Saber, Universidad de Oriente, Venezuela.Vol. 26 Nº 2: 168-173. (2014)
ISSN: 2343-6468 Digital / ISSN: 1315-0162 Impreso/ Depósito Legal pp 198702SU187

PROPUESTA DE MODELO EN CINCO CAPAS PARA APLICACIONES WEB

PROPOSAL OF A FIVE LAYERS MODEL FOR WEB APPLICATIONS

Loly Valentina Gómez Fermín [1], Tomás Rafael Moreno Poggio[2]

Universidad de Oriente, Núcleo de Nueva Esparta, [1]Escuela de Hotelería y Turismo, Programa de Licenciatura en Informática, [2]Servicios de Computación Académica, Guatamare, Isla de Margarita, Venezuela
E-mail: loly.gomez@ne.udo.edu.ve / tomas.moreno@ne.udo.edu.ve

RESUMEN

La posibilidad de crear programas multiplataforma es un objetivo para muchos programadores; sin embargo, es complejo de lograr. En la actualidad, las aplicaciones web se han presentado como el camino por excelencia para alcanzar este anhelado fin, pero al estar el proyecto atado a una plataforma de software o hardware éste, irremediablemente, está sujeto a la obsolescencia. Esto no es algo exclusivo de la aplicación cliente, sino que también es aplicable del lado del servidor, por lo que al desarrollar software, aun sin desearlo, existen ataduras a un manejador de bases de datos y a otras aplicaciones. Por esto, el realizar sistemas que sean capaces de operar con múltiples bases de datos y no dependientes de una plataforma específica se convierte en un trabajo titánico para un equipo de desarrollo. En este sentido, la presente investigación propone un modelo de trabajo que permita realizar aplicaciones web capaces de operar en una estructura de capas, donde cada capa definida pueda ser sustituida sin necesidad de reescribir las demás, y permitiendo así real abstracción a la aplicación sobre cualquier plataforma de software o hardware, tanto del lado del cliente, como del servidor. Para este fin la investigación se apoyó en Sommerville (2005), quien plantea una clasificación detallada de los modelos de desarrollo de software, según su organización y su descomposición modular, lo que sirvió de base para el desarrollo de la propuesta. Esta investigación es de carácter documental, puesto que se basó en la recopilación de material bibliográfico referente a arquitecturas de software existentes.

Palabras clave: Arquitectura de software, diseño en capas, multiplataforma.

ABSTRACT

The ability to create cross-platform programs is a goal for many software developers; however, it is complex to achieve. At present, web applications have been introduced as the chosen way to achieve this desired goal, but being the project tied to a software or hardware platform it is, inevitably, subject to obsolescence. This is not unique to the client application, but also applies to the server side, so that when developing software, even unwillingly, there are strings attached to a database and other applications. Because of this, to develop systems capable of operating with multiple databases and not dependent on a specific platform becomes a Herculean task for a development team. Therefore, this research proposes a working model which enables web applications capable of operating in a layered structure, where each defined layer can be replaced without the need to rewrite the other, and allowing real abstraction for application on any platform software or hardware, both on the client and server sides. For this purpose, the research is supported by Sommerville (2005), who presents a detailed classification of models of software development, according to its organization and modular decomposition, which layed the basis for the development of this proposal. This research is documentary in nature, since it is based on the collection of bibliographic material relating to existing software architectures.

Key words: Software architecture, design layers, multiplatform.

INTRODUCCIÓN

Las Aplicaciones Web se han convertido en una de las implementaciones de software más común en la actualidad, esto debido principalmente a su abstracción sobre el software y el hardware en el que se ejecuta, es decir, su habilidad de ser utilizada sobre un navegador web lo cual le permite no depender de un sistema operativo específico, no estar atado a un navegador determinado y hasta el abstraerse de la arquitectura de hardware en la que es usada. Sin embargo, el crear Aplicaciones Web altamente flexibles y escalables requiere de un arduo trabajo que en muchos casos compromete la mantenibilidad del software o en otros obliga a comprometer la flexibilidad del mismo.

La mayoría de las arquitecturas actuales persiguen la independencia de ciertos componentes en el diseño de software, sin embargo, el irrespeto de los estándares por parte de algunos fabricantes[1] de software, dificultan en gran medida el desarrollo y "obligan" a que el producto desarrollado basado en sus herramientas esté inexorablemente ligado a su producto.

[1] Como es el caso de los Sistemas Manejadores de Bases de Datos Relacionales (SMBDR), que pese a tener un estándar vigente, como el SQL en su séptima revisión, muchos de ellos difieren en operaciones básicas como la creación de tablas o el largo en los nombres de los atributos de una tabla, lo cual se puede considerar un incumplimiento del estándar, acción que dificulta enormemente la portabilidad de una base de datos entre SMBDR. Para comprobar estas aseveraciones se puede referir a la documentación sobre definición de tablas usando Oracle como SMBDR, y compararlo con su equivalente en otros SMBDR como Firebird o SQL Server en cualquiera de sus versiones. En tal caso, esta práctica no puede llamarse un estándar corporativo, porque no es algo que afecte solo al fabricante, sino también a la

Recibido: octubre 2013. Aprobado: febrero 2014.
Versión final: mayo 2014.

comunidad de desarrollo, por lo que al existir más de un estándar, aplicable a una misma situación, el concepto mismo de estándar se ve comprometido.

En tal sentido, el objetivo de esta investigación fue proponer un modelo para aplicaciones Web que brinde flexibilidad en el desarrollo, alta escalabilidad sin comprometer el mantenimiento del código.

MATERIALES Y MÉTODOS

La investigación fue de tipo documental, debido a las revisiones criticas del estado del conocimiento: integración, organización y evaluación de la información teórica y empírica existente sobre el problema, focalizado ya sea en el progreso de la investigación actual y posibles vías para su solución, en el análisis de la consistencia interna y externas de las teorías y conceptualizaciones para señalar sus fallas o demostrar la superioridad de unas sobre otras, o en ambos aspectos. De acuerdo con Arias (2006, p. 27). La investigación documental es un proceso basado en la búsqueda, recuperación, análisis, crítica e interpretación de datos secundarios, es decir, los obtenidos y registrados por otros investigadores en fuentes documentales: impresas, audiovisuales o electrónicas. Como en toda investigación, el propósito de este diseño es el aporte de nuevos conocimientos.

Para la investigación se realizó una revisión de las arquitecturas más utilizadas en el desarrollo de software con base en lo expuesto por Sommerville (2005), posteriormente se indagó en las debilidades que presentaban las mismas para implementaciones que operen con distintas bases de datos en entornos de Aplicaciones Web

RESULTADOS Y DISCUSIÓN

Todo desarrollo de software debe adoptar un modelo para el desarrollo de su proyecto, para Sommerville (2005), estos modelos pueden ser clasificados de dos formas: según su organización y según su descomposición modular. Según la organización, un software puede responder a tres modelos específicos, de Repositorio, Cliente/Servidor o por Capas, cada modelo brinda ciertas ventajas a la hora de diseñar aplicaciones web, pero también presentan limitaciones, como se muestra en la Tabla 1.

Tabla 1. Cuadro comparativo de los modelos de desarrollo de software según su organización.

Modelo	Ventajas	Limitaciones
De Repositorio	Es eficiente compartiendo datos entre sus subsistemas (1)	Es difícil la integración de nuevos subsistemas (1)
Cliente/Servidor	Permite usar mayor poder de cómputo. Brinda mayor oportunidad de personalizar el software (2)	Puede resultar costosa su implementación (2)
De Capas	El desarrollo se puede llevar a cabo en varios niveles y, en caso que sobrevenga algún cambio, sólo se ataca al nivel requerido (1)	Dependiendo del diseño pueden llegar a tener una alta complejidad.

(1) Sommerville 2005, (2) Kendall y Kendall 2005

La flexibilidad del Modelo por Capas, brinda una ventaja considerable a la hora de lograr la escalabilidad de un producto de software, por tal motivo se seleccionó este modelo como base del desarrollo de la propuesta, sin embargo, esto solo define la organización de los componentes funcionales, mas no define el cómo están estructuradas cada una de las piezas que conforma dichos componentes, es decir, su descomposición modular.

Para Sommerville (2005) la descomposición modular se define como "un nivel estructural adicional donde los subsistemas son descompuestos en módulos" Existen dos estilos presentados por este autor, la Descomposición Modular Orientada a Objetos y la Descomposición Modular Orientada a Flujo de Funciones, siendo la Descomposición Modular Orientada a Objetos la seleccionada para aplicar en la propuesta, dada su escalabilidad y potencia a la hora de la implementación, sin embargo existen otros autores que presentan la necesidad de una Orientación a Procesos (Berrocal *et al*. s.f.), la cual se adaptaría mejor a realidades empresariales, por ello se propone emplear la orientación a objetos, pero con una visión de organización orientada a procesos, a fin de poder lograr objetos de negocio funcionales y si se

quiere independientes entre sí.

Para lograr esa independencia entre los objetos de negocio se requiere un sistema de comunicación que garantice el funcionamiento del objeto, pero sin generar un fuerte acoplamiento de los mismos, en este punto es cuando se consideran los principios de Arquitectura Orientada a Servicios, o SOA, por sus siglas en inglés, la cual brinda los medios para lograr esa comunicación entre los diversos elementos del sistema, sin la necesidad de conocer la organización, funcionamiento o morfología del elemento destino. El objetivo detrás de esta compleja estructura es lograr una abstracción real sobre la base de datos, mediante el uso de un middleware de dos capas, basado en la transferencia de información en formato de texto plano usando HTTP y HTTPS.

En cuanto a aspectos de calidad del software como lo son la portabilidad y mantenibilidad, esta propuesta busca facilitar las tareas de mantenimiento del software al minimizar la dependencia de los elementos que componen el proyecto de software, no obstante, aunque la portabilidad no se ve sacrificada, la complejidad de un proyecto desarrollado bajo la visión de esta propuesta seria considerable, lo que podría dificultar un poco su movilidad si no se organiza de manera adecuada, o se desconoce su estructura.

En la Figura 1 se muestra un esquema del modelo propuesto, detallando la función principal de cada capa.

Capa 0 Base de Datos

Representa a la base de datos en sí misma, asociada a su Sistema Manejador de Base de Datos, interactúa con la Capa 1 a través de una conexión nativa de base de datos.

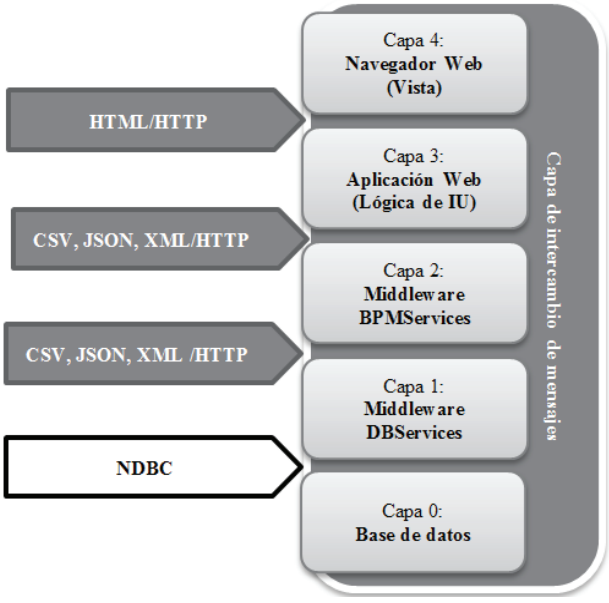

Figura 1. Modelo de Arquitectura de Cinco Capas para Aplicaciones Web.

Dependiendo de él o los manejadores que se seleccionen para una implantación se deberá modificar la Capa 1, dado que la lógica será la asociada al manejador mas no a la lógica del negocio de la aplicación.

Capa 1 Lógica del Manejo de los Datos

Es la capa responsable de conectarse directamente a la Base de Datos (Capa Inferior) usando una conexión nativa a base de datos y recibir peticiones de acceso a la información de la BD (Capa Superior), para regresarla en un formato de codificación de texto y de presentación neutro como XML, JSON, CSV según requerimientos y configuración. El transporte usado de Capa 1 a Capa 2 es HTTP, la codificación es en texto, y el formato es alguno de los antes indicado, esto aplica también para las capas 2 y 3.

La lógica asociada a esta capa es la inherente netamente al manejo de la Base de a Datos, es decir, si se requiere que un sistema opere con base de datos PostgreSql y MySql, en la Capa 1 se definirán los servicios que permitan acceder, eliminar y modificar los datos, para convertirlos en un formato de texto, haciendo uso del respectivo driver de conexión a base de dato para el manejador.

Capa 2 Procesos de Negocios

En esta capa se realiza toda la lógica del negocio en si misma a petición de su capa superior. Al tener la lógica del negocio aislada se puede orientar el desarrollo a los procesos de negocio existentes en cada situación particular, ganando como beneficio el poder desarrollar solo una vez lo inherente a la lógica del negocio, pese a tener uno o más sistemas manejadores de bases de datos para su implantación.

En este nivel es que se observa la ventaja real de este modelo, pues la lógica del negocio puede ser lo más cambiante en un desarrollo de software, sin embargo, bajo este modelo se podría modificar las veces que sea necesario la lógica del negocio, sin tener que afectar a la Capa 1. Logrando con esto una abstracción real sobre los datos y ganando flexibilidad y mantenibilidad.

Capa 3 Control de la Interfaz de Usuario

Esta capa es la responsable de la lógica de la interfaz de usuario, en este nivel se puede estudiar el uso de diversas herramientas RIA como por ejemplo ZK, Vaadin o Google Web Tools Kit, o cualquier otra herramienta que brinde una interfaz de usuario amigable y ágil para el desarrollo y que permita una experiencia de usuario tan rica como la de una aplicación nativa pero en ambiente web. Para esta capa el trasporte utilizado sigue siendo HTTP, la codificación es texto, sin embargo cambia el formato de representación para la comunicación con la Capa 4, que para este caso es HTML y para la comunicación con la Capa 2 continua siento XML, JSON, CSV.

Capa 4 Vista

Es la capa final con la que el usuario interactúa realmente (interfaz de usuario ya materializada en un navegador web). Dependiendo de la herramienta de desarrollo el manejo de esta última capa podría cambiar, lo importante a considerar es que el software a desarrollar debe poderse ejecutar en cualquier navegador, tal vez esto resulte un poco obvio, sin embargo existen desarrolladores web que limitan su aplicación a uno o dos navegadores, esta inadecuada práctica se logra llenando el software de código autogenerado por algunas herramientas de edición, o utilizando características exclusivas de algún Sistema Operativo, esto resulta un sin sentido, pues el desarrollo web persigue el ser multiplataforma, si se está dispuesto a casarse con un sistema operativo, es preferible hacer un desarrollo nativo, pues se economiza tiempo y recursos.

Despliegue de componentes para el modelo de cinco capas

Monolítico

Es la implementación en la que todas las capas del software pueden ejecutarse en un mismo servidor físico o virtual, ideal para aplicaciones de bajo consumo de recursos como se muestra en la Figura 2. Su principal ventaja es la fácil implementación, su desventaja no puede manejar gran cantidad de clientes (Fig. 2).

Escalabilidad Lineal

En la Figura 3 se muestra esta opción de despliegue, cada capa del modelo se aloja en un servidor distinto, lo que permite mayor escalabilidad y robustez en la implementación. La desventaja de esta modalidad de despliegue es el mayor coste en hardware, sin embargo las prestaciones obtenidas justifican la inversión.

Escalabilidad Independiente

Para esta implementación se tiene cada capa alojada en un servidor, pero además se tienen diversas instancias de capa 3, capa 2 y capa 1 trabajando en forma lineal, esto permite repartir mejor la carga de trabajo de las diversas solicitudes de los clientes, la principal ventaja de este despliegue es que puede manejar un mayor número de solicitudes, y es adecuada para sistemas que requieran de alta disponibilidad, pues existen respaldos de cada línea que permitirá mantener el sistema *On Line*, como se muestra en la Figura 4, sin embargo su implementación es más compleja y costosa, tanto en la implantación como en su mantenimiento, lo cual es una desventaja. Otro detalle a considerar es que a pesar de las réplicas de las capas 3 a la 1, en caso de fallar capa 0 el sistema quedaría fuera de línea.

En Clúster

La implementación en clúster es la más compleja, pero quizás la más potente de las opciones de despliegue, es muy similar a la implementación de escalabilidad independiente, solo que la base de datos está implementada en un clúster, lo cual rinda un poco más de resistencia a fallos y permite aumentar el volumen de datos a manejar, como desventaja resalta la complejidad del despliegue y su elevado coste como se detalla en la Figura 5.

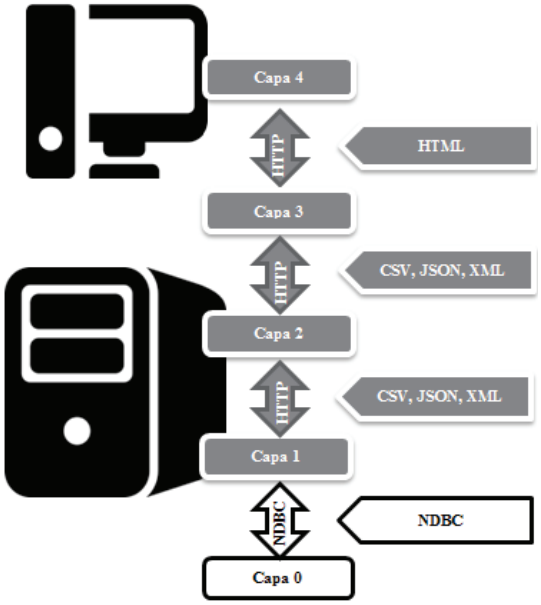

Figura 2. Despliegue Monolítico del modelo de cinco capas.

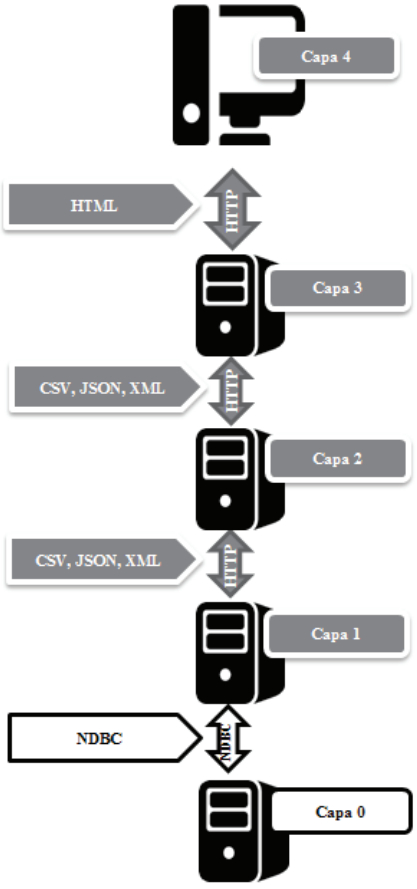

Figura 3. Despliegue de Escalabilidad Lineal del Modelo de cinco capas.

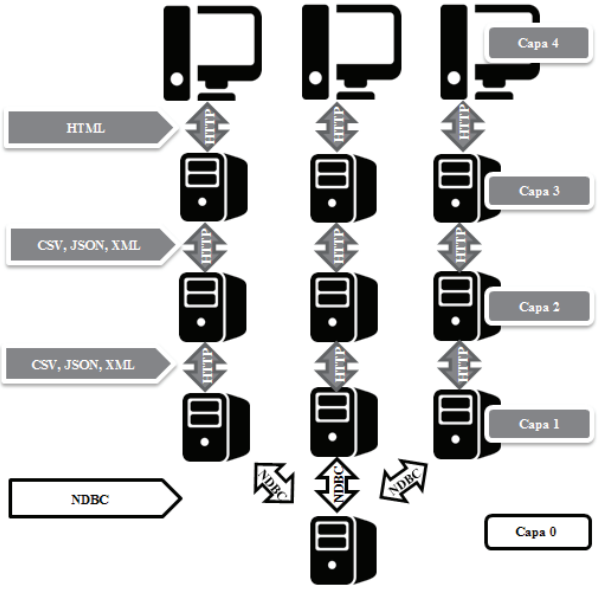

Figura 4. Despliegue de Escalabilidad Independiente.

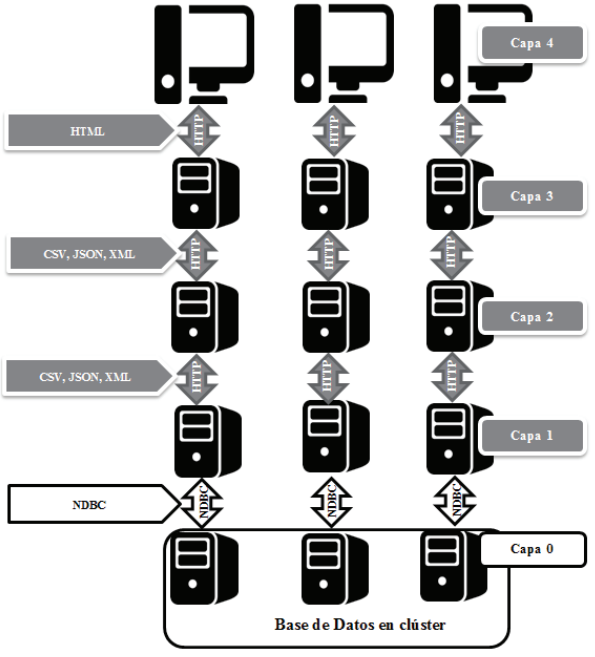

Figura 5 Despliegue en Clúster para modelo de cinco capas.

CONCLUSIONES

El desarrollo Web es quizás una de las áreas con mayor avance en los últimos 10 años, sin embargo muchos desarrolladores con experiencia en aplicaciones nativas o desktop se han alejado del desarrollo Web por lo laborioso o demandante que puede ser su mantenibilidad, sacrificando así la posibilidad de que sus aplicaciones puedan ser multiplataforma.

En tal sentido, la propuesta de esta investigación es una forma de organización que busca lograr un desarrollo Web mantenible, flexible y realmente abstraído de la plataforma, no solo a nivel del cliente, sino también a nivel del servidor donde se implemente la aplicación.

La organización en capas y la orientación a servicios son elementos que brinda al modelo propuesto las características necesarias para lograr la flexibilidad requerida por las aplicaciones Web, y la posibilidad de operar con diversas bases de datos sin tener que invertir muchas horas de desarrollo.

El modelo planteado está basado en la orientación a servicios y la organización por capas, lo cual procura mezclar las ventajas de ambos modelos para aportar flexibilidad al desarrollar.

Se recomienda en una investigación posterior desarrollar un middleware que implemente las capas 1 y 2 de forma genérica a fin de comprobar la efectividad del modelo y medir su rendimiento en condiciones reales de operatividad.

REFERENCIAS BIBLIOGRÁFICAS

Arias F. 2006. El Proyecto de Investigación. Introducción a la Metodología Científica. Episteme, Caracas, Venezuela, pp. 56-63.

Berrocal J, García JM, Murrillo JM. (s.f.). Hacia una gestión del proceso software, dirigida por procesos de negocio. Grupo Alarcos, Univeridad de Castilla La Mancha. Disponible en línea en: http://alarcos.inf-cr.uclm.es/pnis/articulos/pnis-07-Berrocal-GPSDPN.pdf (Acceso 16.08.2013).

Kendall KE, Kendall JE. 2005. Análisis y diseño de sistemas. Pearson, México, México, pp. 917.

Sommerville I. 2005. Ingenieria del Software (Séptima ed.). Pearson Educación S- A., Madrid, España.