CIS1410IS06

AUTOMATIZACIÓN DEL DESARROLLO DE APLICACIONES WEB MEDIANTE EL ENFOQUE MDA-MDE

FERNEY SANTOS AGUILLÓN MARTÍNEZ

MARIO ALONSO MATEUS GÓMEZ

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERÍA
CARREA DE INGENIERÍA DE SISTEMAS
BOGOTÁ D.C.
2014

CIS1410IS06

AUTOMATIZACIÓN DEL DESARROLLO DE APLICACIONES WEB MEDIANTE EL ENFOQUE MDA-MDE

Autor(es):

Ferney Santos Aguillón Martínez

Mario Alonso Mateus Gómez

MEMORIA DEL TRABAJO DE GRADO REALIZADO PARA CUMPLIR UNO DE LOS REQUISITOS PARA OPTAR AL TÍTULO DE INGENIERO DE SISTEMAS

Director(a):

María Consuelo Franky de Toro

Jurados del Trabajo de Grado:

Ingeniero Jaime Andrés Pavlich Mariscal PhD.

Ingeniera María Catalina Acero Rozo

Página web del Trabajo de Grado:

http://pegasus.javeriana.edu.co/~CIS1410IS06

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERÍA
CARREA DE INGENIERÍA DE SISTEMAS
BOGOTÁ D.C.
2014

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERÍA
CARRERA INGENIERÍA DE SISTEMAS

Rector Magnífico

Jorge Humberto Peláez Piedrahita S.J.

Decano Académico Facultad de Ingeniería

Ingeniero Jorge Luis Sánchez Téllez

Decano del Medio Universitario Facultad de Ingeniería

Padre Antonio José Sarmiento Novoa S.J.

Director de la Carrera de Ingeniería de Sistemas

Ingeniero Germán Alberto Chavarro Flórez

Director del Departamento de Ingeniería de Sistemas

Rafael Andrés González Rivera

Artículo 23 de la Resolución No. 1 de Junio de 1946

"La Universidad no se hace responsable de los conceptos emitidos por sus alumnos en sus proyectos de grado. Sólo velará porque no se publique nada contrario al dogma y la moral católica y porque no contengan ataques o polémicas puramente personales. Antes bien, que se vean en ellos el anhelo de buscar la verdad y la Justicia"

AGRADECIMIENTOS

Agradecemos a Dios por acompañarnos durante el desarrollo del trabajo de grado, a nuestras familias por acompañarnos y brindarnos todo su apoyo.

Adicionalmente un agradecimiento a la profesora María Consuelo Franky por su colaboración y dedicación durante este proceso.

TABLA DE CONTENIDO

TABLA DE FIGURAS

LISTA DE TABLAS

ABSTRACT

This paper covers the problem of building web applications using an MDA (Model Driven Architecture) approach. The aim is to build a domain specific language (DSL) independent of technology with that can be represent web applications, and then generate a transformer to get the specific code in the Java EE 6 technology. Once generated is guaranteed that the application can automate significant parts of the business processes in a web application.

To work with the MDA approach in an enterprise 4 key roles for its implementation are introduced. The first role is the maintenance of meta-model, which will be responsible to modify or add elements to the meta-model, the second role is the Web application modeling, the third developer role is responsible for completing the generated code and finally the fourth role is the that maintain the transformer.

With the 4 roles generated in this new way of building software, can be seen a cultural change in the business, with the high costs that roles generated, but productivity increases in building web applications.

RESUMEN

Este trabajo de grado abarca el problema de construcción de aplicaciones Web mediante un enfoque MDA (Model Driven Architecture). Lo que se pretende lograr es construir un lenguaje especifico de dominio (DSL) independiente de tecnología con el que se pueda representar aplicaciones Web, y posteriormente, generar un transformador para obtener el código específico en la tecnología Java EE 6. Una vez generada la aplicación se garantiza la automatización de partes significativas de los procesos de negocio en una aplicación Web.

Para trabajar con el enfoque MDA en una empresa, se introducen 4 roles fundamentales para su ejecución. El primer rol es del mantenimiento del meta-modelo, que será el encargado de modificar o agregar elementos al meta-modelo, el segundo rol es el que modela la aplicación web, el tercer rol es del desarrollador encargado de completar el código generado y finalmente el cuarto rol es el que realiza el mantenimiento al transformador.

Con los 4 roles generados en esta nueva forma de construir software, se puede apreciar de un cambio cultural en la empresa, ya que se suben los costos por los roles generados, pero se incrementa la productividad en la creación de aplicaciones web.

RESUMEN EJECUTIVO

La necesidad de construir aplicaciones Web cada día es más común, debido a la gran demanda por parte de las empresas de tener los procesos de negocios disponibles en la nube, el problema de desarrollar estas aplicaciones es el tiempo invertido, la repetición de algunos procesos en cada proyecto y el desarrollo en distintas tecnologías.

Este trabajo de grado busca dar solución al problema descrito anteriormente mediante el enfoque MDA (Model Driven Architecture), donde se establece un lenguaje específico del dominio (DSL) independiente de tecnología que representa el meta-modelo y un transformador de código para Java EE6, este conjunto de herramientas desarrolladas se han denominado Wapp Generator. En el futuro la idea es que en una siguiente fase se pueda seguir construyendo transformadores para otras tecnologías como .NET a partir del mismo meta-modelo.

El enfoque del desarrollo dirigido por modelos establece como idea central que los artefactos son modelos y no programas, por lo que disminuye el coste e incrementa la calidad del software, adicionalmente aporta la automatización de partes significativas del proceso e implementación de componentes reutilizables por múltiples proyectos.

La base de este trabajo de grado es la creación del meta-modelo, en donde se identifica los elementos transversales de una aplicación web y se establecen las correspondientes meta-clases, para poder expresar el dominio del problema que se va a solucionar.

Para el desarrollo del meta-modelo se utilizaron herramientas Open Source, en caso particular Eclipse Modeling Project (Ver sección 1.2.3 Tecnologías de Soporte y herramientas para MDD), donde se realizó la creación del meta-modelo con sus respectivas meta-clases.

Las meta-clases modeladas en la herramienta Wapp Generator permiten la creación de una página Web, navegabilidad entre las páginas Web y finalmente indicar los roles asociados a cada página Web modelada; cabe resaltar que las meta-clases antes descritas se componen a su vez de otros elementos que definen el meta-modelo. (Ver sección 1.3 Definición de meta-modelo)

Un aspecto importante que se tuvo en cuenta en el desarrollo del meta-modelo, es que cada componente modelado no debe estar sujeto a ninguna tecnología concreta, el meta-modelo es un punto de inicio independiente de tecnología para comenzar en este caso una aplicación Web.

Una vez creado el meta-modelo se desarrolla el transformador de código sujeto a una tecnología concreta, para este caso se eligió Java EE 6 para poder verificar con una compañía de software los resultados de la herramienta.

El desarrollo del transformador se realizó con la ayuda de plantillas Acceleo, para la generación de código en Java EE 6, cada plantilla garantiza una función específica dentro del meta-modelo establecido. Dentro del conjunto de plantillas se genera el respectivo código referente a la navegabilidad de páginas, seguridad, archivos de configuración para la aplicación Web (web.xml), templates respecto a las páginas web entre otros. (Ver sección 1.4 Desarrollo del transformador)

El flujo de invocación de las plantillas depende en primera instancia si el atributo security se encuentra activo, el cual si es verdadero ejecuta una serie de plantillas referentes a la seguridad de la aplicación Web, si no invoca las plantillas correspondientes a los componentes modelados de la aplicación Web, entre estos se encuentran las plantillas de generar entidades, mangedBeans, ejbs y los respectivos CRUDs por entidades.

Una vez creado el meta-modelo y el transformador, queda como resultado una herramienta para modelar aplicaciones Web independiente de tecnología y un transformador de generación de código para Java EE 6.

Finalmente con la herramienta construida Wapp Generator, se identifican una serie de roles que proporcionan el mantenimiento y la gestión de la herramienta. El primer rol tiene el deber de mantener el meta-modelo, por lo que si se debe generar algún cambio en el meta-modelo él es el encargado de realizarlo, el segundo rol es el encargado de modelar la aplicación Web asociada al problema que se quiere solucionar, el tercer rol tiene como objetivo mantener el transformador y el último rol corresponde al programador que va a completar el código generado por la herramienta.

INTRODUCCIÓN

Este documento describe el proceso de desarrollo del Trabajo de Grado titulado "*Automatización del desarrollo de aplicaciones web mediante el enfoque MDA-MDE*", enmarcado dentro del grupo de investigación ISTAR del Departamento de Ingeniería de Sistemas de la Facultad de Ingeniería de la Pontifica Universidad javeriana.

En el transcurso de este Trabajo de grado, se propone aplicar la metodología MDA (Model Driven Architecture) - MDE (Model Driven Engineering) para la creación de aplicaciones web, definiendo un lenguaje de modelado para representar aplicaciones web independientes de tecnología.

Adicionalmente, con base en el modelado, se realiza un transformador de código para la tecnología específica JEE6 (Java Enterprise Edition 6).

En la *Sección I – Descripción general del Trabajo de Grado*, el lector encuentra la problemática a solucionar, la formulación del problema y, su justificación del por qué es viable realizar el Trabajo de grado. Adicionalmente, se presenta la descripción del proyecto especificando los objetivos y la metodología propuesta originalmente.

En la *Sección II – Marco Conceptual*, se introducen los conceptos y definiciones necesarios para la comprensión adecuada de este Trabajo de grado.

En la *Sección III – Marco Contextual*, se realiza un acercamiento de la situación actual del problema en el ámbito empresarial y académico.

En la *Sección IV – Desarrollo del trabajo*, se describe las herramientas que se utilizaron en la elaboración tanto del modelo como del transformador de código, además, se especifica el proceso de desarrollo en cada iteración del trabajo de grado.

En la *Sección V – Resultados y validación*, se presentan los resultados obtenidos durante el desarrollo del Trabajo de grado y, se valida mediante pruebas de aceptación al caso de estudio brindado por una empresa del sector de software, en este caso Heinsohn Business Technology [44].

En la *Sección VI – Conclusiones, recomendaciones y trabajos futuros*, se encuentran las conclusiones que se obtuvieron tras la realización del presente Trabajo de grado y los caminos que se abren a futuros Trabajos.

En la *Sección VII – Referencias y bibliografía*, se enumeran las referencias correspondientes al Trabajo de grado.

En la *Sección VIII – Glosario,* se encuentra el significado de algunas palabras que se utilizan en el documento.

En la *Sección IX – Anexos*, se enumeran los archivos adicionales al Trabajo de grado.

La presente memoria no pretende ahondar en los detalles técnicos de la solución propuesta ni del software implementado. Dichos detalles se profundizan en los documentos que se entregan como anexos.

I – DESCRIPCIÓN GENERAL DEL TRABAJO DE GRADO

1. Oportunidad, Problemática, Antecedentes

El crecimiento de empresas dedicadas a la construcción de software, hacen que cada día se necesite nuevos paradigmas, para garantizar calidad de software y no caer en retrasos a la hora de cumplir el cronograma en la entrega del producto [42].

En esta sección se busca identificar los antecedentes en los que se encuentra enmarcado este trabajo de grado, establecer una problemática concreta y dar una posible solución justificando del porque es necesario incursionar en este tipo de temas de desarrollo dirigido por modelos.

1.1 Descripción del contexto

Históricamente, el proceso de desarrollo de software ha resultado caro, riesgoso, incierto y demasiado tedioso para las condiciones de negocio modernas. Éstos inconvenientes dieron origen al concepto de "*Crisis del Software*" [9] que surgió conjuntamente con la creación de software. La crisis del software es un término informático acuñado en 1968, en la primera conferencia organizada por la OTAN (Organización del Tratado del Atlántico Norte) sobre desarrollo de software. La causa de esta crisis reside en la complejidad inherente a la tarea de construir sistemas de software y los cambios constantes a los que tiene que someterse el software para adaptarse a las necesidades cambiantes de los usuarios y a las innovaciones tecnológicas [15].

En la actualidad, un ejemplo particular para dichos problemas son las aplicaciones Web, que hacen frente a importantes dificultades en los sistemas de desarrollo. Muchas de estas dificultades surgen de la interacción entre las diversas áreas que forman la base de los sistemas Web, como lo son: los sistemas de información, el diseño gráfico y la seguridad, entre otras. Los desarrolladores tienen que enfrentar un sin fin de aspectos y pensar conjuntamente en una solución global que no afecte requerimientos funcionales y, garantice por otra parte atributos de calidad [19].

Sin embargo, el desarrollo de aplicaciones Web y de software en general poseen dos grandes limitaciones que son: el tiempo y la tecnología, cada proyecto de software debe realizarse en un tiempo específico y éste, a su vez, debe garantizar que sea compatible con una tecnología en un

largo periodo de tiempo [20], por lo que es necesario utilizar una metodología que ayude a lograr los objetivos planteados del proyecto durante el tiempo establecido; en el futuro, será deseable además, que los proyectos se diseñen de manera independiente de la tecnología.

Actualmente, el crecimiento en el tamaño y la complejidad de los sitios Web han causado problemas con respecto al desarrollo de las páginas Web, debido a que tienen poca reusabilidad, tiempos limitados de desarrollo y, difícil detección de errores. Con la ayuda del enfoque MDA/MDE, en el proceso de modelado de una aplicación Web, se busca poder desarrollar fácilmente la interfaz de usuario y parte de la lógica de negocio de la aplicación a través de un modelo independiente de tecnología, reduciendo la dificultad en el mantenimiento y evolución de las aplicaciones Web [3].

Las nuevas tendencias hacia el comercio electrónico, la seguridad de la información, la transformación de aplicaciones tradicionales (Legacy Systems) a aplicaciones con interfaces Web y, la expansión de Internet hacia nuevos servicios, hace pensar que este gran auge va continuar por mucho más tiempo. Se va a requerir de aplicaciones cada vez más sofisticadas y con mayor facilidad de uso para el usuario. Asimismo, es de esperar que se desarrollen más y mejores herramientas para el desarrollo de aplicaciones Web que apoyen tanto funcionalidad como el diseño gráfico de su interfaz y contenido [11].

1.2 Formulación del problema que se resolvió

¿De qué forma se puede generar código de manera correcta y eficiente para la creación de aplicaciones Web mediante la metodología MDA/MDE en el desarrollo de páginas Web?

1.3 Justificación

En el mundo de hoy y en un futuro, es cada vez más evidente que el desarrollo de software se realizará por medio del enfoque MDA (Model Driven Architecture) - MDE (Model Driven Engineering), ya que propone una metodología de creación de modelos, o abstracciones independientes de tecnología, para hacer transformaciones a código [2].

Los beneficios que brinda el enfoque MDA (Model Driven Architecture) son: productividad independiente de cómputo y tecnología específica, portabilidad para cualquier plataforma, inter-operatividad generando puentes entre plataformas tecnológicas, mantenimiento y documentación del software [20].

Si se hace un desarrollo de software sin utilización de modelos, el programador enfrenta continuos cambios en las tecnologías de implementación, lo que implica realizar importantes

cambios en el diseño de la aplicación para integrar diferentes tecnologías, para éste problema, MDE (Model Driven Engineering) ofrece una alternativa para el desarrollo de software independiente a la plataforma tecnológica y en donde va a operar es independiente de cualquier tecnologia, esto es debido a la utilización de estándares establecidos por la OMG, la cual es una organización que propone estándares para el desarrollo de software orientado a objetos, reconocido por el establecimiento de diferentes estándares como UML , XMI y CORBA [2].

Este trabajo de grado se justifica realizarlo porque beneficiará en productividad y tiempo a los desarrolladores en el proceso de construcción de aplicaciones Web a través del enfoque MDA (Model Driven Architecture) - MDE (Model Driven Engineering), obteniendo como resultado una herramienta para modelar aplicaciones independiente de tecnología y un transformador de código para JEE6 (Java Enterprise Edition 6).

1.4 Impacto Esperado

Al terminar con el desarrollo del trabajo de grado se espera poder determinar el impacto en diferentes contextos, para establecer si realmente hubo un cambio significativo positivo a la hora de llevar a cabo la solución del problema planteado. (Ver sección 1.2 Formulación del problema que se resolvió)

1.4.1 Impacto Ambiental

El desarrollo de este trabajo de grado no beneficiará ni perjudicará al medio ambiente, ya que al ser un proyecto de software sólo se requerirá el consumo de energía de los computadores y los servicios públicos necesarios.

El resultado de este proyecto es una herramienta de software (intangible) que no causará impacto ambiental en ningún aspecto.

1.4.2 Impacto Social

Al ser este un proyecto de desarrollo de software, con el que se intentan resolver aspectos de carácter técnico, no se espera tener ningún impacto social directo. Sin embargo se intenta mostrar un enfoque de construcción de software poco utilizado para incentivar a los desarrolladores a la hora de desarrollar software.

1.4.3 Impacto Tecnológico

Debido a que este es un proyecto de desarrollo de software con metodologías relativamente recientes se considera de gran impacto tecnológico, debido a que aportará conocimientos sobre

el desarrollo dirigido a modelos a la construcción de software en aplicaciones Web, a través de modelos para la creación de aplicaciones independientes de tecnología.

La trasformación de un modelo a tecnología Java EE 6 permitirá a los desarrolladores poder implementar de mejor forma una interfaz de usuario y lógica asociada a tecnología JEE6 (Java Enterprise Edition 6), ganando tiempo y productividad.

2. Descripción del proyecto

En esta sección se encuentra el objetivo general y la metodología que se propone para cumplir con los objetivos específicos propuestos en el trabajo de grado.

2.1 Visión Global

El desarrollo de este proyecto dio como resultado una herramienta que permite modelar aplicaciones Web independientes de cualquier tecnología y por otra permite generar código para Java EE 6.

La herramienta desarrollada se encuentra dentro del enfoque MDA-MDE y describe una variedad de funcionalidades para la navegabilidad, seguridad, diseño, descripción de entidades de negocio y procesos de negocio en un entorno de aplicación Web.

Finalmente se obtiene como producto final la herramienta descrita anteriormente que cumple los objetivos específicos enmarcado dentro de este trabajo de grado. (Ver sección 2.3 Objetivos Específicos)

2.2 Objetivo General

Aplicar la metodología MDA-MDE para la creación de aplicaciones web, definiendo un lenguaje de modelado para representar una aplicación Web e implementar el transformador para la tecnología JEE6 (Java Enterprise Edition 6), que toma un modelo y genera su respectivo código fuente de páginas y clases Java.

2.3 Objetivos Específicos

1. Analizar las Herramientas utilizadas por la empresa de desarrollo de software que constituyen el caso de estudio, para identificar los elementos que compondrán el meta-modelo.

2. Investigar Herramientas MDA para hacer transformaciones que generen código fuente, enfatizando en las más usadas y en las que ofrezcan mayores beneficios para nuestro proyecto (curva de aprendizaje, interoperabilidad, uso de estándares, etc.).
3. Profundizar en tecnologías JEE6, enfatizando en el desarrollo web dado que el foco del proyecto es implementar un primer transformador para JEE6.
4. Aprender tecnologías empleadas para el desarrollo del transformador, como manejo de plantillas u otros componentes que sean necesarios.
5. Definir un meta modelo, en donde se establezcan los diferentes elementos del modelo para las aplicaciones Web, como el flujo de las páginas web y las clases para el soporte de las páginas.
6. Desarrollar un transformador que genere la aplicación web en la tecnología Java EE6 (Con interfaz de páginas JSF).

2.4 Metodología que se propuso para satisfacer cada objetivo

En esta sección se muestra la metodología que se propuso para este trabajo de grado.

2.4.1 Fase de Comprensión

La fase de comprensión permite alcanzar el primer objetivo específico. Por medio de las siguientes actividades:

- **Análisis de herramientas**

Por medio del análisis de herramientas de una casa desarrolladora, es posible la identificación de los diversos elementos que harán parte del modelo, permitiendo de esta forma el diseño de las aplicaciones Web.

- **Definición de elementos**

Una vez analizadas las diversas herramientas utilizadas para el desarrollo de páginas web, se procede a definir la tecnología que soporta el transformador, el alcance y complejidad de la aplicación web.

2.4.2 Fase de Investigación

La fase de investigación permite alcanzar el primer objetivo específico. Las siguientes son las actividades necesarias para alcanzar este objetivo.

- **Obtención y localización de bibliografía**

Actividad relacionada con la búsqueda bibliográfica sobre la metodología MDA y MDE por medio de libros, bases de datos, tesis, y a través de personas que estén relacionadas con el tema, para consolidar una bibliografía inicial.

- **Selección de Bibliografía**

Se seleccionaron los documentos de acuerdo a la relevancia que tengan estos dentro de la metodología MDA y MDE.

- **Análisis de la Bibliografía**

En esta actividad se utiliza por método el análisis de la información, de acuerdo a la relevancia que tenga la bibliografía dentro de la metodología MDA – MDE.

- **Organización de la Bibliografía**

En esta actividad la bibliografía seleccionada, se organiza por medio de etiquetas, notas y resúmenes que permitan una clasificación coherente de la bibliografía para su posterior búsqueda.

2.4.3 Fase de Estudio de Plataforma JEE

La fase de Estudio de Plataforma permite alcanzar el segundo objetivo específico. Las siguientes son las actividades necesarias para alcanzar este objetivo.

- **Investigación de bibliografía**

Buscar material bibliográfico en las diferentes tecnologías usadas en JEE para la creación de páginas web, enfocándose en los elementos y características que se presentan en estas tecnologías.

2.4.4 Fase de análisis de Herramientas

La fase de análisis de herramientas permite establecer las herramientas que se van a utilizar para desarrollar la herramienta Wapp Generator.

- **Tutorial de las Herramientas MDA y MDE**

En esta etapa se busca desarrollar habilidades y familiarizarse en las diferentes herramientas que existen para la metodología MDA y MDE, por medio de tutoriales.

- **Elección de las herramientas a utilizar**

Se define el conjunto de herramientas apropiado para poder llevar a cabo el desarrollo del transformador, de acuerdo a las necesidades que se deben resolver.

2.4.5 Fase de definición del meta-modelo

La fase de definición del meta-modelo permite alcanzar el cuarto objetivo específico. Cabe resaltar que dentro de esta fase se estableció una metodología ágil SCRUM para llevar a cabo pequeños incrementos en cada etapa del trabajo de grado. Las siguientes son las actividades necesarias para alcanzar este objetivo.

- **Definición del Meta-modelo**

En esta actividad se define los elementos que harán parte del meta-modelo para la modelación de las aplicaciones Web que serán generadas por el transformador.

- **Validación del Meta-modelo**

En esta actividad se verifica que el meta-modelo definido en la anterior actividad sea coherente y completo de acuerdo a la modelación de páginas web.

- **Documentación del Meta-modelo**

En esta actividad se documenta el meta-modelo definido en la primera actividad, detallando los elementos que componen el meta-modelo, sus características y relaciones entre estos elementos.

2.4.6 Fase de desarrollo del transformador

En esta etapa se desarrolla el transformador, identificando los diversos módulos existentes en el transformador para JSF, cabe resaltar que para esta fase también se usó la metodología SCRUM para aumentar en cada iteración un incremento del producto a entregar.

- **Desarrollo del transformador**

En esta etapa se lleva a cabo el desarrollo del transformador, de forma coherente con el diseño del transformador.

- **Validación del transformador**

En esta etapa se busca que el transformador logre generar código coherentemente tomando como base el caso de estudio definido anteriormente.

II – MARCO CONTEXTUAL

1. Contexto Actual

El desarrollo de software ha sido considerado tanto un arte como una ciencia [41]. Como ciencia, esta disciplina se fundamenta en la aplicación de prácticas de ingeniería que permiten estimar, medir y evaluar el proceso de desarrollo, de manera repetitiva y controlada. Como arte, el desarrollo de software requiere producir resultados que reflejen ingenio y habilidad para seleccionar, diseñar y construir el producto software que mejor satisfaga los requerimientos de una organización [42].

En los últimos años se han experimentado grandes cambios en el área de la informática, tanto en la proliferación de nuevas tecnologías, como en las metodologías para la construcción de software, ya que los cambios constantes en los requerimientos han repercutido en la forma de hacer y ejecutar software en las organizaciones actuales. La explosión del uso de internet en las empresas, plantea varias ventajas y desafíos para los procesos de negocio que necesitan embarcarse en el ámbito WEB, ya que, la forma en que informatizan sus procesos e interactúan con otras organizaciones no es suficientes. Lo que se hace notorio es que una necesidad que antes pudo ser medianamente satisfecha con diversidad de enfoques y tecnologías, actualmente está requiriendo respuestas más integradas que contemplen el centro del negocio en las organizaciones [41].

Las organizaciones intentan conjuntar dos visiones para realizar su negocio: la visión del negocio centrada en especificar y mejorar sus procesos mediante análisis del negocio y, la visión de TI centrada en informatizar dichos procesos evolucionando en la tecnología y metodologías de desarrollo de software. En general esta conjunción ha sido compleja y problemática sin alcanzar una visión común del negocio por ambas partes. Sin embargo, las organizaciones son cada vez más dependientes de sus sistemas informáticos, cuentan con diversidad de sistemas que tienen entre sí dependencias complejas donde estos sistemas han ido creciendo en forma separada y heterogénea. Los avances en tecnología y los cambios en los requerimientos del negocio se retroalimentan y deben ser gestionados [40].

Los sistemas de software son cada vez más herramientas para el día a día tanto en el trabajo como en los hogares. Las Organizaciones en que son usadas estas aplicaciones deben “encajar” en el trabajo diario de las personas, dando valor a las tareas realizadas, así como permitir cambios asociados con la realización de las mismas. Un objetivo importante de las organizaciones actuales es el modelado e informatización de sus procesos del negocio, el

monitoreo y la mejora de los mismos a partir de los datos de ejecución obtenidos. Se hace necesario contar con elementos y enfoques para realizar este modelado, diseño e implementación de procesos del negocio, como forma de cubrir las expectativas de las organizaciones [42].

Dos grandes enfoques para modelar los procesos del negocio se presentaron, desde el punto de vista del desarrollo de software: con entrada de modelado realizado por analistas del Negocio luego traducidos, con BPMN y BPML; como parte del modelado realizado directamente por el proyecto de software y con UML como parte del modelado del negocio para que el desarrollo de software sea exitoso en cuanto al cubrimiento de las expectativas del desarrollo por parte de la organización [41].

No parece ser suficiente, se hace necesario también un cambio de enfoque en la construcción del software. Los conceptos manejados por las aplicaciones deben ser los del negocio, además de la infraestructura de software. El enfoque de diseño orientado a servicios promete ayudar a cerrar ese gap entre el negocio y el software para soportarlo. El enfoque de desarrollo basado en modelos promete proveer herramientas que ayuden a la automatización de los modelos realizados [40].

2. Contexto Empresarial

La rápida evolución de la tecnología informática ha propiciado la definición de nuevos servicios y esquemas de trabajo en las organizaciones, que le permiten a éstas mejorar la calidad de servicio a sus clientes, definir nuevos servicios, conquistar nuevos mercados y, en general, ser más competitivas. La operación de la compañía se encuentra cada vez más soportada en sistemas de información intensivos en software los cuales son fundamentales para apoyar el liderazgo estratégico de la organización en el mercado o, por el contrario, propiciar su fracaso. Estas condiciones de contexto generan permanentes demandas a la Ingeniería de software, con una premisa fundamental de la calidad.

Una de las empresas Colombianas con mayor reconocimiento en la construcción de software es Heinsohn Business Tecnology [44], destaca entre las 10 empresas de Colombia con mayor calidad de desarrollo de software [45], la organización se encuentra actualmente con un nivel 5 de CMMI (Capability Maturity Model Integration), prestando servicios para la gestión estratégica de las organizaciones, gestión humana, colaboradores, nómina y relaciones laborales.

Uno de sus procesos internos para la construcción rápida de software es el generado de páginas WEB, el cual logra generar código para la creación de aplicaciones web en una tecnología

como JEE5. Uno de los principales problemas de este generador de páginas WEB es la dependencia con la tecnología, ya que tanto su modelado y transformación tienen asociada una tecnología específica, además carece de desarrollo dirigido por modelos por lo que se vuelve dependiente de tecnología y fuertemente acoplado.

Problemas como los anteriores llevan a pensar si se puede agilizar el desarrollo de algunos procesos para la construcción de software sin tener que arriesgar la organización.

3. Contexto Académico

Existe una brecha marcada entre lo que se enseña en el aula de clase y la realidad del desarrollo de software en las organizaciones. Esta situación ha motivado una reflexión en el mundo académico, acerca de las competencias y habilidades que deben ser desarrolladas en los futuros desarrolladores de software y las estrategias pedagógicas que pueden ser utilizadas de manera que sus experiencias de aprendizaje estén altamente influenciadas por las prácticas, técnicas y modos de trabajo que exige el desarrollo de software de calidad a escala industrial [43].

III - MARCO CONCEPTUAL

El desarrollo de aplicaciones web involucra decisiones no triviales de diseño e implementación que inevitablemente influyen en todo el proceso de desarrollo, afectando la división de tareas. Los problemas involucrados, como el diseño del modelo del dominio y la construcción de la interfaz de usuario, tienen requerimientos disjuntos que deben ser tratados por separado [1].

Las tecnologías usadas en este proyecto se subdividen en 2, tecnologías para modelar y transformar por medio del enfoque MDE y tecnologías utilizadas en el transformador para convertir a código fuente.

La metodología de desarrollo enmarcada dentro de este trabajo de grado fue SCRUM, para gestionar las tareas y actividades correspondientes al desarrollo de la herramienta Wapp Generator.

1. Metodología Ágil SCRUM

SCRUM es un proceso para la gestión y control del producto que trata de eliminar la complejidad en estas áreas para centrarse en la construcción de software que satisfaga las necesidades del negocio. Es simple y escalable, ya que no establece prácticas de ingeniería del software sino que se aplica o combina, fácilmente, con otras prácticas ingenieriles, metodologías de desarrollo o estándares ya existentes en la organización [49].

SCRUM se concentra, principalmente, a nivel de las personas y equipo de desarrollo que construye el producto. Su objetivo es que los miembros del equipo trabajen juntos y de forma eficiente obteniendo productos complejos y sofisticados. Podríamos entender SCRUM como un tipo de ingeniería social que pretende conseguir la satisfacción de todos los que participan en el desarrollo, fomentando la cooperación a través de la auto-organización. De esta forma se favorece la franqueza entre el equipo y la visibilidad del producto. Pretende que no haya problemas ocultos, asuntos u obstáculos que puedan poner en peligro el proyecto. Los equipos se guían por su conocimiento y experiencia más que por planes de proyecto formalmente definidos. La planificación detallada se realiza sobre cortos espacios de tiempo lo que permite una constante retroalimentación que proporciona inspecciones simples y un ciclo de vida adaptable. Así, el desarrollo de productos se produce de forma incremental y con un control empírico del proceso que permite la mejora continua [50].

En la figura 1 se muestra el ciclo de vida de la metodología SCRUM:

Figura 1: Ciclo de vida SCRUM

La planificación de cada sprint hace referencia a las actividades establecidas para desarrollar el producto de software, según esa planificación se realiza un seguimiento iterativo y unas reunión al finalizar cada sprint para poder revisar y retroalimentar el incremento del producto que se está realizando [50].

1.1 Ejecución de la iteración (Sprint)

Se realizan las actividades y tareas correspondientes en el sprint, dejando un pequeño incremento del producto [49].

1.2 Revisión de iteración (Sprint Review)

Es necesario que al finalizar cada sprint se realice una reunión con los stakeholders, para retroalimentar los resultados de la ejecución del sprint y así corregir los posibles errores que se hayan presentado [49].

1.3 Lista de tareas de la iteración (Back log)

Esta lista de tareas permite visualizar el estado actual de las actividades programadas en cada iteración, mitigando los escenarios de riesgo más críticos del proyecto como retrasos en las entregas o los cambios sujetos en cada revisión del proyecto [50].

2. Tecnologías de desarrollo dirigido por modelos

La ingeniería de software establece que el problema de construir software debe ser encarado de la misma forma en que los ingenieros construyen otros sistemas complejos, como puentes, edificios, barcos y aviones. La idea básica consiste en observar el sistema de software a construir como un producto complejo y a su proceso de construcción como un trabajo ingenieril [15]. Es decir, un proceso planificado basado en metodologías formales apoyadas por el uso de herramientas.

A lo largo de estos años hemos visto surgir el Desarrollo de Software Dirigido por Modelos (MDD) como una nueva área dentro el campo de la ingeniería de software. MDD plantea una nueva forma de entender el desarrollo y mantenimiento de sistemas de software con el uso de modelos como principales artefactos del proceso de desarrollo. En MDD, los modelos son utilizados para dirigir las tareas de comprensión, diseño, construcción, pruebas, despliegue, operación, administración, mantenimiento y modificación de los sistemas [20].

El contenido de esta sección describirá el proceso de desarrollo dirigido por modelos, partiendo de forma general hasta llegar al detalle del proceso.

La figura 2 visualiza el contenido que se explicará:

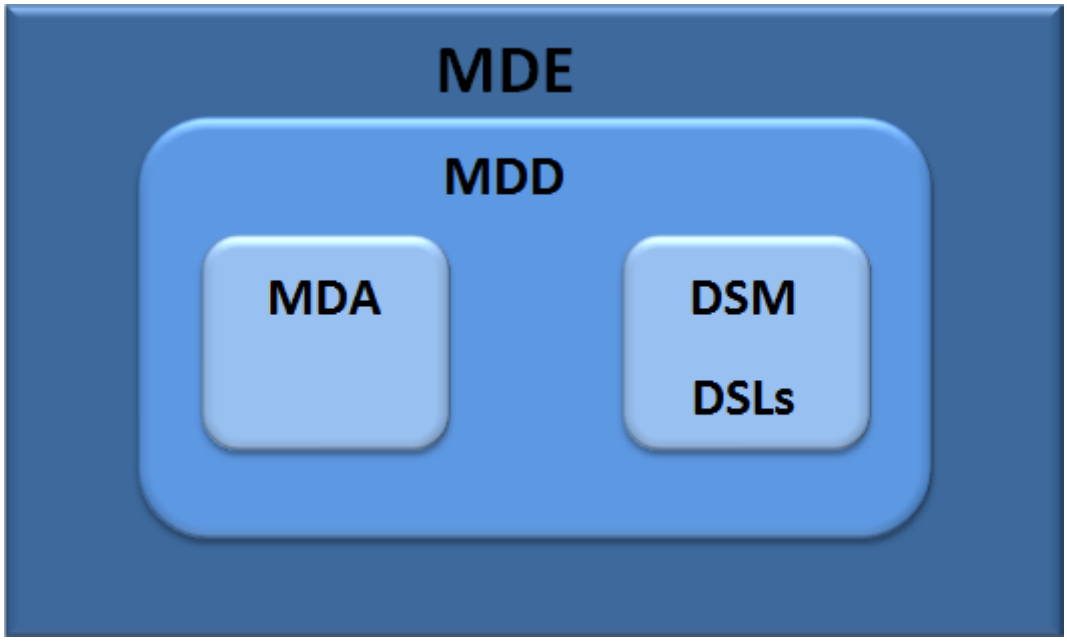

Figura 2: Esquema de la ingeniería dirigida por modelos [15]

2.1 Desarrollo de Software dirigido por modelos

La ingeniería dirigida por modelos MDE (por sus siglas en inglés Model Driven Engineering) hace referencia al uso sistemático de modelos, como los elementos principales en la ingeniería de software, durante el ciclo de vida de un proyecto. Su enfoque principal consiste en que los artefactos fundamentales del desarrollo de software son los modelos y no los programas [15].

La instancia particular de MDE para el enfoque de desarrollo de software es MDD (por sus siglas en inglés Model Driven Software Development), ya que mejora el proceso de construcción de software basándose como paradigma en el proceso guiado por modelos y soportándose de potentes herramientas para su desarrollo [21]. Los modelos se van generando desde los más abstractos a los más concretos a través de pasos de transformación o refinamientos, hasta llegar al código aplicando una última transformación [6].

Los puntos clave de la iniciativa MDD [22] son:

- El uso de un mayor nivel de ***abstracción*** en la especificación tanto del problema a resolver como de la solución correspondiente, en relación con los métodos tradicionales de desarrollo de software.
- El aumento de confianza en la ***automatización*** asistida por computadora para soportar el análisis, el diseño y la ejecución.
- El uso de ***estándares*** industriales como medio para facilitar las comunicaciones, la interacción entre distintas aplicaciones y productos, y la especialización de tecnología.

2.1.1 Abstracción

El enfoque de MDD para incrementar los niveles de abstracción es definir lenguajes de modelado específicos de dominio cuyos conceptos reflejen estrechamente los conceptos del domino del problema, mientras se ocultan o minimizan los aspectos relacionados con las tecnologías de implementación. Estos lenguajes utilizan formas sintácticas que resultan amigables y que transmiten fácilmente la esencia de los conceptos del dominio. Por otra parte el modelo permite reducir el impacto que la evolución tecnológica impone sobre el desarrollo de aplicaciones, permitiendo que el mismo modelo abstracto se materialice en múltiples plataformas de software [22].

2.1.2 Automatización

La automatización es el método más eficaz para aumentar la productividad y la calidad. En MDD la idea es utilizar a las computadoras para automatizar tareas repetitivas que se puedan

mecanizar, tareas que los seres humanos no realizan con particular eficacia. Esto incluye, entre otras, la capacidad de transformar modelos expresados mediante conceptos de alto nivel, específicos del dominio, en sus equivalentes programas informáticos ejecutables sobre una plataforma tecnológica específica. Además las herramientas de transformación pueden aplicar reiteradas veces patrones y técnicas con éxito ya comprobado, favoreciendo la confiabilidad del producto [22].

2.1.3 Estándares

MDD debe ser implementado mediante una serie de estándares industriales abiertos. Estas normas proporcionan numerosos beneficios, como por ejemplo la capacidad para intercambiar especificaciones de herramientas complementarias, o entre herramientas equivalentes de diferentes proveedores [22].

El desarrollo dirigido por modelos permite mejorar las prácticas corrientes de desarrollo de software [15]. Las ventajas de MDD son las siguientes:

- **Incremento en la productividad:** MDD reduce los costos de desarrollo de software mediante la generación automática de código y otros artefactos a partir de los modelos, lo cual incrementa la productividad de los desarrolladores [22].
- **Adaptación a los cambios tecnológicos:** El progreso de la tecnología hace que los componentes de software se vuelvan obsoletos rápidamente. MDD ayuda a solucionar este problema a través de un arquitectura fácil de mantener donde los cambios se implementan rápida y consistentemente, habilitando una migración eficiente de los componentes hacia las nuevas tecnologías [22].
- **Adaptación a los cambios en los requisitos:** Poder adaptarse a los cambios es un requerimiento clave para los negocios, y los sistemas informáticos deben ser capaces de soportarlos. Cuando usamos un proceso MDD, agregar o modificar una funcionalidad de negocios es una tarea bastante sencilla, ya que el trabajo de automatización ya está hecho. Cuando agregamos una nueva función, solo necesitamos el desarrollar el modelo específico para esa nueva función. El resto de la información necesaria para generar artefactos de implementación ya ha sido capturada en las transformaciones y puede ser re-usada [22].
- **Consistencia:** La aplicación manual de las prácticas de codificación y diseño es una tarea propensa a errores. A través de la automatización de MDD favorece la generación consistente de los artefactos [22].
- **Re-uso:** En MDD se invierte en el desarrollo de modelos y transformaciones. Esta inversión se va amortizando a medida que los modelos y las transformaciones son re-

usados. Por otra parte el re-uso de artefactos ya probados incrementa la confianza en el desarrollo de nuevas funcionalidades y reduce los riesgos ya que los temas técnicos han sido previamente resueltos [22].

- **Los modelos son productos de larga duración:** En MDD los modelos son productos importantes que capturan lo que el sistema informático de la organización hace. Los modelos de alto nivel son resistentes a los cambios a nivel plataforma y sólo sufren cambios cuando lo hacen los requisitos de negocio [22].
- **Posibilidad de demorar las decisiones tecnológicas:** Cuando aplicamos MDD, las primeras etapas del desarrollo se focalizan en las actividades de modelado. Esto significa que es posible demorar la elección de una plataforma tecnológica específica o una versión de producto hasta más adelante cuando se disponga de información que permita realizar una elección más adecuada [22].

2.2 Propuestas concretas para MDD

Entre las propuestas más concretas y utilizadas en el ámbito MDD son, por un lado MDA desarrollada por el OMG (Object Management Group) [46] y por otro lado el modelado específico del dominio (DSM acrónimo inglés de Domain Specific Modeling) acompañado por los lenguajes específicos del dominio (DSLs acrónimo inglés de Domain Specific Languaje) [3]. Ambas iniciativas guardan naturalmente una fuerte conexión con los conceptos básicos de MDD. Específicamente, MDA tiende a enfocarse en lenguajes de modelado basados en estándares OMG, mientras que DSM utiliza otras notaciones no estandarizadas para definir sus modelos [23].

2.2.1 Arquitectura Dirigida por Modelos (MDA)

La arquitectura dirigida por modelos (MDA acrónimo inglés de Model Driven Architecture) es un concepto promovido por OMG (acrónimo inglés de Object Management Group) a partir del año 2000, con el objetivo de afrontar los desafíos de integración de las aplicaciones y los continuos cambios tecnológicos. MDA igualmente es una arquitectura que proporciona un conjunto de guías para estructurar especificaciones expresadas como modelos, siguiendo el proceso MDD [20].

MDA añade al enfoque dirigido por modelos la inclusión de varios niveles de abstracción (CIM -Computation Independent Model-, PIM -Platform Independent Model-, PSM -Platform Specific Model-) y varias transformaciones entre niveles, realizando de esta manera descripciones del sistema a varios niveles de [2].

En la figura 3 se representa el planteamiento de desarrollo de MDA:

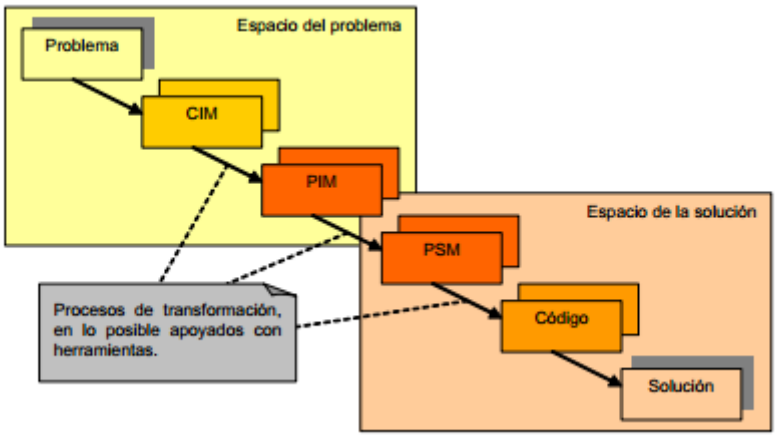

Figura 3: La transformación en el proceso basado en MDA [7]

Los pasos de desarrollo son:

1. Los requerimientos del sistema se presentan en un modelo CIM, que describe la situación en que el sistema será usado.
2. El modelo CIM es transformado en un modelo PIM que describe el sistema, pero no muestra los detalles de su uso en una plataforma tecnológica particular.
3. Después de obtener el modelo PIM se realiza otra transformación hacia el modelo PSM, que contiene el detalle necesario para utilizar la plataforma tecnológica en que el sistema funcionará.
4. Por último teniendo el modelo PSM se realiza una transformación que resulta en la generación de código para lograr una solución o modelo ejecutable.

2.2.2 Modelo Específico del Dominio (DSM y DSLs)

DSM (Domain-Specific Modeling) es principalmente conocida como la idea de crear modelos para un dominio, utilizando un lenguaje focalizado y especializado para dicho dominio. Estos lenguajes se denominan DSLs (por su nombre en inglés: Domain Specific Languaje) y permiten especificar la solución usando directamente conceptos del dominio del problema [22]. Los productos finales son luego generados desde estas especificaciones de alto nivel. Esta automatización es posible si ambos, el lenguaje y los generadores se ajustan a los requisitos de un único dominio [23].

2.2.3 Tecnologías de Soporte y herramientas para MDD

Las tecnologías de soporte proporcionan el ambiente para modelar y transformar el problema planteado en el trabajo de grado.

Las siguientes herramientas proporcionan diferentes utilidades para el desarrollo dirigido por modelos, en estas se encuentran:

A. Eclipse Modeling Project

Eclipse Modeling Project es el principal mecanismo de evolución y promoción de las tecnologías de desarrollo basadas por modelos, el proyecto principal es el Eclipse Modeling Framework (EMF) [25], que permite la creación de modelos a partir de interfaces Java, esquemas XML o diagramas UML, para posteriormente re alizar la generación del código que los implementa, o cualquiera de las representaciones del modelo.

En tiempo de desarrollo puede ser usado como la fuente principal de información del generador de EMF. En tiempo de ejecución es usado para determinar el correcto comportamiento del modelo [24].

EMFtext es un conjunto de plug-ins de Eclipse, altamente integrado con EMF, que permite derivar sintaxis textuales a partir de lenguajes concretos de meta-modelo, refinar dicha sintaxis y crear herramientas que permiten crear y manipular modelos según dicha sintaxis [25].

B. OptimalJ

Es una herramienta MDA que utiliza Meta Object Facility (define un lenguaje abstracto para definir lenguajes de modelado) para soportar estándares como UML y XMI. OptimalJ permite generar transformaciones para generar código fuente a aplicaciones Java EE6 completas a partir de un PIM o plataforma independiente del modelo [4].

En OptimalJ existen tres tipos de modelos:

- Modelo del Dominio: modelo que describe el sistema a un alto nivel de abstracción. Corresponde al PIM de la aplicación y su elemento principal es un modelo de clases del negocio [15].
- Modelo de la Aplicación: modelo del sistema desde el punto de vista de una tecnología determinada (Java EE6). Contiene los PSM de la aplicación [4].
- Modelo de Código: código de la aplicación, generado a partir de un modelo de la aplicación [15].

C. ArcStyler

Es una herramienta basada en MOF (Meta Object Facility) un lenguaje utilizado para crear meta-modelos. Permite expresar metadatos (igual que XML), es independiente de la plataforma, y está descrito con la notación UML y Object Constraint Language (OCL). Cada elemento del lenguaje se representa mediante una clase y sus propiedades como atributos. Las relaciones entre elementos se representan como asociaciones e incluye la generalización permitiendo expresar que un elemento es una especialización de otro [47].

Permite generar transformaciones de código de n capas codificadas en java/J2EE o c#/.NET a partir de diagramas UML y la especificación de los procesos del negocio, además permite extender las capacidades de transformación, generando componentes a partir de UML, también incluye herramientas relacionadas con el modelado del negocio y el modelado de requisitos por lo que cubre todo el ciclo de vida [4].

D. Acceleo

Es una herramienta de transformación de modelo a texto, es una herramienta con un lenguaje basado en plantillas o templates, teniendo en cuenta los estándares definidos por EMF, Acceleo brinda un enfoque MDA simple permitiendo la generación de archivos a partir de modelos UML, MOF o EMF [5].

Acceleo cuenta con las siguientes características:

- Una completa integración con el ambiente de Eclipse y el framework de EMF [5].
- Navegación por los elementos de cualquier modelo que siga los estándares de EMF (XMI) [5].
- Administración de la sincronización entre el código y el modelo [5].
- Simplicidad en el mantenimiento y actualización de todas las plantillas [5].
- Coloreado sintáctico de las plantillas así como detección de errores basada en el meta-modelo [5].
- Pre visualización del código [5].

Acceleo permite definir plantillas donde se define el texto a generar de acuerdo a las propiedades encontradas dentro del modelo. Los Templates son almacenados en archivos con extensión ‘.mt’. La sintaxis ofrece un conjunto de instrucciones que permiten realizar ciclos, tomar decisiones y navegar por los elementos del modelo [5].

E. Atlas Language Transformation (ATL)

ATL (ATLAS Language Transformation) es un lenguaje de transformación de modelos, proporciona maneras de producir un conjunto de modelos de destino a partir de un conjunto de modelos de fuente, es un framework para administrar transformaciones basadas en modelos. Un modelo fuente se transforma en un modelo destino mediante una definición de transformación escrita en ATL, que también es un modelo. Los modelos fuente, destino y la definición de la transformación, responden a sus meta-modelos respectivos y, a su vez, todos los meta modelos se ajustan a MOF [10].

La transformación de ATL es unidireccional, opera sobre un modelo fuente de sólo lectura y produce un modelo destino de sólo escritura. Durante la ejecución de una transformación, el modelo fuente puede ser navegado pero no cambiado, en cambio el modelo destino no puede ser navegado [10].

ATL ha implementado un ambiente de desarrollado sobre la plataforma Eclipse, las herramientas disponibles para la transformación de modelos el núcleo de funcionalidades ATL, que incluye el motor de transformación y las facilidades de administración de modelos. [10].

La parte básica de ATL incluye todos los componentes requeridos para configurar y ejecutar transformaciones, en particular, el EMF (Eclipse Modelling framework) y MDR (Meta Data repository) que permiten, respectivamente, manejar modelos definidos de acuerdo a la semántica Ecore y MOF [10].

F. JaMoPP (Java Model Parser and Printer)

JaMoPP, Java Model Parser and Printer es una herramienta que intenta tratar al lenguaje Java como un nuevo lenguaje de modelado, de forma que permita ser integrado con otras tecnologías de modelado [36]. Para lograr esto JaMoPP presenta un meta-modelo del lenguaje Java en toda su extensión, dicho meta-modelo se encuentra definido en el lenguaje de meta-modelado Ecore. JaMoPP define una sintaxis conforme con la especificación de Java, mediante la cual se genera un Parser, el cual crea instancias del meta-modelo según el código fuente, y un Printer que transforma instancias del meta-modelo en código fuente Java. Con JaMoPP, herramientas de modelado basadas en Ecore pueden procesar archivos Java en la misma forma en que procesan otros modelos [36].

2.3 Tecnologías Java Enterprise Edition 6 (JEE6)

Java Enterprise Edition [28] (Java EE), fue desarrollado por Sun Microsystems y lanzado en 1999 con el nombre de J2EE. Proporciona un conjunto de especificaciones técnicas para el desarrollo de aplicaciones empresariales. Puede ser visto como una extensión de Java SE para facilitar el desarrollo de aplicaciones distribuidas, robustas, potentes y de alta disponibilidad.

La plataforma Java EE es desarrollada por la JCP (Java Community Process) que es responsable de todas las tecnologías Java. Estos grupos de expertos crean las JSR (Java Specification Requests), especificaciones para las diversas tecnologías y permiten una estandarización y compatibilidad entre plataformas.

Una de sus últimas ediciones corresponde a Java EE 6, que tiene como objetivo ofrecer a los desarrolladores un conjunto de APIs y tecnologías para reducir el tiempo de desarrollo, la complejidad y aumentar el rendimiento en las aplicaciones.

La principal novedad de esta versión es intentar simplificar el desarrollo de software ofreciendo varios componentes internos. Los desarrolladores se benefician de menos configuraciones XML y más anotaciones, más POJOS (Plain Old Java Objects), y un empaquetado simplificado.

Java EE 6 usa un modelo de capas distribuidas para el desarrollo de aplicaciones empresariales. Las aplicaciones se dividen en capas dependiendo su funcionalidad y cada capa está compuesta por componentes, donde un componente Java EE 6 es unidad de software propia y funcional que es ensamblada dentro de una aplicación relacionada con clases y ficheros (recursos externos) y comunicándose con otros componentes. La especificación Java EE 6 define los siguientes componentes:

- Componentes de la capa de cliente corriendo en la máquina cliente.
- Componentes de la capa web corriendo en el servidor Java EE 6.
- Componentes en la capa de negocio corriendo en el servidor Java EE 6.
- Software de la capa EIS (Enterprise Information System) corriendo en un servidor EIS, con bases de datos.

La figura 4 es una representación gráfica del modelo Java EE 6:

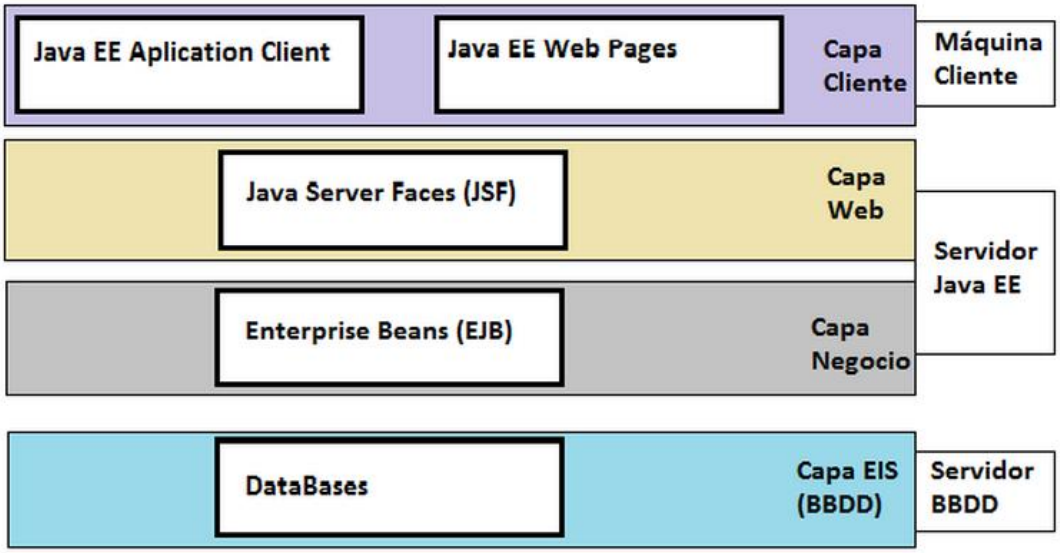

Figura 4: Modelo Java EE 6

A pesar de ser aplicaciones multicapas normalmente se consideran o se denominan aplicaciones de tres capas porque son distribuidas alrededor de tres localizaciones: máquinas clientes, Servidores Java, y bases de datos o maquinas antiguas (legacy machines).

Java EE 6 se divide en dominios lógicos llamados contenedores. Cada contenedor tiene una función específica, soporta un conjunto de APIs y ofrece servicios a los componentes tales como seguridad, acceso a la base de datos, gestión de transacciones, nombres de directorios, e inyección de recursos. Los contenedores ocultan la complejidad técnica y mejoran la portabilidad. El contenedor EJB es responsable de administrar la ejecución de los beans que contiene la lógica de negocio [27].

La inyección de dependencia puede usarse en todos los recursos que un componente necesite. Puede ser usada en contenedores EJB, contenedores web, y aplicaciones cliente. La inyección de dependencia permite a un contenedor Java EE 6 insertar automáticamente referencias a otros componentes o recursos usando anotaciones [29].

La figura 5 representa los contenedores de Java EE 6:

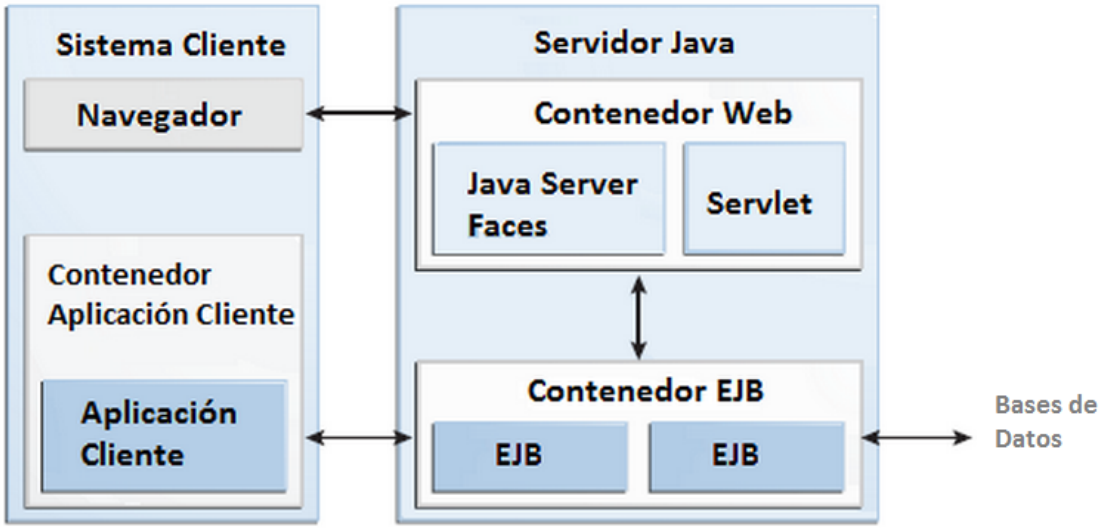

Figura 5: Contenedores Java EE 6

Los contenedores proporcionan servicios subyacentes a sus componentes desplegados, esto permite al desarrollador centrarse en la lógica de aplicación en lugar de resolver problemas técnicos. Algunos de los servicios que proporciona Java EE 6 se describen a continuación [11]:

- Java Transaction API (JTA): Este servicio ofrece una demarcación de transacciones API utilizada por el contenedor y la aplicación. También proporciona una interfaz entre el administrador de transacciones y el administrador de recursos en el nivel Service Provider Interface (SPI).
- Java Persistence API (JPA): API estándar para el mapeo de objeto-relacional (ORM). Con Java Persistence Query Language (JPQL), se puede consultar objetos almacenados en la base de datos subyacente.
- Validación: El Bean de validación proporciona un nivel de declaración de restricción de la clase y la facilidad de validación.
- Java Message Service (JMS): Permite que los componentes se comuniquen de forma asincrónica a través de mensajes.
- Java Naming and Directory Interface (JNDI): Esta API, incluida en Java SE, se utiliza para acceder a los sistemas de nombres y directorios. La aplicación se utiliza para asociar (enlazar) los nombres de los objetos y luego encontrar estos objetos (búsqueda) en un directorio. Puede buscar fuentes de datos, JMS, EJB y otros recursos.
- JavaMail: Muchas aplicaciones requieren la capacidad de enviar correos electrónicos que pueden ser implementadas a través del uso de la API JavaMail.

2.3.1 JSF (Java Server faces)

La tecnología JSF básicamente, establece un estándar en el desarrollo de la capa de presentación en las aplicaciones web Java EE 6, que consiste en el uso de componentes para formar esta presentación. Estos componentes son completamente Independientes y pueden proceder de diferentes librerías de terceros. Además, proporciona un motor de navegación de reglas así como la gestión de beans manejados y de respaldo [31].

JSF específica cómo debe ser la parte de la Vista de una aplicación web, haciendo que esté basada en componentes. JSF ha supuesto un gran avance en la manera de concebir la presentación de la información y la interacción con el usuario en la parte cliente [12].

Una página JSF utiliza la extensión *.xhtml, es decir, una combinación de XML con HTML y puede incluir componentes como CSS, JavaScript, entre otros [32].

La especificación de JSF define seis fases distintas en su ciclo de vida [31]:

1. Restauración de la vista: Crea un árbol de componentes en el servidor para representar la información de un cliente.

2. Aplicar valores de la petición: Actualiza los valores del servidor con datos del cliente.

3. Proceso de validación: Valida los datos del usuario y hace la conversión.

4. Actualización de valores del modelo: Actualiza el modelo del servidor con nuevos datos.

5. Invocar la aplicación: Ejecutar cualquier lógica de aplicación para cumplir con la solicitud.

6. Procesar la respuesta: Guarda un estado y da una respuesta al cliente.

2.3.2 JPA (Java Persistence API)

JPA [33] proporciona un modelo de persistencia para mapear bases de datos relacionales. En java EE 6, JPA 2.0 sigue el mismo camino de simplicidad y robustez y agrega nuevas funcionalidades. Se puede utilizar para acceder y manipular datos relacionales de Enterprise Java Beans (EJBs), componentes web y aplicaciones Java SE.

JPA es una abstracción que está por encima de JDBC lo que permite ser independiente de SQL. Todas las clases y anotaciones de esta API se encuentran en el paquete javax.persistence. Los principales componentes de JPA son [34]:

- Mapeo de base de datos relacionales (ORM). Es el mecanismo para mapear objetos a los datos almacenados en una base de datos relacional.
- Un API administrador de entidad para realizar operaciones en la base de datos tales como crear, leer, actualizar, eliminar (CRUD).
- El Java Persistence Query Language (JPQL) que permite recuperar datos con un lenguaje de consultas orientado a objetos.
- Las transacciones y mecanismos de bloqueo cuando se accedan a los datos concurrentemente, la API Java Transaction (JTA).

Una entidad es objeto de dominio de persistencia. Por lo general, una tabla en el modelo de datos relacional es representada por una entidad y sus instancias corresponden a los registros de dicha tabla. El estado de persistencia de una entidad es representado por propiedades persistentes, estas propiedades a su vez usan anotaciones para el mapeo de las entidades y relaciones entre entidades. Las relaciones entre entidades persistentes deben mapearse explícitamente como llaves foráneas o uniones de tablas, la manera en que se estructura una entidad, sus atributos y sus relaciones [32].

2.3.3 EJB (Enterprise Java Beans)

Los JavaBeans empresariales (EJB) son una tecnología (API) que forma parte del estándar de Java EE. Están diseñados para desarrollo y despliegue de aplicaciones (distribuidas) de negocio basadas en componentes del lado del servidor. Una vez que se desarrolla una aplicación, ésta puede ser desplegada en cualquier servidor que soporte la especificación de EJB. Con esta tecnología es posible desarrollar aplicaciones empresariales sin tener que crear de nuevo los servicios de transacción, seguridad, persistencia, concurrencia y lo que se pueda necesitar en el proceso de creación de una aplicación; permitiendo a los desarrolladores enfocarse en la implementación de la lógica de negocio [35].

EJB divide la capa de negocio en dos partes: Capa de lógica de negocio donde se encuentra EJB y capa de persistencia. EJB cuenta con dos componentes de proceso de negocio, los beans de sesión (Session Beans) y los beans dirigidos por mensajes (Message-Driven Beans, MDBs), ambos son desarrollados por una aplicación de negocio e implementados y ejecutados por el Contenedor de EJB [35].

2.3.4 PrimeFaces

PrimeFaces es una librería de componentes visuales de código abierto para el conjunto Java Server Faces 2.0 desarrollada y mantenida por Prime Technology. Su objetivo principal es

ofrecer un conjunto de componentes para facilitar la creación y diseño de aplicaciones web [30].

Los componentes de PrimeFaces cuentan con soporte nativo de Ajax, pero no se encuentra implícito, de tal manera que se tiene que especificar que componentes se deben actualizar al realizar una petición proporcionando así mayor control sobre los eventos. Cuenta también con un módulo adicional TouchFaces para el desarrollo de aplicaciones web para dispositivos móviles con navegadores basados en WebKit [30]. Las principales características de PrimeFaces son:

- Soporte nativo de Ajax, incluyendo Push/Coment.
- Kit para crear aplicaciones web móviles.
- Es compatible con otras librerías de componentes como Jboss RichFaces.
- Uso de JavaScript no intrusivo.
- Es un proyecto open source, activo y estable.

2.3.5 Ambientes IDE

A. Eclipse

Eclipse es un programa informático compuesto por un conjunto de herramientas de programación de código abierto multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente liviano" basadas en navegadores. Esta plataforma, típicamente ha sido usada para desarrollar entornos de desarrollo integrados (del inglés IDE), como el IDE de Java llamado *Java Development Toolkit* (JDT) y el compilador (ECJ) que se entrega como parte de Eclipse (y que son usados también para desarrollar el mismo Eclipse) [37].

B. NetBeans IDE

Netbeans es un entorno de desarrollo, una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java - pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el NetBeans IDE. NetBeans IDE es un producto libre y gratuito sin restricciones de uso [38].

También está disponible NetBeans Platform; una base modular y extensible usada como estructura de integración para crear grandes aplicaciones de escritorio. Empresas independientes asociadas, especializadas en desarrollo de software, proporcionan extensiones adicionales que se

integran fácilmente en la plataforma y que pueden también utilizarse para desarrollar sus propias herramientas y soluciones [38].

IV - DESARROLLO DEL TRABAJO

Para el desarrollo del presente trabajo de grado se estableció previamente la metodología y las actividades a realizar. (Ver sección 2.4 Metodología que se propuso para satisfacer cada objetivo).

Con la ayuda de la metodología ágil SCRUM se establecen iteraciones para cada fase del trabajo de grado, de tal forma que se pueda administrar, controlar y desarrollar el proceso de construcción de software. El desarrollo se realiza en forma iterativa e incremental. Cada ciclo o iteración termina con una pieza de software ejecutable que incorpora nueva funcionalidad al proyecto [49].

1. Proceso de desarrollo

La herramienta Wapp Generator está compuesta de dos proyectos que son *javeriana.edu.co.emf.webgenerator.model*, que sirve para modelar la aplicación web, utilizando el meta-modelo definido. Por otro lado el proyecto *javeriana.edu.co.acceleo.webgenerator.m2t*, que permite transformar el modelo a la aplicación web en tecnología Java EE 6.

1.1 Requerimientos

Los requerimientos detallados del ambiente Wapp Generator, describen la funcionalidad tanto del meta-modelo como del transformador de código. (Ver Anexo Especificación de requerimientos)

1.1.1 Descripción Global del ambiente

El sistema se encuentra dentro del marco MDE, para la generación de código dirigido por modelos, dando como resultado una herramienta para el desarrollo de aplicaciones WEB, en una tecnología especifica como Java EE 6.

El desarrollo del sistema se realiza por medio de diferentes herramientas de modelado y transformación soportadas por MDA, donde se especifica los requerimientos funcionales que debe tener la herramienta Wapp Generator para su construcción. (Ver sección 1.2.3 Tecnologías de Soporte y herramientas para MDD)

1.1.2 Funciones del ambiente

La siguiente lista muestra las meta-clases del ambiente:

- Meta-clase página web.
- Meta-clase Formulario.
- Meta-clase Transición.
- Meta-clase Tabla.
- Meta-clase Entidades.
- Meta-clase Roles de seguridad.

Cada conjunto de meta-clases especifica los requerimientos para cada funcionalidad del sistema, estableciendo su respectiva verificación, prioridad, Descripción y requerimientos asociados. (Ver Anexo Especificación de Requerimientos)

1.1.3 Priorización de los requerimientos

La priorización de los requerimientos se realizó según la relevancia en tiempo en el cronograma para cada elemento, estimando el valor agregado en cada fase del sprint establecido. (Ver sección 1.2 Definición de meta-modelo y Anexo Cronograma TG)

En la tabla 1 se muestra los requerimientos del ambiente Wapp Generator:

Código Requerimiento	Descripción	Sprint	Prioridad
REQ 001	El sistema debe permitir modelar una aplicación web.	Sprint 1	Alta
REQ 002	El sistema debe permitir modelar una página web que es parte del modelo de una aplicación web.	Sprint 1	Alta
REQ 003	El sistema debe permitir definir y modificar en el modelo de una página web, las siguientes propiedades: título, nombre de la página, dimensiones de la página y hoja de estilos css.	Sprint 1	Media
REQ 004	El sistema debe permitir al usuario indicar si el modelo	Sprint 1	Baja

	de la página web que hace parte de una aplicación web, posee banner.		
REQ 005	El sistema debe permitir modelar un elemento formulario dentro de una página que es parte del modelo de una aplicación web.	Sprint 2	Alta
REQ 006	El sistema debe permitir agregar un campo dentro de un formulario que es parte del modelo de una página, los campos que se podrán definir son campos de texto, campos checkbox, campos de radio, campos de fecha y áreas de texto.	Sprint 2	Media
REQ 007	El sistema debe permitir agregar al formulario varios botones, que hace parte del modelo de la página web.	Sprint 2	Media
REQ 008	El sistema debe permitir asociar en el modelo de una página web, a un botón un componente de negocio que invoca una acción o a otro modelo de página web.	Sprint 2	Media
REQ 009	El sistema debe permitir al usuario modelar el flujo de navegación de la aplicación web mediante la indicación de la siguiente página asociada a cada botón,	Sprint 3	Alta

	asimismo el usuario puede definir si la transición posee condición.		
REQ 010	El sistema debe poder modelar un elemento tabla como parte del modelo de una página web.	Sprint 4	Alta
REQ 011	El sistema debe permitir agregar campos a una tabla en el modelo de una página web.	Sprint 4	Media
REQ 012	El sistema debe permitir al usuario la opción de transformar del modelo de la aplicación web al código para tecnología Java EE 6.	A lo largo del desarrollo del ambiente Wapp Generator	Alta
REQ 013	El sistema debe permitir a un usuario modelar una entidad indicando atributos propios y atributos de relación con otras entidades.	Sprint 5	Alta
REQ 014	El sistema debe permitir definir propiedades de un atributo de un modelo de una entidad. Las propiedades del atributo que se podrán definir son: tipo, nombre e indicaciones de persistencia.	Sprint 5	Media
REQ 015	El sistema debe generar las páginas y clases necesarias para soportar el CRUD a partir del modelo de una entidad, en tecnología Java	Sprint 5	Media

	EE 6.		
REQ 016	El sistema debe permitir al usuario agregar propiedades de seguridad al modelo de cada página web, definiendo que tipo de roles pueden acceder a la página.	Sprint 6	Media

Tabla 1: Especificación de requerimientos.

1.1.4 Restricciones de la herramienta Wapp Generator

Las siguientes restricciones corresponden al desarrollo de la herramienta:

- En la generación se cargara un estilo por defecto a las páginas web, excepto si el usuario carga otro estilo.
- La herramienta de modelaje se presentara en forma de diseño jerárquico de un modelo para la creación de cada componente de la aplicación web, no gráficamente.
- Se podrá modelar cuales roles de seguridad están permitidos para una página y para un caso de uso sin llegar al detalle de cuales roles pueden ver cada elemento de una página o invocar una acción de un caso de uso.

1.2 Metodología de desarrollo

Para el desarrollo del ambiente Wapp Generator se utilizó la metodología ágil SCRUM para cumplir de una manera efectiva, rápida y con calidad el desarrollo de Wapp Generator. Adoptando SCRUM se propuso 6 sprints, donde en cada uno se agregaba una nueva funcionalidad que debía tener la herramienta, cumpliendo en cada uno de estos srpint, un grupo de requerimientos relacionados con la nueva funcionalidad que debería cumplir Wapp Generator.

La duración de cada sprint fue de una semana, exceptuando los dos últimos sprint, donde debido a la complejidad de estos dos últimos era necesario asignar un mayor tiempo para cumplir con los requerimientos de estos sprints. Dentro de cada sprint se agregaban nuevos elementos al meta-modelo y se extendía el transformador de Wapp Generator, para que tuviera en cuenta dentro de su transformación los nuevos elementos agregados. Asimismo se probaba las aplicaciones generadas cada vez que se agregaba un nuevo elemento dentro del meta-modelo, utilizando como servidor web GlassFish 4.0. Por ultimo al final de cada sprint, se mostraba el

ambiente Wapp Generator a la directora del trabajo de grado María Consuelo Franky, donde ella daba su retroalimentación sobre el desarrollo de Wapp Generator.

1.3 Definición de meta-modelo

Para el desarrollo de la generación de la aplicación web es necesario definir un modelo independiente de tecnología (DSL). Para el diseño apropiado de éste fue necesario identificar los elementos transversales en una aplicación web, una vez identificados estos elementos se establecieron en el meta-modelo como meta-clases. Posteriormente es necesario establecer la asociación que existen entre las meta-clases. El meta-modelo resultante es observado en la figura 6, que se muestra a continuación:

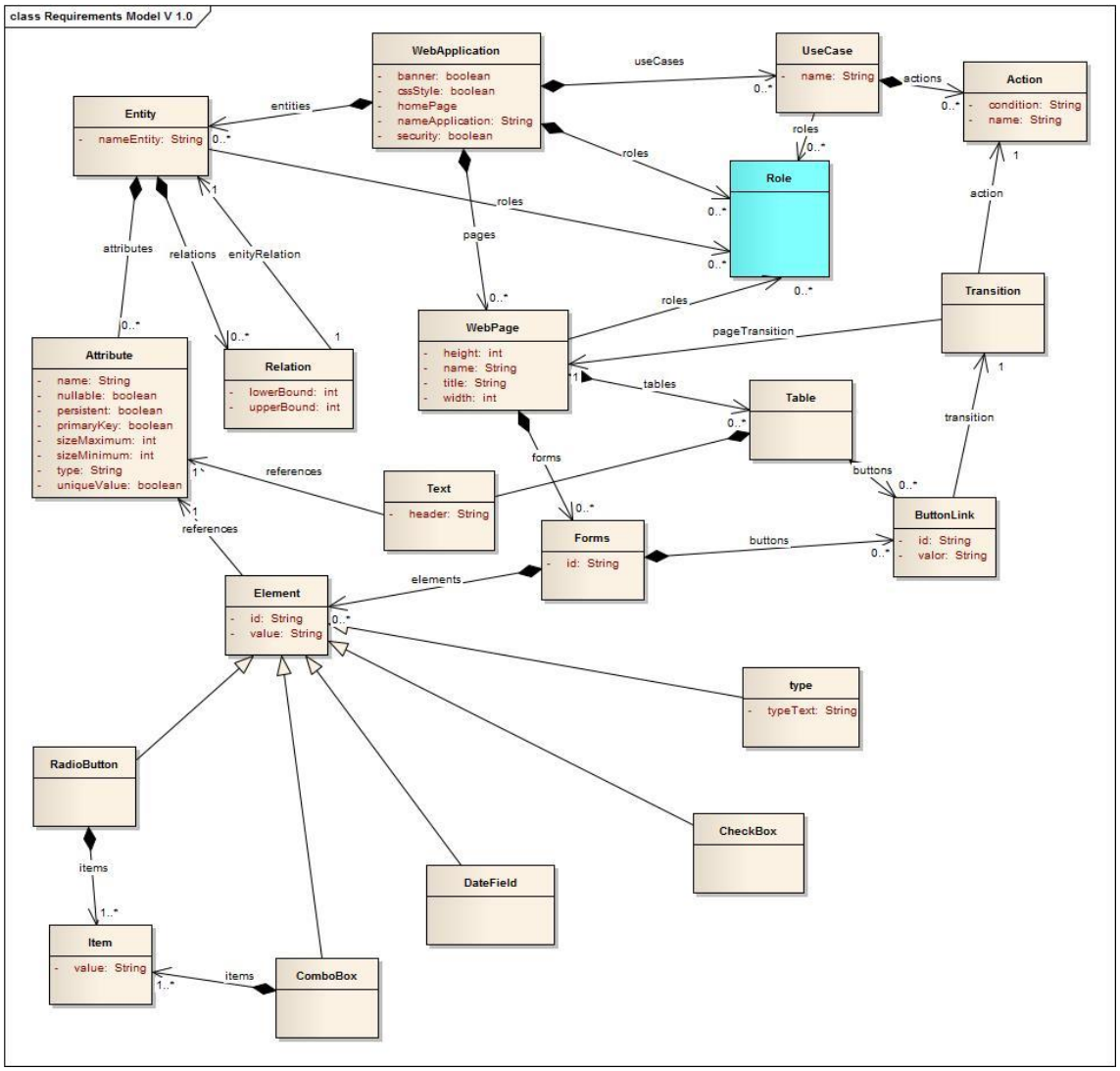

Figura 6: Diseño del Meta-modelo.

Asimismo es posible observar una descripción detallado de la meta-clase en la tabla 2:

Meta-Clase	Definición	Atributos
WebApplication	Es el punto de partida para modelar la aplicación web. En esta meta-clase se definen las características básicas de la aplicación web.	• NameAplication: Nombre de la aplicación. • Banner: condicional para definir si la página tiene banner. • Security: condicional para determinar si la aplicación tendrá seguridad. • cssStyle: define si la página usara un css.
UseCase	Representa un caso de uso dentro de la aplicación, en la cual la meta-clase WebApplication tiene un conjunto de metaclases UseCases. Estos casos de usos además contienen un conjunto de acciones, que son soportados por este.	• Name: nombre del caso de uso.
Action	Meta-clase que representa una acción dentro del meta-modelo. Esta meta-clase es usada por la meta-clase UseCase.	• Name: nombre de la acción. • Condition: se define un nombre de la condición si la acción tiene una condición.
WebPage	Representa dentro del meta-modelo una página web. La meta-clase WebApplicationWeb tiene un conjunto de esta meta-clase.	• Name: nombre del archivo de la página. • Title: título de la página web. • Width: ancho de la página web. • Height: altura de la

		página web.
Table	Esta clase es utilizada para modelar una tabla dentro de una página web. Siguiendo este orden de ideas la meta-clase WebPage tiene una colección de la meta-clase Table. Asimismo la meta-clase está relacionada con la meta-clase Entity, ya que este elemento es el encargado de mostrar el valor de los atributos de todas las entidades.	
Text	Meta-clase utilizado para determinar que campos de la entidad serán mostrados en la tabla, esta meta-clase está relacionada con los atributos de la entidad.	• Header: Titulo que tendrá cada columna.
Forms	Esta meta-clase es utilizada cuando se desea modelar un formulario, asimismo la meta-clase WebPage tiene un conjunto de Forms.	• Id: identificación único del formulario.
Element	Meta-clase que sirve para modelar un elemento dentro de un formulario. Asimismo como esta meta-clase sirve como padre para las metaclases: • RadioButton. • ComboBox. • TextField. • DateField.	• Id: identificación único del elemento. • Value: label relacionado a los campos de entrada

	• CheckBox. Cada Element está relacionado con un atributo de la entidad.	
ButtonLink	ButtonLink permite modelar un botón dentro del meta-modelo. ButtonLink asimismo permite definir una transición de una página a otra.	• Id: identificación único del boton. • Value: texto que se mostrara en el botón.
Transition	Permite modelar una transición de una página a otra. Está relacionada con la meta-clase ButtonLink, WebPage y Action.	
Action	Permite modelar una acción que es llevada a cabo por un botón en medio de una transición entre páginas. Esta meta-clase se encuentra relacionada con la meta-clase UseCase, debido a que este último tiene definido un conjunto de acciones.	• Name: Nombre de la aplicación. • Condition: nombre de una condición si la acción posee alguna condición.
Role	Permite modelar un rol dentro de la aplicación web.	• Name: nombre del rol.

Tabla 2: Descripción elementos meta-modelo.

1.4 Desarrollo del transformador para Java EE 6 con plantillas Acceleo

En esta sección se contempla la fase del transformador, con su respectivo desarrollo.

1.4.1 Diagrama de invocación (Gestión de plantillas)

En la figura 7 se muestra en el diagrama de invocación de plantillas iniciando con la plantilla *generate.mtl*, que actúa como *main* de la herramienta Wapp Generator:

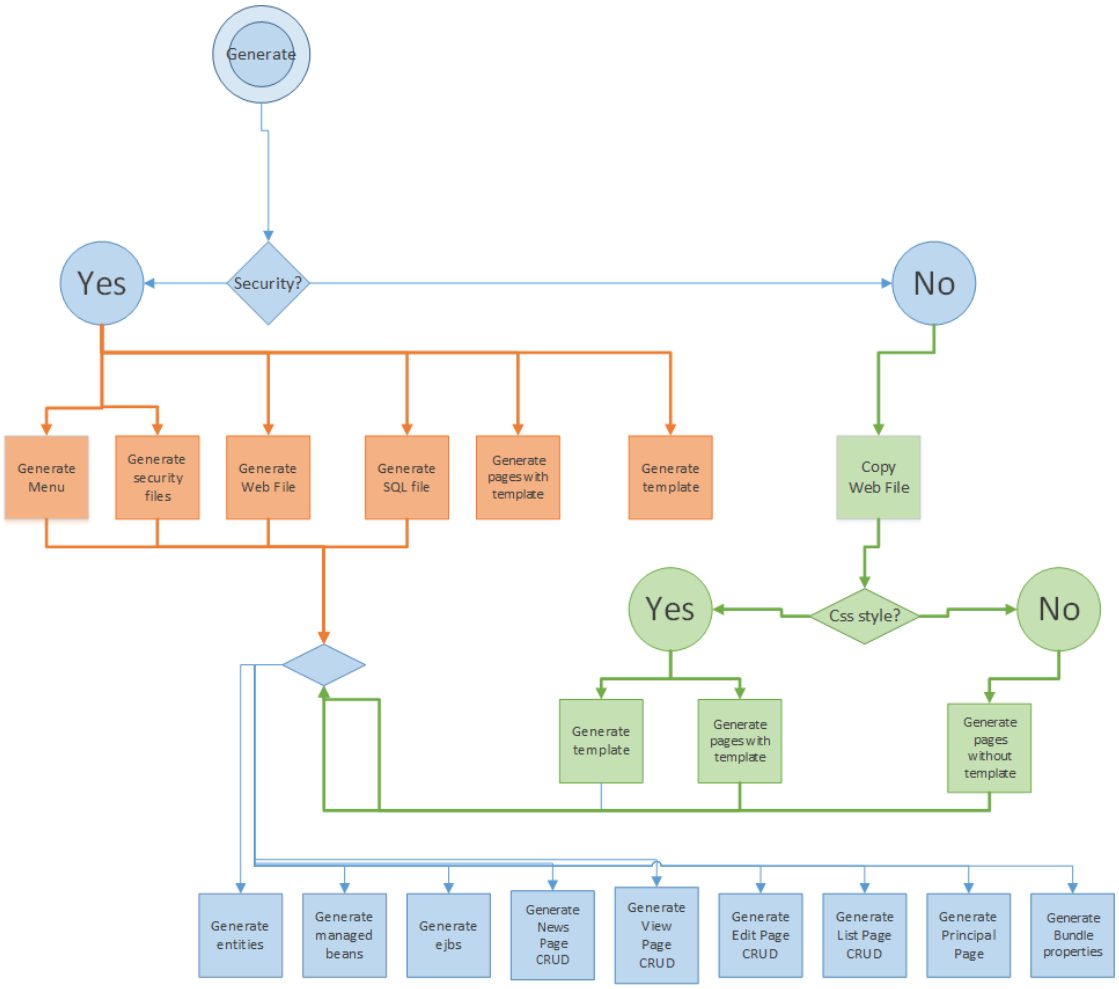

Figura 7: Diagrama de invocación de plantillas.

Como se observa en la anterior imagen, el flujo inicia con el modulo *generate.mtl*, el cual comienza consultando el atributo *security* del modelo de la aplicación, si este atributo se encuentra en verdadero se invocaran las siguientes plantillas que serán necesarias para la creación de una aplicación web con seguridad en Java EE 6:

Plantilla	Descripción
Generate Menu	Página web que contiene los enlaces a las distintas páginas teniendo en cuenta las restricciones de acceso de cada página.
Generate Security Files	Página web para el ingreso a la aplicación y página de error, entidades *Role* y *User*, y por

	ultimo managed bean *Auth.java*.
Generate web file	Generación del archivo *web.xml*, teniendo en cuenta las restricciones de seguridad modeladas.
Generate SQL file	Generación de script para la inserción de los roles en la tabla *Role*.
Generate page with template:	Generación de las páginas web modeladas con un template.
Generate template	Generación del *Facelet* que será utilizada por las páginas web.

Tabla 3: Descripción diagrama de invocación con seguridad.

Si por el contrario, la aplicación web modelada no cuenta con módulo de seguridad, el archivo *web.xml* no es generado si no es copiado al proyecto generado. Enseguida el módulo *generate.mtl* verifica si el valor del atributo *cssStyle* del modelo de la aplicación web es verdadero, para generar las paginas utilizando un template y asimismo generar el facelet que utilizaran estas páginas web. Si por el contrario el atributo *cssStyle* es falso se generan páginas que no utilizan un template.

Una vez verificados estos atributos del modelo, el archivo *generate.mtl*, prosigue a invocar las siguientes plantillas:

Plantillas	Descripción
Generate entities	Se encarga de generar los archivos *.java* asociados a la entidad modelada.
Generate managedBeans	Encargada de generar los managed bean para soportar los casos de uso.
Generate ejbs	Encargada de generar los ejbs asociados a los casos de uso, cabe aclarar que la generación de estos métodos no es completa y por lo tanto deben ser implementados por los desarrolladores.
Generate New Page CRUD	Genera las páginas web para la creación de las

	entidades modeladas.
Generate View Page CRUD	Genera las páginas web para ver los detalles de las entidades modeladas.
Generate Edit Page CRUD	Genera las páginas que permite la edición de las entidades modeladas.
Generate List Page CRUD	Genera la lista que permite ver las colecciones instanciadas de las entidades modeladas.

Tabla 4: Descripción diagrama de invocación plantillas.

1.4.2 Plantilla Principal

El modulo que se encarga en llamar las diferentes plantillas es el archivo *generate.mtl*. En la figura 8 se muestra este archivo:

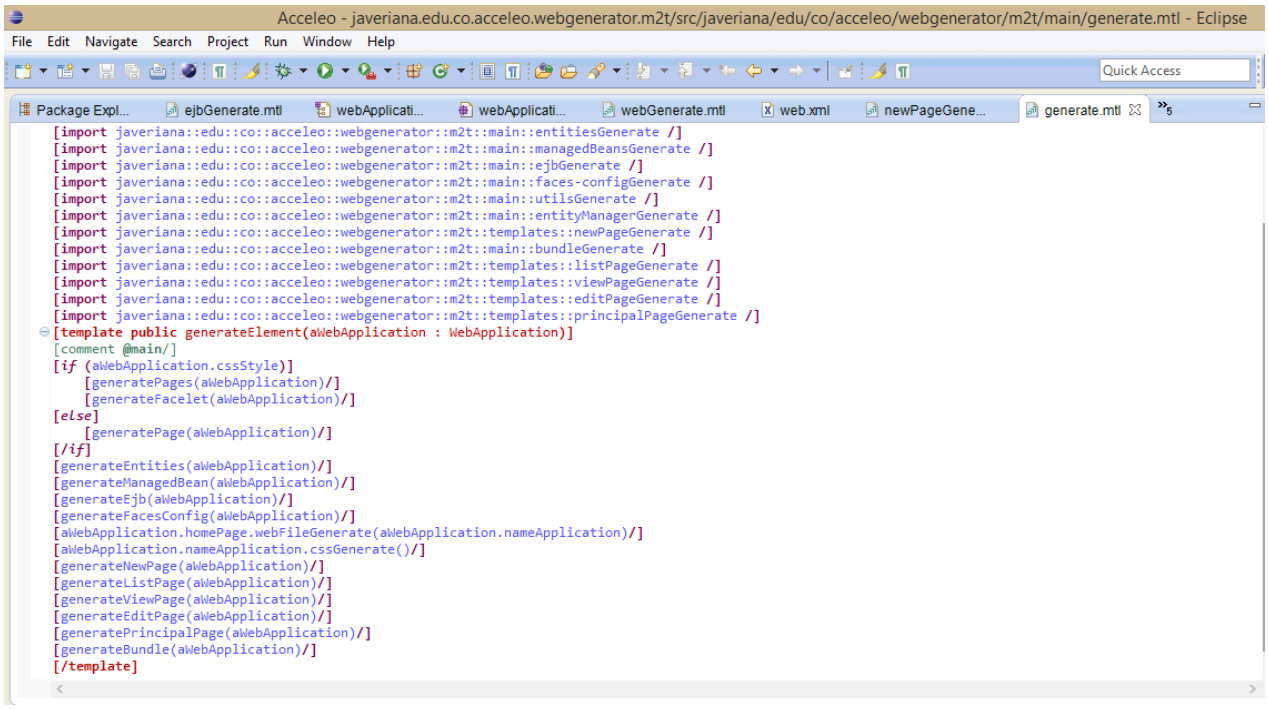

Figura 8: Archivo generate.mtl.

Como se puede observar en la anterior imagen, el archivo *generate.mtl*, se encarga de utilizar las otros módulos para generar la aplicación web. Las primeras instrucciones que se ven en la figura anterior, son utilizadas para importar los módulos que serán utilizados, enseguida se observa la etiqueta [template] es el método del módulo *generate.mtl* donde se coloca que módulos serán llamados y toma como entrada una instancia de la meta-clase WebApplication,

asimismo con ayuda de la etiqueta [comment @main] permite ejecutar este módulo como archivo principal.

Para importar un archivo es utilizada la etiqueta *[import]* donde se especifica la plantilla que se desea utilizar. En la figura 9 se muestra el ejemplo de importar la plantilla *entitiesGenerate* para ser utilizada en *generate.mtl:*

```
[import javeriana::edu::co::acceleo::webgenerator::m2t::main::pageGenerate /]
[import javeriana::edu::co::acceleo::webgenerator::m2t::main::pageGenerate2 /]
[import javeriana::edu::co::acceleo::webgenerator::m2t::main::FaceletTemplateGenerate /]
[import javeriana::edu::co::acceleo::webgenerator::m2t::main::entitiesGenerate /]
[import javeriana::edu::co::acceleo::webgenerator::m2t::main::managedBeansGenerate /]
[import javeriana::edu::co::acceleo::webgenerator::m2t::main::ejbGenerate /]
[import javeriana::edu::co::acceleo::webgenerator::m2t::main::faces-configGenerate /]
[import javeriana::edu::co::acceleo::webgenerator::m2t::main::utilsGenerate /]
```

Figura 9: Importar una plantilla.

Una vez importada la plantilla se llama al método definida esta para la generación del archivo, siguiendo el ejemplo se generan las entidades en tecnología Java EE 6. En la figura 10 se muestra la invocación del método:

```
[generateEntities(aWebApplication)/]
[generateManagedBean(aWebApplication)/]
[generateEjb(aWebApplication)/]
[generateFacesConfig(aWebApplication)/]
[aWebApplication.nameApplication.cssGenerate()/]
[generateNewPage(aWebApplication)/]
[generateListPage(aWebApplication)/]
[generateViewPage(aWebApplication)/]
[generateEditPage(aWebApplication)/]
[generatePrincipalPage(aWebApplication)/]
[generateBundle(aWebApplication)/]
```

Figura 10: Invocación método generación de las entidades.

Este método recibe como parámetro el modelo de una aplicación web en particular para generar todas las entidades modeladas dentro de esta entidad.

1.4.3 Elementos plantilla típica

Para entrenar en un análisis de como la plantilla toma el modelo de una aplicación web y transforma este en una aplicación en Java EE 6. Tomaremos como ejemplo el modulo *pageGenerate.mtl*. En la figura 11 mostraremos un parte de esta plantilla.

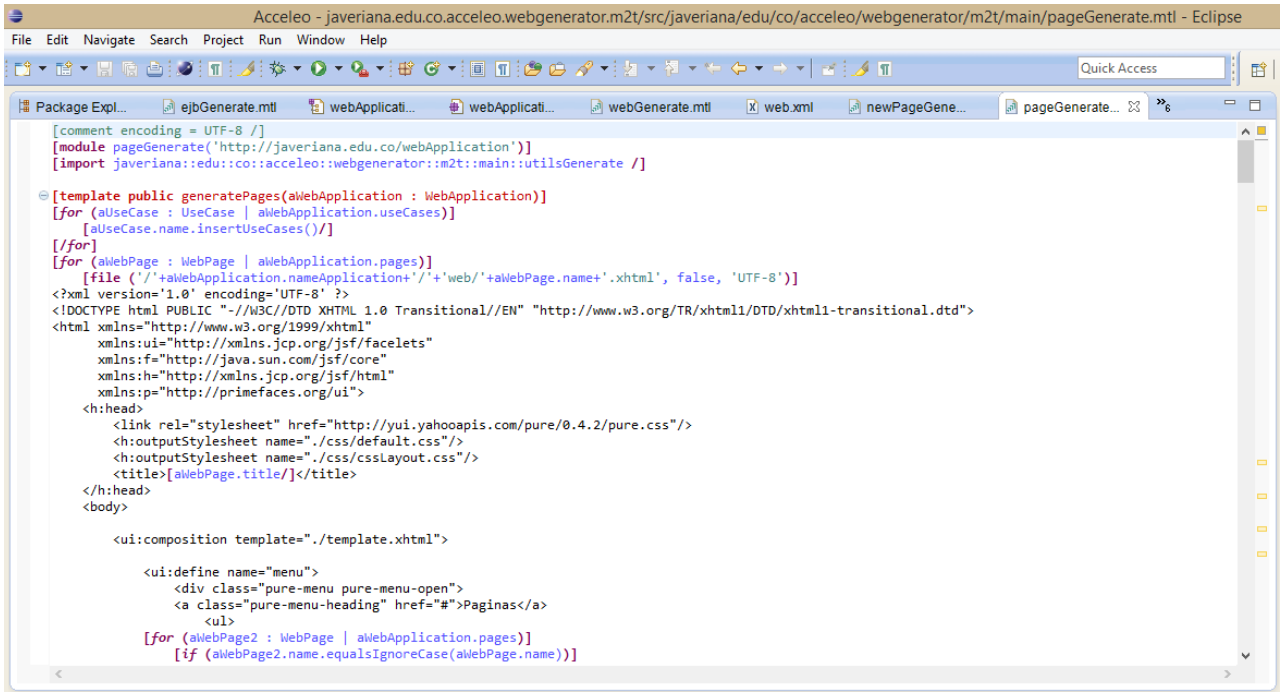

Figura 11: Archivo pageGenerate.mtl.

Como se puede observar la etiqueta *[template]* es el punto de partida de nuestro modulo, tomando como nombre *generatePages* y como entrada una instancia de la meta-clase *WebApplication*. La función de este módulo es transformar el modelo en archivos con extensión *xhtml*, por lo que para poder generar cada página web modelada es necesario recorrer todas las páginas instanciadas de la aplicación web, para poder lograr esto Acceleo cuenta con la instrucción *[for]* que sirve para poder recorrer los elementos de una colección, en este caso para recorrer las páginas de la aplicación web, se accede a esta por medio de la siguiente instrucción *aWebApplication.pages*, la que es representada en el meta-modelo como una relación entre la aplicación web y las páginas. Durante cada iteración debe ser creada el archivo necesario, para llevar esto a cabo Acceleo proporciona la instrucción *[file]*, esta toma como parámetro la ubicación y nombre del archivo. Dentro de la instrucción *[file]* escribimos la plantilla del archivo que deseamos generar, para este caso es escrita la plantilla para la generación de una página web en Java EE 6.

Dos de las etiquetas con la que es necesario empezar una plantilla en Acceleo serán mostradas utilizando como ejemplo la plantilla *entitiesGenerate.mtl*.

```
[comment encoding = UTF-8 /]
[module entitiesGenerate('http://javeriana.edu.co/webApplication')/]
[import javeriana::edu::co::acceleo::webgenerator::m2t::main::utilsGenerate /]

[template public generateEntities(aWebApplication : WebApplication)]
```

Figura 12: Fragmento plantilla entitiesGenerate.

Con la etiqueta *[module]*, se definen el nombre de la plantilla, así como el meta-modelo con el cual trabajara la plantilla por medio de la instrucción *'http://javeriana.edu.co/webApplication'*

Por otro lado la etiqueta *[template]* define el nombre del método de la plantilla *entitiesGenerate.mtl*, estableciendo el nombre del método de la plantilla y el parámetro para llevar a cabo el método, en este caso el parámetro es una instancia de la meta-clase *webApplication*.

Acceleo posee una implementación del lenguaje OCL para utilizar las diferentes instrucciones de control para poder transformar el modelo de una aplicación web a código en tecnología Java EE 6, este cuenta con las etiquetas *[for]*, utilizada al igual que lo hace un ciclo para iterar sobre una colección, la etiqueta *[if]*, representa una instrucción típica de *if*, en donde se eligen entre dos caminos posibles y por último la etiqueta *[else]*, utilizada para definir el camino alternativo a una etiqueta *[if]*.

Comenzaremos con la etiqueta *[if]* analizándola mediante la plantilla de Acceleo *entitesGenerate.mtl*. En la siguiente imagen veremos cómo se utiliza esta etiqueta dentro de la plantilla:

```
        [if (aAttributte.type.toString().equalsIgnoreCase('String'))]
            [if (aAttributte.sizeMinimum > 0 and aAttributte.sizeMaximum > 0 )]
@Size(min=[aAttributte.sizeMinimum/], max=[aAttributte.sizeMaximum/])
            [elseif (aAttributte.sizeMinimum > 0)]
@Size(min=[aAttributte.sizeMinimum/])
            [elseif (aAttributte.sizeMaximum > 0)]
@Size(max=[aAttributte.sizeMaximum/])
            [/if]
        [/if]
```

Figura 13: Ejemplo etiqueta [if].

En el anterior fragmento anterior se utiliza la etiqueta para evaluar si las restricciones de tamaño de un atributo de una entidad son mayor a 0. Por otro lado si no se cumple esta condición por medio de la etiqueta *[elseif]*, se evalúa si se cumple otra condición, en este caso se evalúa si la restricción mínima de tamaño es mayor a 0. Esta etiqueta es finalizada con la etiqueta *[/if]*.

Ahora entraremos a analizar la etiqueta *[for]* dentro de una plantilla de Acceleo, para recorrer las colecciones que posee el modelo. A continuación se muestra una figura ilustrando el uso de la etiqueta *[for]*:

```
[for (aEntity : Entity | aWebApplication.entities)]
    [file ('/'+aWebApplication.nameApplication+'/src/jpa/entities/'+aEntity.nameEntity+'.java', false)]

package jpa.entities;
import java.io.Serializable;
import java.util.List;
import java.util.Date;
import javax.persistence.Basic;
import javax.persistence.CascadeType;
```

Figura 14: Etiqueta for dentro de la plantilla.

En la anterior figura la etiqueta *[for]* es utilizada para recorrer todas las entidades modeladas dentro de la aplicación web. Esta etiqueta está compuesta por el tipo de iterador y por otro lado la colección que será recorrida, para este caso el tipo del iterador es *Entity* y la colección son las entidades modeladas, accediendo por medio de la instrucción *aWebApplication.entities*. La etiqueta *[for]* es delimitada por la etiqueta *[/for]*.

1.4.4 Clase Java Utils: construye estructuras y colecciones de elementos reconocidos en el modelo Invocación de métodos

Debido a las limitaciones del lenguaje OCL utilizado en Acceleo, para algunos puntos en la transformación del modelo de la aplicación web a JavaEE, fue necesario utilizar una Clase Java que permitiera poder utilizar las prestaciones de un lenguaje de programación completa, como era el caso de estructuras de datos.

Para poder acabo la integración, primero se define la clase en Java que se llevara a cabo los algoritmos complejos. Para la herramienta Wapp Generator se utilizó la clase *Utils.java*, ubicada en el paquete *javeriana.edu.co.acceleo.util*, como se ve en la figura 15:

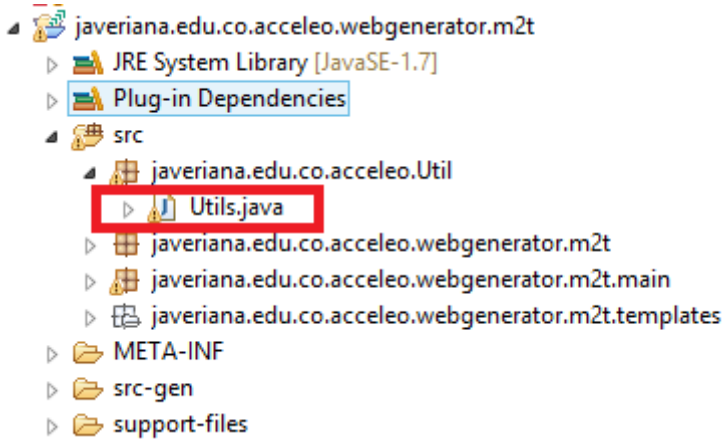

Figura 15: Ubicación clase Utils.java.

En la figura 16 se muestra el cuerpo esta clase Java.

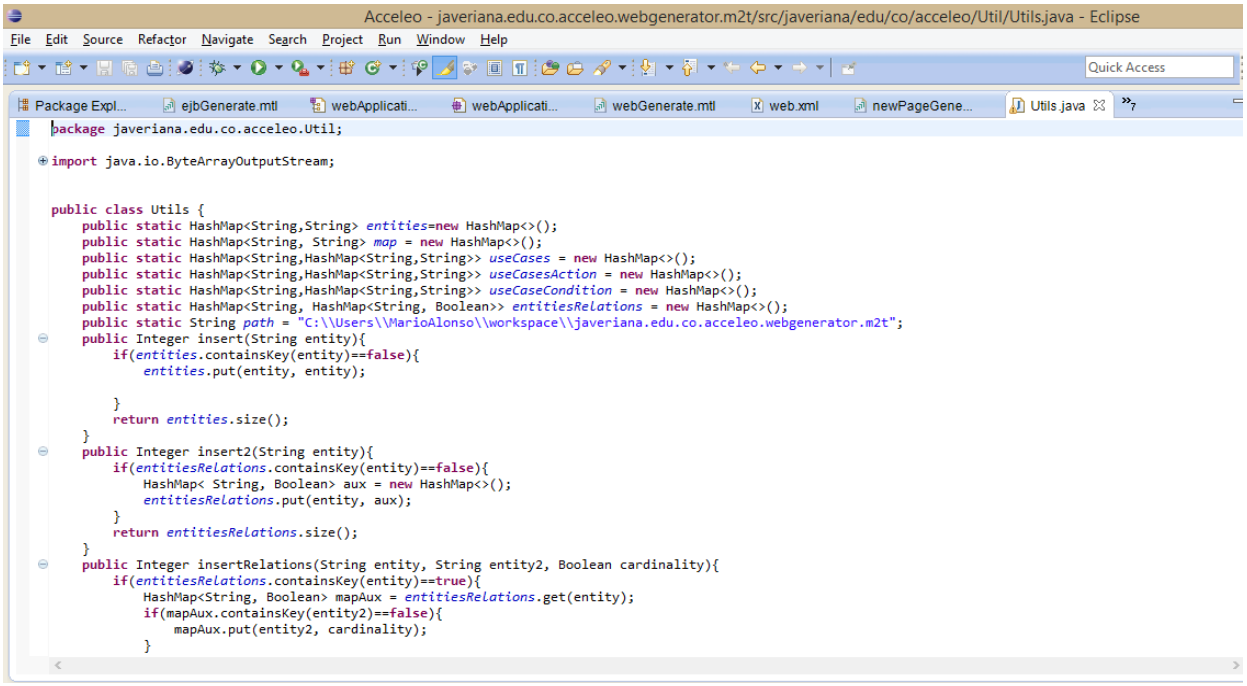

Figura 16: Clase Utils.java.

Como se observa en la imagen la clase *Utils* contiene varias estructuras de datos que son utilizados en los casos en que el lenguaje OCL no permite expresar de forma correcta la forma en que se deben generar los elementos del archivo. Por ejemplo en la figura 17 se muestra un fragmento de código de la clase *Utils*:

```
public String generateRelations(String entity){
    String relations = "";
    if(entitiesRelations.containsKey(entity)==true){
        HashMap<String, Boolean> mapAux = entitiesRelations.get(entity);
        for(Entry<String, Boolean> entry : mapAux.entrySet()){
            String aux = Character.toLowerCase(entry.getKey().charAt(0)) + (entry.getKey().length() > 1 ? entry.getKey().substring(1) : "");
            String aux2 = Character.toUpperCase(entry.getKey().charAt(0)) + (entry.getKey().length() > 1 ? entry.getKey().substring(1) : "");
            String aux3 = Character.toLowerCase(entity.charAt(0)) + (entity.length() > 1 ? entity.substring(1) : "");
            if(entry.getValue()){
                relations += "  @ManyToOne\n";
            }
            else{
                relations +="   @OneToOne(mappedBy = \""+aux3+"\")\n";
            }
            relations += "  private "+aux2+" "+aux+";\n";
        }
    }
    return relations;
}
```

Figura 17: Fragmento clase Utils.java.

Dentro de este método se utilización una *HashMap* para generar las relaciones entre entidades, ya sea *@ManyToOne* o *@OneToOne*. Finalmente este método retorna un *String* que contiene las relaciones que deben ser generadas.

1.4.5 Invocación de métodos de la clase Java Utils desde las plantillas

Para poder utilizar los algoritmos y estructuras definidas en la clase Utils de Java es necesario crear un módulo en Acceleo que permita una integración con esta clase. En la figura 18 se muestra el archivo *mtl* que permite esta integración.

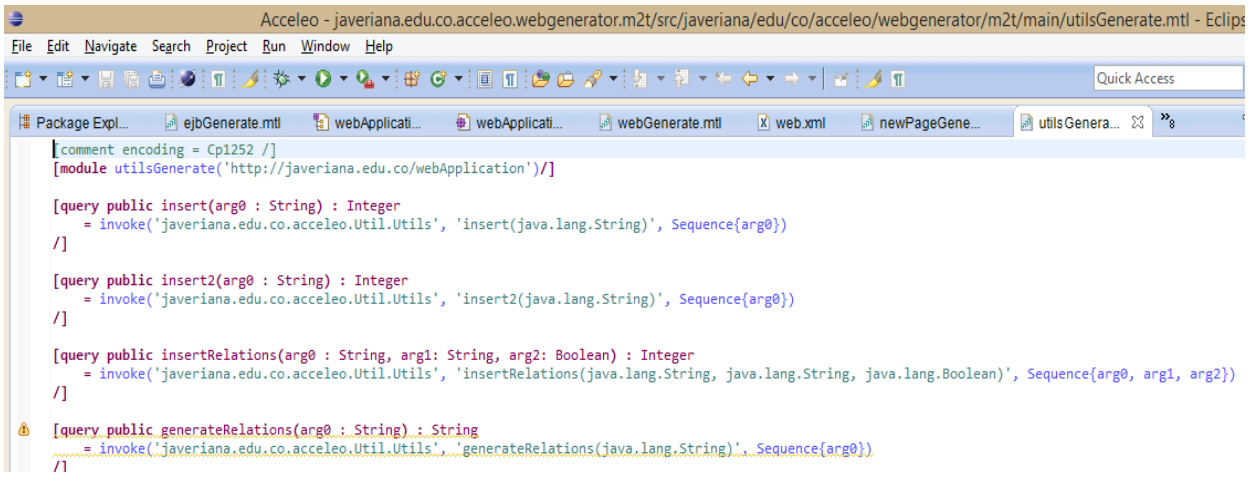

Figura 18: Archivo utilsGenerate.mtl.

Por medio de la instrucción *[query]* se definen la integración entre los métodos en java y OCL, definiendo el nombre del método, los parámetros y la referencia al método que se encuentra en la clase Java.

Una vez se definen los métodos integrados a Acceleo dentro del archivo *mtl*, es importado donde se desea ser utilizado y se procede a utilizar este método donde sea necesario. En la figura 19 se muestra un módulo que utiliza uno de los métodos.

```
    private [aRelation.entityRelation.nameEntity.toUpperFirst()/] [aRelation.entityRelation.nameEntity.toLowerFirst()/];
            [/if]
        [/for]
[aEntity.nameEntity.generateRelations()/]

    public [aEntity.nameEntity.toUpperFirst()/](){
    }

        [for (aAttribute : Attribute | aEntity.attributes)]
    public [aAttribute.type/] get[aAttribute.name.toUpperFirst()/](){
        return [aAttribute.name.toLowerFirst()/];
    }

    public void set[aAttribute.name.toUpperFirst()/]([aAttribute.type/] [aAttribute.name.toLowerFirst()/]){
        this.[aAttribute.name.toLowerFirst()/] = [aAttribute.name.toLowerFirst()/];
    }
```

Figura 19: Invocación método definido en Utils.java.

En este caso el servicio Java es utilizado para generar las clases java que deben ser importados por el *ManagedBean*, utilizando la instrucción *[aWeb.Page.useCase.name.generateImports()/]* se llama al método que se encuentra en la clase Utils, utilizando como parámetro el nombre del caso de uso, para finalmente generar las importaciones necesarias.

1.4.6 Aprovechamiento de archivos de texto prexistentes

Debido a que algunos archivos no cambian y son estándar para todas las aplicaciones web en Java EE 6, no era necesaria su generación, por lo que se definieron como archivos prexistentes o de soporte y son incluidos dentro del proyecto de transformación. Los archivos que fueron identificados como prexistentes son:

- Auth.java: *ManagedBean* utilizado cuando la aplicación web tienen seguridad.
- cssLayout.css: Pagina de estilo para poder definir el template cuando la aplicación web tiene el atributo cssStyle en verdadero.
- Error.xhtml: página de error cuando la autenticación no se llevó a cabo correctamente.
- Login.xhtml: Pagina de autenticación cuando la aplicación web tiene seguridad.
- Role.java: Entidad para persistir los roles que tendrá la aplicación web.

- User.java: Entidad para persistir los usuarios que podrán autenticarse en la aplicación web.
- UserFacade: *Ejb* de la entidad User para poder llevar a cabo los métodos de negocio de la entidad.
- Utils.java: Clase de soporte que contiene para encriptar la contraseña durante la autenticación.

1.4.7 Secciones marcadas para modificación por parte de los desarrolladores

Debido a que la transformación del modelo a la aplicación web en tecnología Java EE 6, es solamente una punto inicial para generar una aplicación web, por lo que es lógico pensar en que los desarrolladores modificaran el código generado de la aplicación web, sobre todo en los métodos de negocio que están dentro de los *ejb*. Por esta razón es necesario que sean protegidas secciones de los archivos que son generados, para que cuando sea regenerada la aplicación web, no se pierdan las modificaciones hechas en estas secciones. Acceleo para poder cumplir con esto, permite establecer las secciones protegidas dentro de su plantilla utilizando las etiquetas *<%starUserCode%>* y la etiqueta *<%endUserCode%>*, permitiendo a los desarrolladores insertar el código en específico.

Dentro de la herramienta Wapp Generator se decidió proteger las secciones donde se generaban los métodos soportados por los *ejbs*, debido a que la implementación de la lógica de negocio no se puede abstraer del modelo de la aplicación web. En la figura 20 se ve el código generado de la aplicación modelado y las áreas protegidas:

```
public void guardar(){
    try{
    // <%startUserCode%> before
    //TODO Write here the code to execute before
    // <%endUserCode%>
    }catch (Exception e) {
    }
}
public void edit(){
    try{
    // <%startUserCode%> before
    //TODO Write here the code to execute before
    // <%endUserCode%>
    }catch (Exception e) {
    }
}
```

Figura 20: Protección áreas de desarrollo.

Como se observa se generan las áreas protegidas, que estarán vinculadas con el modelo de la aplicación. Asimismo no solamente se deben definir las áreas protegidas, sino que también se debe configurar Acceleo para que permita la trazabilidad entre el modelo y el código generado. Esta configuración se hace marcando la opción de trazabilidad dentro de la configuración de Acceleo, a continuación se muestra la ilustración para habilitar la trazabilidad:

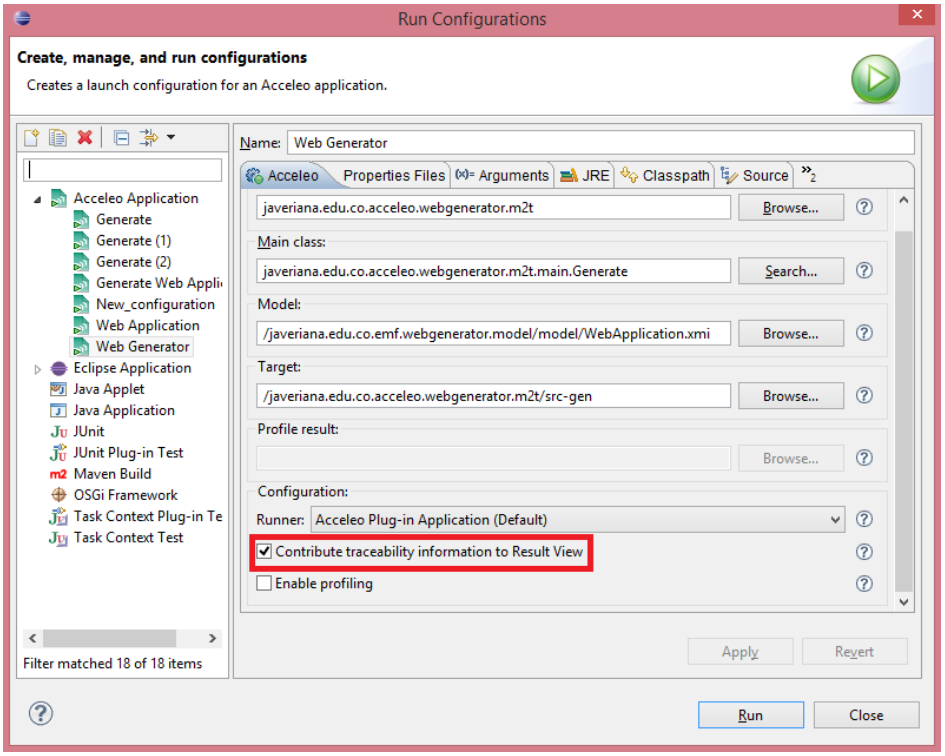

Figura 21: Configuración ejecución trazabilidad.

El cuadro rojo de la anterior imagen muestra la opción para habilitar la trazabilidad en Acceleo.

2. Descripción de uso de las herramientas construidas: ambiente Wapp Generator

A continuación se muestra el uso de la herramienta Wapp Generator especificando los diferentes roles que intervienen a lo largo del uso de la herramienta. Durante el desarrollo de la herramienta se identificaron cuatro roles principales: Rol de mantenimiento de meta-modelo, rol modelador de las aplicaciones web, rol desarrollador que completa la aplicación y el rol de mantenimiento de los transformadores.

2.1 Rol de mantenimiento del meta-modelo

El rol de mantenimiento del meta-modelo es el encargado de modificar y agregar el meta-modelo de Wapp Generator. Mediante el archivo *.ecorediag*, se permite definir nuevos elementos del meta-modelo por medio de la facilidad que presta la paleta, incorporada en EMF. En la figura 22 se muestra el ambiente de Acceleo para definir el meta-modelo:

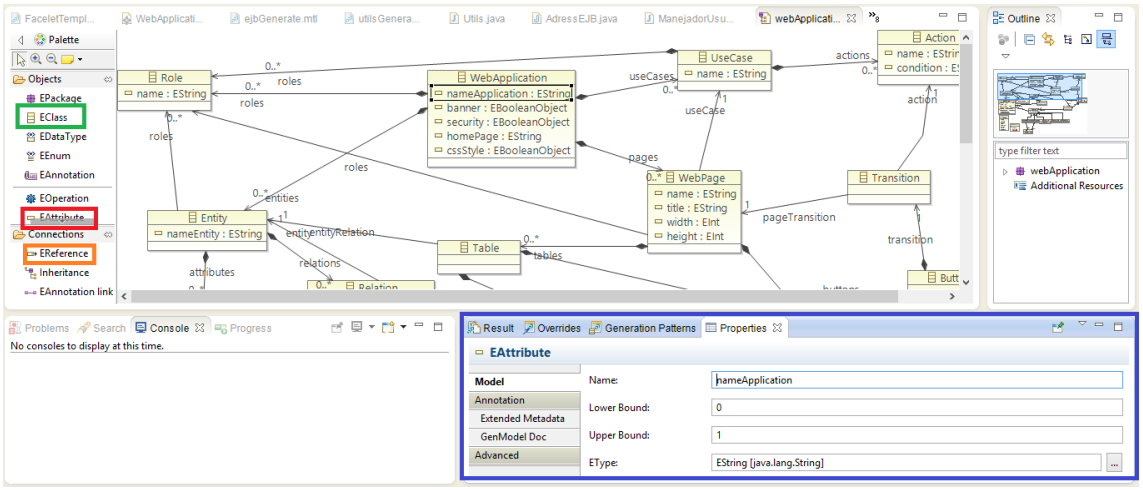

Figura 22: Diagrama archivo webApplication.ecorediag.

En la anterior ilustración se muestra señalados los elementos más utilizados durante la elaboración del meta-modelo. El recuadro verde muestra dentro de la paleta la opción para agregar una nueva meta-clase, por otro lado el recuadro rojo muestra la opción para agregar un nuevo atributo y adicionalmente el recuadro naranja muestra la forma para relacionar las meta-clases en el meta-modelo. Por ultimo en el recuadro azul se muestra el cuadro de propiedades que es utilizado a lo largo de la definición del meta-modelo.

2.2 Rol modelador de aplicaciones web

La función de este rol es modelar una aplicación web en particular utilizando el meta-modelo definido previamente. Utilizando una instancia dinámica del modelo, nos permite modelar este de forma jerárquica, donde el nodo de nivel más alto será la aplicación, teniendo como hijos un conjunto de páginas web, un conjunto de entidades, un conjunto de roles y un conjunto de casos de uso. Para ilustrar como se modela la aplicación se muestra en la figura 23 con un ejemplo:

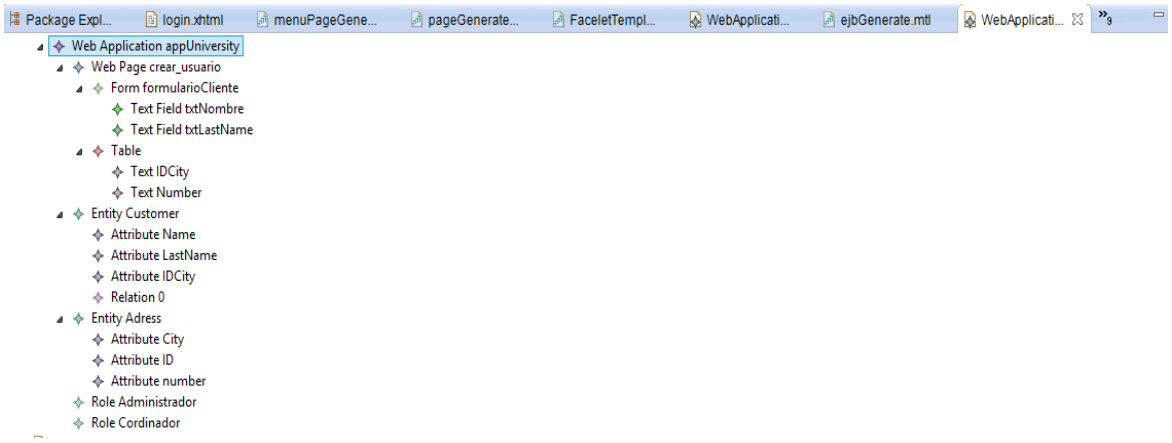

Figura 23: Modelo particular de aplicación web.

En el anterior ejemplo se muestra el modelo de una aplicación web llamada *appUniversity*, que tiene dos roles llamados administrador y coordinador. Por otro lado se modelaron dos entidades, una llamada *Customer* y la segunda llama *Adress*. Finalmente se modelo una página web, lo cual tiene por un lado un formulario con dos campos y una tabla que tiene dos columnas.

En la figura 24 se muestra la ventana de propiedades de una instancia de la meta-clase *WebApplication*:

Property	Value
Banner	true
Css Style	true
Home Page	usuario
Name Application	webapplication1
Security	true

Figura 24: Propiedades meta-clase WebApplication.

Como se observa se dan valores a los atributos de la instancia de la meta-clase *WebApplication*, definiendo los atributos de, *Banner*, booleano que indica si la aplicación web posee banner, *CssStyle*, atributo también booleano que define si la aplicación web posee un estilo Css predefinido, *HomePage*, que define la página de inicio de la aplicación web, *NameApplication*, define el nombre de la aplicación web y por último el atributo *Security* que indica si la aplicación web cuenta con seguridad.

En la figura 25 se muestra las propiedades de la meta-clase *Entity*:

Property	Value
Name Entity	Estudiante
Roles	Role administrador, Role Cordinador

Figura 25: Propiedades meta-clase Entity.

Las propiedades de la *NameEntity*, define el nombre que tiene la entidad y el atributo *Roles* define los roles que podrán acceder al CRUD de la entidad.

Una vez modelada la aplicación se transforma está a la tecnología Java EE 6. La transformación se lleva a cabo ejecutando el archivo *generate.mtl*, que se encuentra en el proyecto *javeriana.edu.co.acceleo.webgenerator.m2t*. En la figura 26 se muestra la ubicación del archivo *generate.mtl*:

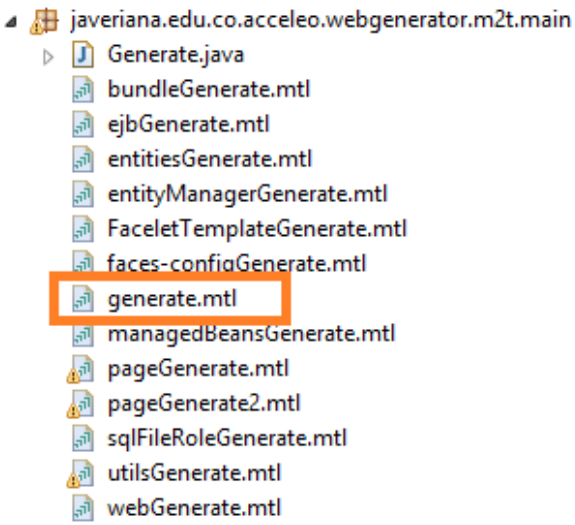

Figura 26: Ubicación archivo generate.mtl

Una vez seleccionado el archivo *genrate.mtl*, se da click derecho, eligiendo la opción *Run Configurations* del menú, una vez elegido esto se muestra la siguiente ventana:

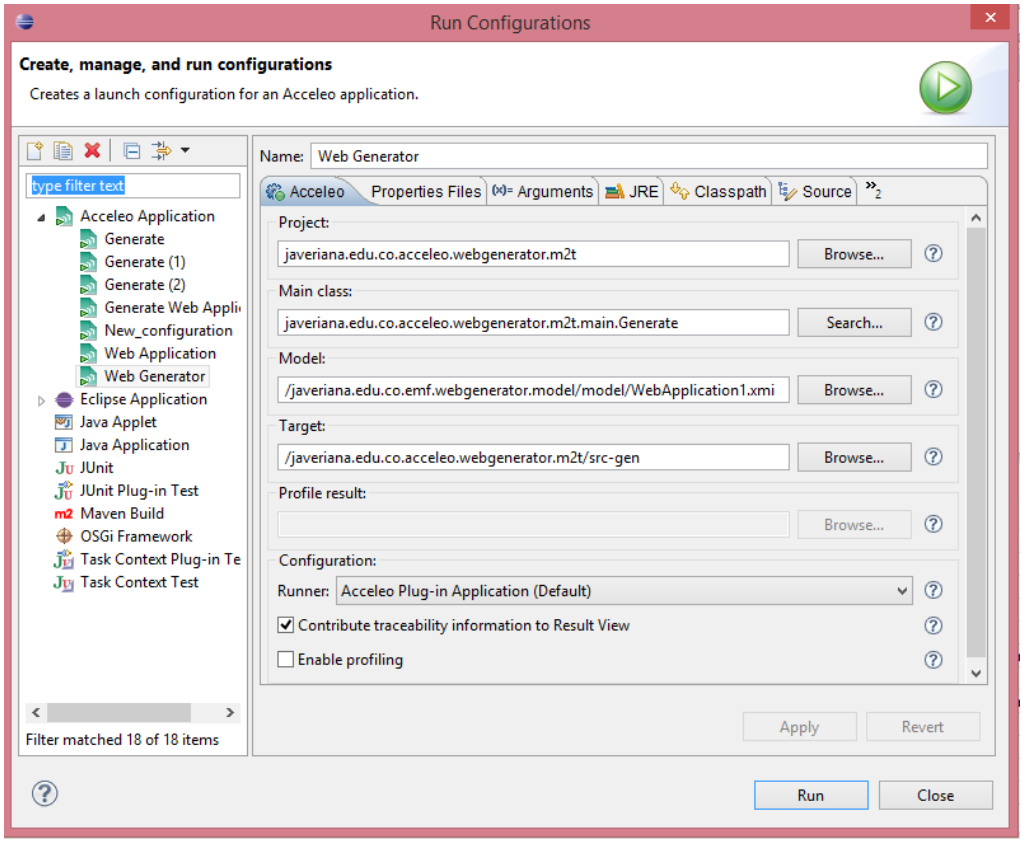

Figura 27: Ventana *Run Configurations.*

En esta ventana se configura las opciones para la ejecución donde se elige: el modelo de la aplicación web y si se desea habilitar la opción de trazabilidad del proyecto.

Una vez transformado el modelo a la aplicación Java EE 6, este es generado dentro de la carpeta *src-gen* en el proyecto *javeriana.edu.co.acceleo.webgenerator.m2t*, la estructura del proyecto generado se muestra en la figura 28:

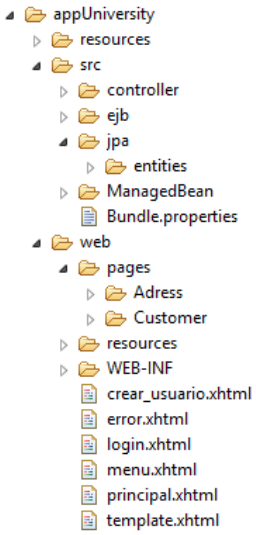

Figura 28: Estructura proyecto generado.

El nombre que toma la aplicación es el nombre que se le dio a la meta-clase *WebApplication*, dentro de la *src/ejb* se encuentran los *ejbs* de la aplicación, que son tomados de los casos de uso modelados y de los CRUD de las entidades. Por otro lado dentro de la carpeta *jpa/entities* se encuentra las entidades modeladas de la aplicación web. Dentro de la carpeta web se encuentran las páginas modeladas y en la carpeta *Pages* se encuentran las carpetas de cada entidad modelada, que contienen las páginas necesarias para poder llevar a cabo el CRUD de estas. Por ultimo en la carpeta *WEB-INF* se encuentra el archivo *web.xml* y el archivo *faces-config.xml*.

Por último una vez configurado el servidor de aplicaciones y la base de datos donde se almacenara los datos de la aplicación, se despliega la aplicación generada. En la figura 29 se muestra una página en ejecución:

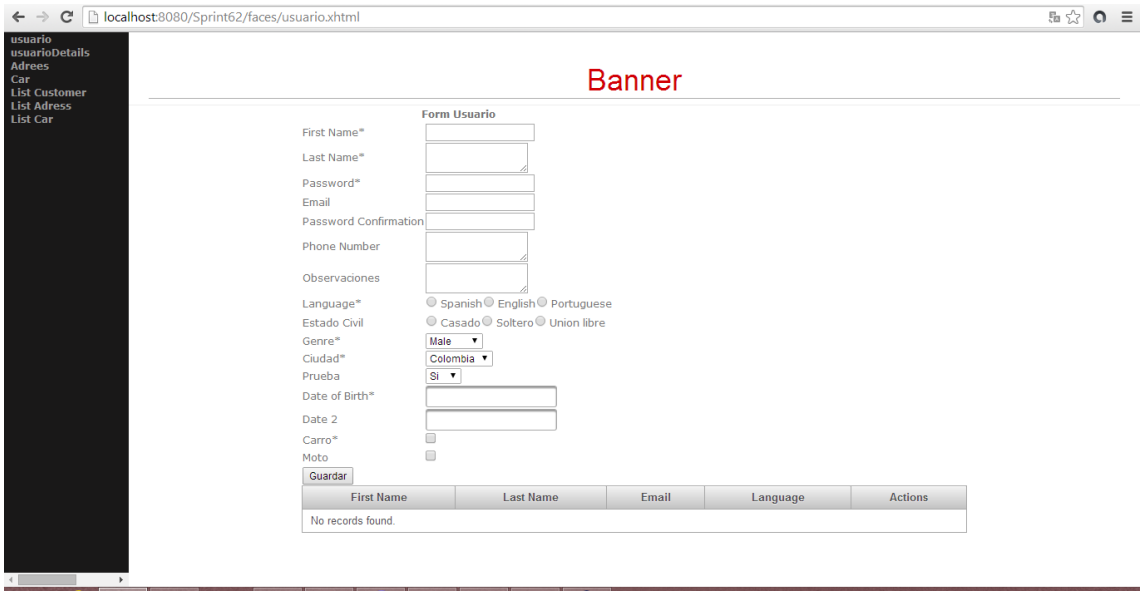

Figura 29: Página en ejecución.

Como se observa se modelo una página web con un formulario y una tabla, asimismo se observa que esta tiene una plantilla y un *Banner* en la aplicación web.

Igualmente en la figura 30 se ve la página View generada dentro del CRUD de entidades:

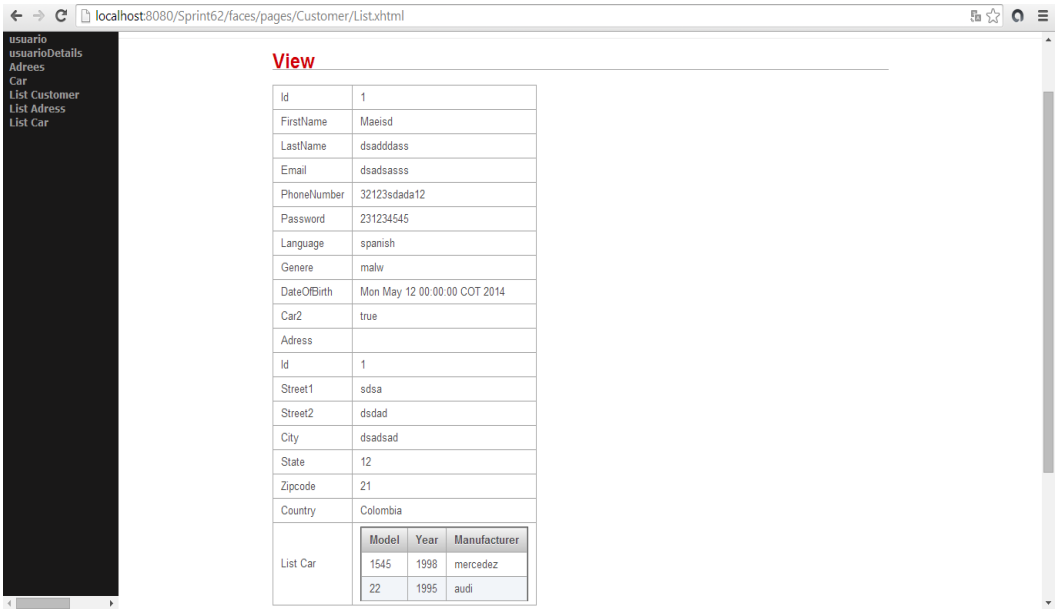

Figura 30: Pagina CRUD Entidad.

Como se observa esta es la página del detalle de una entidad *Customer*, donde se muestran los valores de los atributos de la entidad, los valores también de las entidades que se relacionan con la entidad *Customer*, que en este caso son por un lado la entidad *Adress* que es uno a uno y por último la relación con la entidad *Car* que es uno a muchos donde se muestra la lista de las entidades relacionadas con *Customer*.

Finalmente en la figura 31 se muestra la arquitectura de la aplicación generada por el ambiente Wapp Generator:

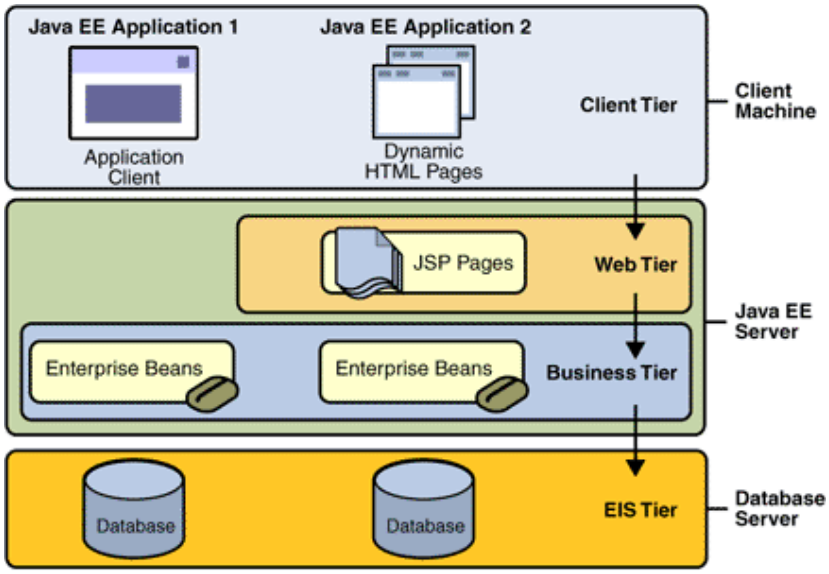

Figura 31: Arquitectura Aplicación web generada.

Por un lado se encuentra el contenedor web, en donde se encuentra las páginas web con tecnología JSF. En la capa de negocio se encuentra los *Enterprise Java Beans*, que contienen la lógica de negocio de la aplicación, que será utilizada por la capa Web, para poder llevar a cabo las solicitudes del cliente. Por último la capa de negocio se comunica con la capa de datos, donde estará la base de datos que almacena la información de la aplicación web.

2.3 Rol desarrollador que completa el código generado

La aplicación web generada en Java EE 6 no es completa, debido por un lado a las configuraciones del *realm* de seguridad, descriptores específicos del servidor, para tener seguridad en la aplicación web. Asimismo se debe crear la unidad de persistencia donde se almacenaran los datos de la aplicación web. Por último, debido a que por medio del modelo de una aplicación web no se puede abstraer la lógica de negocio de un método de un *ejb*, este debe ser implementado por parte del desarrollador para cumplir con la lógica de negocio.

2.4 Rol de mantenimiento de los transformadores para Java EE 6

Este rol se encarga de mantener las plantillas en Acceleo, modificar estas cuando sea necesario cambiar las reglas de transformación. Por otro lado debe mantener la integridad entre el meta-modelo y las plantillas de Acceleo para Java EE 6, debido a cambios que se pueden presentar en la definición del meta-modelo, como la inclusión de nuevos elementos, modificación de una meta-clase, etc.

V – RESULTADOS Y VALIDACIÓN

1. Resultados

Los resultados de este trabajo de grado se derivan en dos grandes componentes:

1. Meta-modelo
2. Transformador

El primer componente hace referencia al meta-modelo, donde se puede modelar aplicaciones Web mediante un lenguaje específico de dominio (DSL). (Ver sección 1.3 Definición de meta-modelo)

El segundo componente hace referencia al transformador, donde se puede generar código para la tecnología Java EE 6. (Ver sección 1.4 Desarrollo del transformador para Java EE 6 con plantillas Acceleo)

La utilidad de la herramienta Wapp Generator es brindar un nuevo enfoque organizacional mediante 4 roles de gestión (Ver sección 2. Descripción de uso de las herramientas construidas: ambiente Wapp Generator) automatizando partes del proceso para la construcción de aplicaciones web.

2. Validación

La validación de la herramienta Wapp Generator se generó mediante pruebas de aceptación brindadas en encuestas a la empresa Heinsohn.

2.1. Pruebas de Aceptación

Para la aceptación del ambiente Wapp Generator se realizó una prueba a tres ingenieros de sistemas de Heisohn. La encuesta estaba compuesta por preguntas dirigidas a los diferentes roles que se encuentran en el ambiente Wapp Generator, estos roles son: Rol Mantenimiento del Meta-modelo, Rol Modelador de la Aplicación, Rol Desarrollador que Completa el Código Generado y se calificaba de 1 a 5, donde 1 representa la nota más baja y cinco la nota más alta. Los empleados entrevistas fueron: María Catalina Acero Rozo Consulting Manager, Angie Zambrano Arquitecta software y Leonardo Corral Arquitecto de software. En los anexos se muestran en detalle las encuestas contestadas por estos tres ingenieros.

A continuación se muestra un promedio de cada pregunta se presenta el promedio a cada pregunta realizada, donde 1 es mala, 2 es deficiente 3 es regular, 4 es Bueno y 5 Excelente:

1.¿Qué tan fácil le parece agregar y modificar meta-clases al metamodelo de la herramienta WAPP Generator?	4
2.¿Qué tan fácil le parece establecer propiedades y asociaciones entre meta-clases?	4
3. ¿Qué tan completo le parece el meta-modelo del Wapp Generator para representar una aplicación web independiente de tecnología?	4
4.¿Qué tan fácil le parece definir los elementos de una aplicación web utilizando el meta-modelo Wapp Generator?	5
5. ¿Qué tan suficiente le parece el diagrama jerárquico combinado con la vista de propiedades para definir toda una aplicación web?	4
6. ¿Qué tan fácil le parece generar la aplicación web para la tecnología Java EE 6?	4.7
7. ¿Qué tan fácil le parece realizar los pasos manuales para poner en ejecución una aplicación generada en la tecnología Java EE 6 en el servidor Glassfish 4.0?	4
8. ¿Qué tan fácil le parece completar los Ejbs generados?	4.7
9. ¿Qué tan útil le parecen los CRUD generados?	5
10. ¿Considera que las interfaces resultantes de la aplicación Web son claras?	4
11. ¿Qué tan fácil le parece programar los transformadores Acceleo para Java EE 6?	3.7
12. ¿Qué tan fácil le parece modificar los transformadores de Wapp Generator?	3.7
13. ¿Qué tan fácil le parece agregar nuevos transformadores de Wapp Generator?	3.3
14. ¿La herramienta Wapp Generator está acorde con las necesidades de la empresa?	4
15. Respecto a la herramienta actual que tiene la empresa para generar aplicaciones web, ¿Qué tan conveniente le parece cambiarla por Wapp Generator?	3.7
16. ¿Qué tan viable le parece que la empresa asuma el cambio cultural de trabajar con MDE introduciendo los cuatro roles de personas que se han visto en esta presentación?	3.7

Figura 32: Promedios encuesta de satisfacción.

Como se observan los promedios de los puntajes de la encuesta, se muestra en general una buena respuesta hacia el ambiente Wapp Generator, teniendo en la mayoría de preguntas un promedio por arriba de 4. Teniendo únicamente un promedio de 3.5, en las preguntas relacionadas con el mantenimiento del transformador a tecnología Java EE 6, sin embargo en las demás preguntas se muestra un gran interés y aceptación por parte de los entrevistados hacia el ambiente Wapp Generator para modelar y generar las aplicaciones web. Asimismo cabe resaltar tres preguntas que reflejan la importancia del trabajo de grado dentro de la industria de desarrollo de software. La primera es que si la herramienta se ajusta a las necesidades de la empresa, obteniendo un promedio de 3.7, demostrando que el ambiente Wapp Generator está apto para poder ser usado en una empresa que desarrolla software. La segunda pregunta tiene que ver la comparación del ambiente Wapp Generator con la herramienta que tiene Heisohn, teniendo una aceptación al cambio a Wapp Generator. Por último se acepta el cambio cultural hacia el uso de herramientas que se apoyen en MDE.

VI - CONCLUSIONES, RECOMENDACIONES Y TRABAJO FUTUROS

1. Conclusiones

La investigación de herramientas y metodologías asociadas con el desarrollo de este trabajo de grado fue crucial para poder proporcionar los resultados esperados en el meta-modelo y transformador de la herramienta generada.

Elegir una metodología ágil para el desarrollo del trabajo de grado fue determinante para proporcionar en cada iteración un incremento de la herramienta Wapp Generator.

El meta-modelo de la herramienta Wapp Generator brinda la creación de aplicaciones Web independiente de tecnología, por lo que no sería necesario hacer modificaciones para tecnologías concretas.

El transformador de la herramienta Wapp Generator genero satisfactoriamente el código relacionado a una aplicación Web para la tecnología especifica Java EE 6, para un futuro en una siguiente fase se podría generar transformadores para tecnologías diferentes como .NET.

Este proyecto beneficiará en gran medida a las empresas de desarrollo de software, ya que cambia el paradigma de cómo construir aplicaciones Web, optimizando tiempo de desarrollo, costos y reciclaje de componentes ya elaborados.

2. Recomendaciones

Mediante el enfoque de desarrollo dirigido por modelos es necesario investigar adecuadamente las herramientas para implementar tanto el lenguaje específico de dominio y el transformador, ya que puede ser un poco confuso utilizar varias herramientas para el mismo proceso.

Utilizar una metodología ágil para representar pequeños incrementos fue de gran ayuda, ya que a la medida que se iba modelando, se transformaba cada requerimiento, por lo que al final se podía tener certeza de las funcionalidades de la herramienta Wapp Generator con sus respectivas retroalimentaciones.

3. Trabajos Futuros

Como trabajo futuro se identificaron los siguientes puntos:

- Mostrar el lenguaje específico de dominio (DSL) en un entorno gráfico.
- Generar transformadores para otra tecnología particular (.NET), logrando mantener el meta-modelo establecido.
- Modificar el meta-modelo actual para generar mayores componentes dentro de una página web.
- Modificar el transformador actual para soportar Java EE 7.
- Convertir el DSL en un instalador para que no sea necesario abrir una nueva instancia del meta-modelo, cada vez que se quiera utilizar la herramienta Wapp Generator.

VII - REFERENCIAS Y BIBLIOGRAFÍA

1. Referencias

[1] P. Cáceres, E. Marcos, and G. Kybele, "Procesos ágiles para el desarrollo de aplicaciones Web," *Taller de Web Engineering de las Jornadas de Ingeniería del Software y Bases de Datos de*, vol. 2001, 2001.

[2] D. Montoro Mouzo, "Generación automática de código para la plataforma domótica KNX/EIB en un marco dirigido por modelos," 2010.

[3] C. E. M. MARÍN, P. A. G. GARCÍA, J. M. C. LOVELLE, and O. S. MARTÍNEZ, "Aplicación de ingeniería dirigida por modelos (MDA), para la construcción de una herramienta de modelado de dominio específico (DSM) y la creación de módulos en sistemas de gestión de aprendizaje (LMS) independientes de la plataforma," *Dyna*, vol. 78, no. 169, pp. 43–52, 2011.

[4] J. G. Molina, J. Rodríguez, M. Menárguez, M. J. Ortín, and J. Sánchez, "Un estudio comparativo de dos herramientas MDA: OptimalJ y ArcStyler," *Actas del I Taller sobre Desarrollo Dirigido por Modelos, MDA y Aplicaciones (DSDM'04), Málaga, España*, vol. 9.

[5] M. Vanzetti, "Desarrollo Dirigido por Modelos de Procesos de negocio Colaborativos: Análisis de herramientas para la transformación de modelos."

[6] Modelum, G. Master MNTI Desarrollo Dirigido por Modelos Seminario 2 - Sintaxis concretas textuales, 1–11, 2010.

[7] J. B. Quintero and R. A. de Páez, "Marco de Referencia para la Evaluación de Herramientas basadas en MDA.," in *CIbSE*, 2007, pp. 225–238.

[8] D. Colque and R. Valdivia, "Integración de Tecnologías en una plataforma J2EE dirigida por modelos," *Ingeniare. Revista chilena de ingeniería*, vol. 14, no. 3, pp. 265–275, 2006.

[9] R. L. Glass, "The Standish report: does it really describe a software crisis?," *Communications of the ACM*, vol. 49, no. 8, pp. 15–16, 2006.

[10] N. M. Báez, C. Neil, and C. Pons, "Usando ATL en la transformación de modelos multidimensionales temporales," in *XIII Congreso Argentino de Ciencias de la Computación*, 2007.

[11] L. A. Guerrero, "Modelando Interfaces para Aplicaciones Web," *En Ingeniería del Software en la Década del*, 2000.

[12] J. Villalobos Beato, "Sistema para la maquetación de componentes JSF: JSFcomposer," 2009.

[13] Franky, M. C. MDA : Arquitectura Dirigida por Modelos Temario, 1–52, 2011.

[14] Franky, M. C. Aplicando enfoque MDE a aplicaciones Temario, 1–41, 2011.

[15] C. Pons, R. S. Giandini, and G. Pérez, "Desarrollo de software dirigido por modelos," 2010.

[16] F. Heidenreich, J. Johannes, S. Karol, M. Seifert, and C. Wende, "Model-Based Language Engineering with EMFText," in *Generative and Transformational Techniques in Software Engineering IV*, Springer, 2013, pp. 322–345.

[17] F. Jouault and I. Kurtev, "Transforming models with ATL," in *Satellite Events at the Models 2005 Conference*, 2006, pp. 128–138.

[18] M. J. Escalona and N. Koch, "Ingeniería de Requisitos en Aplicaciones para la Web–Un estudio comparativo," *Universidad de Sevilla*, 2002.

[19] M. P. Díaz, S. Montero, and I. Aedo, *Ingeniería de la Web y Patrones de Diseño*. Pearson Prentice Hall, 2005.

[20] E. López, M. González, M. López, and E. L. Iduñate, "Proceso de desarrollo de software mediante herramientas MDA," *Revista Iberoamericana de Sistemas, Cibernética e Informática*, vol. 3, no. 2, pp. 6–10, 2006.

[21] Booch, Grady. Saving myself. Booch04a. http://booch.com/architecture/blog. July 22, 2004.

[22] Booch, G. et al. An MDA Manifesto. In Frankel, D. and Parodi J. (eds) The MDA Journal: Model Driven Architecture Straight from the Masters, 2004.

[23] Francisco Durán Muñoz, Javier Troya Castilla, and Antonio Vallecillo Moreno, *Desarrollo de Software Dirigido por Modelos*. Madrid, 2007.

[25] D. Steinberg, F. Budinsky, M. Paternostro, and E. Merks, EMF: Eclipse Modeling Framework 2.0, 2nd ed. Addison-Wesley Professional, 2009.

[26] F. Heidenreich, J. Johannes, S. Karol, M. Seifert, and C. Wende, "Derivation and Refinement of Textual Syntax for Models," in *Model Driven Architecture - Foundations and Applications*, vol. 5562, D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, R. F. Paige, A. Hartman, and A. Rensink, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 114–129.

[27] T. Groussard, Recursos Informáticos Java Enterprise Edition-Desarrollo de aplicaciones web con Java EE 6. Ediciones ENI, 2010.

[28] A. Goncalves. Beginning Java EE 6 Platform with GlassFish 3, Apress, 2 ed. 2010.

[29] N., C. Zakas, J. Mcpeak and J. Fawcett. Professional Ajax Wiley, 2 ed 2007.

[30] Fernando Pech-May, Mario A. Gomez-Rodriguez, Luis A. de la Cruz-Diaz, and Salvador U. Lara-Jeronimo, "Desarrollo de Aplicaciones web con JPA, EJB, JSF y PrimeFaces." Tabasco, México, 2010.

[31] C Schalk and E. Burns. JavaServer Faces 2.0: The Complete Reference. MC Graw

Hill 2010.

[32] D. Geary and C. Horstmann. Core JavaServer Faces. Prentice hall 3 ed 2010.

[33] D. Yang. Java Persistence with JPA. Outskirts Press. 2010.

[34] M. Keith and M. Schincariol. Pro JPA 2 Mastering the Java Persistence API. Apress, 2009.

[35] A. Lee Rubinger E Bill Burke. Enterprise JavaBeans 3.1. Safari, 6 Ed. 2010.

[36] F. Heidenreich, J. Johannes, M. Seifert, and C. Wende, "JaMoPP: The Java Model Parser and Printer," *TU Dresden Technical Report*, Sep. 2009.

[37] "Eclipse Foundation." [Online]. Available: http://www.eclipse.org/.

[38] "NetBeans IDE." [Online] Available: http://netbeans.org/.

[40] M. Á. S. Vidales, A. H. García, and L. J. Aguilar, "Una recomendación basada en MDA, BPM y SOA para el desarrollo de software a partir de procesos de negocio en un contexto de Negocio Bajo Demanda‖," University Pontificia of Salamanca, 2006.

[41] A. Delgado, "Desarrollo de Software con enfoque en el Negocio," in Conference Proceeding JISBD: I Taller sobre Procesos de Negocio e Ingeniería del Software (PNIS'07), September, Zaragoza, España, 2007.

[42] R. Anaya, "Una visión de la enseñanza de la ingeniería de software como apoyo al mejoramiento de las empresas de software," Revista Universidad EAFIT, vol. 42, no. 141, pp. 60–76, 2012.

[43] Lawrence P., Shari. *Ingeniería de software. Teoría y práctica.* Prentice Hall, 2002.

[44] "Heinsohn Business Technology." [Online]. Available: http://hgs.com.co/.

[45] "Heinsohn entre las 10 mejores empresas de Colombia con mayor calidad de desarrollo de software." [Online]. Available: http://fiti.gov.co/Noticia/Detalle/14.

[46] OMG *Documents "Model Driven Architecture". [Online]. Available: http://www.omg.org/mda/.*

[47] Q Omg, "Meta object facility (mof) 2.0 query/view/transformation specification," *Final Adopted Specification (November 2005)*, 2008.

[48] E. Jendrock, R. Cervera-Navarro, I. Evans, D. Gollapudi, K. Haase, W. Markito, and C. Srivathsa, *The Java EE 6 Tutorial: Advanced Topics*. Addison-Wesley, 2013.

[49] X. Notario Rubí, "Metodología SCRUM," 2014.

[50] Schwaber, K., Beedle, M.: *Agile Software Development with SCRUM*. In: Conchango 2006.

VIII - GLOSARIO

Meta-modelo: Definición de las restricciones, reglas y construcción de un modelo.

Meta-clase: Elemento dentro de la definición del meta-modelo. Al igual que una clase puede tener un conjunto de atributos y otro de relaciones.

UML: Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema.

MOF: establece un marco común de trabajo para las especificaciones del OMG, a la vez que provee de un repositorio de modelos y metamodelos. Mediante MOF puede definirse cualquier lenguaje de modelado, incluido UML.

XMI: define una traza que permite transformar modelos UML en XML para poder ser tratados automáticamente por otras aplicaciones.

Dirigido por modelos: se dice que MDA es dirigido por modelos porque usa los modelos para dirigir el ámbito del desarrollo, el diseño, la construcción, el despliegue, la operación, el mantenimiento y la modificación de los sistemas.

Independencia de la plataforma: Significa que un modelo es independiente de las facilidades o características que implementan las plataformas, de cualquier tipo (ej, JEE, .net).

Aplicación web: Aplicación informática cuyo medio de presentación es un

Navegador web.

Bean: Objeto Java de almacenamiento de información volátil en aplicaciones web J2EE.

Contenedor de servlets: Servidor web capaz de manejar el ciclo de vida de servlets.

Framework: Librería o herramienta para facilitar el desarrollo en una tecnología o entorno concreto.

HTML: HyperText Markup Language, Lenguaje de Marcado de Hipertexto.

Es el lenguaje en el que se escriben las páginas web.

Java EE 6: Sexta versión de Java Enterprise Edition lanzada el año 2009.

ManagedBean: Bean común de java, que da soporte a un objeto manejado dentro de la aplicación web.

Web.xml: Archivo descriptor de despliegue para una aplicación web en Java.

Facelet: Sistema de código abierto de plantillas web bajo la Licencia Apache y tecnología de controlador *JSF*.

EJB: modelo de componentes de negocio distribuido estándar del lado del servidor. Permite al programador abstraerse de los problemas de una aplicación empresarial para centrarse en el desarrollo de la lógica de negocio.

Realm: Dominio de seguridad, donde se definen las restricciones de una aplicación.

J2EE: Java Enterprise Edition, es la versión de Java para implementar aplicaciones web.

JSF: JavaServer Faces, estándar de Sun Microsystems para la capa de presentación de las aplicaciones web J2EE.

MVC: Modelo-Vista-Controlador, patrón de diseño web.

Servidor web: Aplicación especial instalada en un ordenador que permite que recibir y procesar peticiones de clientes remotos o locales de recursos locales.

Servlet: Clase Java capaz de procesar una determinada petición web y generar el resultado correspondiente.

XML: Extensible Markup Language, lenguaje para transporte y almacenamiento de datos.

Modelo: Un modelo consiste de un conjunto de elementos que describen alguna realidad física o hipotética (siendo una simplificación de esa realidad), típicamente un modelo se presenta combinando gráficos y texto.

Abstracción: Ignorar información que no es de interés en un contexto particular.

Clasificación: Agrupar información en base a propiedades comunes.

Metamodelo: Es un modelo de un lenguaje de modelado: define la estructura,

Semántica y restricciones de una familia de modelos.

Plataforma: Especificación de un ambiente de ejecución para un conjunto de modelos (ej: CORBA, .NET, JEE).

EMF: Eclipse Modeling Framework de modelado y para la generación de código, con características similares a MOF.

MDA: Especificación del paradigma MDD propuesta por el OMG.

MDD: Enfoque para el desarrollo de aplicaciones donde todo el proceso viene dirigido por modelos abstractos de alto nivel.

MOF: Lenguaje para el diseño de Metamodelos, a partir del que se pueden definir lenguajes de modelado.

MTF: Motor de transformación de modelos propuesto por IBM Rational. Parte de EMF.

OCL: Lenguaje adoptado por el OMG como parte de UML 2.0 para la descripción formal de expresiones en los modelos UML.

OMG: Consorcio dedicado al cuidado y al establecimiento de diversos estándares de tecnologías orientadas a objetos**.**

PIM: Modelo propuesto por el OMG para el diseño de modelos independientes de plataforma alguna.

PSM: Modelo propuesto por el OMG para el diseño de modelos dependientes de una plataforma específica.

Acceleo: Implementación pragmática del *Object Management Group (OMG)* al estándar MOF.

MTL: Extensión de una plantilla en un proyecto Acceleo.

CRUD: Acrónimo en inglés que significa *Create, Read, Update* y *Delete.*

Wapp Generator: Nombre de la herramienta desarrollada durante el transcurso de este trabajo de grado

IX - ANEXOS

Anexo 1. Post-Morten

El Post-morten del presente trabajo de grado se encuentra en el Anexo 1: Post Morten.

Anexo 2. Lista de requerimientos

La lista de los requerimientos detallada del ambiente Wapp Generator se encuentra en el documento Lista de Requerimientos.

Anexo 3. Documentación Meta-modelo

La documentación y el proceso del diseño del meta-modelo se encuentra dentro del documento DOCUMENTACION DEL METAMODELO DE LA APLICACIÓN WEB.

Anexo 4. Documentación del Transformador

La documentación y el proceso de desarrollo del transformador se encuentran dentro del documento DOCUMENTACIÓN DEL TRANSFORMADOR DEL METAMODELO DE LA APLICACIÓN WEB.

Anexo 5. Manual de Configuración

El manual de configuración del sistema para hacer uso del ambiente Wapp Generator se encuentra en el documento Manual de Configuración.

Anexo 6. Manual de Administración y Usuario

El manual donde se detalla la administración y el uso del ambiente Wapp Generator se encuentra en el documento Manual de Administración y Usuario

Anexo 7. Pruebas de Aceptación

El formato de las Pruebas de aceptación realizadas a los ingenieros de Heisohn se encuentra en los siguientes documentos:

- Encuesta de satisfaccion1.jpg.
- Encuesta de satisfaccion2.jpg.
- Encuesta satisfacción Wapp Generator MCA.xls

Anexo 8. Cronograma

Se muestra el cronograma con la planeación durante el desarrollo del presente trabajo de grado.