Uso de AJAX en el desarrollo de aplicaciones web con Java y Php

Use of AJAX in Web Application Development with Java and Php

Sonia Alexandra Pinzón Núñez*

Fecha de recepción: 1 de agosto de 2012
Fecha de aceptación: 3 de septiembre de 2012

Resumen

El desarrollo web se ha multiplicado de forma que existen aplicaciones para realizar cualquier función, desde una simple página con un servicio que activa una calculadora o un programa de conversiones hasta aplicaciones que muestran nuestra ubicación por medio de GPS en el caso de Google Maps (Servicio de Google).

Estas aplicaciones permiten que un usuario pueda hacer uso de dicha funcionalidad desde cualquier computador. El proceso es sencillo: solo se utiliza un navegador para ingresar a la página requerida que se encuentra alojada en un servidor y activar la función que necesite para que se genere la respuesta correspondiente. Pero en ocasiones los tiempos de respuesta son altos y si el usuario activa varias solicitudes este proceso puede ser una molestia ya que se debe recargar la página cada vez que haya una solicitud. Para dar solución a este tipo de inconvenientes se han agrupado varias tecnologías que permiten ejecutar la mayoría de los procesos desde el cliente (recurso del usuario que hace la solicitud al servidor) y solo ejecutar las funciones más relevantes en el servidor, disminuyendo los tiempos de respuesta. Esto se hace utilizando AJAX (AsynchronousJavaScript +XML).

* Ingeniera de Sistemas y especialista en Multimedia Educativa, Universidad Antonio Nariño de Bogotá. Especialista en Educación en Tecnología, Universidad Distrital. Magíster en Ciencias de la Información y las Telecomunicaciones, Universidad Distrital. Docente investigadora del grupo Metis, adscrito a la Facultad Tecnológica de la Universidad Distrital Francisco José de Caldas. spinzon@udistrital.edu.co

En el presente artículo se describen las características y tecnologías que integra AJAX y se presentan dos ejemplos de su uso en los lenguajes de programación Java y Php, los cuales son muy utilizados en la implementación de aplicaciones web como una alternativa para el mejoramiento de la funcionalidad e interacción para los usuarios.

Palabras clave: aplicaciones cliente servidor, multicapa, comunicación asíncrona, JavaScript.,

Abstract

Web develop has been multiplied in such way so that nowadays it exist applications to performed any function, from a simple page with a service to activate a calculator or a program conversion to applications that shows our location over the earth by using GPS as in the case of Google Maps (Google Services).

These applications allow users to do use of those functionalities from any computer. The whole process is quite simple, it is just necessary a browser to access the require web page, which is hosted in a server and activates the function which is request so that the answer could be generated. However, in some occasions, time response is high and also it is also troublesome when the user require many functions at the same time because the web page has to be reloaded whenever there is a request.

So that we can solve this kind of problems several technologies have been grouped, which allow execute most of the processes in the client side (user resource that makes the server request) and execute the most relevant functions in the server side, decreasing response times, the process described below is possible by implementing AJAX (AsynchronousJavaScript +XML).

This paper describes the features and AJAX integration technologies and is also presented two different examples of how it is apply using Java and Php programming languages, which are widely used in the implementation of Web Applications as an alternative to enhance functionality and interaction for the users.

Key words: Client server applications, asynchronous communications, JavaScript, multilayer.

1. Introducción

Desde su creación, el objetivo de internet ha sido intercambiar información rápidamente. Este objetivo se ha transformando de tal modo que los usuarios no solo buscan o transfieren información, sino que también pueden realizar varias tareas, como compartir, publicar, comunicarse con otros usuarios e incluso poner a disposición aplicaciones en áreas específicas, lo cual se ha mejorado mediante el desarrollo de sitios web y de las tecnologías asociadas.

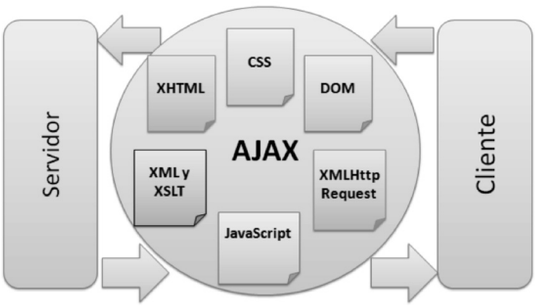
Normalmente el proceso que realiza el usuario cuando accede a una página es activar o seleccionar opciones o elementos que producen una respuesta, que es generada por un servidor al ejecutar acciones de forma transparente para dicho usuario. Inicialmente esta tarea no tenía ningún problema ya que las operaciones que debían ser ejecutadas no eran tan complicadas, pero actualmente las aplicaciones web son más robustas y utilizan gran cantidad de recursos que permiten procesar bastante información y poner a disposición múltiples servicios, lo que en algunas ocasiones hace que el usuario tenga que esperar a que el servidor procese la solicitud y genere la respuesta. Esta espera constituía una desventaja, hasta que apareció AJAX, un conjunto de tecnologías que usa el lenguaje JavaScript incorporando metadatos por medio de XML, cuya integración hace que la ejecución se realice en el cliente: manteniendo una comunicación asíncrona con el servidor facilita que los procesos se ejecuten más rápido sin necesidad de recargar las páginas, mejorando así la funcionalidad y los tiempos de respuesta de estas [1].

Existen muchos lenguajes que son utilizados por los desarrolladores para implementar aplicaciones web, entre ellos Java y Php, los cuales pueden hacer uso de AJAX para mejorar el uso de los recursos que se encuentran en internet.

2. Tecnologías que integran AJAX

AJAX es el resultado de la unificación de varias especificaciones y tecnologías que se han definido a partir de los estándares que el consorcio W3C ha planteado para el diseño web; estas integran elementos como lenguajes que permiten crear las páginas web, por ejemplo HTML y XHTML, el conjunto de reglas de estilo que permite mejorar y separar la apariencia de estas páginas de su contenido usando hojas de estilo CSS, lenguajes que facilitan la incorporación de metadatos como XML y XSLT, elementos que permiten hacer la transferencia asíncrona de datos entre el cliente y el servidor por medio del objeto XMLHttpRequest y el uso de los *scripts* que agregan la funcionalidad requerida para ejecutar las peticiones por medio de JavaScript. En la figura 1 se describen brevemente algunas características de dichas tecnologías.

Figura 1. Tecnologías que integran AJAX



Fuente: elaboración propia.

XHTML eXtensible Hyper Text Markup Language (lenguaje extensible de marcado de hipertexto): es un lenguaje de etiquetas que sirve para crear páginas web, aunque mejora la

estructura que se ha definido en el lenguaje HTML.

Como ya se conoce, todas las páginas están hechas en lenguaje HTML (HyperText Markup Language), pero dado el desarrollo del diseño web y la necesidad de cubrir nuevos requerimientos de los usuarios, las entidades que definen las pautas en esta área, como el consorcio W3C, pretenden crear páginas con significado, es decir que las páginas no solo sean un conjunto de elementos, como texto, imágenes y vínculos, sino que estén asociadas a un significado y además permitan separar claramente las características que definen la apariencia y la información que se va a presentar. Por lo cual se desarrolló el estándar XHTML, que es una adaptación de HTML agregando las capacidades del lenguaje XML, el cual permite que las páginas contengan elementos con una estructura bien definida y se puedan configurar para ser presentarlas desde cualquier dispositivo o navegador haciéndolas más flexibles.

CSS Cascading Style Sheets (hojas de estilo en cascada): este estándar, también definido por el W3C, está muy relacionado con los lenguajes HTML y XHTML, ya que se encarga de definir el estilo o apariencia de la página web; permite manipular los formatos de los elementos teniendo en cuenta características como color, posición y fuentes y mejora la presentación de la información que contiene la página de acuerdo con el tipo de dispositivo donde se va a visualizar, por lo cual es fácil acceder a las páginas desde un teléfono celular, una tableta o un computador personal sin que la presentación se altere [2].

Las hojas de estilo facilitan el mantenimiento de las páginas web ya que se definen de forma independiente y separada de la estructura de estas y se convierten en una buena práctica para el desarrollo web.

DOM Document Object Model (Modelo de Objetos del Documento): el W3C: "define la estructura lógica de los documentos y el modo en que se accede y manipula" [3]; es un API (Interfaz de Programación de Aplicaciones) que permite a los desarrolladores web realizar operaciones sobre una página también llamada documento, para agregar, modificar o eliminar cualquier objeto o contenido que contenga esta.

Por lo general la estructura de los objetos dentro de la página está definida por una jerarquía de objetos que tienen relacionadas características y funciones que hacen más fácil su uso desde cualquier lenguaje de programación.

XML y XSLT XML Stylesheets Language for Transformation (Lenguaje de Estilo Extensible): es una especificación del W3C que permite definir una plantilla para transformar documentos XML en otro tipo de documentos, como páginas Web en HTML y XHTML. Con este lenguaje se pueden crear plantillas que contienen un conjunto de reglas de transformación aplicadas al documento XML, que también se llaman reglas de estilo XSLT; estas luego son analizadas por un procesador de XSLT, el cual hace la transformación para generar el archivo correspondiente para visualizarlo en un dispositivo específico [4]. Este tipo de transformaciones favorece separar el contenido de la presentación.

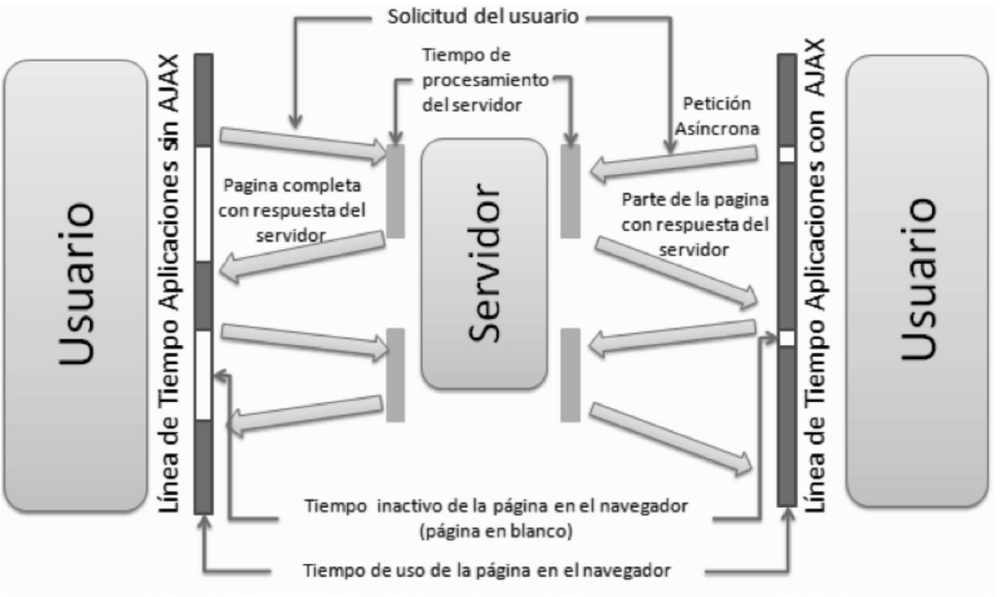
XML HttpRequest (Extensible Markup Language/Hypertext Transfer Protocol): es un objeto que permite hacer peticiones al servidor de forma asíncrona y sin tener que actualizar las páginas en las que se hacen dichas peticiones [5]. Este objeto está definido en todos los navegadores para realizar las acciones de transferencia de datos.

JavaScript: es un lenguaje basado en *script* desarrollado por Netscape y Sun Microsystems, que permite agregar funcionalidad y hacer más dinámicas las páginas web. Los *scripts* se pueden insertar en el código HTML y este es interpretado por cualquier navegador, por lo cual se dice que JavaScript se implementa y ejecuta del lado del cliente, pero también se puede usar del lado del servidor como Server-side JavaScript o SSJS. Aunque su nombre hace que se relacione con el lenguaje de programación Java, no son lo mismo, su sintaxis es similar al lenguaje C y permite definir prototipos de objetos sin necesidad de definir tipos de datos [6].

3. Funcionamiento de AJAX respecto a los tiempos de respuesta

Como ya se indicó, el objetivo principal de AJAX es que el usuario pueda disponer del contenido de una página sin que los tiempos de respuesta hagan que la página esté inactiva por mucho tiempo. En la figura 2 se puede observar la línea de tiempo que muestra los tiempos de uso, procesamiento y respuesta que se generan cuando el usuario realiza una solicitud desde el navegador hacia el servidor, los cuales varían cuando las aplicaciones web implementan, o no, AJAX.

Figura 2. Línea de tiempos de aplicaciones con AJAX



Fuente: elaboración propia.

En el caso en que el usuario accede a una aplicación que no utiliza AJAX, esta es enviada desde el servidor al navegador donde el usuario puede visualizar y manipular los elementos de la o las paginas (tiempo de uso de la página en el navegador), y cuando el usuario activa una acción que requiera ser ejecutada por el servidor, este deberá actualizar la página nuevamente

hasta que se genera la nueva respuesta (tiempo de procesamiento del servidor). Este proceso incluye el momento en el que se envía la solicitud hasta que la página completa es visualizada nuevamente en el navegador y depende del tipo de operación que deba resolver el servidor, lo que genera un tiempo de inactividad en el navegador en el cual el usuario no recibe ninguna información y se mantiene una página en blanco (tiempo inactivo en el navegador); es en este tiempo cuando se presentan las demoras que disminuyen la eficiencia de este tipo de aplicaciones.

Por otra parte, cuando se implementan aplicaciones con AJAX, este proceso tiene algunas variaciones. Inicialmente el servidor también envía la página completa y el usuario puede actuar sobre esta, de la misma manera que en el caso anterior, pero cuando el usuario activa una o varias acciones, estas se realizan con peticiones asíncronas al servidor generando un proceso que se puede decir que se ejecuta en paralelo, es decir, mientras el servidor se encarga de procesar la parte de la página que requiere ser actualizada, en el navegador se mantiene el resto de la página para que el usuario pueda seguir usando la aplicación. Cuando se obtiene la respuesta del servidor se podrá actualizar la sección de la página que mostrará los resultados; este proceso es implementado en Javascript, el cual permite activar un manejador de eventos que verifica cuándo se ha generado dicha respuesta para realizar la actualización. Este proceso hace que el tiempo de procesamiento en el servidor no interfiera con el tiempo de uso de la página en el navegador, por lo cual hace más rápidos los tiempos de respuesta.

4. Implementación de AJAX en el desarrollo de aplicaciones web

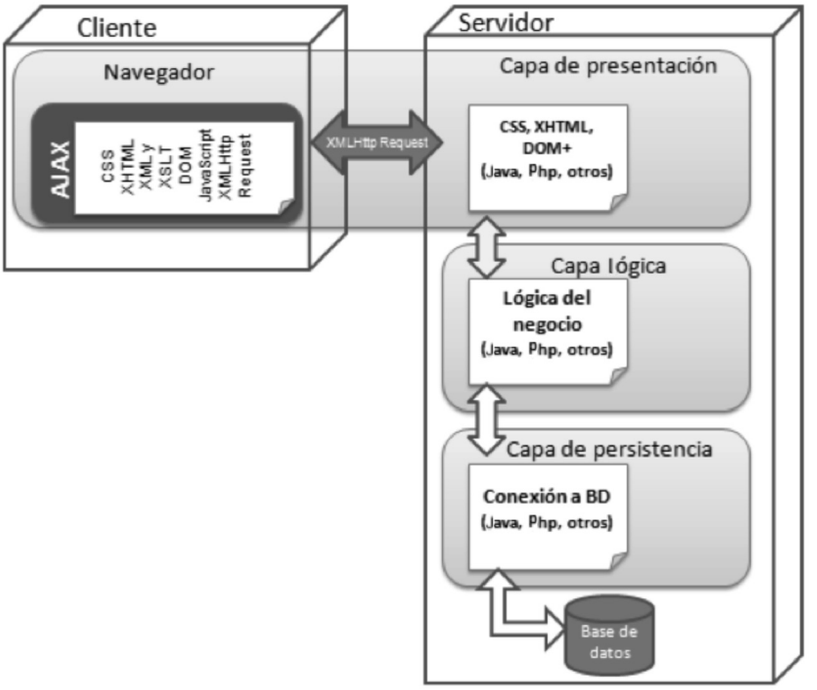
Para explicar cómo se puede implementar AJAX en una aplicación web se desarrollará un ejemplo que permita mostrar y consultar registros de una base de datos en Mysql que contiene datos de vehículos cuyo criterio de búsqueda es la marca, utilizando los lenguajes Php y Java para presentar las dos perspectivas.

4.1. Arquitectura de la aplicación web

Inicialmente se deberá definir la arquitectura de la aplicación; por lo general la arquitectura de capas es una buena alternativa para la implementación de aplicaciones cliente-servidor, ya que permite separar los elementos que acceden a los datos de la lógica y a la vez de la presentación que corresponderá a los elementos que el usuario visualizará en la página web. Adicionalmente, el desarrollo se hará bajo el paradigma orientada a objetos dado que los lenguajes Java y Php permiten este tipo de programación y es lo más recomendable si se piensa realizar aplicaciones robustas y de calidad.

La capa de presentación estará definida por la incorporación de una hoja de estilo que definirá la apariencia de los elementos que se visualizarán en la página HTML y la definición del objeto XMLHttpRequest para el uso de AJAX que permitirá realizar las peticiones desde el cliente al servidor; se podría decir que esta capa se ejecutará en el cliente y en el servidor, dado que en el servidor es donde estará alojada la aplicación, y cuando el navegador solicita la página, esta es descargada completamente (figura 3).

Figura 3. Arquitectura de la aplicación web



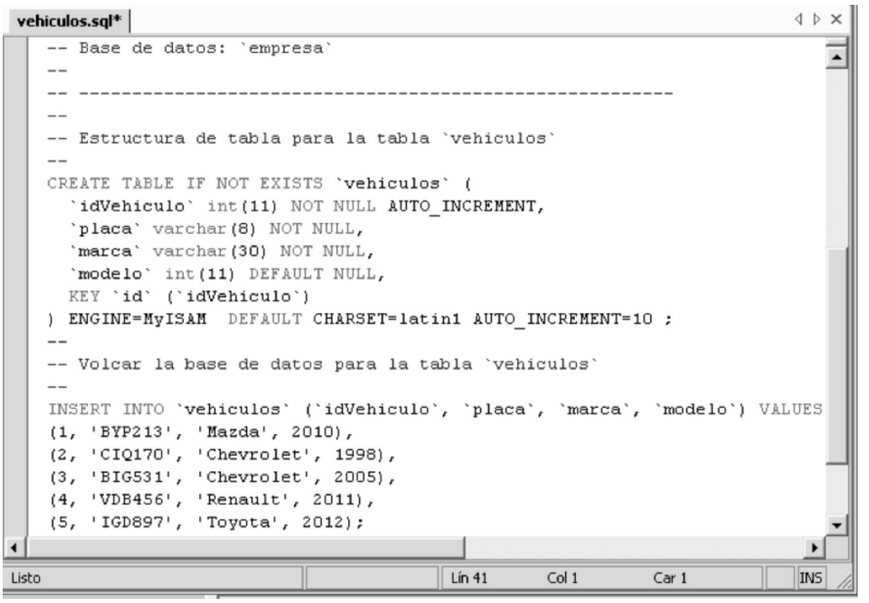
Fuente: elaboración propia.

La capa lógica contendrá las clases que incorporan la funcionalidad para realizar los procesos de filtrado y consulta de la aplicación.

La capa de persistencia contendrá la conexión a la base de datos que permitirá la gestión de la información que será manipulada por la aplicación. Para ello el uso de Mysql puede ser una buena opción ya que es un manejador de base de datos libre y tiene un buen soporte de datos.

La estructura de la base de datos definida para el ejemplo se puede importar del siguiente código en formato SQL, donde se define el nombre de la base de datos como *empresa* que contiene una tabla llamada *vehiculos* cuyos campos son idVehiculo, placa, marca y modelo; además se agregaron varios registros inicialmente para que se presenten en la página al ser mostrada por primera vez en el navegador (figura 4).

Figura 4. Vista de archivo SQL, estructura de la base de datos de la aplicación



Fuente: elaboración propia.

4.2. Capa de presentación

La propuesta de visualización para la página de ejemplo se muestra en la figura 4; simplemente contendrá una caja de texto para ingresar el criterio de filtrado y el botón buscar que activará la solicitud de ejecución. Para el diseño de esta se usará el lenguaje HTML el cual permite utilizar un conjunto de capas o contenedores para agrupar los elementos con el fin de mejorar la visualización, a las cuales se les definirán las reglas de estilo por medio de CSS (figura 5).

Figura 5. Vista de página web ejemplo consulta de vehículos

Fuente: elaboración propia.

Hoja de estilo: para que la página tenga la apariencia que se muestra en la figura 5 se pueden definir varias reglas de estilo; algunas de estas se proponen a continuación. Cada una define la ubicación y formato de los elementos dentro de la página. Para ello se han determinado tres clases que se definen en capas tipo div:

#*encabezado* es la clase que define el formato que agrupa la imagen de AJAX y el título *Consulta BD con AJAX*.

#*contenido* es la clase que define los formatos de la capa que contiene la etiqueta "Digite la marca", la caja de entrada donde el usuario digitará el criterio de búsqueda y el botón que activará la acción de filtrado.

Y finalmente la clase llamada #*vistaConsulta* que implementa el formato de la capa donde se mostrará la tabla con el conjunto de registros que genera la consulta. Esta es la capa o parte de la página que será actualizada por medio de AJAX.

```
/* Hoja de estilos CSS para ejemplo AJAX */

    body {
    font-size: 0.875em;
    font-weight: normal;
    font-family: Tahoma, Arial, Helvetica, Verdana, sans-serif;
    background: #edeae8;
    color: inherit;
    }
    #encabezado, #contenido{
    float: left;
    width: 100%;
    text-align: left;
    }
    #contenido{
    float: left;
    margin: 0 0 0 2em;
    }
    #vistaConsulta{
    float: left;
    width: 100%;
    padding: 2em 0 0 0;
    margin: 0 0 0 2em;
    text-align: left; }
    img{float: left;
    margin: 0 2em 0 2em; }
    h1{ font-size: 1.4em; }
    h2{ font-size: 1.1em; }
    h1, h2 {
    font-weight: normal;
    margin: 1em 0 2em 0;
    line-height: 100%;
    font-family: 'Oxygen', sans-serif; }
    #vistaConsulta table {
    margin: 0 0 1em 2em;
    border:1px solid #0000ff;
    width:80%;
    }
    th { font-weight: bold; }
    th, td {
    border: 1px solid #181e23;
    font-size: 1em;
    padding: 0.5em;
    }
    label, button, input {
    font-size: 1.125em;
    margin: 0 0.5em 0 0;
    width:6em;
    }
```

Página principal: corresponde a la vista; es el componente que le permite al usuario interactuar con la aplicación. Por lo general esta página tiene el nombre de índex y su extensión depende del lenguaje en que se ha desarrollado (*.php, *.jsp, *.aspx, entre otros). Básicamente el código utilizado para implementar esta página es el HTML. En la figura 6 se presentan los códigos que pueden implementarse en los lenguajes Php y Java que permiten generar una página con la interfaz que se ha definido en la figura 5.

Figura 6. Vista de código archivo index.php

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
5      <title>AJAX con php</title>
6      <meta charset=utf-8>
7      <meta name=description content="Ejemplo Ajax con lenguaje php">
8      <link rel=stylesheet type="text/css" href="css/estilo.css" media=screen>
9      <script type="text/javascript" src="lib/ajax.js">
10     </script>
11 </head>
12 <body>
13     <div id="encabezado">
14     <img src="imagenes/ajax.png" alt="Logo">
15         <h1>Consulta BD con AJAX</h1>
16     </div>
17     <div id="contenido">
18     <form name="ingresaVehiculo" action="Logica/Consulta.php"
19           onSubmit="enviar(); return false">
20         <label for="marca"> Ingrese la marca </label>
21         <input id="marca" name="marcaV" type="text" />
22         <input type="submit" name="buscar" value="Buscar" />
23     </form>
24     </div>
25     <div id="vistaConsulta">
26        <script> enviar();</script>
27     </div>
28 </body>
29 </html>
```

Fuente: elaboración propia.

Código en lenguaje Php: este código se almacena con el nombre de *index.php* e incorpora las reglas de estilo que se plantearon previamente en un archivo llamado *estilo.css*, tal como se observa en la línea 8. Adicionalmente se agrega un *script* de Java llamado *ajax.js* en la línea 9, que contiene el código para crear el objeto XMLHttpRequest y la función *enviar()* que será activada por el usuario desde el botón Buscar o al cargar la página, como se observa en las líneas 19 y 26.

Como se indicó, se están usando capas *div* para agrupar elementos de la página; una de las más importantes es la *div* identificada como *id=vistaConsulta*, en la línea 25, en la cual se mostrará siempre el resultado de la función *enviar()*, la que permite visualizar los datos generados por las consultas hechas por el usuario.

Código de un servlet en Java: se puede crear un *servlet* con el nombre *index.jsp*, se puede utilizar el mismo código de Php, ya que, como se ha dicho, esta página es en HTML; solo se agrega la línea 7 para importar la ruta del archivo que contiene la lógica de la aplicación y se modifica la línea 11 para cambiar el título de la página (figura 7).

Figura 7. Vista código archivo index.jsp

```
1  <%--
2      Document   : primero
3      Created on : 14-sep-2012, 10:58:17
4      Author     : Sonia Pinzón
5  --%>
6  <%@page contentType="text/html" pageEncoding="UTF-8"%>
7  <%@page import="Logica.Consulta"%>
8  <!DOCTYPE html>
9  <html>
10 <head>
11 <title>AJAX con java</title>
12 <meta name=description content="Ejemplo Ajax con lenguaje java">
13 <link rel=stylesheet type="text/css" href="css/estilo.css" media=screen>
14 <script type="text/javascript" src="lib/ajax.js"></script>
15 </head>
```

Fuente: elaboración propia.

Archivo del script AJAX: el archivo *ajax.js* es el *script* que se agrega en la página principal y, como ya se ha indicado, contiene la definición de las funciones de JavaScript que permiten realizar las siguientes tareas:

- Definir el objeto XMLHttpRequest.
- Definir la función para enviar la información al servidor.
- Validar y obtener la información en el cliente.

Definición objeto XMLHttpRequest: este objeto permite generar las peticiones asíncronas desde la página, aunque se obtiene de forma diferente en cada tipo de navegador, por lo cual es necesario validar la creación de este.

Dado que el objeto fue creado por Microsoft para la versión 5.0 del Internet Explorer utilizando un objeto ActiveX [7, p. 114], el cual permite crear objetos que pueden ser usados en internet de forma independiente de un lenguaje de programación, se usa el objeto *window.ActiveObject* para obtener el XMLHttpRequest en versiones menores de Internet Explorer 7.0, como se observa en las líneas 3 y 4 del código.

Por otra parte, para crear el objeto XMLHttpRequest en navegadores como Firefox, Opera, Safari, Opera, versiones de Internet Explorer superiores a 7.0, entre otros, se utiliza la forma nativa del mismo con el objeto *window.XMLHttpRequest* [8] empleado en las líneas 5 y 6.

En la figura 8 se muestra el código que corresponde a un formato estándar para realizar dicho proceso.

Figura 8. Vista del código función objetoAjax del archivo ajax.js

```
1  function objetoAjax(){
2          var xmlhr=false;
3           if(window.ActiveXObject){
4              xmlhr=new ActiveXObject("Microsoft.XMLHttp"); }
5          else if((window.XMLHttpRequest) || (typeof XMLHttpRequest)!=undefined){
6              xmlhr=new XMLHttpRequest();
7                  }
8          else{
9              xmlhr=null;
10         }
11       return xmlhr;
12 }
```

Fuente: elaboración propia.

5. Definir la función para enviar la información al servidor

La función enviar ejecuta varios métodos del objeto AJAX que permiten hacer la petición al servidor y enviar la información necesaria para realizar el proceso que el cliente ha solicitado (figura 9).

Figura 9. Vista del código función enviar del archivo ajax.js.

```
13 function enviar(){
14    divConsulta = document.getElementById('vistaConsulta');
15    divConsulta.innerHTML= '<img src="imagenes/reloj.gif">';
16    marcaV=document.getElementById('marca').value;
17    objAjax=objetoAjax();
18    objAjax.open("POST", "Consulta",true);
19    objAjax.onreadystatechange=validaEnvio;
20    objAjax.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
21    objAjax.send("marca="+marcaV+"&funcion=consultaMarca");
22 }
```

Fuente: elaboración propia.

Inicialmente se obtiene el contenedor o capa donde se presentarán los resultados del proceso de consulta y filtrado cuyo nombre es *vistaConsulta*; para ello se utiliza el DOM donde se accede al arreglo de objetos de la página. De esta forma se puede asignar a un objeto dentro de la función llamado *divConsulta*, tal como se observa en la línea 14; adicionalmente se obtienen los datos o información que será procesada por el servidor, en este caso el criterio de búsqueda que se encuentra en el control llamado *marca* dentro de la página; este también es manipulado con el DOM. Dicho contenido se asigna a una variable llamada *marcaV* con la instrucción que se observa en la línea 16, luego se invoca la función que crea el objeto AJAX en la línea 17 y posteriormente se abre la conexión desde la página y el archivo *consulta.php* que procesará la solicitud; para ello se utiliza el método

open() del objeto XMLHttpRequest, tal como aparece en la línea 18; este método requiere como mínimo dos parámetros: el primero corresponde al método de envío de los datos y el segundo contiene el URL del archivo con el que se desea establecer la conexión, en este caso *consulta.php*. El parámetro *true* se agrega para especificar que la conexión se hará de modo asíncrono, aunque por defecto se realizan este tipo de peticiones.

En Java la línea 18 cambia por la siguiente instrucción, la cual puede ser un *servlet* que procesa los datos de la petición:

```
objAjax.open("POST", "Logica/
consulta",true);
```

Después de activar la conexión se ejecuta el evento *onreadystatechange* y el método *setRequestHeader* utilizados para obtener el estado de la petición [9] y definir el formato de la cabecera de la petición [10], respectivamente. Por último se ejecuta el método *send* que envía los datos a la URL con la que se estableció la conexión utilizando el método *open*.

6. Validar y obtener la información en el cliente

El proceso de validación evalúa el estado de la petición por medio de las propiedades readyState y status del objeto XMLHttpRequest. La propiedad *readyState* permite identificar si la petición se ha enviado satisfactoriamente y se encuentra cargada en el servidor, en este caso tendrá el valor 4, y la propiedad *status* identifica si la respuesta del servidor es correcta cuando tiene el valor 200 [11]. El código que se puede utilizar para realizar este proceso se incluye en la función *validaEnvio* que se muestra en la figura 10.

Figura 10. Vista del código función validaEnvio() del archivo ajax.js

```
23  function validaEnvio(){
24      if (objAjax.readyState==4) {
25       if (objAjax.status==200) {
26         divConsulta.innerHTML = objAjax.responseText
27         document.getElementById('marca').value="";   }
28      }
29  }
```

Fuente: elaboración propia.

Cuando la respuesta del servidor es válida, simplemente se agrega al contendedor *divConsulta* para que se muestre en la página. La respuesta se obtiene de la propiedad *responseText* en formato de texto tal como se muestra en la línea 26, aunque también se puede usar *responseXML* cuando se utilizan servicios que transfieren archivos XML.

6.1. Capa lógica

La implementación de la lógica solo requiere definir las funciones que permitirán obtener los datos que se encuentran almacenados en la base de datos; para ello se crean las funciones consultaTodos(), consultaMarca() y mostrarDatos(). Cada una de estas funciones manipula un objeto instanciado de la clase *Vehiculo*, el cual contiene los métodos de conexión y ejecución de las consultas correspondientes. A continuación se muestra el código propuesto para implementar dichas funciones con los lenguajes Php y Java.

*Código Php archivo consulta.php***:** este código recibe los datos marca y función por el método post. El parámetro función se utiliza para validar qué función debe ser ejecutada, es decir, si se recibe el parámetro marca con el nombre de la función *consultaMarca()* se hace el correspondiente llamado, de lo contrario se ejecuta la función *consultaTodos()***.** Finalmente la función *mostrarDatos()* manipula el conjunto de registros que se obtiene de cualquiera de las funciones de consulta y lo visualiza dentro de una tabla; este es el resultado que se presenta en el contenedor *vistaConsulta* de la página del cliente (figura 11).

Figura 11. Vista del código archivo consulta.php

```php
<?php
$mar=$_POST["marca"];
$funcion=$_POST["funcion"];
include_once("Vehiculo.php");
if (function_exists($funcion)){consultaMarca($mar);}
sleep(2);
function consultaTodos(){
    $objV=new Vehiculo("","","");
    $lista= $objV->consultar();
    mostrarDatos($lista);
}
function consultaMarca($marca){
    $objV=new Vehiculo("","","");
    $lista= $objV->consultarMarca($marca);
    if (mysql_num_rows($lista)<>0){
        echo "Marca ".$marca." encontrada...";
        mostrarDatos($lista);   }
    else
    { echo "La marca ".$marca." no esta registrada...";
    consultaTodos();}
}
function mostrarDatos($lista){
    echo "<table><tr><th>Placa</th><th>Marca</th><th>Modelo</th></tr>";
    while($row = mysql_fetch_array($lista)){
    echo "<tr>";
    echo "<td>".$row['placa']."</td>";
    echo "<td>".$row['marca']."</td>";
    echo "<td>".$row['modelo']."</td>";
    echo "</tr>";    }
    echo "</table>";
 }
?>
```

Fuente: elaboración propia.

*CódigoJava del servlet Consulta***:** este archivo no es muy diferente del archivo consulta Php, solo incorpora las funciones necesarias para ejecutar la petición de servidor por medio de la función *processRequest*. En las figuras 12, 13 y 14 se presenta el código correspondiente.

Figura 12. Vista del código función processRequest del archivo consulta.jsp

```
1   package Logica;
2   import ...;
13  @WebServlet(name = "Consulta", urlPatterns = {"/Consulta"})
14  public class Consulta extends HttpServlet {
15  PrintWriter out;
16      /**...*/
26  protected void processRequest(HttpServletRequest request, HttpServletResponse response)
27              throws ServletException, IOException {
28      response.setContentType("text/html;charset=UTF-8");
29      out = response.getWriter();
30
31      try {
32          String marca=request.getParameter("marca").toString();
33          String funcion=request.getParameter("funcion").toString();
34          if (funcion.equals("consultaMarca")&&(!marca.equals("")))
35              consultaMarca(marca);
36          else
37              consultaTodos();
38          out.println("<html>");
39          out.println("<head>");
40          out.println("<title>Ajax con java</title>");
41          out.println("</head>");
42          out.println("<body>");
43      } finally {
44          out.close();
45      }
46  }
```

Fuente: elaboración propia.

Funciones consulta del *servlet*

Figura 13. Vista del código funciones de consulta del archivo consulta.jsp

```
47  public void consultaTodos(){
48      Vehiculo objV=new Vehiculo();
49      ResultSet lista= objV.consultaTodos();
50      mostrarDatos(lista);
51      objV.getConexion().cerrarconexion();
52     }
53  public void consultaMarca(String marc) {
54      Vehiculo objV=new Vehiculo();
55      ResultSet lista= objV.consultarMarca(marc);
56      try {
57          if(lista.first())
58              {out.println("<p>Marca " + marc +" encontrada... </p>");
59              lista.previous();
60              mostrarDatos(lista);}
61          else{
62              out.println("<p>La marca " + marc +" no esta registrada...</p>");
63              consultaTodos();  }
64      } catch (SQLException ex) {
65          Logger.getLogger(Consulta.class.getName()).log(Level.SEVERE, null, ex);
66      }
67      objV.getConexion().cerrarconexion();
68  }
```

Fuente: elaboración propia.

Función mostrarDatos del *servlet*

Figura 14. Vista del código función mostrarDatos del archivo consulta.jsp

```
69  public void mostrarDatos(ResultSet registros){
70      try{
71          ResultSet lista = registros;
72          out.println("<table border=\"1\">");
73          out.println("<tr><td>Id</td><td>placa</td><td>marca</td><td>Modelo</td></tr>");
74          while (lista.next()) {
75          out.println("<tr>");
76          out.println("<td>"+lista.getObject("idVehiculo")+"</td>");
77          out.println("<td>"+lista.getObject("placa")+"</td>");
78          out.println("<td>"+lista.getObject("marca")+"</td>");
79          out.println("<td>"+lista.getObject("modelo")+"</td>");
80          out.println("</tr>");
81          }
82          out.println("</table>");
83          out.println("</body>");
84          out.println("</html>");
85          }catch (Exception e)
86          {
87              out.println("Error "+e);
88          }
89      }
```

Fuente: elaboración propia.

Definición de clases: como ya se indicó, cada método de consulta crea un objeto de tipo Vehiculo, por lo tanto es necesario definir la clase que contendrá la información y la funcionalidad de dicho objeto que permitirán hacer una conexión con la base de datos y luego ejecutar la sentencia de consulta correspondiente. En el código que se presenta a continuación se hace el llamado directo a la conexión y a la función que ejecuta las sentencias, aunque lo más aconsejable es utilizar patrones de acceso a datos, entre ellos el patrón DAO (Data Access Objects) [13].

Clase Vehiculo en Php

Figura 15. Vista del código clase Vehiculo.php

```php
<?php
include_once("ConexionBD.php");
class Vehiculo{
    private $placa;
    private $marca;
    private $modelo;
    private $con;
function Vehiculo($pl, $marc, $model){
    $this->placa=$pl;
    $this->marca=$marc;
    $this->modelo=$model;
    $this->con = new ConexionBD;
}
function consultar(){
if($this->con->conectar()==true){
$resp = $this->con->consultar("select * from vehiculos order by placa");
return $resp;    }
}
function consultarMarca($marca){
if($this->con->conectar()==true){
$resp = $this->con->consultar("select * from vehiculos where marca='$marca'" );
return $resp;
        }
}
function getConexion(){...}
}
?>
```

Fuente: elaboración propia.

Clase Vehiculo Java

Figura 16. Vista del código clase Vehiculo.java

```java
/*...*/
package Logica;
import java.sql.ResultSet;
public class Vehiculo {
   private String placa, marca;
   private int id,modelo;
   private ResultSet registros;
   private ConexionBD con ;
public Vehiculo(int id,String placa, String marca, int modelo) {
    this.id=id;
    this.placa = placa;
    this.marca = marca;
    this.modelo = modelo;
    }
public ResultSet consultaTodos() {
 if (this.con.conectar()!=null){
   this.registros = con.consultar("select * from vehiculos order by placa");
   return this.registros;  }
 else
   return null;
}
public ResultSet consultarMarca(String marca) {
if (this.con.conectar()!=null){
this.registros = con.consultar("select * from vehiculos where marca='"+marca+"'");
 return this.registros;  }
return null;
}
    public ConexionBD getConexion() {...}
```

Fuente: elaboración propia.

6.2. Capa de persistencia

En esta capa se ha definido la clase ConexiónBD que conecta con MySQL y básicamente contiene los métodos conectar() para establecer la conexión con la base de datos y consultar() para ejecutar la sentencia MySQL que genera el conjunto de registros requeridos. En las figuras 17 y 18 se presenta el código propuesto de esta clase en Php y Java.

Clase Conexión BD en Php

Figura 17. Vista código clase ConexionBD.php

```php
<?php
class ConexionBD{
private var $con;
private var $servidor;
private var $usuario;
private var $clave;
private var $bd;
function ConexionBD(){
   $this->con=null;
   $this->servidor = "localhost";
   $this->usuario = "root";
   $this->clave = "123";
   $this->bd = "empresa";
 }
function conectar() {
  if(!($this->con = mysql_connect($this->servidor, $this->usuario, $this->clave))){
     echo"Error al conectar MySQL";
     exit();   }
  if (!@mysql_select_db($this->bd, $this->con)) {
     echo "Error al conectar la base de datos";
     exit();   }
   return true;
 }
function consultar($query){
   $resp = @mysql_query($query);
         if (!$resp)
           return false;
         else
           return $resp;
}
function cerrarConexion(){...}
```

Fuente: elaboración propia.

Clase Conexión en Java

Figura 18. Vista del código clase ConexionBD.java

```
/*...*/
package Logica;
import java.sql.*;
public class ConexionBD {
    private Statement statement;
    private Connection connection;
    private String driver;
    private String bd;
    private String usuario;
    private String clave;
public ConexionBD(){
    this.connection = null;
    this.statement = null;
    this.driver = "com.mysql.jdbc.Driver";
    this.bd = "jdbc:mysql://localhost:3306/empresa";
    this.usuario ="root";
    this.clave ="123";
}
public Connection conectar() {
try {
    Class.forName(this.driver);
    this.connection = DriverManager.getConnection(this.bd, this.usuario, this.clave);
    return connection;
} catch (SQLException e) {
    System.out.println("Error en conexión:" + e.toString());
}catch (ClassNotFoundException e) {
    System.out.println("Error en conexión:" + e.toString());}
return null;
}
public ResultSet consultar(String sentencia){
    ResultSet resultado=null;
    try {
       this.statement = this.connection.createStatement();
       resultado=statement.executeQuery(sentencia);
    } catch (SQLException e){
        System.out.println("Error " + e.toString());    }
    return resultado;
}
public void cerrarconexion(){
    try { this.connection.close();
    } catch (SQLException e){
        System.out.println("Error " + e.toString());   }
}
}
```

Fuente: elaboración propia.

7. Resultado de la ejecución

Para ejecutar este tipo de aplicaciones es necesario aclarar que se debe tener un servidor de aplicaciones dado que se están ejecutando aplicaciones web cliente-servidor; una propuesta es usar *xampp*, ya que provee el API de Php, el servidor Apache y el manejador de base de datos MySQL; además es fácil de activar desde Netbeans para la implementación.

Finalmente, al invocar el archivo *index* de cualquier lenguaje, y agregar un criterio de búsqueda, para el ejemplo la marca Chevrolet, el resultado será el que se presenta en la figura 19.

Figura 19. Vista del resultado de la ejecución consulta con AJAX

Fuente: elaboración propia.

8. Conclusión

Para desarrollar aplicaciones web que le brinden al usuario una buena funcionalidad, es necesario utilizar las tecnologías que se encuentran a nuestro alcance, sin dejar de lado los estándares establecidos para su diseño e implementación.

AJAX es solo una de las múltiples formas de proporcionar esta característica. Lo importante es comenzar y una de las maneras de hacer nuevas aplicaciones es reutilizando el código que ya está disponible.

9. Referencias

[1] J. J. Garrett, *Ajax: A New Approach to Web Applications*. Disponible en: www.adaptivepath.com/ideas/ajax-new-approach-web-applications

[2] Consorcio W3C, *HTML y CSS*. Disponible en: www.w3.org/standards/webdesign/htmlcss

[3] P. Le Hégaret, L. Wood y J. Robie, ¿Qué es el Modelo de Objetos del Documento? Disponible en: www.w3.org/2005/03/DOM3Core-es/introduccion.html

[4] Consorcio W3C, *XSL Transformations (XSLT) Version 1.0*. Disponible en: www.w3.org/TR/xslt.

[5] Consorcio W3C, *XMLHttpRequest Level 2*. Disponible en: www.w3.org/TR/XMLHttpRequest

[6] Mozilla Developer Network, *About JavaScript*. Disponible en: https://developer.mozilla.org/en-US/docs/JavaScript/About_JavaScript.

[7] J. Eguiluz Pérez, *Introducción a AJAX*. Disponible en: http://librosdeingenieriagratis.com/introduccion-a-ajax-por-javier-eguiluz-perez/

[8] W3schools, *The XMLHttpRequest Object*. Disponible en: www.w3schools.com/xml/xml_http.asp

[9] Consorcio W3C. l. cit. Disponible en: www.w3.org/TR/XMLHttpRequest/#handler-xhr-onreadystatechange

[10] Consorcio W3C, ibíd. Disponible en: www.w3.org/TR/XMLHttpRequest/#the-setrequestheader-method

[11] W3schools, *AJAX - The onreadystatechange Event*. Disponible en: www.w3schools.com/ajax/ajax_xmlhttprequest_onreadystatechange.asp

[12] W3schools, *AJAX - Server Response*. Disponible en: www.w3schools.com/ajax/ajax_xmlhttprequest_response.asp

[13] Sun MicrosystemsCore J2EE Patterns, Data Access Object. Disponible en: www.oracle.com/technetwork/java/dataaccessobject-138824.html

9.1. Referencias adicionales

w3schools.com, *AJAX Introduction*. Disponible en:
www.w3schools.com/php/php_ajax_intro.asp

XMLHttpRequest Microsoft, *XMLHttpRequest Object*. Disponible en: http://msdn.microsoft.com/es-ES/library/ie/ms535874.aspx

Netbeans.org, *Introduction to Ajax for Java Web Applications*. Disponible en: http://netbeans.org/kb/docs/web/ajax-quick-start.html