1. **Describe los siguientes parámetros del servidor Apache: ServerName, ServerAlias, DocumentRoot y DirectoryIndex:**

**-ServerName:** Este parámetro sirve para suministrar una dirección de email del administrador del sistema al usuario que se conecta mediante un host virtual y será mostrada en las páginas de error generadas. Permitirá ponerse en contacto con el administrador del sistema.

**-ServerAlias:** Esta directiva permite establecer nombres adicionales y complementarios al host declarado en la directiva anterior.

**-DocumentRoot:** Esta directiva establece el directorio a partir del cual es visible la ruta desde el servidor web.

**-DirectoryIndex:** Es la lista de recursos que se van a buscar cuando un cliente solicite un directorio.

1. **Describe la arquitectura de las aplicaciones web MVC:**

**• Patrón de software** que separa la lógica de negocio y los datos de la parte representativa que observa el usuario, los eventos y las comunicaciones entre los distintos componentes.

• **Se basa en la separación** de módulos para su posterior mantenimiento y reutilización de código, además de la facilidad de detectar errores en caso de fallos.

**• Se distinguen los siguientes componentes:**

• **Modelo:** es el componente que se encarga de representar la información con la que la aplicación trabaja. Realiza las consultas y modificaciones con base en los privilegios definidos previamente en el análisis de requisitos de la aplicación. La petición llega por parte del controlador y este componente ejecuta la acción y la presenta a la Vista.

• **Vista:** visualiza el modelo con un tipo de representación para interactuar con el usuario. Normalmente es la interfaz de la aplicación, ya que puede ser una aplicación web, aplicación de escritorio, o cualquier aplicación en cualquier entorno o sistema operativo.

• **Controlador:** controla los otros dos componentes, respondiendo a peticiones realizadas por el usuario (pulsar un botón de la interfaz), y a partir de ahí se comunica con el modelo para manipularlo haciendo cambios en la vista. Es el mediador entre el Modelo y la Vista.

1. **Indica la estructura (principales carpetas y ficheros) de una aplicación web TOMCAT:**

La estructura de una aplicación web en Tomcat sigue una convención específica para asegurar que el servidor pueda localizar y servir los diferentes componentes de la aplicación correctamente. A continuación, se describe la estructura típica de directorios y ficheros de una aplicación web desplegada en Tomcat:

**Raíz del Proyecto (root directory)**

* **/WEB-INF/**
  + **/classes/**: Contiene los archivos .class (compilados) de las clases Java que forman parte de la aplicación.
  + **/lib/**: Carpeta donde se almacenan las bibliotecas (JAR files) necesarias para la aplicación.
  + **web.xml**: Archivo de configuración de la aplicación web (despliegue). Aquí se definen servlets, filtros, listeners y otras configuraciones relacionadas con la aplicación web.

**Directorios y Ficheros Comunes**

* **/META-INF/**: Contiene metadatos sobre el proyecto y opcionalmente el archivo context.xml que puede tener configuraciones específicas para el contexto de Tomcat.
* **/static/**: (convencional) Carpeta donde se almacenan recursos estáticos como imágenes, CSS, JavaScript, etc.
* **/jsp/**: (opcional, nombre convencional) Carpeta que contiene las páginas JSP (JavaServer Pages) que serán interpretadas por el servidor.
* **/HTML/**: (opcional, nombre convencional) Carpeta para almacenar archivos HTML estáticos.
* **index.html** o **index.jsp**: Página principal de la aplicación web que puede ser un archivo HTML o JSP.

1. **Indica las ventajas que supone que las sesiones en Tomcat sean persistentes, e incluso puedan ser guardadas en ficheros (disco):**

**Resiliencia ante Reinicios del Servidor:**

Recuperación de Sesiones: Almacenar las sesiones en disco permite que las sesiones de los usuarios se recuperen después de un reinicio del servidor. Esto asegura que los usuarios no pierdan su estado y datos de sesión, mejorando así la experiencia del usuario.

**Escalabilidad y Balanceo de Carga:**

Distribución de Cargas: En un entorno con múltiples instancias de Tomcat, la persistencia de sesiones en disco permite que cualquier instancia pueda acceder y restaurar una sesión desde un repositorio común. Esto facilita el balanceo de carga entre servidores y la escalabilidad horizontal.

**Tolerancia a Fallos:**

Minimización del Impacto de Fallos: Si un servidor falla, las sesiones almacenadas en disco pueden ser recuperadas por otra instancia del servidor. Esto reduce el impacto de fallos en la disponibilidad del servicio y mejora la continuidad del negocio.

**Mejora en la Gestión de Memoria:**

Liberación de Memoria: Al persistir sesiones en disco, se puede reducir la carga de memoria en el servidor, permitiendo manejar más usuarios simultáneamente sin agotar la memoria del servidor.

**Optimización del Reinicio del Servidor:**

Mantenimiento y Actualizaciones: Durante las actualizaciones o mantenimientos del servidor, las sesiones persistentes permiten que los usuarios mantengan su estado sin interrupciones significativas en su experiencia.

**Recuperación de Datos:**

Persistencia de Datos Críticos: En aplicaciones donde los datos de sesión contienen información crítica, almacenar estos datos en disco asegura que no se pierdan debido a problemas de memoria volátil.

1. **Para ser documentado con JavaDoc, describe como deben ser los comentarios y las etiquetas, e indica y describe al menos 3 etiquetas**

Los comentarios JavaDoc comienzan con **/\*\*** y terminan con **\*/**. Se colocan inmediatamente antes de las clases, métodos, constructores, interfaces y campos que se desean documentar.

**Ejemplos:**

**@param:**

* Descripción: Describe un parámetro del método.
* Uso: Se usa para documentar cada parámetro de un método.

**@return:**

* Descripción: Describe el valor de retorno de un método.
* Uso: Se utiliza en métodos que devuelven un valor.

**@throws:**

* Descripción: Describe las excepciones que un método puede lanzar.
* Uso: Se usa para documentar las excepciones que se espera que el método pueda lanzar.

1. **Describe los principales conceptos básicos asociados a un sistema de control de versiones: Revisión, Repositorio y Conflicto:**

**-Revisión:** Es una visión estática en el tiempo del estado de un grupo de archivos y directorios. Posee una etiqueta que la identifica. Suele tener asociado metadatos como pueden ser:

• **Identidad** de quién hizo las modificaciones. • Fecha y hora en la cual se almacenaron los cambios.

• **Razón** para los cambios.

**• De qué revisión** y/o rama se deriva la revisión.

• **Palabras** o términos clave asociados a la revisión.

**-Repositorio:** Lugar en donde se almacenan las revisiones. Físicamente puede ser un archivo, colección de archivos, base de datos, etc.; y puede estar almacenado en local o en remoto (servidor).

**-Conflicto:** Ocurre cuando varias personas han hecho cambios contradictorios en un mismo documento (o grupo de documentos); los sistemas de control de versiones solamente alertan de la existencia del conflicto. El proceso de solucionar un conflicto se denomina resolución.

1. **Describe e indica las ventajas de los sistemas de control de versiones centralizados y los sistemas de control de versiones distribuidos:**

**-Centralizados:** Tienen un único servidor que contiene todos los archivos versionados, y varios clientes que descargan los archivos de ese lugar central.

• **Implementaciones:** Subversion y Perforce

**• Ventajas:**

• **Control** detallado de qué puede hacer cada uno

• **Administración simplificada** frente a la gestión de bases de datos locales en cada cliente

**• Desventajas:**

• Punto único de fallo que representa el servidor centralizado.

**-Distribuido:** Los clientes no sólo descargan la última instantánea de los archivos sino que replican completamente el repositorio.

• **Implementaciones:** Git, Mercurial, Bazaar o Darcs

• **Ventajas:** Si un servidor muere, y estos sistemas estaban colaborando a través de él, cualquiera de los repositorios de los clientes puede copiarse en el servidor para restaurarlo, ya que cada vez que se descarga una instantánea se hace una copia de seguridad completa de todos los datos.

1. **Describe los tres estados principales en los que se pueden encontrar los archivos en GIT:**

**-Confirmado (Commited):** Significa que los datos están almacenados de manera que la base de datos local. Si una versión concreta de un archivo esta en el directorio de Git.

**-Modificado (Modified):** Estado en el que se ha modificado el archivo, pero todavía no se ha confirmado a la base de datos. Si ha sufrido cambios desde que se obtuvo del repositorio, pero no se ha preparado.

**-Preparado (Staged):** Significa que se ha marcado un archivo modificado en su versión actual para que vaya en la próxima confirmación. Si ha sufrido cambios desde que se obtuvo del repositorio, pero ha sido añadida al área de preparación.