

**Ejercicio 1.** Tipos de Aplicaciones Web. En función de cómo se presenta la aplicación web junto con el contenido que pretende mostrar, se ha establecido una clasificación. Indica cuál es dicha clasificación y sus características.

En función de cómo se presenta la aplicación web junto con el contenido que pretende mostrar, se ha establecido la siguiente clasificación:

**Página web Estática**. Están implementadas en HTML y pueden mostrar en alguna parte de la página objetos en movimiento tales como banners, GIF animados, vídeos, etc.

**Página web Animada.** Se realizan con la tecnología FLASH; ésta permite que una página web presente el contenido con ciertos efectos animados continuados. El uso de esta tecnología permite diseños más vanguardistas, modernos y creativos.

**Página web Dinámica.** Existen muchos lenguajes de programación que son la base para la mayoría de páginas web dinámicas. Los que destacamos aquí son los lenguajes PHP y ASP. Estos lenguajes permiten una perfecta estructuración del contenido. Por una parte crearemos la estructura de las páginas web y por otra, almacenaremos el contenido en determinados archivos. A partir de ahí, crearemos el código de llamada, que insertaría el contenido en la propia página web estructurada. Este es el principio básico que siguen los lenguajes de programación. A partir de aquí se desarrollan aplicaciones para poder gestionar el contenido a través de un panel de control.

**Portal.** Es un sitio web que en su página principal permite el acceso a múltiples secciones que, por lo general, son foros, chats, cuentas de correo, buscador, acceso registrado para obtener ciertas ventajas, las últimas noticias de actualidad, etc.

**Tienda virtual o comercio electrónico**. Sitio web que publica los productos de una tienda en Internet. Permite la compra on-line a través de tarjeta de crédito, domiciliación bancaria o transferencia bancaria en general. Ofrece al administrador un panel de gestión para poder subir los productos, actualizarlos, eliminarlos, etc.

**Página web con "Gestor de Contenidos".** Se trata de un sitio web cuyo contenido se actualiza a través de un panel de gestión por parte del administrador del sitio. Este panel de gestión suele ser muy intuitivo y fácil de usar. En aquellas páginas web que requieran una actualización constante, se suele incorporar este panel de gestión para que la web pueda

controlarse día a día por parte del cliente.

**Ejercicio 2.** Arquitectura Web. Se puede establecer que la arquitectura de un sitio web comprende los sistemas de organización y estructuración de los contenidos junto con los sistemas de recuperación de información y navegación que provea el sitio web, con el objetivo de servir de ayuda a los usuarios a encontrar y manejar la información. ¿Cuáles son los modelos de arquitectura web relacionados con la implementación de cada una de las capas establecidas en una aplicación web?

Los distintos modelos de aplicación sobre los que se ha ido desarrollando,

según diversos autores, se podrían clasificar del siguiente modo:

**Modelo 1**

En este caso las aplicaciones se diseñan en un modelo web CGI, basadas en la ejecución de procesos externos al servidor web, cuya salida por pantalla era el HTML que el navegador recibía en respuesta a su petición. Presentación, negocio y acceso a datos se confundían en un mismo script perl.

**Modelo 1.5**

Aplicado a la tecnología java, se da con la aparición de las JSP y los servlets. En este modelo, las responsabilidades de presentación recaen en las páginas JSP, mientras que los beans incrustados en las mismas son los responsables del modelo de negocio y acceso a datos.

**Modelo 2**

Como evolución del modelo anterior, con la incorporación del patrón MVC en este tipo de aplicaciones, se aprecia la incorporación de un elemento controlador de la navegación de la aplicación. El modelo de negocio queda encapsulado en los javabeans que se incrustan en las páginas JSP.

**Modelo 2X**

Aparecen con el objetivo de dar respuesta a la necesidad, cada vez más habitual, de desarrollar aplicaciones multicanal, es decir, aplicaciones web que pueden ser atacadas desde distintos tipos de clientes remotos. Así, una aplicación web multicanal podrá ejecutarse desde una PDA, desde un terminal de telefonía móvil, o desde cualquier navegador HTML estándar. El medio para lograr publicar la misma aplicación para distintos dispositivos es emplear plantillas XSL para transformar los datos XML.

**Ejercicio 3.** Escalabilidad y Balanceador de Carga. La escalabilidad de un sistema web puede ser verticalmente, horizontalmente y clúster. Indica las características principales de cada uno e incluye los balanceadores de carga de la escalabilidad horizontal.

La escalabilidad de un sistema web puede ser:

**Verticalmente:** de manera ascendente "upgrades" a cada nodo.

**Horizontalmente:** consiste en aumentar el número de nodos.

**Cluster:** consiste en crear agrupaciones de servidores.

**Escalabilidad vertical.**

Habitualmente, la separación lógica en capas se implementa de tal forma que se permita una separación física de las mismas.

Interponiendo elementos conectores que actúen de middlewares es posible distribuir la aplicación de forma vertical (una máquina por cada capa del sistema), e incluso si esto no fuera suficiente, distribuyendo los elementos de una misma capa entre distintas máquinas servidoras.

**Escalabilidad horizontal.**

Se trata de clonar el sistema en otra máquina de características similares y balancear la carga de trabajo mediante un dispositivo externo. El balanceador de carga puede ser:

**- Balanceador Software:** Por ejemplo, habitualmente encontramos un servidor web apache junto con el módulo mod\_jk, que permite la redirección de las peticiones http que a tal efecto sean configuradas entre las distintas máquinas que forman la granja de servidores. Este tipo de balanceadores examinan el paquete http e identifican la sesión del usuario, guardando registro de cuál de las máquinas de la granja se está encargando de servir a dicha sesión. Este aspecto es importante, dado que nos permite trabajar (de cara al diseño de la aplicación) apoyándonos en el objeto sesión propio del usuario y almacenando información relativa a la sesión del mismo, puesto que tenemos la garantía de que todas las

peticiones de una misma sesión http van a ser redireccionadas hacia la misma máquina.

**- Balanceador hardware:** Se trata de dispositivos que, respondiendo únicamente a

algoritmos de reparto de carga (Round Robin, LRU, etc.), redireccionan una petición http del

usuario a la máquina que, según dicho algoritmo, convenga que se haga cargo de la petición.

Son mucho más rápidos que los anteriores, dado que se basan en conmutación de circuitos

y no examinan ni interpretan el paquete http. Sin embargo, el no garantizar el mantenimiento de la misma sesión de usuario en la misma máquina, condiciona seriamente el diseño, dado que fuerza a que la información relativa a la sesión del usuario sea almacenada por el implementador del mismo, bien en cookies o bien en base de datos.

**- Balanceador hardware http:** Se trata de dispositivos hardware pero que examinan el

paquete http y mantienen la relación usuario-máquina servidora. Mucho más rápidos que los

balanceadores software, pero algo menos que los hardware, suponen hoy en día una de las

soluciones más aceptadas en el mercado.

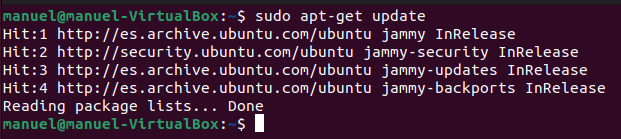
**Cluster**

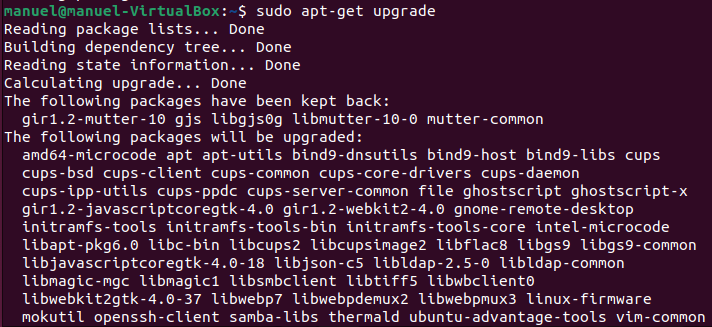
Con la aparición de los servidores de aplicaciones en cluster se abre una nueva capacidad de escalabilidad que, dependiendo de cómo se aplique, podría clasificarse como vertical u horizontal. Un cluster de servidores de aplicaciones permite el despliegue de una aplicación web corriente, de forma que su carga de trabajo vaya a ser distribuida entre la granja de servidores que forman el cluster, de modo transparente al usuario y al administrador. El cluster, mediante el mecanismo de replicación de sesión, garantiza que sea cual sea la máquina que sirva la petición http, tendrá acceso a la sesión del usuario (objeto HttpSession en java). Este tipo de sistemas, debido precisamente a la replicación de sesión, suele presentar problemas de rendimiento.

**Ejercicio 4.** Accediendo a la consola de administración. Dispones de una máquina (o máquina virtual) que cuenta con el sistema operativo Ubuntu 20.04 LTS recientemente actualizado, esta máquina tiene el entorno de red configurado y, además, dispones de conexión a Internet. Además, estás trabajando con la cuenta del usuario root. Indica cada uno de los pasos, y comandos implicados en ellos, (añade todas las capturas de pantalla necesarias que demuestren que todo ha funcionado correctamente, en todas ellas se debe ver al fondo la plataforma con vuestra foto del perfil, estos requisitos son imprescindibles para valorar la tarea) para conseguir hacer lo siguiente:

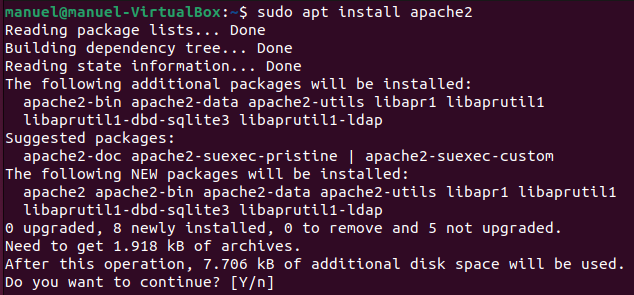
* **Instalar el servidor web Apache desde terminal.**

Para instalar el servidor web Apache, nuestro primer paso es actualizar los paquetes de Ubuntu, haciendo **apt-get update** y **apt-get upgrade**.



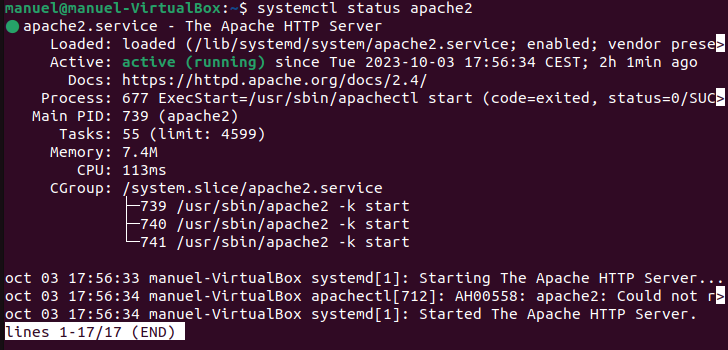


Una vez que hemos actualizado, vamos a instalar apache2.

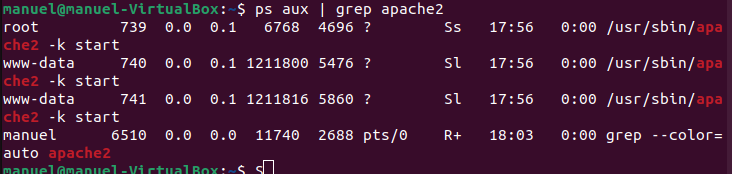


* **Comprobar que está funcionando y bien configurado el servidor Apache desde un terminal o consola de comandos. Si conoces varios comandos o formas de comprobarlo desde el terminal indícalo.**

El primer comando que podemos utilizar para comprobar que apache2 está funcionando es **systemctl status apache2**

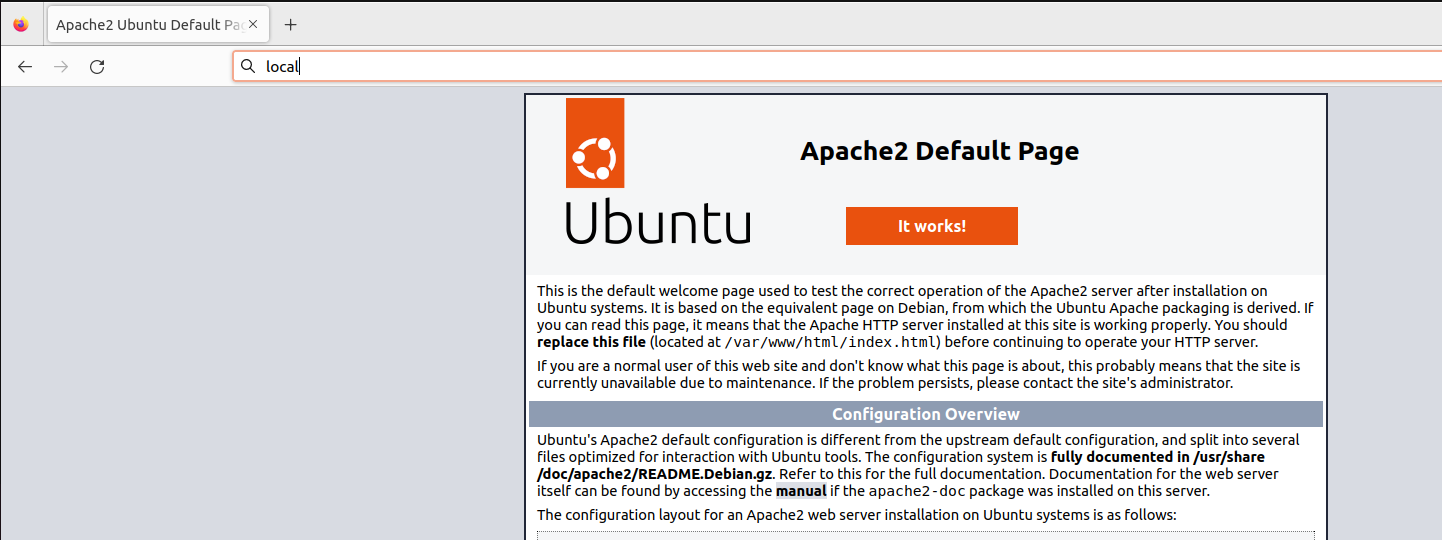


También podemos ver mediante **ps** los procesos en ejecución, filtrando por apache2.



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

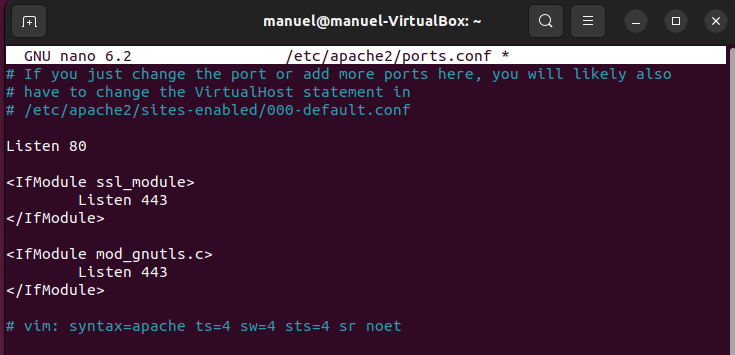
Para comprobar que funciona Apache2 desde un navegador web, simplemente accedemos a **<http://localhost.>**



* **Cambiar el puerto por el cual está escuchando Apache pasándolo al puerto 81 y comprueba de nuevo que está funcionando.**

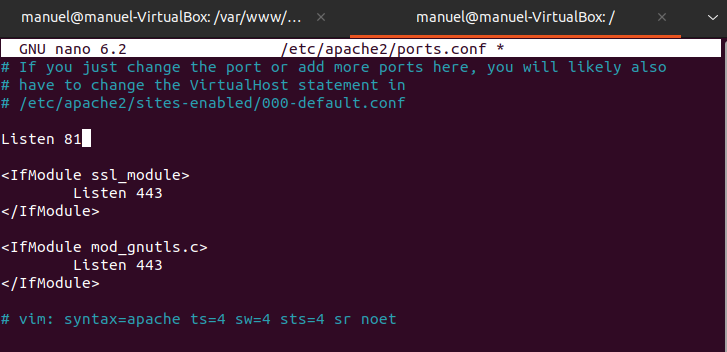
Para cambiar el puerto por el que escucha Apache2, vamos a modificar el fichero de configuración ubicado en **/etc/apache2/ports.conf**.

Al abrir este fichero, por defecto está establecido el puerto 80. Vamos a cambiarlo al puerto 81 como nos pide la actividad.

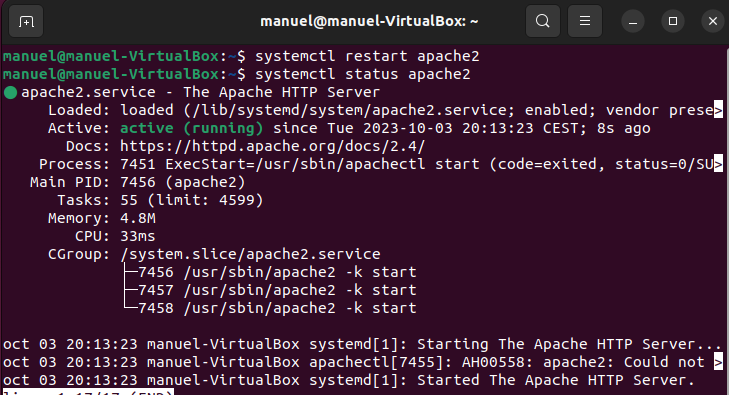


“Listen 80” hace referencia al puerto por el cual escucha apache2 al protocolo http, y “Listen 443” al puerto por el que escucha al protocolo https.

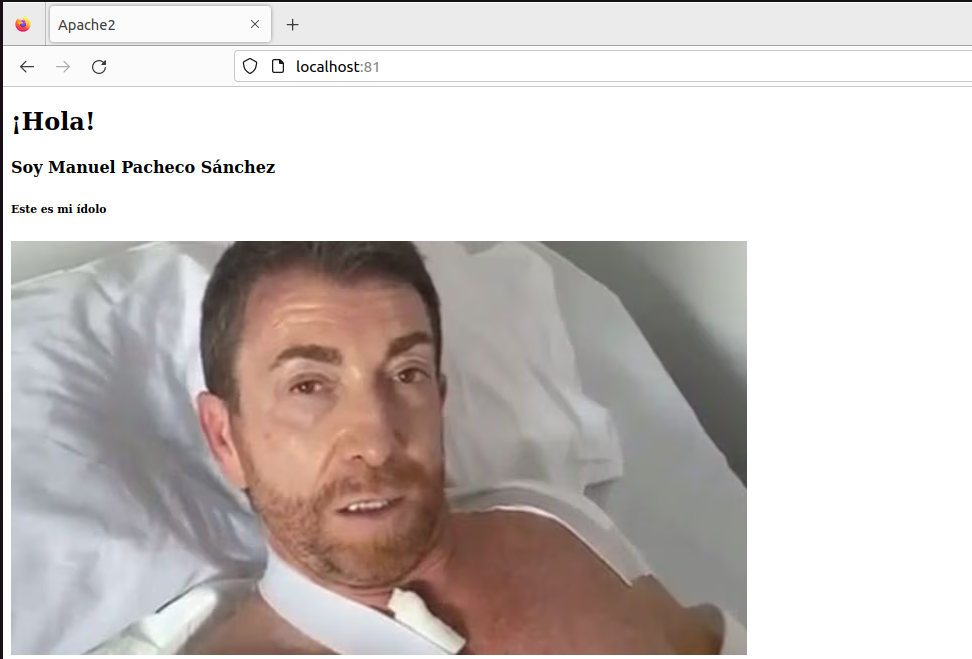
Vamos a modificar el puerto por el que escucha a http por el 81.



Ahora, vamos a reiniciar el servicio con el comando **systemctl restart apache2**, y vamos a comprobar que ahora escucha al puerto 81.



Vamos a abrir el navegador, y vamos a ir a la dirección [http://localhost:81.](http://localhost:8080.)

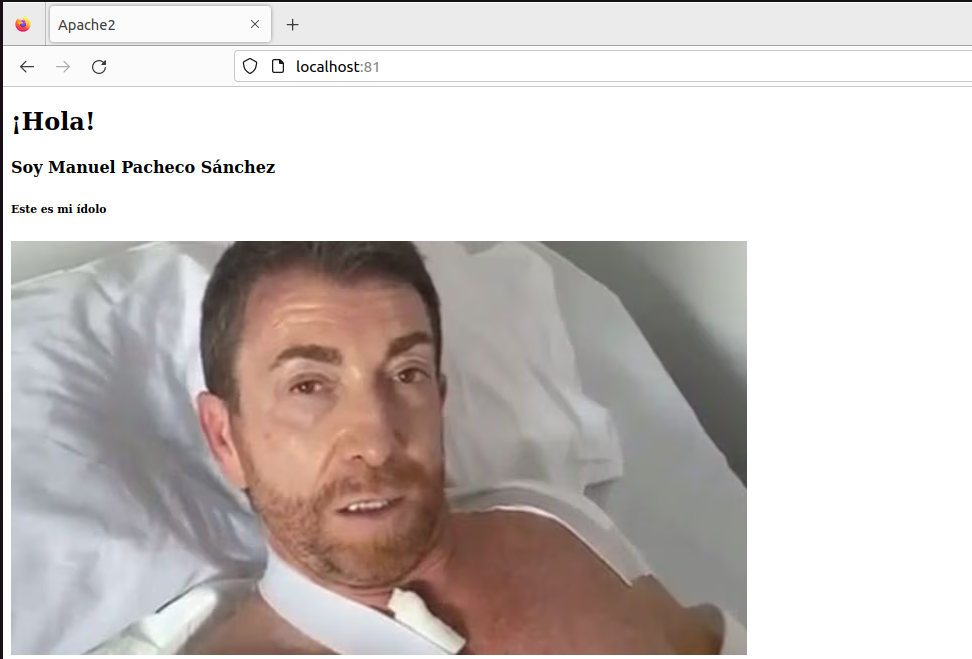


(La web ya está modificada porque me equivoqué al poner el puerto y continué con la siguiente actividad).

* **Cambiar la página web por defecto para que aparezcan tus apellidos más una imagen personalizada.**

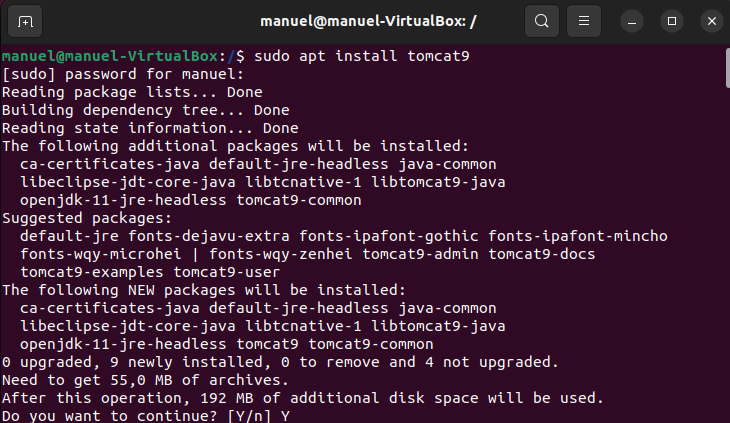
El html por defecto que muestra apache2 se ubica en el directorio **/var/www/html** y su nombre es index.html.

Vamos a modificar este fichero html para que cumpla con los requisitos.

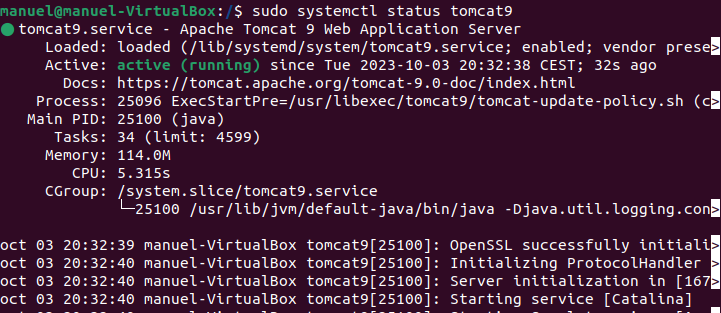


* **Instalar el servidor de aplicaciones Tomcat (en su última versión disponible) y comprobar que funciona.**

Para instalar tomcat, vamos a ejecutar el comando **apt install tomcat9**.



Una vez instalado, vamos a ver que el servicio funciona correctamente ejecutando el comando **systemctl status tomcat9**.



Para comprobar que funciona, accedemos a un navegador web y vamos a nuestro localhost, al puerto 8080.

