Introducción a la programación web avanzada

Jordi Sánchez Cano

PID_00172704

UOC
Universitat Oberta de Catalunya

Los textos e imágenes publicados en esta obra están sujetos –excepto que se indique lo contrario– a una licencia de Reconocimiento-Compartir igual (BY-SA) v.3.0 España de Creative Commons. Se puede modificar la obra, reproducirla, distribuirla o comunicarla públicamente siempre que se cite el autor y la fuente (FUOC. Fundació per a la Universitat Oberta de Catalunya), y siempre que la obra derivada quede sujeta a la misma licencia que el material original. La licencia completa se puede consultar en: http://creativecommons.org/licenses/by-sa/3.0/es/legalcode.ca

Índice

Introducción

En este módulo se ofrece una introducción a las tecnologías y los lenguajes relacionados con la Web desde sus inicios hasta la actualidad.

Inicialmente se describen las diferencias entre páginas tradicionales y páginas dinámicas. Hecho esto, se explica qué son las aplicaciones web y cuáles son sus ventajas sobre las aplicaciones de escritorio. Más adelante se detallan las principales tecnologías y lenguajes de cliente y de servidor de forma cronológica. Por último, se introduce la definición de *aplicaciones de Internet enriquecidas*, RIA (*rich Internet applications*) y se presentan las tecnologías más populares relacionadas con este tipo de aplicaciones.

Objetivos

Los objetivos de este módulo didáctico son los siguientes:

1. Entender la diferencia entre páginas estáticas y páginas dinámicas.

2. Conocer qué son las aplicaciones web y RIA.

3. Ver la evolución de las tecnologías y los lenguajes de servidor y de cliente en el entorno web.

1. Evolución de la programación web

Este módulo se centra en las aplicaciones web que son por naturaleza cliente-servidor. A continuación, se explican los conceptos principales sobre arquitectura cliente-servidor y páginas y aplicaciones web.

1.1. Introducción a la arquitectura cliente-servidor

Una arquitectura cliente-servidor está formada por diferentes equipos que forman parte de una misma red, en la que unos proporcionan servicios a otros.

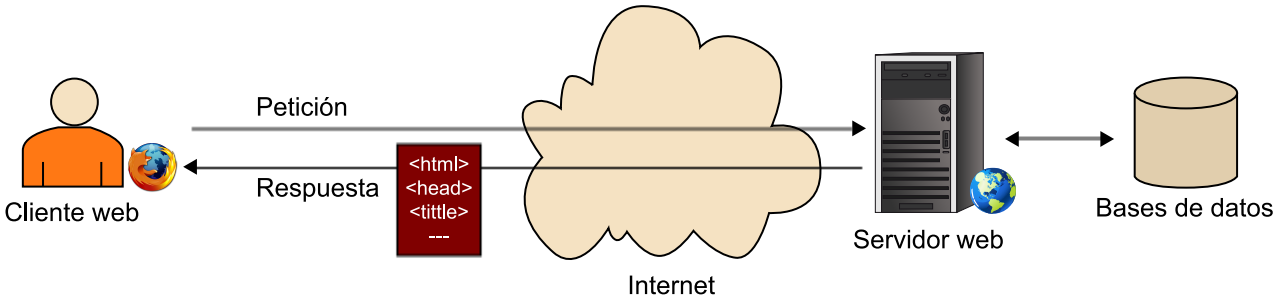
El uso de servicios se realiza mediante el intercambio de peticiones, de respuestas y de datos. Los equipos que solicitan los servicios se llaman **equipos cliente** y los que proporcionan el servicio, **equipos servidor**.

Los equipos proporcionan servicios mediante la ejecución de aplicaciones que se encargan de recibir, procesar y responder a las diferentes solicitudes. Estas aplicaciones se denominan también *servidores* (son ejemplos de servidores los servidores de correo, los servidores web y los servidores de archivos). Un mismo dispositivo puede ofrecer varios servicios según las aplicaciones que esté ejecutando.

Las aplicaciones encargadas de hacer uso de los servicios se denominan **clientes**, como por ejemplo, los clientes de correo o los clientes FTP[1]. Los navegadores web son otro ejemplo de clientes web.

(1)FTP es la sigla de *file transfer protocol* ('protocolo de transferencia de archivos').

Figura 1. Peticiones de páginas por navegadores web



En la figura 1 se muestra un ejemplo de cliente-servidor web. Los clientes envían peticiones al servidor, y éste las procesa y envía páginas web de vuelta.

La arquitectura de aplicaciones cliente-servidor presenta ventajas como las siguientes:

- Los recursos están centralizados en uno o varios servidores, lo que permite la unificación y la integridad de los datos, el control de los diferentes accesos de clientes, etc.

- Esta arquitectura posibilita la escalabilidad del servicio, así como la ejecución en paralelo. Los dispositivos físicos pueden ser ampliados o reemplazados por otros que ofrezcan un mejor rendimiento. También es posible ampliar el número de servidores de un mismo servicio para permitir el proceso de una misma petición en paralelo, de manera que cada servidor realizaría una parte de la tarea.

- Su administración es más sencilla, ya que se centra en la parte servidor.

1.2. Páginas web

Una página web es un documento escrito en un lenguaje de marcado que permite representar información principalmente textual. La característica fundamental de las páginas web son los enlaces o hiperenlaces, que posibilitan la navegación de un documento a otro.

Lenguaje de marcado

El lenguaje de marcado más extendido es el HTML (*hypertext markup language*, 'lenguaje de marcado de hipertexto').

Las páginas web pueden ser estáticas o dinámicas según su comportamiento desde que se solicitan hasta que están disponibles en el navegador.

Imágenes en páginas web

Las páginas HTML también pueden mostrar imágenes y objetos incrustados, que dependerán de la implementación del navegador o de los complementos instalados.

1.2.1. Páginas estáticas

Las páginas web estáticas son aquellas que no ofrecen ninguna interacción ni modificación de su contenido durante todo el proceso de solicitud y presentación en el navegador: el mismo contenido que reside en el servidor se envía sin experimentar ninguna modificación.

Una vez que el navegador muestra la página, ésta no realiza ninguna acción ni respuesta a eventos.

Primeras páginas web

Las primeras páginas web eran estáticas y sólo presentaban documentos con enlaces o *links* hacia otras páginas.

La tecnología necesaria para mostrar páginas estáticas es un servidor HTTP y un navegador web. El navegador solicita las páginas mediante peticiones HTTP, y el servidor devuelve los archivos correspondientes.

HTTP

HTTP es la sigla de *hypertext transfer protocol* ('protocolo de transferencia de hipertexto'). Se trata de un protocolo orientado a transacciones para solicitar, entre otras operaciones, recursos alojados en un servidor. Los servidores HTTP también se denominan **servidores web**.

1.2.2. Páginas dinámicas

Las páginas dinámicas, a diferencia de las estáticas, ofrecen dinamismo, ya sea en el servidor al solicitar la página o en el navegador del cliente.

El dinamismo en el lado del servidor permite generar partes HTML de un documento de forma dinámica según los parámetros de solicitud, el contexto del usuario, el navegador que se esté utilizando, etc.

El dinamismo generado durante la representación de la página en el navegador se realiza mediante lenguajes interpretados, como Java-Script, y mediante el DOM[2]. Estas páginas pueden responder a diferentes eventos: carga inicial de la página, clic sobre un botón, movimiento del ratón sobre un área, etc. También pueden llevar a cabo modificaciones sobre el mismo documento sin actualizar la página con nueva petición: ocultar una tabla, mostrar una imagen, añadir elementos a una lista, etc.

[(2)]DOM es la sigla de *document object model* ('modelo de objetos del documento').

DOM

DOM es una API (*application programming interface*, 'interfaz de programación de aplicaciones') que permite a los programas y los *scripts* acceder y manipular el contenido de un documento de forma dinámica.

Con la evolución de las tecnologías de servidor y de cliente, surge un nuevo concepto de aplicación centralizada por medio de la Web que se explica a continuación.

1.3. Aplicaciones web

Una aplicación web es una aplicación residente en un servidor remoto a la que se accede por medio de una aplicación cliente, generalmente un navegador.

Navegadores

Los navegadores web se consideran clientes ligeros, ya que la mayor parte del proceso se lleva a cabo en el servidor.

Con la evolución y la aparición de diferentes tecnologías, protocolos y estándares, tanto de cliente como de servidor, los desarrolladores pueden optar por un modelo de aplicación centralizada, en el que la lógica de negocio y los datos residen en servidores remotos. Las interfaces captan las acciones y los datos introducidos por los usuarios y los envían al servidor en espera de una respuesta. Asimismo, los clientes reciben interfaces en forma de páginas web con datos procesados desde el servidor.

En los últimos años las aplicaciones web se han extendido por Internet ofreciendo diferentes servicios y comercios: correo electrónico, tiendas en línea, *blogs*, etc. Este tipo de aplicaciones ofrece algunas ventajas respecto a las aplicaciones de escritorio, como por ejemplo, las siguientes:

- **Compatibilidad y multiplataforma**. Las aplicaciones web envían interfaces utilizando estándares compatibles con los navegadores más populares: HTML, Java-Script, CSS[3], etc.

[(3)]CSS es la sigla de *cascading style sheets* ('hojas de estilo en cascada').

- **Facilidad de distribución y modificación**. Toda la aplicación está alojada en un servidor y no en los terminales cliente, por lo que las modificaciones realizadas se reflejarán en los accesos posteriores.

- **Portabilidad**. Los datos de los usuarios residen generalmente en el servidor, por lo que se podrá utilizar la aplicación desde cualquier terminal con un navegador compatible.

Complementos

Algunas tecnologías de servidor utilizan mecanismos propios que requieren la instalación de conectores (*plug-ins*) en el navegador. Son ejemplos muy extendidos de ello Flash Player o los *applets* de Java.

Las aplicaciones web presentan también algunos inconvenientes:

- No ofrecen interfaces ni funcionalidades tan ricas como las aplicaciones de escritorio debido a las limitaciones de las tecnologías del lado del cliente. Aun así, la evolución de los estándares y la aparición de otros nuevos hace que en la actualidad se disponga de aplicaciones en las que es difícil distinguir entre aquellas que son web y aquellas que son de escritorio.

- Dado que la aplicación se distribuye por la Red, la disponibilidad, la respuesta y la experiencia del usuario dependerán de la velocidad de ésta.

Evolución de estándares

Por ejemplo, la nueva especificación de HTML versión 5 permite la reproducción de audio y vídeo sin necesidad de incorporar complementos adicionales como Flash Player.

1.4. Evolución de la Web

La evolución de la Web ha venido impulsada tanto por cambios tecnológicos como sociales. En sus inicios, la denominada Web 1.0 se basaba en la lectura de contenidos por parte de los clientes y en la publicación de dichos contenidos por los administradores de los sitios web. Por lo tanto, la Web 1.0, desde un punto de vista de los usuarios, era un medio sólo informativo.

Con la aparición de nuevas tecnologías y tendencias de uso, surge el término Web 2.0, asociado a la web orientada a la interacción y las redes sociales, en la que la participación de los usuarios es fundamental para la incorporación de nuevos contenidos.

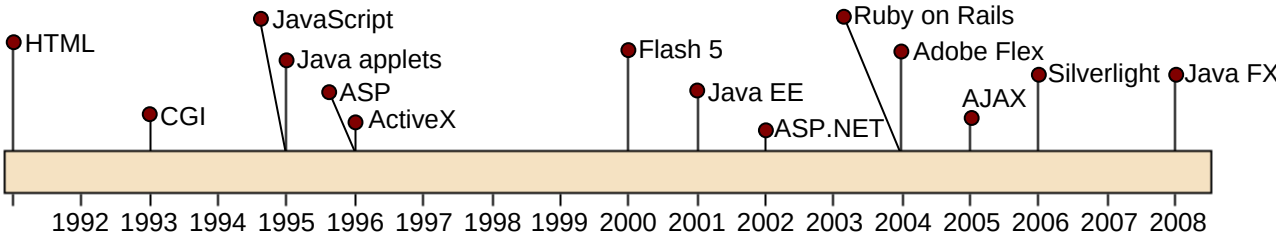
Por lo tanto, en la Web 2.0 los usuarios son también partícipes de los contenidos, como por ejemplo, en foros de discusión, *wikis*, bitácoras, etc.

Wikis

Las *wikis* son páginas web cuya información puede ser editada por los internautas por medio de su navegador.

La figura siguiente muestra un cronograma con la aparición de las tecnologías web más importantes.

Figura 2. Principales tecnologías web



Las páginas estáticas aparecen con el lenguaje HTML y, aunque posteriormente surgen las páginas dinámicas con CGI[4], Java-Script, ASP[5], etc., aquéllas siguen coexistiendo con éstas de forma paralela.

Algunas de estas tecnologías se ejecutan en el navegador, mientras que otras lo hacen tanto en el cliente como en el servidor. En los apartados siguientes explicamos cada una de ellas.

(4)CGI es la sigla de *common gateway interface* ('interfaz común de pasarela').

(5)ASP es la sigla de *active server pages* ('páginas de servidor activo').

2. Tecnologías y lenguajes de cliente

A continuación, se explican algunas tecnologías que se ejecutan en el cliente.

2.1. Java-Script

Java-Script es un lenguaje de *script* creado por Netscape Communications que se incluyó en su navegador Netscape Navigator 2.0 en 1995. En 1997 Java-Script fue aceptado como estándar ECMA con el nombre ECMAScript. Posteriormente fue declarado estándar por la ISO.

Origen de Java-Script

Java-Script se denominó inicialmente LiveScript, expresión en la que *live* hacía referencia al hecho de dar vida a las páginas estáticas. En diciembre de 1995 este lenguaje fue rebautizado como Java-Script.

En la actualidad, todos los navegadores incorporan una implementación de Java-Script y permiten ejecutar acciones en páginas HTML para dotarlas de dinamismo e interactividad. Algunos de sus usos pueden ser los siguientes:

- Responder a acciones del usuario sobre una página, como clics, movimiento del puntero del ratón, pulsaciones de teclas, etc.

- Comprobar los campos de formularios antes de que sean enviados al servidor. Esto ayuda a detectar entradas de datos inválidas en campos con un formato concreto. Por ejemplo, un campo que solicite la edad del usuario no debe permitir caracteres alfabéticos.

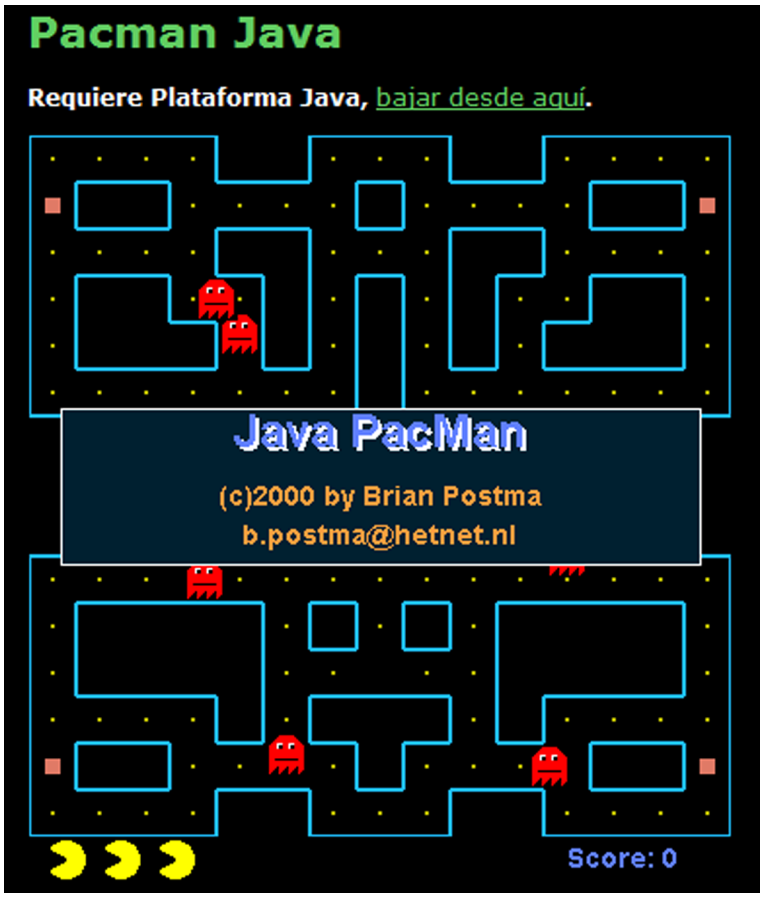
- Realizar, con el uso conjunto de otras tecnologías, peticiones asíncronas y actualizar porciones de páginas sin necesidad de cargar la página entera. Esta tecnología, que se conoce como AJAX, se estudia en profundidad en otros módulos.

- Modificar el contenido de una página de forma dinámica. Por ejemplo, una función Java-Script puede reestructurar una tabla, alterar su color de fondo, cambiar el estilo de un párrafo, etc. Para modificar el contenido de un documento, se utiliza el DOM.

2.2. *Applets* de Java

Los *applets* de Java son programas desarrollados en lenguaje Java que se pueden incorporar en una página HTML como objetos incrustados. Se descargan y se ejecutan en el navegador del cliente mediante una máquina virtual.

Figura 3. Juego Pacman desarrollado en un *applet* de Java



Ejecución de los *applets*

Los *applets* de Java se ejecutan mediante la JVM (*Java virtual machine*, 'máquina virtual de Java'). El navegador web requerirá tener instalado un complemento de la JVM para ejecutarlos.

Los *applets* presentan casi tantos inconvenientes como ventajas y, antes de elegir esta tecnología, se deben tener en cuenta varias consideraciones.

1) Ventajas

- Son multiplataforma y se pueden ejecutar en múltiples navegadores. Existen entornos de ejecución de Java para los sistemas operativos más utilizados. Asimismo, los navegadores más extendidos también disponen de complementos para la ejecución de *applets* en páginas HTML.

- Permiten realizar aplicaciones más sofisticadas que Java-Script.

- La ejecución de un *applet* precompilado es más rápida que Java-Script.

- Si se otorgan los permisos necesarios en el navegador, un *applet* puede tener acceso a los recursos del sistema, aunque en algunos contextos de seguridad, esto podría ser un inconveniente.

Entornos de ejecución de un *applet*

Un mismo *applet* alojado en una página se puede ejecutar en diferentes navegadores (como Firefox, Chrome, Safari, etc.) y en diferentes sistemas operativos (como la familia de Windows, Linux o Mac OS).

2) Inconvenientes

- La inicialización de un *applet* puede tardar en exceso si la JVM no está en funcionamiento.

- Un *applet* depende de una versión de JRE[6] específica. Si el navegador no dispone de esta versión, habrá que actualizarla.

- El *applet* desarrolla su ejecución dentro del contenedor que lo encapsula, sin tener acceso al resto de la página.

(6)JRE es la sigla de *Java runtime environment* ('entorno de ejecución de Java').

2.3. Adobe Flash

Adobe Flash es una plataforma multimedia para añadir animaciones e interacción en páginas web. Fue introducida en 1996 y actualmente Adobe Systems es quien la mantiene y la distribuye.

Uso principal de Flash

Flash se utiliza generalmente para crear animaciones, anuncios publicitarios, reproducción de audio y vídeo, etc.

Está basado en fotogramas que incluyen gráficos vectoriales, imágenes y *streams* bidireccionales de audio y vídeo. La lógica se realiza mediante un lenguaje de *script* y orientado a objetos llamado ActionScript.

Las animaciones se compilan en ficheros con extensión SWF[7] y se incluyen en las páginas como objetos incrustados. Su ejecución se realiza en el navegador cliente, que debe incorporar un complemento Flash Player.

(7)SWF es la sigla de *small web format* ('pequeño formato web').

Flash Player

Flash Player es un entorno basado en una máquina virtual que interpreta ActionScript y que ejecuta las animaciones Flash. Existen complementos de este entorno para los navegadores web y en gran variedad de dispositivos móviles.

2.4. ActiveX

ActiveX es una tecnología propietaria de Microsoft para el desarrollo de componentes reusables entre diferentes aplicaciones. Fue presentada en 1995 como el resultado del uso de otras tecnologías de Microsoft ya existentes.

ActiveX ofrece una serie de mecanismos para encapsular componentes de software con una interfaz que permite incluirlos dentro de otras aplicaciones heterogéneas. De esta forma, una aplicación puede incluir componentes ActiveX independientemente del lenguaje en el que esté escrita.

ActiveX frente a *applets*

Los controles ActiveX se pueden comparar con los *applets* de Java. Los *applets* se pueden ejecutar en una amplia gama de plataformas y dispositivos frente a los controles ActiveX, que dependen del navegador y las plataformas de Microsoft.

En la actualidad existen un gran número de aplicaciones, muchas de ellas de Microsoft, que exponen funcionalidades como controles ActiveX. Entre ellas destacan Internet Explorer, Visual Studio y Windows Media Player.

Los componentes ActiveX pueden ser distribuidos como aplicaciones por Internet para mostrar elementos complejos, como animaciones, calendarios, rejillas de datos ricas, documentos Word o Excel, etc.

3. Tecnologías y lenguajes de servidor

A continuación, veremos algunos ejemplos de tecnologías que se ejecutan en el servidor.

3.1. CGI

CGI[8] aparece en el año 1993 como un modo estándar de solicitar y transferir datos a un programa alojado en un servidor mediante HTTP. Estos programas son comúnmente conocidos como CGI.

[8]CGI es la sigla de *common gateway interface* ('interfaz de entrada común').

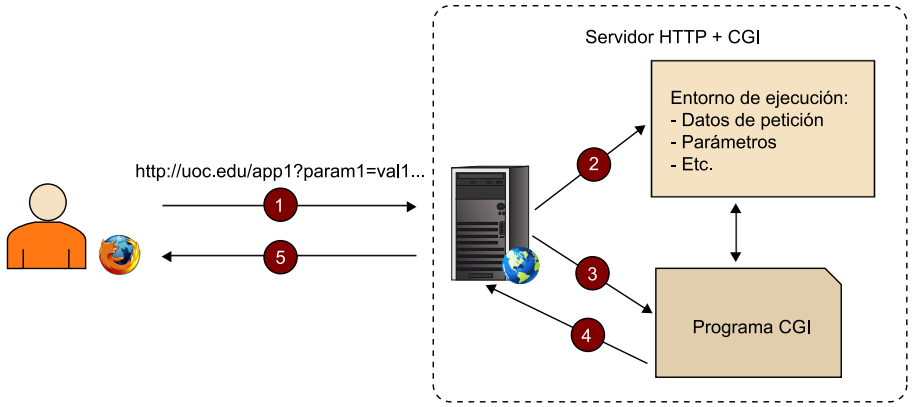
La primera tecnología para webs dinámicas

Las aplicaciones CGI fueron las primeras soluciones que permitían generar páginas web con contenidos dinámicos.

> CGI fue una de las primeras tecnologías que permitieron distribuir páginas web dinámicas desde el lado del servidor.

Los CGI pueden estar escritos en cualquier lenguaje, como C, C++, Java o Perl. Permiten generar contenidos dinámicos por la salida del programa a partir de los datos de entrada. Generalmente generan páginas HTML, por lo que suelen utilizar lenguajes con un gran soporte para la manipulación de cadenas.

Figura 4. Funcionamiento básico de un CGI



El funcionamiento básico de un CGI es el siguiente:

1) Un cliente realiza una petición HTTP a una aplicación CGI. Esta petición puede realizarse a partir de un envío (*submit*) de un formulario o un enlace.

2) El servidor prepara el entorno de ejecución con los parámetros de entrada obtenidos de la petición, el contexto de ejecución del CGI, etc.

3) El servidor ejecuta el programa CGI proporcionando los datos de entrada.

4) El programa se ejecuta escribiendo código HTML por la salida del proceso. Durante esta ejecución, puede obtener datos adicionales a partir de ficheros locales o una base de datos.

5) Finalmente, el servidor envía de vuelta la respuesta HTTP con los datos de salida del CGI.

Un ejemplo de programa CGI escrito en C++ sería el siguiente:

```
#include "stdafx.h"
#include <iostream>
#include <cstdlib>

int _tmain(int argc, _TCHAR* argv[])
{
   /* La siguiente salida indica al servidor el tipo
   de información de salida. En este caso enviamos
   texto HTML.*/
   std::cout<<"Content-type: text/html\n\n";
   //Se envía la página por la salida del programa.
   std::cout<<"<html><head><title>Ejemplo de CGI
   </title></head><body>";
   //Dentro de cuerpo, se muestra "Hola mundo!" siete veces
   entre etiquetas de cabecera mediante un bucle"
   for(int i = 7 ; i > 0; i--)
   {
      std::cout<<"<h"<<i<<">Hola mundo!</h"
      <<i<<">";
   }
   std::cout<<"</body></html>";
   return 0;
}
```

La salida generada por el programa tendría este aspecto:

```
Content-type: text/html

<html>
<head><title>Ejemplo de CGI</title></head>
<body>
   <h7>Hola mundo!</h7>
   <h6>Hola mundo!</h6>
   <h5>Hola mundo!</h5>
   <h4>Hola mundo!</h4>
   <h3>Hola mundo!</h3>
   <h2>Hola mundo!</h2>
   <h1>Hola mundo!</h1>
```

```
</body>
</html>
```

Un programa CGI puede llevar a cabo cualquier operación que permitan el lenguaje y el entorno utilizados dentro del servidor, y se puede utilizar, por ejemplo, para consultar una base de datos, para trabajar con archivos del servidor, etc.

Por otro lado, los CGI presentan un inconveniente, ya que cada petición requiere la ejecución de un programa en el servidor que finaliza al enviar la respuesta. Esto exige una sobrecarga al llevar a cabo la creación del proceso y su contexto, la preparación del entorno de ejecución y la destrucción del mismo. Estas tareas pueden sobrecargar el servidor, o incluso saturarlo en el peor de los casos.

3.2. ISAPI

Microsoft creo la interfaz ISAPI[9] para resolver el problema de sobrecarga respecto a la tecnología CGI.

(9)ISAPI es la sigla de *Internet server application programming interface.*

Básicamente, ISAPI permite alojar la funcionalidad de los programas en DLL[10] en lugar de en ejecutables.

(10)DLL es la sigla de *dynamic-link library* ('biblioteca de enlace dinámico'). Término que se refiere a archivos con código fuente ejecutable.

El código de una extensión ISAPI se carga en memoria una sola vez, lo que permite la ejecución del mismo código por diferentes peticiones. Los parámetros y el contexto de ejecución los gestiona el servidor de aplicaciones. Las aplicaciones basadas en ISAPI se ejecutan bajo IIS[11].

(11)IIS es la sigla de *Internet information services* ('servicios de información de Internet').

ISAPI también presenta algunos inconvenientes, como la complejidad de desarrollo y de depuración, que son solventados por posteriores tecnologías de servidor que se explican en los siguientes subapartados.

3.3. ASP

ASP[12] es una tecnología de Microsoft para generar páginas dinámicas bajo un servidor de aplicaciones IIS.

(12)ASP es la sigla de *active server pages* ('páginas de servidor activas').

IIS

Inicialmente las páginas ASP se ejecutaban dentro del servidor de aplicaciones IIS. En la actualidad existen diversos servidores web que permiten ofrecer páginas ASP.

ASP ofrece la ventaja de poder mezclar código Java-Script o VBScript con HTML en el mismo documento mediante marcas que diferencian el código incrustado. Al solicitarse una página, el servidor interpreta las partes de código VBScript y las sustituye por la salida que generan. El resultado enviado a los clientes es una página HTML.

VBScript

VBScript (Visual Basic Script) es un lenguaje interpretado basado en Visual Basic.

Un ejemplo sencillo de página ASP podría ser el siguiente:

```
<html>
<body>
<%
response.write("Hola Mundo!")
%>
</body>
</html>
```

El resultado del ejemplo es una simple página que muestra el texto "Hola Mundo!".

ASP ofrece ventajas sobre CGI, ya que crear y mantener aplicaciones ASP resulta mucho más sencillo. Por otro lado, ASP también ofrece las ventajas de reutilización de código y procesos de ISAPI.

3.4. PHP

PHP[13] es un lenguaje interpretado y utilizado generalmente para el desarrollo de páginas web dinámicas.

[13]PHP es la sigla de *hypertext preprocessor* ('preprocesador de hipertexto').

Fue creado por Rasmus Lerdorf en 1994 y continuado por The PHP Group. Este grupo define el estándar del lenguaje al no disponer de uno oficial. PHP es un lenguaje basado en herramientas con licencia de software libre.

Al igual que las páginas ASP, PHP se puede incrustar dentro de páginas HTML para generar páginas dinámicas. El servidor que hospeda las páginas debe disponer de un módulo para la interpretación de este lenguaje.

Servidor de PHP

Generalmente, se utiliza el servidor de aplicaciones Apache con el módulo PHP.

PHP ha tenido un éxito generalizado en el desarrollo web gracias a su sencillez, al hecho de que es gratuito y a que dispone de un gran conjunto de extensiones que permiten el acceso a diferentes proveedores de bases de datos y otras tareas habituales, como expresiones regulares, soporte XML, *sockets*, etc.

La siguiente página PHP muestra cinco veces el texto "Hola mundo" de forma dinámica:

```
<html>
   <head>
   <title>Ejemplo PHP </title>
```

```
    <meta http-equiv="Content-Type" content="text/html;
    charset=ISO-8859-1">
    </head>

    <body>
       <?php
          $makeHTML = '<h%d>Hola mundo</h%d>';
          for($i = 6; $i > 1; $i--){
             printf($makeHTML,$i,$i);
          }
       ?>
    </body>
</html>
```

El código en el interior de la etiqueta `<?php...?>` es interpretado por el servidor al solicitar la página. La salida generada se insertará en el interior del cuerpo de la página mostrando el resultado siguiente:

```
<html>
   <head>
   <title>Ejemplo PHP </title>
   <meta http-equiv="Content-Type" content="text/html;
   charset=ISO-8859-1">
   </head>

   <body>
      <h6>Hola mundo</h6>
      <h5>Hola mundo</h5>
      <h4>Hola mundo</h4>
      <h3>Hola mundo</h3>
      <h2>Hola mundo</h2>
   </body>
</html>
```

3.5. Java EE

Java Platform Enterprise Edition es una plataforma independiente y centrada en el lenguaje Java para el desarrollo y la ejecución de aplicaciones distribuidas. Fue creada y desarrollada por Sun Microsystems y posteriormente adquirida por Oracle.

Aplicación distribuida

Una aplicación distribuida consiste en un conjunto de componentes que se ejecutan en entornos separados, generalmente conectados por una red.

Java EE permite desarrollar aplicaciones portables, robustas y escalables con una arquitectura distribuida en capas bien diferenciadas y la interacción entre éstas. Dispone de un conjunto de servicios, API y protocolos, y comprende un amplio conjunto de tecnologías y especificaciones para desarrollar aplicaciones web, acceso a bases de datos, servicios web, soporte para XML, correo electrónico, etc.

Acceso a bases de datos

Java dispone de su propia API de acceso a bases de datos, JDBC (*Java database connectivity*).

Java EE está orientado a aplicaciones de gran envergadura, y sus especificaciones permiten la comunicación, la distribución y la reutilización de los componentes. Al realizar una separación del código en niveles bien diferenciados, también se simplifica el mantenimiento de las aplicaciones.

3.6. ASP.NET

ASP.NET es una tecnología para crear aplicaciones web interactivas y dinámicas. Forma parte de la plataforma .NET de Microsoft y fue introducida en el año 2000 como respuesta a la plataforma Java EE.

Plataforma .NET

La plataforma .NET es un entorno (*framework*) de Microsoft para el desarrollo de aplicaciones web, aplicaciones de escritorio, servicios, etc. Incluye un conjunto de lenguajes y bibliotecas de clases.

La ejecución de las aplicaciones .NET se realiza bajo una máquina virtual comparable a la JVM.

ASP.NET ofrece un modelo de programación muy similar al de las aplicaciones de escritorio mediante las páginas ASP.NET. Éstas están compuestas por una parte visual, que se ejecuta en el lado del cliente, y una parte lógica, que se ejecuta en el lado del servidor para responde a las acciones y los eventos desde el navegador cliente.

Las páginas ASP.NET tienen un aspecto muy parecido a las páginas HTML y pueden incorporar controles y código *script* de servidor. Estos elementos se ejecutarán en el servidor para generar código HTML antes de que se sirva la página.

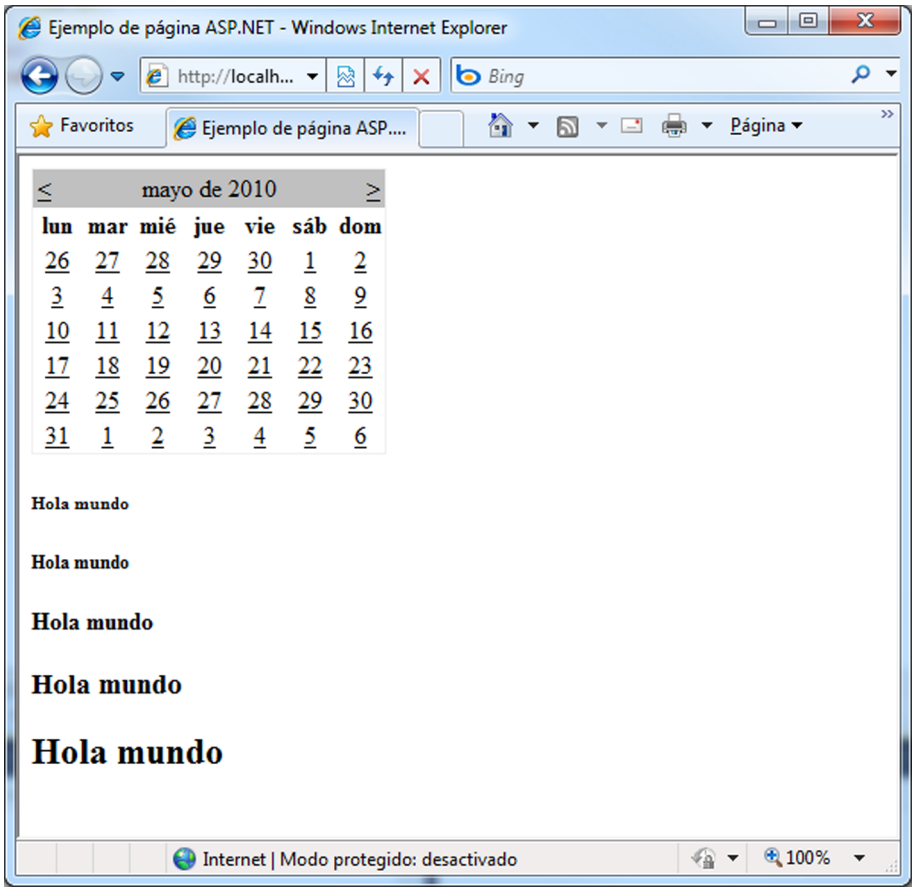
El código siguiente corresponde a una página ASP.NET que contiene un control de servidor y un *script*:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default2.aspx.cs"
Inherits="Default2" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
   <title>Ejemplo de página ASP.NET</title>
</head>
<body>
   <form id="form1" runat="server">
   <div>
```

```
        <asp:Calendar ID="Calendar1" runat="server"></asp:Calendar>
        <% for (int i = 6; i > 1; i--)
           {%>
           <h<%=i%>>Hola mundo</h<%=i%>>
           <%} %>
    </div>
    </form>
</body>
</html>
```

La página tiene un aspecto muy parecido al de un documento HTML corriente salvo por la directiva `<%@ Page %>` en la parte superior. Ésta indica algunas características de la página, como el lenguaje de programación utilizado (en este caso C#) o el fichero en el que reside el código. En el cuerpo, se encuentra el bloque `<form>`, en el que se hallan un control de servidor y un *script*. El control de servidor corresponde a la etiqueta `<asp:Calendar>`, que permite mostrar un calendario. Posteriormente, entre etiquetas `<% %>`, se encuentra el código *script* de la página. Éste simplemente realiza un bucle para escribir "Hola mundo" cinco veces, entre cabeceras de `<h6>` a `<h2>`. El resultado de la página se muestra en la figura 5.

Figura 5. Resultado del ejemplo de ASP.NET



Si comprobamos el código fuente, podemos ver que el control de servidor y el *script* se han transformado en código HTML para que el navegador del cliente los pueda visualizar.

3.7. Ruby on Rails

Ruby es un lenguaje de programación interpretado y orientado a objetos que se inspira en otros lenguajes, como Python, Perl y Smalltalk. El resultado es un lenguaje de programación conciso y legible pero potente al mismo tiempo. Por otro lado, Rails es un entorno de desarrollo[14] basado en el lenguaje Ruby que facilita el desarrollo, el despliegue y el mantenimiento de aplicaciones web.

(14)En inglés, *framework*.

Ruby es un lenguaje de código abierto y gratuito creado por Yukihiro Matsumoto. Matsumoto inició el desarrollo de Ruby en 1993 y lo presentó finalmente en 1995. Más tarde, surgió el entorno de desarrollo Rails, en julio de 2004, creado inicialmente por David Heinemeier y mantenido en la actualidad por una comunidad de programadores.

Rails está construido sobre dos principios muy sólidos:

1) No te repitas (DRY[15]**)**. En la aplicación, todo concepto debe definirse sólo una vez. Esto evita la duplicación de la información.

(15)DRY es la sigla de *don't repeat yourself* ('no te repitas').

2) Convención sobre configuración. Sólo será necesario definir aquellas configuraciones que no sigan las convenciones de Rails.

Estos dos principios hacen que Ruby on Rails sea un entorno de desarrollo ágil válido para casi todas las necesidades del mercado.

Algunas de las características más importantes de Rails son las siguientes:

a) El desarrollo de toda aplicación sigue el patrón Modelo Vista Controlador (MVC), que divide la aplicación en tres capas lógicas bien diferenciadas:

- **Modelo**: capa encargada de la gestión de los datos.
- **Vista**: capa encargada de la interacción con el usuario mediante la interfaz gráfica.
- **Controlador**: capa que recibe las acciones del usuario y lleva a cabo la obtención y la representación de datos.

b) Rails dispone de una capa ORM para el acceso a las bases de datos.

ORM

El *object-relational mapping* (ORM) podría traducirse como 'mapeo objeto-relacional'. Se trata de un patrón de diseño que relaciona de forma automática clases de entidad con tablas en una base de datos relacional.

c) Las vistas se basan en HTML y código Rails incrustado para ofrecer dinamismo del lado del servidor.

d) Rails dispone de un conjunto de controles y funciones AJAX[16] basadas en el entorno de desarrollo Prototype.

Prototype

Prototype es una biblioteca escrita en Java-Script que facilita el desarrollo de páginas web dinámicas con AJAX. Ofrece funciones que minimizan las diferencias entre las implementaciones del DOM por los navegadores.

(16)AJAX es la sigla de *asynchronous Java-Script and XML* ('Java-Script asíncrono y XML').

4. RIA

Las RIA[17] son aplicaciones web que ofrecen características adicionales sobre las aplicaciones tradicionales basadas en páginas dinámicas. Estas características adicionales se basan en mejoras gráficas, de interacción y de usabilidad.

(17)RIA es la sigla de *rich Internet applications* ('aplicaciones de Internet enriquecidas').

Las RIA ofrecen resultados comparables a las aplicaciones de escritorio. Según la tecnología utilizada, se requiere la instalación de un complemento en el navegador para poder interpretar las interfaces recibidas del servidor, como es el caso de Silverlight y JavaFX.

Generalmente, las aplicaciones web requieren recargar la página completa para mostrar nueva información por insignificante que sea. Durante el uso de la aplicación, se produce un tráfico innecesario de páginas entre cliente y servidor, lo que degrada la interactividad y la experiencia del usuario. Por su parte, las aplicaciones RIA permiten mantener una comunicación asíncrona con el servidor para el intercambio de información sin interferir con la interacción del usuario.

A continuación, se enumeran algunas de las tecnologías que permiten desarrollar aplicaciones RIA.

4.1. AJAX

El modelo tradicional de una aplicación web se basa en la recarga total de una página para mostrar nueva información. AJAX[18] implica el uso conjunto de diferentes técnicas y tecnologías que permiten mantener una comunicación asíncrona con el servidor y así actualizar partes del documento de forma dinámica.

(18)AJAX es la sigla de *asynchronous Java-Script and XML* ('Java-Script asíncrono y XML').

Aunque con anterioridad ya se hacía uso de estas técnicas y tecnologías, hasta el año 2005 no recibieron el nombre de AJAX. El uso de AJAX se popularizó en parte gracias a Google, con sus aplicaciones GMail, Google Docs y otras.

Las principales tecnologías utilizadas por AJAX son XHTML, Java-Script, XML y CSS.

4.2. Adobe Flex

Adobe Flex es un entorno para el desarrollo y el despliegue de aplicaciones RIA basadas en su reproductor Flash Player. Surge en el 2004 como un conjunto de tecnologías propietarias de Macromedia entre las que se encuentran ActionScript y MXML[19].

(19)MXML es la sigla de *multimedia extensible markup language* ('lenguaje de marcado extensible multimedia').

MXML

MXML es un lenguaje para describir interfaces de usuario. Al ser textual, los desarrolladores pueden leer y modificar las interfaces con un simple editor de texto.

ActionScript es un lenguaje de *script* orientado a objetos para implementar la lógica de la aplicación. Por otro lado, MXML (*multimedia extensible markup language*) es el lenguaje de marcado basado en XML que se emplea para describir las interfaces de usuario. MXML puede semejarse a XHTML, y ActionScript, a Java-Script.

Flex permite crear aplicaciones sofisticadas con interfaces atractivas y una gran interactividad: arrastrar y soltar objetos, hacer listas ordenables y animaciones, etc. Las aplicaciones se ejecutan por medio del navegador, que requerirá el complemento de Flash Player.

4.3. JavaFX

JavaFX engloba un conjunto de tecnologías para desarrollar aplicaciones RIA ejecutables en una gran variedad de dispositivos. Actualmente es posible desarrollar aplicaciones JavaFX para escritorio, navegadores y dispositivos móviles.

Aplicaciones JavaFX

Sun planea implementaciones de JavaFX para otros dispositivos, como reproductores DVD o Blu-ray, televisiones, reproductores ligeros (como los que podemos encontrar en los coches), etc.

Las aplicaciones JavaFX utilizan el lenguaje JavaFX Script. Éste fue desarrollado por Sun Microsystems para implementar de forma rápida interfaces de usuario para la plataforma Java. Las aplicaciones JavaFX pueden hacer uso de cualquier clase Java de forma directa.

4.4. Silverlight

Silverlight es un entorno de desarrollo para construir aplicaciones RIA hospedadas en servidores. Al igual que Adobe Flex y JavaFX, Silverlight requiere tener instalado un complemento en el navegador. Cuando el navegador carga una página que incluye código Silverlight, el complemento correspondiente se ejecuta para renderizar e interpretar el contenido en una región específica de la página.

Con Silverlight se pueden crear páginas interactivas con gráficos, especialmente gráficos vectoriales, y elementos multimedia.

Silverlight está basado en un subconjunto de la plataforma .NET. Para programar las interfaces, se utiliza un lenguaje declarativo basado en XML llamado XAML[20]. La lógica se lleva a cabo mediante los lenguajes C# o Visual Basic.

(20)XAML es la sigla de *extensible application markup language* ('lenguaje extensible de marcado para aplicaciones'). XAML es un lenguaje de marcas basado en XML para describir interfaces.

Microsoft dispone de la herramienta Blend, que permite diseñar interfaces de forma gráfica y rápida. La lógica, la compilación y el despliegue pueden llevarse a cabo mediante las herramientas de Visual Studio.