Universidad de Alcalá

Escuela Politécnica Superior

Universidad de Alcalá

# Arquitectura y diseño de sistemas web y C/S

***Roberto Barchino Plata***

Jueves 9:00 – 12:00

Grado en Ingeniería Informática – Curso 2019/2020

Marcos Barranquero Fernández – 51129104N

## INFORMACIÓN BÁSICA

- Prof. Roberto Barchino Plata
- Práctica final: hacer app. Web con modelo vista – controlador. Se entrega en enero en la fecha del examen final.
- El 17 de octubre no hay clase.
- Dos retos voluntarios:
    - Usar cliente Bootstrap
    - Usar servidor que sea de java (java server page)
- Contactar por correo de la uah.
- Exámenes con teoría y práctica.

# 1 – ARQUITECTURA C/S

## INTRODUCCIÓN

- Un **servidor provee un servicio** o aplicación.
- Un **cliente demanda un servicio** o aplicación.
- Un sistema C/S reparte las tareas entre un **cliente** y un **servidor**, trabajando de manera **integrada** para ejecutar una aplicación de manera **distribuida** y **transparente**.
- Un servidor puede interactuar con varios clientes, por lo que debe gestionar **el acceso a sus recursos compartidos** para garantizar la seguridad del acceso.

| **Comparativa cliente / servidor** | |
|---|---|
| **Cliente** | **Servidor** |
| Inicia diálogo solicitando algún servicio. | Espera pasivamente peticiones de los clientes. |
| Solicita que se realice un trabajo en el servidor y espera los resultados. | Realiza trabajo y envía los datos solicitados o informa sobre el estado de la tarea. |
| No suele contestar peticiones de otras máquinas. | No suele enviar peticiones a otras máquinas. |

## CARACTERÍSTICAS DEL SISTEMA C/S

- **Transparencia sobre la localización física**
- **Separación de funciones** en módulo de **presentación, negocio** y **datos.**
- **Funcionalidad heterogénea multiplataforma**, con independencia de la máquina y SSOO que utilice el cliente.
- **Encapsulación de servicios**, modelo de caja negra para los clientes, permitiendo modificar la funcionalidad interna sin que los clientes tengan que ser modificados.
- **Protocolo asimétrico:** el cliente debe iniciar la comunicación, e **intercambiarán mensajes** en forma de petición y respuesta.
- **Escalabilidad:**
    - **Horizontal:** añadir o quitar servidores en función de la demanda sufrida.
    - **Vertical:** añadir o quitar hardware en función de la demanda sufrida.

## BENEFICIOS

- **Robusted:** protección a fallos en el cliente.
- **Amigabilidad:** interfaces interactivas intuitivas para el usuario.
- Además de características mencionadas anteriormente.

## PROBLEMAS

- **Pocas herramientas** para evaluar en tiempo real la carga de los sistemas.
- **Mecanismos de comunicación heterogéneos:** sockets, RPC, etc. No estandarizado.
- **Seguridad:** se debe montar transversalmente tanto en cliente como en servidor.
- **Problemas asociados a la red:** congestión, tráfico, etc.
- **Distribución de los datos** en distintas localizaciones, duplicidad de datos, etc.

# GENERACIONES DE SISTEMAS C/S

## ARQUITECTURA C/S EN UNA CAPA

Redes de área local, comunicaciones homogéneas, ineficientes para grandes organizaciones, grandes BBDD, dispersión geográfica.

## ARQUITECTURA C/S EN DOS CAPAS

Funciones distribuidas y coordinadas entre clientes y servidores. Falta de estándares y herramientas, resultando en complejidad arquitectural y desarrollo costoso.

## ARQUITECTURA C/S EN TRES CAPAS

Se distingue entre:

- Front-end que proporciona lógica de presentación al cliente.
- Capa interfaz que permite controlar la lógica de negocio sin modificar la capa real.
- Back-end que proporciona acceso a servicios dedicados y BBDD.

# MODOS DE LLAMADA C/S

- **Llamada síncrona**: la aplicación cliente se queda en modo de espera hasta que el servicio ejecuta y devuelve resultados.
- **Llamada asíncrona**: la aplicación cliente solicita el servicio pero no se queda esperando, sigue realizando otras tareas. Cuando el servidor finaliza y devuelve los datos, la aplicación es informada de ello y lo gestiona en ese momento.

# MODALIDADES Y CANALES DE LLAMADAS EN C/S

- **Sockets:** protocolo propio que permite enviar paquetes por IP:puerto.
- **RPC:** llamada a procedimiento remoto, sin tener que conocer la lógica de ese procedimiento. Se deben proporcionar los parámetros para que se pueda ejecutar la tarea solicitada.
- **RMI:** invocación de método remoto, propia de Java.
- **Corba:** estándar para invocación y solicitud de arquitecturas orientadas a objetos heterogéneas.
- **Servicios web:** mecanismos que crean servicios distribuidos sobre protocolo http y basados en tecnología SOAP o REST.

# COMPUTACIÓN EN LA NUBE

Permite ofrecer recursos y servicios de computación, permitiendo contratar el hardware lo que necesites de forma descentralizada.

# 2 – ARQUITECTURA WEB

## INTRODUCCIÓN A INTERNET

**Internet** es una red de ordenadores de tamaño mundial, que se comunica por el protocolo TCP/IP.

Inicialmente fue una red desarrollada por el Dpto. de Defensa, llamada **ARPAnet.**

Tiene una arquitectura **C/S**.

Para comunicarse con otros dispositivos, utiliza **dominios** que permiten enmascarar la IP. Estos dominios se pueden comprar de webs como iana.org o dominios.es.

## SERVICIOS

Un **servicio** es un conjunto de programas y utilidades que permiten realizar una tarea.

Un **servicio web** es un servicio alojado en internet, con un servidor asociado esperando conexiones entrantes. Los clientes solicitan **hipertextos** e **hipermedia.**

- **Hipertexto:** documentos web con enlaces a otros documentos.
- **Hipermedia:** hipertexto con información multimedia.

Cada web tiene un servidor que está escuchando por el **puerto 80 (por TCP)**. Los clientes, utilizando normalmente un **navegador** acceden a estos servicios a través de la **URL** y mediante el protocolo **http**, los clientes hacen solicitudes de los hipertextos escritos en **html**.

La forma general de una URL es la siguiente:

**servicio://host:puerto/recurso**

## HTTP

**HTTP** es un protocolo:

- A nivel de **aplicación:** capa de aplicación.
- **No orientado a estado**: tan solo solicitudes y respuestas.
- **Utiliza el puerto 80** aunque se puede configurar para emplear otros.
- Utiliza **MIME (multipart internet mail extension)** para determinar el tipo de datos que transporta. Esto se detalla en la **cabecera** del mensaje.

### COMANDOS HTTP

- **GET <archivo> <versión>:** petición de recurso.
- **POST <archivo> <versión> <args>:** petición de recurso **con parámetros.**
- **HEAD <datos>:** petición de datos sobre un recurso.
- **PUT <archivo>:** creación o envío de recurso.
- **DELETE<archivo>:** elimina recurso.

### RESPUESTAS HTTP

- **2XX:** petición procesada correctamente.
- **3XX:** petición redirigida.
- **4XX:** error o fallo en cliente.
- **5XX:** error o fallo en servidor.

### CABECERA HTTP

Contiene metadatos sobre el mensaje.

- **Host: url:** URL del host.
- **Accept: tipo:** tipo de fichero que admite recibir.
- **User-agent:** información sobre la aplicación utilizada para conectarse.

## ARQUITECTURA Y DESARROLLO WEB

Para el desarrollo de servicios web, actualmente se utiliza:

- Herramientas de Microsoft (ej. .NET)
- Herramientas Java (ej. servlets)
- Herramientas freeware (ej. Apache)
- Frameworks (ej. Spring)

Normalmente, se emplea una arquitectura **de tres capas:**

- **Capa de presentación:** funcionalidad de cara al cliente: navegador y servidor web responsable de presentar datos de forma adecuada.
- **Capa de lógica de negocio:** programas, scripts, funcionalidad real.
- **Capa de datos:** proporciona a la capa de lógica los datos necesarios (BBDD).

El cliente accede por un navegador al servidor web (por **http**).

El servidor dispone de los ficheros, organizados en una jerarquía, y devuelve los solicitados al cliente (por **http**).

# 3 – LENGUAJES DE MARCADO

## INTRODUCCIÓN A LENGUAJE DE MARCAS

Un lenguaje de marcas permite **codificar** un documento mediante **etiquetas.** Las etiquetas actúan como metadatos, definiendo el tipo de contenido que enmarcan.

Surge en los 60 para solventar problemas relacionados con el intercambio de los diferentes tipos de ficheros.

De cada a tecnología web, destacan **html y xhtml**.

## EVOLUCIÓN

En los 70 IBM realizó su propia implementación, en los 80 se realizó una versión estandarizada llamada **SGML.**

### HTML

En los 90 se estandarizó **html** para la publicación de información en la web.

Ha tenido varias versiones:

- 1995 – HTML2: primer estándar oficial.
- 1997 – HTML3.2: admite applets, y tablas.
- 1999 – HTML4.01: último estándar realizado por la W3C.
- 2008 – HTML5: realizado por la WHATWG y revisado en 2010 por W3C.

El **W3C** es una comunidad donde se desarrollan estándares web.

HTML5 es un compendio de:

**[Estructura + Contenido]** + Apariencia + Comportamiento

### XHTML

XHTML es una e**x**tensión de html, que lo hace compatible con **xml.**

Desarrollado tras HTML por la W3C.

### XML

XML es un **meta-lenguaje** que sirve para organizar y almacenar datos de forma legible. También permite definir la gramática de lenguajes formales y propone un estándar de intercambio de información estructurada en diferentes plataformas.

Existen otros estándares que complementan su funcionalidad:

- **DTD**: complemento que define elementos y atributos permitidos, definiendo limitaciones de su **xml asociado.**
- **XML Schema:** similar al DTD en el sentido de que define qué admite y qué no en un XML, pero usando sintaxis **XML.**

## EJEMPLO HTML

```
<!DOCTYPE html>
<html lang="es" dir="ltr">

<!-- Cabecera -->

<head>
  <title>Contacto | Apartamentos Tripi</title>
  <meta charset="UTF-8">
  <link rel="stylesheet" href="styles/contacto/layout.css" type="text/css">
</head>

<!-- Cuerpo -->

<body>
  <!-- Div -->
  <div class="wrapper row1">
    <!-- Header -->
    <header id="header" class="clear">
      <!-- Lista -->
      <ul>
        <!-- Enlace -->
        <li><a href="./benidorm.html">Benidorm</a></li>
        <li><a href="./galicia.html">Galicia</a></li>
        <li class="last"><a href="./reserva.html">Reserva</a></li>
      </ul>
    </header>
  </div>


  <section>
    <img src="images/contacto.jpg" alt="">
    <p> Párrafo </p>
  </section>

  <!-- Artículo -->
  <article class="two_quarter">

    <!-- Formulario -->

    <form action="mailto:marcos.barranquero@edu.uah.es" method="post">

      <input type="text" name="nombre" placeholder="Nombre: ">
      <input type="email" name="email" placeholder="Email: ">
      <input type="text" name="contenido" placeholder="Mensaje: ">

      <input type="submit">
    </form>

  </article>


  <footer id="footer" class="clear">
    <p class="fl_right">Arquitectura y diseño de sistemas web y C/S - UAH - 2018 </p>
  </footer>
</body>

</html>
```

EJEMPLO XML

```xml
<?xml version="1.0" encoding="UTF-8"?>
<menu_restaurante>
<plato>
    <nombre>Puré de patata</nombre>
    <precio>5.95€</precio>
    <descripcion>Puré de patata roja con picatostes</descripcion>
    <calorias>350</calorias>
</plato>
<plato>
    <nombre>Macarrones con queso</nombre>
    <precio>4.95€</precio>
    <descripcion>Macarrones con queso parmesano</descripcion>
    <calorias>450</calorias>
</plato>
</menu_restaurante>
```