Ubuntu Linux como File Server y AppServer

Benitez Elio Gaston, Zini Horacio, Almiron Augusto, Santiago Pioli[1]

[1] Fac. de Ciencias Exactas Naturales y Agrimensura, Universidad Nacional del Nordeste, Corrientes
gastongrillo2001@gmail.com, horazini@gmail.com,
almironaugusto404@gmail.com,
santiago.pioli@comunidad.unne.edu.ar

Resumen. Desde que compartir información entre usuarios dentro de una red se volvió un elemento esencial en toda organización se comenzaron a implementar los servidores, que podrían cumplir con este objetivo. A medida que el tiempo avanzaba estos servidores cada vez comenzaban a tener funciones mucho más específicas y a la par de esto se comenzaron a implementar distintas formas de instalación de componentes hardware y software, además de prestar mucha atención a la infraestructura que sostiene este tipo de sistemas.

Con esta perspectiva aparecen en escena servidores especializados, en particular, los servidores de archivo y aplicación, foco de atención del presente trabajo. El servidor de archivos ofrece aquellas características correspondientes a compartir archivos por la red logrando que la red local tenga un punto de centralización de información , teniendo así la administración un mayor poder y control sobre la información que se está compartiendo. Por otro lado, el servidor de aplicaciones ofrecerá a los usuarios todo su equipo hardware y software con el propósito de ejecutar los programas requeridos, liberando de toda dependencia de hardware al usuario final.

En el marco del trabajo de investigación en equipo de la asignatura Redes de Datos, cuarto año de la Carrera LSI, proponemos la implementación y despliegue en red de un servidor de archivos y un servidor de aplicaciones, soportado en un sistema operativo Ubuntu Linux. En toda organización ya sea pequeña o grande su uso beneficia mucho a todo tipo de gestión involucrada en la empresa.

Palabras clave: Servidor, Linux, Aplicaciones, Archivos, Open Source, Samba, Eventlet, UNNE.

1. Introducción

En el presente trabajo se realiza una investigación a dos tipos de servidores particulares, servidores de archivo y servidores de aplicación. Esto lo haremos teniendo en cuenta que usaremos como sistema operativo base a Linux versión Ubuntu/CentOS.

Estos servidores están orientados principalmente a la atención de los usuarios que requieran de servicios de almacenamiento y búsqueda de información , y también para

aquellos usuarios que requieran de entornos de ejecución de determinadas aplicaciones.

Dentro de éste escenario surge una cuestión que nos importa: ¿Es necesario estar conectado directamente a la red local? No necesariamente tendremos que estarlo, ya que en la actualidad gracias a soluciones VPN es posible acceder y hacer uso de los servicios brindados por los servidores dando lugar al trabajo remoto. Inclusive hoy en día es posible asociar todas estas funcionalidades a un servidor WEB, para que todos se puedan conectar mediante internet.

Otro aspecto importante que exploramos es el de los permisos, ya que en estos o en cualquier sistema hoy en día existen las jerarquías de permisos. Esto es un punto muy importante a la hora de hablar de seguridad, ya que evitamos que cualquier usuario realice operaciones indebidas en la red que perjudique al servidor y a la totalidad del sistema.

Estas y otras cuestiones surgen de las reglas de negocio y de la lógica manejada en una organización. Dependiendo de esto, muchas opciones de configuración y operaciones posibles, se llevarán a cabo de determinadas formas, y también el equipo software y hardware está compuesto de elementos específicos, todo esto se ejecuta siempre buscando cumplir con los objetivos (fig.1).

Conceptos básicos a tratar en el documento:
– Servicio: Es un recurso que expone a una aplicación a tráfico externo. [1]
– Servidor: Es una máquina, física o virtual, integrada a una red, que ofrece un servicio especial que otros programas denominados clientes pueden usar a nivel local o a través de internet. El tipo de servicio depende del tipo de software del servidor. La base de la comunicación es el modelo cliente-servidor y, en lo que concierne al intercambio de datos, entran en acción los protocolos de transmisión específicos del servicio.
Una denominación alternativa para un servidor basado en hardware es "host" o anfitrión. En principio, todo ordenador puede usarse como "host" con el correspondiente software para servidores. [1]

2. Instrumentación

Para el desarrollo de las soluciones, podemos dividir la instrumentación en dos partes, ya que se utilizaron distintos programas para el servidor de archivos y aplicaciones.

Instrumentación del servidor de archivos
Para la construcción del servidor son necesarios los siguientes elementos:

- Oracle VM VirtualBox
- Sistema operativo Ubuntu Linux

- Servidor Samba

Cabe aclarar que el sistema operativo estará montado sobre Oracle VM. Y el servidor Samba será una aplicación que se puede instalar sobre nuestro sistema operativo Linux. Dejando esto en claro ahora pasaremos a la instalación del servidor Samba dentro de Linux.

Luego de ingresar al sistema operativo con nuestras credenciales ingresamos el siguiente comando en la consola, que dará inicio a la instalación de la aplicación:

$sudo apt install samba

Luego de esto solo restará revisar su estado mediante otro comando:

$sudo service nmbd status

Podemos apreciar entonces que el estado del servidor es activo y en funcionamiento

Instrumentación del servidor de aplicaciones

Para la construcción del servidor de aplicaciones se requieren las siguientes herramientas:

- Python
- Flask
- Socket.IO
- Servidor WSGI de Eventlet
- Oracle VM Virtual box
- Sistema operativo linux ubuntu.

De igual manera que en la sección anterior el sistema operativo estará montado sobre Oracle VM. Lo primero que tendremos que hacer es instalar la aplicación Python introduciendo los siguientes comandos:

$sudo apt-get update
$sudo apt-get install python3

Lograda la instalación de Python ahora trabajaremos con pip, es un sistema de gestión de paquetes utilizado para instalar y administrar paquetes de software escritos en Python. Los comandos que introduciremos serán:

$pip install Flask==1.1.2
$pip install Flask-Session==0.3.2
$pip install Flask-SocketIO==4.3.1
$pip install python-engineio==3.13.2
$pip install python-socketio==4.6.0
$pip install Werkzeug==2.1.0 --force-reinstall
$pip install eventlet

Las versiones usadas pueden variar en cada caso, pero esta es una combinación válida de versiones para evitar problemas de compatibilidad. Con esto estamos logrando la instalación de lo siguiente:

- Flask: Framework web de Python.
- Socket.IO: Biblioteca de Python.
- Eventlet: Biblioteca de redes de Python que proporciona un servidor.

3. Servidor de archivos en Linux Ubuntu

Un servidor de archivos es un servidor central responsable del almacenamiento y la administración de archivos de datos para que otras computadoras en la misma red, o que estén conectadas de manera remota, puedan acceder a una serie de archivos, carpetas, y documentos. [2]

Estas principales características del servidor de archivos permite determinar desde un principio algunos aspectos del hardware, más que nada por la gran cantidad de información que tienen que controlar. Algunos de estos aspectos son:

- El servidor debe proveer el suficiente espacio en disco duro.
- Memoria RAM con suficiente capacidad para manejar archivos que son consultados con mucha frecuencia.
- CPU con suficiente capacidad para soportar las múltiples operaciones y consultas que son realizadas desde y hacia el servidor.

El protocolo que utiliza el servidor de archivos es SMB (server message block), es del tipo cliente / servidor y busca gobernar el acceso a archivos y directorios dentro de una red compartida por usuarios. [3] En el funcionamiento de SMB lo primero que se logra es establecer una conexión, mediante mensajes correspondientes. Y tanto esto como el constante transporte de información utilizara el protocolo TCP.

Seguridad en los servidores de archivos

El servidor ofrecerá un lugar de almacenamiento centralizado donde existiría una política de permisos que buscan controlar la lectura, escritura, creación de archivos, e inclusive el acceso a determinadas direcciones. Esto busca controlar los accesos indebidos, ataques, errores por parte de los usuarios y otras cuestiones. Además el hecho de contener una cantidad de datos lo hace un objetivo muy recurrente de ataques y ransomware.

A la hora de crear y configurar usuarios, el administrador del servidor tendrá que asignar un nombre de usuario y contraseña para que sea posible la conexión. Y luego de esto tendrá que asignar los permisos correspondientes a cada usuario.

Cabe aclarar que existe una jerarquía en estos permisos, algunos usuarios solo podrán realizar lectura sin posibilidad de escritura de algunas cosas, mientras que otros tienen el poder de modificar cualquier archivo del servidor.

Implementación del servidor

En esta sección iremos se detallan los pasos para crear y administrar el servidor. Una vez que tenemos instalado el servicio Samba, tendremos que acceder al archivo de configuración para detallar los parámetros que se manejan en la conexión y recursos compartidos. Ingresamos a la configuración de samba introduciendo en la consola:

$sudo nano /etc/samba/smb.cof

Al final del archivo establecemos los datos correspondientes para el levantamiento del servidor. Estos son:

samba-share	Es el nombre del recurso compartido
comment	Descripción del recurso compartido
path	Directorio del recurso compartido
real only	Se especifica si los usuarios solo tienen permiso de lectura
browsable	Se habilita para que figure en la lista de recursos compartidos

En nuestra implementación tendrá la forma que se detalla en la imagen (fig.3).

De esta forma declaramos el recurso compartido servidor-samba que tiene como directorio fuente a carpeta_servidor y los usuarios pueden realizar modificaciones sobre el mismo. Con esto ya cumplimos la parte de creación y puesta en escena del servidor.

Permisos de la carpeta fuente

Algunos problemas que pueden surgir al conectar con el recurso compartido se pueden dar con los permisos que posee el directorio del recurso compartido. Dependiendo de la ubicación del directorio tendremos que habilitar distintos tipos de permisos. [4] Lo logramos ingresando lo siguiente en el directorio home:

$sudo chmod 755 carpeta_servidor/

El número ‘7’ corresponde a los permisos de lectura, escritura y ejecución para el administrador del sistema.

Los números ‘55’ asignan permisos de lectura y ejecución a los usuarios y grupos que se conecten al recurso compartido. Esta configuración nos permitirá seguir adelante con el desarrollo.

Creación de los usuarios y grupos

Con el fin de realizar las tareas de asignación de permiso de una manera más simple, se sugiere asignar permisos a grupos de cuentas. Para este caso en particular decidimos utilizar una simulación de “gerentes” y “empleados”. Los permisos se

asignan a nivel de grupo de forma que cada uno tenga permisos específicos sobre los directorios donde dichos grupos son los propietarios. Los pasos previstos son:

1) crear los grupos

Ingresamos en la consola los siguientes comandos:

$sudo groupadd gerentes
$sudo groupadd empleados

2) asignar grupo a los directorios correspondientes

$sudo chgrp gerentes carpeta_gerente/
$sudo chgrp empleados carpeta_empleado/

3) asignar permisos a nivel de grupos

$sudo chmod 070 carpeta_empleado/
$sudo chmod 070 carpeta_gerente/

4) Cuando creemos un usuario le asignamos un grupo
En primer instancia creamos los usuarios:

$sudo useradd gerente1
$sudo useradd empleado 1

Asignamos contraseñas con modo Samba:

$sudo smbpasswd gerente1
$sudo smbpasswd empleado1

Asignamos los grupos:

$sudo usermod -g gerentes gerente1
$sudo usermod -g empleados empleado1

Ejecución del sistema

En caso de tener clientes de Windows, será necesario modificar un archivo del sistema operativo de antemano para lograr una autenticación correcta. Lo que buscamos lograr con esto es que cada vez que un usuario se conecte al recurso compartido, el mismo sistema le requiera un usuario y contraseña. El problema está en que por defecto el sistema operativo Windows no permite la conexión a distintos recursos remotos utilizando distintas credenciales.

Para solucionar esto tendremos que ingresar al archivo hosts ubicado en el directorio C:\Windows\System32\drivers\etc. Este archivo permite establecer un servicio DNS para el direccionamiento a una IP específica. Es posible editar el archivo con un bloc de notas, lo que haremos será ingresar una IP y su NOMBRE DE IP asociado. Para nuestro caso de uso, la ip del servidor es 192.168.0.17, y le pondremos el nombre ‘servidor’, nos tendría que quedar como en la figura (fig.4).

Una vez hecho esto ya podremos conectarnos al recurso por nombre IP ‘servidor’, y cada vez que se logre la conexión, nos solicitará credenciales.

Ingreso al recurso compartido desde un cliente

En el caso desarrollado, nuestro servidor está en un sistema operativo Linux Ubuntu y nuestros clientes serán de sistemas operativos Windows. En sistemas

Windows, para conectar a una unidad de red, tendremos que ir al apartado red del explorador de archivos, y lo habilitaremos en caso de estar desactivado. Le daremos click derecho y elegiremos la opción conectar a unidad de red, se nos abrirá una ventana como la (fig.5).

En la sección carpeta ingresamos la dirección del recurso compartido con el formato indicado. Recordar que en este caso no usamos un número de IP en servidor sino una nombre de ip que su configuración fue explicada al inicio de la sección. Hacemos click en Finalizar y luego se nos solicitara nuestras credenciales para confirmar (fig.6). Por ejemplo, si ingresamos una cuenta del grupo gerentes, será posible visualizar el recurso compartido para dicho grupo.

Al intentar el acceso sobre recursos compartidos al que no poseemos permisos, surgen mensajes de error (fig.7). Desde el lado del servidor toda esta actividad puede ser monitoreada desde la consola:

$sudo smbstatus
Este comando nos brindara información y detalles de los usuarios conectados (fig.8).

Ventajas y desventajas del servidor de archivos
A continuación detallaremos de manera general ventajas y desventajas más importantes para el desarrollo del servidor.

Ventajas:
Centralidad: Toda la información estará almacenada en un solo lugar, de forma que todo archivo puede ser controlado y regulado. Esto además es bueno para un usuario ya sea cliente o trabajador en algún sistema por el simple hecho de que todas los archivos con los que se trabaja no se almacenan en el disco duro de usuario.
Seguridad interna: Podemos establecer un protocolo de permisos que siga la lógica del negocio, de forma que ,además de organizar a los usuarios, también protegerá aquellos archivos y carpetas que son cruciales para el funcionamiento lógico y administrativo del negocio.
Trabajo remoto: Si el servidor de archivos está configurado para el acceso por Internet, los archivos también están disponibles de forma remota.

Desventajas:
Dependencia de hardware: al ser un servidor centrado en el almacenamiento su punto de mayor importancia a nivel de hardware serán los dispositivos encargados de
almacenar la información. Será necesario tener presente una infraestructura centrada en la presencia de estos dispositivos y en su mantenimiento.
Ataques al servidor: por la cantidad de información que poseen es muy habitual tener que lidiar con intentos de hackeo. Por esto será necesario tener presente un equipo encargado de la ciberseguridad.
Efecto - Cuello de botella: dependiendo de la cantidad de usuarios y transacciones que se van realizando , la prestación de servicios del servidor puede verse limitada.
Esto tiene una solución apuntando a una buena infraestructura de redes.

Aplicación Web Cliente-Servidor de Chat para redes de área local

Un servidor de aplicaciones es un aplicativo que trabaja sobre una red y proporciona el entorno de ejecución para una aplicación. Más específicamente, el servidor de aplicaciones es el componente de tiempo de ejecución principal en todas las configuraciones y donde una aplicación se ejecuta realmente. [5]

En la actualidad por lo general el servidor de aplicaciones colabora con el servidor web para ofrecer una respuesta dinámica y personalizada a una solicitud de cliente. De forma que el esquema se representa como lo muestra la figura (fig.9). En el desarrollo del aplicativo esta colaboración se aprecia mediante el desarrollo de la aplicación utilizando Python , que es ejecutada en la máquina virtual, y la prestación del servidor web WSGI.

La Interfaz de Puerta de Enlace del Servidor Web, "WSGI" (Web Server Gateway Interface), es una convención de llamada simple para que los servidores web re-envíen solicitudes a aplicaciones web o marcos escritos en el lenguaje de programación Python. [6]. Eventlet es una biblioteca de redes escrita en Python que, entre otras cosas, ofrece un servidor WSGI el cual proporciona una manera simple y fácil de iniciar un servidor controlado por eventos. Esto puede servir como un servidor web integrado en una aplicación o como base para un paquete de servidor web más completo. Logra una alta escalabilidad y concurrencia mediante el uso de IO no-bloqueante y, al mismo tiempo, conserva una alta usabilidad del programador mediante el uso de corrutinas. [7]

Una vez instalados los recursos especificados en la sección 2.2, el procedimiento para implementar la aplicación consiste de los siguientes pasos:

1. Crear un directorio, preferentemente en el directorio “home”, que será la base del proyecto.
2. En dicho directorio, se crearán los archivos “app.py” y “app.wsgi” que contendrán, respectivamente, el código Python que permite la comunicación por Sockets; y el código que permite la iniciación del servidor.
3. Se creará también la carpeta la carpeta “templates” que contendrá las plantillas o interfaces que usará nuestra aplicación
4. En “templates” irán los archivos “index.html” y “chat.html”.

Ejecución de la aplicación

Una vez completados los pasos anteriores, para ejecutar la aplicación basta con definir la variable de ambiente FLASK_APP que contendrá el camino absoluto al archivo .py que es la aplicación FLASK_APP = "/home/…/app/app.py" o bien, posicionarse, con el comando cd, en el directorio que contenga el archivo, y ejecutar el siguiente comando: sudo flask run -h <IP local>

La ejecución correcta del comando devolverá la línea “Running on http://192.168.0.25:5000” que indica que la aplicación está siendo servida exitosamente en el puerto 5000 del local host. A partir de este momento y hasta que se dé de baja el servidor, es posible acceder a la aplicación desde los navegadores de

cualquier dispositivo conectado a la misma red local que la máquina que ejecuta el servidor, ingresando IP:PUERTO en la barra de dirección. En las siguientes figuras, es posible visualizar el resultado de la ejecución de la aplicación (fig.10), (fig.11), (fig.12), incluso desde conexiones realizadas desde Android (fig.13) y (fig.14).

Ventajas y desventajas del servidor de aplicaciones

Ventajas:

Ahorro de recursos para el cliente: El cliente que consume las aplicaciones ofrecidas no tiene necesidad de presentar un equipo de 'alto nivel' para su uso. Así como no necesita tener instalado los intérpretes de los lenguajes que utiliza la aplicación.

Aplicación en web: al implementar este servicio mediante un servidor web tenemos la posibilidad de desarrollar la aplicación a un mayor nivel permitiendo así clientes de forma remota que se conecte mediante algún servicio DNS.

Adaptabilidad: Otra ventaja de poseer una aplicación web es que la misma se adapta a los distintos dispositivos. Este caso se aprecia en la sección 4.3 donde se logró un acceso mediante un sistema operativo Android desde un dispositivo móvil.

Desventajas:

Complejidad de despliegue: Una vez desarrollada la aplicación, es muy importante conocer de las características de los lenguajes utilizados para poder decidir la mejor opción entre los servidores a utilizar para el despliegue, así como todo los recaudos necesarios. Mayor consumo del lado del servidor: El peso de ejecutar aplicaciones, teniendo en cuenta la posibilidad de múltiples llamadas al servicio de manera simultánea, es una característica a considerar, ya que un servidor que no cumpla con los requerimientos sería muy susceptible a fallos y a caídas.

Necesidad de infraestructura de redes: Al hacer uso del servidor de aplicación mediante la web será necesario tener un equipamiento de infraestructura de redes que soporte la gran cantidad de solicitudes de información.

5. Conclusiones y posibles mejoras

Sobre los servidores de archivos podemos ver que fue posible su implementación, y apreciamos que no es necesario presentar con equipos hardware de gran capacidad. Por su funcionalidad hoy en día es un elemento que podríamos decir que es casi obligatorio en cualquier organización. Más que nada porque los datos están centralizados, organizados, con un protocolo de seguridad y su administración será más eficiente.

Por el lado del servidor de aplicaciones, logramos apreciar su utilidad a la hora de ofrecer una gama de soluciones más amplia que la permitidos por un servidor web tradicional, destacando la importancia de realizar un buen análisis de la situación y diseño previo a la etapa de desarrollo, para asegurarnos que estamos utilizando las herramientas más aptas para lograr los objetivos requeridos.

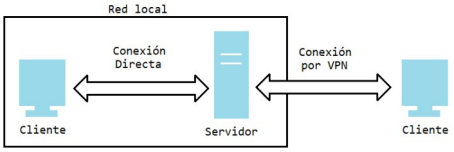
La aplicación de estos servidores fue tratada de forma separada para poder considerar mejor las características de cada uno, como su manera de aplicar, funcionamiento, protocolos utilizados ya sea en la aplicación o en la capa de

transporte y también para entender a nivel de capa de red como funciona la transmisión de información.

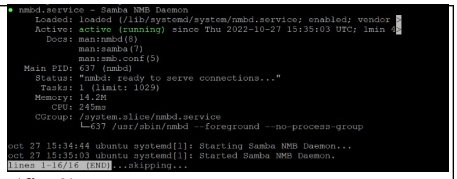
Referencias

[1] https://es.wikipedia.org/wiki/Servidor
[2] https://www.samba.org/samba/docs/
[3] https://ayudaleyprotecciondatos.es/2021/03/04/protocolo-smb/
[4] https://computernewage.com/2016/05/22/gestionar-usuarios-y-permisos-en-linux/#grupos
[5] https://eventlet.net/doc/modules/wsgi.html
[6] https://peps.python.org/pep-3333/
[7] http://eventlet.net/doc/

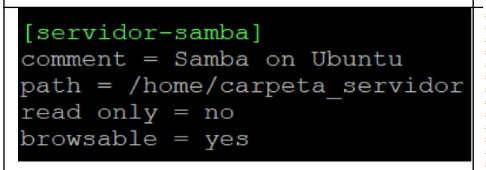
Lista de Figuras.



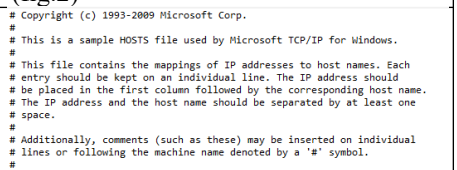
(fig.1)



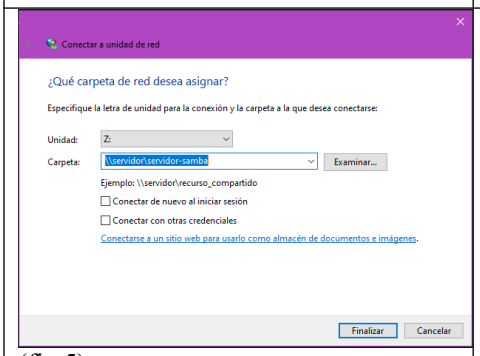
(fig.2)



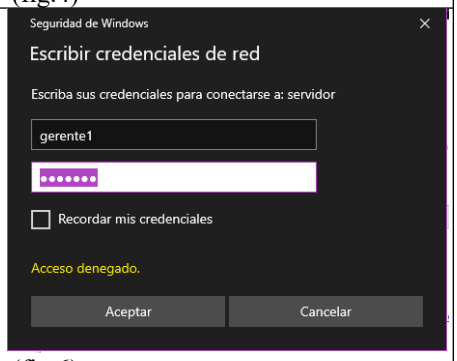
(fig.3)



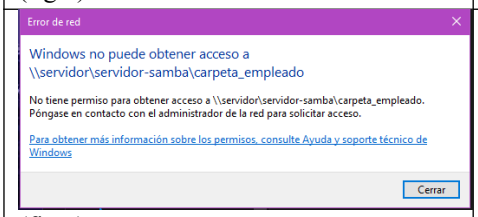
(fig.4)



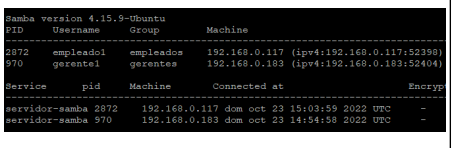
(fig.5)



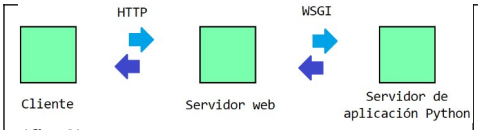
(fig.6)



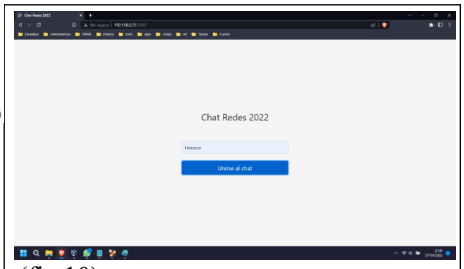
(fig.7)



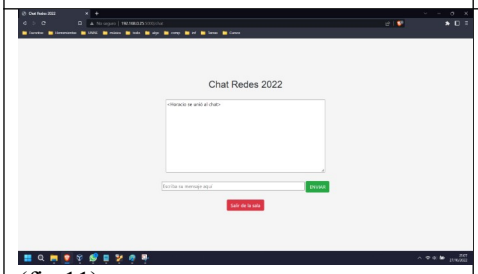
(fig.8)



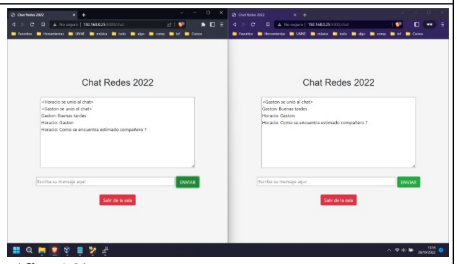
(fig.9)



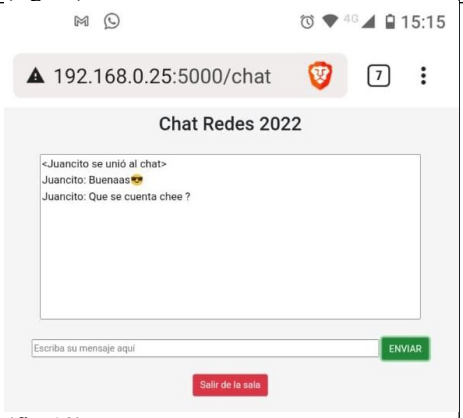
(fig.10)



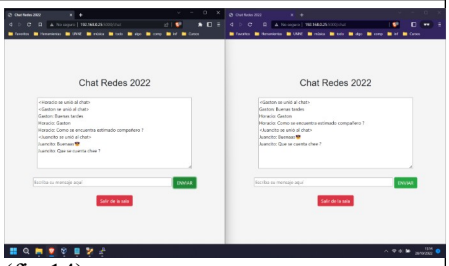
(fig.11)



(fig.12)



(fig.13)



(fig.14)