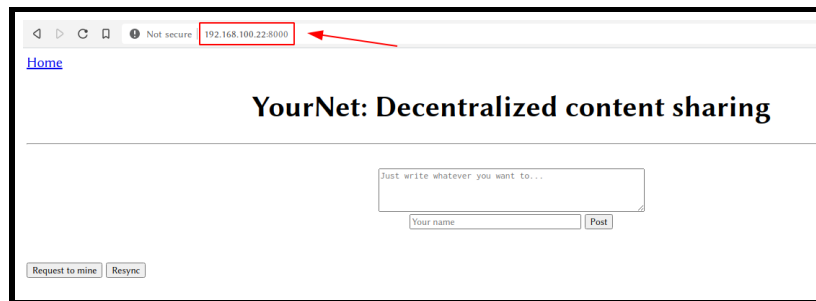
```

Se procede a abrir otra terminal con **Alt+F2** (tty2), y luego de ingresar el mismo primer comando que en la terminal tty1 y se ingresa el siguiente comando:

```
$ flask run --port 8000 --host=0.0.0.0
```

A terminal window showing the execution of the 'flask run' command. The prompt is '(venv) linuxtest@NodoMaestro: ~/laboratorio3/python_blockchain_app-master\$'. The command entered is 'flask run --port 8000 --host=0.0.0.0'. The output shows the environment is production, a warning not to use it in production, debug mode is off, and the server is running on http://0.0.0.0:8000/. It also shows some HTTP requests being received, such as 'GET /chain HTTP/1.1' 404 - and 'GET / HTTP/1.1' 200 -.

Ahora bien, desde la máquina huésped donde se está ejecutando el programa *Virtual Box*, procedemos a levantar un navegador (en nuestro caso Brave) e ingresar la dirección que fue ingresada en el archivo *app/views.py* junto con el puerto donde está escuchando el servidor:



6. Ejercicio nº 1

6.1. Implementación en dos máquinas

Para llevar a cabo el ejercicio propuesto por la cátedra, se deben configurar dos máquinas virtuales. Una de ellas será la configurada en la sección 3, y la otra será una copia exacta (clonada) para mantener las mismas configuraciones que en la primera.

La configuración para la primera máquina será la siguiente:

- **Nombre de la máquina:** NodoMaestro
- **Tipo de Servicio:** Servidor
- **Dirección IP:** 192.168.100.22

```
Debian GNU/Linux 10 NodoMaestro tty1
NodoMaestro login: linuxtest
Password:
Last login: Sat Nov 28 01:26:46 -03 2020 on tty1
Linux NodoMaestro 4.19.0-6-amd64 #1 SMP Debian 4.19.67-2+deb10u2 (2019-11-11) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
^[[flinuxtest@NodoMaestro:~$ cd laboratorio3/python_blockchain_app-master
linuxtest@NodoMaestro:~/laboratorio3/python_blockchain_app-master$ ^Ch linuxtest@LinuxNo
linuxtest@NodoMaestro:~/laboratorio3/python_blockchain_app-master$
linuxtest@NodoMaestro:~/laboratorio3/python_blockchain_app-master$
linuxtest@NodoMaestro:~/laboratorio3/python_blockchain_app-master$
linuxtest@NodoMaestro:~/laboratorio3/python_blockchain_app-master$
linuxtest@NodoMaestro:~/laboratorio3/python_blockchain_app-master$
linuxtest@NodoMaestro:~/laboratorio3/python_blockchain_app-master$ hostname -I
192.168.100.22 2803:9800:a011:85d0:a00:27ff:fe18:2992
linuxtest@NodoMaestro:~/laboratorio3/python_blockchain_app-master$ _
```

La configuración para la segunda máquina será:

- **Nombre de la máquina:** blockEsclavo
- **Tipo de Servicio:** Cliente
- **Dirección IP:** 192.168.100.26

```
Debian GNU/Linux 10 blockEsclavo tty1
blockEsclavo login: linuxtest
Password:
Last login: Sat Nov 28 00:44:50 -03 2020 on tty1
Linux blockEsclavo 4.19.0-6-amd64 #1 SMP Debian 4.19.67-2+deb10u2 (2019-11-11) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
linuxtest@blockEsclavo:~$ hostname -I
192.168.100.26 2803:9800:a011:85d0:a00:27ff:fe0b:576b
linuxtest@blockEsclavo:~$ _
```

6.1.1. Configuración de la máquina Servidor

Como se estableció anteriormente, la máquina n° 1 será la que nos provea de Servidor. Cabe recordar que los archivos *app/views.py* como *run_app.py* para esta máquina, deberán quedar sin ninguna modificación en su código, sólo se debe modificar para la máquina n° 2 que servirá como cliente. Ahora bien, para configurar la máquina n° 1 se procede a realizar los pasos:

Como primer paso se debe activar el entorno virtual con el comando:

```
$ source venv/bin/activate
```

Una vez realizado esto, se debe exportar la variable *FLASK_APP* ingresándole el valor “node_server.py” con el siguiente comando:

```
$ export FLASK_APP=node_server.py
```

Por último, lanzamos la instancia de servidor mediante el comando:

```
$ flask run --port 8000 --host=0.0.0.0
```

```
linuxtest@NodoMaestro:~/laboratorio3/python_blockchain_app-master$  
linuxtest@NodoMaestro:~/laboratorio3/python_blockchain_app-master$  
linuxtest@NodoMaestro:~/laboratorio3/python_blockchain_app-master$  
linuxtest@NodoMaestro:~/laboratorio3/python_blockchain_app-master$ source venv/bin/activate  
(venv) linuxtest@NodoMaestro:~/laboratorio3/python_blockchain_app-master$ export FLASK_APP=node_serv  
er.py  
(venv) linuxtest@NodoMaestro:~/laboratorio3/python_blockchain_app-master$ flask run --port 8000 --ho  
st=0.0.0.0  
* Serving Flask app "node_server.py"  
* Environment: production  
  WARNING: This is a development server. Do not use it in a production deployment.  
  Use a production WSGI server instead.  
* Debug mode: off  
* Running on http://0.0.0.0:8000/ (Press CTRL+C to quit)
```

6.1.2. Configuración de la máquina Cliente

Para configurar la máquina n° 2 que nos servirá de Cliente, como primer paso se debe activar el entorno virtual dentro del directorio donde se encuentran los archivo fuente con el comando:

```
$ source venv/bin/activate
```

Debemos modificar el archivo *app/views.py* para colocarle la ip del Servidor a donde deberá de apuntar, junto con el puerto que debe estar escuchando, mediante el programa *vi*.

```
1 import datetime  
2 import json  
3  
4 import requests  
5 from flask import render_template, redirect, request  
6  
7 from app import app  
8  
9 # The node with which our application interacts, there can be multiple  
10 # such nodes as well.  
11 # CONNECTED_NODE_ADDRESS = "http://127.0.0.1:8000"  
12 CONNECTED_NODE_ADDRESS = "http://192.168.100.22:8000"  
13  
14 posts = []  
15  
16  
17 def fetch_posts():  
18     """  
19     Function to fetch the chain from a blockchain node, parse the
```

Luego también se debe modificar el archivo *run_app.py* (como se explicó en la sección 3) ingresando lo siguiente:

```
1 from app import app
2
3 #app.run(debug=True)
4 app.run(debug=True, host='0.0.0.0')
```

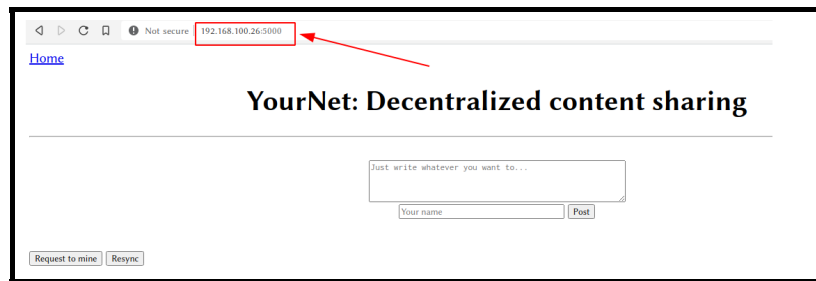
Lanzamos en la máquina el comando que nos permite crear la instancia de cliente:

```
$ python run_app.py
```

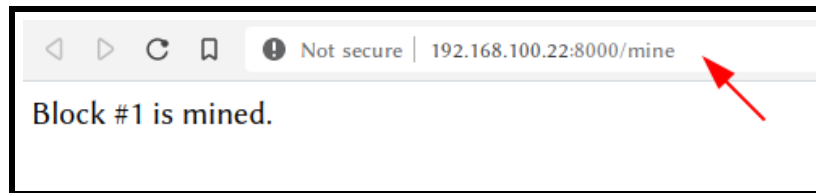
```
linuxtest@blockEsclavo:~/laboratorio3/python_blockchain_app-master$ source venv/bin/activate
(venv) linuxtest@blockEsclavo:~/laboratorio3/python_blockchain_app-master$
(venv) linuxtest@blockEsclavo:~/laboratorio3/python_blockchain_app-master$
(venv) linuxtest@blockEsclavo:~/laboratorio3/python_blockchain_app-master$ python run_app.py
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 657-759-307
```

6.1.3. Minado mediante el Navegador Brave

Una vez que tenemos tanto el servidor como el cliente levantados, procedemos a ingresar al navegador web por defecto en nuestra máquina huésped (en nuestro caso Brave) e ingresamos la dirección del cliente, la cual era **192.168.100.26** junto con el puerto donde estará escuchando, el **5000**. Una vez ingresado esto, nos lleva a una página para poder minar con el contenido deseado:



Una vez aquí, ingresamos el contenido a ingresar en la caja de texto, luego ingresamos nuestro nombre y presionamos “Post”, y luego “Request to mine” el cual nos va a redirigir a la dirección del servidor, indicándonos que el contenido ha sido minado:



Como se puede ver a continuación, al volver a la página principal y al presionar “Resync” se puede ver la información minada, la hora y por quién ha sido ingresada:



7. Ejercicio nº 2

Para llevar a cabo este ejercicio, además de las máquinas empleadas en la sección 6.1, se deberá clonar una tercer máquina para utilizarla como Servidor Alternativo, para luego registrarlo como tal en el sistema distribuido mediante las terminales que se acceden a través de **Alt+F2** (empleadas en la sección 5) y mediante el programa *curl* instalado en la sección 4.1, replicar la base de datos en el SERVIDOR2.

7.1. Configuración de los Nodos Distribuidos

La configuración para la primer máquina será la siguiente:

- **Nombre de la máquina:** SERVIDOR1
- **Tipo de Servicio:** Servidor
- **Dirección IP:** 192.168.100.22
- **Nº de Puerto:** 8000

```
Debian GNU/Linux 10 Servidor1 tty1
Servidor1 login: linuxtest
Password:
Last login: Mon Nov 30 22:32:31 -03 2020 on tty1
Linux Servidor1 4.19.0-6-amd64 #1 SMP Debian 4.19.67-2+deb10u2 (2019-11-11) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
linuxtest@Servidor1:~$ hostname -I
192.168.100.22 2803:9800:a011:85d0:a00:27ff:fe18:2992
linuxtest@Servidor1:~$
```

La configuración para la segunda máquina será:

- **Nombre de la máquina:** CLIENTE
- **Tipo de Servicio:** Cliente
- **Dirección IP:** 192.168.100.26

```
Debian GNU/Linux 10 Cliente tty1
Cliente login: linuxtest
Password:
Last login: Mon Nov 30 22:35:13 -03 2020 on tty1
Linux Cliente 4.19.0-6-amd64 #1 SMP Debian 4.19.67-2+deb10u2 (2019-11-11) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
linuxtest@Cliente:~$ hostname -I
192.168.100.26 2803:9800:a011:85d0:a00:27ff:fe0b:576b
linuxtest@Cliente:~$
```

La configuración para la tercer máquina será:

- **Nombre de la máquina:** SERVIDOR2
- **Tipo de Servicio:** Servidor alternativo
- **Dirección IP:** 192.168.100.27
- **Nº de Puerto:** 8001

```
Debian GNU/Linux 10 Servidor2 tty1
Hint: Num Lock on

Servidor2 login: linuxtest
Password:
Last login: Mon Nov 30 22:43:31 -03 2020 on tty1
Linux Servidor2 4.19.0-6-amd64 #1 SMP Debian 4.19.67-2+deb10u2 (2019-11-11) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
linuxtest@Servidor2:~$ hostname -I
192.168.100.27 2803:9800:a011:85d0:a00:27ff:fed0:3691
linuxtest@servidor2:~$ _
```

7.2. Procedimiento

Para llevar a cabo este ejercicio, será necesario realizar una serie de pasos para poder establecer con exactitud que todo funciona de forma correcta y que se puede establecer de forma distribuida un blockchain local. Cabe recordar que para las 3 máquinas hace falta activar el entorno virtual mediante el comando:

```
$ source venv/bin/activate
```

Los pasos a seguir serán los siguientes:

1. Se debe levantar la máquina n° 1 que se establecerá como SERVIDOR1 , ejecutando el comando:

```
$ export FLASK_APP=node_server.py
```

Y luego:

```
$ flask run --port 8000 --host=0.0.0.0
```

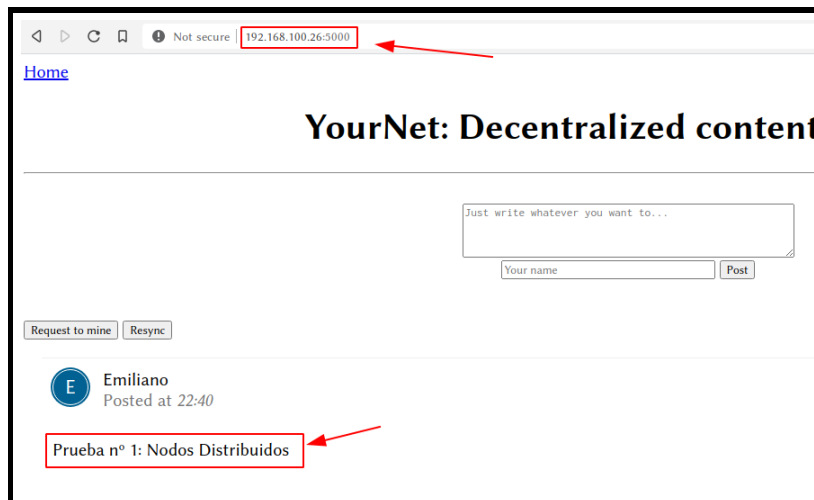
```
linuxtest@Servidor1:~$ cd laboratorio3/python_blockchain_app-master
linuxtest@Servidor1:~/laboratorio3/python_blockchain_app-master$ ls
app      node_server.py  README.md      run_app.py  venv
CONTRIBUTING.md  __pycache__  requirements.txt  screenshots
linuxtest@Servidor1:~/laboratorio3/python_blockchain_app-master$ source venv/bin/activate
(venv) linuxtest@Servidor1:~/laboratorio3/python_blockchain_app-master$ export FLASK_APP=node_server.py
(venv) linuxtest@Servidor1:~/laboratorio3/python_blockchain_app-master$
(venv) linuxtest@Servidor1:~/laboratorio3/python_blockchain_app-master$ flask run --port 8000 --host=0.0.0.0
 * Serving Flask app "node_server.py"
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://0.0.0.0:8000/ (Press CTRL+C to quit)
192.168.100.26 - - [30/Nov/2020 22:40:14] "GET /chain HTTP/1.1" 200 -
192.168.100.26 - - [30/Nov/2020 22:40:42] "POST /new_transaction HTTP/1.1" 201 -
192.168.100.26 - - [30/Nov/2020 22:40:42] "GET /chain HTTP/1.1" 200 -
192.168.100.5 - - [30/Nov/2020 22:40:46] "GET /mine HTTP/1.1" 200 -
192.168.100.5 - - [30/Nov/2020 22:40:46] "GET /favicon.ico HTTP/1.1" 404 -
192.168.100.26 - - [30/Nov/2020 22:40:49] "GET /chain HTTP/1.1" 200 -
```


2. Al mismo tiempo se debe levantar la máquina nº 2 y establecerla como CLIENTE como se puede ver a continuación, mediante el siguiente comando:

```
$ python run_app.py
```

```
linuxtest@Cliente:~/laboratorio3/python_blockchain_app-master$ source venv/bin/activate
(venv) linuxtest@Cliente:~/laboratorio3/python_blockchain_app-master$
(venv) linuxtest@Cliente:~/laboratorio3/python_blockchain_app-master$
(venv) linuxtest@Cliente:~/laboratorio3/python_blockchain_app-master$ python run_app.py
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 657-759-307
192.168.100.5 - - [30/Nov/2020 22:40:14] "GET / HTTP/1.1" 200 -
192.168.100.5 - - [30/Nov/2020 22:40:14] "GET /favicon.ico HTTP/1.1" 404 -
192.168.100.5 - - [30/Nov/2020 22:40:42] "POST /submit HTTP/1.1" 302 -
192.168.100.5 - - [30/Nov/2020 22:40:42] "GET / HTTP/1.1" 200 -
192.168.100.5 - - [30/Nov/2020 22:40:49] "GET / HTTP/1.1" 200 -
```

3. Acto seguido, para probar la relación entre el Servidor y el Cliente, se debe abrir el navegador de la máquina huésped, ingresando la dirección del front-end, y tal y como se realizó en la sección 6.1.3 se procede a minar con datos la base del Servidor.



4. Se procede a levantar la máquina nº 3, que se establecerá como SERVIDOR2. En este paso, para incluir este nuevo nodo como Servidor Alternativo a todo el sistema, es necesario abrir otra terminal, mediante **Alt+F2**, y emplear el programa **curl** instalado en la sección 4.1.

A continuación se procede a ingresar:

```
$ export FLASK_APP=node_server.py
```

Y luego:

```
$ flask run --port 8001 --host=0.0.0.0
```

A terminal window showing the execution of the 'flask run' command. The prompt is 'linuxtest@Servidor2:~/laboratorio3/python_blockchain_app-master\$'. The command entered is 'source venv/bin/activate' followed by 'export FLASK_APP=node_server.py' and then 'flask run --port 8001 --host=0.0.0.0'. The output shows that the Flask app is running on port 8001, with a warning that it is a development server and not for production use. Red arrows point to the command and the output.

```
linuxtest@Servidor2:~/laboratorio3/python_blockchain_app-master$ source venv/bin/activate
(venv) linuxtest@Servidor2:~/laboratorio3/python_blockchain_app-master$ export FLASK_APP=node_server
.py
(venv) linuxtest@Servidor2:~/laboratorio3/python_blockchain_app-master$ flask run --port 8001 --host
=0.0.0.0
 * Serving Flask app "node_server.py"
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://0.0.0.0:8001/ (Press CTRL+C to quit)
```

Se replica el contenido del SERVIDOR1 en el SERVIDOR2, con el programa **curl**. Este es necesario para poder sincronizar la base de datos del Servidor n° 1 con el nuevo Servidor Alternativo, para ello se ingresa el siguiente comando en la nueva terminal:

```
$ curl -X POST \
http://192.168.100.22 /register_with \
-H 'Content-Type: application/json' \
-d '{"node_address": "http://192.168.100.27"}'
```

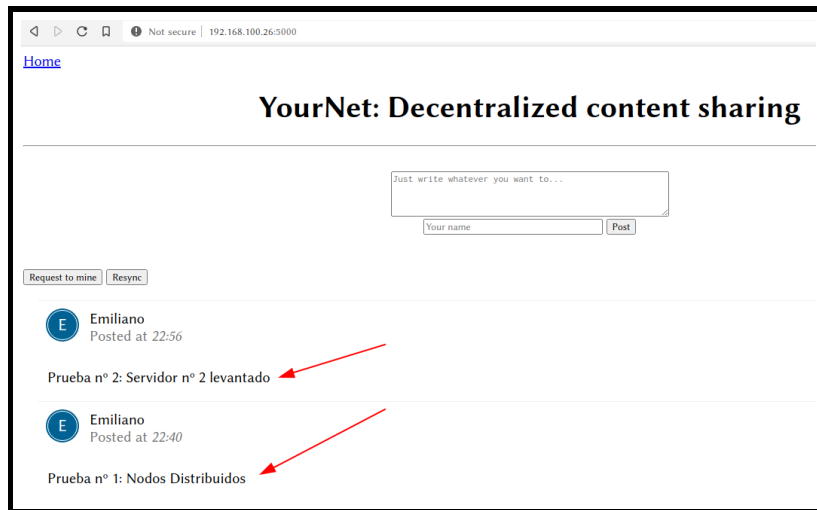
A terminal window showing the execution of the 'curl' command. The prompt is 'linuxtest@Servidor2:~/laboratorio3/python_blockchain_app-master\$'. The command entered is 'curl -X POST http://192.168.100.27:8001/register_with -H 'Content-Type: application/json' -d '{"node_address": "http://192.168.100.22:8000"}'. The output shows that the registration was successful. Red arrows point to the command and the output.

```
Debian GNU/Linux 10 Servidor2 tty2
Servidor2 login: linuxtest
Password:
Last login: Mon Nov 30 22:44:14 -03 2020 on tty1
Linux Servidor2 4.19.0-6-amd64 #1 SMP Debian 4.19.67-2+deb10u2 (2019-11-11) x86_64

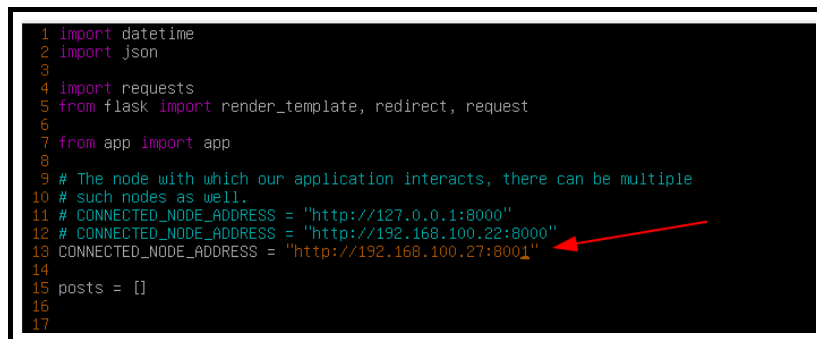
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
linuxtest@Servidor2:~/laboratorio3/python_blockchain_app-master$
linuxtest@Servidor2:~/laboratorio3/python_blockchain_app-master$ curl -X POST http://192.168.100.27:
8001/register_with -H 'Content-Type: application/json' -d '{"node_address": "http://192.168.100.22:8
000"}'
Registration successfullinuxtest@Servidor2:~/laboratorio3/python_blockchain_app-master$ _
```

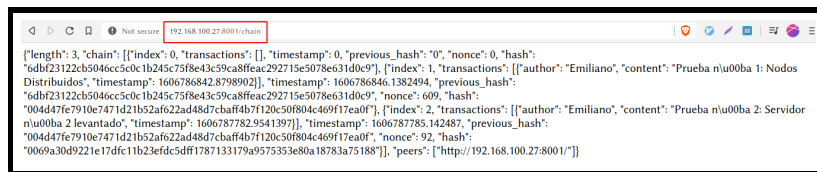
5. Se agrega un nuevo contenido a la cadena, para asegurarnos que todo sigue funcionando correctamente:



6. Luego de ello se procede, a bajar el Cliente (mediante **Ctrl+c**) y modificando el archivo del Front-end *view.py* apuntando a la máquina nº 3.



7. Procedemos a visualizar que el nuevo cambio ha sido impactado en la base de datos ingresando en el navegador de la máquina huésped con la dirección del SERVIDOR2, la cual es: 192.168.100.27 junto con el puerto donde está escuchando, el cual es el 8001:



Y se procede a levantarlo mediante:

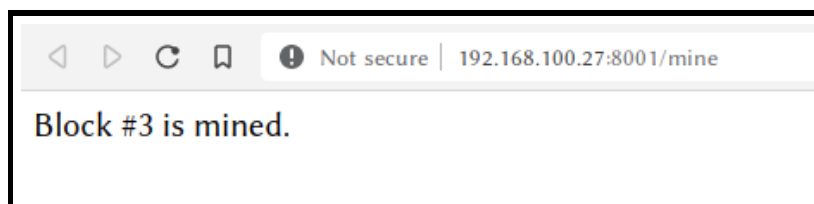
```
$ python run_app.py
```

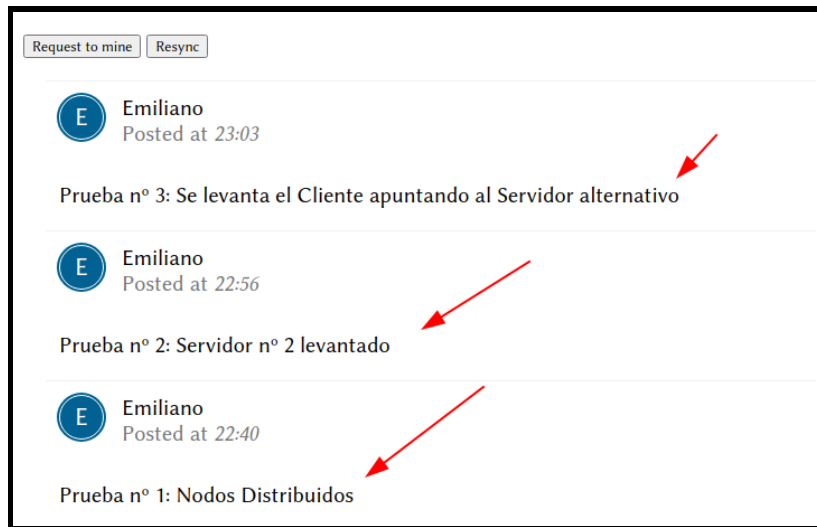
```
(venv) linuxtest@Cliente:~/laboratorio3/python_blockchain_app-master$ python run_app.py
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 657-759-307
```

8. Una vez que la base de datos del blockchain se encuentran replicadas tanto en el SERVIDOR1 como en SERVIDOR2, se procede a bajar el SERVIDOR1 para evidenciar que el sistema sigue funcionando aún cuando uno de los nodos ya no está disponible.

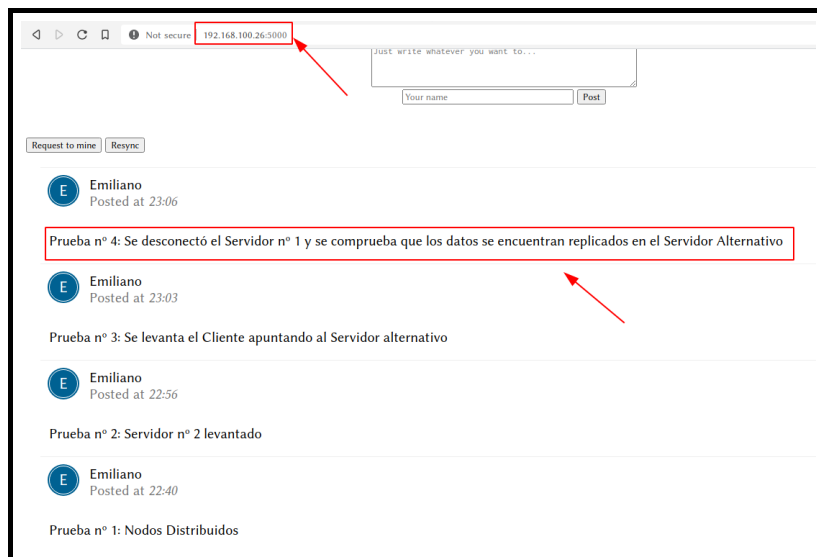
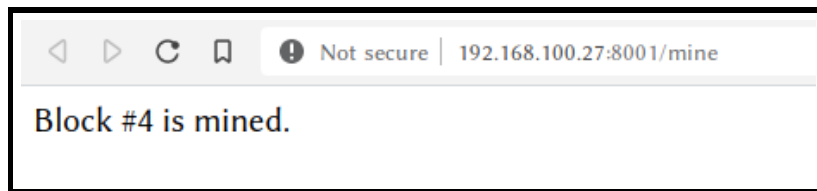
```
(venv) linuxtest@Servidor1:~/laboratorio3/python_blockchain_app-master$
(venv) linuxtest@Servidor1:~/laboratorio3/python_blockchain_app-master$ flask run --port=0.0.0.0
* Serving Flask app "node_server.py"
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:8000/ (Press CTRL+C to quit)
192.168.100.26 - - [30/Nov/2020 22:40:14] "GET /chain HTTP/1.1" 200 -
192.168.100.26 - - [30/Nov/2020 22:40:42] "POST /new_transaction HTTP/1.1" 201 -
192.168.100.26 - - [30/Nov/2020 22:40:42] "GET /chain HTTP/1.1" 200 -
192.168.100.5 - - [30/Nov/2020 22:40:46] "GET /mine HTTP/1.1" 200 -
192.168.100.5 - - [30/Nov/2020 22:40:46] "GET /favicon.ico HTTP/1.1" 404 -
192.168.100.26 - - [30/Nov/2020 22:40:49] "GET /chain HTTP/1.1" 200 -
192.168.100.27 - - [30/Nov/2020 22:55:16] "POST /register_node HTTP/1.1" 200 -
192.168.100.26 - - [30/Nov/2020 22:56:22] "POST /new_transaction HTTP/1.1" 201 -
192.168.100.26 - - [30/Nov/2020 22:56:23] "GET /chain HTTP/1.1" 200 -
192.168.100.5 - - [30/Nov/2020 22:56:25] "GET /mine HTTP/1.1" 200 -
192.168.100.26 - - [30/Nov/2020 22:56:42] "GET /chain HTTP/1.1" 200 -
C(venv) linuxtest@Servidor1:~/laboratorio3/python_blockchain_app-master$
```

9. Para visualizar que todo haya quedado funcionando como se comentó en el punto anterior, se procede a ingresar en el navegador web la dirección 192.168.100.26 junto con el puerto 5000, que es la del cliente el cual debería de seguir funcionando con todos los minados de datos que se le realizaron en los primeros puntos.





10. Se mina nuevamente la información para demostrar que el sistema aún sigue en pie luego de haber dado de baja el SERVIDOR1 :



8. Conclusiones

Mediante este laboratorio, se pudo comprobar y verificar el funcionamiento básico de un *blockchain* público mediante el uso del lenguaje de programación Python; el cual permitió implementar desde cero un sistema distribuido con varios nodos (como se demostró en la sección 7) sincronizados, creando una aplicación que permite a varios usuarios compartir información mediante esta cadena de bloques, siempre teniendo en cuenta la confiabilidad de la información suministrada, lo que significa que los datos que se suscriben a la base pública, siempre estarán a salvo de cualquier intento de modificación por un agente externo, como también el beneficio de no tener un único punto de falla, punto que vuelve aún más robusto el sistema en su completitud.

Por otra parte, junto con la utilización de la aplicación de blockchain, se adentró al uso de distintos programas externos como lo puede ser **curl**, **flask** o **request** los cuales no se tenía noción de su aplicabilidad y funcionamiento.

Bibliografía

- [1] Allen Downey, Jeffrey Elkner y Chris Meyers. *Think Python: How to Think Like a Computer Scientist*. Addison Wesley, 2002.
- [2] Kansai Satwik. *Develop a blockchain application from scratch in Python*. <https://developer.ibm.com/technologies/blockchain/tutorials/develop-a-blockchain-application-from-scratch-in-python/>. (Online; accedido el 04/12/2020). 2020.