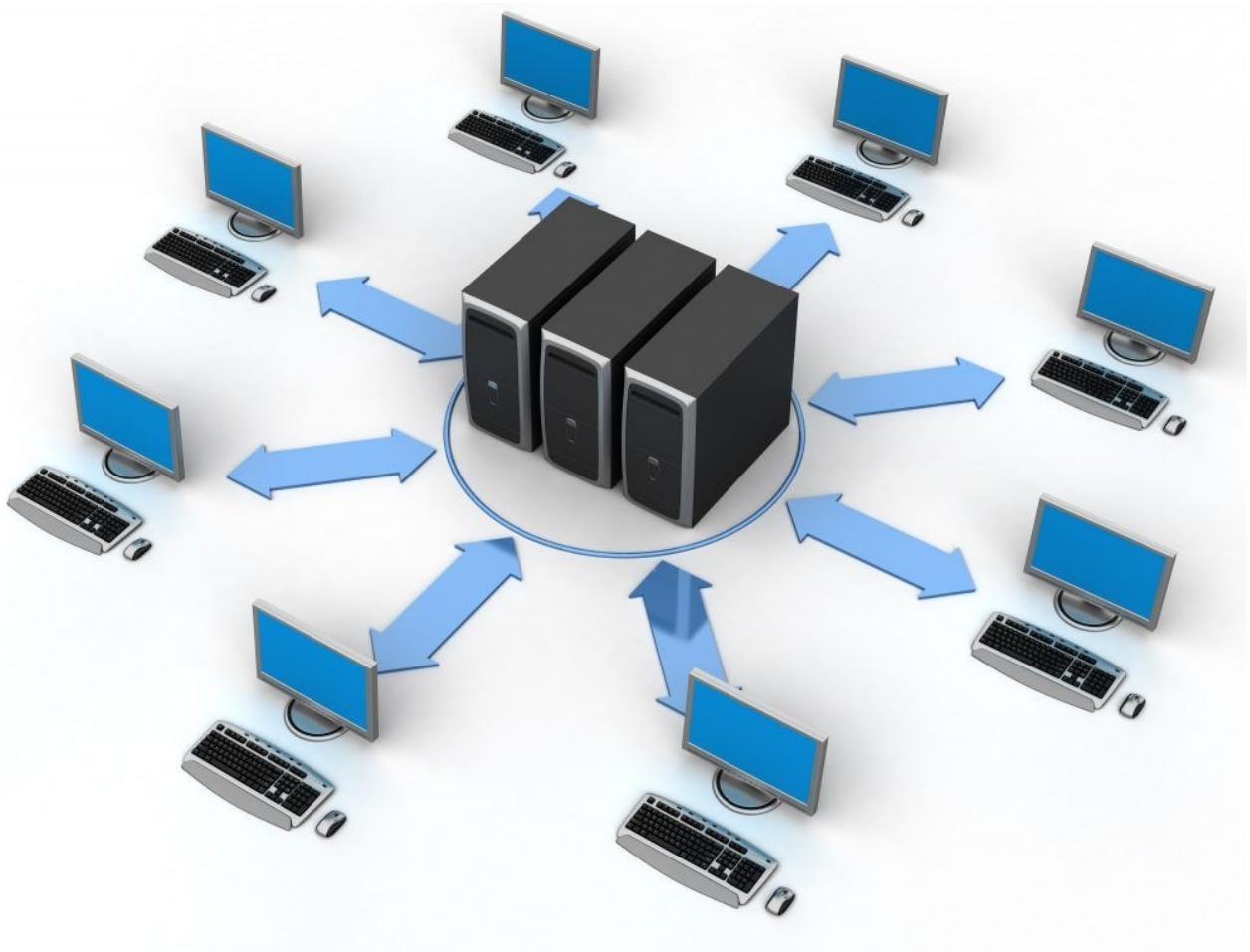


TAREA UNIDAD 1

Implantación, configuración y administración de servidores web



CFGS DAW. Módulo: Despliegue de aplicaciones web (2023/24)
Dayro Morales Cruz

Indice

Ejercicio 1	3
Ejercicio 2	3
Ejercicio 3	5
Ejercicio 4	12
Ejercicio 5	17
Ejercicio 6	21

Ejercicio 1.

Arquitecturas web. Una plataforma o arquitectura web es el entorno empleado para diseñar, desarrollar y ejecutar un sitio web. Hay muchos modelos de plataformas y una de las piezas más importantes y determinantes a la hora de elegirla es el sistema operativo. Describe dos plataformas web, una basada en Linux y otra basada en Windows, indicando qué tecnologías utilizan para cada una de las capas, es decir, para cubrir aspectos como el servicio web (HTTP), el contenido dinámico, o el acceso a datos.

1. Linux:

La plataforma **LAMP** trabaja completamente con software libre y no está sujeta a restricciones de autor.

LAMP es un acrónimo de las siguientes palabras:

- **Linux:** Sistema operativo.
- **Apache:** Servidor web.
- **MySQL:** Gestor de bases de datos.
- **PHP:** Lenguaje interpretado PHP

2. Windows:

La plataforma **WISA** está basada en tecnologías desarrolladas por la compañía Microsoft, hablamos de un software propietario.

WISA es un acrónimo de las siguientes palabras:

- **Windows:** Sistema operativo.
- **Internet Information Services:** servidor web.
- **SQL Server:** gestor de bases de datos.
- **ASP o ASP.NET:** como lenguaje para scripting.

Ejercicio 2.

Clasificación de las Aplicaciones web. Enumera y explica brevemente cada una de las diferentes tecnologías asociadas a las aplicaciones web que se ejecutarán tanto del lado del servidor como del cliente, especificando lo que corresponde a cada uno de los casos.

Del lado del cliente:

- **Página web Estática.** Muestra el mismo contenido para todos los usuarios.
- **Página web Animada.** Realizado con tecnología FLASH.
- **Página web Dinámica.** Puede cambiar según el usuario.
- **Portal.** Tiene acceso a recursos y servicios relacionados con un tema común.
- **Tienda virtual o comercio electrónico.** Tienda de Internet.
- **Página web con "Gestor de Contenidos".** Sitio web cuyo contenido se actualiza a través de un panel de gestión por parte del admin, la web puede controlarse por el cliente.

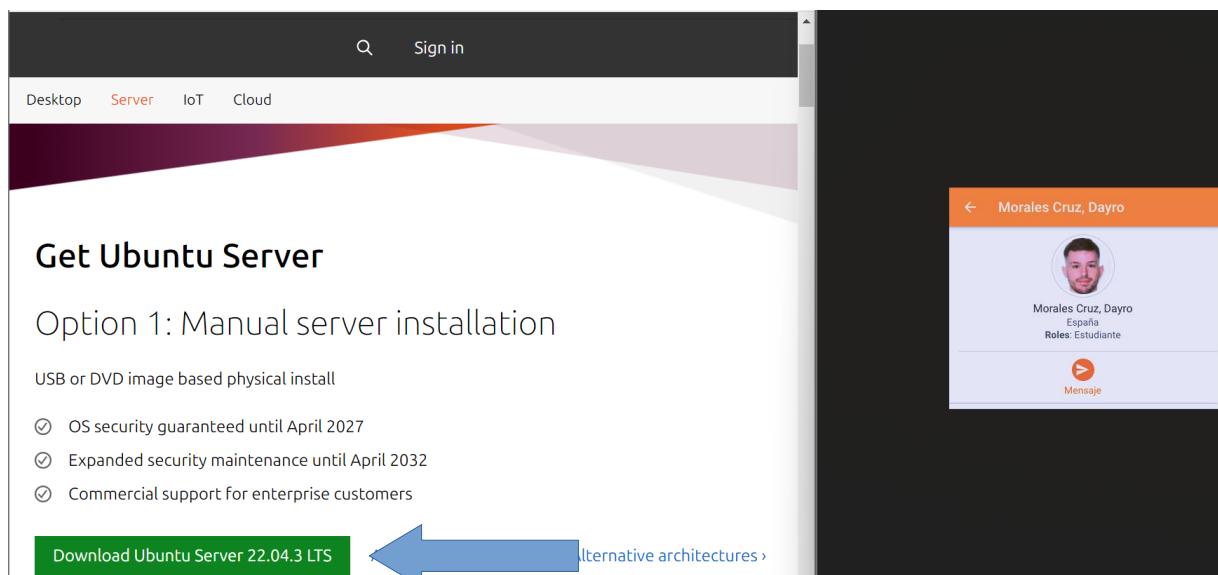
Del lado del servidor:

- **CGI (Common Gateway Interface):** Está pensado para permitir que un cliente web pueda acceder a un programa que se ejecuta en un servidor web, donde este generará contenido **dinámico**. Este estándar es utilizado para acceder a información almacenada en bases de datos, para implementar motores de búsqueda, gestionar datos de formularios, foros, juegos en línea, etc.
- **ASP (Active Server Pages):** Se ejecutan del lado del servidor, de este modo se forman los resultados que luego se mostrarán en el navegador de cada equipo cliente. Existen versiones de ASP para Unix y Linux, a pesar de que fue una tecnología desarrollada por Microsoft. Un ejemplo son los buscadores, donde un usuario realiza una petición de información y el servidor nos entrega un resultado de la petición.
- **PHP (Hypertext Preprocessor):** PHP es similar a ASP y puede ser usado en circunstancias similares. Es muy eficiente, permitiendo el acceso a bases de datos empleando servidores como **MySQL**. Suele utilizarse para crear páginas dinámicas complejas.
- **Java:** en el ecosistema de Java existe un conjunto de tecnologías pensadas para desarrollo de aplicaciones web. Las tecnologías más conocidas en el entorno web de Java son **JSP** (JavaServer Pages), **JSF** (JavaServer Faces) y los **servlets**. JSP es a PHP y ASP.
- **JavaScript:** Conseguir que el mismo desarrollador/a pueda implementar tanto la parte que se ejecuta en el servidor (**back-end**) como la página que se visualiza en el cliente (**front-end**) se denomina **Full Stack Development**. Un ejemplo son Express.js y Hapi.js (ambos funcionan sobre Node.js).

Ejercicio 3.

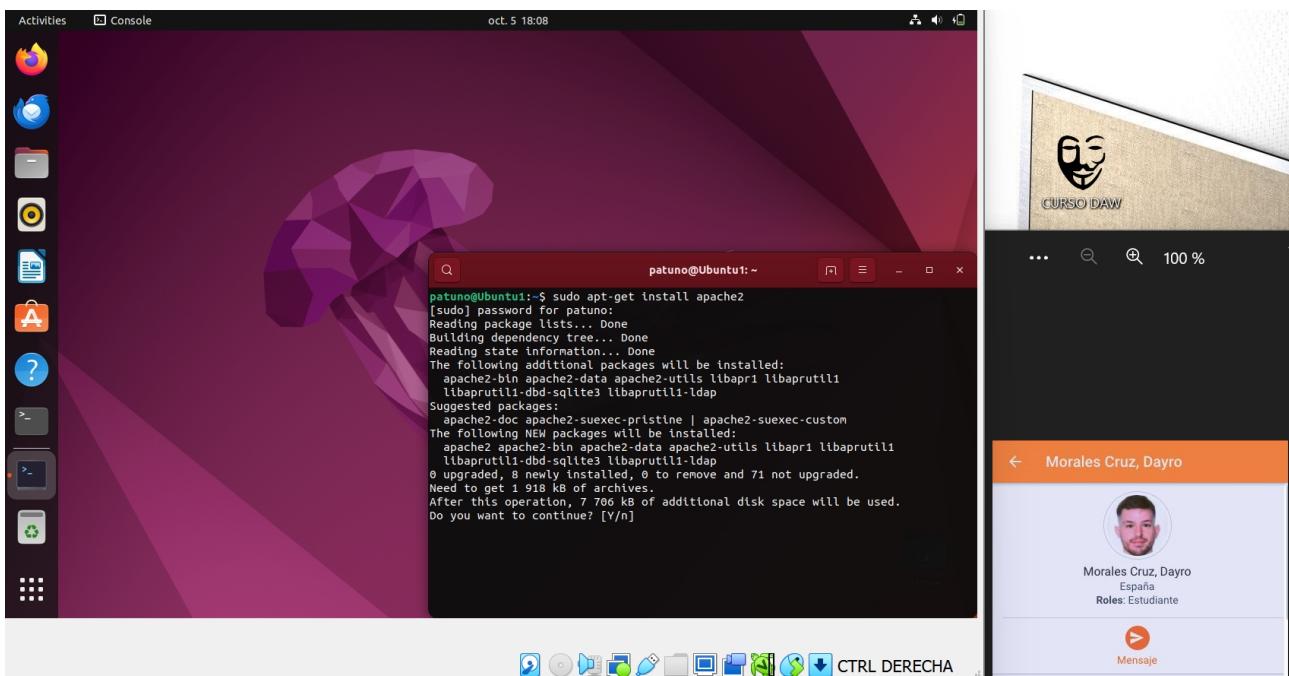
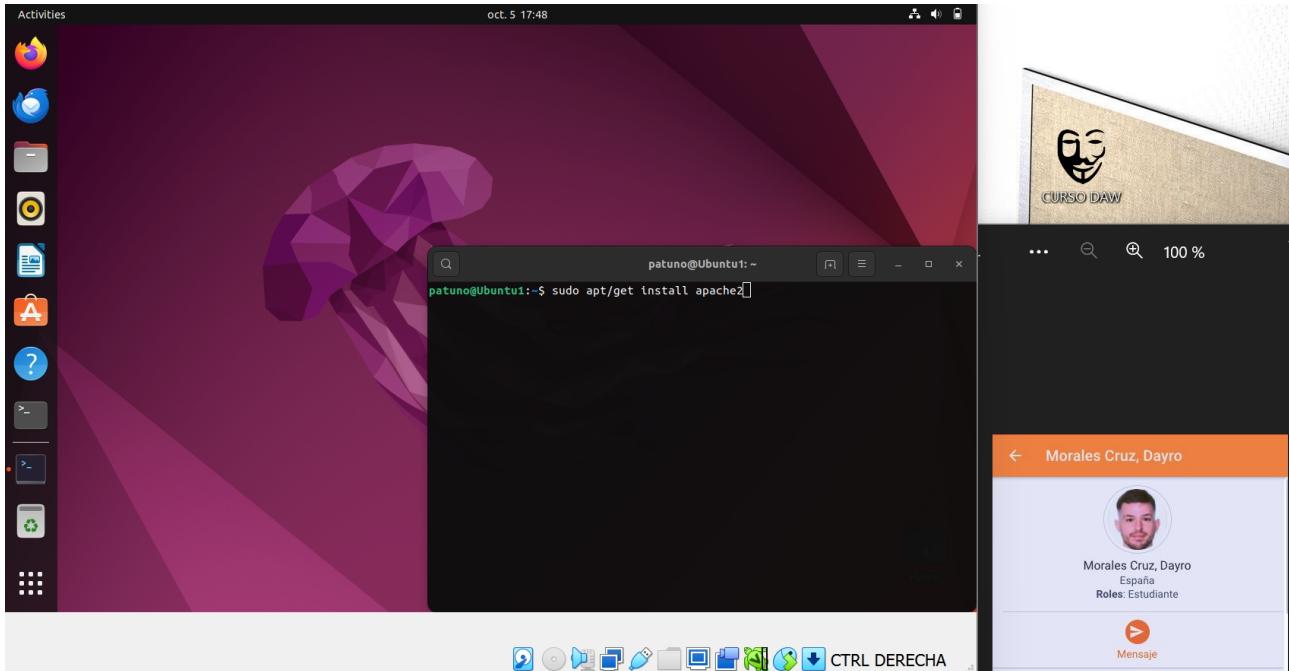
Instalando nuestro servidor web. Dispones de una máquina (o máquina virtual) que cuenta con el sistema operativo Ubuntu 20/22 recientemente actualizado, con el entorno de red configurado y conexión a Internet. Además, estás trabajando con la cuenta del usuario root. Indica cada uno de los pasos, y comandos implicados en ellos, para conseguir hacer lo siguiente:

Primeros con nuestra máquina virtual instalada, descargaremos el servido de Ubuntu desde la página oficial de [Ubuntu](#)



- Instalar el servidor web Apache desde terminal de comandos.

Instalamos apache



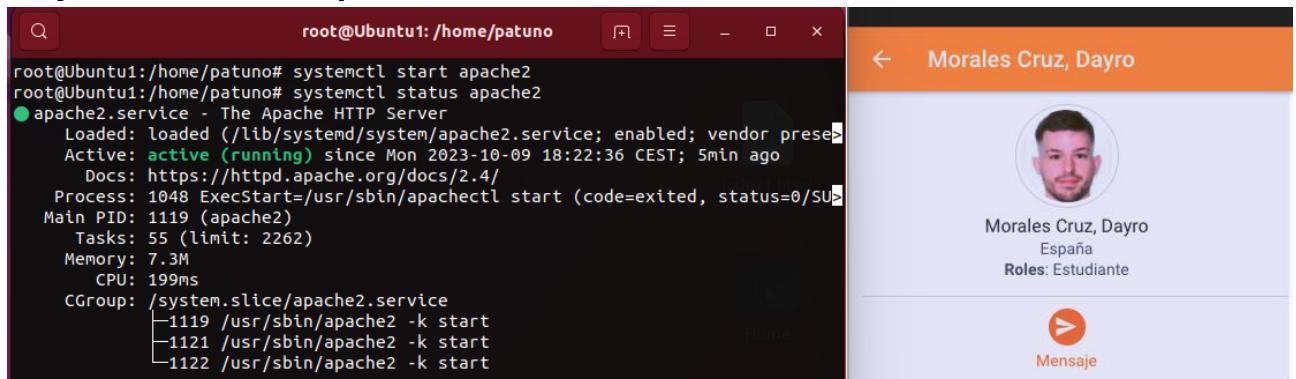
- Arrancar, reiniciar, comprobar el estado y parar el servidor web Apache.

Iniciamos apache con el comando

\$ systemctl start apache2

y a continuación, comprobamos que esta iniciado con el comando

\$ systemctl status apache2



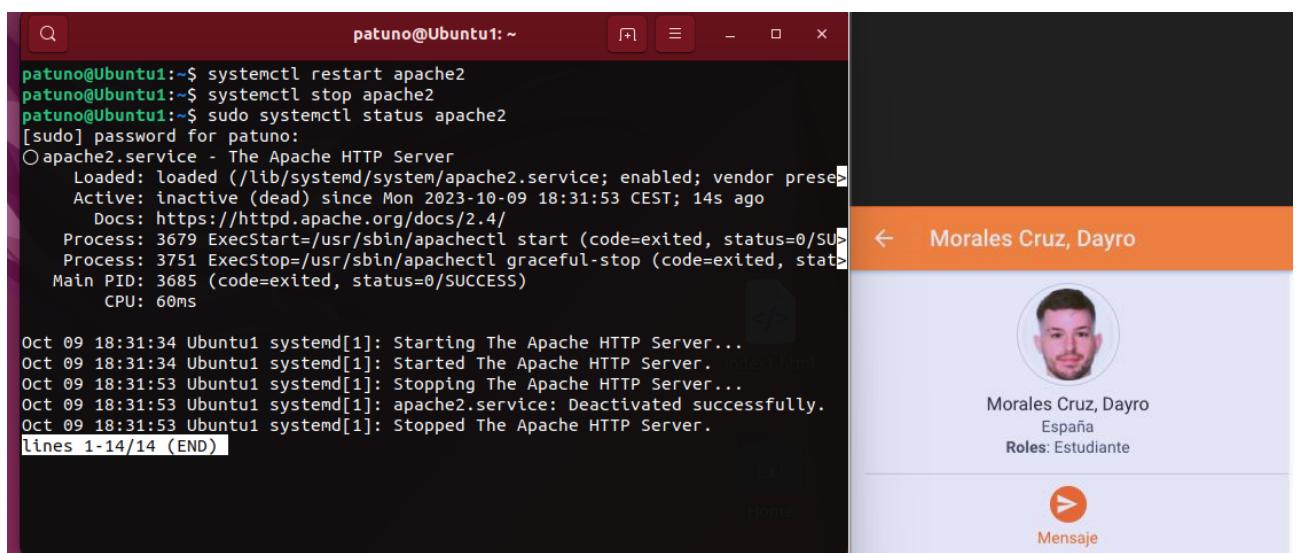
```
root@Ubuntu1:/home/patuno# systemctl start apache2
root@Ubuntu1:/home/patuno# systemctl status apache2
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor prese...
   Active: active (running) since Mon 2023-10-09 18:22:36 CEST; 5min ago
     Docs: https://httpd.apache.org/docs/2.4/
   Process: 1048 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SU...
 Main PID: 1119 (apache2)
    Tasks: 55 (limit: 2262)
   Memory: 7.3M
      CPU: 199ms
     CGroup: /system.slice/apache2.service
             └─1119 /usr/sbin/apache2 -k start
                 ├─1121 /usr/sbin/apache2 -k start
                 ├─1122 /usr/sbin/apache2 -k start
                 
```

Reiniciamos apaches con el comando

\$ systemctl restart apache2

y para parar el programa usamos

\$ systemctl stop apache2

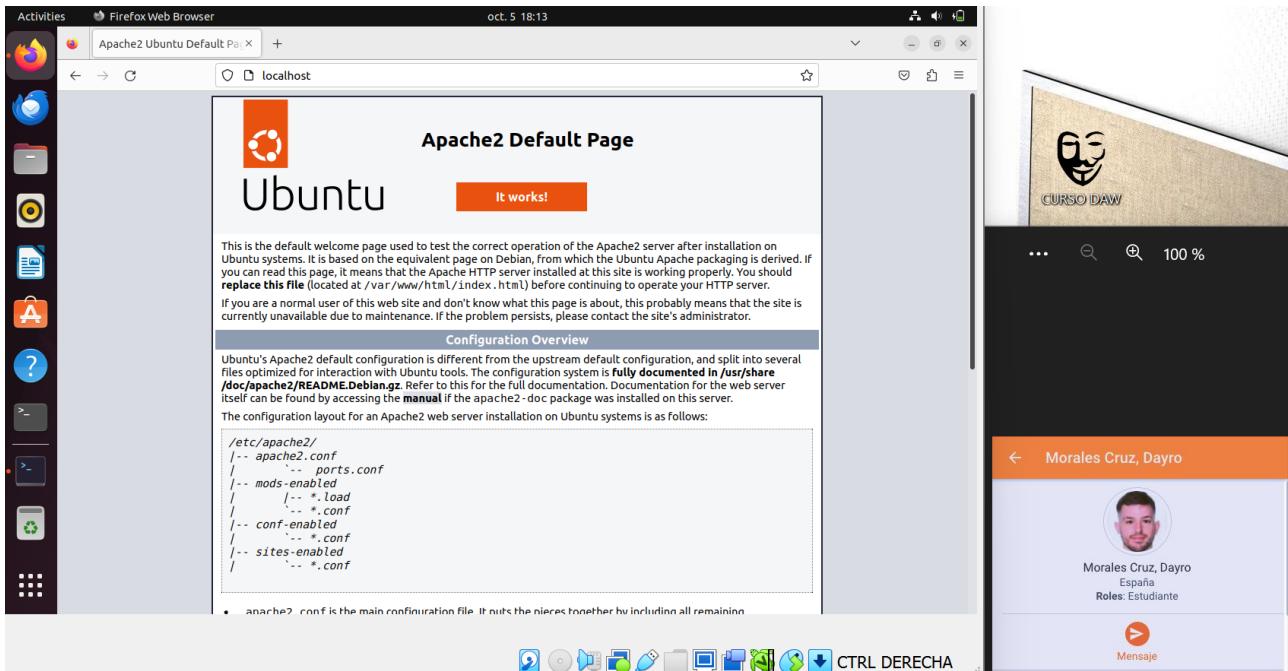


```
patuno@Ubuntu1:~$ systemctl restart apache2
patuno@Ubuntu1:~$ systemctl stop apache2
patuno@Ubuntu1:~$ sudo systemctl status apache2
[sudo] password for patuno:
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor prese...
   Active: inactive (dead) since Mon 2023-10-09 18:31:53 CEST; 14s ago
     Docs: https://httpd.apache.org/docs/2.4/
   Process: 3679 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SU...
   Process: 3751 ExecStop=/usr/sbin/apachectl graceful-stop (code=exited, stat...
 Main PID: 3685 (code=exited, status=0/SUCCESS)
    CPU: 60ms

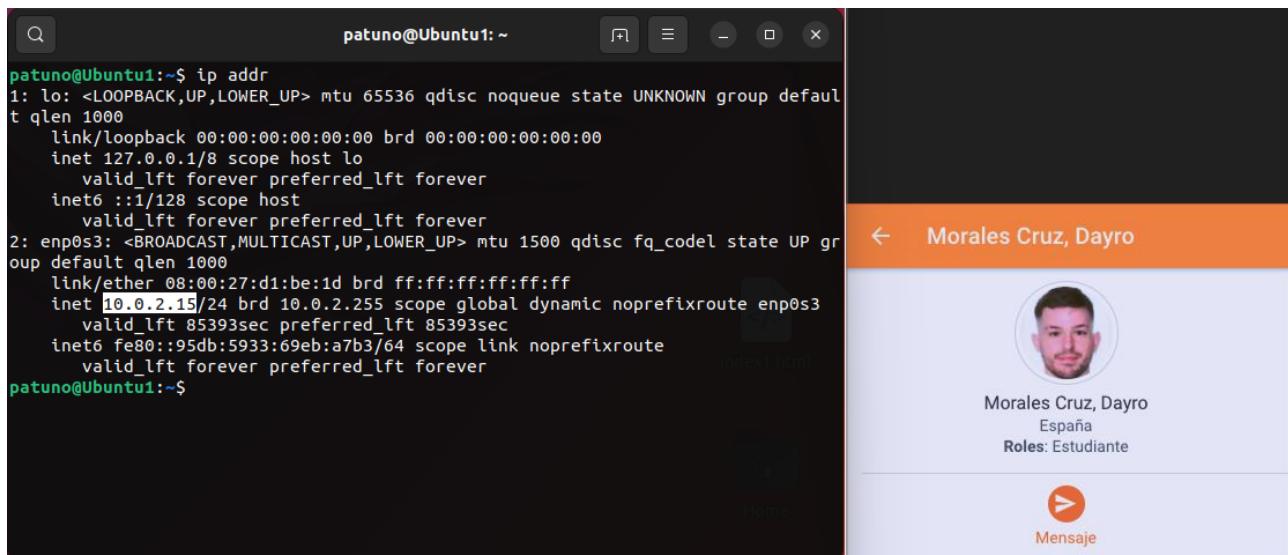
Oct 09 18:31:34 Ubuntu1 systemd[1]: Starting The Apache HTTP Server...
Oct 09 18:31:34 Ubuntu1 systemd[1]: Started The Apache HTTP Server. index.html
Oct 09 18:31:53 Ubuntu1 systemd[1]: Stopping The Apache HTTP Server...
Oct 09 18:31:53 Ubuntu1 systemd[1]: apache2.service: Deactivated successfully.
Oct 09 18:31:53 Ubuntu1 systemd[1]: Stopped The Apache HTTP Server.
lines 1-14/14 (END)
```

- Comprobar que está funcionando el servidor Apache desde un navegador web.

Podemos acceder a la página principal introduciendo en el navegador <http://localhost> o <http://127.0.0.1>

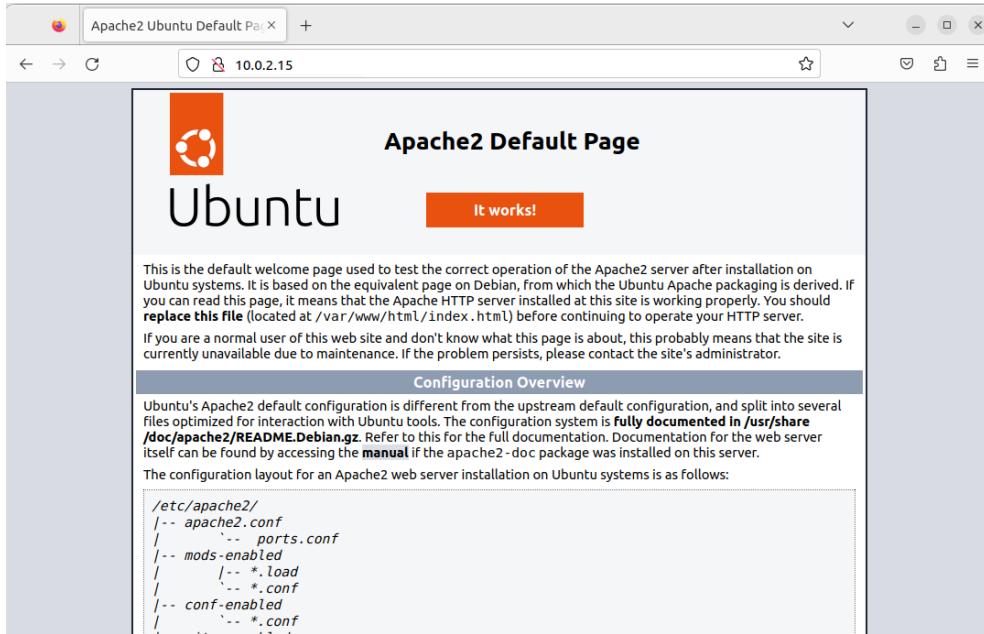


Para averiguar nuestro puerto podemos hacerlo mediante el comando
\$ ip addr

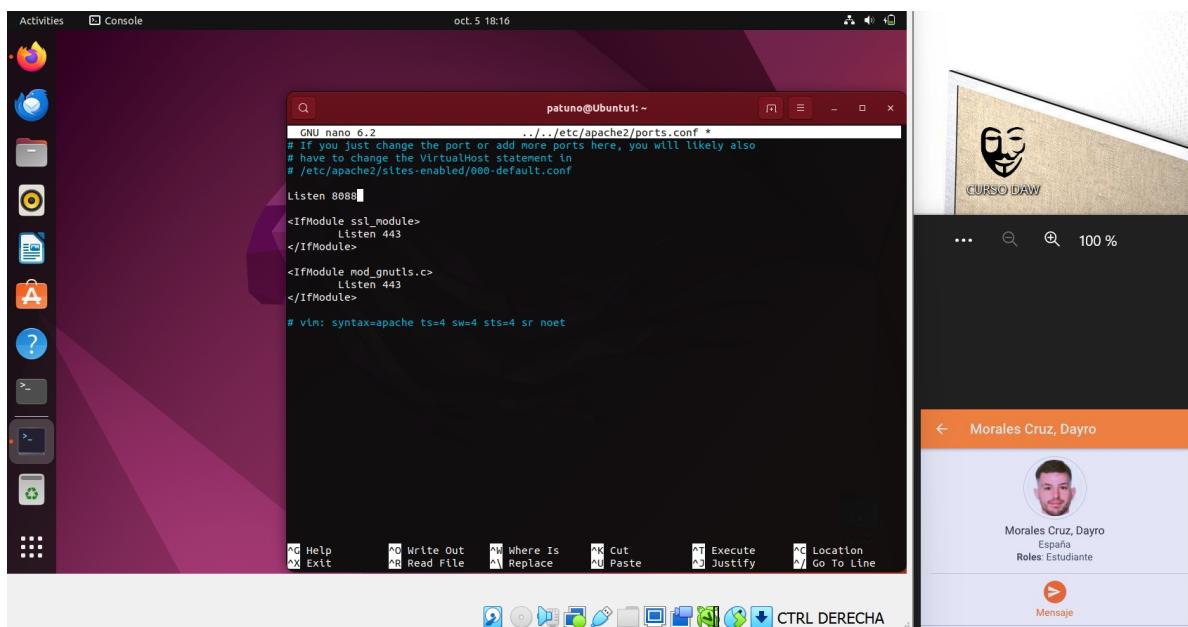


En mi caso la IP es **10.0.2.15**

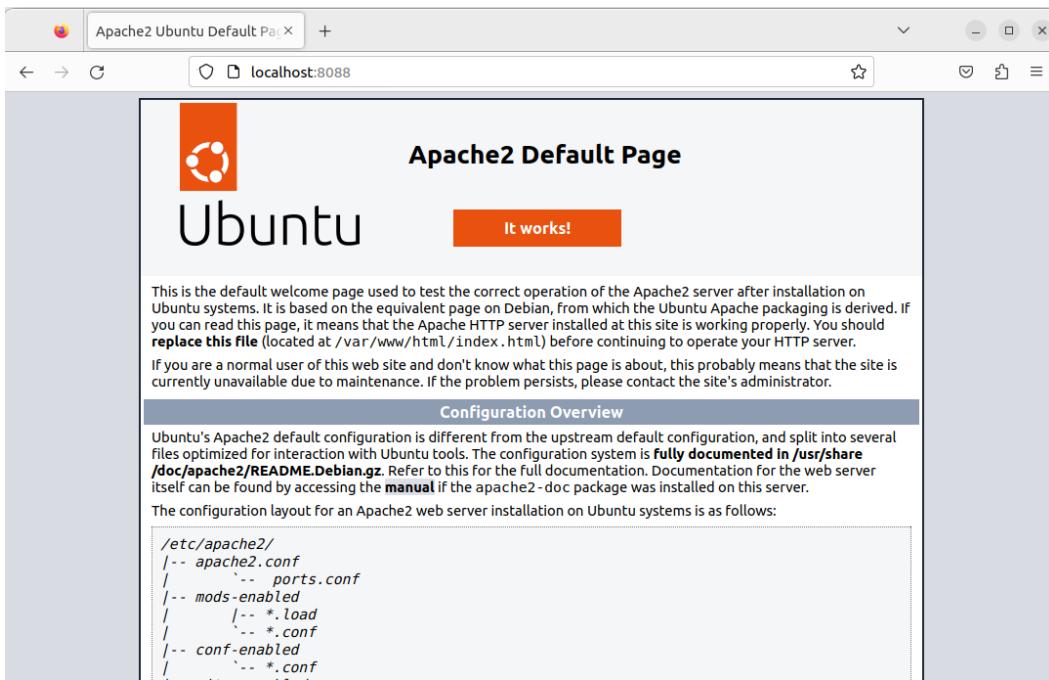
- Cambiar el puerto por el cual está escuchando Apache pasándolo al puerto 8088 y comprueba de nuevo desde tu navegador que está funcionando.



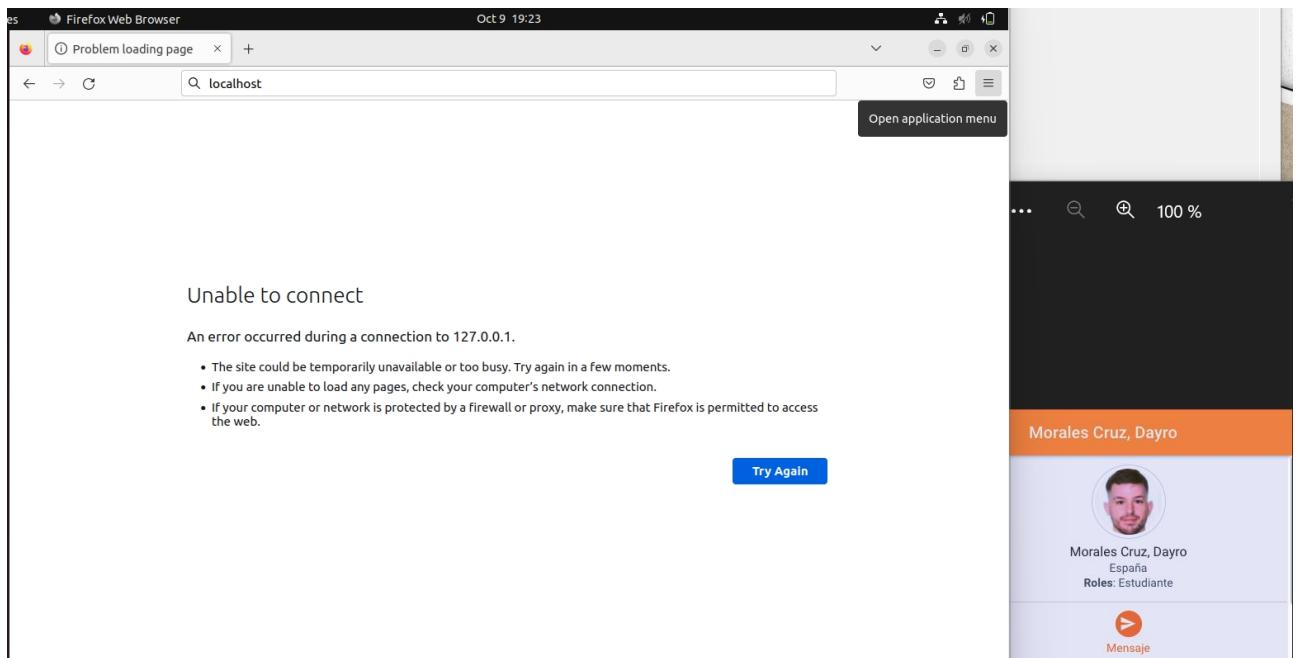
Cambiamos nuestro canal de escucha del 80 al 8088
\$ nano /etc/apache2/ports.conf para modificar el archivo



Comprobamos que podemos acceder desde el nuevo puerto



y que el anterior ya no es accesible



- Cambiar la página web por defecto para que aparezcan tus apellidos mas un pantallazo de tu inicio del curso.

Accedemos al archivo archivo HTML por defecto de apache y lo cambiamos por uno con los datos solicitados con el comando

\$ nano /var/www/html/index.html

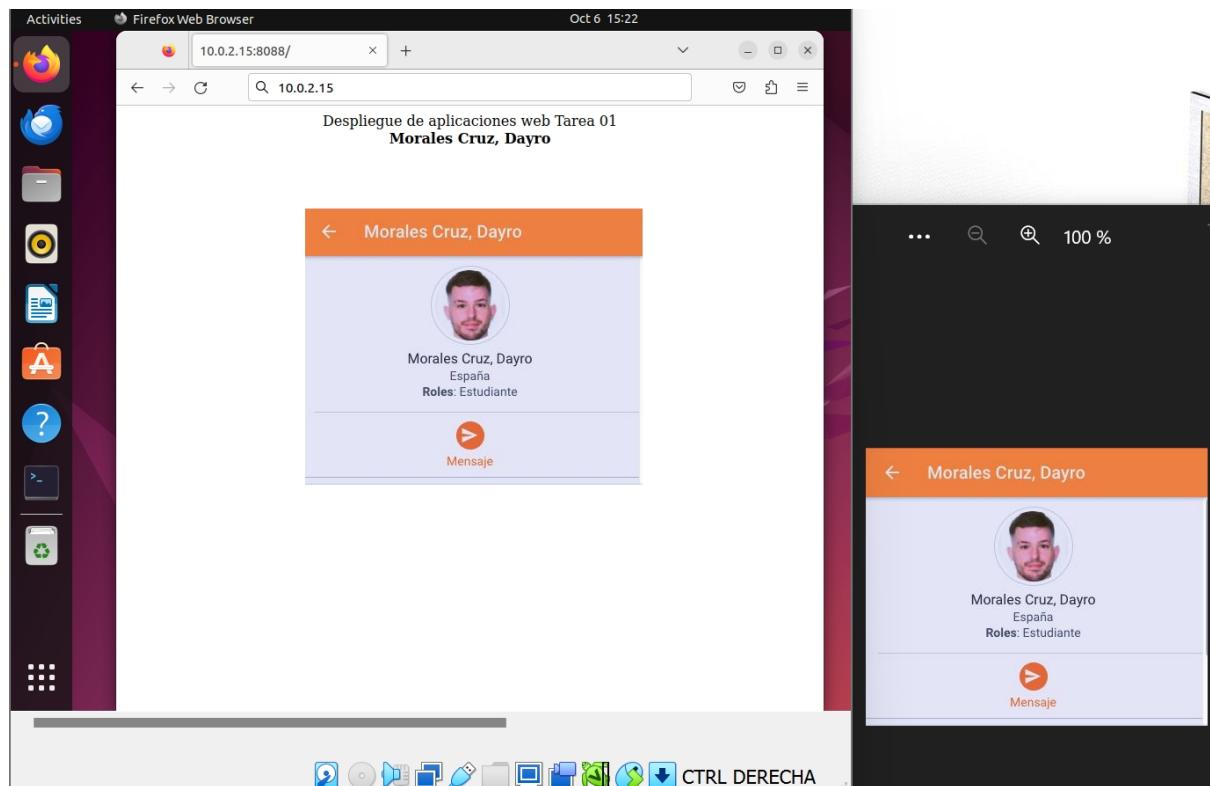
The terminal window shows the command `root@Ubuntu1: /home/patuno` and the file `/var/www/html/index.html`. The content of the file is:

```
GNU nano 6.2                               /var/www/html/index.html *
<!DOCTYPE html>
<html lang="es" dir="ltr">
  <meta charset="UTF-8">
  <head>
    <tittle><center> Despliegue de aplicaciones web Tarea 01 </cent><tittle>
  </head>
  <body>
    <hi><center><strong> Morales Cruz, Dayro </strong></center></hi>
    <center></center>
  </body>
</html>
```

The browser window shows a profile page with the following details:

Morales Cruz, Dayro
Morales Cruz, Dayro
España
Roles: Estudiante
Morales Cruz, Dayro
España
Roles: Estudiante
Mensaje

Comprobamos que el cambio se ha producido

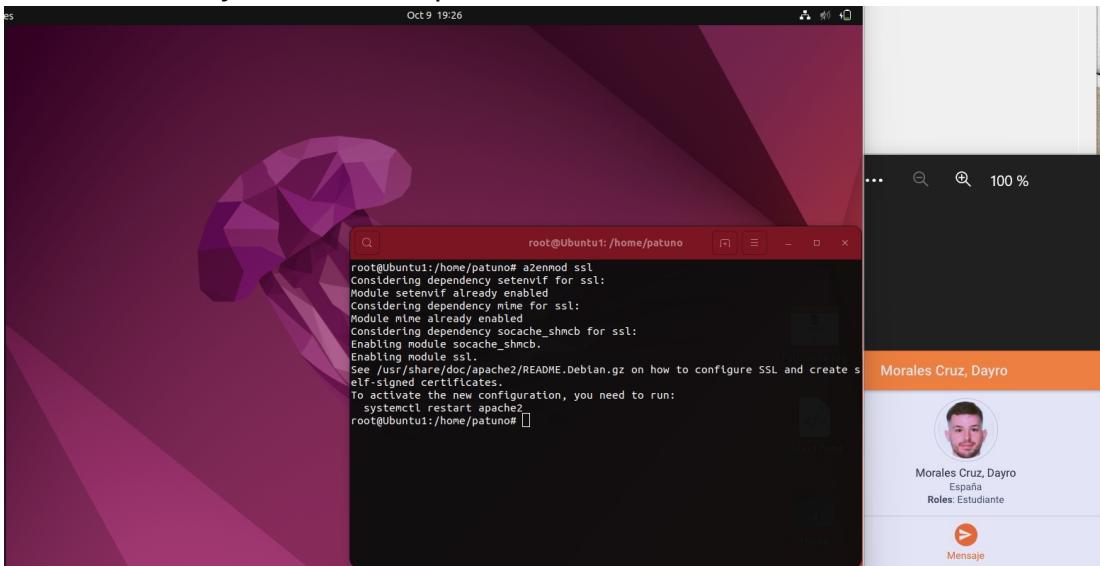


Ejercicio 4.

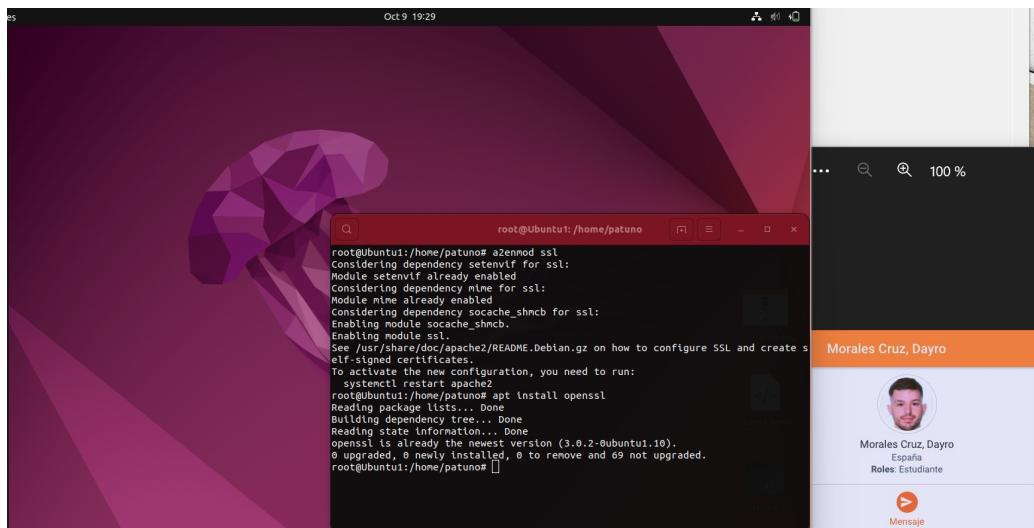
Haciendo las comunicaciones más seguras. Partiendo de la instalación de Apache del ejercicio anterior, instala un certificado de seguridad (SSL) en tu servidor y habilita el módulo de Apache correspondiente para que funcione correctamente. Comprueba desde tu navegador que ahora puedes acceder al servidor usando el protocolo seguro (https).

Habilitamos el módulo de apache encargado de las gestiones SSL

\$ a2enmod ssl y reiniciamos apache



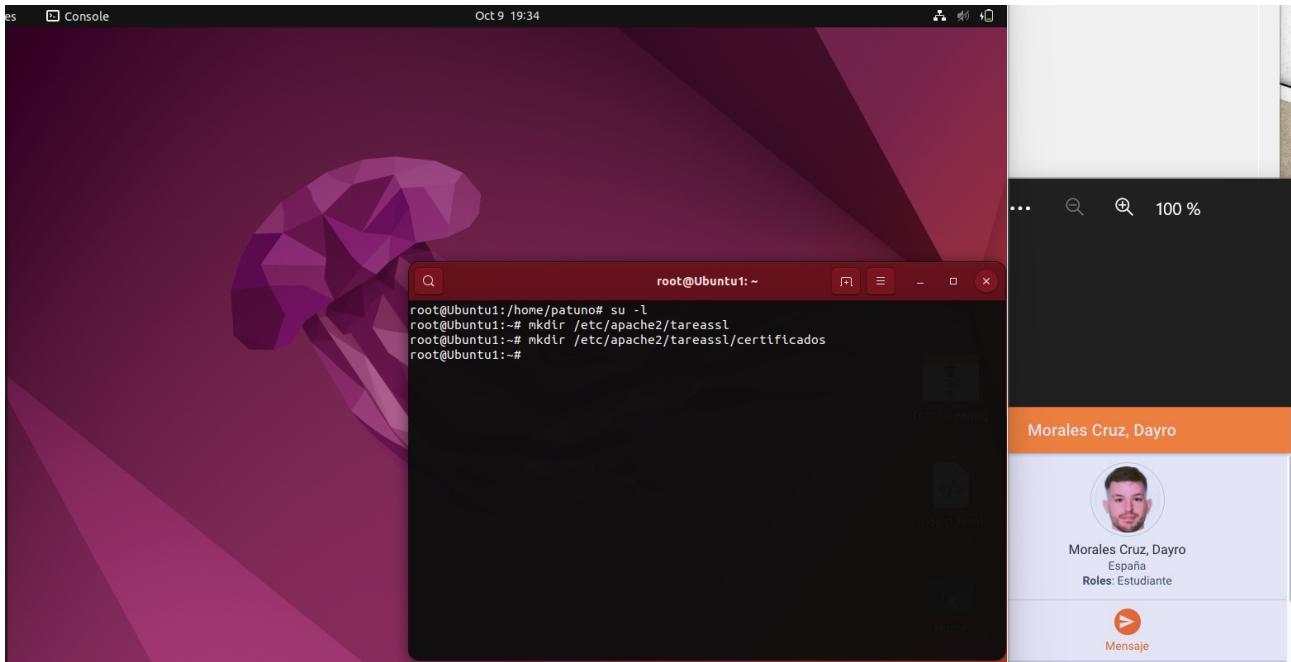
Ahora instalaremos openssl mediante \$ apt install openssl



Creamos las carpetas donde almacenaremos nuestros certificados mediante

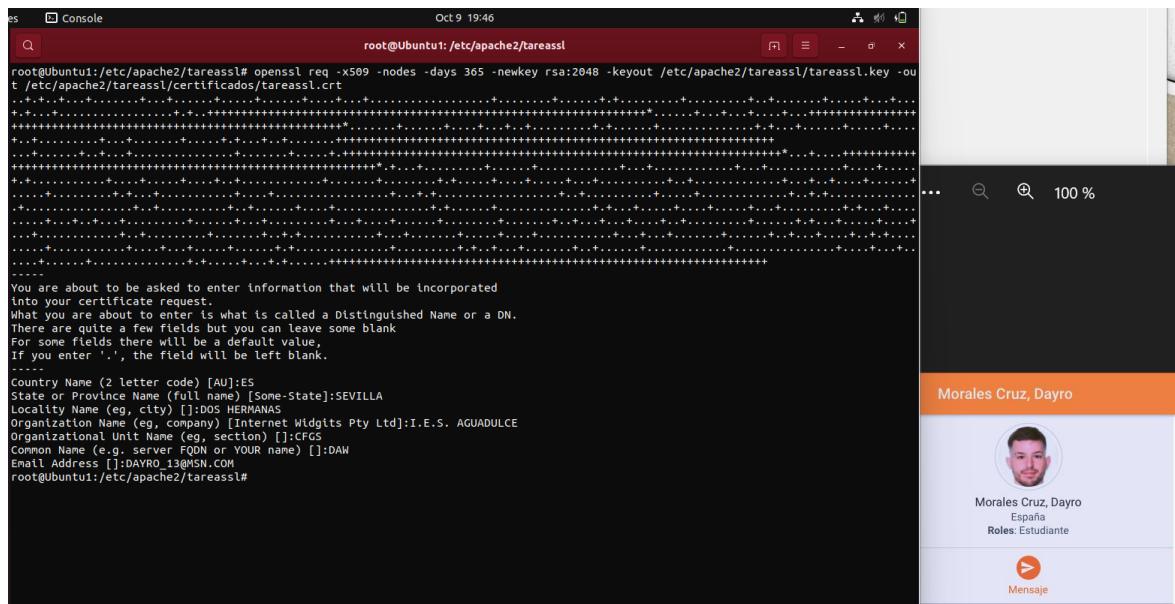
```
$ mkdir /etc/apache2/tareassl
```

```
$ mkdir /etc/apache2/tareassl/certificados
```



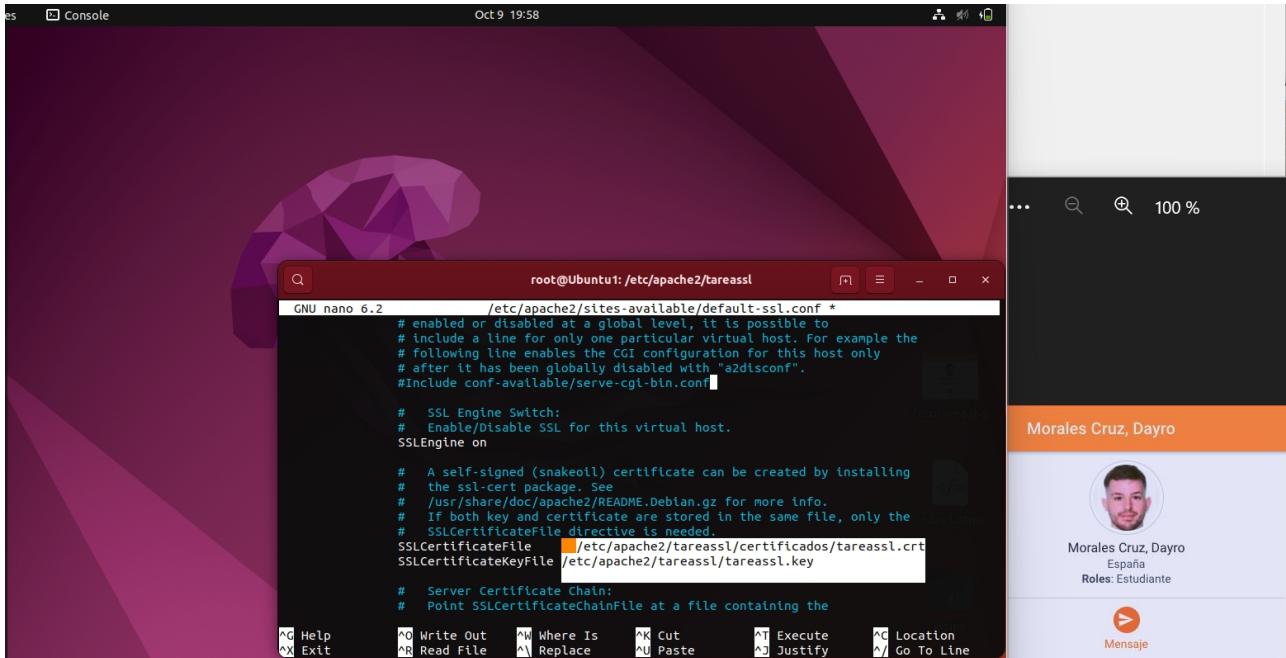
Ingresamos en openssl los parámetros para que nos genere un certificado en nuestras carpetas

```
$ openssl req -x509 -nodes -days 365 -newkey rsa:2048  
-keyout /etc/apache2/tareassl/tareassl.key -out /etc/apache2/tareassl/certificados/tareassl.crt
```



Ahora modificaremos el fichero por defecto de Apache en la carpeta sites-available

\$ nano /etc/apache2/sites-available/default-ssl.conf



```
root@Ubuntu1: /etc/apache2/tareassl          Oct 9 19:58
GNU nano 6.2          /etc/apache2/sites-available/default-ssl.conf *
# enabled or disabled at a global level, it is possible to
# include a line for only one particular virtual host. For example the
# following line enables the CGI configuration for this host only
# after it has been globally disabled with "a2disconf".
#Include conf-available/serve-cgi-bin.conf

#   SSL Engine Switch:
#   Enable/Disable SSL for this virtual host.
SSLEngine on

#   A self-signed (snakeoil) certificate can be created by installing
#   the ssl-cert package. See
#   /usr/share/doc/apache2/README.Debian.gz for more info.
#   If both key and certificate are stored in the same file, only the
#   SSLCertificateFile directive is needed.
SSLCertificateFile    /etc/apache2/tareassl/certificados/tareassl.crt
SSLCertificateKeyFile /etc/apache2/tareassl/tareassl.key

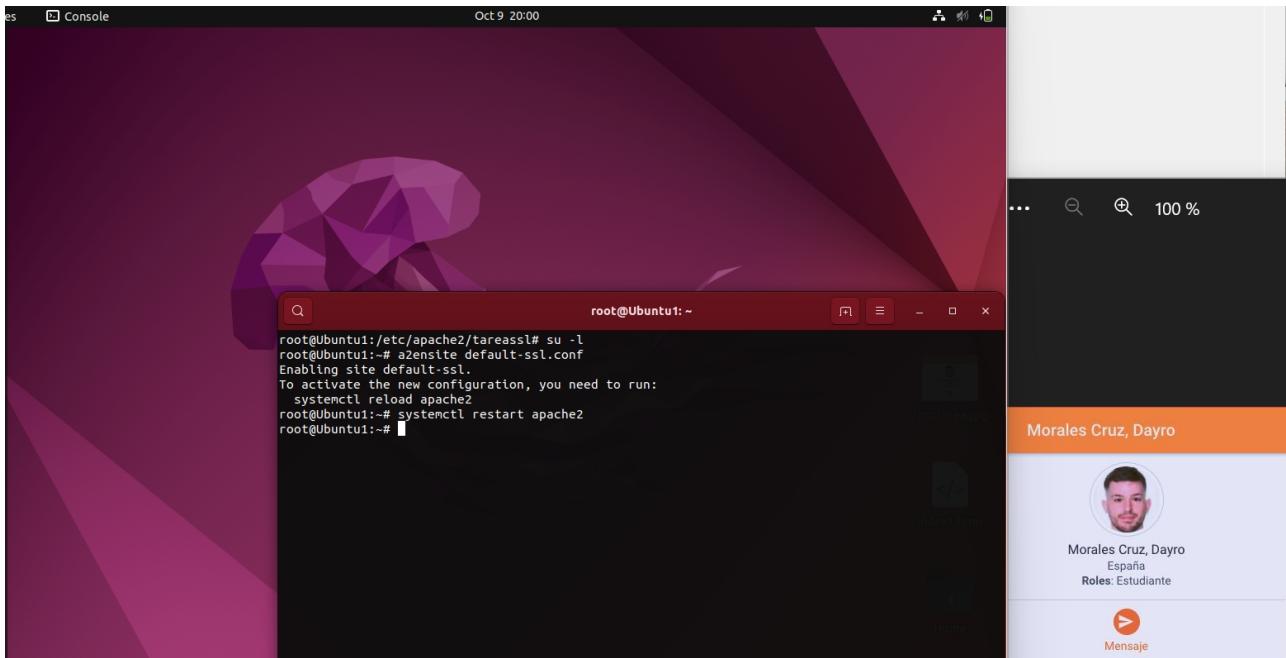
#   Server Certificate Chain:
#   Point SSLCertificateChainFile at a file containing the
#   intermediate certificates in the chain.
#SSLCertificateChainFile

^G Help      ^W Write Out     ^M Where Is      ^K Cut        ^T Execute     ^C Location
^X Exit      ^R Read File     ^L Replace      ^U Paste       ^J Justify     ^A Go To Line
```

habilitamos el sitio web seguro por defecto y reiniciamos

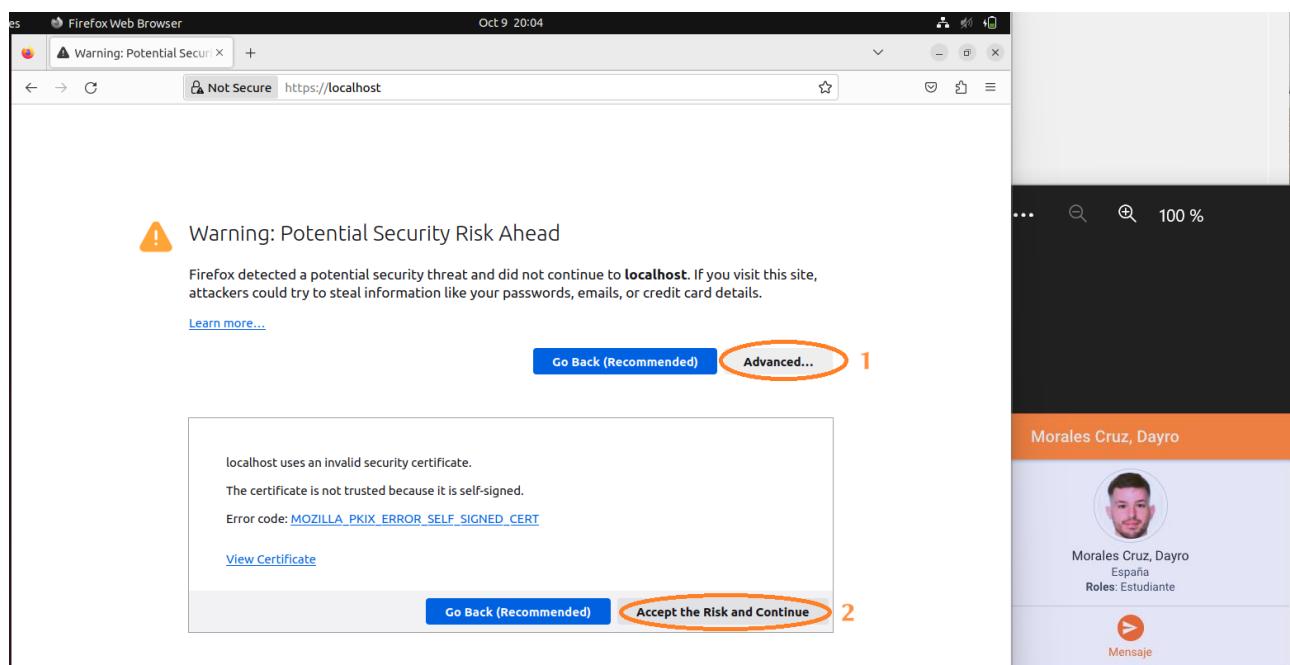
\$ a2ensite default-ssl.conf

\$ systemctl restart apache2

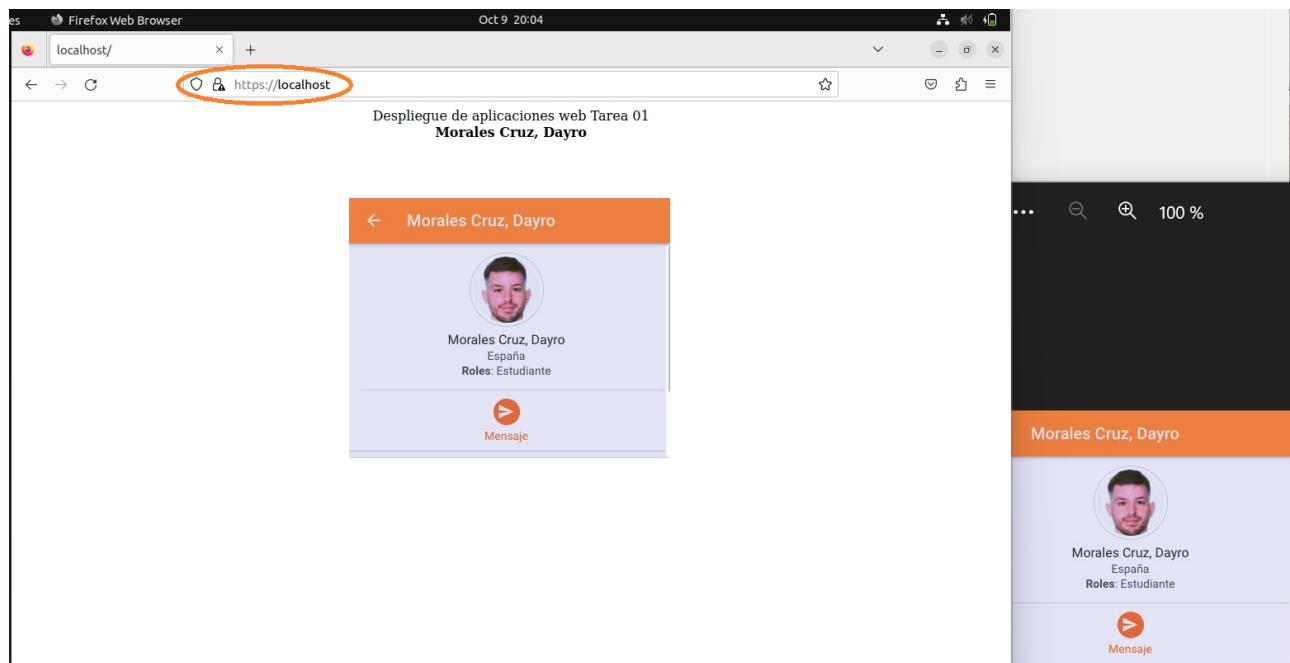


```
root@Ubuntu1:/etc/apache2/tareassl# su -
root@Ubuntu1:# a2ensite default-ssl.conf
Enabling site default-ssl
To activate the new configuration, you need to run:
  systemctl reload apache2
root@Ubuntu1:# systemctl restart apache2
root@Ubuntu1:~#
```

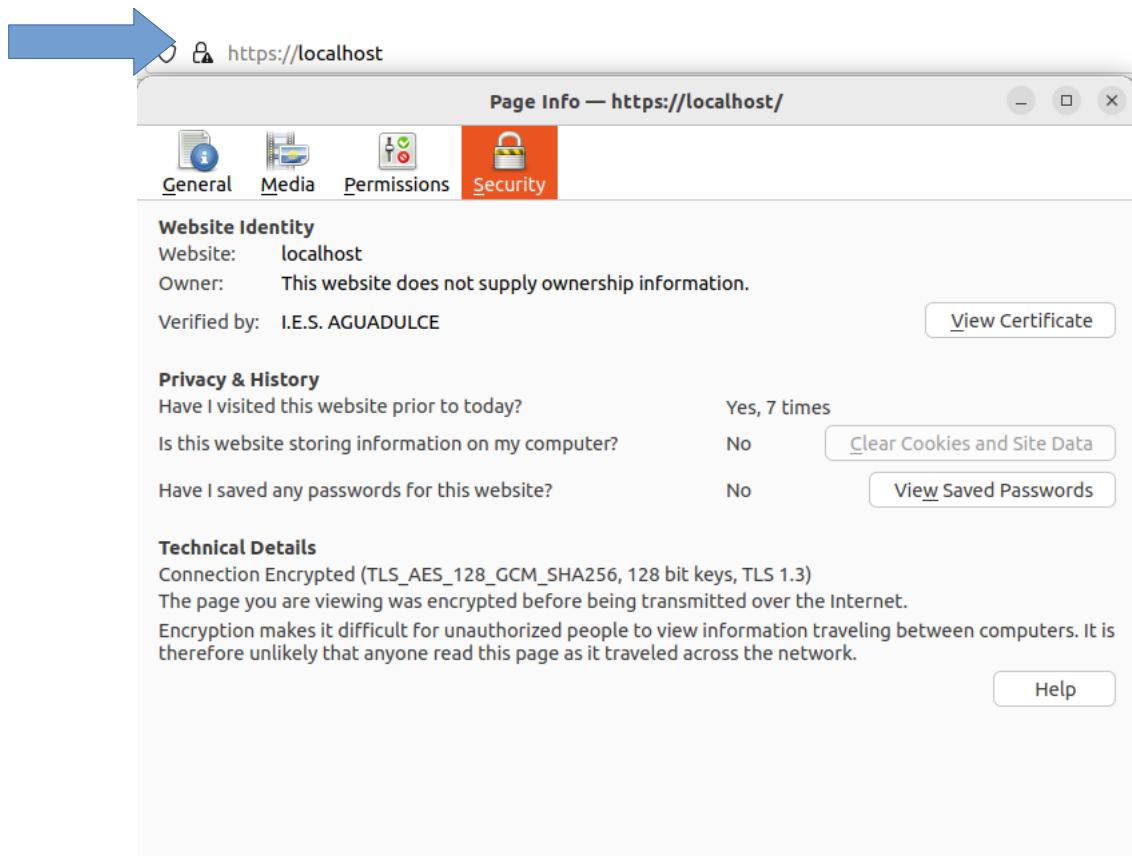
Accederemos a nuestro localhost mediante el certificado de seguridad **https://** Nos saldra un aviso de seguridad, pulsaremos en avanzado y en asumir el riesgo y continuar



ahora accedemos a nuestra pagina por defecto con certificado seguro **https://**



si abrimos el icono del candado de la derecha podremos ver la informacion de nuestro certificado



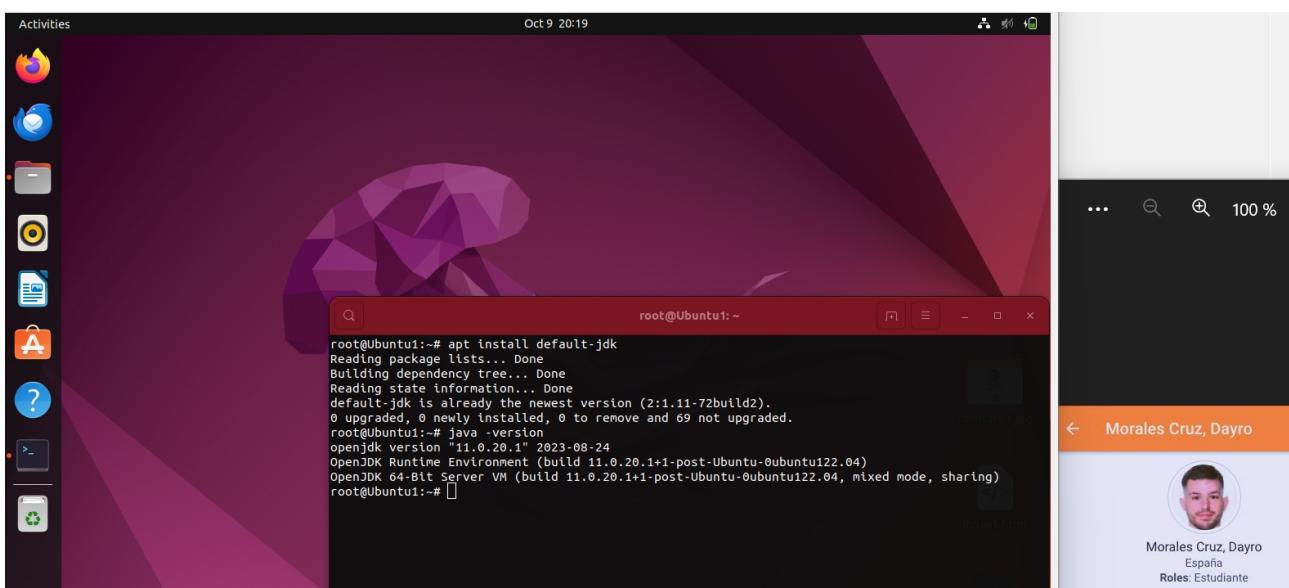
Ejercicio 5.

Instalando un servidor de aplicaciones. Partiendo de la instalación de Apache del ejercicio 3, realiza la instalación del servidor de aplicaciones Apache Tomcat (es recomendable instalar una versión lo más reciente posible, tomcat10). Importante: para esta actividad solo hay que hacer la instalación y comprobar desde nuestro navegador que tenemos acceso a la página principal del servidor de aplicaciones. No hay que configurar el acceso al panel de administración ni crear usuarios.

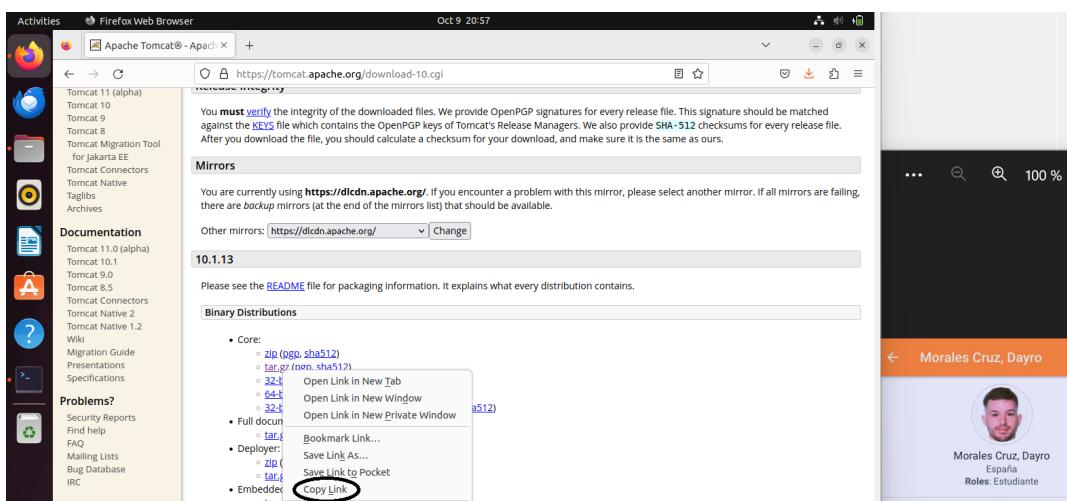
Instalaremos primero las librerías de java y corroboramos la versión instalada

```
$ apt install default-jdk
```

```
$ java -version
```



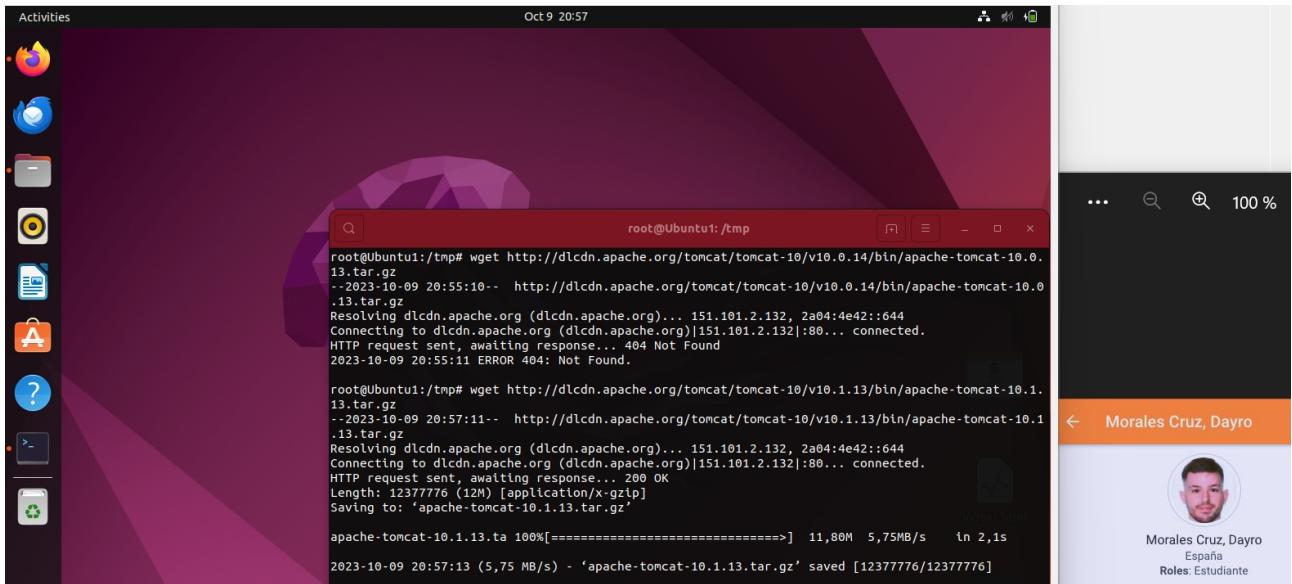
utilizaremos el Link del archivo .gz para descargar nuestro programa en la página oficial de [Tomcat](#)



instalamos en la carpeta tmp mediante el comando

```
$ cd /tmp
```

```
$ wget http://dlcdn.apache.org/tomcat/tomcat-10/v10.0.14/bin/apache-tomcat-10.0.13.tar.gz
```

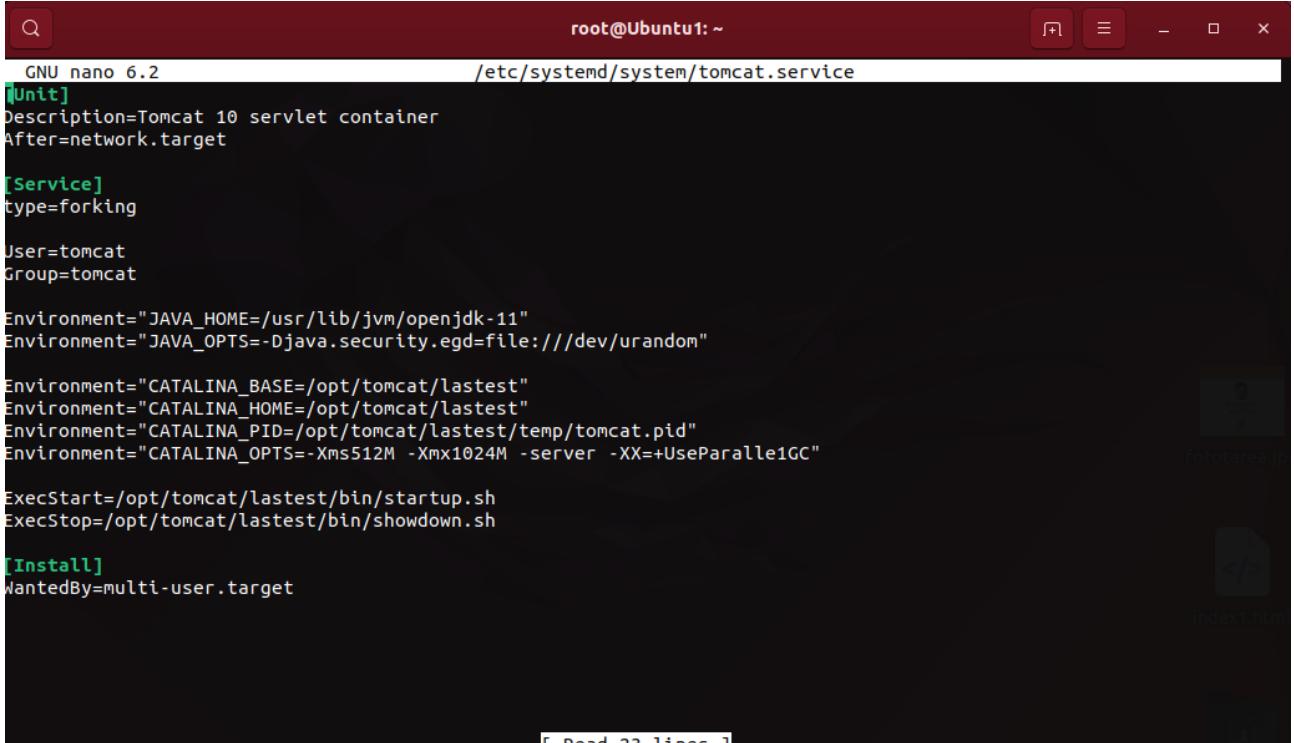


A continuación, comprobaremos nuestra versión de jdk para, posteriormente, crear un archivo para no tener que ejecutar de manera manual los script de tomcat, similar al manejo de Apache Web Server.

```
root@Ubuntu1:~# ls /usr/lib/jvm
default-java  java-1.11.0-openjdk-amd64  java-11-openjdk-amd64  openjdk-11
root@Ubuntu1:~#
```

Nosotros tenemos la versión 11

Copiaremos el siguiente contenido en el archivo .service de Tomcat



```
GNU nano 6.2                               /etc/systemd/system/tomcat.service
[Unit]
Description=Tomcat 10 servlet container
After=network.target

[Service]
type=forking

User=tomcat
Group=tomcat

Environment="JAVA_HOME=/usr/lib/jvm/openjdk-11"
Environment="JAVA_OPTS=-Djava.security.egd=file:///dev/urandom"

Environment="CATALINA_BASE=/opt/tomcat/lastest"
Environment="CATALINA_HOME=/opt/tomcat/lastest"
Environment="CATALINA_PID=/opt/tomcat/lastest/temp/tomcat.pid"
Environment="CATALINA_OPTS=-Xms512M -Xmx1024M -server -XX:+UseParallelGC"

ExecStart=/opt/tomcat/lastest/bin/startup.sh
ExecStop=/opt/tomcat/lastest/bin/showdown.sh

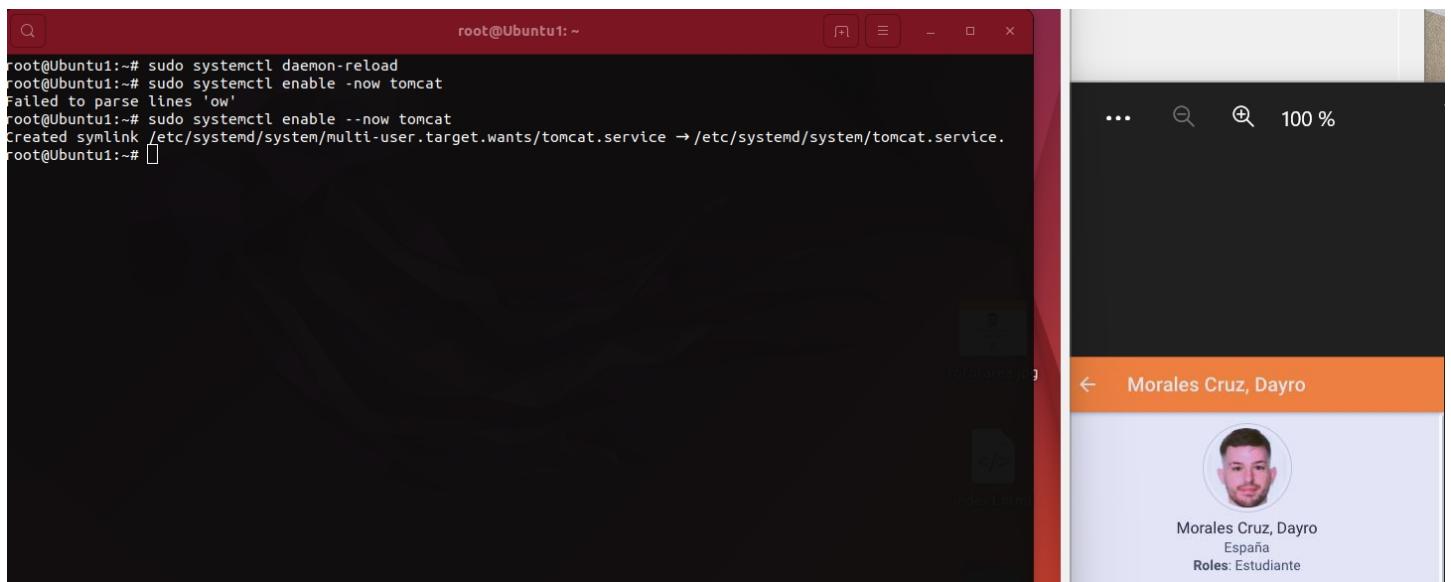
[Install]
WantedBy=multi-user.target
```

Ahora informamos al sistema operativo de que hemos creado un nuevo fichero de sistema

\$ sudo systemctl daemon-reload

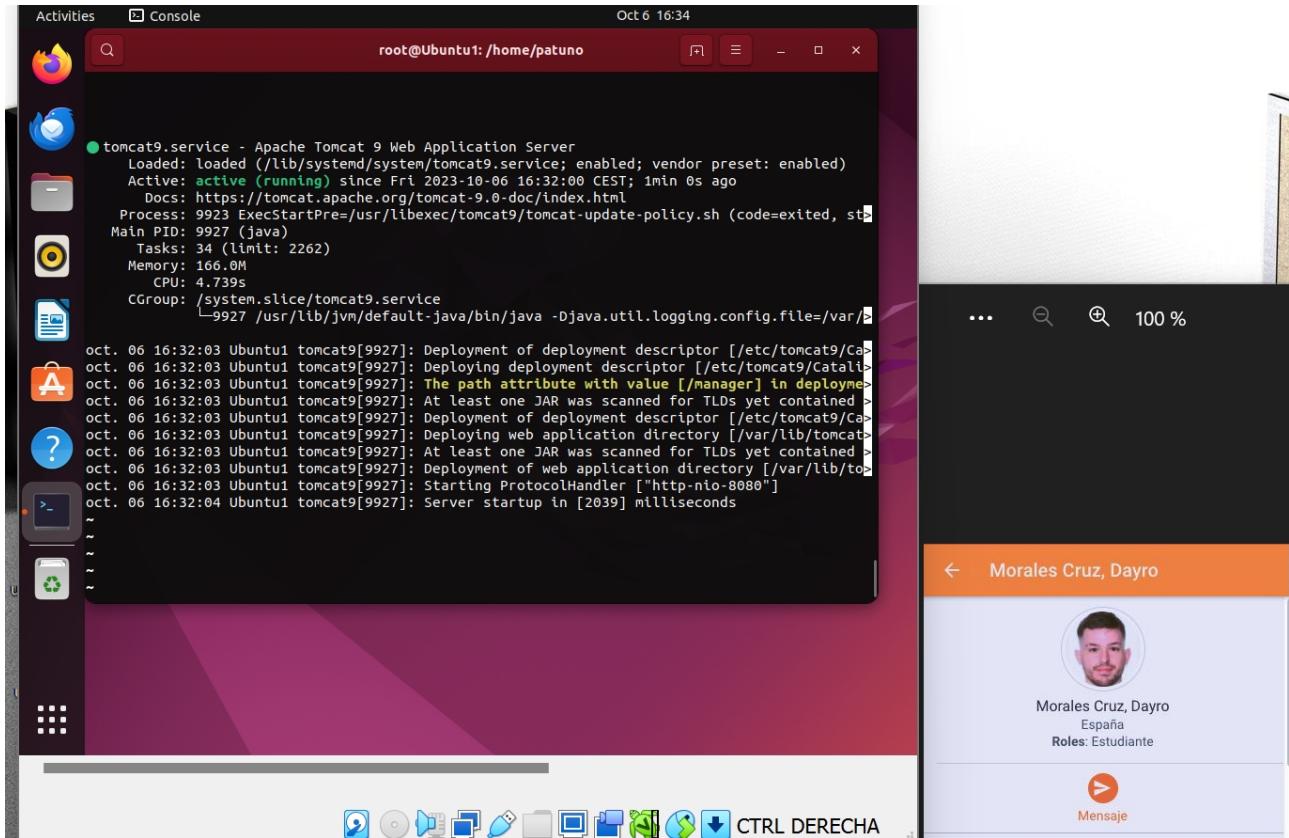
Lo habilitamos y arrancamos

\$ sudo systemctl enable --now tomcat10

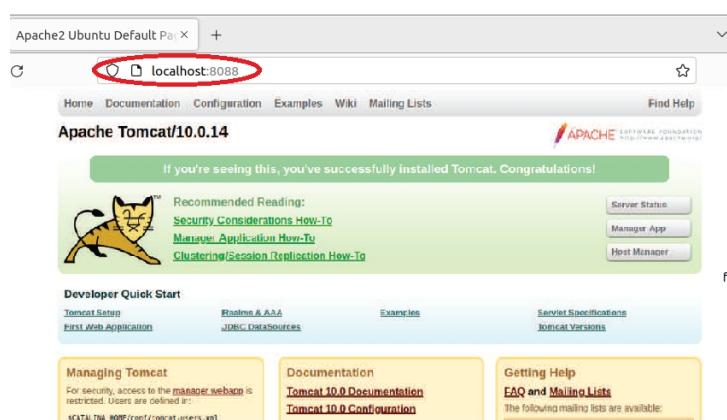


```
root@Ubuntu1:~# sudo systemctl daemon-reload
root@Ubuntu1:~# sudo systemctl enable -now tomcat
Failed to parse lines 'ow'
root@Ubuntu1:~# sudo systemctl enable --now tomcat
Created symlink /etc/systemd/system/multi-user.target.wants/tomcat.service → /etc/systemd/system/tomcat.service.
root@Ubuntu1:~#
```

Comprobamos que nuestro programa esta activo



Comprobamos que sale nuestra web de servicios de tomcat por defecto



Ejercicio 6.

Hosts virtuales. Ya sabemos que Apache permite tener más de un sitio web en un servidor, donde cada sitio puede estar asociado a un dominio diferente. Revisa el punto 5 de la documentación y explica qué es, para que se utiliza y cuál es la diferencia entre virtualhost por nombre y por IP. Pon un ejemplo de configuración de cada uno, usando las directiva 'VirtualHost', incluyendo capturas del fichero de configuración.

- **Hosts virtuales basados en nombre** son servidores web con una única dirección IP pública, y que sirve diferentes sitios web (**www.midominio.local**, **www.misitio.local**, **www.miempresa.local**, etc.). El mismo servidor web está configurado para servir diferente contenido dependiendo del dominio que se solicite por el cliente. Esta configuración se la conoce como
- **Hosts virtuales basados en IP** son servidores web tiene asignadas varias direcciones IP públicas, y una dirección IP está asociada a un sitio web y otra dirección IP está asociada a otro sitio web. Cada dominio por tanto necesita su propia dirección IP.

Aquí pondremos algunos ejemplos de virtualhost basado en **nombre** y otro en **IP**

Ejemplo de virtualhost basado en nombre:

```
<VirtualHost *:80>
    ServerName www.miservidor1.com
    DocumentRoot /var/www/html/miservidor1.com
</VirtualHost *:80>
<VirtualHost *:80>
    ServerName www.miservidor2.com
    DocumentRoot /var/www/html/miservidor2.com
</VirtualHost *:80>
```

Ejemplo de virtualhost basado en IP:

```
<VirtualHost 192.168.1.25:80>
    ServerName www.miservidor1.com
    DocumentRoot /var/www/html/miservidor1.com
</VirtualHost *:80>
<VirtualHost 192.168.1.50:80>
    ServerName www.miservidor2.com
    DocumentRoot /var/www/html/miservidor2.com
</VirtualHost *:80>
```