

Tarea 1: Implantación, Configuración y Administración de Servidores Web

Francisco Javier Sueza Rodríguez

8 de diciembre de 2023

Centro: IES Aguadulce
Ciclo Formativo: Desarrollo Aplicaciones Web (Distancia)
Asignatura: Despliegue de Aplicaciones Web
Tema: Tema 1 - Implantación, Configuración y Administración de Servidores Web

1. Ejercicio 1: Arquitecturas Web

1.1. Enunciado

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.

1.2. Solución

Una **plataforma web** es el conjunto de aplicaciones y tecnologías que nos permiten el desarrollo, despliegue y funcionamiento de una aplicación web. Aquí se van a comentar las dos más utilizadas en Linux y Windows, que son **LAMP** y **WISA**. Las cuales pasamos a analizar en la siguiente lista:

■ Linux, Apache, MySQL y PHP (LAMP)

Esta es una de las plataformas de desarrollo web más empleadas en la actualidad, aunque podemos encontrar muchas variaciones. Por ejemplo, últimamente se está empleando **PostgreSQL** como base de datos alternativa a MySQL, y el uso de **Python** en vez de **PHP** cada vez se está extendiendo más, aunque aquí vamos a analizar la configuración original, que aún sigue usándose mucho.

- **Linux** (Sistema Operativo): como sistema operativo de esta plataforma se usa Linux, un sistema operativo basado en Unix y que desde sus inicios tiene una gran orientación al trabajo en red, permitiéndonos, prácticamente con la configuración por defecto, la puesta en marcha y administración de una gran variedad de servidores.
- **Apache** (Servicio Web): Apache es uno de los servidores web más usados en el mundo. Es desarrollado por una comunidad de voluntarios y supervisado por **The Apache Software Foundation**. En la plataforma LAMP, Apache será el encargado de proporcionar el **servicio web**, procesando las peticiones de los clientes y generando las respuestas adecuadas.
- **MySQL** (Acceso a Datos): esta base de datos, actualmente desarrollada por Oracle, también es una de las más ampliamente utilizadas. Utiliza el lenguaje **SQL** con algunas diferencias respecto a otras implementaciones. En la plataforma LAMP, MySQL será la encargada gestionar el **acceso a datos**, permitiendo la creación, modificación, eliminación y lectura de los datos necesarios para el funcionamiento de la aplicación web.
- **PHP** (Contenido Dinámico): este lenguaje de programación, que tiene ya unos cuantos años, es muy empleado en el desarrollo web desde hace mucho años. Y aunque hoy en día está siendo sustituido por otros lenguajes como Java, Javascript, Ruby, Python, etc., aún siguen siendo muy empleados, en parte debido a que Wordpress, uno de los gestores de contenido más usados en el mundo está desarrollado en este lenguaje. En esta plataforma, PHP se encargará de la **generación de contenido dinámico**, es decir, estará en la **capa de negocio** de nuestra plataforma.

■ Windows, IIS, SQL y ASP (WISA)

Esta plataforma de desarrollo web, también muy empleada en la actualidad, está prácticamente desarrollada por **Microsoft**, usando en cada una de las capas diferentes aplicaciones desarrolladas por ellos. Es una plataforma más costosa que LAMP, ya que todo el software empleado es software propietario, pero también tiene muy buen soporte por parte de Microsoft.

- **Microsoft Windows** (Sistema Operativo): como sistema operativo esta plataforma usa Microsoft Windows, el sistema operativo más usado en PC actualmente, aunque en entornos de servidor Linux se usa más. Tiene diferentes versiones, algunas enfocadas más a los servidores que sus versiones de escritorio.
- **Internet Information Services** (Servidor Web): ISS es un servidor web y conjunto de aplicaciones y servicios incluidos en los sistemas Windows y que nos ayuda al despliegue de diferentes tipos de servicios, aunque en este caso, se usan para el despliegue y funcionamiento del **servidor web**, que se encargara de procesar las peticiones del cliente y generar las respuestas adecuadas.
- **SQL Server** (Acceso a Datos): SQL Server es un servidor de bases de datos desarrollado por Microsoft y uno de los más usados en la plataforma WISA. Al igual que MySQL en LAMP, proporcionará la infraestructura de base de datos para poder almacenar, leer, modificar y eliminar los datos que necesitará la aplicación para su funcionamiento.
- **Active Server Pages** (Contenido Dinámico): ASP es una tecnología que permite la generación de contenido web dinámico. Aunque no es un lenguaje de programación en si mismo, esta pensado para trabajar con otros lenguajes de programación. Frecuentemente se asocia con la plataforma **.NET** de Microsoft y con el lenguaje de programación **C#**, que en última instancia es el que nos permite realizar la programación de la plataforma.

2. Ejercicio 2: Clasificación de la Aplicaciones Web

2.1. Enunciado

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.

2.2. Solución

En este ejercicio se van a enumerar y explicar la diferentes tecnologías web que se emplean tanto en el **lado del servidor** como en el **lado del cliente**, tal y como hemos visto en esta unidad, aunque también se añadirán algunas extras.

■ Tecnologías del Lado del Servidor

- **CGI (Common Gateway Interface)**: es uno de los estándares más antiguos que permiten que los clientes accedan a los programas que se ejecutan en el servidor, donde se generará el contenido dinámico. Actualmente esta cada vez más en desuso.
- **ASP (Active Server Pages)**: esta tecnología desarrollada por Microsoft se ejecuta en el lado del servidor y se encarga de generar el contenido dinámico. Actualmente se emplea **ASP.NET**, que es la evolución de ASP y que se integra con la plataforma de desarrollo **.NET**, que es un entorno de desarrollo que integra lenguajes, bibliotecas, herramientas, etc...
- **Java**: este lenguaje de programación también se ejecuta en el lado del servidor y permite, mediante diferentes tecnologías como **JSP**, **JSF** y **servlets**, el desarrollo de páginas web dinámicas encargándose de la generación de contenido. Actualmente se usa con diferentes bibliotecas y frameworks como **Spring Boot**, que aceleran enormemente los tiempos de desarrollo y crean aplicaciones web de mayor calidad.

- **PHP (Hypertext Preprocessor)**: es uno de los lenguajes interpretados del lado del servidor que más se ha usado en el desarrollo web. Es un lenguaje bastante potente que permite la interacción con diferentes bases de datos y la generación de contenido dinámico. Actualmente suele emplearse con algunos frameworks como **Laravel** que han introducidos mejoras sustanciales en el proceso de desarrollo agilizándolo y generando código de mayor calidad.
- **Javascript**: es el lenguaje de script por antonomasia del lado del cliente, pero con los años y el interés de poder desarrollar aplicaciones del lado del servidor con el se han desarrollado algunos frameworks, como **Node**, que nos permiten realizar aplicaciones en la parte del servidor aprovechando la enorme versatilidad que nos proporciona Javascript. Recientemente, se está empezando también a usar **Deno**, un fork de Node que incluye diferentes mejoras, aunque aún no está tan extendido como el primero.
- **Python**: este lenguaje interpretado se está empleando cada vez más en el desarrollo web de la parte del servidor. Es un lenguaje muy versátil y que genera código muy elegante y sencillo. Además, posee algunos frameworks especializados en el desarrollo web, siendo el más conocido **Django**, un framework orientado al desarrollo rápido de aplicaciones web.
- **Ruby**: este lenguaje de programación interpretado y orientado a objetos, surgió en 1995 y está basado en lenguajes como Python, Perl o Smalltalk. Es un lenguaje que genera código muy limpio y elegante y que actualmente ofrece uno de los frameworks de desarrollo web más potentes, como es **Ruby On Rails**.

■ Tecnologías del Lado del Cliente:

- **HTML (HyperText Markup Language)**: HTML es un lenguaje de marcado basado en SGML y que actualmente es el estándar para la maquetación de páginas web, encontrándose en su versión HTML 5.
- **CSS (Cascading Style Sheet)**: es un lenguaje de diseño gráfico que permite definir la presentación de un documento estructurado en un lenguaje de marcado, por norma general HTML, y que hace posible la separación entre la presentación y la estructura de una página web. Actualmente suele emplearse con algunos preprocesadores como pueden ser **SASS** o **SCSS**, que añaden funcionalidades al lenguaje.
- **Java**: aunque también podemos encontrarnos con Java en el lado del cliente, cada vez se suele usar menos, aunque su presencia en los primeros años de la web era enorme, ya que permite crear pequeñas aplicaciones, conocidas como **applets**, que se pueden insertar en las páginas web.
- **JavaScript**: el lenguaje más empleado en el entorno del cliente. Es un lenguaje interpretado que permite añadir interactividad a la página web sin necesidad de realizar peticiones al servidor. Actualmente todos los navegadores web lo soportan, aunque cada navegador implementa un motor de javascript diferentes, siendo los más comunes **Spidermonkey** o **V8**.
- **TypeScript**: este lenguaje, desarrollado por Microsoft, es una versión de JavaScript a la que se le han añadido algunas funcionalidades de las que carece JavaScript, como el **tipado estático**, las **interfaces**, etc. Actualmente va ganando terreno, ya que proporciona todas las ventajas de JavaScript mejorando algunos de sus puntos flacos.
- **Librerías y Frameworks**: aunque no es una tecnología concreta, cabe destacar que hoy en día la mayor parte del desarrollo de páginas web en el lado del cliente se lleva a cabo usando alguno de los diferentes framework y librerías para el front-end que podemos encontrar. La gran mayoría usa Javascript (o TypeScript) como base, siendo los más empleados **React.js**,

Vue.js o **Angular**. Cada uno ofrece diferentes características, permitiendo más o menos libertad a la hora de desarrollar la web, aunque todos tienen algo en común, y es que aceleran enormemente los tiempos de desarrollo, y por lo tanto, los costes.

3. Ejercicio 3: Instalando Nuestro Servidor Web

3.1. Enunciado

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:

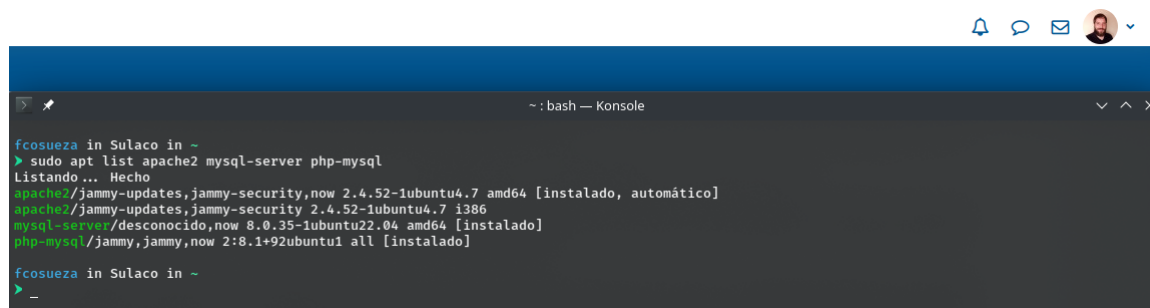
- Instalar el servidor web Apache desde terminal de comandos.
- Arrancar, reiniciar, comprobar el estado y parar el servidor web Apache.
- Comprobar que está funcionando el servidor Apache desde un navegador web.
- Cambiar el puerto por el cual está escuchando Apache pasándolo al puerto 8088 y comprueba de nuevo desde tu navegador que está funcionando.
- Cambiar la página web por defecto para que aparezcan tus apellidos mas un pantallazo de tu inicio del curso.

3.2. Solución

En este ejercicio se va a realizar la instalación de **Apache 2** como servidor web en un sistema **Linux Ubuntu 22**. En mi caso, uso **Kubuntu 22.04** como sistema operativo de escritorio, por lo que la instalación se realizará en este sin necesidad de máquina virtual, ya que solo difiere de Ubuntu en el gestor de ventanas por defecto, que es KDE en vez de Gnome.

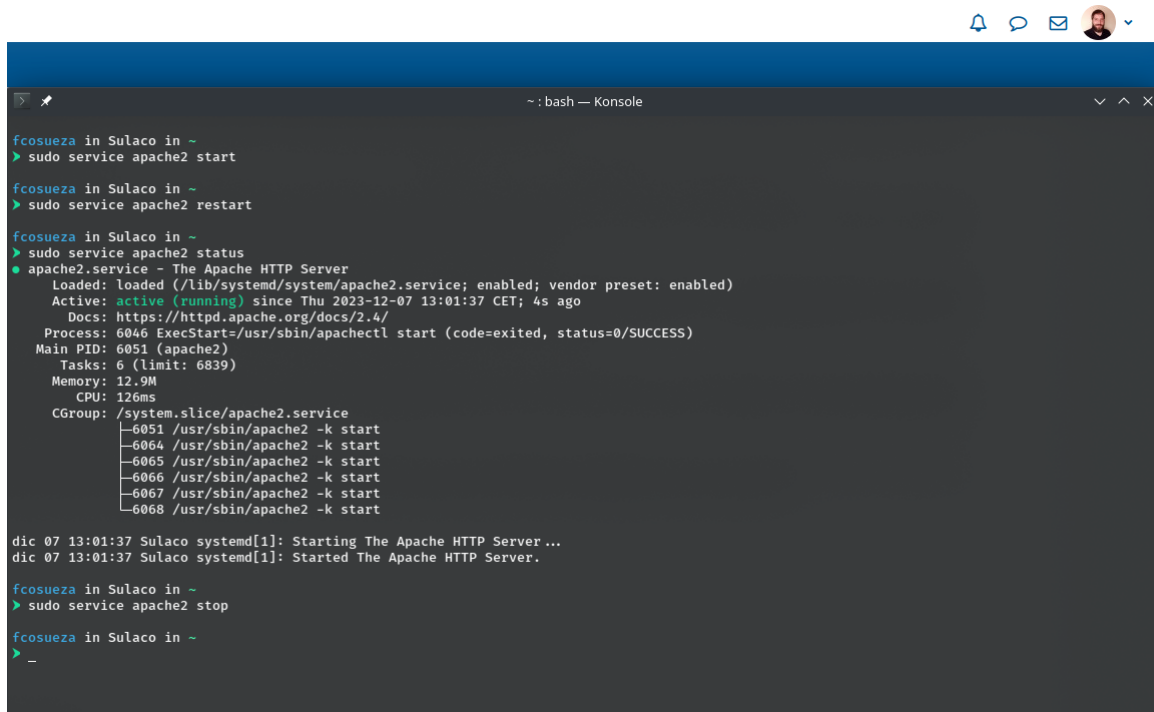
Además de la instalación, se realizarán algunas tareas básicas de administración sobre el servidor instalado. En la siguiente lista, podemos ver todos los pasos y comandos empleados:

- **Instalación de Apache:** en este caso, Apache2 ya esta instalado, ya que la instalación se realizó para la asignatura Desarrollo Web en el Entorno del Servidor. Para su instalación se utilizó el comando **APT**, en concreto ***sudo apt-get install apache2***, así que si ejecutamos dicho comando, no va a tener efecto. También están instalados todos los paquetes necesarios, por lo que se mostrará la salida del comando ***apt list***, en su lugar, donde se muestra la correcta instalación de los paquetes, como vemos en la siguiente captura:



```
~ : bash — Konsole
fcosueza in Sulaco in ~
> sudo apt list apache2 mysql-server php-mysql
Listando ... Hecho
apache2/jammy-updates,jammy-security,now 2.4.52-1ubuntu4.7 amd64 [instalado, automático]
apache2/jammy-updates,jammy-security 2.4.52-1ubuntu4.7 i386
mysql-server/desconocido,now 8.0.35-1ubuntu22.04 amd64 [instalado]
php-mysql/jammy,jammy,now 2:8.1+92ubuntu1 all [instalado]
fcosueza in Sulaco in ~
> _
```

- **Arranque, Parado y Estado de Apache:** para realizar estas labores de administración de ha usado el comando *service* de Kubuntu, en concreto, los comandos empleados han sido, por orden: *sudo service apache2 start*, *sudo service apache2 restart*, *sudo service apache2 status* y *sudo service apache2 stop*, como podemos ver en la siguiente captura de la terminal.



```

~ : bash — Konsole

fcosueza in Sulaco in ~
> sudo service apache2 start

fcosueza in Sulaco in ~
> sudo service apache2 restart

fcosueza in Sulaco in ~
> sudo service apache2 status
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2023-12-07 13:01:37 CET; 4s ago
     Docs: https://httpd.apache.org/docs/2.4/
   Process: 6046 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
  Main PID: 6051 (apache2)
    Tasks: 6 (limit: 6839)
   Memory: 12.9M
      CPU: 126ms
   CGroup: /system.slice/apache2.service
           └─6051 /usr/sbin/apache2 -k start
             6064 /usr/sbin/apache2 -k start
             6065 /usr/sbin/apache2 -k start
             6066 /usr/sbin/apache2 -k start
             6067 /usr/sbin/apache2 -k start
             6068 /usr/sbin/apache2 -k start

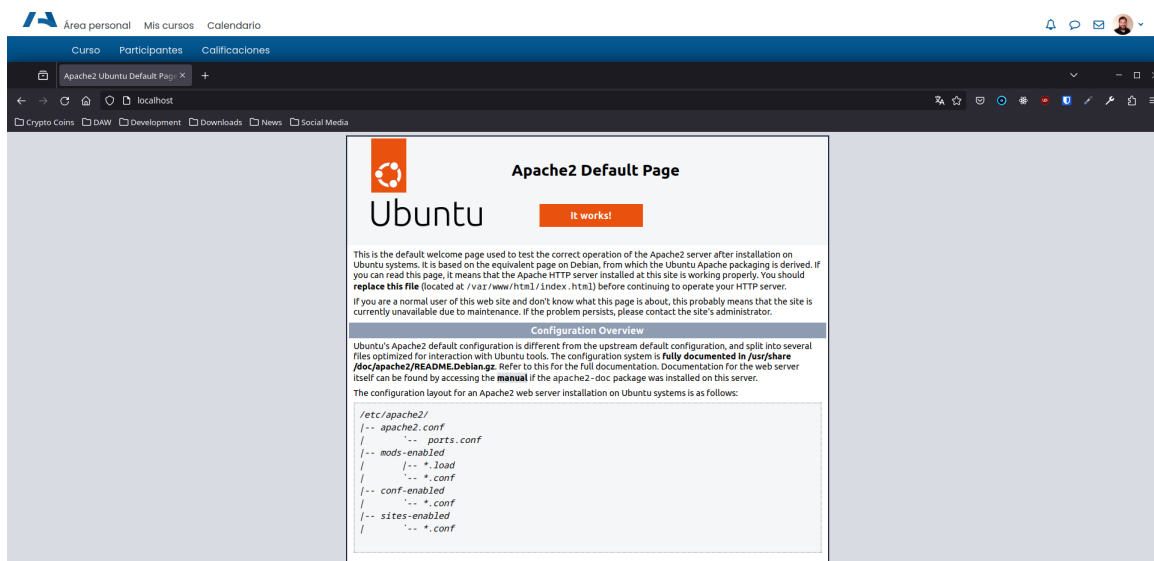
dic 07 13:01:37 Sulaco systemd[1]: Starting The Apache HTTP Server ...
dic 07 13:01:37 Sulaco systemd[1]: Started The Apache HTTP Server.

fcosueza in Sulaco in ~
> sudo service apache2 stop

fcosueza in Sulaco in ~
> _

```

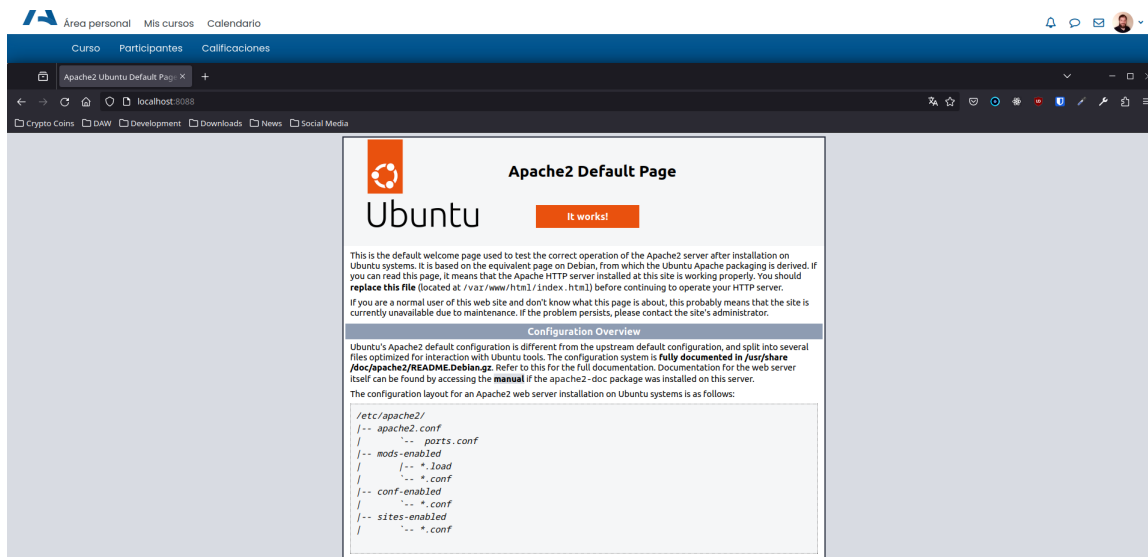
- **Comprobar Instalación:** una vez realizada la instalación y probados algunos comandos básicos, vamos a comprobar que el servidor funciona correctamente, conectándonos a nuestro localhost desde el navegador, que como podemos ver en la siguiente captura, muestra la página por defecto de Apache2.



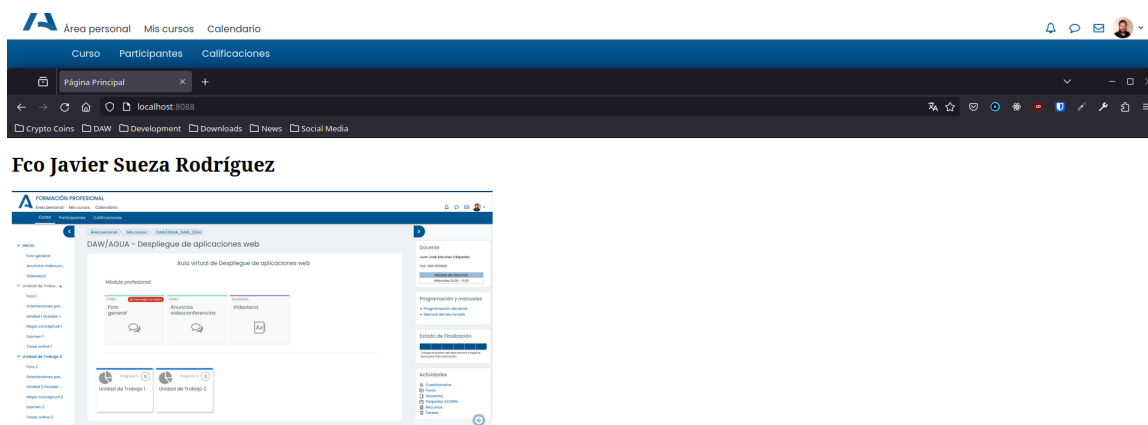
- **Cambiar Puerto por Defecto:** para cambiar el puerto por defecto se ha consultado el archivo */etc/apache2/apache2.conf*. En este fichero, se nos indica, mediante la directiva **Include ports.conf**,

y su comentario asociado, que es en ese fichero donde debemos modificar los puertos en los que estará escuchando el servidor.

Se ha editado ese fichero, cambiando la directiva **Listen 80** por **Listen 8088**. Tras reiniciar el servidor hemos comprobado su correcto funcionamiento conectando desde un navegador al localhost, pero cambiando el puerto de conexión 8080, como se puede ver en la siguiente captura.



- **Modificación Página Principal:** por último, se ha modificado la página por defecto de Apache, usando el editor VS Code, para que muestre mi nombre y apellido así como una captura del inicio de curso, como se puede ver en la siguiente imagen.



4. Ejercicio 4: Haciendo las Comunicaciones Más Seguras

4.1. Enunciado

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).

4.2. Solución

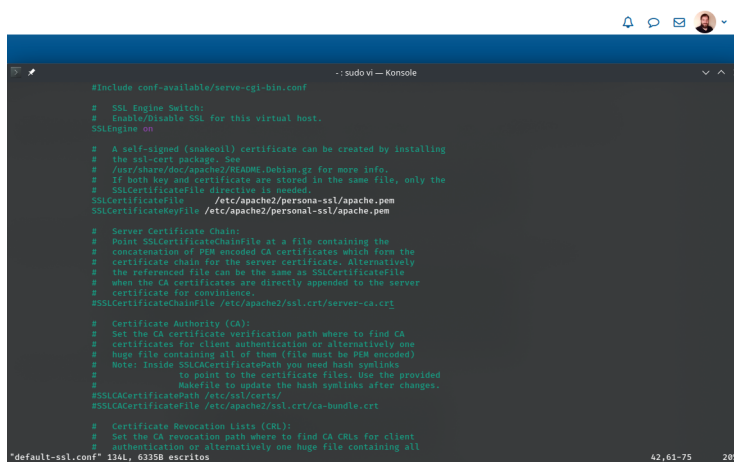
En este ejercicio vamos a instalar un certificado SSL en Apache, con lo siguientes pasos:

1. En primer lugar, se ha generado el certificado con **openssl**, que ya estaba instalado, y se ha almacenado en el directorio **/etc/apache2/personal-ssl**, que se ha creado mediante **mkdir**. A continuación se ha usado el comando **make-ssl-cert** para generar el certificado, mostrando el menú que podemos ver en la siguiente captura.

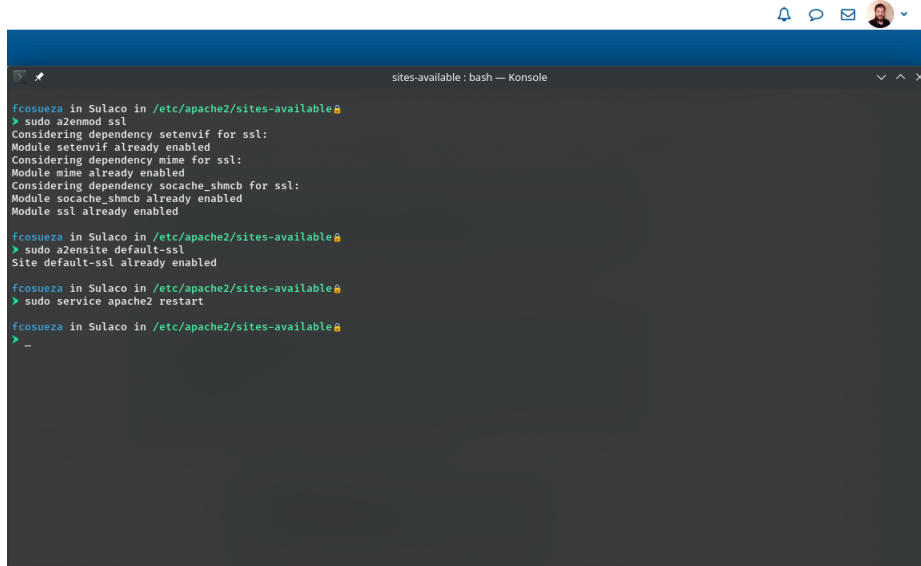


Tras completar los datos, se ha generado el archivo **apache.pem** en la carpeta anteriormente mencionada.

2. El siguiente paso es **cambiar la configuración** de Apache, para indicarle el nuevo directorio donde deberá buscar el certificado SSL. En la siguiente captura podemos ver la modificación que se ha realizado en el archivo **/etc/apache2/sites-available/default-ssl.conf**.



3. Tras comprobar que está habilitado el puerto 433 cuando el módulo SSL está cargado, se ha **cargado el módulo** ssl en Apache y se ha **añadido a available-sites** el fichero default-ssl, para posteriormente resetear el servidor web. En la siguiente captura, vemos la secuencia de comandos empleada.



```
sites-available : bash — Konsole

fcosueza in Sulaco in /etc/apache2/sites-available#
> sudo a2enmod ssl
Considering dependency setenvif for ssl:
Module setenvif already enabled
Considering dependency mime for ssl:
Module mime already enabled
Considering dependency socache_shmcb for ssl:
Module socache_shmcb already enabled
Module ssl already enabled

fcosueza in Sulaco in /etc/apache2/sites-available#
> sudo a2ensite default-ssl
Site default-ssl already enabled

fcosueza in Sulaco in /etc/apache2/sites-available#
> sudo service apache2 restart

fcosueza in Sulaco in /etc/apache2/sites-available#
> _
```

4. Por último se ha comprobado que funciona correctamente la conexión SSL en servidor. Para ello, nos hemos conectado al **localhost** usando la dirección **https://localhost**. No hemos empleado otra dirección porque cuando se configuró el certificado no se especificaron nombres alternativos, para poder conectarnos de esta forma sin necesidad de modificar más ficheros. En la siguiente captura se ve como nos conectamos mediante SSL al servidor.



NOTA: La página que se muestra es la que viene por defecto con Apache, ya que después del ejercicio anterior se han revertido los cambios en esta página para que vuelva a mostrar la que viene por defecto.

5. Ejercicio 5: Instalador de un Servidor de Aplicaciones

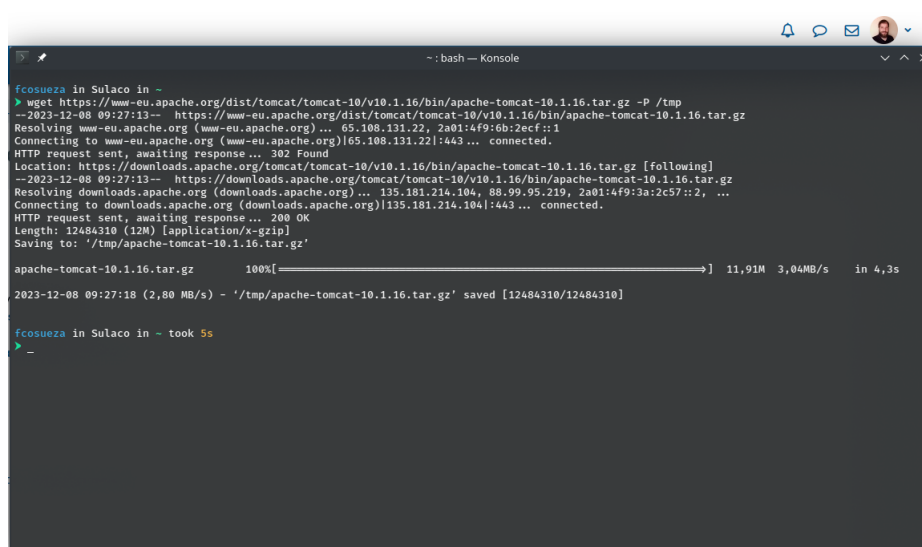
5.1. Enunciado

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.

5.2. Solución

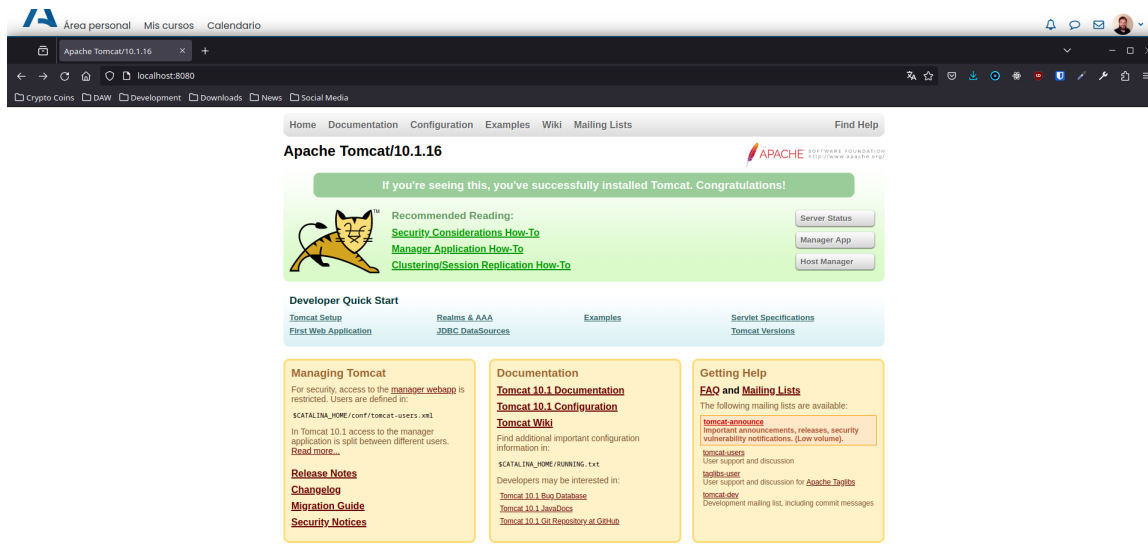
En este ejercicio vamos a realizar una instalación básica del servidor de aplicaciones Tomcat, probando a continuación que se ha instalado correctamente desde un navegador. Los pasos que se han seguido han sido los siguientes:

1. En primer lugar, hemos descargado la distribución binaria de Tomcat mediante el comando **wget**. En nuestro caso, se ha descargado la **versión 10.1.6**, como podemos ver en la siguiente captura.



```
fcosueza in Sulaco in ~  
➤ wget https://www-eu.apache.org/dist/tomcat/tomcat-10/v10.1.6/bin/apache-tomcat-10.1.6.tar.gz -P /tmp  
--2023-12-08 09:27:13-- https://www-eu.apache.org/dist/tomcat/tomcat-10/v10.1.6/bin/apache-tomcat-10.1.6.tar.gz  
Resolving www-eu.apache.org (www-eu.apache.org) ... 65.108.131.22, 2a01:4f9:6b:2ecf::1  
Connecting to www-eu.apache.org (www-eu.apache.org)[65.108.131.22]:443... connected.  
HTTP request sent, awaiting response... 302 Found  
Location: https://downloads.apache.org/tomcat/tomcat-10/v10.1.6/bin/apache-tomcat-10.1.6.tar.gz [following]  
--2023-12-08 09:27:13-- https://downloads.apache.org/tomcat/tomcat-10/v10.1.6/bin/apache-tomcat-10.1.6.tar.gz  
Resolving downloads.apache.org (downloads.apache.org) ... 135.181.214.104, 88.99.95.219, 2a01:4f9:3a:2c57::2, ...  
Connecting to downloads.apache.org (downloads.apache.org)[135.181.214.104]:443... connected.  
HTTP request sent, awaiting response... 200 OK  
length: 12484310 (12M) [application/x-gzip]  
Saving to: '/tmp/apache-tomcat-10.1.6.tar.gz'  
  
apache-tomcat-10.1.6.tar.gz 100%[====>] 11,91M 3,04MB/s in 4,3s  
2023-12-08 09:27:18 (2,80 MB/s) - '/tmp/apache-tomcat-10.1.6.tar.gz' saved [12484310/12484310]  
  
fcosueza in Sulaco in ~ took 5s  
➤
```

2. El siguiente paso ha sido descomprimir el archivo descargado en la carpeta **/opt**, que va a ser donde se almacene definitivamente la distribución binaria de Tomcat. Para ello hemos usado el comando **sudo tar -zxvf /tmp/apache-tomcat-10.1.6.tar.gz -C /opt/**.
3. Como vamos a realizar una instalación básica sin mucha configuración, ya solo nos queda ejecutar el script de inicio de Tomcat. En este caso, hemos ejecutado el script **startup.sh**, que se encuentra en la carpeta **bin/bin** del directorio de Tomcat.
4. Por último, nos hemos conectado al **localhost** usando el puerto **8080** para comprobar que se está ejecutando Tomcat correctamente, y como podemos ver en la siguiente captura, su ejecución e instalación se ha realizado correctamente.



6. Ejercicio 6: Hosts Virtuales

6.1. Enunciado

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.

6.2. Solución

Los **VirtualHosts** son un mecanismo de Apache que nos permite que un servidor albergue diferentes webs con diferentes dominios, manteniendo cada dominio y su estructura de directorios independiente de los demás. Los Virtualhost se pueden utilizar de varias formas diferentes, aunque las más comunes son:

- **VirtualHost por Nombre:** en este tipo de VirtualHosts tenemos varios dominios asociados a una misma IP. Cuando un cliente se conecta a dicha IP, se servirá una página u otra dependiendo del dominio que se especifica.
- **VirtualHost por IP:** en este caso, hay diferentes IPs asociadas a la misma máquina y estas se asocian con las diferentes páginas que alberga el servidor, por lo que se servirá una página u otra dependiendo de la dirección IP que especifique el cliente.

Para configurar los Virtualhost en Ubuntu, que es el sistema que estamos empleando se deben añadir un archivo a la carpeta **/etc/apache2/sites-available/**, donde se especifiquen las diferentes directivas del servidor. Habrá que añadir un archivo por cada Virtualhost que queramos, y una vez que estén bien definidos, deberemos copiarlos o crear un enlace simbólico en la carpeta **/etc/apache/sites-enabled**. A continuación, vamos a mostrar dos ejemplos de configuración de VirtualHosts.

El primer ejemplo es un **VirtualHost por nombre** que ya teníamos configurado para la asignatura DWES. En este caso, en este caso, además de las opciones habituales, se ha añadido una directiva para que no se puede acceder a la **carpeta nbproject** desde el servidor, ya que es irrelevante para el sitio web ya que contiene la información del proyecto de Netbeans.

```
sudo vi - Konsole

<VirtualHost *:80>
# The ServerName directive sets the request scheme, hostname and port that
# the server uses to identify itself. This is used when creating
# redirection URLs. In the context of virtual hosts, the ServerName
# specifies what hostname must appear in the request's Host: header to
# match this virtual host. For the default virtual host (this file) this
# value is not decisive as it is used as a last resort host regardless.
# However, you must set it for any further virtual host explicitly.
#ServerName www.example.com

ServerAdmin webmaster@localhost
DocumentRoot /var/www/php-projects
ServerName php-projects
ServerAlias www.php-projects

<Directory /var/www/php-projects>
    Options Indexes FollowSymLinks MultiViews
    AllowOverride All
    Require all granted
</Directory>

<Directory /var/www/php-projects/*>
    Allow from none
    Order allow,deny
</Directory>

# Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
# error, crit, alert, emerg.
# It is also possible to configure the loglevel for particular
# modules, e.g.:
#LogLevel info ssl:warn

ErrorLog ${APACHE_LOG_DIR}/error.log
CustomLog ${APACHE_LOG_DIR}/access.log combined

# For most configuration files from conf-available/, which are
# 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
</VirtualHost>

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
```

Ahora vamos a mostrar un ejemplo del fichero de configuración de un **Virtualhost por IP**. En este caso como no teníamos ninguno configurado, se ha creado desde cero y hemos simulado dos hosts virtuales a las paginas de las asignaturas de programación y desarrollo web en el entorno cliente. El fichero, quedaría como podemos ver a continuación.

```
vi - Konsole

26 <VirtualHost 192.168.0.140:80>
25     ServerAdmin webmaster@localhost
24     DocumentRoot /var/www/programacion
23     ServerName programacion
22     ServerAlias www.programacion
21
20     <Directory /var/www/programacion>
19         Options Indexes FollowSymLinks MultiViews
18         AllowOverride All
17         Require all granted
16     </Directory>
15
14     ErrorLog ${APACHE_LOG_DIR}/error.log
13     CustomLog ${APACHE_LOG_DIR}/access.log combined
12 </VirtualHost>
11
10 <VirtualHost 192.168.0.150:80>
9     ServerAdmin webmaster@localhost
8     DocumentRoot /var/www/dwes
7     ServerName desarrollocliente
6     ServerAlias www.desarrollocliente
5
4     <Directory /var/www/dwes>
3         Options Indexes FollowSymLinks MultiViews
2         AllowOverride All
1         Require all granted
0     </Directory>
1
2     ErrorLog ${APACHE_LOG_DIR}/error.log
3     CustomLog ${APACHE_LOG_DIR}/access.log combined
4 </VirtualHost>
5
6 # vim: syntax=apache ts=4 sw=4 sts=4 sr noet
```