Artículo de investigación

MDDCLOUD: Framework MDD para Aplicaciones Web Empresariales según Especificación Requisitos de Software[9]

MDDCLOUD: Framework for Enterprise Web Applications MDD according Specification Software Requirements

MDD CLOUD: Framework for Enterprise Applications Web MDD de acordo Especificação de Requisitos de Software

Fecha de recepción: 21 de marzo de 2016 / Fecha de aceptación: 18 de abril de 2016

Escrito por: Sergio Andrés Ñustes[10]
Jorge Luis Hurtado[11]
Yois Pascuas Rengifo[12]

Resumen

Este artículo tiene como objetivo describir el desarrollo del MDDCloud, propuesta de framework MDD en la nube para el desarrollo de aplicaciones web empresariales PHP con arquitectura modelo vista controlador (MVC). Metodología: abarcó las fases de comunicación, planeación y construcción, utilizando la metodología de desarrollo ágil *Scrum,* orientada a unir las etapas de comunicación, modelado y construcción, a partir de la trazabilidad en la captura de requisitos, diagramación UML y generación automática de código funcional. Resultados: MDDCloud está compuesto por cinco módulos: un sistema de captura de requerimientos, modelamiento, generación automática de aplicaciones web empresariales, administración de usuarios y proyectos y, una wiki de ayuda para el manejo del sistema. Conclusiones: se comprueba que el desarrollo de aplicaciones web empresariales es posible mediante la automatización de conceptos. Es posible desarrollar implementaciones amigables y en la nube de MDD.

Palabras Clave: Marco de trabajo; Nube; Aplicaciones web empresariales; arquitectura modelo vista controlador; Trazabilidad.

Abstract

This article aims to describe the development of MDDCloud, proposed framework MDD cloud for developing enterprise web applications PHP architecture model view controller (MVC). Methodology: it encompassed the phases of communication, planning and construction, using *Scrum* agile development oriented methodology to join the stages of communication, modeling and construction, from traceability requirements capture, UML diagramming and automatic generation of functional code. Results: MDDCloud consists of five modules: A system requirements capture, modeling, automatic generation of enterprise web applications, user management and projects, and, help wiki management system. Conclusions: checks that the development of enterprise web applications is possible through the automation of concepts. It is possible to develop friendly and MDD cloud implementations.

Keywords: Framework; Cloud; Enterprise web applications; Model view controller architecture; Traceability.

[9] Informe del desarrollo del proyecto de investigación: *MDDCLOUD: Un framework MDD en la nube para la generación de aplicaciones web empresariales a partir de la especificación de requisitos de software.*

[10] Ingeniero de Sistemas, Estudiante de Maestría en ingeniería de sistemas y computación Universidad Nacional Bogotá. sanustes@unal.edu.co

[11] Ingeniero de Sistemas, Estudiante de Maestría en ingeniería de sistemas y computación Universidad Nacional Bogotá. jlhurtadoi@unal.edu.co

[12] Ingeniera de Sistemas, Magíster en Ciencias de la Información y las comunicaciones. Docente Universidad de la Amazonia. Florencia. y.pascuas@udla.edu.co

Resumo

Este artigo visa descrever o desenvolvimento de MDDCloud, propôs quadro MDD em nuvem para desenvolvimento de aplicações web corporativas arquitetura PHP controlador de vista de modelo (MVC). Metodologia: compreendia as fases de planejamento, comunicação e construção, usando a metodologia de desenvolvimento ágil *Scrum* orientada para juntar-se os estágios da comunicação, modelagem e construção, de rastreabilidade requisitos de captura, diagramação UML e geração automática de código funcional. Resultados: MDDCloud consiste em cinco módulos: Uma captura de requisitos do sistema, modelagem, geração automática de aplicativos corporativos web, gerenciamento de usuários e projetos, e, sistema de ajuda de gestão de wiki. Conclusões: verifica que o desenvolvimento de aplicações web da empresa é possível através da automação de conceitos. É possível desenvolver implementações de nuvem MDD e amigável.

Palavras-chave: Quadro; Nuvem; Aplicativos de web corporativos; Arquitetura de controlador de modo de exibição de modelo; Rastreabilidade;

Introducción

La ingeniería de requisitos es un conjunto de tareas que conducen a comprender el impacto del software, qué es lo que el cliente quiere y cómo interactuarán los usuarios finales con el software. Por lo tanto, la ingeniería de requisitos representa una etapa fundamental, de acuerdo con Presman (2005), el diseño y la construcción de un sólido programa de computadora que resuelva el problema incorrecto no satisface las necesidades de nadie. Por lo tanto, es muy importante entender lo que el cliente quiere antes de comenzar a diseñar y construir un sistema basado en computadora.

Actualmente, el desarrollo de aplicaciones a la medida se hace bajo el enfoque de aplicaciones web empresariales, este tipo de soluciones, según Pan (2008), se caracterizan por su: escalabilidad, seguridad, acceso a datos, disponibilidad, integración, diseño arquitectónico e interfaces de usuario web en internet o intranet.

No obstante, la ingeniería de requisitos aún presenta algunos problemas en relación a las plantillas y los modelos usados para la captura y especificación de requisitos, puesto que no se sincronizan de manera automática; además, las herramientas de la Ingeniería de Software Asistida por Computador (CASE, por sus siglas en inglés), que ayudan a cada uno de los procesos de desarrollo software se presentan de manera aislada y no hay una trazabilidad entre las distintas etapas; esto genera más tiempo de desarrollo, complejidad en los sistemas desarrollados y dificultad en la mantenibilidad, como explica Cohn (2013) provoca en algunos casos el fracaso en los proyectos software.

Dada la anterior problemática se planteó la siguiente pregunta de investigación: ¿Cómo desarrollar una framework para la generación de aplicaciones web empresariales a partir de la especificación de requisitos de software? En coherencia con este cuestionamiento, el objetivo de este artículo es presentar el desarrollo MDDCloud, un framework MDD (por sus siglas en inglés: Model Driven Development y en español Desarrollo Dirigido por Modelos) en la nube para la generación de aplicaciones web empresariales a partir de la especificación de requisitos de software.

En consecuencia, como una alternativa de solución al interrogante formualdo, se propone MDD Cloud: un framework (marco de trabajo) de desarrollo de aplicaciones web empresariales PHP con arquitectura modelo vista controlador (MVC), que está orientado a unir las etapas de comunicación, modelado y construcción, a partir de la trazabilidad en la captura de requisitos, diagramación UML y generación automática de código funcional.

MDDCloud es una aplicación en la nube (Cloud computing), pues una problemática actual que tienen las implementaciones MDD es que se presentan en archivos binarios para un sistema

operativo especificado, sesgando su uso a éste, al igual, algunos de éstos necesitan de herramientas extras como (Eclipse 13) y/o requieren un alto conocimiento para su instalación y configuración (Pascuas, Bocanegra, Ortiz y Pérez, 2012). Con la computación en la nube, se solventa este problema pues la aplicación puede ser accedida desde cualquier sistema operativo o dispositivo a través del navegador web.

En este sentido, el desarrollo dirigido por modelos es un enfoque de desarrollo software que permite establecer una relación entre las fases de modelación y construcción, incluso existen implementaciones MDD para la ingeniería de requisitos, es decir la posibilidad de la generación de código desde la fase de comunicación; sin embargo, estas se basan en la diagramación de los procesos de negocios y no desde la especificación de requisitos, objeto de estudio de esta investigación, se reconoce el MDDCloud como una aplicación en la nube (Cloud computing), que puede ser accedida desde cualquier sistema operativo o dispositivo a través del navegador web. Por otra parte, el metamodelo en una aplicación basada en MDD representa un característica crucial debido a que es el encargado de definir el dominio del problema, de especificar los conceptos concernientes del mismo y de especificar la relación entre dichos conceptos.

Para el desarrollo de esta investigación se utilizó la metodología de desarrollo ágil *Scrum*. En su realización se ejecutaron cuatro etapas: a) análisis del sistema: elementos del proceso de elicitación de requisitos, elementos del proceso de análisis de requisitos, generación de aplicación web empresarial, plantillas (para actores del sistema, para requisitos funcionales – especificación de casos de uso, para requisitos no funcionales), diagramas; b) descripción de especificación de caso de uso, c) diagramas de actividades. El presente artículo se estructura en cuatro secciones, la primera orientada a mostrar los fundamentos teóricos de la investigación, la segunda el método propuesto y en las siguientes, los resultados y discusión.

Fundamentos Teóricos

Las aplicaciones empresariales son un tipo de desarrollo software enfocado a soluciones a la medida, o necesidades de cada organización; de acuerdo con Pan (2008), se define como un conjunto de características que permite las siguientes aplicaciones empresariales:

- Acceso a base de datos, normalmente en bases de datos relacionales.
- Transaccionales, propiedades ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad.
- Escalables, deberían poder soportar más carga de trabajo sin necesidad de modificar el software (sólo añadir más máquinas)
- Disponibilidad, idealmente no deben dejar de prestar el servicio -Seguras, no todos los usuarios pueden acceder a la misma funcionalidad.
- Integración, es preciso integrar aplicaciones construidas con distintas tecnologías.
- Tipo de Interfaz, de entorno de ventanas, Web: en internet e intranets Separación clara entre la interfaz gráfica y el modelo

En este sentido, se considera que la ingeniería de software aún carece de métodos para capturar adecuadamente los requerimientos del cliente, y por lo tanto, de la generación de una representación correcta de las necesidades de éste (Pastor et al., 1999), con lo cual se hace necesario la creación de herramientas a nivel de requisitos de software que facilitan la captura de esos requerimientos, su especificación y al mismo tiempo, generen automáticamente, la aplicación según las necesidades planteadas.

Al respecto, el desarrollo dirigido por modelos es un enfoque de desarrollo software que permite generar soluciones software a partir de la transformación de modelos a textos (M2T), que la OMG (2013), ha definido en un marco de trabajo para MDD conocido como MDA (Model Driven Architecture, arquitectura dirigida por modelos), con el fin de establecer un conjunto de estándares que soporta MDD. En este sentido, la MDA, es concebida como tecnología de modelación y construcción para MDD como

UML y estándares web (XML, SOAP, entre otros) respectivamente (Selic, 2003).

El desarrollo dirigido por modelos es típicamente usado para describir los enfoques de desarrollo software en los cuales, los modelos abstractos de sistemas software son creados y sistemáticamente transformados a implementaciones concretas (France y Rumpe, 2007); por lo tanto, los beneficios potenciales del uso de modelos de acuerdo con Selic (2003), son significativamente mayores en el software que en otras disciplinas de ingeniería debido a la posibilidad de una relación fluída entre los modelos y los sistemas que representan.

En consecuencia, se han identificado los siguientes framework que actualmente implementan MDD:

- El *OpenXAVA***,** es un framework de desarrollo rápido de aplicaciones empresariales utilizando Java. Dicho framework permite que a partir de la definición de clases en el lenguaje Java más la insercción de anotaciones, generar aplicaciones web con acceso a datos. Permite consultar, editar, agregar y eliminar los registros en la base de datos. Así mismo, admite establecer relaciones de pertenencia entre distintas clases (Paniza, 2011), Sin embargo, una debilidad de esta herramienta es que necesita tener instalado Eclipse y requiere de cierto grado de conocimiento para su configuración con el entorno integrado de desarrollo (IDE, por sus siglas en inglés).

- *El XEO*, es un framework de código abierto para diseñar y crear de manera ágil aplicaciones web empresariales utilizando el patrón MVC y el framework gráfico ExtJS, es una herramienta bastante potente, pues permite especificar requerimientos a partir de plantillas; sin embargo, no permite crear diagramas UML por lo que no hay una trazabilidad entre las etapas de comunicación y modelado. Así mismo, depende del IDE Eclipse y no soporte el trabajo en ambientes colaborativos (ltds, 2013).
- El framework de código abierto *AndroMDA* (2013), el cual permite generar aplicaciones web empresariales y tiene soporte con varios modeladores como ArgoUML11, No Magic18, Poseidon19. Su configuración requiere el manejo de diversas herramientas, definición de variables globales en el sistema y configuración de servidor de aplicaciones, lo cual la hace una herramienta muy poco usable para el desarrollo ágil de aplicaciones.

Método

Para el desarrollo de la presente investigación, se implementó la metodología de desarrollo ágil *Scrum*, dado que de acuerdo con Sutherland (2010), es un método diseñado para añadir energía, enfoque, claridad y transparencia en la planificación y ejecución de los proyectos software. Desde su nacimiento en 1993 hasta la actualidad, es una de las metodologías más populares en el mundo; es utilizada en empresas como Yahoo, Microsoft, Google, entre otras, debido a que algunass de sus características principales son aumentar la velocidad en el desarrollo de software y lograr una comunicación estable y consistente entre todos los niveles de desempeño.

Dado que en MDDCLOUD se especifican requisitos, se transforman a modelos y éstos se convierten en una aplicación web empresarial y las arquitecturas sólo incluyen dos pasos, los cuales son: la generación de modelos y la transformación a código; se diseña un metamodelo. Este representa un característica crucial debido a que es el encargado de definir el dominio del problema, de especificar los conceptos concernientes del mismo y de especificar la relación entre dichos conceptos.

Para el desarrollo del metamodelo se utilizó un Modelo de Dominio, el cual se ajusta a las necesidades de especificación de MDDCLOUD además que es estándar cuando se utiliza la arquitectura Eclipse Modelling Framework, en su herramienta GMF (Graphical Modelling Project).

La base para la conversión de un modelo directamente en la nube de una abstracción a otra se logró utilizando la librería *Backbone.js* y para la generación de la aplicación web el motor de plantillas *Handlebars.js*. En el metamodelo propuesto, MDDCLOUD, sus interrelaciones son las siguientes:

- Dependencia**:** la cual tiene representa la composición de un dominio y tiene dos subtipos *hasMany* y *hasOne* de esta forma, se le puede indicar a un dominio la composición por muchos o por un subdominio respectivamente.
- Abstracción: relaciona dos dominios que representan el mismo concepto en diferentes niveles de abstración.
- Uso**:** es utilizado cuando un dominio no es necesariamente compuesto por otro dominio, pero sí lo usa de alguna forma.

De esta manera, se considera *Project* como el dominio principal y elemento fundamental de MDDCLOUD, el cual representa los diferentes proyectos software que un usuario del *framework* propuesto puede tener y a la final, convierta en una aplicación web empresarial. Contiene los siguientes atributos:

- *Name*: representa el nombre del proyecto.
- *Description*: guardamos una pequeña descripción del proyecto.
- *CreationDate*: representa la fecha en que se creó el proyecto.
- *Template*: representa la plantilla que se utilizará para darle un aspecto
- visual a la aplicación generada, estas plantillas son libres y las encontramos en: https://bootswatch.com/

En la descripción de la aplicación desde el punto de vista MDD se tienen las transformaciones. Como se mencionó anteriormente, las arquitecturas estándar de MDD ya poseen diferentes herramientas y software diseñados estratégicamente para poder transformar los modelos (no metamodelos) ya sea a otros modelos o texto, código. Pero antes de involucrar la parte técnica es importante tener en cuenta algunos dominios y según la propuesta MDDCLOUD, cómo son transformados en distintos niveles de abstracción; además, dado el sistema colaborativo de MDDCLOUD en *real time*, se debió utilizar una implementación propia, pues implementar estos lenguajes sugeridos del lado web representaría un trabajo mucho mayor.

En este sentido, en MDDCLOUD, para desarrollar aplicaciones *MVC* o *API RESTFul* con *Node.js*, se estableció el siguiente orden de carpetas para el proyecto Backend:

- *app:* en esta carpeta se encuentran los módulos del servidor, archivos de configuración y el módulo para real time.
- *locales***:** se encuentran los JSON encargados para la internacionalización, los idiomas incluidos son inglés, español, francés y portugués.
- *models***:** se encuentra todos los modelos del sistema, cada uno de ellos referencia una colección en la base de datos Mongo. También se encuentra el módulo de conexión a la base de datos.
- *routes***:** en Express.js los routers se equiparan a controladores, los cuáles reciben las peticiones y las procesan.
- *test***:** se crean módulos de prueba.
- *views***:** aquí se encuentran todas las vistas utilizando el motor de plantillas Handlebars.js
- *backend.js***:** este es el módulo de inicialización.

Por otra parte, se seleccionó *Mongoose*, por ser una librería muy potente que permite mapear las colecciones de una base de datos Mongo, a archivos de modelo en la aplicación- Esto permite definir esquemas de los datos a guardar, también validaciones y su API, es muy sencilla. Así mismo, la selección del motor de base de datos es Persistencia, dado que algunos de los elementos tendidos en cuenta, fueron:

- Tipo: relacional, no-sql.
- Robustez.
- Conocimiento de la misma.
- Integración con el lenguaje Backend.
- Licencia, costos.

- Multiplataforma. Ver figura 1:

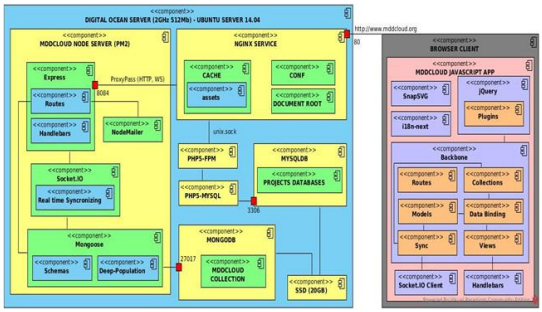

Figura 1. Diagrama de Componentes. - Arquitectura MDDCLOUD

Fuente: hhtp//www.mddcloud.org

Por lo tanto, para dibujar diagramas desde el navegador, transformar diagramas en conceptos, luego, conceptos a diagramas en tiempo real, sincronizar la información de un proyecto en tiempo real con otros usuarios que están viendo el mismo proyecto, utilizar una arquitectura de desarrollo modular que permitiese el trabajo en equipo y organización del proyecto. Se seleccionó *Frontend*, como la aplicación que corre del lado cliente. De acuerdo con lo expuesto, se seleccionó la librería *Browserify***,** que se encarga de modularizar una aplicación *Javascript* en diferentes archivos utilizando el estándar *Common.js*. Luego, al utilizarla en línea de comandos convertir todos esos archivos independientes en un único archivo con toda la lógica del proyecto.

- *Arquitectura MDDCLOUD*. Si se pudiera definir en una palabra la arquitectura de MDDCLOUD la palabra más acertada seguramente sería: *Javascript*. El *Javascript* ha sido un lenguaje que ha avanzado vertiginosamente (NPM, 2015). Sin embargo, aunque MDDCLOUD utiliza cómo núcleo *Javascript* al utilizarlo de lado *backend*, *frontend* e incluso a nivel de base de datos, es difícil poder definir su arquitectura en una sola palabra, existen otras muchas tecnologías con razón de ser, además de configuraciones extras para hacerla una aplicación en la nube consumible, como se muestra en la figura 2:

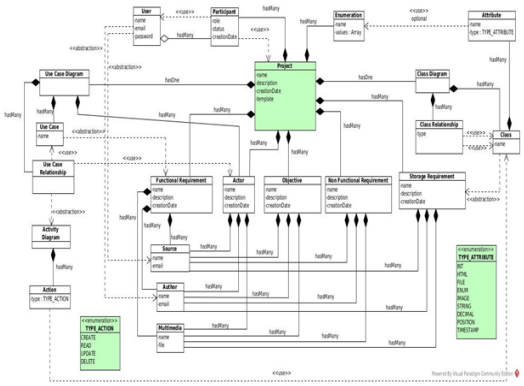

Figura 2. Metamodelo propuesto para MDDCLOUD

- El *backend* de la aplicación corresponde al conjunto de servicios que se encuentran del lado del servidor los cuales son consumidos del lado cliente. Normalmente, tiene un acceso privado, restringido, con acceso a las bases de datos y en otras cosas, es capaz de dar estadísticas, reportes y conectar diferentes clientes a una misma información. MDDCLOUD concebida en su momento como una aplicación en la nube, implicaba que el desarrollo de lado *backend* debía garantizar: x Excelente Performance/Rendimiento. x Capacidad de respuesta en menor tiempo. x Menor consumo de recursos, pues los usuarios son muchos y dejan todo en la nube. x Responder a la mayor cantidad de usuarios sin colapsar.

Finalmente, después de aprender las nociones básicas para desarrollar en *Javascript* de lado del servidor con *Node.js* hubo la necesidad de incorporar librerías y frameworks para hacer un desarrollo ágil pero a la misma vez potente, con lo cual fue muy importante el gestor de paquetes NPM (*Node Package Manager*) para acceder a librerías y aplicarlas a MDDCLOUD. En la figura 2 se presenta la estructura del metamodelo desarrollador para MDDCLOUD y en la tabla I la conversión de dichos conceptos al código y documentación generado, lo cual representa el aporte principal de esta investigación y que explica como requisitos, diagramas y código están relacionados y enlazados en una misma línea inseparable de desarrollo:

Tabla 1. *Conversión conceptos de la estructura del metamodelo*.

#	Atributo	Dominio	Transformación
1	Todos	Todos	Documentación.
2	Name	Project	Títulos en la aplicación generada.
3	Description	Project	Legenda en la página de inicio de la aplicación generada.
4	Name	Functional requirement	Nombre del caso de uso.
5	Name	Use case	Nombre de un menú en la aplicación generada, el cual sólo aparecerá para el rol que tiene la relación con el caso de uso.
6	Name	Storage requirement	Nombre de la clase.
7	Name	Class	Nombre de una tabla en el SQL generado.
8	Name	Attribute	Nombre de un atributo en una tabla del SQL generado.
9	Name	Attribute	Nombre de los campos de formularios concernientes a la clase en la aplicación generada.
10	Type	Attribute	Nombre de un tipo de atributo generado en el SQL.
11	Type	Attribute	Generación de vista HTML adecuada para que el usuario ingrese dicho atributo en un formulario, más validaciones.
12	Name	Actor	Nombre de los roles generados en el SQL.
13	Name	Actor	Se crean carpetas independientes para cada rol específico en la aplicación MVC generada.
14	No aplica	class relationship	Las relaciones entre clases crearán las relaciones uno a uno o uno a muchos entre las tablas SQL
15	No aplica	Use case relationship	Las relaciones entre casos de actores reflejaron los menús generales que tendrá cada rol en la aplicación generada.
16	Name	Storage requirement	Título que acompañara a cada paso generado en código de un diagrama de actividades.
17	No aplica	Activity diagram	Creará el algoritmo HTML/PHP/Javascript usado para reflejar el comportamiento del requisito funcional.
18	Type	Action	Acompañará en el título de cado paso generado en el código de un diagrama de actividades
19	No aplica	Action	Representa un llamado AJAX o IFRAME para consultar, insertar, actualizar o eliminar en un modelo.
20	Values	Enumeration	Representa los valores de las enumeraciones creadas en SQL.
21	No aplica	Class	Cada uno de los modelos en la arquitectura MVC de la aplicación generada.
22	No aplica	Functional requirement	Cada uno de los controladores y vistas en la arquitectura MVC de la aplicación generada.
23	Template	Project	La plantilla de bootstrap utilizada para los estilos de la aplicación generada.

Resultados

El framework MDD para la generación de aplicaciones web empresariales a partir de la especificación de requisitos de software, tiene los siguientes módulos:

- *Definición del Proyecto*: En esta fase se debe editar la información del proyecto accediendo al menú lateral Proyecto. Se debe escoger una plantilla la cuál será utilizada en el momento de la exportación de la aplicación, para el presente ejemplo se tomó la plantilla *UNITED*.

- *Definición de los Participantes*. Se deben agregar los participantes vía e-mail que harán parte del proyecto especificando su respectivo rol en el mismo. MDDCLOUD tiene tres roles disponibles, *ADMIN* para ser propietario del proyecto, *EDITOR* para poder hacer modificaciones de la información y *READER* para sólo poder observar la información sin hacer cambios. Si el email ingresado ya se encuentra en el sistema, entonces el usuario aparece con su nombre y foto de perfil.

- *Definición de las reuniones*. Las reuniones representan el archivo documental de las entrevistas con el cliente. MDDCLOUD permite subir audios, vídeos, imágenes y archivos binarios como odt, docx, pdf, etc. Los archivos tienen un nombre y descripción y sirven de soporte para los requisitos soportados, así se puede corroborar de qué reunión salió tal requisito.

- *Definición de Objetivos*. Los objetivos no hacen parte de la aplicación generada pero representan de vital importancia en la documentación generada.

- *Definición de Actores*. Los actores representan los roles generados en el sistema, son utilizados más adelante en el diagrama de casos uso.

- *Definición de Requisitos Funcionales*. Estas son elementos importantes pues representan los menús generados en la aplicación generada, también son usados en el diagrama de casos de uso.

- *Definición de los requisitos de almacenamiento de información*. Estos definen la información que debe ser guardaba en el sistema y también son usados en el diagrama de clases.

- *Definición de requisitos no funcionales*. Estos requisitos no influyen en la aplicación generada pero hacen parte vital de la documentación del proyecto.

- *Definición del Diagrama de Casos de Uso*. Los actores y casos de uso aparecen automáticamente pues han sido creados con las plantillas de actores y requisitos funcionales respectivamente, aquí el usuario debe crear las

relaciones y asignar permisos de ejecutar funciones a los actores.

- *Definición del diagrama de clases.* Las clases aparecen automáticamente por que han sido creadas en la plantilla de requisitos de almacenamiento de información, sin embargo aquí se deben agregar los atributos y las relaciones entre las clases.

- *Definición de los diagramas de actividades*. Al crear una relación en el diagrama de casos entre los actores y los casos de uso, se crea un diagrama de casos de uso, en el cuál se debe especificar la forma en que dicho actor realiza dicha acción. Para lo cual MDDCLOUD ya tiene definidas las acciones posibles: *CREATE, READ, UPDATE y DELETE* (CRUD). Las cuáles se aplican sobre las diferentes clases especificadas.

- *Aplicación Generada: Inicio*. En la página de inicio de la aplicación generada encontramos el nombre del proyecto, y la descripción del proyecto. En la parte superior el login de la aplicación. Y en la parte inferior la marca de que la aplicación fue creada con @mddcloud. Así mismo, los enlaces a la documentación y al archivo del proyecto web para descarga. También vemos, que se ha aplicado la plantilla CSS *UNITED* cómo especificamos en los primeros pasos.

- *Aplicación Generada: Login*. Al entrar al login se observa en la parte superior que está seleccionado el rol del usuario que ingresó. Después de este los requisitos funcionales como menú. Al final el nombre del usuario que ingresó, y el botón de salida. En la parte central un mensaje de bienvenida invitando a utilizar las opciones. Un punto importante es que los menús mostrados aparecen porque en el diagrama de casos de uso especificamos que dicho actor podría ejecutar esas funciones

- *Aplicación Generada: Ingreso a un Menú*. Al ingresar a un menú encontramos la lógica implementada en los diagramas de casos de uso, en los cuáles dimos interactividad a la aplicación. Un punto importante es que la información de tablas y formularios es presentada conforme a la definición de atributos en los diagramas de clases.

- *Wiki del Sistema*. Para la creación de la wiki para la comunidad de usuarios se utilizó el software MediaWiki (MediaWiki, 2015) el cual es una herramienta Open Source, con la que está creado Wikipedia.

Conclusiones

Algunas conclusiones de la realización de esta innovación tecnológica, son, entre otras:

Se comprueba que el desarrollo de aplicaciones web empresariales es posible mediante la automatización de conceptos.

Es posible desarrollar implementaciones amigables y en la nube de MDD.

La propuesta de transformación de MDDCLOUD funciona:

- Requisitos funcionales -> Casos de Uso -> Menús de la aplicación
- Requisitos de almacenamiento de información -> Diagrama de Clases -> Base de Datos -> SQL -> Formularios Web
- Diagramas de Actividades -> Secuencia normal de flujo de un requisito funcional -> Bloques de código de lógica en aplicación generada.
- Enumeraciones -> Enum SQL -> HTML Select.
- Actores -> Roles del sistema.
- Node.js como plataforma de desarrollo es idónea para construcción de aplicaciones con componentes en real-time.

Referencias Bibliográficas

AndroMDA. (2013). Generate components quickly with AndroMDA. Recuperado de http://www.andromda.org/index.html

Cohn, M. (2013). Agile Succedds Three Times More Often Than Waterfall. Mountain Goat Software. Recuperado de http://www.mountaingoatsoftware.com/blog/agile-succeeds-three-times-moreoften-than-waterfall.

France and Rumpe. R. (2007). Model-driven Development of Complex Software: A Research Roadmap. In 2007 Future of Software Engineering (FOSE '07). *IEEE Computer Society*, 37-54. DOI=10.1109/FOSE.2007.14 Recuperado de http://dx.doi.org/10.1109/FOSE.2007.14.

ITDS. XEO (2013). Extensible Enterprise Objects Community Edition. Recuperado de http://www.xeoframework.org/xportal/xmain?xpid=xeoopensource

MediaWiki (2015) Welcome to MediaWiki.org. Consultado el 22/05/2015. Recuperado de http://www.mediawiki.org/wiki/MediaWiki

NPM (2015) Node Package Manager. Consultado el 20 de mayo de 2015. Recuperado de https://www.npmjs.com/

OMG. MDA (2013). The Architecture Of Choice For A Changing World. Object Management Group. Recuperado de http://www.omg.org/mda/

Pan, A. (2008) Introducción al Desarrollo de Aplicaciones Empresariales. Universidade Da Coruña. Departamento de Tecnoloxías da Información e as Comunicacións (TIC).

Paniza, J. (2011) Learn OpenXava by example. Recuperado de http://www.openxava.org/en/book

Pastor, O., Canós, J., Ramos, I. (1999). From CASE to CARE Computer- Aided Requirements Enginnering. Universitat Politècnica de Valencia. Departament de Sistemes Informàtics i Computació. Recuperado de http://link.springer.com/chapter/10.1007%2F3-540-47866-3_19#page-1

Pascuas Rengifo, Y., Bocanegra, J., Ortiz, J. & Pérez, N. (2012). Desarrollo dirigido por modelos para la creación de laboratorios virtuales. *Scientia et Technica*

Pressman, R. (2005) *Ingeniería del Software: un enfoque práctico*. 6 ed. Mexico: Mcgraw-Hill Interamericana de México.

Selic, B. (2003) The Pragmatics of Model-Driven Development. *IEEE Software, IEEE Computer Socierty,* 20, (5), 19-25.

Sutherland, J. (2010). *Scrum Handbook. Somerville*. USA: Scrum Training Institute Press.a.