RESUMEN TEMA 01

# IMPLANTACIÓN DE ARQUITECTURAS WEB

Se explican los aspectos generales de las arquitecturas web, entre otros: tecnologías, tipos, modelos, etc.

Se analizan las principales características del servidor web Apache, su instalación y configuración básica; posteriormente se realiza el estudio comparativo entre servidor web y servidor de aplicaciones web, explicando el trabajo básico de instalación y puesta en funcionamiento del servidor de aplicaciones Tomcat.

Se aborda el estudio de la estructura y despliegue de aplicaciones web, en donde se introduce el concepto de archivos WAR y descriptor de despliegue.

# 1.- Aspectos generales de arquitecturas web.

La arquitectura World Wide Web (WWW) de Internet provee un modelo de programación sumamente poderoso y flexible.

Las aplicaciones y los contenidos son presentados en formatos de datos estándar y son localizados por web browsers, que envían requerimientos al servidor y éste los responde.

Los estándares WWW especifican muchos de los mecanismos necesarios para construir un ambiente de aplicación de propósito general:

* **Modelo estándar de nombres:** se denominan según un Localizador Uniforme de Recursos, de servidores y contenido web (Uniform Resource Locator: URL).
* **Contenido:** a todos los contenidos se les especifica un determinado tipo permitiendo ser localizados por los navegadores correctamente.
* **Formatos de contenidos estándar:** todos los navegadores soportan un conjunto de formatos estándar.
* **Protocolos estándar:** cualquier navegador puede comunicarse con cualquier servidor web, vía HTML y el protocolo TCP/IP.

Cada organismo de estandarización tiene un enfoque diferente y define estándares para diferentes tecnologías o aspectos de la web y las tecnologías relacionadas, por ejemplo:

* **Word Wide Web Consortium (W3C):** 
  + **HTML**: define los estándares para HyperText Markup Language, que se utiliza para crear la estructura de páginas web.
  + **CSS**: especifica los estándares para Cascading Style Sheets, que se usan para el diseño y la presentación visual de páginas web.
  + **XML**: responsable de eXtensible Markup Language, que se utiliza para estructurar datos de manera jerárquica y es fundamental en la interoperabilidad de datos en la web.
  + **SOAP**: define el protocolo SOAP que describe cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML.
* **WHATWG (Web Hypertext Application Technology Working Group):** trabaja en la evolución y Desarrollo de las especificaciones de HTML, junto con el W3C.
* **Internet Engineerging Task Force (IETF):**
  + **HTTP**: define el protocolo Hypertext Tranfer Protocol, que se utiliza para la comunicación entre web y navegadores.
  + **SMTP**: especifica SMTP (Simple Mail Transfer Protocol) para el envió de correos electrónicos a través de Internet.
  + **TCP/IP:** trabaja en los estándares de TCP/IP, forman la base de la comunicación de Internet.
* **Internet Corporation for Assigned Names and Numbers (ICANN):** es responsable de la gestión de asignación de nombres de dominio y direcciones IP en Internet. Su trabajo es fundamental para el funcionamiento de la web al garantizar que los nombres de dominio sean únicos y se resuelvan adecuadamente en direcciones IP.
* **Ecma International:** define el estándar ECMAScript, que es la base de JavaScript.
* **ISO (Organización Internacional de Normalización):** no se enfoca exclusivamente en tecnologías web pero tiene una amplia gama de estándares relacionados con la tecnología de la información y las comunicaciones.
* **ITU (Unión Internacional de Telecomunicaciones):** se centra en estándares para las telecomunicaciones y las redes de comunicación.

Estos **organismos** a veces trabajan en colaboración en el desarrollo de estándares. Sus esfuerzos colectivos contribuyen a la construcción y la evolución de la web y las tecnologías asociadas.

Esta **infraestructura** permite a los usuarios acceder a una gran cantidad de aplicaciones y servicios de terceros, permite a los desarrolladores crear aplicaciones y servicios para una gran comunidad de cliente.

**Aspectos** generales en una arquitectura web:

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

Tres elementos fundamentales en el funcionamiento de los servicios web:

* **Proveedor de servicio web,** quien lo diseña, desarrolla e implementa y lo pone disponible para su uso público o privado.
* **Consumidor del servicio**, quien accede al componente para utilizar los servicios que éste presta.
* **Agente del servicio**, que sirve como enlace entre proveedor y consumidor para efectos de publicación, búsqueda y localización del servicio.

## 1.1.- Evolución de los servicios web.

La evolución del uso del Servicios web en las organizaciones está fuertemente ligada al desarrollo de Internet como red prestadora de servicios.

Factores que han impulsado el uso de los servicios web:

* **El contenido se está volviendo más dinámico**, contenidos instantáneos y un servidor web debe ser capaz de combinar contenido de fuentes muy diferentes.
* **El ancho de banda es menos costoso**, un servicio web puede entregar tipos variables de contenidos como vídeo o audio. A medida que crezca el ancho de banda, los servicios web deban adaptarse a nuevos tipos de contenidos.
* **El almacenamiento es más barato,** un servicio web debe ser capaz de manejar cantidades masivas de datos y de forma inteligente.
* **El éxito de la computación extendida**, con cientos de millones de dispositivos electrónicos, los servicios web deben servir a todo tipo de dispositivos, plataformas y navegadores, entregando contenido sobre una amplia variedad de tipos de conexión.

Cuando se empiezan a utilizar servicios web en una organización, estos se desarrollan e implementan como servicios simples, que poco a poco se van integrando hasta llegar a servicios web mucho más complejos.

Breve descripción de las principales fases de la evolución de la web (WWW):

* Web 1.0 (Década de 1990): se caracterizó por ser estática y de solo lectura.
* Web 2.0 (Principios de la década de 2000): interacción y la colaboración en línea, permitiendo a los usuarios crear, colaborar y compartir en línea.
* Web 3.0 (años 2010 en adelante): web semántica se centra en la IA, búsqueda semántica, recomendación de contenido y automatización de tareas.
* Web 4.0 (Futuro): fase en desarrollo, pero centrada en aun más en la IA, IoT, la salud, la movilidad.

Estas fases representan una evolución continua de la web, con un enfoque cada vez mayor en la interacción, la colaboración y la inteligencia artificial. La web está en constante cambio.

## 1.2.- Tecnologías asociadas a las aplicaciones web.

Las aplicaciones web emplean páginas dinámicas, éstas se ejecutan en un servidor web y se muestran en el navegador de equipo cliente que es el que ha realizado previamente la solicitud.

Cuando una página web llega al navegador, es posible que también incluya algún programa o fragmento de código que se deba ejecutar.

Tecnologías asociadas a las aplicaciones web tanto del lado servidor como cliente:

* **ASP (Active Server Pages):** Páginas activas se ejecutan del lado del servidor que se mostrarán en el navegador de cada equipo cliente. Versiones tanto para Unix, Linux como Windows.
* **CGI (Cascading Style Sheets):** se usan para formatear las páginas web; se trata de separar el contenido de un documento, de su presentación.
* **JAVA:** trabaja en el cliente, es decir, se ejecuta en el navegador del equipo cliente y no en el servidor. Es eficiente y muy poderoso y se caracteriza por:
  + Una misma aplicación puede funcionar en diversos tipos de ordenadores, dispositivos y sistemas operativos.
  + Los programas Java pueden ser aplicaciones independientes o applets.
  + Lenguaje orientado a objetos, es decir, los programas se construyen a partir de módulos independientes y estos módulos se pueden transformar o ampliar fácilmente.
  + De software libre bajo la licencia GNU GPL (Creada y desarrollada por Sun Microsystems).
* **JavaScript:** lenguaje que se interpreta y se ejecuta en el cliente. Da más dinamismo a las páginas web.
* **PHP (Hypertext Preprocessor):** es parecido a ASP, ejecutado en el lado del servidor. PHP es muy eficiente, acceso a base de datos como MYSQL y crear páginas dinámicas complejas.
* **VBScript (Visual Basic Scripting):** la respuesta de Microsoft a JavaScript. Es una buena herramienta para cualquier sitio destinado a ser mostrado exclusivamente en el navegador Microsoft Internet Explorer. Se puede ejecutar tanto en el lado servidor como cliente.

## 1.3.- Tipos de aplicaciones web.

Una aplicación web es una plataforma orientada a automatizar los procesos de servicios que se quieran ofrecer a usuarios.

Clasificación según su función y su contenido:

* **Página web Estática:** implementa HTML, imágenes, vídeos, etc.
* **Página Animada:** tecnología descontinuada y desaconsejada ya que los navegadores la bloquean hacen uso de tecnología de animación permitiendo diseños vanguardistas, modernos y creativos.
* **Página web Dinámica:** los lenguajes más destacados son PHP y ASP, permiten una perfecta estructuración de contenido. Por un lado, crearíamos la estructura y por otra almacenaríamos el contenido en base de datos. A partir de ahí, crearíamos el código de llamada, que insertaría el contenido en la propia página web estructurada. Y así se desarrollan aplicaciones para poder gestionar el contenido a través de un panel de control.
* **Portal**: sitio web que en su página principal permite el acceso a múltiples secciones que, por lo general, son foros, chats, buscadores, noticias, etc.
* **Página web con Gestor de Contenidos o CRM:** sitio web cuyo contenido se actualiza a través de un panel de gestión por parte del administrador del sitio.

Webs dependiendo del área al que están dirigidas:

* **Aplicaciones web de consumo:** redes sociales, motores de búsqueda, comercio electrónico, contenido multimedia.
* **Aplicaciones web empresariales:** gestión de relaciones con el cliente (CRM), Salesforce. Gestión de proyectos (Trello), Herramientas de colaboración (Slack, Teams), software de recursos humanos (HR), Workday.
* **Aplicaciones web de productividad:** procesadores de Texto en línea, hojas de cálculo en línea, aplicaciones de notas y tareas.
* **Aplicaciones web de entretenimiento y juegos:** juegos en línea, plataformas de streaming (Twitch).
* **Aplicaciones web educativas y de formación:** plataformas de aprendizaje en línea (LMS) Moodle, Blackboard. Tutoriales y cursos en línea (Udemy, Coursera, etc).
* **Aplicaciones web de noticias y medios:** portales de noticias, blogs y foros.
* **Aplicaciones web de salud y bienestar:** plataformas de citas médicas, aplicaciones de seguimiento de salud.
* **Aplicaciones web de viajes y reservas:** reserva de vuelos y hoteles, planificación de rutas.

## 1.4.- Arquitecturas web. Modelos.

Existen varios modelos de arquitectura que se utilizan para estructurar y organizar el código y los componentes de una aplicación y los más comunes son:

* **Modelo Vista Controlador (MVC):** es uno de los modelos más populares. Divide la aplicación en tres componentes principales.
  + Modelo, representa los datos y la lógica de la aplicación.
  + Vista, se encarga de la presentación de la información al usuario.
  + Controlador, controla la interacción del usuario y la lógica de flujo de la aplicación.
  + Separa las responsabilidades y reutiliza el código.
* **Modelo Vista Modelo de Vista (MVVM):** es un modelo de arquitectura similar al MVC, pero se utiliza especialmente en aplicaciones de interfaz de usuario ricas y con enfoque en la interacción del usuario.
  + Lo divide en tres partes, Modelo, Vista y Modelo de Vista (actúa como intermediario entre el modelo y la vista, proporcionando una representación de los datos para su visualización en la vista).
* **Modelo Vista Presentador (MVP):**
  + Es otro modelo que separa la lógica de presentación de la vista en una aplicación web.
  + Tres partes: modelo, vista y presentador (actúa como intermediario entre el modelo y la vista, gestionando la lógica de la interfaz de usuario).
* **Arquitectura de Microservicios:**
  + Divide una aplicación en pequeños servicios independientes y autónomos.
  + Cada microservicio se enfoca en una función o característica específica de la aplicación.
  + Esto facilita la escalabilidad, la implementación y el mantenimiento de aplicaciones complejas al permitir que cada servicio sea desarrollado y gestionado por separado.
* Arquitectura de Capas:
  + Una aplicación se divide en capas distintas, cada una con una función específica.
  + Las capas comunes incluyen la capa de presentación (interfaz de usuario), la capa de lógica de negocio y la capa de acceso a datos.
  + Promueve la modularidad y la separación de roles.
* Arquitectura de Componentes:
  + Se centra en la creación de componentes reutilizables que encapsulan la funcionalidad y la interfaz de usuario.
  + Los componentes se pueden combinar para construir aplicaciones más grandes y complejas.
* Arquitectura Serverless:
  + El proveedor de servicios en la nube se encarga de la administración de servidores y recursos, permitiendo a los desarrolladores centrarse en escribir código sin preocuparse por la infraestructura.
  + Se basa en funciones individuales que se ejecutan en respuesta a eventos.

## 1.5.- Plataformas web libres y propietarias.

Web recomendada: <https://w3techs.com/>

Una plataforma web es el entorno de desarrollo de software empleado para diseñar y ejecutar un sitio web. Cuatro componentes básicos:

* **El Sistema Operativo:** bajo el cual opera y funciona el equipo donde se hospedan las webs.
* **El servidor web:** es el software que maneja las peticiones desde equipos remotos a través de la Internet.
* **El gestor de bases de datos** se encarga de almacenar un conjunto de registros de datos relacionados.
* **Un lenguaje de programación interpretado:** que controla las aplicaciones de software que corren en el sitio web.

La plataforma **LAMP** trabaja enteramente con componentes de software libre.

* **L**inux, sistema operativo.
* **A**pache, servidor web.
* **M**ySQL, gestor de bases de datos.
* **P**HP, lenguaje interpretado PHP.

La plataforma **WISA** está basada en tecnología desarrolladas por la compañía Microsoft de software propietario.

* **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 **WAMP** (Windows, Apache, MySQL, PHP). **WIMP** un servidor Windows que corre MySQL y PHP. Y otras con Unix, MacOS, Solaris, con otros servidores web como Lighttpd, LiteSpeed, u otras bases de datos como Postgre SQL y otros lenguajes de programación Python, Perl.

## 1.6.- Escalabilidad

Uno de los principales factores que puede afectar al rendimiento de las mismas es el número de usuarios.

El éxito o el fracaso de un sitio web entre otros aspectos es por el dimensionamiento del sistema sobre el que se instala y soporta el software, 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.**

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:** 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.
* **Balanceador hardware:** se trata de dispositivos que, respondiendo únicamente al algoritmo de reparto de carga (Round Robin, LRU, …) redireccionan una petición http del usuario a la máquina que, según dicho algoritmo, convenga que se haga cargo de la petición. Mucho más rápido, pero no garantiza la misma sesión de usuario en la misma máquina
* **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 softwares, pero algo menos que los hardware.

**Cluster.**

Con la aparición de los servidores de aplicaciones en cluster se abre una nueva capacidad de escalabilidad que, podría clasificarse como vertical u horizontal. 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.

## 2.- Servidor Web Apache.

Un servidor web es un programa que se ejecuta de forma continua en un ordenador (también se utiliza el término para referirse al ordenador que lo ejecuta), se mantiene a la espera de peticiones por parte de un cliente y contesta a estas peticiones de forma adecuada, sirviendo una página web que será mostrada en el navegador o mostrando el mensaje correspondiente si se detectó algún error.

Servidor web más popular del mercado actualmente es Apache, de código abierto y gratuito, disponible para Windows, GNU/Linux, entre otros.

Su arquitectura:

* Estructurado en módulos.
* Cada módulo contiene un conjunto de funciones relativas a un aspecto concreto del servidor.
* El archivo binario **httpd** contiene un conjunto de módulos que han sido compilados.
* La funcionalidad de estos módulos puede ser activada o desactivada al arrancar el servidor.
* Los módulos se pueden habilitar o deshabilitar según las necesidades y se pueden dividir en varias categorías:
  + **Módulos de Núcleo (Core Modules):** fundamentales para el funcionamiento del servidor web y no se pueden deshabilitar: *mod\_mime, mod\_dir, mod\_log\_config.*
  + **Módulos de Autenticación y Autorización:** se utilizan para autenticar usuarios y controlar el acceso a recursos del servidor: *mod\_auth\_basic, mod\_auth\_digest, mod\_authz\_host.*
  + **Módulos de Procesamiento de Contenido:** permiten el procesamiento del contenido antes de que se envíe al cliente: *mod\_php, mod\_cgi.*
  + **Módulos de Seguridad:** ayudan a proteger el servidor web y sus recursos de amenazas de seguridad: *mod\_security.*
  + **Módulos de Compresión:** permite la compresión de contenido antes de ser entregado al cliente y ayuda a acelerar la carga de páginas web: *mod\_deflate*.
  + **Módulos de SSL/TLS:** habilitan la seguridad de capa de transporte (SSL/TSL) para establecer conexiones seguras y cifradas con los clientes: *mod\_ssl.*
  + **Módulos de Registro (Logging):** permiten la creación de registros de eventos y actividades del servidor: *mod\_log\_config*.
  + **Módulos de Caché:** permiten la implementación de caché para acelerar la entrega de contenido al cliente: *mod\_cache.*
  + **Módulos de Redirección y Reescritura:** permiten redireccionar URL y reescribir rutas de URL: mod\_redwrite.
* La configuración específica de un servidor Apache dependerá de las necesidades y requisitos de la aplicación o sitio web que se esté ejecutando.
* El servidor Apache se desarrolla dentro del proyecto HTTP Server (httpd) de la Apache Software Fountation. La licencia Apache permite la distribución de derivados de código abierto y cerrado a partir de su código fuente original.

## 2.1.- Instalación y configuración.

* Instalación de Apache en una máquina virtual, real o contenedor con distribución Ubuntu 22.04 LTS.
* Con usuario root.
  + sudo apt update
  + sudo apt install apache2
* Montar una Plataforma LAMP:
  + Sudo apt install php libapache2-mod-php php-mysql
  + Sudo apt install mysql-server
* Apache arranque automáico:
  + Sudo systemctl enable apache2
* Comprobación
  + <http://dir-ip-servidor>
  + <http://www.dawdistancia.net> si hemos configurado el hosts en el PC (c:\Windows\System32\drivers\etc\hosts)
* **Apache** sirve las páginas web que están en la carpeta **“/var/www/html**” y podemos crear más carpetas donde ubicaremos nuestras páginas web. Puerto **80**.
* Si se pretende servir más páginas web, es recomendable la utilización de **Host Virtuales**; para ello accedemos a la carpeta /**etc/apache2/sites-available**, y el fichero 000-default.conf que nos sirve de plantilla.
* **Apache** se configura colocando directivas en archivos de configuración de texto plano.
* El archivo principal de configuración se llama **apache2.conf.**
* Se puede añadir otros archivos de configuración mediante la directiva **Include** y se puede usar comodines para incluir muchos archivos de configuración.
* Todas las directivas deben colocarse en alguno de esos archivos de configuración.
* Apache2 sólo reconocerá los cambios realizados en los archivos principales de configuración cuando se inicie o reinicie.
* El archivo de configuración predeterminado de Apache2 es **/etc/apache2/apache2.conf**, se puede editar este archivo para configurar el número de puerto, la raíz de documentos, los módulos, los archivos de registro, los hosts virtuales, etc.
* Principales directivas:
  + **ServerRoot**, para especificar el directorio raíz del servidor web, donde se encuentran los archivos de configuración y los recursos del servidor.
  + **Listen**, para establecer el número de puerto y las interfaces de red en las que Apache escuchará las solicitudes entrantes**. /etc/apache2/ports.conf**
  + **ServerTokes**, para configurar la cantidad de información que Apache aporta sobre sí mismo.
  + **ServerSignature**, para indicar datos sobre Apache en el pie de los mensajes de error.
  + **User y Group**, para especificar el usuario y el grupo bajo los cuales se ejecutará el servidor web. Por defecto, el usuario y el grupo es **www-data.**
  + **Timeout**, para especificar el número de segundos antes de enviar un error de timeout.
* Para modificar el servidor virtual predeterminado, editamos el archivo **/etc/apache2/sites-available/00-default.conf**, algunas directivas son:
  + **ServerName**, nombre de dominio o servidor
  + **CustomLog**, define el archivo .log
  + **ServerAdmin**, especifica la dirección de correo del administrador del servidor.
  + **ErrorLog**, indica dónde guardar los registros de errores registrados en el servidor.
  + **DocumentRoot**, especifica dónde Apache debe buscar los archivos que forman el sitio.

## 2.2.- Iniciar Apache

* Probar su configuración:
  + Sudo apachectl configtest
* Introducir en un navegador la **URL**: http:ip-servidor.
* Puerto de escucha 80, por defecto.
* **Httpd** demonio o proceso principal
* **Apachectl**, es capaz de interpretar los argumentos start, restart y stop
* Nos mostrará si el servidor está arrancado, desde cuándo, la versión, etc.
  + Sudo systemctl status apache2
* También podemos utilizar estos comandos:
  + Sudo systemctl stop apache2
  + Sudo systemctl restart apache2
  + Sudo systemctl start apache2

## 3.- Aplicaciones web y servidores 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. Estos se distinguen de los servidores web en el uso extensivo del contenido dinámico y por su frecuente integración con base de datos.

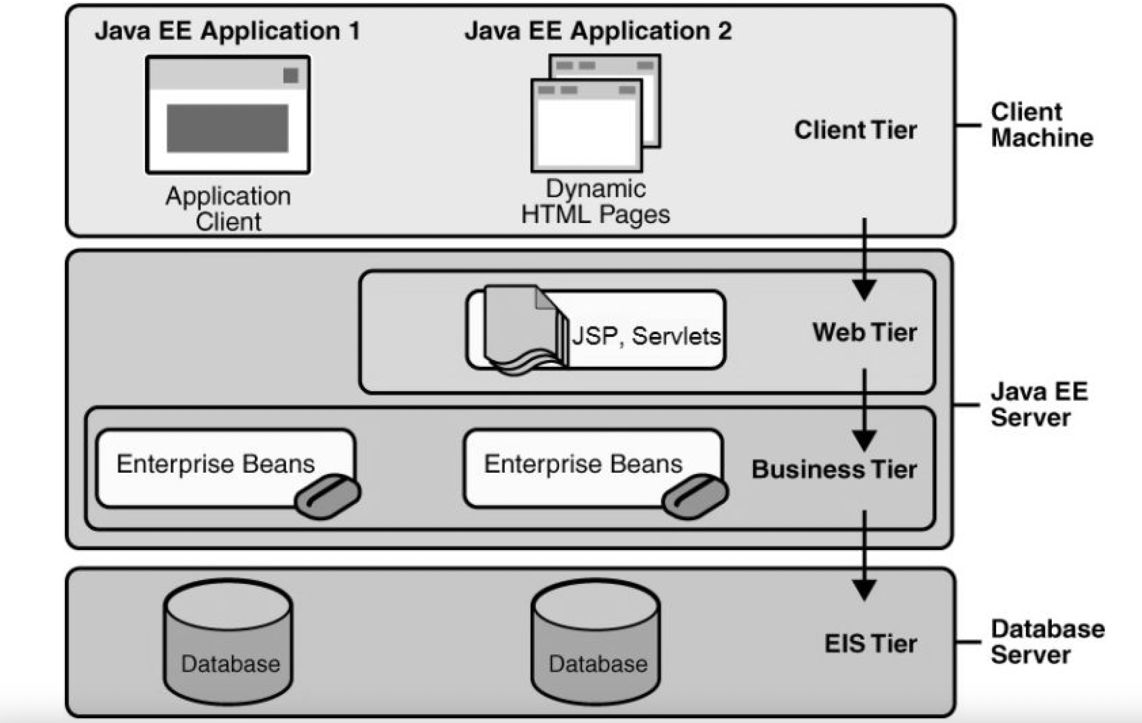
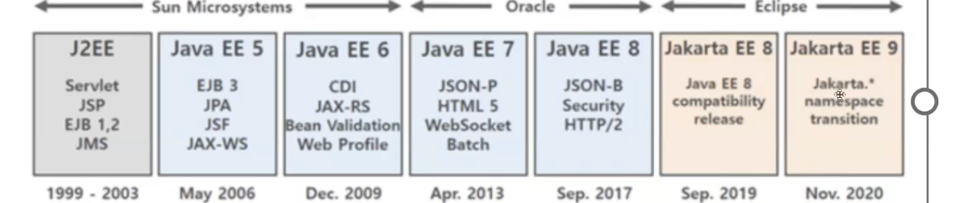
Un servidor de aplicaciones también es una máquina en una red de computadores que ejecuta determinadas aplicaciones, gestionando la mayor parte de las funciones de acceso a los datos de la aplicación.

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.

Otra ventaja es la **integridad de datos y código.**

**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.

Jakarta EE nivel medio y consiste en un servidor web y un servidor EJB.



## 3.1.- El servidor de aplicaciones Tomcat.

Tomcat es un servidor web (incluye el servidor Apache) y de aplicaciones del proyecto Jakarta, con lo cual, gestiona las solicitudes y respuestas HTTP y además es un servidor de aplicaciones o contenedor de Servlets y JSP.

Incluye el compilador Jasper, que compila JSP convirtiéndolas en servlets.

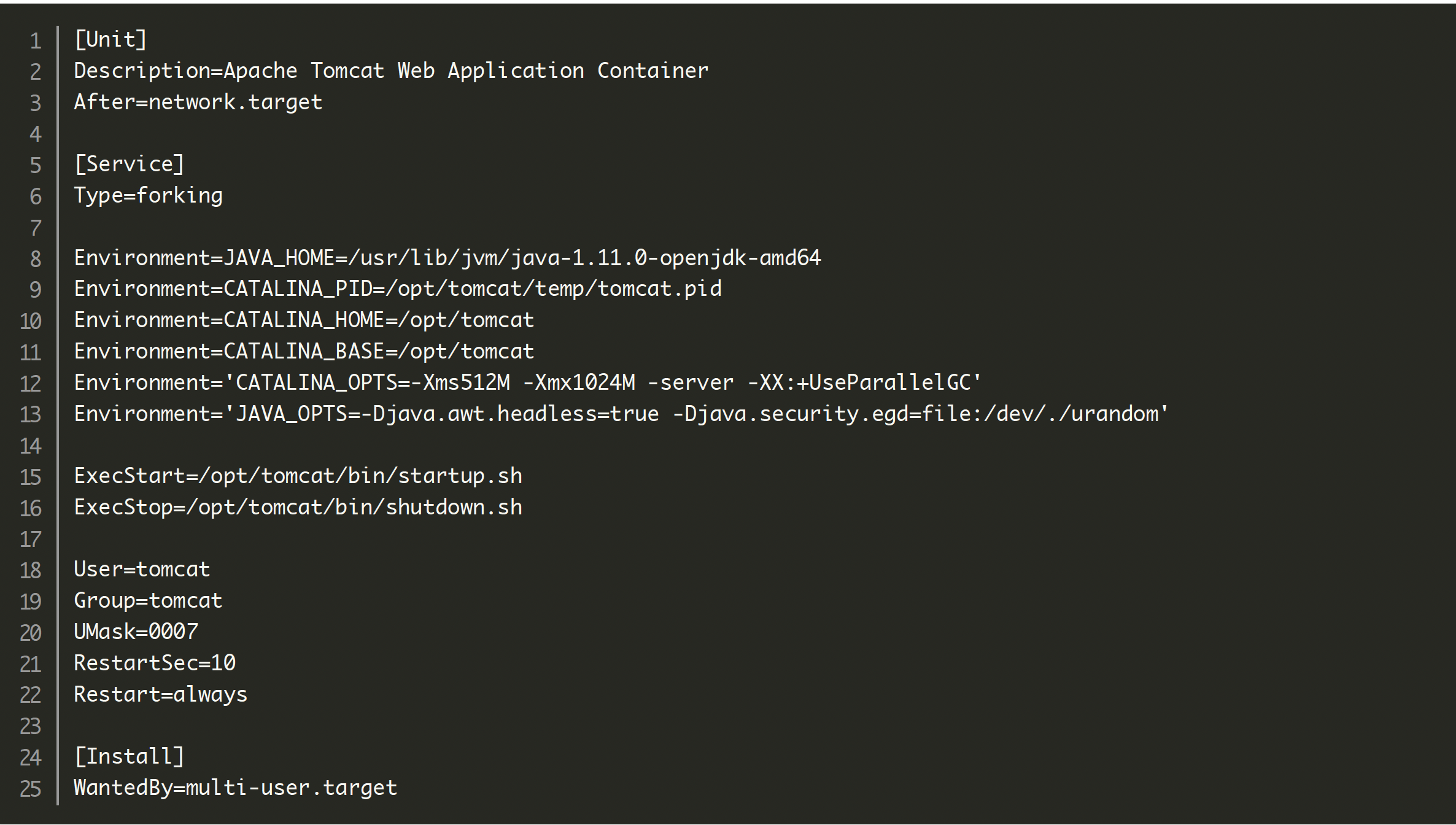
Tomcat es un contenedor de servlets con un entorno JSP. Un contenedor de servlets es un Shell de ejecución que maneja e invoca servlets por cuenta del usuario. Tipos:

* Contenedores de **servlets stand-alone (independientes):** estos son una parte integral del servidor web. Este es el caso en el que se usa un servidor web basado en Java.
* Contenedores de **servlets dentro de proceso:** el contenedor servlets es una combinación de un plugin para el servidor web y una implementación de contenedor Java. El plugin del servidor web abre una JVM dentro del espacio de direcciones del servidor web y permite que el contenedor Java se ejecute en él. En el caso de que una petición debiera ejecutar un servlet, el plugin toma el control sobre la petición y lo pasa al contenedor Java. Adecuado para servidores multi-thread de un solo proceso y el rendimiento está limitado en escalabilidad.
* Contenedores de **servlets fuera de proceso:** el contenedor servlets es una combinación de un plugin para el servidor web y una implementación de contenedor Java que se ejecuta en una JVM fuera del servidor web. El plugin del servidor web y el JVM del contenedor Java se comunican usando algún mecanismo IPC (sockets TCP/IP). Si una cierta petición tuviese que ejecutar un servlets, el plugin toma el control sobre la petición y lo pasa al contenedor Java (usando IPCs).

Tomcat puede utilizarse como un contenedor solitario (principalmente para desarrollo y depuración) o como plugin para un servidor web existente (actualmente soporta Apache, IIS).

## 3.1.1.- Instalación y configuración básica.

* Tener instalado JDK ya que las peticiones a Apache se redirijan a Tomcat empleando un conector proporcionado por Java.
  + Sudo apt update
  + Sudo apt install default-jdk
* Grupo y usuario tomcat, directorio /opt/tomcat y un shel /bin/false.
  + Sudo groupadd tomcat
  + Sudo useradd -s /bin/false -g tomcat -d /opt/tomcat tomcat
* Descargamos e instalamos Tomcat
  + Cd /tmp
  + Curl -O <https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.82/bin/apache-tomcat-9.0.82.tar.gz>
  + Sudo mkdir /opt/tomcat
  + Sudo tar xzvf apache-tomcat-\*tar.gz -C /opt/tomcat –strip-components=1
* Permisos
  + Cd /opt/tomcat
  + Sudo chgrp -R tomcat /opt/tomcat
* Mas permisos:
  + Sudo chmod -R g+r conf
  + Sudo chmod g+x conf
  + Sudo chown -R tomcat webapps/ work/ temp/ logs/
* Crear el archive tomcat.service
  + Sudo nano /etc/systemd/system/tomcat.service



* + En la variable de entorno **JAVA\_HOME** tendremos que poner la ruta que tengamos en nuestro equipo.
* Recargamos el demonio systemd
  + Sudo systemctl daemon-reload
  + Sudo systemctl start tomcat
* Comprobación
  + Sudo systemctl status tomcat
* Gestión del servicio a través del script **Catalina** y podemos proporcionarle parámetros start y stop.
* Comprobar que nuestro servidor está escuchando.
  + <http://dirip-servidor:8080>

## 3.1.2.- Iniciar Tomcat.

Tomcat va a estar escuchando en el puerto 8080 y va a tener su propio directorio de trabajo.

La misión de apache2 va a ser interceptar todas las peticiones en el puerto 80 y derivar las que considere necesarias a Tomcat (escalabilidad y balanceo de carga).

Por defecto Apache busca en /var/www/html y Tomcat trabaja en $CALINA\_HOME/webapps. La petición de una URL se puede gestionar, parte por Apache y parte por Tomcat.

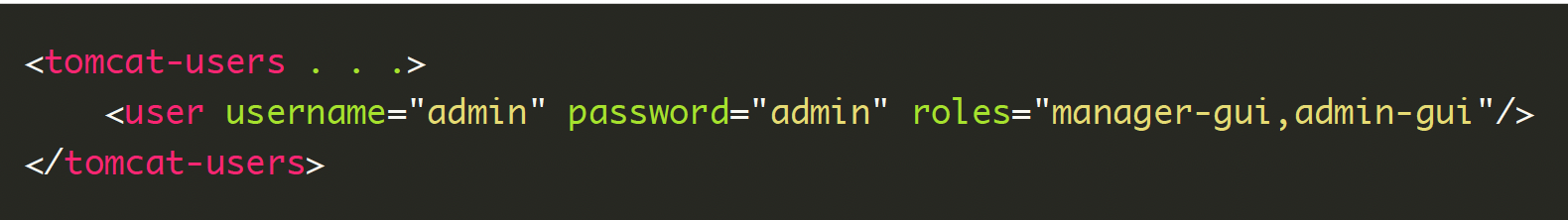
Editamos el fichero $CATALINA\_HOME/conf/server.xml

* Sudo nano /opt/tomcat/conf/server.xm.
* Host name=”localhost” appBase=”webapps”

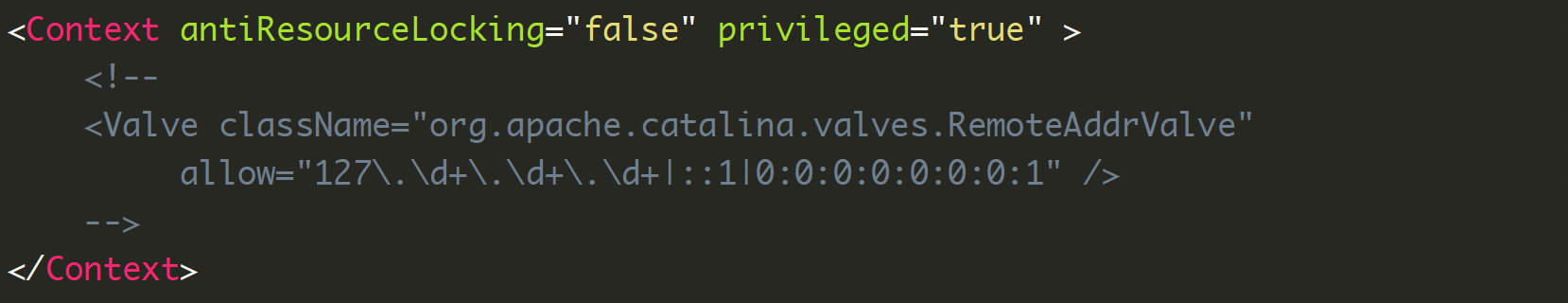
Para usar la aplicación de administración web que viene con Tomcat, debemos añadir un inicio de sesión a nuestro servidor de Tomcat. Editamos:

* Sudo nano /opt/tomcat/conf/tomcat-users.xml.

Deberíamos añadir un usuario que puede acceder a manger-gui y admin-gui:



Sino queremos quitar el acceso remoto editar el fichero $CATALINA\_HOME/webapps/manager/META-INF/context.xml



Comandas para el servidor:

* Sudo systemctl stop tomcat
* Sudo systemctl start tomcat
* Ó
* Sudo service tomcat stop
* Sudo service tomcat start

## 4.- Estructura y despliegue de una aplicación web.

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 servlet 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 app web se encuentra distribuido en una estructura de directorios, agrupándose ficheros según su funcionalidad.

De forma genérica una aplicación web se estructura en **tres capas:**

* Navegador web
* Tecnología web dinámica (PHP, Java Servlets, ASP, etc)
* Base de datos encargada de almacenar de forma permanente y actualizada la información que la aplicación web necesita.

## 4.1.- Archivos WAR

**Web Application Archive (Archivo de Aplicación Web)**, permiten empaquetar en una sola unidad aplicaciones web de Java completas, es decir, que su contenido sea:

* Servlets y JSP
* Contenido estático: HTML, imágenes, etc.
* Otros recursos web.

Ventaja, la simplificación del despliegue de aplicaciones web, sencilla instalación y solamente es necesario un fichero para cada servidor en un cluster, incrementa la seguridad ya que no permite el acceso entre aplicaciones web distintas.

Estructura:

* **/:** en la raíz del proyecto se almacenan elementos empleados en los sitios web.
* **/WEB-INF/:** se encuentran los elementos de configuración del archivo .**war** y el más importante es **web.xml.**
* **/WEB-INF/classes/:** contiene las clases Java empleados en el archivo .war y donde normalmente están los servlets.
* **/WEB-INF/lib/:** Contiene los archivos JAR utilizados por la aplicación y que normalmente son las clases empleadas para conectarse con la base de datos o las empleadas por librerías de JSP.

Para generar archivos **.war** se pueden emplear diversas herramientas:

* IDE: Netbeans, Eclipse, IntelliJIDEA
* Herramienta OpenSource: Ant
* Apache Maven y Gradle

Copiar el archivo .war al servidor: scp sample.war ubuntu@ip-server:/home/ubuntu

# Creación de máquinas virtuales

Ubuntuserver

* Dos adaptadores de red: Red Nat y sólo adaptador de red.
* Instalar SSH
* Instalar las Tools additions
  + Mount /dev/cdrom /media/cdrom
  + Sh VirtualBoxLinuxAddition.run

Ubuntuclient

* Un adaptador de red nat.
* Instalar los guest addition.
* Netstat –putan (puertos abiertos)

Mac

* Sudo /etc/hosts : nombres de dns local
* sudo killall -HUP mDNSResponder : borrar cache