DESPLIEGUE DE APLICACIONES WEB

CFGS. Diseño de Aplicaciones Web (DAW). 18-19 IES

FRANCESC DE BORJA MOLL

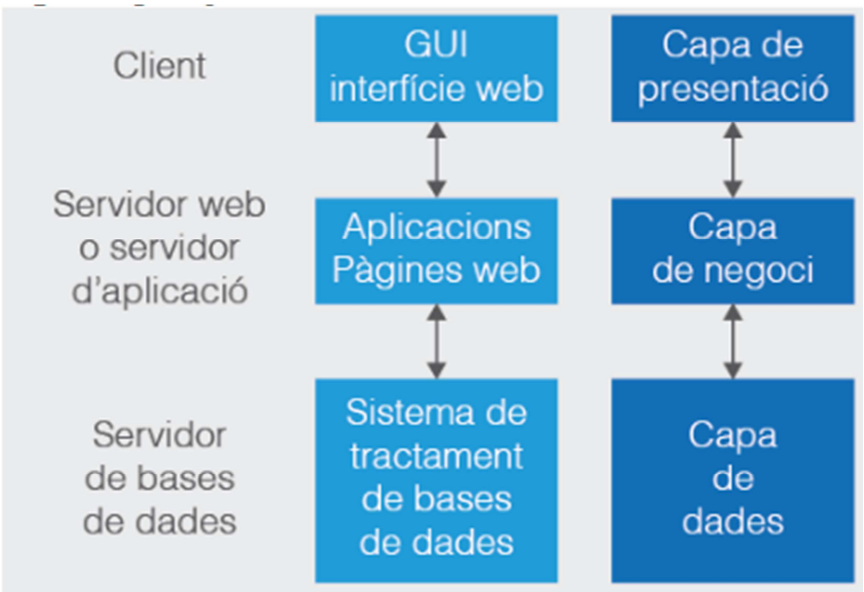
TEMA 1. IMPLANTACIÓN DE ARQUITECTURAS WEB.

1.1. LA ARQUITECTURA WEB (AW) Y ALGUNOS MODELOS.

Una aplicación web necesita de una estructura que permita su acceso desde diferentes lugares, esta estructura es lo que se denomina Arquitectura Web.

La mayoría de AW se basan en el modelo cliente/servidor (uno ofrece servicios y el otro hace uso de él). A parte de este modelo (que es el que se estudiará) existen otros, como son: P2P (peer-to-peer), B2B (business-to-business) (utilizado principalmente en operaciones bancarias) , etc.

La estructura de una AW actual sigue el siguiente modelo:



- **La capa de presentación** es la que ve el usuario (también se la denomina "capa de usuario"), presenta el sistema al usuario, le comunica la información y captura la información del usuario en un mínimo de proceso (realiza un filtrado previo para comprobar que no hay errores de formato). También es conocida como interfaz gráfica y debe tener la característica de ser "amigable" (entendible y fácil de usar) para el usuario. Esta capa se comunica únicamente con la capa de negocio.

- **La capa lógica de negocio** se implementa en un servidor de aplicaciones. Es ahí donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio (e incluso de lógica del negocio) porque es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos almacenar o recuperar datos de él. También se consideran aquí los programas de aplicación.

- **La capa de datos** es donde residen los datos y es la encargada de acceder a los mismos. Está formada por uno o más gestores de bases de datos que realizan todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

En el caso de utilizar páginas Web estáticas, no existiría la capa de datos, ya que éstos van incorporados en los propios archivos de marcas que serán las que conforman la página web.

Debido a la introducción de dinamismos en las páginas, la estructura anterior se ha visto alterada:

- Los navegadores Web son capaces de interpretar diferentes elementos dinámicos de forma automática, o mediante plugins (javascript, flash, etc).

- Los servidores Web también pueden interpretar código para generar las páginas Web, de esta manera se pueden introducir pequeños programas que alteran el contenido o el aspecto final de una página dependiendo de elementos como: el usuario que accede o la información solicitada en cada momento. Para interpretar este código el servidor necesita un módulo adicional (o se empotra ese módulo en el propio servidor, o se incorpora un servidor aparte).

Las principales ventajas de un modelo en capas son:

- Desarrollos paralelos (varios programadores en cada capa).
- Aplicaciones más robustas debido al encapsulamiento.
- Mantenimiento y soporte más sencillo (es más sencillo cambiar un componente que modificar una aplicación monolítica).
- Mayor flexibilidad (se pueden añadir nuevos módulos para dotar al sistema de nueva funcionalidad).
- Alta escalabilidad. La principal ventaja de una aplicación distribuida bien diseñada es su buen escalado, es decir, que puede manejar muchas peticiones con el mismo rendimiento simplemente añadiendo más hardware. El crecimiento es casi lineal y no es necesario añadir más código para conseguir esta escalabilidad.

1.2. UN MODELO SIMPLE PARA EL DESPLIEGUE DE APLICACIONES WEB.

Actualmente la mayor parte de la información y lógica de un negocio debe ser accesible desde diferentes lugares, es aquí donde entran en juego las aplicaciones web.

Aspecto básicos a considerar en una aplicación Web son:

- La seguridad (disponibilidad, confidencialidad, integridad, autenticación, no repudio).
- La velocidad.
- La estabilidad.

A la hora de diseñar e implantar una aplicación Web, se ha de considerar:

- Las posibles soluciones técnicas existentes, contra más soluciones técnicas conozcamos será mejor, ya que la combinación de varias puede hacer que la aplicación sea mejor e incluso más barata.

- El coste.

1.2.1. QUÉ ES UNA APLICACIÓN WEB.

Es una aplicación que se ejecutará a través de Internet, y constará al menos de dos partes:

- El servidor.
- Navegador Web de la máquina del cliente.

Ejemplo de aplicaciones Web son: correo electrónico, tiendas on-line, redes sociales, etc.

1.2.2. FASES DE UN PROYECTO DE UNA APLICACIÓN WEB.

De forma simplificada, pueden considerarse 5 fases:

1. Fase de requerimientos. En esta fase:

   - Determinar qué quiere el cliente.

   - Se ha de obtener una idea de cómo se llevará a cabo

   - Determinar si es o no viable.

2. Diseño. En esta fase:

- Cómo se realizará la aplicación, determinando las tecnologías y cómo se van a comunicar.

- Determinar los módulos y las interfaces a utilizar.

- Realizar un plan de proyecto en el que se dividan las tareas y responsabilidades y se calculen los tiempos para cada elemento así como su secuencia y dependencias.

- Obtener una especificación funcional en la que se detallen el funcionamiento y en el flujo de aplicación.

3. Desarrollo: en esta fase se desarrolla el proyecto en sí.

4. Pruebas

5. Implantación.

Una fase común a todo proyecto informático y que no se incluye aquí es el mantenimiento. Existen dos tipos de mantenimiento:

- Mantenimiento correctivo del servicio y corrección de errores. Asegurarse de que todo sigue funcionando de forma correcta y solucionar posibles errores y caídas del servicio.

- Mantenimiento evolutivo (Mejoras). Ampliar el proyecto.

Ambos casos pueden considerarse contratos a parte.

El orden correcto para el desarrollo es empezar de abajo a arriba, es decir:

1. la capa de datos.

2. la capa de negocio.

3. la presentación al cliente.

1.3. SERVIDORES WEB.

Un servidor Web es uno o varios programas que proporcionan un determinado servicio a través de una red. La comunicación con un servidor Web suele realizarse con el protocolo http.

El hardware que proporciona soporte físico a un servidor Web, puede albergar varios de ellos.

El objetivo principal de un servidor Web es proporcionar los medios para permitir la comunicación entre dos o más programas o grupos de software sin importar la tecnología usada para crear y operar cada unode ellos.

Ejemplos de servidores Web:

- Nginx.
- Apache.
- Internet Information Services (IIS).
- Cherokee.
- Tomcat.

Los servidores Web se engloban en un conjunto de sistemas más general que se denomina modelo distribuido, debido a que el sistema no es unitario, sino que está repartido entre diferentes máquinas. Este modelo tiene que afrontar algunos problemas que hay que tener siempre en cuenta:

1. La latencia y poca fiabilidad del transporte (red).
2. La falta de memoria compartida entre las partes.
3. Los problemas derivados de fallos parciales.
4. La gestión del acceso concurrente a recursos remotos.
5. Problemas derivados de actualizaciones de alguna/s de las partes.

1.3.1. ELEMENTOS PARA MONTAR SERVIDOR WEB.

Los elementos a emplear son:

- Máquina/s con las prestaciones mínimas necesarias en base a los requerimientos del servicio a desarrollar.
- Sistema operativo, lo más estable posible.
- Dirección IP estática.
- Conexión a Internet 24 horas.
- Software del propio servidor.

1.4. SERVIDOR APACHE.

Apache destaca sobre otros servidores por:

- Tener un diseño modular y altamente configurable.

- Ser de código abierto.
- Funciona bien con Perl, PHP y otros lenguajes de script.
- Existen versiones para muchos sistemas operativos incluyendo Windows, Linux y Mac.

1.4.1. DIRECTIVAS Y ARCHIVOS DE CONFIGURACIÓN.

El servidor Apache contiene una serie de archivos que será necesario configurar para dotarlo de la funcionalidad deseada. La configuración se realiza a través de directivas o reglas en archivos de configuración concretos.

En concreto, en sistemas Debian/Ubuntu se utiliza la siguiente estructura de archivos de configuración diseñada para facilitar la administración:

- /etc/apache2/apache2.conf : el archivo raíz es el que incluye a los demás. Es peligroso modificar este archivo.

- mods-enabled/*.load y mods-enabled/*.conf: la finalidad de estos archivos es la carga y configuración de los módulos de Apache.

- httpd.conf: es un fichero de configuración del servidor web Apache. Almacena información acerca de diversas funciones del servidor, que pueden añadirse o eliminarse agregando un "#" a comienzo de línea, las cuales determinan los valores para cada directiva con el fin de configurar Apache de acuerdo a nuestras necesidades.

- ports.conf : define en qué puertos «escuchará» Apache.

- conf.d/ : directorio que contiene archivos de configuración para cada funcionalidad de apache (charset, php, security, etc.)

- sites-enabled/ : directorio que contiene los archivos de configuración de cada virtual host.

1.5. SERVIDOR CON XAMPP.

XAMPP es un paquete de software libre, que consiste principalmente en el sistema de gestión de bases de datos MySQL, el servidor web Apache y los intérpretes para lenguajes de script PHP y Perl.
El nombre es en realidad un acrónimo: X (para cualquiera de los diferentes sistemas operativos), y además está compuesto por:

- Servidor web Apache
- Gestor de base de datos MariaDB/MySQL
- Lenguajes de programación PHP, Perl.

1.6. TOMCAT

Apache Tomcat es un servidor web creado para alojar Servlets y Java Server Pages (JSP). Tomcat es gratuito y de código abierto.

Cuando un cliente hace una petición al servidor web, éste trata de gestionarlo, pero hay muchos elementos con los que no sabe qué hacer. Aquí entra en juego el servicio de aplicaciones, que descarga al servidor web de la gestión de determinados tipos de archivos, en nuestro caso servlets y JSP.

Si un cliente hace una petición al servidor pidiendo un JSP, ésta llega al servidor Web que lee un archivo XML que le proporciona el servidor de aplicaciones y determina que el archivo lo gestionará el servidor de aplicaciones.

En el archivo XML también se incluye la dirección del servidor de aplicaciones y el servidor web la petición mediante HTTP.

Tomcat es un servidor web que da soporte a JSP y servlets. Además, es capaz de procesar peticiones en http y servir archivos HTML con bastante eficiencia.

1.7. APLICACIONES WEB.

Una aplicación Web se diferencia de una aplicación estándar (programas instalados en las máquinas) en que se accede a ella a través de una red como Internet o una intranet. En muchos casos, es una aplicación que se escribe en lenguajes soportados por los navegadores web (por ejemplo Javascript, HTML) y que necesita de un navegador Web para ejecutarse.

Un ejemplo de aplicación web es un programa que permita acceder a datos de una empresa desde el exterior de ésta. De esta forma, ya queda patente el principal problema de las aplicaciones Web: la seguridad.

Ventajas de las aplicaciones Web:

- No es necesario ningún tipo de distribución, instalación o actualización compleja de la aplicación. Simplemente el uso de un navegador compatible nos permitirá usar la aplicación.

- No se necesitan máquinas clientes potentes.
- Son fáciles de integrar con otras funcionalidades de servidor como el correo electrónico.
- Generalmente, permiten eliminar los problemas derivados del uso de diferentes plataformas informáticas (arquitecturas, sistemas operativos, etc.).

Desventajas de las aplicaciones Web:

- Las tecnologías Web son muy dinámicas y cambiantes por lo que se puede usar alguna funcionalidad que desaparezca o que se modifique, de forma que sea necesario realizar una adaptación.
- Dependen del correcto funcionamiento de la red.
- La privacidad nula.
- La seguridad es el parámetro más importante de cara a accesos de usuarios, ya que, al ser el acceso público (uso de internet con un navegador), la seguridad se ha de garantizar en todo momento. Se ha de proteger tanto la información crítica de la empresa, como los datos privados de los usuarios.

A la hora de implementar la seguridad de una aplicación web, se ha de tener en cuenta los siguientes aspectos:

- La autenticación de los usuarios. Es decir que sólo se conecten usuarios autorizados y no haya suplantaciones de identidad.
- La autorización de cada usuario. No todos los usuarios ha de poder acceder a todos los datos, operaciones o servicios. La identificación y creación de diferentes roles es un aspecto a tener en cuenta.
- La gestión de los recursos. Es necesario proteger los datos tanto cuando se encuentran en la base de datos como cuando se encuentran en tránsito entre el cliente y el servidor.
- La entrada de datos. Qué puede escribir el usuario en cada punto de entrada de datos. Cuanto más se limiten los caracteres y tipo de entrada que puedan realizar los usuarios, más fácil será impedir que la gente acceda a puntos no deseados de la aplicación e incluso a otras aplicaciones de la misma red.
- Auditorias y registros. Para controlar y corregir las deficiencias en seguridad es importante saber qué ha pasado. Por eso es conveniente tener métodos de registro de todo lo que suceda en nuestra aplicación.

Para el control de estos problemas existen dos recomendaciones básicas:

- Pruebas: cuanto más se pruebe una aplicación, menos fallos tendrá.

- Uso de marcos de trabajo (Framework): si hay un equipo de trabajo, es conveniente que todos los miembros trabajen de forma común, y no cada uno por su cuenta. Un marco de trabajo para aplicaciones Web debe de tener en cuenta los siguientes aspectos: persistencia de datos, gestión de sesiones y autenticación de usuarios, seguridad, uso de caché, uso de plantillas e incluir una interfaz de administración.

1.8.1. Estructura.

La estructura de una aplicación Web es equivalente a la de un servidor Web.
La aplicación debe funcionar en todos los niveles por lo que se tendrá una estructura en capas.

La más común es la estructura en tres capas: presentación, aplicación y datos.

La estructura de una aplicación Web también se usa para referirse a la distribución en directorios de todos los elementos que componen dicha aplicación.

1.8.2. Descriptor de despliegue.

Un descriptor de despliegue (en inglés Deployment Descriptor) (DD) es un componente de aplicaciones J2EE que describe cómo se debe desplegar (o implantar) una aplicación web. Esto dirige una herramienta de despliegue (o publicación) para desplegar un módulo o aplicación con opciones de contenedor específicas y describe requisitos de configuración específicos que puede resolver un desplegador.

En aplicaciones J2EE, XML se usa para la sintaxis del fichero descriptor de despliegue. Debe ser llamado web.xml, y debe ser colocado en un subdirectorio llamado WEB-INF, directamente debajo de la raíz de la aplicación web.

Ejemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.0" xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd">
    <servlet>
      <servlet-name>HolaMundoServlet</servlet-name>
      <servlet-class>org.prueba.servlet.HolaMundoServlet</servlet-class>
    </servlet>
    <servlet-mapping>
      <servlet-name>HolaMundoServlet</servlet-name>
      <url-pattern>/HolaMundoServlet</url-pattern>
    </servlet-mapping>
```

```
    <session-config>
      <session-timeout>30</session-timeout>
    </session-config>
</web-app>
```