UNIDAD 1: IMPLANTACIÓN DE APLICACIONES WEB

1. Arquitecturas Web. Aspectos generales.

Una **aplicación web** consiste de componentes web; recursos estáticos como imágenes y clases de ayuda y librerías. El contenedor web nos provee servicios para mejorar las capacidades de nuestros componentes web y hacer más fácil del desarrollo.

El proceso para desarrollar, empaquetar y ejecutar una aplicación web se puede resumir en:

1. Desarrollar el código de los componentes web.
2. Desarrollar un descriptor de despliegue (si fuera necesario).
3. Compilar los componentes de la aplicación web y clases de ayuda referenciadas por estos componentes.
4. Empaquetar la aplicación en una unidad desplegable (opcional).
5. Desplegar la aplicación en un contenedor web.
6. Acceder a la url que referencia la aplicación web.

APLICACIONES WEB VS. APLICACIONES DE ESCRITORIO

- **No requiere instalar software especial (en los clientes).** En esencia, para acceder a un software web solo necesitamos disponer de un navegador de páginas web (Internet Explorer, Firefox, Opera, Chrome, etc.), los cuales suelen venir con el propio sistema operativo. No es necesario tener nada más. Debido a la arquitectura de las aplicaciones web, el navegador suele quedar relegado a mostrar la interfaz de usuario (menús, opciones, formularios, etc.), mientras que toda la compleja lógica de negocio se lleva en el lado del servidor.
- **Bajo coste en actualizar los equipos con una nueva versión.** Los navegadores web visualizan las páginas web que son servidas por el servidor web dinámicamente. En ese sentido, es el servidor quien ejecuta la mayor parte del código de la aplicación y suministra de forma centralizada las vistas (las páginas) a los navegadores conectados. En consecuencia, no hay que instalar nada en los puestos de trabajo, ya que la actualización se realiza en el servidor y automáticamente la ven todos los usuarios.
- **Acceso a la última y mejor versión.** Como consecuencia del punto anterior, se evita que pueda existir algún equipo que ejecute una versión diferente y desactualizada. Si existen ordenadores con distintas versiones del programa se pueden originar problemas de consistencia en la información o pérdida de funcionalidad.
- **Información centralizada.** En una aplicación web, no solamente la lógica de negocio está centralizada en el servidor, sino también los datos que se ubican en una base de datos centralizada (en ese servidor u otro destinado a tal fin). La centralización tiene la ventaja de facilitar el acceso a la misma.

- **Seguridad y copias de seguridad.** Este es un corolario del punto anterior, es decir, una consecuencia. Como disponemos de los datos centralizados es más fácil establecer y llevar el control de una política de copias de seguridad centralizada. Es más, al no ubicarse los datos en el puesto de trabajo, en caso de robo o incendio, la empresa no ha perdido información y puede desplegar rápidamente un nuevo puesto de trabajo (PC con un navegador web).
- **Movilidad.** Este es un concepto relativo y dependiente de la implantación concreta. Si el software está ubicado en un servidor web en Internet o bien disponemos de una intranet externalizada (extranet), cualquier usuario con un portátil y una conexión a Internet móvil podría acceder a la aplicación.
- **Reducción de costes en los puestos cliente (mayor longevidad).** Debido a que las páginas se ofrecen desde el servidor web (donde se suelen ejecutar la mayoría de los procesos y la lógica de negocio), el equipo cliente queda relegado a mostrar los resultados y formularios, para lo cual no es necesario un hardware potente en los puestos de trabajo, lo que se traduce en reducción de costes y una mayor longevidad en el uso de los mismos (no hay que cambiar el hardware de los puestos porque ahora se requieran operaciones más complejas).

Sin embargo no todo son ventajas. Debemos recordar que en el mundo real de los requisitos y restricciones no existe la solución perfecta, sino la más o menos adecuada al caso planteado. Por ello, una solución web también tiene sus inconvenientes, unos derivados del modelo web y otros como consecuencia de cómo se implante.

ASPECTOS GENERALES

Existen principalmente dos tipos de aplicaciones web:

- **Orientadas a la presentación:** Una aplicación web orientada a la presentación comunmente genera páginas web interactivas usando varios lenguajes de marcado (HTML, XHTML, XML y demás) al igual que contenido dinámico para cada solicitud hecha por un cliente.
- **Orientadas al servicio:** Una aplicación web orientada al servicio implementa un endpoint para un servicio web. Las aplicaciones orientadas a la presentación son generalmente clientes de las aplicaciones orientadas al servicio.

Las aplicaciones web tienen las siguientes características:

- Se ejecutan en un servidor (físico o virtual), no en el dispositivo del usuario.
- Pueden atender a miles de usuarios simultáneos, no a uno sólo.
- Es muy habitual que se necesiten varios servidores para una única aplicación web (por escalabilidad y tolerancia a fallos).
- Están formadas por código y por recursos (imágenes, documentos html, css, js, ficheros estáticos …)
- Utilizan servicios adicionales: base de datos, servidor de correo, servidor de vídeo …
- Requieren de un proceso de instalación y configuración (despliegue) en el servidor o servidores

Normalmente un aplicación web se suele alojar en una empresa de alojamiento o hosting. Que son empresas que:

- Permiten a sus clientes ejecutar sus aplicaciones web en sus instalaciones.
- Se encargan del suministro eléctrico, conexión a Internet, compra y mantenimiento de servidores, control de acceso físico, etc…
- Existe una amplia variedad de servicios que dependen del proveedor y de las necesidades del cliente.

Los tipos de alojamiento de estos hosting se suelen clasificar en alojamiento:

- **Compartido:** Varias aplicaciones web conviven en la misma máquina física, El desarrollador no tiene control total sobre el servidor.
- **Dedicado:** El cliente dispone de una máquina física que controla completamente.
- **Virtual:** el cliente dispone de una máquina virtual que controla completamente y se ejecuta en un servidor físico compartido con otras máquinas virtuales.

Existen muchos proveedores con diferentes servicios (compartido, dedicado y virtual)

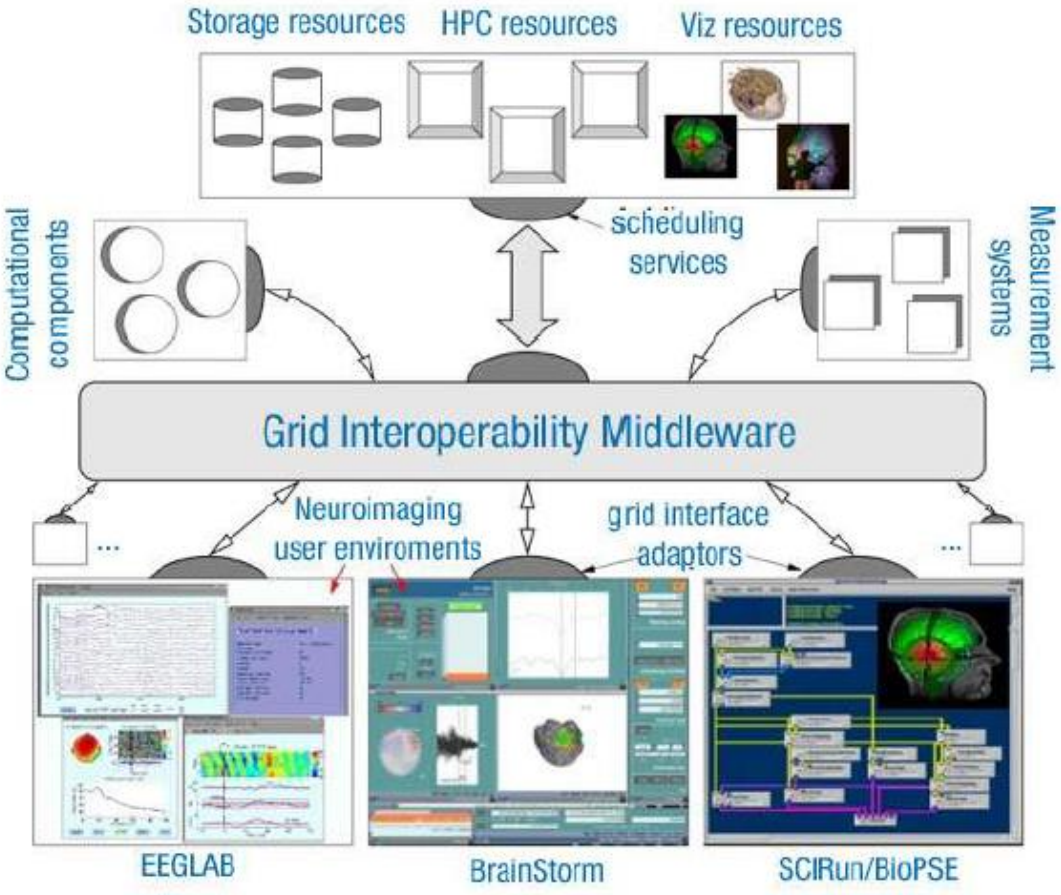
Algunos de los más conocidos en España son:

ESCALABILIDAD

Las aplicaciones web se ejecutan en un entorno donde el número de clientes que solicitan el servicio puede variar en gran medida en función del momento. Es por ello que hay una característica de esencial importancia como es la escalabilidad, al que Juan ha dedicado un apartado de su wiki para documentar esta característica.
En el entorno en que se ubican las aplicaciones web, uno de los principales factores que puede afectar al rendimiento de las mismas es el número de usuarios, ya que éste puede verse incrementado de forma vertiginosa en un periodo de tiempo relativamente corto. El éxito o el fracaso de un sitio web orientado al usuario común vendrá determinado, entre otros aspectos, por el dimensionamiento del sistema sobre el que se instala y soporta el software que sustenta dicho sitio. En consecuencia, uno de los requisitos fundamentales de una aplicación web es que sea completamente escalable sin que un aumento de los recursos dedicados a la misma suponga modificación alguna en su comportamiento o capacidades.

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.

2. MODELOS.

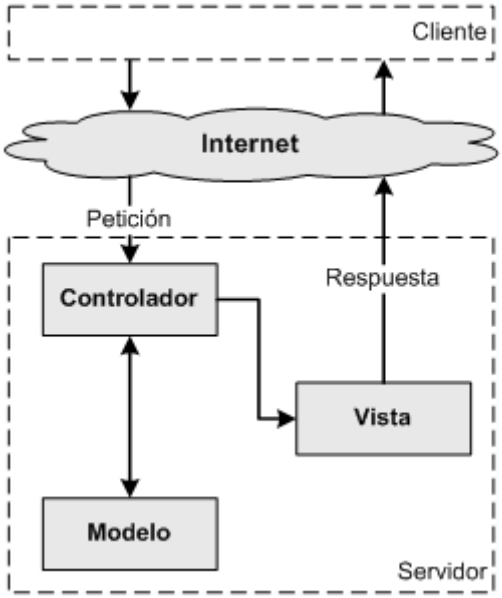
MODELO VISTA CONTROLADOR

El patrón **Modelo – Vista – Controlador** fue inventado en el contexto de Smalltak para realizar una separación entre la interfaz gráfica y el código del funcionamiento de una aplicación. Esta idea teórica afectó, de forma importante, a gran parte del código de Smalltalk y fue posteriormente aplicada a los lenguajes que están basados en objetos.

En el paradigma **MVC**, las entradas del usuario, los modelos del mundo exterior y la retroalimentación visual son explícitamente separados y manejados por tres tipos de objetos, cada uno especializado para un conjunto de tareas específicas.

El objetivo primordial del patrón es dar soporte a los modelos funcionales y mapas mentales de la información relevante para los usuarios, permitiendo un modelo que facilite la consulta y manejo de los mismos. La única manera de construir artefactos manejables es ayudar al usuario a construir modelos del sistema. Pero esto es imposible si el modelo mental no ha

sido diseñado dentro del artefacto desde el principio. Intentar adicionar los modelos mentales del usuario cuando ya se ha avanzado en el desarrollo puede ser imposible. A continuación un gráfico que resume el patrón



Ventajas y desventajas del uso del patrón

Se tienen muchas ventajas como:

La implementación se realiza de forma modular.
Sus vistas muestran información actualizada siempre. El programador no debe preocuparse de solicitar que las vistas se actualicen, ya que este proceso es realizado automáticamente por el modelo de la aplicación.
Cualquier modificación que afecte al dominio, como aumentar métodos o datos contenidos, implica una modificación sólo en el modelo y las interfaces del mismo con las vistas, no todo el mecanismo de comunicación y de actualización entre modelos.
Las modificaciones a las vistas no afectan al modelo de dominio, simplemente se modifica la representación de la información, no su tratamiento.
MVC está demostrando ser un patrón de diseño bien elaborado pues las aplicaciones que lo implementan presentan una extensibilidad y una mantenibilidad únicas comparadas con otras aplicaciones basadas en otros patrones.
Como desventajas tenemos:

Para desarrollar una aplicación bajo el patrón de diseño MVC es necesario una mayor dedicación en los tiempos iniciales del desarrollo. Normalmente el patrón exige al programador desarrollar un mayor número de clases que, en otros entornos de desarrollo, no son necesarias. Sin embargo, esta desventaja es muy relativa ya que posteriormente, en la etapa de mantenimiento de la aplicación, una aplicación MVC es mucho más mantenible, extensible y modificable que una aplicación que no lo implementa.
MVC requiere la existencia de una arquitectura inicial sobre la que se deben construir clases e interfaces para modificar y comunicar los módulos de una aplicación. Esta arquitectura inicial debe incluir, por lo menos, un mecanismo de eventos para poder proporcionar las

notificaciones que genera el modelo de aplicación; una clase Modelo, otra clase Vista y una clase Controlador genéricas que realicen todas las tareas de comunicación, notificación y actualización que serán luego transparentes para el desarrollo de la aplicación.
MVC es un patrón de diseño orientado a objetos por lo que su implementación es sumamente costosa y difícil en lenguajes que no siguen este paradigma.

COMPUTACIÓN EN LA NUBE

La computación en la nube (Cloud computing) es un concepto de marketing más que un concepto técnico.

Cuando los proveedores utilizan la palabra cloud se refieren a la posibilidad de configurar y redimensionar los recursos que se usan de forma rápida y sencilla, o manualmente vía web o usando **APIs REST**.

Dado que son tan dinámicos, se suele cobrar por tiempo de uso de los recursos (horas o minutos), sin tener que hacer un contrato previo con un tiempo de permanencia determinado.

Los recursos de computación en la nube suelen estar virtualizados, aunque en algunas ocasiones pueden ser máquinas físicas.

Los proveedores de cloud computing ofrecen diversos tipos de servicios, tanto de bajo nivel como de alto nivel.

- Servidores virtuales (instancias).
- Gestión del sistema operativo que tendrán los servidores (imagen)
- Sistema de copias de seguridad de los servidores completos.
- Balanceadores de carga entre servidores.
- Base de datos administradas.
- Servicios de gestión de logs, monitorización, alarmas…
- Plataforma auto-escalable para ejecución de aplicaciones.

La diferencia con los alojamientos tradicionales consiste en la elasticidad de los recursos.
Desde una consola web o una app móvil se pueden activar o desactivar recursos (servidores, sistemas operativos, copias de seguridad).

También se pueden gestionar de forma automática con APIs REST o librerías en lenguajes de programación.

El software es verdaderamente escalable, Si necesita más recursos hardware, los puede conseguir de forma automática.

Los proveedores más conocidos

Los servicios ofrecidos por los proveedores pueden ser de diferentes niveles de abstracción:

- Desarrolladores.
  - Infraestructura como servicio (bajo nivel)
  - Plataforma como servicio (nivel medio)
- Usuario final
  - Software como servicio (alto nivel)

INFRAESTRUCTURA COMO SERVICIO (IaaS)

Infraestructure as a Service (IaaS)

Servicios:

- Servidores (intances)
- Balanceadores de carga (load balancer)
- Gestión de sistemas operativos (images)
- Copias de seguridad de servidores
- Almacenamiento de datos
- Direcciones IP
- Servidores DNS

PLATAFORMA COMO SERVICIO (PaaS)

Platform as a Services (PaaS).

Plataforma para el despliegue de aplicaciones web:

- Plataforma para el despliegue de aplicaciones web:
  - La plataforma está diseñada para ser escalable de forma automática (sin intervención del desarrollador/administrador)
  - Dependiendo del proveedor existen plataformas para las tecnologías mas usadas: Java, PHP, Ruby, .NET ...
  - La plataforma ofrece servicios adicionales como un servicio: Bases de datos, servidor de correo, bus de comunicaciones, etc.

SOFTWARE COMO SERVICIO (SaaS)

Software as a Servicie (SaaS)
- El software se ofrece como un servicio por Internet (vía web)
- El usuario no instala el software , lo usa vía web.
- Generalmente se paga por uso o por número de usuarios.
- Normalmente están destinados al usuario final, aunque algunos servicios están orientaddos a desarrolladores.

Ejemplos: Dropbox, Gmail, GoogleDocs, Microsoft 365, iCloud, flickr

3. Plataformas Web libres y propietarias.

Una plataforma web es el entorno de desarrollo de software empleado para diseñar y ejecutar un sitio web. En términos generales, una plataforma web consta de cuatro componentes básicos:

1. 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.

2. 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.

3. El gestor de bases de datos se encarga de almacenar sistemáticamente un conjunto de registros de datos relacionados para ser usados posteriormente.

4. 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:

- **L**inux: Sistema operativo.
- **A**pache: Servidor web.
- **M**ySQL: Gestor de bases de datos.
- **P**HP: 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:

- **W**indows: Sistema operativo.
- **I**nternet Information Services: servidor web.
- **S**QL Server: gestor de bases de datos.
- **A**SP 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. De forma similar, un servidor Windows puede correr con MySQL y PHP. A esta configuración se la conoce como plataforma WIMP. Existen muchas otras plataformas que trabajan con distintos sistemas operativos (Unix, MacOS, Solaris), servidores web (incluyendo algunos que se han cobrado relativa popularidad como Lighttpd yLiteSpeed), bases de datos (Postgre SQL) y lenguajes de programación.

3. Servidores Web y de aplicaciones.

Se define una aplicación web como una aplicación informática que se ejecuta en un entorno web, de forma que se trata de una aplicación cliente-servidor junto con un protocolo de comunicación previamente establecido:

- ✓ Cliente: navegador.
- ✓ Servidor: servidor web
- ✓ Comunicación: protocolo HTTP

Un **servidor de aplicaciones** es un software que proporciona aplicaciones a los equipos o dispositivos cliente, por lo general, a través de Internet y utilizando el protocolo http. Los servidores de aplicación se distinguen de los servidores web en el uso extensivo del contenido dinámico y por su frecuente integración con bases de datos.
Un servidor de aplicaciones también es una máquina en una red de computadores que ejecuta determinadas aplicaciones, gestionando la mayor parte de las funciones de acceso a los datos de la aplicación.

Las principales ventajas de la tecnología de los servidores de aplicaciones es la **centralización** y **disminución** de la complejidad en el desarrollo de las aplicaciones, ya que no necesitan ser programadas, sino que son ensambladas desde bloques provistos por el servidor de aplicación.
Otra de las ventajas es la **integridad de datos y código** ya que, al estar centralizada en una o un pequeño número de máquinas servidoras, las actualizaciones están garantizadas para todos los usuarios.

El término servidor de aplicaciones se aplica a todas las plataformas. Dicho término se utiliza para referirse a los servidores de aplicaciones basadas en web, como el control de las

plataformas de comercio electrónico integrado, sistemas de gestión de contenido de sitios web y asistentes o constructores de sitios de Internet.
Uno de los ejemplos destacados es el de Sun Microsystems, la plataforma J2EE. Los servidores de aplicaciones Java se basan en la Plataforma Java ™ 2 Enterprise Edition (J2EE ™). J2EE utiliza un modelo de este tipo y en general, incluye un nivel Cliente, un nivel Medio, y un EIS. El servidor de tipo Cliente puede contener una o más aplicaciones o navegadores. La Plataforma J2EE es del Nivel Medio y consiste en un servidor web y un servidor EJB. (Estos servidores son también llamados "contenedores".)
También podría haber subniveles adicionales en el nivel intermedio. El nivel del SistemaEnterprise Information System (EIS, o "Sistema de Información Empresarial") contiene las aplicaciones existentes, archivos y bases de datos.

4. Estructura y recursos que componen una aplicación Web. Descriptor de despliegue.

Una aplicación web está compuesta de una serie de servlets, páginas jsp, ficheros html, ficheros de imágenes, ficheros de sonidos, texto, clases, etc.; de forma que todos estos recursos se pueden empaquetar y ejecutar en varios contenedores distintos.

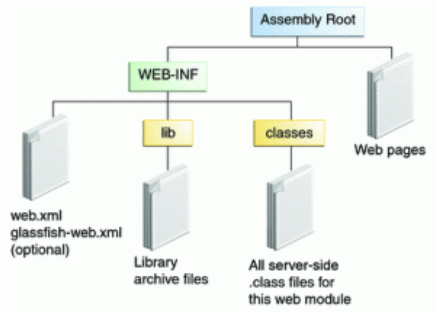
Un servlets es una aplicación java encargada de realizar un servicio específico dentro de un servidor web. La especificación Servlet 2.2 define la estructura de directorios para los ficheros de una aplicación web. El directorio raíz debería tener el nombre de la aplicación y define la raíz de documentos para la aplicación web. Todos los ficheros debajo de esta raíz pueden servirse al cliente excepto aquellos ficheros que están bajo los directorios especiales META-INF y WEB-INF en el directorio raíz.
Todos los ficheros privados, al igual que los ficheros class de los servlets, deberían almacenarse bajo el directorio WEB-INF

Durante la etapa de desarrollo de una aplicación web se emplea la estructura de directorios, a pesar
de que luego en la etapa de producción, toda la estructura de la aplicación se empaqueta en un archivo
`.war`

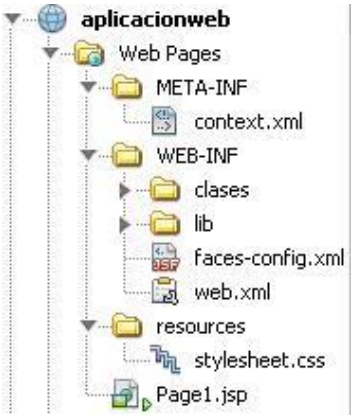
El código necesario para ejecutar correctamente una aplicación web se encuentra distribuido en una
estructura de directorios, agrupándose ficheros según su funcionalidad. Un ejemplo de la estructura
de carpetas de una aplicación web puede ser el siguiente:

```
/index.jsp
/WebContent/jsp/welcome.jsp
/WebContent/css/estilo.css
/WebContent/js/utils.js
/WebContent/img/welcome.jpg
/WEB-INF/web.xml
/WEB-INF/struts-config.xml
/WEB-INF/lib/struts.jar
/WEB-INF/src/com/empresa/proyecto/action/welcomeAction.java
/WEB-INF/classess/com/empresa/proyecto/action/welcomeAction.class
```

De forma genérica podríamos decir que una aplicación web se estructura en tres capas:
1. Navegador web.
2. Tecnología web dinámica (PHP, Java Servlets, ASP, etc.)
3. Base de datos encargada de almacenar de forma permanente y actualizada la información que la aplicación web necesita.



Estructura de una aplicación Java EE 6



Despliegue de aplicaciones

Una aplicación web puede ser desplegada empleando uno de los siguientes métodos:

- Por medio de archivos WAR (Web Archive).

- Editando los archivos web.xml y server.xml. Este método es el que se pasa a tratar a continuación.

Los directorios que forman una aplicación compilada suelen ser : www, bin, src, tomcat y gwt-cache.

La carpeta www contiene, a su vez, una carpeta con el nombre y ruta del proyecto que contiene los ficheros que forman la interfaz (html, js, css, etc.). La carpeta bin contiene las clases de java de la aplicación.

Ejemplo despliegue

Para desplegar la aplicación en Tomcat:

1. Copiar la carpeta contenida en www (con el nombre del proyecto) en el directorio webapps de Tomcat.

2. Renombrar la nueva carpeta así creada en Tomcat con un nombre más sencillo. Esa será la carpeta de la aplicación en Tomcat.

3. Crear, dentro de dicha carpeta, otra nueva, y darle el nombre WEB-INF (respetando las mayúsculas).

4. Crear, dentro de WEB-INF, otros dos subdirectorios, llamados lib y classes.

5. Copiar en lib todas las librerías (.jar) que necesite la aplicación para su funcionamiento.

6. Copiar el contenido de la carpeta bin de la aplicación en el subdirectorio WEB-INF/classes de Tomcat.

7. Crear en WEB-INF un fichero de texto llamado web.xml, con las rutas de los servlets utilizados en la aplicación.

8. A la aplicación ya puede accederse en el servidor, poniendo en el navegador la ruta del fichero html de entrada, que estará ubicado en la carpeta de la aplicación en Tomcat.

Descriptor de despliegue

Un Descriptor de Despliegue es un documento XML que describe las características de despliegue de una aplicación, un módulo o un componente. Por esto, la información del descriptor de despliegue es declarativa, y esta puede ser cambiada sin la necesidad de modificar el código fuente.

Cualquier aplicación web tiene que aportar un descriptor de despliegue situado en WEB-INF/web.xml; en el caso concreto de Tomcat el descriptor /conf/web.xml es un descriptor por defecto que se ejecuta siempre antes del descriptor de la aplicación y, solamente, debería contener información general y no específica de la aplicación. Un ejemplo de descriptor de despliegue puede ser el siguiente archivo web.xml:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app PUBLIC
  "-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN"
  "http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">
<web-app>
  <!-- Tus definiciones van aquí -->
</web-app>
```

Situadas entre las etiquetas y / estarían los descriptores de despliegue de servlets, los cuales deben contener las siguientes etiquetas en el siguiente orden:

```
<servlet>
  <servlet-name>nombre</servlet-name>
  <servlet-class>package.nombre.MiClass</servlet-class>
 </servlet>
```

Para probar el servlet, una vez arrancado el servidor Tomcat, abrimos un navegador web, en el cual escribiríamos una URL con el siguiente formato:

http://{address}:{port}/{servletName}

por ejemplo:

http://localhost:8080/Servlet_de_prueba