Montando una web en apache2 Ubuntu

# 

[1. Arquitecturas web](#_y0h9mi2q3qmq)

[1.1 Modelo Cliente-Servidor (Client-Server):](#_ipylbb8tbze1)

[1.2 Arquitectura de Tres Capas (Three-Tier Architecture):](#_1iwwf3hz47ww)

[1.2 Arquitectura de Microservicios (Microservices Architecture):](#_3dwbxefz59so)

[1.3 Arquitectura de Servicios Web (Service-Oriented Architecture - SOA):](#_6rd4ut9gq8fa)

[1.4 Arquitectura de Eventos (Event-Driven Architecture):](#_3vhvja32sgib)

[2. Instalación y configuración del servidor web apache - Ubuntu](#_zba6sl2xa64y)

[2.1 Instalación del servidor apache2](#_8ft2yuu1j98y)

[2.2 Un vistazo a los archivos de configuración](#_m17nn1t4446)

[2.3 Configurar el puerto en el archivo ports.conf](#_py5dn2fn8aco)

[2.4 Configurar un virtual host para el puerto 8080](#_kakn6zo1paj5)

[2.5 Mover la carpeta de nuestra web](#_xuvpga2o7lk7)

[2.6 Comprobación del sitio](#_2p6vqit7crfj)

[3. Códigos de estado HTTP de error de la web de prueba](#_mijcagh9z0sc)

# 

# **Arquitecturas web**

Estudiamos las arquitecturas web para ver cual es la más conveniente para nuestro proyecto.

## 1.1 Modelo Cliente-Servidor (Client-Server):

**Descripción**: Es una de las más tradicionales y básicas. Consiste en dos componentes principales: el cliente y el servidor. El cliente solicita servicios o recursos al servidor, que a su vez proporciona los servicios o recursos solicitados.

**Características**: Separación clara de responsabilidades entre el cliente y el servidor, lo que facilita el mantenimiento y la escalabilidad. La comunicación se realiza a través de solicitudes y respuestas.

**Ventajas:**

* Simple y fácil de entender.
* Separación clara de responsabilidades.
* Escalabilidad al poder mejorar el servidor sin afectar a los clientes.

**Desventajas:**

* Puede haber cuellos de botella si hay muchos clientes solicitando al mismo tiempo.
* La dependencia del servidor puede generar problemas si este falla.

## 1.2 Arquitectura de Tres Capas (Three-Tier Architecture):

**Descripción**: Divide la aplicación en tres capas: la capa de presentación (cliente), la capa lógica de negocios y la capa de almacenamiento de datos (servidor de base de datos). Cada capa tiene funciones y responsabilidades específicas.

**Características clave**: Proporciona una mayor modularidad y flexibilidad. Permite cambios en una capa sin afectar las otras. Facilita la gestión y el mantenimiento de la aplicación.

**Ventajas**:

* Modularidad y flexibilidad.
* Cambios en una capa no afectan a las demás.
* Facilita la gestión y el mantenimiento.

**Desventajas**:

* Puede haber cierta complejidad adicional en la comunicación entre capas.
* Puede requerir más recursos en comparación con arquitecturas más simples.

## 1.2 Arquitectura de Microservicios (Microservices Architecture):

**Descripción**: La aplicación se desarrolla como un conjunto de servicios independientes y pequeños, cada uno ejecutándose en su propio proceso y comunicándose entre sí a través de API. Cada servicio realiza una función específica y puede ser desarrollado, implementado y escalado de forma independiente.

**Características**: Escalabilidad y mantenimiento simplificado. Permite la implementación continua y facilita la adopción de tecnologías específicas para cada servicio.

**Ventajas:**

* Escalabilidad y mantenimiento simplificado.
* Facilita la adopción de tecnologías específicas para cada servicio.
* Independencia en el desarrollo y despliegue de cada microservicio.

**Desventajas:**

* Mayor complejidad en la gestión y monitorización.
* La comunicación entre microservicios puede ser un desafío.

## 1.3 Arquitectura de Servicios Web (Service-Oriented Architecture - SOA):

**Descripción**: Los servicios son componentes de software que se pueden reutilizar para diferentes propósitos dentro de la misma aplicación o en diferentes aplicaciones. La comunicación entre servicios se realiza a través de estándares como SOAP (Simple Object Access Protocol) o REST (Representational State Transfer).

**Características**: Favorece la reutilización de servicios, promueve la interoperabilidad y la escalabilidad. Sin embargo, puede resultar complejo de gestionar y mantener en entornos grandes.

**Ventajas:**

* Reutilización de servicios para diferentes propósitos.
* Promueve la interoperabilidad y la escalabilidad.
* Flexibilidad en la elección de tecnologías.

**Desventajas:**

* Puede ser complejo de gestionar en entornos grandes.
* La definición clara de los servicios es crucial para evitar problemas.

## 1.4 Arquitectura de Eventos (Event-Driven Architecture):

**Descripción**: Los componentes de un sistema se comunican a través de eventos. Los eventos pueden ser generados por cambios de estado o por solicitudes específicas. Los componentes pueden ser productores de eventos, consumidores de eventos o ambos.

**Características**: Desacopla los componentes, permitiendo una mayor flexibilidad y escalabilidad. Facilita la implementación de sistemas distribuidos. Sin embargo, puede ser difícil de depurar y mantener en entornos complejos.

Cada una de estas arquitecturas tiene sus propias ventajas y desventajas, y la elección entre ellas depende de los requisitos específicos del proyecto, las consideraciones de rendimiento, escalabilidad y mantenimiento, entre otros factores.

**Ventajas:**

* Desacopla componentes, permitiendo flexibilidad y escalabilidad.
* Facilita la implementación de sistemas distribuidos.
* Respuestas rápidas a eventos específicos.

**Desventajas:**

* Puede ser difícil de depurar y mantener en entornos complejos.
* La gestión de eventos puede volverse complicada a medida que crece el sistema.

**La arquitectura más apropiada para un e-commerce sería una arquitectura de microservicios ya que tenemos escalabilidad y flexibilidad así como un mantenimiento simple, pudiendo también usar una tecnología para cada microservicio haciendo cada servicio independiente, pero para la práctica utilizaremos una arquitectura modelo cliente-servidor en local para hacer las pruebas**

# Instalación y configuración del servidor web apache - Ubuntu

## 2.1 Instalación del servidor apache2

realizamos un update con **sudo apt-get update**

instalamos apache2 con **sudo apt-get install apache2**

para verificar que se ha instalado correctamente podemos ver la versión instalada con **apache2 -v**

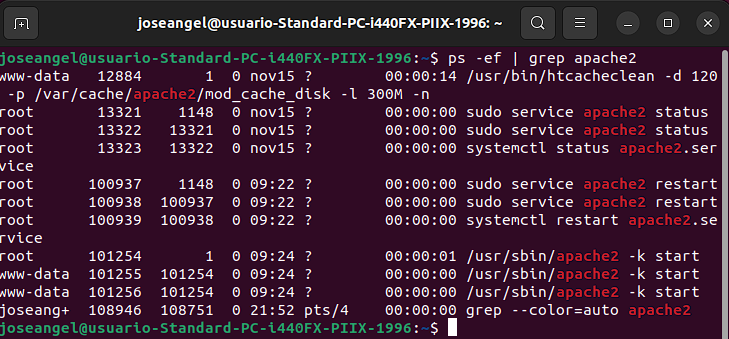
lanzamos apache2 con el comando **sudo service apache2 start**

una vez lanzado el servicio comprobamos el proceso de apache activo **ps -ef | grep apache**

podemos gestionar apache con los siguiente comandos

**sudo service apache2 {start | stop | restart | reload | status}**

**vemos que el proceso usr/sbin/apache2 pertenece al usuario www-data y al grupo www-data**



## 

## 2.2 Un vistazo a los archivos de configuración

cuando se instala apache2 tenemos 3 archivos de configuración esenciales, en la ruta **/etc/apache2**

**/etc/apache2/apache2.conf** Archivo de configuración global

**/etc/apache2/ports.conf** Archivo de configuración de puertos

**/etc/apache2/envvars** Archivo de configuración de variables de entorno

El fichero de configuración de puertos **/etc/apache2/ports.conf** se usa establecer puertos de escucha a cada módulo o a todos los módulos

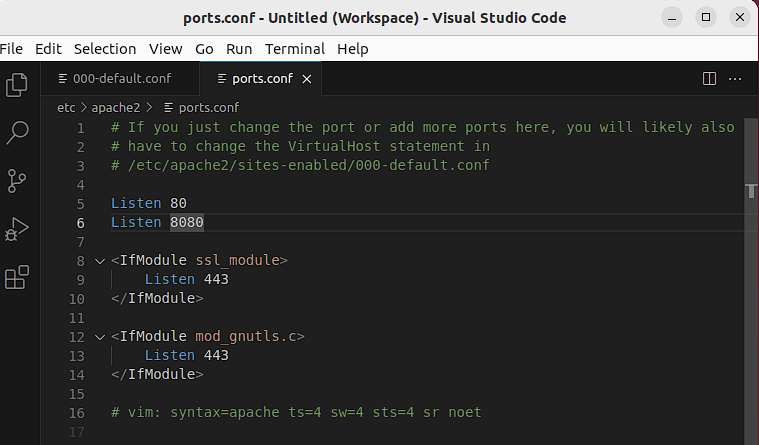
El fichero de configuración de variables de entorno **etc/apache2/envvars** contiene variables que almacenan directorios que se podrán usar en cualquier otro archivo de configuración

**Usaremos la siguiente configuración:**

* Puerto en el que escuchará: **8080**
* La página principal se llamará **landing.html**
* Cuando una página no se encuentre deberá de mostrar: **“No se ha encontrado la página en nuestro servidor”**

## 2.3 Configurar el puerto en el archivo ports.conf

Para configurar el puerto primero vamos al archivo de configuración de puertos en  **/etc/apache2/ports.conf** y añadimos el puerto 8080 con un **Listen 8080**

****

## 2.4 Configurar un virtual host para el puerto 8080

Luego tendremos que ir al archivo de configuración del sitio por defecto que se encuentra en **etc/apache2/sites-enabled/000-default.conf**

En este archivo tendremos que añadir otro **virtual-host para el puerto 8080**

En este archivo vemos **diferentes sentencias** dentro de la etiqueta virtual host:

**La sentencia ServerAdmin** webmaster@localhost

**La sentencia DocumentRoot** determina cuál será la ruta donde se encuentra la web por defecto al entrar a nuestro servidor

**En nuestro caso, tendremos la carpeta tienda-online y dentro el archivo landing.html por lo que tenemos que añadir el directorio tienda-online quedándonos la sentencia**

**DocumentRoot /var/www/html/tienda-online**

**La sentencia** **DirectoryIndex** determina cuál será el archivo index por defecto

**En nuestro caso tiene que ser landing.html, añadimos la sentencia**

**DirectoryIndex landing.html**

Podemos hacer que cargue **varios archivos** **DirectoryIndex landing.html index.html**

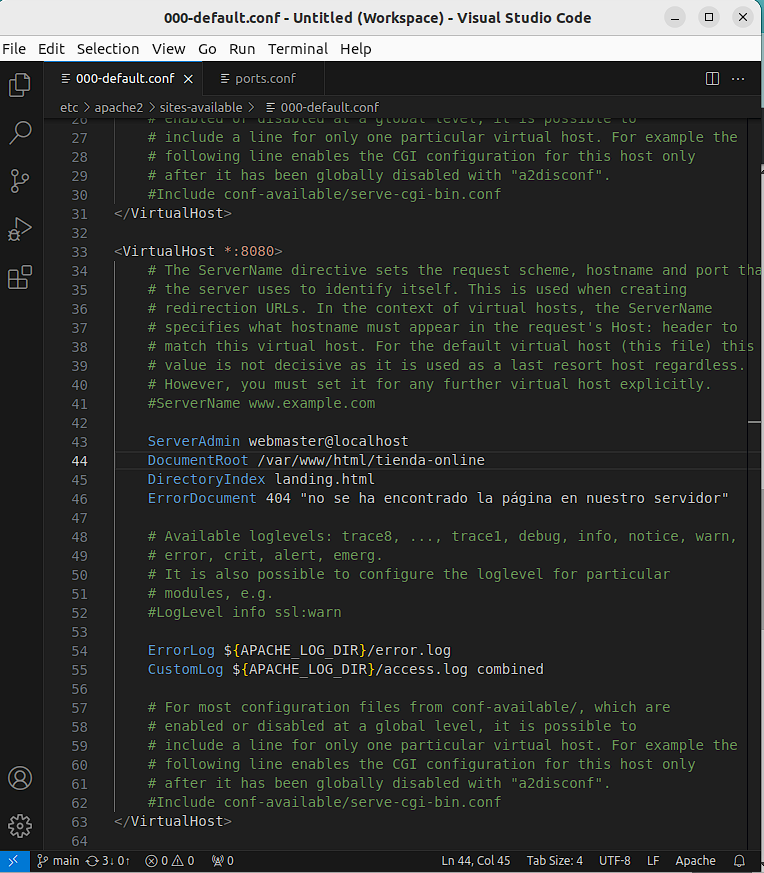
Intentará abrir landing.html si existe el archivo, si no encuentra ese archivo abrirá index.html

**La sentencia ErrorDocument 404** determinará “una línea de informe” para que cuando apache lanza un error de estado 404 muestre “una línea de informe” entonces si ahora accedemos a localhost/”unrecursoquenoexiste”:8080 tenemos que ver el mensaje especificado

**En nuestro caso el mensaje de error tiene que ser “no se ha encontrado la página en nuestro servidor”, por lo tanto añadimos la sentencia ErrorDocument 404 “no se ha encontrado la página en nuestro servidor”**

**La sentencia ErrorLog${APACHE\_LOG\_DIR}/error.log** usa una variable de entorno definida en **/etc/apache2/envvars** que apunta al directorio donde se guardará el archivo log y aqui definimos como se llamará ese archivo

Nuestro archivo **000-default.conf** tiene que quedar de la siguiente forma:



## 2.5 Mover la carpeta de nuestra web

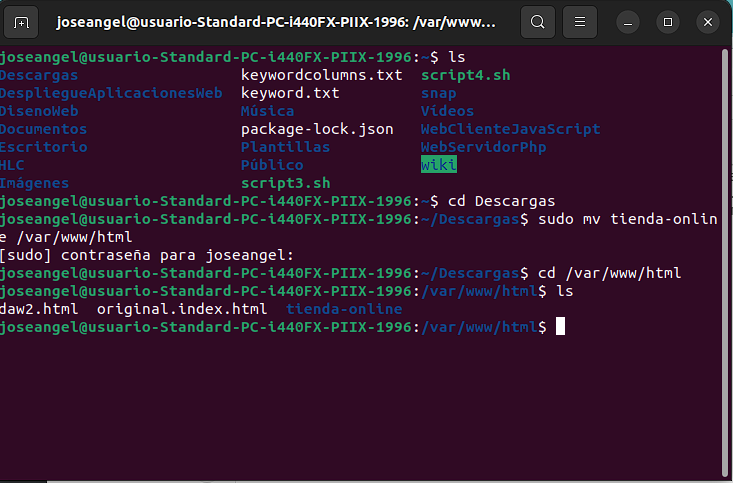
Por último tenemos que añadir los archivos de la web en nuestra carpeta **/var/www/html ,** descargamos el archivo tienda-online.zip y lo añadimos al directorio

Para ello utilizamos el comando mv como sudo para mover de la carpeta descargas a la carpeta **/var/www/html** la carpeta **tienda-online**

**desde la carpeta Descargas** usamos el comando

**sudo mv tienda-online /var/www/html**

vamos al directorio **/var/www/html**  y comprobamos que hemos movido correctamente la carpeta tienda-online

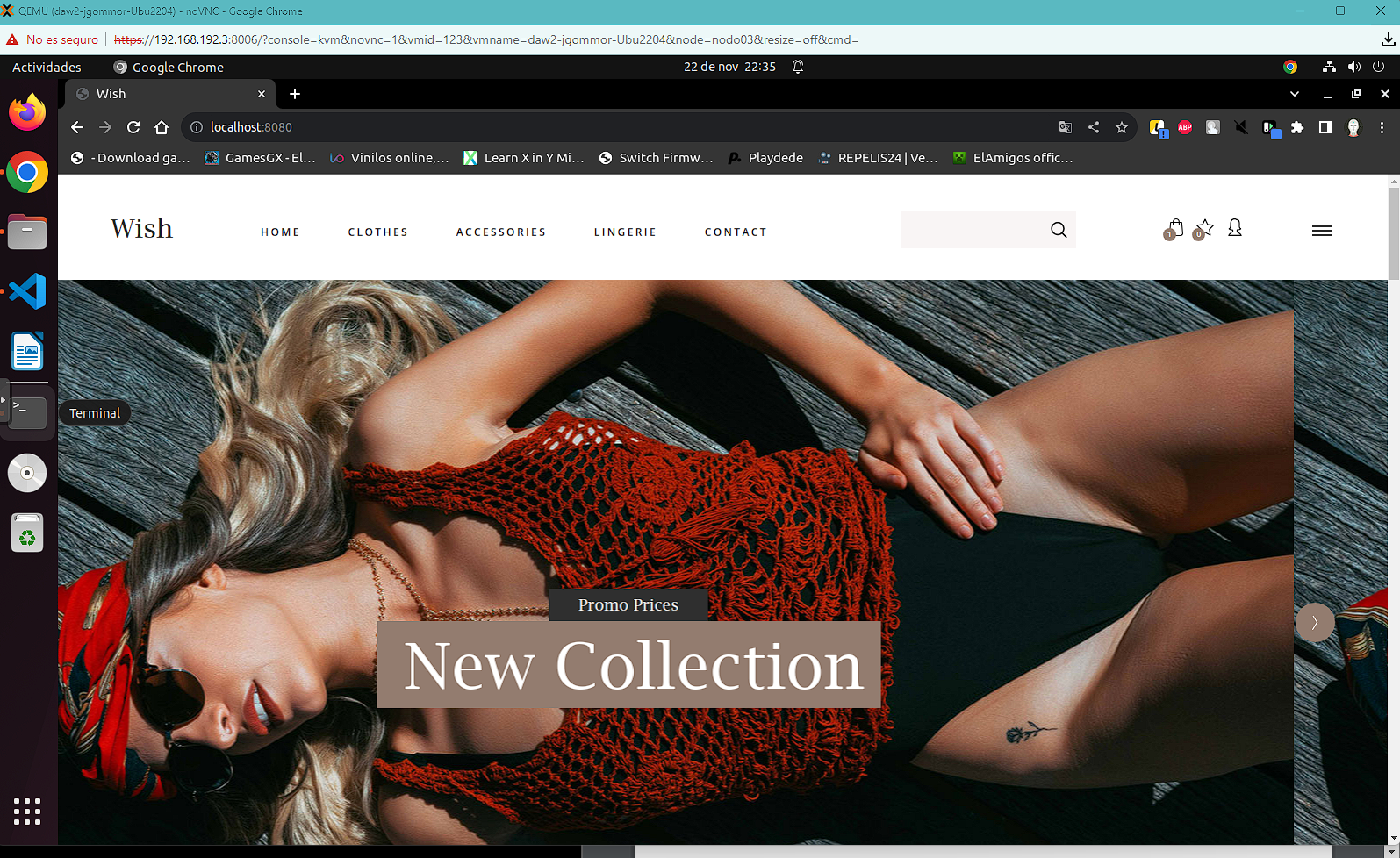


Reiniciamos el servicio apache2 para que los cambios en los archivos de configuración tengan efecto con sudo service apache2 restart

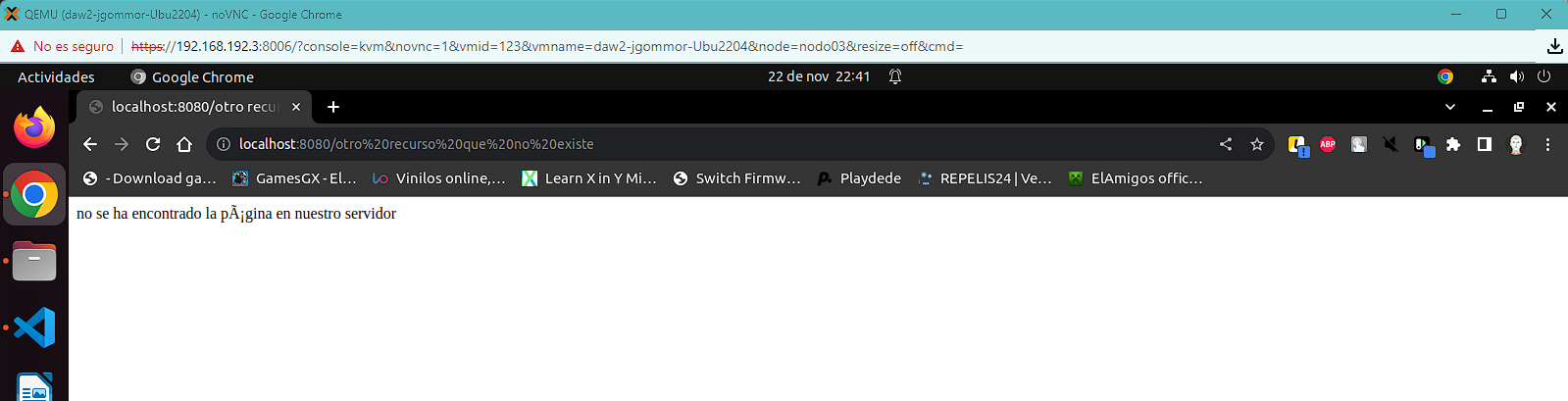
## 2.6 Comprobación del sitio

Si todos los ficheros de configuración están bien , debe de reiniciar y podemos empezar con las pruebas

Si accedemos desde nuestro navegador web con la dirección **localhost:8080** tiene que cargar el archivo **landing.html** y tenemos que ver la web de prueba

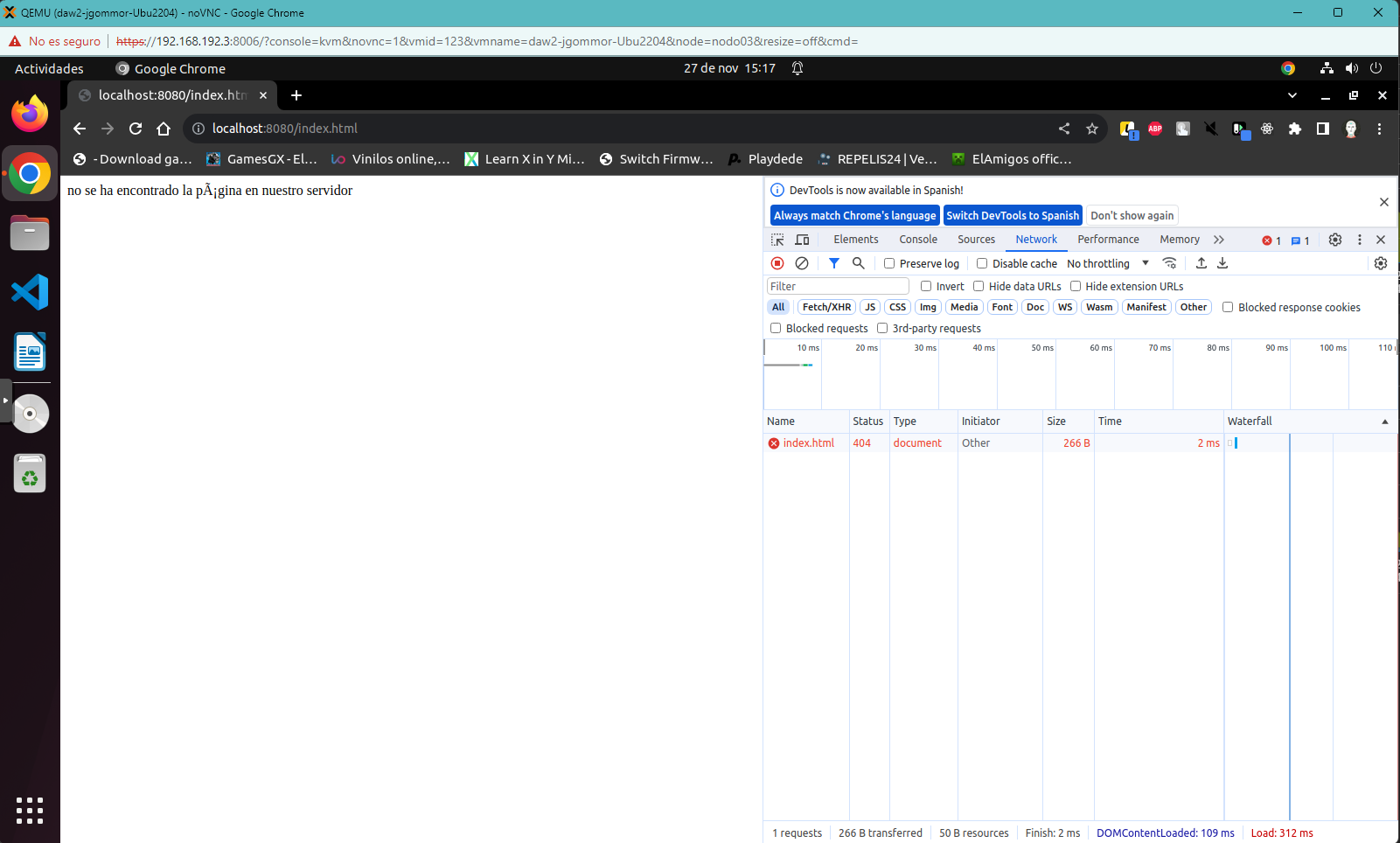


Para comprobar que la sentencia Error 404 “no se ha encontrado la página en nuestro servidor” está funcionando, podemos acceder a **localhost:8080/”otro recurso que no exista”** y debemos ver el mensaje



# Códigos de estado HTTP de error de la web de prueba

**Error 404:** al pulsar el botón home del menú desplegable o el botón home del menú principal en la cabecera , redirige a index.html y el recurso no existe, se puede solucionar, cambiando el redireccionamiento del botón home a landing.html



Todos los demás botones, tienen un redireccionamiento a la misma web o a webs externas como google maps o la web de la template por lo que todos los botones nos devuelve un estado 200.