Cómo mantener el patrón modelo vista controlador en una aplicación orientada a la WEB

Carlos Armando López S.

Recibido el 03 de septiembre de 2009. Aprobado el 23 de octubre de 2009

Resumen
Actualmente, en el desarrollo de software orientado a la Web, es imprescindible la utilización de patrones de diseño que permitan el mejoramiento de la calidad del software otorgando a las aplicaciones características que les permitan ser fácilmente mantenibles, flexibles y evolutivas. Uno de los patrones básicos para el desarrollo de aplicaciones orientadas a la Web es el patrón modelo vista controlador (MVC) con el cual se hace una separación entre la parte gráfica de la aplicación (Formularios) y los procesos de la misma. Este artículo presenta la utilización del patrón Modelo Vista Controlador en el desarrollo de aplicaciones orientadas a la Web, y como éste se debe adaptar al constante cambio de tecnologías. Se hace una mirada a la evolución del desarrollo Web haciendo énfasis en la influencia de las tecnologías como AJAX, FLEX y OpenLaszlo. Finalmente se presenta la posibilidad de la unificación de todas las tecnologías para llegar a la creación de aplicaciones Web que funcionen completamente del lado del cliente, y que todos los procesos complejos actuales de la arquitectura Cliente-Servidor se reduzcan a la simple toma de información que se encuentran en servidores de bases de datos.

Palabras clave:
Patrones de diseño, MVC, aplicaciones Web, cliente-servidor.

Abstract:
Currently, the development of Web-oriented software, is an indispensable the use of design patterns that allow the improvement of the quality of software applications providing features that allow them to be easily maintainable, flexible and evolutionary. One of the basic patterns for the development of Web-oriented applications is the model view controller pattern (MVC) with which it is a separation between the graphical part of the application (forms) and its processes. This paper presents the use of Model View Controller pattern in the development of Web oriented applications, and how it must adapt to the constantly changing technologies. It is a look at the evolution of Web development with emphasis on the influence of technologies like AJAX, Flex and OpenLaszlo. Finally, it presents the possibility of unification of all technologies to reach the creation of Web applications that run entirely on the client side, and that all current complex processes client-server architecture reduced to simply making information servers are in databases.

Keywords
Design patterns, MVC, Web applications, client-server.

I. Introducción.

Desde los orígenes de Internet, la evolución de las comunicaciones ha sido vertiginoso, se ha pasado de necesitar un simple intercambio de datos a necesitar aplicaciones que realicen procesos sofisticados y de forma distribuida.

Estos grandes cambios, como es de esperar, han abierto un nuevo nicho donde los desarrolladores de software pueden aprovechar las posibilidades de distribución, escalabilidad y flexibilidad que las tecnologías de Internet presentan. Es por ello que cuando un desarrollador Web diseña una aplicación, no lo debe hacer como lo haría en el diseño de aplicaciones standalone (aplicaciones que funcionan en computadores sin necesidad de estar conectadas a una red) para aprovechar un sin número de tecnologías que puede aplicar y que, en la mayoría de los casos, debe combinar para lograr un resultado eficiente y elegante.

Otro elemento importante que los desarrolladores Web deben mantener y que genera dificultades, es la evolución de las aplicaciones, ya que Internet es un "mar de bits", que tiene cambios constantes y las aplicaciones (los "seres") que lo habitan deben adaptarse constantemente. Desde el cambio de moda hasta la llegada de una nueva tecnología afectan el futuro de las aplicaciones Web, por lo que es necesario que el diseño de las mismas tenga la flexibilidad para "mantenerse a flote".

De lo anterior, surge la gran necesidad del uso de patrones, que al igual que en las aplicaciones standalone, buscan dar solución a los problemas que puedan surgir en el desarrollo. En el caso de las aplicaciones Web, surgen problemas similares a los que pueden tener las aplicaciones standalone, pero adicionalmente se deben mitigar, al menos, los inconvenientes del cambio constante: el uso de diversos lenguajes, la adaptación de paradigmas, el uso de frameworks, entre otros.

II. ¿Qué es un patrón?

Bushmann y Meunier (1996) señalan que un patrón, en el ámbito del desarrollo de software, puede definirse como una solución o grupo de soluciones que ya han sido probadas antes para problemas recurrentes en ciertos contextos.

En otras palabras, un patrón es la forma o método con el que se han podido solucionar problemas en el desarrollo de aplicaciones, aprovechando la experiencia de las corporaciones y en general, de los desarrolladores.

En el desarrollo Web, surge el inconveniente adicional del "constante cambio", por lo que los patrones deben ser aplicados teniéndolo en cuenta, siendo el objetivo final el de mantener un lenguaje común en una comunidad de desarrolladores para comunicar, de forma clara y concisa, las experiencias de quienes han tenido buenos resultados con un método específico.

Entre los muchos patrones existentes hay uno que debe ser aplicado siempre, que permite separar las partes de una aplicación en capas y que en el desarrollo de aplicaciones Web es de gran importancia, se trata del patrón Modelo Vista Controlador.

III. El Patrón Modelo Vista Controlador (MVC)

Este patrón soluciona los inconvenientes debidos al uso de interfaces gráficas en aplicaciones, pero adicionalmente permite definir claramente las capas que se aplican en la arquitectura Cliente-Servidor, muy común en Internet. El patrón Modelo Vista Controlador, separa la parte gráfica de una aplicación, de los procesos lógicos y de los datos de la misma. Fue desarrollado a finales de los años 70 por Trygve M. H. Reenskaug, quien trabajaba en el proyecto SmallTalk de Xerox.

El patrón MVC está compuesto por los siguientes elementos:

La vista: Es la Interfaz gráfica, es con lo que el usuario de la aplicación interactua y es donde se reciben las "ordenes" y se presentan los resultados de los procesos al usuario.

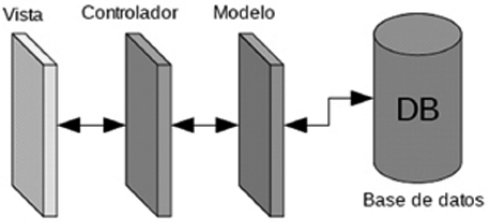

Figura 1. *Esquema de capas del patrón modelo vista controlador. Fuente: El autor*

El controlador: Es el modificador del modelo, es decir, es el que modifica los valores de las variables, objetos, datos en general, lo hace de acuerdo a lo solicitado por el usuario a través de la interfaz gráfica (vista).

El modelo: Es la representación específica de la información con la que se está operando en el instante de la ejecución de la aplicación, representa las variables, objetos, datos en general que se están modificando de acuerdo a lo que el usuario solicite.

Cuando la aplicación requiere del almacenamiento de datos, regularmente se utiliza una base de datos. La información almacenada en esa base de datos depende de los procesos desarrollados en el modelo. Cuando no existe base de datos se utilizan archivos para el almacenamiento de la información, dependiendo de la implementación, dichos archivos pueden ser incluidos dentro del modelo.

IV. Flujo de procesos más común del patrón modelo vista controlador

[1]Aunque se pueden encontrar diferentes implementaciones de **MVC**, el flujo de procesos que sigue generalmente es el siguiente:

1. El usuario interactúa, de alguna forma, con la interfaz de usuario (por ejemplo: el usuario pulsa un botón, enlace, etc.)

2. El controlador recibe (por parte de los objetos de la interfaz-vista), la notificación de la acción solicitada por el usuario. El controlador gestiona el evento que llega, frecuentemente a través de un gestor de eventos (handler) o callback[2].

3. El controlador accede al modelo, actualizándolo, posiblemente modificándolo de forma adecuada a la acción solicitada por el usuario (por ejemplo: el controlador actualiza el carro de la compra del usuario). Los controladores complejos están a menudo estructurados usando un patrón de comando que encapsula las acciones y simplifica su extensión.

4. El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se refleja los cambios en el modelo (por ejemplo: produce un listado del contenido del carro de la compra). El modelo no debe tener conocimiento directo sobre la vista. Sin embargo, el patrón de observador puede ser utilizado para proveer cierta indirección entre el modelo y la vista, permitiendo al modelo notificar a los interesados de cualquier cambio. Un objeto vista puede registrarse con el modelo y esperar a los cambios, pero aun así el modelo en sí mismo sigue sin saber nada de la vista. El controlador no pasa objetos de dominio (el modelo) a la vista, aunque puede dar la orden a la vista para que se actualice[3].

5. La interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente.

V. Nuevas tecnologías y su influencia en el MVC

Cuando se desarrolla una aplicación Web, surge la pregunta ¿Qué capa pertenece al cliente y qué capa pertenece al servidor? Esta pregunta en principio, es simple de solucionar, pero se verá que a la luz del "boom" de las nuevas tecnologías la respuesta puede ser un poco más compleja.

Caso tradicional:

Antiguamente, Internet se utilizaba para presentar información que estaba en archivos de texto o en imágenes. Posteriormente surgió la necesidad de no solo presentar información sino de poder interactuar con los usuarios, de allí surgen lo lenguajes como Perl o PHP, los cuales se utilizan del lado del servidor, quedando al lado del cliente los formularios y presentaciones de resultados en lenguaje de etiquetas html.

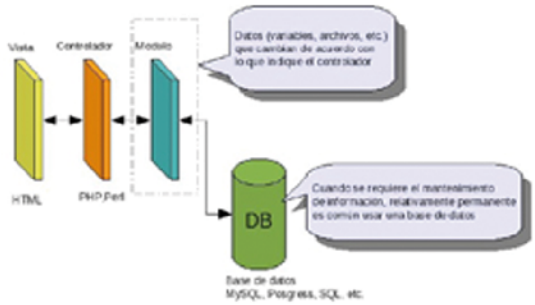

Figura 2. *Lenguajes utilizados en las diversas capas del MVC. Fuente: El autor*

Los lenguajes que más se utilizan en los casos tradicionales del lado del servidor son PHP, ASP y Perl. En años anteriores la programación de los sites en Internet era completamente estructurada, pero con la llegada de Java al ámbito Web y la inclusión del paradigma de orientación a objetos en lenguajes como PHP, cambiaron las costumbres de programación y el patrón MVC debió acomodarse no solo a la arquitectura Cliente-Servidor sino al paradigma de orientación a objetos.

Cabe anotar que, debido a los procesos de transición al paradigma de orientación a objetos en la arquitectura Cliente–Servidor, los desarrolladores del lenguaje PHP (lenguaje más popular en el desarrollo de aplicaciones Web)[4] optaron por la implemen-

[1] Tomado de http://es.wikipedia.org/wiki/Modelo_Vista_Controlador
[2] Callback es una pieza de código de software funcional que puede ser pasada como argumento en un método o función
[3] Nota: En algunas implementaciones la vista no tiene acceso directo al modelo, dejando que el controlador envíe los datos del modelo a la vista.
[4] según el site www.tiboe.com especializado en uso y popularidad de los lenguajes de programación

tación gradual del mismo y hoy PHP es catalogado como un lenguaje multiparadigma.

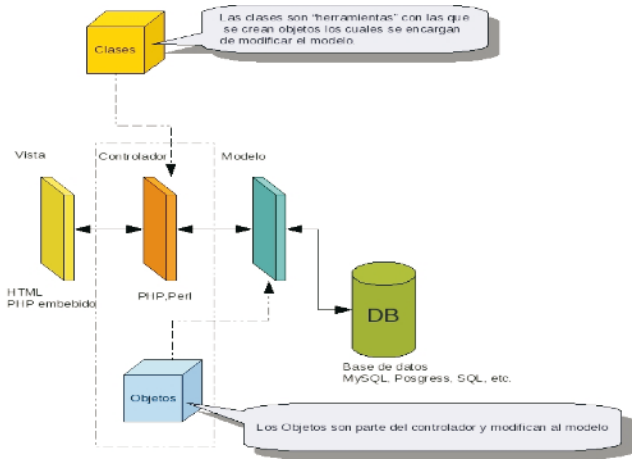

Figura 3. *Patrón MVC y el paradigma de programación a objetos. Fuente: El autor*

Es común encontrar que los desarrolladores Web que trabajan con PHP hagan del controlador un grupo de archivos (o un archivo, cuando la aplicación es pequeña), escritos en forma estructurada en donde se crean objetos a partir de clases que son llamadas mediante el uso de ordenes *require* o *include*. Sin embargo, si se quiere utilizar el paradigma de orientación a objetos de forma más estricta, lo recomendable es hacer clases con métodos estáticos que realicen las operaciones (lo cual es equivalente al uso de la clase Main en Java).

Uno de los "logros" del lenguaje PHP y una de las características que lo han hecho muy popular (otros lenguajes como ASP hicieron lo mismo) ha sido la posibilidad de embeber código PHP dentro de presentaciones HTML. Esto, a pesar de ser de gran utilidad, puede generar problemas al intentar mantener el patrón MVC en el desarrollo de aplicaciones Web, debido a que genera confusión y si no se siguen algunas reglas con estricta disciplina, se puede destruir el patrón llevándose consigo las posibilidades de flexibilidad y escalabilidad que este plantea.

Algunas de las reglas que se recomiendan a la hora de embeber código PHP en HTML son:

1. No haga consultas desde la vista a la base de datos, siempre hágalo desde el controlador.
2. Los datos que tiene que presentar en los formularios en la vista deben ser tomados de arrays o escalares del modelo o del controlador, dependiendo de la implementación.
3. No haga ningún proceso en la vista, ésta solo debe presentar datos si requiere de algún tipo de proceso, este debe ser siempre para presentar resultados en pantalla o para solicitar datos al usuario y nada más.

Antes de la llegada de PHP y los demás lenguajes del lado del servidor, ya había surgido un lenguaje que buscaba dar más dinamismo a la presentación del código HTML, tal es el caso de Javascript, con el cual se pueden generar sistemas de eventos en los elementos HTML y hacer modificaciones en los mismos. Con Javascript surgieron muchos desarrollos que no requerían de procesos en los servidores y que además hacían que las presentaciones HTML fueran más cómodas y elegantes. Así, la presentación de resultados y la creación de formularios, como mínimo utiliza tres lenguajes: HTML, Javascript y PHP.

Caso actual:

Adicionalmente a la implementación del paradigma de programación a objetos, surgen nuevas tecnologías que hacen a las aplicaciones Web más parecidas a las que se utilizan en forma standalone, aprovechando para ello el aumento continuo del ancho de banda de Internet y el poder de procesamiento, que también va en aumento constante en los equipos de computo comunes.

En las aplicaciones Web tradicionales, el usuario viaja de página en página, las cuales pueden contener formularios, gráficos, texto, objetos embebidos, entre otros. Navegación que hace a través de un browser o navegador. Cada vez que el usuario cambia de página, la ventana de navegación aparecerá en blanco mientras que logra ubicarse en la nueva página, proceso que es conocido como "refresco del navegador". Este proceso hace que las aplicaciones Web tengan un comportamiento diferente a lo que ocurre con las aplicaciones standalone. Presentándose de esta forma la necesidad de crear aplicaciones con comportamiento similar a las aplicaciones que están instaladas dentro de los equipos clientes, evitando confusión en los usuarios y presentando nuevas posibilidades de presentación.

Para cubrir esta necesidad surge AJAX[5], un grupo de tecnologías que buscan que las aplicaciones Web tengan un comportamiento similar a las standalone. AJAX es la implementación de un objeto especial del lenguaje Javascript conocido como el xmlhttprequest.

La idea es simple, mediante el uso del objeto xmlhttprequest es posible enviar los datos de un formulario al servidor sin tener que cambiar de página, evitando así el refresco.

Ver figura 4 en la siguiente página

[5] Acrónimo de *Asynchronous JavaScript And XML* (JavaScript asíncrono y XML)

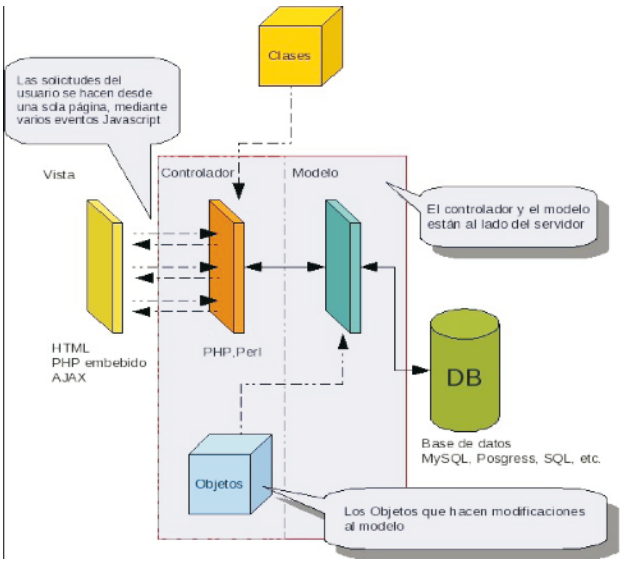

Figura 4. *Patrón MVC, el paradigma de programación a objetos y el uso de AJAX en la Vista. Fuente: El autor*

Además de servir para enviar y recibir datos del servidor, AJAX también sirve para aumentar la calidad visual de la vista, utilizando para ello librerías adicionales de fabricantes expertos en el lenguaje Javascript. Dentro de las librerías más populares se encuentran:

- Dojo
- script.aculo.us
- Rico
- Prototype

Debido a esta variedad, surgen inconvenientes, ya que todas estas librerías son desarrolladas por diferentes fabricantes y cada una de ellas tiene especializaciones, así que por lo general el desarrollador Web debe escoger la o las librerías que va a utilizar y debe, adicionalmente, aprender sus respectivas APIs.

Para solucionar los inconvenientes surgidos de la combinación de estas tecnologías, se han creado frameworks en PHP y Java (entre otros lenguajes), que permiten mantener el patrón MVC de forma fácil unificando el lenguaje de programación. Algunos de los frameworks PHP más populares son:

- Symfony
- Zend Framework
- Seagull
- Prado

Al usar estos framework se producen dos complicaciones:

1.Es necesario comprender el API del Framework que se va utilizar en el desarrollo, siendo en algunos casos difíciles de aprender.

2.Existe dependencia de estas herramientas, lo cual puede ser contraproducente en proyectos de gran envergadura que vayan a ser utilizadas a largo plazo, debido a que en el caso de los Frameworks de software libre, no hay ninguna garantía de su desarrollo y compatibilidad continuos, y en el caso de los Frameworks de pago o propietarios, pueden aumentar el costo de desarrollo a través del tiempo.

Sin embargo, el uso de los frameworks de PHP, así como de otros lenguajes del lado del servidor, incluyen compatibilidad y herramientas para el uso de AJAX (Por lo general contienen las librerías AJAX) y así se tendría que utilizar solo un lenguaje y en el peor de los casos dos con el uso de HTML haciendo más fácil la creación de las aplicaciones. Cabe anotar que la mayoría de estos frameworks están diseñados para soportar correctamente el patrón MVC.

¿Y el futuro?

Existen otras tecnologías que han sido alternas a AJAX y los frameworks de los lenguajes del lado del servidor, los entornos y lenguajes basados en el motor Flash para animaciones.

Antiguamente, la tecnología Flash, desarrollada por Macromedia (hoy parte de Adobe), era utilizada para la creación de animaciones que servían básicamente como decoración o publicidad. Pero poco a poco, ésta tecnología fue convirtiéndose en una poderosa herramienta útil en la creación de pequeños video juegos y otros elementos interactivos, y así fue como surgió el lenguaje ActionScript.

ActionScript es parte de lo que hoy se conoce con el nombre de Flex, un framework para la creación de aplicaciones muy elegantes y visualmente notables del lado del cliente mayormente conocidas como RIA (Rich Internet Appliction).

Las aplicaciones RIA son compiladas para que funcionen bajo el motor de Flash[6] (usándolo como si se tratara de una máquina virtual), y su extensión es swf, tal como ocurre con cualquier animación.

Flex ha demostrado que se puede aprovechar el poder de los equipos de cómputo del lado del cliente realizando todo tipo de procesos que normalmente en la arquitectura Cliente -Servidor serían desarrolladas en el servidor. Un ejemplo del poder de Flex está muy bien representado en el reproductor de video de youtube (www.youtube.com). Al igual que Flex, existe otro framework muy divulgado últimamente, el cual es conocido

[6] Flash es una aplicación en forma de estudio de animación que trabaja sobre "Fotogramas" destinado a la producción y entrega de contenido interactivo

como OpenLaszlo[7]. OpenLaszlo presenta similares características a las de Flex, sin embargo, en su última versión presenta la posibilidad de no sólo usar el motor Flash (para generar compilaciones swf), sino que además el desarrollador puede elegir si desea que su aplicación se presente como Dhtml (HTML dinámico). OpenLaszlo es software libre y aún no cuenta con un buen entorno de desarrollo, sin embargo, su lenguaje LZX es muy sencillo y fácil de aprender.

VI. Recomendaciones para el uso de aplicaciones RIA con el patrón MVC:

Las recomendaciones para el uso de aplicaciones RIA con el patrón MVC son las siguientes:

1. Recuerde que una aplicación RIA que no se conecta a una base de datos, puede considerarse como la Vista del patrón, y está del lado del cliente.
2. Si la aplicación RIA realiza conexiones a bases de datos, es posible que sea conveniente hacer la aplicación para que funcione del lado del cliente sin intermediarios en el servidor.
3. Si la aplicación es software libre, se sugiere el uso de OpenLaszo y que la aplicación sea presentada en Dhtml.

Mientras que los desarrolladores se adaptan a las nuevas tecnologías, las aplicaciones RIA se utilizan para la presentación de formularios, animaciones y demás, en otras palabras, hoy día una aplicación RIA es la vista de una aplicación Web. (Ver figura 5.)

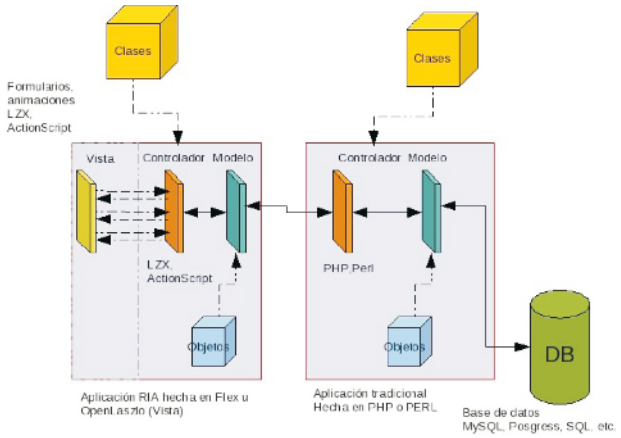

Figura 5. *Patrón MVC, el paradigma de programación a objetos y el uso de FLEX y OpenLaszlo. Fuente: El autor*

[7] OpenLaszlo es una plataforma código abierto para el desarrollo y distribución de RIA Rich Internet Applications (Aplicaciones Ricas de Internet). Ha sido publicado bajo la licencia Common Public License, certificada por la Open Source Initiative.

Pero es muy probable que en el futuro del desarrollo de las aplicaciones orientadas a la Web, se unifiquen y utilicen este tipo de tecnologías facilitando el trabajo de los desarrolladores y haciendo que la arquitectura Cliente-Servidor presente una aplicación completa del lado del cliente, y los datos sean almacenados del lado del servidor, aumentando la eficiencia y la velocidad de respuesta de las mismas. (Ver figura 6.)

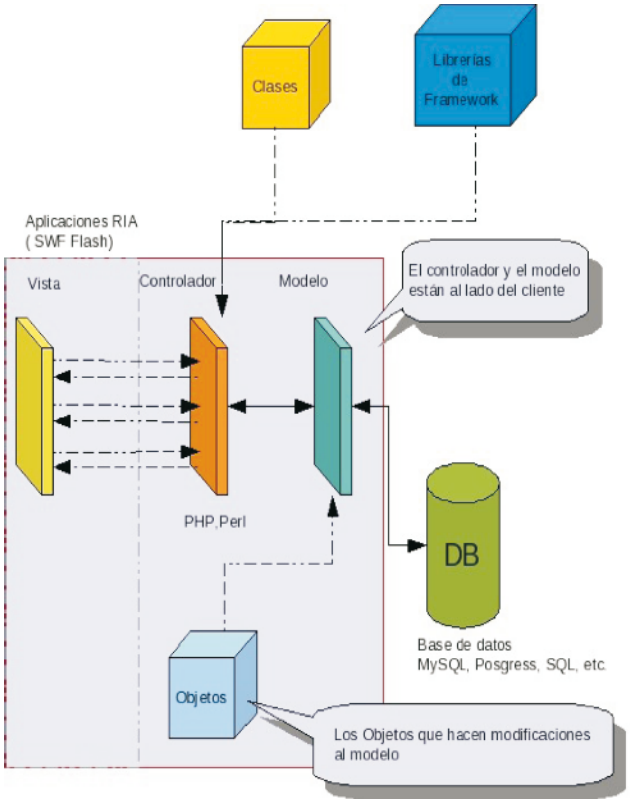

Figura 6. *Patrón MVC, el paradigma de programación a objetos y el uso de FLEX y OpenLaszlo desde el cliente. Fuente: El autor*

VII. Conclusiones

· El patrón modelo vista controlador (MVC) ha facilitado la creación de aplicaciones Web escalables y flexibles desde los orígenes de las mismas, sin embargo, la adaptación de este patrón depende de las tecnologías utilizadas.

· El patrón MVC se adapta muy fácilmente a la arquitectura Cliente-Servidor, sin importar la tecnología que se use en el desarrollo de las aplicaciones. Sin embargo, es necesario mantener una disciplina estricta que impida la propagación de incoherencias que destruyan el patrón y no se logren los cometidos del mismo.

· Muy probablemente, debido a los incrementos del ancho de banda y de capacidades de procesamiento, las aplicaciones RIA serán el futuro de las

aplicaciones Web y todo regresará a la "normalidad", los navegadores nos permitirán navegar por la red y permitirán la descarga y ejecución de aplicaciones del lado del cliente y los datos se mantendrán en la red de redes.

VIII. Referencias

[1] Buschmann, F. & Meunier (2001).*Pattern Oriented Software Architecture, Volume 1: A System of Patterns*. John Wiley and Sons Ltd. U.S.A.

[2] Gamma E., Helm R., Johnson R. & Vlissides J. (2006). *Elements of Reusable Object-Oriented Software*. Addison-Wesley. U.S.A

[3] Reenskaug M. MVC XEROX PARC 1978-79. (2009) extraído el 10 de enero de 2009 de la página oficial de Trygve M. H. Reenskaug: http://heim.ifi.uio.no/~trygver/themes/mvc/mvc-index.html

Carlos Armando López S. *Ingeniero civil de la Universidad De La Salle, 1998. Tecnólogo en Informática de la Corporación Universitaria Minuto de Dios (UNIMINUTO), 2006. Estudiante de Maestría en Ingeniería de Sistemas y computación de la Universidad de Los Andes; Fundador de la comunidad de software libre Arca-csl de UNIMINUTO; docente del programa de Tecnología en Informática de UNIMINUTO.*
clopez@uniminuto.edu