UNIDAD 01 Implantación de arquitecturas web

ÍNDICE

Arquitectura web

La WWW (World Wide Web) es un servicio de distribución de información:

```
- Permite el acceso, vía Internet, a millones de recursos electrónicos y
aplicaciones distribuidos en servidores.
```

```
- Los recursos están localizados en diferentes URL e identificados de forma única
(URI).
- Los recursos pueden conectarse entre sí a través de hipervínculos.
```

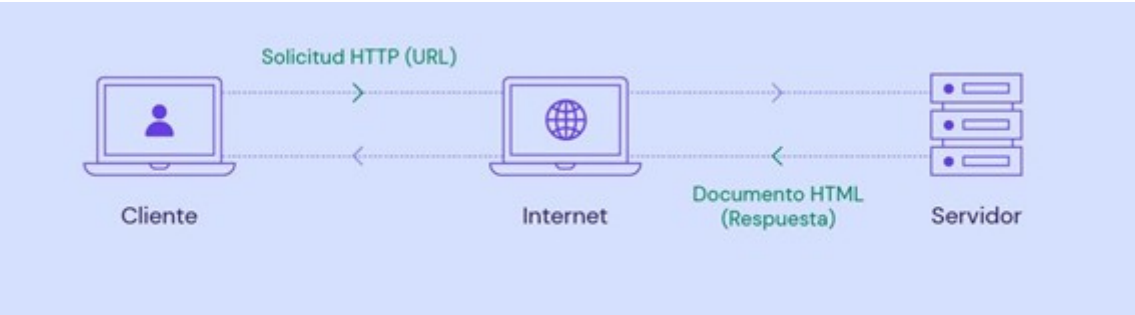
Si observamos el esquema de funcionamiento de los servicios Web, vemos que requiere tres elementos fundamentales:

1. Proveedor del servicio Web, que es quien lo diseña, desarrolla e implementa y lo pone disponible para su uso, ya sea dentro de la misma organización o en público.
2. Consumidor del servicio, que es quien accede al componente para utilizar los servicios que éste presta.
3. Agente del servicio, que sirve como enlace entre proveedor y consumidor para efectos de publicación, búsqueda y localización del servicio.

En una aplicación web el usuario interactúa con el navegador que accede a los servicios y recursos que le ofrece un servidor web. Si trasladamos esta idea una arquitectura web, tenemos que se trata de un conjunto de **capas y protocolos**.

La Arquitectura Web más conocida y utilizada es la WWW (World Wide Web), que permite a muchos usuarios acceder a cantidad de aplicaciones y proporciona la plataforma donde los desarrolladores crean aplicaciones y servicios.

La Arquitectura WWW proviene del modelo **cliente-servidor**:



Funcionamiento cliente-servidor: Las aplicaciones y los contenidos son presentados en formatos de datos estándar y localizados por aplicaciones conocidas como navegadores o web browsers, que envían requerimientos de objetos a un servidor y éste responde con el dato codificado según un formato estándar.

Para que este modelo pueda funcionar, la arquitectura World Wide Web provee un modelo de programación, donde:
- Las aplicaciones y los contenidos son presentados en formatos de datos estándar y son localizados por aplicaciones conocidas como "Web browsers" o navegadores.
- Los navegadores o Web browsers envían sus peticiones de objetos a un servidor y éste responde con el dato codificado según un formato estándar.
Observamos que estamos hablando de datos y formatos estándar. Luego, otro aspecto a tener en cuenta es el uso de estándares W3C para WWW, que se encargan de proporcionar ambientes de aplicación de propósito general. Por ejemplo:
- Modelo estándar de nombres - URL: todos los servidores, así como el contenido de la WWW se denominan según un Localizador Uniforme de Recursos (Uniform Resource Locator: URL).

Diferencia entre URI y URL URL especifica la ubicación de un recurso en Internet, mientras que un URI se puede utilizar para identificar cualquier tipo de recurso, no solo los que se encuentran en Internet

```
- Formatos de contenidos estándar: todos los navegadores soportan un conjunto de
formatos estándar, por ejemplo, HTML, XML JavaScript, etc.
- Protocolos estándar: éstos permiten que cualquier navegador pueda comunicarse
con cualquier servidor Web. El más comúnmente usado en WWW es HTTP (Protocolo de
Transporte de HiperTexto), que opera sobre el conjunto de protocolos TCP o UDP.
```

HTTP

El protocolo HTTP permite realizar la transferencia de **información entre un cliente web y un servidor web** en Internet, dando lugar a lo que hoy día se conoce como World Wide Web. Este protocolo ha pasado por varias versiones, desde la inicial 0.9 hasta la actual 3.0. Veamos cuales son las características principales:

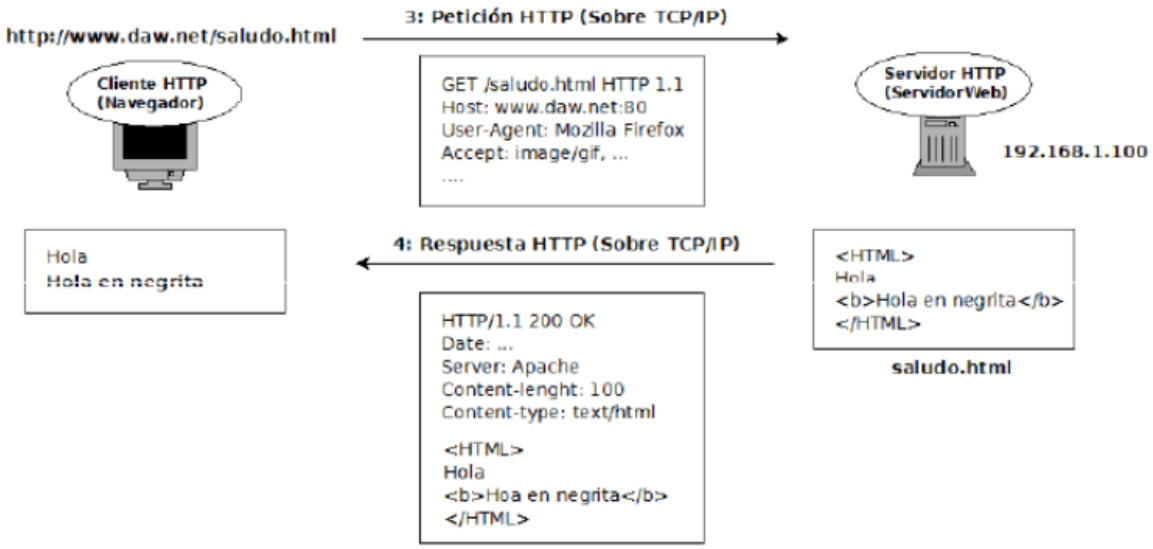
- Es un protocolo del nivel de aplicación, por lo que está destinado a que aplicaciones en sistemas diferentes se comuniquen entre sí.
- Es un protocolo sin estado, no hay consciencia de las peticiones anteriores, simplemente se hace una petición de un recurso y se obtiene el recurso. Por lo que es un protocolo donde la comunicación consiste en realizar una petición y obtener una respuesta del servidor a dicha petición.
- Las peticiones de recursos desde un cliente web al servidor web se hacen en formato texto "legible" por un ser humano, también lo son las respuestas por parte del servidor.
- No existe el concepto de sesión y no es necesario autenticarse para realizar la petición de un recurso. Como podemos ver el protocolo HTTP es muy simple, por lo que cuando desarrollamos una aplicación web es necesario que cosas como la autenticación sea implementada por la aplicación web, dado que el protocolo no provee de mecanismos para ello.

Funcionamiento del protocolo HTTP

Ya se ha comentado que el protocolo HTTP tiene un funcionamiento bastante sencillo basado en el envío de mensajes entre cliente y servidor. Gráficamente podemos resumir el proceso de comunicación HTTP como sigue:

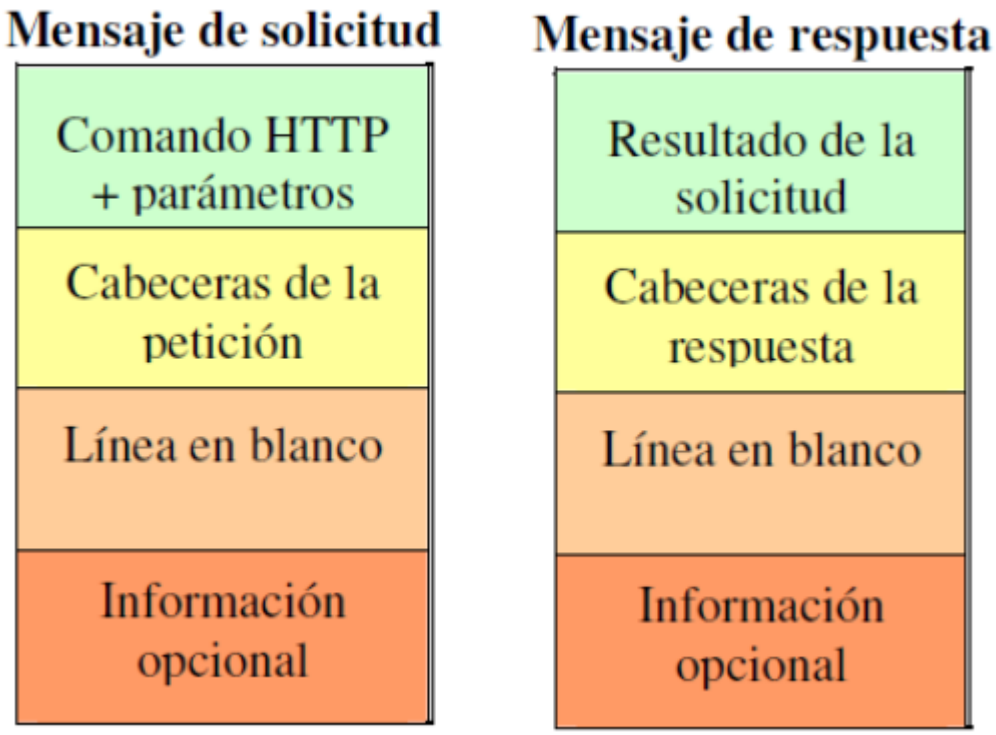
1. Un usuario accede a una URL, seleccionando un enlace de un documento HTML o introduciéndola directamente en el campo correspondiente del cliente Web.
2. El cliente Web descodifica la URL, separando sus diferentes partes: el protocolo de acceso, la dirección DNS o IP del servidor, el posible puerto opcional (el valor por defecto es 80) y el objeto requerido del servidor. http://direccion[:puerto][path] https://campus.educantabria.es/curso/DAW2

3. Se abre una conexión TCP o UDP (dependiendo de la versión HTTP) con el servidor, llamando al puerto TCP o UDP correspondiente. En ese momento, se realiza la petición HTTP. Para ello, se envía el comando necesario (GET, POST, HEAD,...), la dirección del objeto requerido (el contenido de la URL que sigue a la dirección del servidor), la versión del protocolo HTTP empleada y un conjunto variable de información, que incluye datos sobre las capacidades del navegador (browser), datos opcionales para el servidor, etc.
4. El servidor devuelve la respuesta al cliente. Consiste en un código de estado y el tipo de dato MIME de la información de retorno, seguido de la propia información.
5. Se cierra la conexión TCP o UDP. Este proceso se repite en cada acceso al servidor HTTP. Por ejemplo, si se recoge un documento HTML en cuyo interior están insertadas 2 imágenes y 1 vídeo, el proceso anterior se repite cuatro veces, una para el documento HTML y tres más para los recursos (la dos imágenes y el vídeo).



1) El usuario introduce una **URI (o URL)** en la barra de direcciones del navegador o hace clic sobre un hiperenlace.
2) El navegador analiza la URL y establece una conexión TCP con el servidor web.
3) El navegador envía un **mensaje HTTP de petición** que depende del la URI (o URL).
4) El servidor envía un **mensaje de respuesta** que depende de la petición enviada y del estado del servidor.
5) Se cierra la conexión TCP.

El diálogo con los servidores HTTP se establece a través de mensajes formados por líneas de texto, cada una de las cuales contiene los diferentes comandos y opciones del protocolo. Solo existen dos tipos de mensajes, unos para realizar las peticiones y otro para las respuestas.



Métodos HTTP

HTTP define un conjunto de métodos de petición para indicar la acción que se desea realizar para un recurso determinado. Según fueron desarrollándose nuevas versiones del protocolo implementaron nuevos métodos. Algunos de los más utilizados son los siguientes:

MÉTODOS HTTP

Método HTTP	Significado en Restful Web Services
GET	Se utiliza para operaciones de sólo lectura. No generan ningún cambio en el servidor.
DELETE	Elimina un recurso en específico. Ejecutar esta operación múltiples veces no tiene ningún efecto.
POST	Cambia la información de un recurso en el servidor. Puede o no regresar información.
PUT	Almacena información de un recurso en particular. Ejecutar esta operación múltiples veces no tiene efecto, ya que se está almacenando la misma información sobre el recurso.
HEAD	Regresa solo el código de respuesta y cualquier cabecero HTTP asociado con la respuesta.
OPTIONS	Representa las opciones disponible para establecer la comunicación en el proceso de petición/respuesta de una URI.

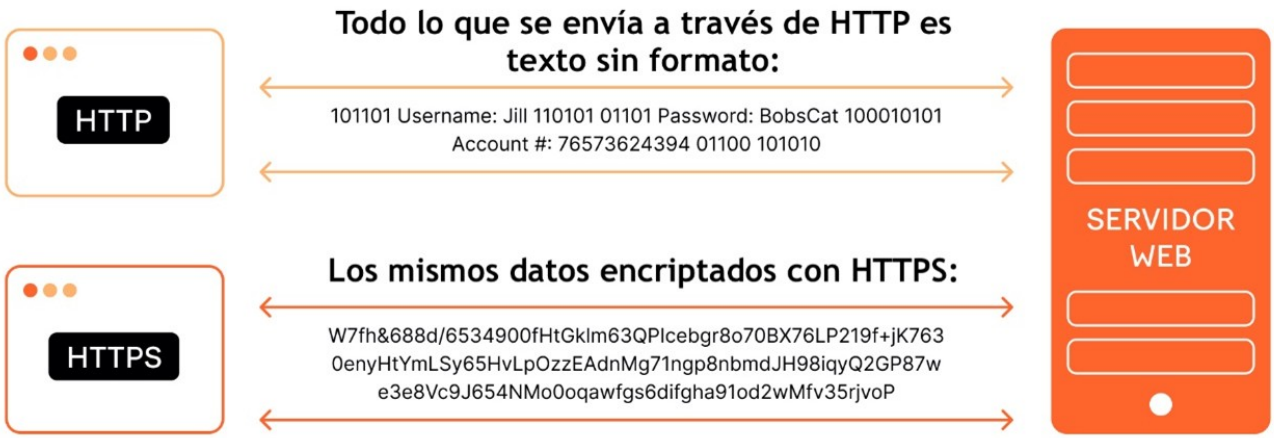
Después de realizar una petición como las anteriores, sea el método que sea, el servidor generará una respuesta HTTP que contendrá un código de estado de respuesta. Existen múltiples códigos de respuestas, por ejemplo:

- 1XX: los códigos de estado en el rango de 100 son respuestas informativas.
  - Ej.: 100 Continue: significa que todo va bien por ahora y que el cliente debería seguir con la petición.

- 2XX: los códigos de estado en el rango de 200 son respuestas de éxito en la transferencia, por ejemplo:
  - Ej.: 200 OK: significa que la respuesta ha tenido éxito y generalmente implica que los datos van en la respuesta (depende del método de la petición).
- 3XX: los códigos de estado en el rango de 300 son respuestas de redirección, el recurso ha cambiado de ubicación.
  - Ej.: 301 Moved Permanently: significa que un recurso se ha movido a otro sitio de forma permanente. El servidor web incluye una cabecera llamada Location: con la nueva ubicación. El navegador realizará una petición a la nueva ubicación de forma transparente al usuario.
- 4XX: los códigos de estado en el rango de 400 son respuestas en las que se indica que hay un error en la petición del cliente o que algo ha ido mal al procesarla. Veamos algunos ejemplos:
  - Ej.: 404 Not Found: posiblemente el código de estado más conocido. El servidor responde con este código cuando el recurso solicitado no existe o no puede ser encontrado.
- 5XX: los códigos de estado que están en el rango de 500 simbolizan que se ha recibido una petición correcta pero que el servidor no ha podido completar la respuesta. Veamos algunos ejemplos:
  - Ej.: 503 Service Unavailable: el servidor no está disponible en este momento.

HTTPS

El Protocolo seguro de transferencia de hipertexto (en inglés, Hypertext Transfer Protocol Secure o HTTPS) es un protocolo de aplicación basado en el protocolo HTTP, destinado a la transferencia segura de datos de hipertexto, es decir, es la versión segura de HTTP. La web es insegura por naturaleza. Cuando se diseñaron los protocolos en los que está basada (TCP o UDP) no se tuvieron en cuenta muchos de los problemas que tiene la Internet moderna. Y el protocolo HTTP, para transferir páginas web, no añadió nada al respecto tampoco hasta mucho después, con la introducción del protocolo HTTPS (la "ese" es de "Seguro") allá por 1994 por la empresa Netscape. Para asegurar la información, el protocolo HTTPS requiere de certificados y siempre y cuando sean validados la información será transferida cifrada. Pero cifrar la información requiere un tiempo de computación, por lo que será perjudicado el rendimiento del servidor web.



Tecnologías asociadas a las aplicaciones web

Actualmente, las aplicaciones Web emplean páginas dinámicas, que se ejecutan en un servidor Web y que se muestran en el navegador del equipo cliente que ha realizado la solicitud. Cuando una página Web llega al navegador, es posible que también incluya algún programa o fragmento de código que se deba ejecutar. Ese

código, normalmente en lenguaje JavaScript, lo ejecutará el propio navegador. Las tecnologías asociadas a las aplicaciones Web que se ejecutarán tanto del lado del servidor como del cliente:

- ASP (Active Server Pages): Las "Páginas Activas" se ejecutan del lado del servidor, de este modo se forman los resultados que luego se mostrarán en el navegador de cada equipo cliente que ha realizado la solicitud. - Un ejemplo son los buscadores, donde un usuario realiza una petición de información y el servidor nos entrega un resultado a medida de nuestra petición. - Existen versiones de ASP para Unix y Linux, a pesar de que fue una tecnología desarrollada por Microsoft para la creación dinámica de páginas Web ofrecida junto a su servidor IIS.

- PHP (Hypertext Preprocessor): Este lenguaje es ejecutado en el lado del servidor. 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 y, por lo tanto, suele utilizarse para crear páginas dinámicas complejas.

- CGI (Common Gateway Interface): La "Interface Común de Entrada" es uno de los estándares más antiguos en Internet para trasladar información desde una página a un servidor Web. - Este estándar es utilizado para bases de datos, motores de búsqueda, formularios, generadores de mail automático, foros, comercio electrónico, rotadores y mapas de imágenes, juegos en línea, etc. - Las rutinas de CGI son habitualmente escritas en lenguajes interpretados como Perl o por lenguajes compilados como C.

- CSS (Cascading Style Sheets): Las "Hojas de Estilo en Cascada" se usan para formatear las páginas Web; se trata de separar el contenido de un documento de su presentación. Cualquier cambio en el estilo afecta a todas las páginas que incluyan eses CSS.

- Java: Lenguaje que se ejecuta en el navegador del equipo cliente y no en el servidor. Características: - Una misma aplicación puede funcionar en diversos tipos de ordenadores y sistemas operativos: Windows, Linux, Solaris, MacOS, etc., así como en otros dispositivos inteligentes. - Los programas Java pueden ser aplicaciones independientes (que corren en una ventana propia) o "applets", que son pequeños programas interactivos que se encuentran incrustados en una página Web y pueden funcionar con cualquier tipo de navegador: Explorer, Netscape, Ópera, etc. - Se trata de un lenguaje "orientado a objetos". Esto significa que los programas se construyen a partir de módulos independientes, y que estos módulos se pueden transformar o ampliar fácilmente. Un equipo de programadores puede partir de una aplicación existente para extenderla con nuevas funcionalidades. - Desarrollado por la empresa Sun Microsystems, pero posteriormente liberado bajo licencia GNU GPL, con lo cual es un software libre.

- JavaScript: Lenguaje que se interpreta y se ejecuta en el cliente. - Útil para realizar tareas como mover imágenes por la pantalla, crear menús de navegación interactivos, utilizar algunos juegos, etc. - En las páginas Web suele preferirse JavaScript porque es aceptado por muchos más navegadores que VBScript (creado por Microsoft).

- VBScript (Visual Basic Scripting): La respuesta de Microsoft a JavaScript. VBScript es una buena herramienta para cualquier sitio destinado a ser mostrado exclusivamente en el navegador Microsoft Internet Explorer. - El código en VBScript puede, además, estar diseñado para su ejecución en el lado del cliente o en el del servidor, la diferencia es que un código que se ejecuta en el lado del servidor no es visible en el lado del cliente. Éste recibe los resultados, pero no el código.

De forma genérica podríamos decir que la arquitectura Web es un modelo compuesto de tres capas:

- Capa de Base de Datos, donde estaría toda la documentación de la información que se pretende administrar mediante el servicio Web y emplearía una plataforma del tipo MySQL, PostgreSQL, etc.
- Capa de servidores de aplicaciones Web, ejecutando aplicaciones de tipo Apache, Tomcat, etc.
- Capa de clientes del servicio Web al que accederían mediante un navegador Web como Firefox, Internet Explorer, Opera, Chrome etc.

Modelos de capas

La arquitectura de un sitio Web tiene el objetivo de servir de ayuda a los usuarios a encontrar y manejar la información, comprenden:

- Los sistemas de organización y estructuración de los contenidos.
- Los sistemas de recuperación de información y navegación que provea el sitio Web. Centraremos el estudio de los modelos de arquitectura Web relacionados, en función de cómo implementan cada una de las capas establecidas en una aplicación Web:

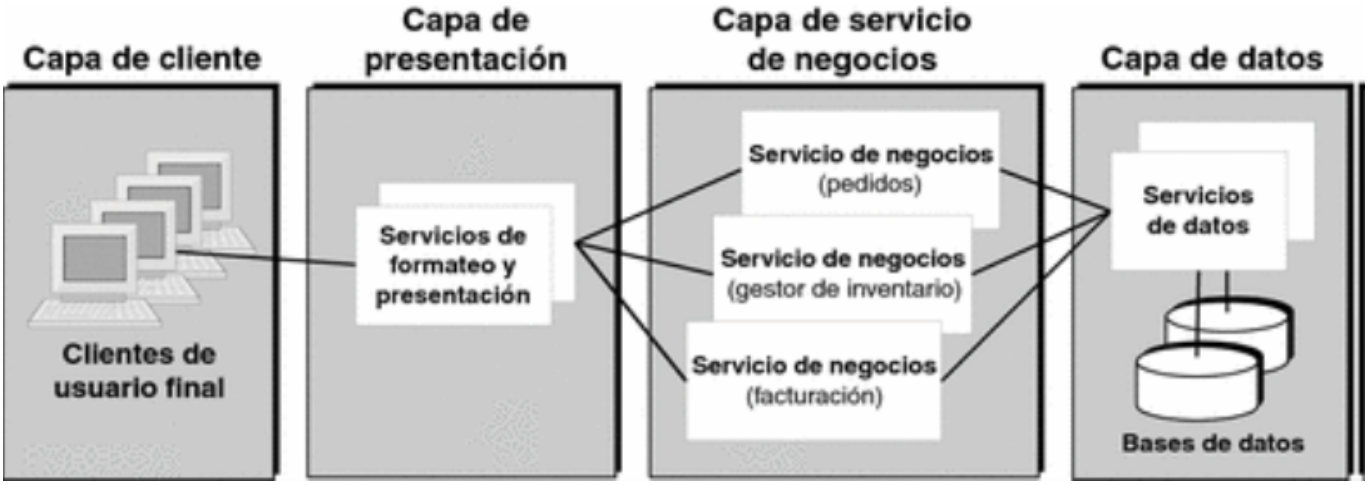
Modelo de 3 capas

Este modelo básico diferencia 3 capas:

1. Capa de presentación es la encargada de la navegabilidad, validación de los datos de entrada, formateo de los datos de salida, presentación de la Web, etc.; se trata de la capa que se presenta al usuario.
2. Capa de negocio es la que recibe las peticiones del usuario y desde donde se le envían las respuestas; en esta capa se verifican que las reglas establecidas se cumplen.
3. Capa de acceso a datos es la formada por determinados gestores de datos que se encargan de almacenar, estructurar y recuperar los datos solicitados por la capa de negocio.

Modelo de 4 capas

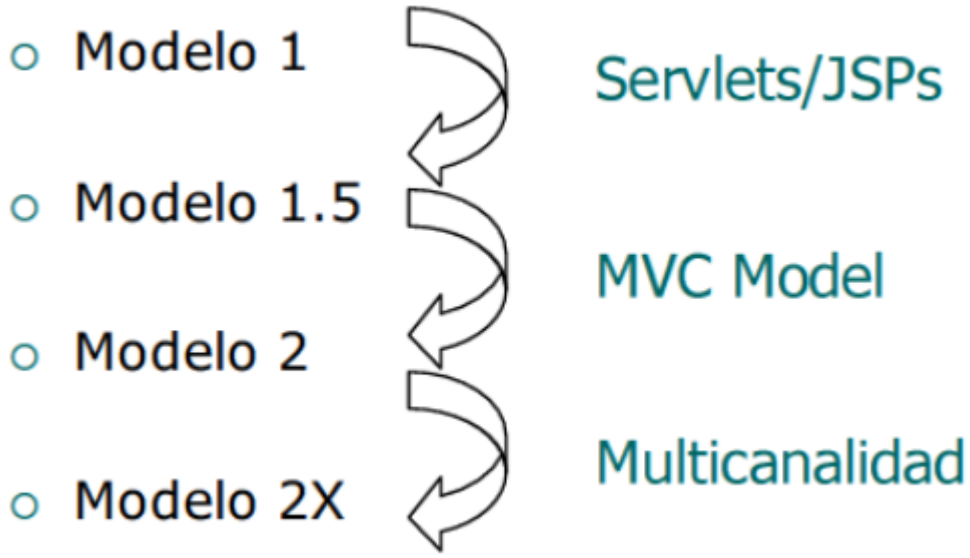
La arquitectura de 4 capas es una ampliación de la anterior. Esa cuarta capa sería la de servicios.
Capa de servicios: permite desacoplar la interfaz de usuario del resto de capas, permitiendo que las funcionalidades de nuestra aplicación sean accesibles por otras aplicaciones u servicios.



La arquitectura de 3 capas también desacopla la interfaz de usuario del resto de capas, pero se desarrollan aplicaciones monolíticas donde la interfaz de usuario se presenta como una interfaz orientada a unas funcionalidades concretas y no una interfaz integradora que permite que el usuario pueda acceder a todos los servicios de su organización. Además, con la arquitectura de 3 capas, tarde o temprano se tiende a producir un fuerte acoplamiento entre la capa de negocio y la de interfaz de usuario que complicará el mantenimiento de nuestra aplicación. Con una arquitectura de 4 capas no se plantea el desarrollo de una aplicación al uso, sino el desarrollo de un sistema compuesto por servicios que interactúan.

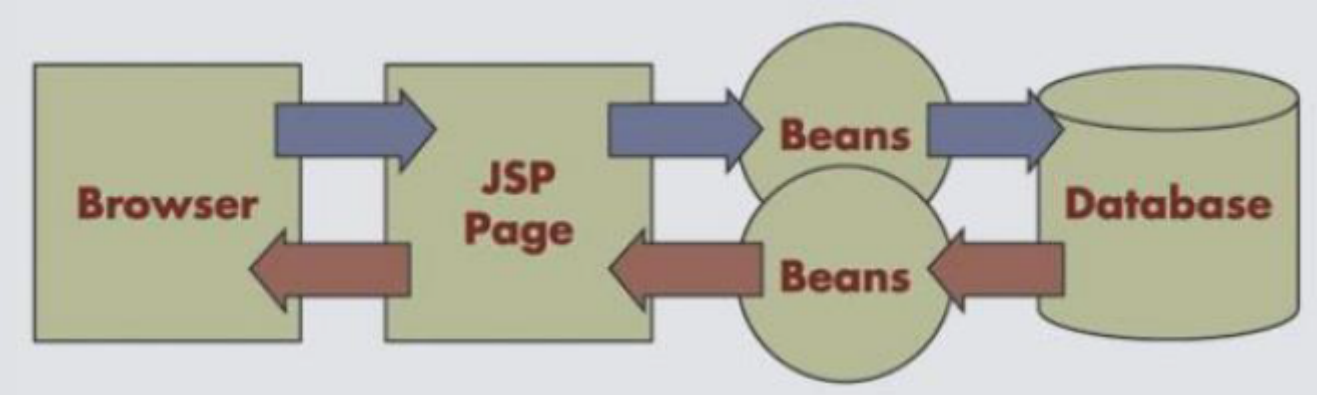
Evolución de los modelos de arquitectura web

Los modelos de arquitecturas Web han ido cambiando con la evolución de las aplicaciones para Internet y de los medios informáticos.

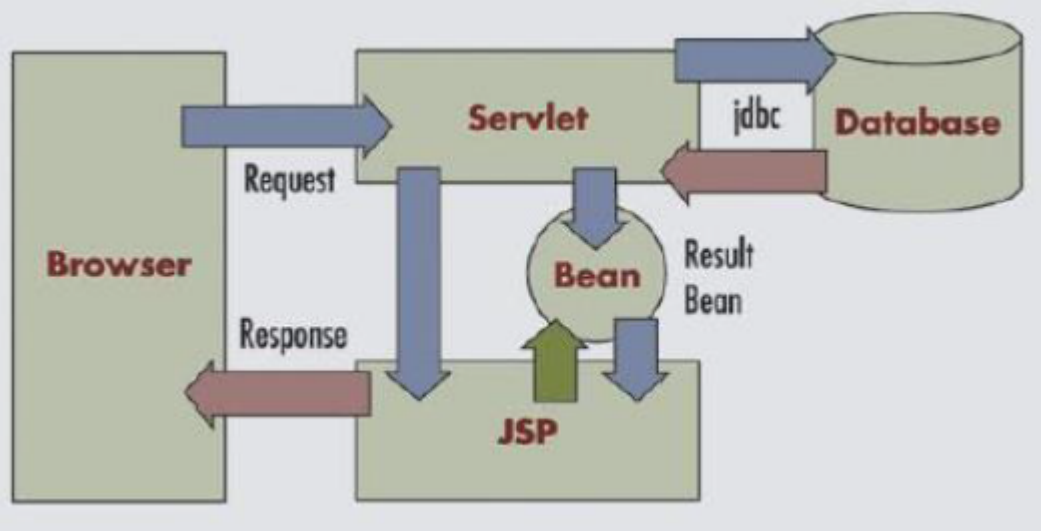


Modelo 1 En este caso las aplicaciones se diseñan en un modelo Web CGI (apartado 3), basadas en la ejecución de procesos externos al servidor Web, cuya salida por pantalla era el HTML que el navegador recibía en respuesta a su petición. Presentación, negocio y acceso a datos se confundían en un mismo script Perl.

Modelo 1.5 Aplicado a la tecnología java, se da con la aparición de las JSP(JavaServer Pages) 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 (Modelo Vista-Controlador) 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 un terminal de telefonía móvil, desde una tablet 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.

Aspectos generales de la arquitectura web

Los aspectos generales que destacar en una arquitectura Web son los siguientes:

- Escalabilidad.
- Separación de responsabilidades.
- Portabilidad.
- Utilización de componentes en los servicios de infraestructura.
- Gestión de las sesiones del usuario.
- Aplicación de patrones de diseño.

Escalabilidad

Es importante tener en cuenta que algunas aplicaciones web pueden presentar un crecimiento vertiginoso del número de usuarios. Para que cuando esto suceda, nuestra aplicación siga funcionando, es importante:

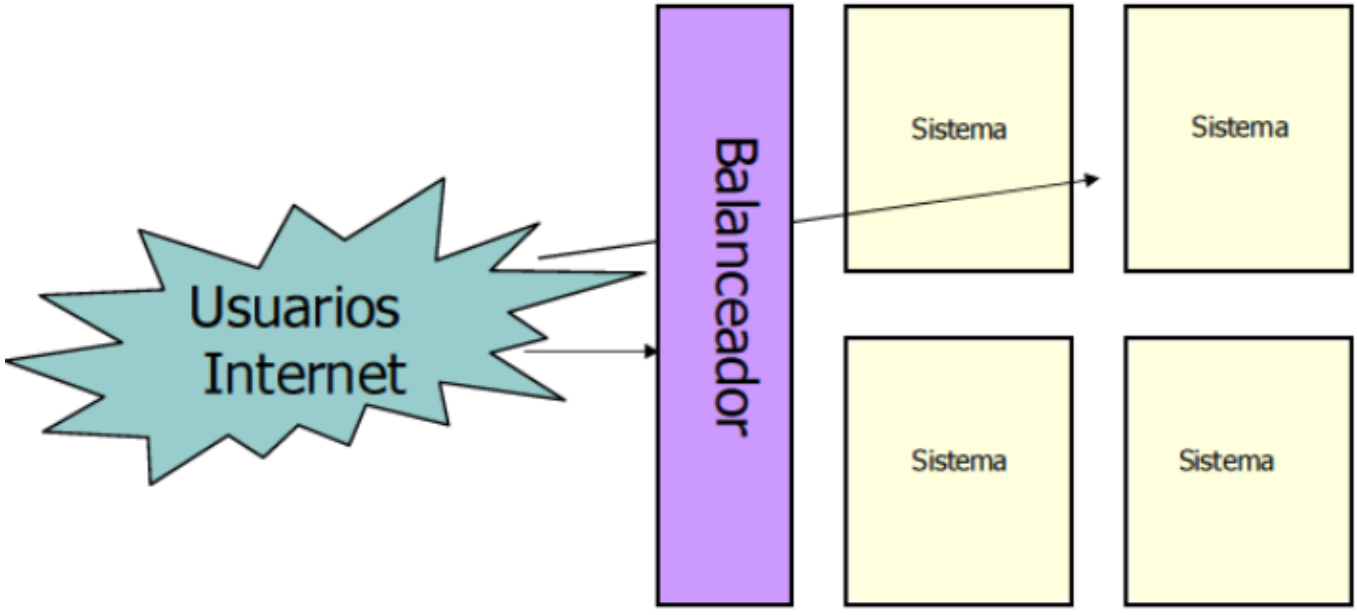
- El correcto dimensionamiento de la aplicación.
- La adaptabilidad del sistema al incremento de la demanda.

Podemos tener varias opciones:

- Escalabilidad Horizontal.
- Escalabilidad Vertical.

Escalabilidad horizontal

La escalabilidad horizontal se basa en añadir más recursos para dividir la carga. Por ejemplo, se clona el sistema y balanceamos la carga:



Escalabilidad Horizontal. Balanceador Hardware

Utilizar algoritmos predeterminados (Round Robin, LRU, etc.), para distribuir las peticiones HTTP entre los diferentes clones del sistema. La selección del clon es aleatoria, de forma que no podemos garantizar que las distintas peticiones del mismo usuario sean atendidas por el mismo clon. Luego tenemos un problema, no se mantiene la sesión del usuario en el servidor, lo cual condicionará nuestro diseño. La sesión la debe mantener mediante el uso de Cookies o en Base de Datos.

Escalabilidad Horizontal. Balanceador Software.

El paquete se examina a nivel de protocolo HTTP para garantizar el mantenimiento de la sesión de usuario. Así, distintas peticiones de usuario son atendidas por el mismo clon del servidor. Resultan más baratos que los balanceadores HW; pero también son más lentos. Ejemplo: Módulo mod_jk de apache. También existe un balanceador intermedio entre el Balanceador Horizontal HW y el SW, que resulta más rápido que los SW pero menos que los HW.

Escalabilidad Vertical.

La separación lógica entre capas se implementa de forma que permita separación física de las mismas. Para ello, necesitamos un Middleware entre las capas que permitan la comunicación remota.

Separación de responsabilidades

La **separación de responsabilidades** es un requisito básico si pensamos en la **separación de capas**: distintas responsabilidades no deben ser delegadas en la misma clase (separación de incumbencias). La tendencia actual en aplicaciones Web es usar Arquitectura n-capas, donde el modelo básico es el modelo de tres capas (como se ha comentado en el apartado 4).
Capa de Presentación: sus responsabilidades, entre otras, son:

- Navegabilidad del sistema.
- Validación de datos de entrada.
- Formateo de datos de salida.
- Renderizado de presentación.
  Capa de negocio: Responsabilidades básicas:
- Lógica de negocio o dominio del sistema.
- Resultado del análisis funcional: Conjunto de reglas de negocio que abstraen del mundo real.
- Debe ser independiente de la capa de presentación.
  Capa de datos:
- Responsabilidades de lógica de persistencia: Inserción, eliminación, actualización, búsquedas, etc.
- No tiene por qué ser siempre una base de datos relacional.

Portabilidad

La aplicación web debe adaptarse a las distintas arquitecturas físicas posibles en el despliegue. Las tareas de adaptación a los entornos nuevos solo deben implicar cambios en la configuración, pero no en el desarrollo.

Utilización de componentes en los servicios de infraestructura

Los componentes de la infraestructura deben ser independientes del dominio. Ejemplos: Pool JDBC, Gestor de permisos de acceso, Servicios de Log, etc.

Gestión de la sesión de usuario

Ya hemos visto que es un tema delicado, que conviene tener claro. Se encarga del cacheado de entidades en Sesión de Usuario y Contexto de la aplicación. Tiene en cuenta la caducidad de la información, el refresco de los datos y el rendimiento del sistema (consumo de sus recursos).

Aplicación de patrones de Diseño

Definición del patrón de diseño. Además de una solución válida para problemas habituales son un medio de entendimiento que facilita la comunicación entre analista y desarrollador. Aceleran el desarrollo del software y facilitan el mantenimiento.

Tipos de aplicaciones web

Cualquier proyecto que se desarrolle en Internet conlleva una aplicación Web, ya sea compra on-line, información meteorológica o académica, simuladores de hipotecas, etc. Las páginas web se pueden clasificar siguiendo multitud de criterios, uno de ellos puede ser la forma de presentar su contenido. En función de esa característica se puede diferenciar:

- **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 Dinámica**. Conjunto de páginas cuyo contenido cambia según la ubicación de los visitantes, acciones pasadas realizadas en el sitio, zonas horarias y más.
- **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.

Requerimientos de un servidor web

Si pensamos en montar un servidor web en una red, debemos tener en cuenta:

- Prestaciones de Hardware del servidor.
- Sistema Operativo. Por ejemplo, Windows Server o Linux.
- Servidor Web: Por ejemplo, IIS o Apache.
- Configuración de la red:
    - IP para el servidor Web.
    - IP para las pruebas desde una máquina cliente.
    - Servidor DNS, que traduzca la IP del servidor a una dirección o nombre articulado. Aunque en una fase de desarrollo, podemos hacer nuestras pruebas con la IP directamente, sin DNS. Si vamos a plantearnos un proyecto de implantación y configuración de un servidor web, en la planificación de nuestro proyecto debemos tener en cuenta:
- Recursos del equipo servidor.
- Conectividad del equipo servidor.
- Servidor web empleado: El porqué de su elección y funcionamiento.
- Posibilidades del servidor web empleado.
- Requisitos de las aplicaciones web del proyecto.
- Entregables y fechas. La configuración de un servidor web dependerá:

- **Del tipo de contenido** de las páginas que debe ofrecer, si son estáticas o dinámicas, si se ofrece contenido seguro, si hay transacción de información, si tiene áreas de acceso restringido, etc. Por lo tanto, según las páginas web que se ofrezcan, el servidor deberá estar configurado para tal fin: con soporte PHP, con soporte de cifrado, con soporte de control de acceso, etc.
- Si el servidor puede albergar **varias páginas o sitios**, si permite hosts virtuales y como crece en el futuro. En el caso de que necesite más funcionalidades, es importante saber si el servidor web admite la carga de nuevos módulos o requiere que sea reinstalado de nuevo.
- **La cantidad o número de páginas web que debe servir**, así que no podemos perder de vista la escalabilidad y la estabilidad (cómo se comporta con varias peticiones simultáneas).
- Además, una vez instalado y configurado nuestro servidor web, debemos programar y **realizar pruebas de funcionamiento** antes de ponerlo al servicio.

Servidor de aplicaciones web

En general, podemos decir que un servidor de aplicaciones es un servidor que **proporciona o ejecuta aplicaciones en una red**. Normalmente se trata de proporcionar servicios de aplicación a los ordenadores clientes de la red. En realidad, el servidor de aplicaciones deriva de un sistema distribuido, que gestiona la mayor parte de las **funciones de la lógica de negocio y acceso a datos** en una aplicación. Los principales beneficios de la aplicación de la tecnología de servidores de aplicación son la centralización y la disminución de la complejidad en el desarrollo de aplicaciones. Tipos de servidores de aplicaciones:

- Servidores de aplicación Java EE.
- Servidores .NET de Microsoft.
- Servidores Apache+PHP+MySQL.

Plataformas web libres

En términos generales, una plataforma Web consta de cuatro componentes básicos:

- El **sistema operativo**, bajo el cual opera el equipo donde se hospedan las páginas Web y que representa la base misma del funcionamiento del computador. En ocasiones limita la elección de otros componentes.
- El **servidor Web**, es el software que maneja las peticiones desde equipos remotos a través de la Internet. En el caso de páginas estáticas, el servidor Web simplemente provee el archivo solicitado, el cual se muestra en el navegador. En el caso de sitios dinámicos, el servidor Web se encarga de pasar las solicitudes a otros programas que puedan gestionarlas adecuadamente.
- El **gestor de bases de datos** se encarga de almacenar sistemáticamente un conjunto de registros de datos relacionados para ser usados posteriormente.
- Un **lenguaje de programación** interpretado que controla las aplicaciones de software que corren en el sitio Web. Diferentes combinaciones de los cuatro componentes señalados, basadas en las distintas opciones de software disponibles en el mercado, dan lugar a numerosas plataformas Web, aunque, sin duda, hay dos que sobresalen del resto por su popularidad y difusión: LAMP y WISA. La **plataforma LAMP** trabaja enteramente con componentes de software libre y no está sujeta a restricciones propietarias. El nombre LAMP surge de las iniciales de los componentes de software que la integran:
- Linux: Sistema operativo.
- Apache: Servidor Web.

- MySQL: Gestor de bases de datos.
- PHP: Lenguaje interpretado PHP, aunque a veces se sustituye por Perl o Python. La **plataforma WISA** está basada en tecnologías desarrolladas por la compañía Microsoft; se trata, por lo tanto, de software propietario. La componen los siguientes elementos:
- Windows: Sistema operativo.
- Internet Information Services: servidor Web.
- SQL Server: gestor de bases de datos.
- ASP o ASP.NET: como lenguaje para scripting del lado del servidor. Existen otras plataformas, como por ejemplo la configuración Windows - Apache - MySQL - PHP que se conoce como WAMP. Es bastante común pero sólo como plataforma de desarrollo local.

Servidor Apache – Host Virtuales

El servidor web Apache que se ha configurado en clase tiene la misma dirección IP para todos nuestros proyectos y, por lo tanto, el mismo nombre DNS. Sin embargo, en realidad un servidor web sirve las páginas para distintas empresas o instituciones, cada una de ellas con su propio nombre de dominio. Si queremos tener distintas IP o distintos nombres en nuestro servidor Apache podemos hacerlo recurriendo a sitios web virtuales; así, tendremos en la misma máquina varios sitios funcionando gracias a un proceso que pasa desapercibido al usuario que visita nuestros sitios. Generalmente, los sitios Web virtuales pueden estar basados en IP, o basados en nombre.

- **Servidores basados en IP**, cada sitio Web tiene una dirección IP diferente. La IP que debemos poner en la definición de la directiva Virtualhost cambia, cada IP corresponde a una interfaz de red del servidor web. Este método no aporta grandes ventajas, es más, aún puede ser más difícil de mantener si las IP del servidor web se modifican con cierta frecuencia.
- **Servidores basados en nombres**, con una sola dirección IP están funcionando sitios Web con diferentes nombres (de dominio). El hecho de que estén funcionando en la misma máquina física pasa completamente desapercibido para el usuario que visita esos sitios Web. Es decir, asociamos a la IP del servidor los nombres de dominio que necesitamos.

Archivo de configuración A la hora de configurar los posibles host virtuales, se deberá modificar el archivo httpd-vhosts.conf de Apache. Dicho fichero contará con diferentes instrucciones, entre las que se destacan:

```
<VirtualHost IP_servidor:80>
    DocumentRoot /htdocs/empresa/
    ServerName www.empresa.com.
    ServerAlias empresa.com empresa.es www.empresa.es
 </VirtualHost>
```

- **IP_servidor:80** → es la IP de nuestro servidor con el puerto para http. Cuando el servidor tiene interfaz única, y lo usamos solo para nuestras fases de desarrollo, podemos sustituir el valor por *:80, es decir: <VirtualHost *:80>
- **DocumentRoot** es la ruta con la carpeta donde tenemos el proyecto.
- **ServerName** es el nombre de dominio escogido.
- **ServerAlias**, son todos los alias que aceptamos en el navegador para el #mismo dominio. La directiva <VirtualHost *:80> nos indica que, para nuestra IP, a través del puerto 80, somos capaces de atender o servir las páginas colocadas en el DocumentRoot, utilizando el nombre DNS especificado en

ServerName. En el caso de tener varias direcciones IP, entonces en la directiva VirtualHost, sustituiríamos el * por las IP para cada sitio. La primera opción corresponde a una configuración basada en nombre y la segunda a una configuración basada en IP.

Servidores de dominio basados en nombre

Es la forma más simple de configurar Hosts Virtuales. Solo necesitamos configurar nuestro servidor DNS para asignar la IP al nombre y configurar Apache para que reconozca los diferentes nombres. Debemos tener claro que tendremos designada una sola IP que nos servirá para atender todas las peticiones de los distintos sitios con nombres diferentes. Por eso usaremos el valor * para .

Servidor Nginx

Características principales de Nginx

- Nginx es un servidor web de alto rendimiento que se ha popularizado en los últimos años como alternativa a Apache. Sus principales características son:
- Ligero y rápido: consume menos memoria que Apache y gestiona miles de conexiones concurrentes con eficiencia.
- Reverse proxy: puede actuar como intermediario entre los clientes y las aplicaciones, mejorando seguridad y rendimiento.
- Balanceo de carga: distribuye peticiones entre varias instancias de una aplicación.
- Gestión de HTTPS: permite manejar certificados SSL/TLS para conexiones seguras.
- Servidor de contenidos estáticos: entrega HTML, CSS, JS e imágenes con gran velocidad.

Se utiliza habitualmente en entornos de producción modernos, a menudo combinado con Docker y frameworks como Node.js, Django o Spring Boot.

Ejemplo básico de configuración

Archivo default.conf para redirigir tráfico al backend:

```
server {
    listen 80;
```

```
    server_name midominio.com;

    location / {
        proxy_pass http://127.0.0.1:3000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }
}
```

En este ejemplo:

- Nginx escucha en el puerto 80.
- Todas las peticiones se redirigen a una aplicación en el puerto 3000 (por ejemplo, un backend Node.js).
- Se añaden cabeceras para preservar información del cliente.

Virtualización y contenedores con Docker

Imágenes y contenedores

- Imagen: plantilla inmutable que contiene el sistema operativo mínimo y la aplicación.
- Contenedor: instancia en ejecución de una imagen.
- Docker Hub: repositorio público de imágenes oficiales (ej: nginx, mysql, node).

#️⃣Una imagen puede considerarse como la “receta” y un contenedor como el “plato ya servido”.

Dockerfile

Archivo de texto que contiene las instrucciones para construir una imagen personalizada.

Ejemplo para una aplicación **Node.js**:

```
FROM node:18
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
EXPOSE 3000
CMD ["npm", "start"]


Con este archivo:

docker build -t mi-app .
docker run -p 3000:3000 mi-app
```

Docker Compose

Herramienta que permite levantar varios contenedores con un único archivo YAML.

Ejemplo: aplicación Node.js + base de datos MySQL:

```
version: '3'
services:
  app:
    build: .
    ports:
      - "3000:3000"
    depends_on:
      - db

  db:
    image: mysql:8
    environment:
      MYSQL_ROOT_PASSWORD: root
      MYSQL_DATABASE: miapp
    ports:
      - "3306:3306"
```

Con docker-compose up, se lanzan ambos servicios automáticamente.

Integración Docker + Nginx

Nginx como proxy reverso

En entornos modernos, se suele colocar Nginx en un contenedor que actúa como proxy reverso frente a una aplicación backend (por ejemplo, Node.js). Esto aporta:

- Seguridad (oculta la aplicación tras Nginx).
- Escalabilidad (balancea tráfico entre varias instancias).
- Gestión de HTTPS.

Ejemplo con Docker Compose

Archivo docker-compose.yml que combina Nginx + app:

```
version: '3'
services:
  nginx:
    image: nginx:latest
    volumes:
      - ./nginx.conf:/etc/nginx/conf.d/default.conf
    ports:
      - "80:80"
    depends_on:
      - app

  app:
    build: .
```

```
    ports:
      - "3000:3000"
```

Con esta configuración:

Los usuarios acceden a http://localhost.

Nginx escucha en el puerto 80.

Redirige las peticiones al servicio app que escucha en el puerto 3000.

Ventajas de la integración

- Portabilidad: funciona igual en local, servidores o la nube.
- Escalabilidad: Nginx puede repartir la carga entre múltiples instancias.
- Simplicidad: un único archivo (docker-compose.yml) permite levantar toda la aplicación.
- Buenas prácticas: alinea el despliegue con los estándares de la industria.