1. El concepto de web y su evolución.

1.1. El concepto de web.

World Wide Web Consortium (W3C): Comunidad internacional que desarrolla los estándares que aseguran el crecimiento de la Web a largo plazo. En su página encuentras estándares para el diseño de aplicaciones o web de los servicios o de los dispositivos.

Web: Subconjunto de Internet.

En Internet existen multitud de servicios, como FTP (File Transfer Protocol).

Web convencional (Surface Web): Web visible para todos los usuarios.

Para poder navegar, se usan los "browser" o "navegadores", siendo Google Chrome el más usado por sus características.

Se confunden los términos ventana y pestaña. Las pestañas permiten cambiar rápidamente de tarea y abrir diferentes páginas web en una misma ventana.

Todos los navegadores permiten su personalización con las extensiones, temas o aplicaciones. En Chrome para instalarlos hay que hacerlo desde la Chrome Web Store.

Internet no es lo mismo que la World Wide Web.

1.2. Evolución de la web.

La Web tiene versiones, empezando por la 1.0 hasta la 4.0:

Web 1.0: Su finalidad era divulgativa y sus páginas estáticas.

Web 2.0: Desarrollada en 2004, esta es llamada la web social, ya que el usuario tiene un papel relevante en la misma por crear y compartir informacion. Este también interactúa con las páginas. Aparecen las redes sociales, wikis, blogs, etc.

Las páginas web 2.0 tienen los siguientes requisitos:

- ***Aplicaciones ricas en Internet (RIA):*** Son aplicaciones que ofrecen prestaciones como las apps de escritorio.
- ***Arquitectura orientada al servicio (SOA):*** Aquí los componentes software se diseñan para usarse como servicio en la red.
- ***Web social:*** El usuario es el centro de las aplicaciones Web 2.0

Web 3.0: También conocida como Web semántica, tiene como objetivo principal acercar al usuario lo más posible al lenguaje neutral y adaptar la navegación a sus preferencias. En esta web han surgido nuevas formas de búsqueda y almacenamiento, también nuevos lenguajes e inteligencia artificial. Esta es una web que analiza factores como preferencias y hábitos de navegación. Crece exponencialmente los sitios web creados y el número de usuarios que usan la web.

Web 4.0: En este modelo el usuario no solo interactuara con la web en búsqueda de informacion, sino que la web le mostrara soluciones completas a sus necesidades.

La exponencial evolución de la web y la cantidad masiva de datos generados en la red han generado el auge del Big Data.

Big Data: Procesamiento masivo de datos de diferente naturaleza que requieren una alta velocidad de proceso. Este se suele definir a través de las 3V:

- *Volumen de cantidad de datos.*
- *Velocidad del procesamiento de datos.*
- *Variedad de datos.*

Hoy día el uso del Big Data es incuestionable para todos los sectores. Para 2025 se estima que se alcanzaran 163 zettabytes; el Big Data tendrá una importancia estratégica en todos los sectores para gestionar tal cantidad de informacion.

2. Elementos de la arquitectura cliente-servidor.

Los elementos de este tipo de arquitectura varían en función de si estamos ante una arquitectura de dos capas o de tres.

La arquitectura cliente-servidor es un modelo de aplicación distribuida con dos componentes principales (servidor y cliente).

Servidor: Su tarea es proporcionar respuestas a las peticiones de los clientes.

Ventajas:

- Escalabilidad: Es posible aumentar la capacidad tanto de servidores como de clientes.
- Disponibilidad: Al estar distribuidas las funciones y responsabilidades entre varios ordenadores, es posible reemplazar, reparar, actualizar o trasladar un servidor sin que los clientes se vean afectados.
- Control centralizado por parte del servidor.

Desventajas:

- Posible congestión de tráfico en la red: Existe gran cantidad de peticiones simultaneas al mismo servidor.
- Potentes recursos hardware en el servidor para que sea capaz de procesar de forma óptima las peticiones de los clientes.
- Mantenimiento complejo.

2.1. Arquitectura de dos niveles.

Es un sistema con dos elementos (cliente y servidor).

En este modelo el servidor resuelve las peticiones que realizan los clientes.

En el mundo web se entiende esta arquitectura secuenciando el proceso por el cual un usuario solicita una web y la recibe en su ordenador siguiendo las siguientes etapas:

Servidor espera la solicitud del cliente -> Busca en su sistema de archivos el archivo solicitado por el cliente -> Si la petición puede ser atendida, la devuelve al cliente. El código HTML la interpreta el navegador del cliente -> Si la petición fracasa, da un mensaje de error.

Si hay una gran cantidad de procesos o son muy complejos, el modelo dos capas no resulta eficaz. Se produce una gran congestión en la red cuando los clientes tienen que acceder a descargar los datos del servidor. A este problema se le denomina cliente pesado.

2.2. Arquitectura de tres niveles.

Para soluciones más seguras, flexibles y escalables, es necesario optimizar el uso de recursos. Para ello hay que repartir las funciones en tres niveles:

- Cliente: Es el que realiza la petición de recursos al servidor.
- Servidor de aplicaciones: El servidor hará uso de otro servidor para poder resolver las peticiones de recursos realizadas por los clientes.
- Servidor de datos: Mediante el nivel de la base de datos es posible darle los datos al servidor de aplicaciones. Existen múltiples interfaces entre los dos servidores, dependiendo de los lenguajes de programación y del tipo de base de datos

A medida que crecen las aplicaciones web, más difícil son de mantener. El paradigma modelo-vista-controlador (MVC) separa en diferentes niveles o capas la arquitectura de una aplicación software. En caso de las aplicaciones web, las capas estarían formadas por la vista (página HTML), el controlador (programa que produce el HTML y

obtiene de forma dinámica los datos) y el modelo (capa que trabaja con los datos, por lo que habrá un base de datos que contiene diversa informacion almacenada).
Algunas ventajas de los MVC:

- *Separación de los datos de su representación visual.*
- *Escalabilidad.*
- *Control de errores.*

3. Aplicaciones web y aplicaciones de escritorio.

Aplicaciones de escritorio: Aplicaciones que se desarrollan en un sistema operativo específico para ser ejecutados bajo este. Estas son más o menos eficaces en función de la configuración hardware del ordenador donde se instalen.
Existen aplicaciones de escritorio tanto de pago como libres. Algunos problemas que se producen al usarlas son:

- *Incompatibilidad entre versiones.*
- *Difícil instalación y actualización.*
- *Coste elevado.*
- *En caso de un ámbito empresarial, podemos encontrarnos ordenadores que tengan diferentes versiones de un programa. Como consecuencia puede haber tanto perdida de funcionalidad como problemas de consistencia.*
- *Posible falta de portabilidad de la aplicación a diferentes sistemas operativos.*

Las aplicaciones de escritorio son ampliamente usadas en todos los ámbitos por su gran cantidad de prestaciones y su elevado tiempo de respuesta.
Aplicación web: Toda aplicación accesible a través de un navegador.

3.1. Ventajas del software web.

Ausencia de costes de actualización: Como son los navegadores los que visualizan las páginas web, es suficiente con realizar las actualizaciones en el servidor.
Datos centralizados: El servidor también aloja datos. Estos últimos suelen estar almacenados en una base de datos centralizada.
Ausencia de instalación: Solo es necesario un navegador para acceder al servicio.
Actualización constante: Los equipos siempre acceden a la última versión actualizada de la aplicación.
Movilidad: Acceso desde cualquier ubicación con conexión a internet.
Nota: Las aplicaciones web actualmente tienen las mismas funcionalidades que las de escritorio.

3.2. Desventajas del software web.

Menor potencia: Las aplicaciones de escritorio suelen tener mejores prestaciones y ser menos potentes.
Infrautilización del hardware: Las aplicaciones web no supeditan su rendimiento a unas mejores prestaciones de hardware del ordenador.
Conectividad rápida y constante: Para usar todas las prestaciones del software web, es necesario un acceso rápido y fiable a internet. En la actualidad no es muy preocupante ya que la conectividad a internet está muy extendida, por otra parte, muchas aplicaciones web pueden ser usadas en modo offline.

4. Tecnologías usadas en aplicaciones web.

La mayoría de aplicaciones web del mercado usan páginas web dinámicas, que se ejecutan en el servidor web y se visualizan en el cliente. Existen páginas de contenido estático y dinámico; cuando existe contenido dinámico se ejecuta código tanto en el cliente como en el servidor. Se va a diferenciar el código que se ejecuta en estos dos nodos.

4.1. En el lado servidor.

CGI (Common Gateway Interface): Al principio de internet, el servidor solo podía ejecutar programas del tipo C, Perl y Powershell. Estas instrucciones eran ejecutadas por el sistema operativo y se transmitían al navegador mediante el CGI.

ASP.NET (Active Server Pages): Framewrok libre de Windows orientado a objetos. Existen versiones para Linux y Unix.

Java: existe un gran grupo de tecnologías desarrollado por Sun Microsystems que se pueden ejecutar en el servidor, y algunas son JSF (JavaServer Faces), JSP (JavaServer Pages) y los servlets. De estas tres, la que más se usa es JSP, ya que es la más extendida y es compatible con casi todos los servidores web.

Ruby: Lenguaje interpretado de propósito general, dinámico y flexible. Es de alto nivel y es software libre y multiplataforma.

Perl: Lenguaje de programación que se ejecuta en el servidor. Apache tiene un módulo que permite ejecutar programas de este tipo. Toma muchas características del lenguaje C y se caracteriza por su destreza a la hora de procesar texto.

PHP: Lenguaje de propósito general del desarrollo backend más usado y popular que existe en el mercado. Es útil en el desarrollo de aplicaciones y motor de Wordpress y parecidos. Existen frameworks muy potentes como Laravel o Symfony. Una de las mejores ventajas que ofrece es que accede fácilmente a bases de datos como Oracle o Mysql, y también a bases de datos NoSQL o MongoDB. PHP tiene un inconveniente, ya que algunas funcionalidades tienen agujeros de seguridad si no se toman las medidas oportunas.

Python: Lenguaje interpretado muy poderoso, fácil de aprender y de alto nivel. Se usa en multitud de aplicaciones y sobretodo en aplicaciones de seguridad. Es un lenguaje orientado a objetos, que destaca por su tipado dinámico.

JavaScript: Lenguaje ligero, interpretado y orientado a objetos. Para aprenderlo es preciso un conocimiento básico de HTML y CSS. Interviene sobretodo en el contenido dinámico y en la interacción con el usuario. Permite controlar archivos de multimedia, creación de imágenes animadas, y animación 3D. Destaca por su aportación a la mayoría de los ámbitos de la tecnología.

4.2. En el lado cliente.

HTML y CSS: HTML es el lenguaje de marcas que se inventó junto con el navegador y el medio para transmitir informacion entre el cliente y el servidor. Posteriormente, surgió CSS que permitirá diseñar gráficamente la página. HTML posee una especificación que parte del W3C. Comenzó en la versión 1 y ya se ha publicado la versión 5.

JavaScript: Se puede usar tanto el cliente como en servidor. Es de los más extendidos porque es soportado en la mayoría de navegadores.

XML: Tecnología relacionada con el lenguaje de marcas, permite compartir datos, incluso formar parte de una base de datos. Los ficheros .xml se centran en la configuración de la mayoría de aplicaciones y de los servidores web y de aplicaciones.

5. Servidores y aplicaciones libres y propietarias.

Componentes de la arquitectura web:

- ***Sistema operativo:*** Software base sobre el cual se sustentan los demás componentes.
- ***Servidor web:*** Servicio que va a atender las peticiones de los clientes y les va a responder lo más pronto posible.
- ***Bases de datos:*** Principal fuente de informacion de la que se nutre el servidor web para mostrar al cliente la informacion que solicita. Componente básico en cualquier portal de hoy día. Existen dos grandes grupos: SQL y NoSQL.
- ***Lenguaje de programación:*** Lenguaje que se encarga de la lógica de la aplicación. Tiene como función recibir las peticiones de los clientes, consultar la informacion en la base de datos y responder a las peticiones. Dependiendo de la aplicación, se pueden usar varios lenguajes de programación en la aplicación delimitando sus funciones. La clave de que funcione todo correctamente es el encapsulamiento y modulación de los componentes de la aplicación.

Teniendo en cuenta los componentes mencionados, existe una plataforma clave para trabajar en el desarrollo de una aplicación, esta es XAMPP, es conocida por su popularidad y uso en gran parte del entorno productivo empresarial.

XAMPP es de software libre e incluye los cuatro componentes mencionados, de ahí su popularidad y crecimiento exponencial en el desarrollo de aplicaciones.

Sus iniciales provienen de los siguientes componentes:

- ***X:*** Cualquier sistema operativo.
- ***Apache:*** Servidor web.
- ***MariaDB:*** Base de datos. En versiones más antiguas incluia MySQL.
- ***PHP, Perl:*** Lenguaje de programación del servidor.

6. Protocolo HTTP vs HTTPS.

El protocolo antonomasia es HTTP y el seguro HTTPS.

HTTP: Protocolo de la capa de aplicación que escucha por el puerto 80 basado en el modelo cliente-servidor. Se basa en TCP, por lo que es orientado a conexión y escucha las peticiones de los clientes. Una vez aceptada la conexión, se encarga de mantenerla y garantizar la transferencia de datos sin errores.

HTTPS: Protocolo de la capa de aplicación que se escucha por el puerto 443, también se basa en el modelo cliente-servidor, pero en modo seguro. Se basa en TCP, es orientado a conexión y encripta los datos para asegurar una transmisión segura entre los dos extremos. Para realizar esta encriptación necesita de certificados, siempre y cuando sean válidos. Al cifrar la informacion en el servidor y descifrarla en el cliente se pierde tiempo de computación, aunque solamente necesite cifrar cierta informacion.

Cualquier servidor web medio serio posee la capacidad de emitir certificador. En este entorno, la parte fundamental son los navegadores, ya que deben validar dichos certificados a partir de entidades certificadoras. Es necesario tener una entidad que sea fiable, como FNMT, que es la encargada en España de suministrar los certificados a los ciudadanos.

El protocolo HTTPS usa la encriptación SSL (Secure Socket Layer) que encripta datos sensibles (actualmente se le llama TLS). Un certificado SSL o TLS se instala en el servidor.

7. Servidores de aplicaciones.

Servidores web más usados, que permiten ejecutar en su entorno diferentes tecnologías:

- ***Apache:*** Uno de los servidores más usados en Internet. Su nombre completo es Apache HTTP Server. Su primera versión fue en 1996, ha sido un referente como servidor y sus mayores competidores son NGinx y Microsoft IIS.
  Tiene como ventaja que es de código abierto y multiplataforma. Entre sus desventajas están el bajo rendimiento cuando se realizan muchas peticiones por parte de los clientes.
- ***NGinx:*** Servidor de código abierto y gratuito nacido en 2004, es de los más usados por las compañías del mundo. Destaca por su alto rendimiento e incluye funciones como balanceador de carga, POP3 e IMAP.
  Se puede instalar en Windows, Linux y Unix. Es altamente escalable, por si fuera necesario una ampliación y consume muy pocos recursos para resolver las peticiones de los clientes.
- ***Microsoft IIS:*** Software de Microsoft conocido por Internet Information Services. Comenzó en Windows NT. Permite el procesamiento de páginas del tipo ASP, ASP.NET, PHP o Perl. Está programado con C++, pero tiene una gran desventaja y es que solamente se puede instalar en Windows, lo que conlleva una amenaza de seguridad constante. Además, posee servicios como FTP y SMTP.

8. Instalación y configuración básica del servidor Apache.

8.1. Definición y características.

Apache HTTP Server es un software gratuito de código abierto para plataformas Linux, Unix y Windows. Es desarrollado y mantenido por Apache Software Foundation.
Algunas características que nos ofrece este servidor:

- Autenticación y autorización.
- Host virtuales.
- Modularización.
- Seguridad mediante cifrado.
- Monitorización mediante archivos de logs.
- Páginas web estáticas y dinámicas.

8.2. Instalación y configuración del servidor Apache en Ubuntu.

Se puede instalar tanto en sistemas operativos sin interfaz gráfica o con interfaz gráfica.
XAMPP es la versión más usada en Windows y Linux. Dependiendo del sistema para el que vayan orientados, estos paquetes se llaman WAMP y LAMP genéricamente. Para instalar se hace un ***sudo apt update/upgrade*** y luego un ***sudo apt install apache2***.
También se puede instalar desde el gestor de paquetes Synaptic, pero eso no nos interesa (a MA si que no le interesa pare).
Tras instalarse, iniciamos el servidor web en la maquina donde hemos instalado y en el cliente nos dirigimos a la ruta: http://localhost o http://127.0.0.1
También puedes usar el nombre del dominio o la dirección IP del servidor.

8.3. Arranque y parada de Apache.

Cuando se instala Apache, se queda iniciado y preparado para iniciarse automáticamente cada vez que se inicie el servidor. Como es lógico puedes detenerlo

y reiniciarlo cuando te salga de los cojones. Los comandos son los siguientes para ver si furula o quieres detener/reiniciar:

- ***service apache2 status*** #Comprueba el servicio.
- ***service apache2 stop*** #Para el servicio
- ***service apache2 start*** #Inicia el servicio.
- ***service apache2 restart*** #Reinicia el servicio.

Teniendo en cuenta la funcionalidad de los comandos anteriores según el orden, estan también los siguientes comandos:

- ***apachectl status***
- ***apachectl stop***
- ***apachectl start***
- ***apachectl restart***
- ***apachectl configtest*** # Ejecuta una prueba sintáctica para comprobar que los ficheros de configuración son correctos

8.4. Estructura de Apache2.

La estructura de Apache2 es la siguiente (suponiendo que estamos mirando en su directorio en /etc/apache2):

- ***apache2.conf:*** Fichero de configuración principal del servidor apache2. Contiene las variables globales. Cualquier cambio en este fichero implicaría reiniciar el servicio.
- ***conf-avaliable:*** Directorio que contiene configuraciones adicionales asociadas a un módulo en particular. Las configuraciones no están activas.
- ***conf-enabled:*** Directorio que contiene enlaces al directorio anterior para poder activar las configuraciones que contiene.
- ***envars:*** Fichero en el que se definen variables de entorno como APACHE_RUN_USER, APACHE_RUN_GROUP, etc. Que en principio no son necesarias de modificar.
- ***mod-avaliable/enabled:*** Directorios similares a los sites, pero para módulos que se pueden acoplar al servidor web.
- ***ports.conf:*** Fichero incluido en el fichero apache2.conf, permite configurar los puertos e IP por las que escucha el servidor. Por defecto el puerto es 80.
- ***sites-avaliable:*** Directorio que contiene los ficheros de los hosts virtuales definidos en el servidor, que pueden ser diferentes. Estos están disponibles, pero no activos.
- ***sites-enabled:*** Directorio similar al anterior pero las definiciones de hosts virtuales que se están usando, normalmente son enlaces simbólicos a los ficheros que se encuentran en sites-available. Por defecto, tiene el directorio /var/www/html, que es donde se almacena la página principal de apache.

Nota: Cuando se instala apache y accedes en la web a localhost, intervienen los ficheros 000-default.conf y apache2.conf. La configuración de ambos es lo que permite acceder a un usuario.

- ***000-default.conf:*** Sitio virtual que se configura en el servidor web. Este fichero tiene varias directivas, siendo una de las más importantes el DocumentRoot. Esta permite configurar una ruta donde se ubicará el fichero index.html. Para ello se ha creado un host virtual que escucha por el puerto 80 con la directiva <VirtualHost *:80>

Para habilitar el sitio es necesario usar el comando a2ensite y para deshabilitarlo es el comando a2dissite. Las directivas relacionadas con esta configuración se encuentran en el fichero apache2.conf.

El primer bloque <Directory /> deniega todos los accesos: Esto se realiza como método de seguridad para que nadie acceda al sistema de ficheros del sistema operativo. Dicho bloque, por defecto tiene las directivas:

- ***Option FollowSymLinks:*** Permite los enlaces simbólicos a otros directorios.
- ***AllowOverride None:*** Ignora el fichero .htaccess para que no existan demasiadas llamadas a este fichero.
- ***Require all denied:*** Deniega el permiso a todo sistema de ficheros.

Y en el bloque <Directory /var/www>, se incluyen las siguientes directivas que afectara al directorio /var/www:

- ***Options Indexes FollowSymLinks:*** Indexes permite la visualización del directorio en caso de que no exista en el directorio el fichero index.html o los patrones declarados en el servidor Apache. FollowSymLinks ya sabéis que hace.
- ***AllowOverride None:*** Esta explicado antes.
- ***Require all granted:*** Da permiso a todo el mundo y desde cualquier IP a directorio /var/www

```
<Directory />
        Options FollowSymLinks
        AllowOverride None
        Require all denied
</Directory>

<Directory /usr/share>
        AllowOverride None
        Require all granted
</Directory>

<Directory /var/www/>
        Options Indexes FollowSymLinks
        AllowOverride None
        Require all granted
</Directory>
```