**Universidad de los Andes**

**Documento de proyecto de grado**

**WebPicture: Generador de editores de modelos basados en EMF y Picture**

**Presentado a:**

**Departamento de ingeniería de sistemas y computación**

**Presentado por:**

**Andrés Felipe Guzmán Bautista.**

**Asesores:**

**Mario Eduardo Sánchez Puccini**

**John Casallas**

**Bogotá**

**Colombia**

**2014**

Tabla de contenido

1. Introducción 6

1.1. Resumen 7

1.2. Objetivos 8

1.3. Motivación 8

1.4. Resultados esperados 8

1.5. Antecedentes 9

2. Marcos de referencia 9

2.1. Modelos y metamodelos 9

2.1.1. Conformidad 10

2.2. Model driven engineering (MDE) 11

2.2.1. Eclipse Modeling Framework (EMF) 11

2.2.2. Ecore 11

2.3. XText 12

2.4. Picture 12

3. Acerca de WebPicture 12

3.1. Arquitectura de la herramienta 13

3.1.1. Capa de presentación 14

3.1.2. Capa de lógica 14

3.1.3. Capa de datos 14

3.2. Puntos de vista y modelos de arquitectura 14

3.2.1. Vista de contexto 14

3.2.2. Vista de despliegue 15

3.2.3. Vista de información 16

3.2.3.1. Modelo de datos 17

3.2.3.2. Modelo de flujo de información 17

3.2.3.3. Modelo de ciclo de vida de información 17

3.3. Tecnologías utilizadas 17

3.3.1. HTML 5 17

3.3.2. JavaScript 17

3.3.2.1. Joint JS 17

3.3.2.2. SVG 17

3.3.3. MySQL Server 17

3.3.4. Eclipse 17

3.4. Transformación de datos y generación de editores 18

3.4.1. Proceso de generación de editores 18

3.4.2. Validación de elementos 18

3.4.3. Transformación de elementos 18

3.4.4. Del metamodelo al editor 18

4. Detalles de la herramienta 18

4.1. Requerimientos de instalación 18

4.1.1. Requerimientos del cliente 18

4.1.2. Requerimientos del servidor 18

4.1.2.1. Java 18

4.1.2.2. Eclipse 18

4.1.2.2.1. Extensiones y complementos 18

4.1.2.3. MySQL Server 18

4.1.2.4. Apache Tomcat 18

4.2. Requerimientos de uso y compatibilidad 19

4.3. Generación de editores 19

4.4. Requerimientos soportados 19

4.5. Validaciones disponibles 19

4.6. Limitaciones 19

4.7. Mejoras propuestas para la siguiente versión 19

5. Ejemplos disponibles 19

5.1. UML 19

5.2. BPMN 19

5.3. Organizational structure 19

6. Conclusiones 19

7. Bibliografía 19

8. Agradecimientos 20

# Introducción

En la practica de arquitectura empresarial, el uso de modelos y representaciones abstractas de la realidad de una organización desde varios puntos de vista aporta un amplio valor a su ejercicio [1]. Los artefactos utilizados en este ejercicio son creados con el fin de representar y/o modelar el estado o la realidad de una organización [2]. Los artefactos utilizados en la arquitectura empresarial son herramientas formales que se estructuran entorno a la especificación de un metamodelo.

Los metamodelos describen las propiedades estructurales tales como reglas, relaciones y construcción de un modelo. Sin embargo el metamodelo por si solo no es suficiente para construir un modelo. Un modelo se constituye de dos partes la primera es el metamodelo y segundo la especificación de su representación grafica. Pese a que un modelo solo cuenta con una sola especificación estructural (metamodelo), un mismo modelo puede tener diferentes representaciones graficas [3].

En los últimos años la practica de la arquitectura empresarial ha evolucionado en un ejercicio basado en el soporte de toma de decisiones basado en entregables. Los entregables de arquitectura empresarial consiste en cualquier documento, artefacto o producto tangible que le permita al arquitecto justificar las decisiones tomadas [4].

Los artefactos generados en la arquitectura empresarial consisten principalmente en documentos que contienen generalmente un conjunto de modelos que pueden representar el estado actual o esperado de la organización y justifican aspectos clave de las decisiones tomadas por el arquitecto.

En la actualidad la arquitectura empresarial se ha convertido en una practica formalizada que cuenta con metodologías, herramientas y *frameworks* formales alineados con estándares globales. No obstante los modelos establecidos por *frameworks* como TOGAF entre otros pueden quedarse cortos en algunas circunstancias, en las cuales el arquitecto requiere representar decisiones o aspectos de la realidad sin embargo no le es posible ya que con los metamodelos disponibles pueden no soportar dichos aspectos [5].

Adicionalmente en el ámbito de los proyectos de arquitectura el gobierno y control de los artefactos generados hacen parte de la complejidad y riesgos con los que debe se debe lidiar. En la practica el arquitecto empresarial debe contar con practicas, habilidades y herramientas que le permitan disminuir la complejidad generada en la creación de artefactos.

Es por esto que surge la necesidad de contar con herramientas que le brinden al arquitecto la capacidad de primero poder representar en modelos aspectos no soportados por los metamodelos estándares y segundo disminuir efectivamente la complejidad generada a partir de la construcción de artefactos.

Un ambiente integrado de arquitectura empresarial consiste en un extenso marco de herramientas concebidas para el diseño, construcción, difusión y análisis de artefactos de arquitectura.

Dicho ambiente cuenta con herramientas entre las cuales se destacan editores especializados, gestores documentales, repositorios de modelos, herramientas para construcción de editores y herramientas de análisis. Para el ejercicio de arquitectura empresarial contar con un ambiente integrado para modelado tiene amplias ventajas desde el ámbito de proyecto de arquitectura, principalmente la reutilización y cocreación de modelos.

En el marco de los proyectos de investigación del grupo de tecnologías de información y construcción de software (TISCw) de la universidad de los Andes, se han presentado distintos acercamientos y avances en la creación de un ambiente integrado de modelado para arquitectura; no obstante los prototipos construidos para distintas funcionalidades se han implementado de manera independiente por lo cual no se cuenta aun con ambiente completamente integrado y todavía no se ha estudiado de manera completa los requerimientos del ambiente en su totalidad.

No obstante no se existe una herramienta que permita al arquitecto crear, modificar y extender modelos; y a la vez controlar los artefactos generados de su ejercicio. En este documento se detalla la construcción de WebPicture una herramienta para la generación de editores ad hoc basado en tecnologías web.

WebPicture toma metamodelos construidos sobre el framework de EMF y se integra con Picture un lenguaje creado para describir de manera formal la representación grafica de un modelo para construir un editor grafico para el modelo especificado.

# Resumen

En la actualidad la arquitectura empresarial se ha convertido en una practica formalizada que cuenta con metodologías, herramientas y *frameworks* formales alineados con estándares globales. No obstante en el desarrollo del ejercicio los metamodelos definidos pueden quedarse cortos en algunas circunstancias.

Es por esto que surge la necesidad de contar con herramientas que le brinden al arquitecto la capacidad de primero poder representar en modelos aspectos no definidos por metamodelos convencionales y segundo disminuir efectivamente la complejidad generada a partir de la construcción de artefactos.

Un ambiente integrado de arquitectura empresarial consiste en un extenso marco de herramientas concebidas para el diseño, construcción, difusión y análisis de artefactos de arquitectura.

En este documento se detalla la construcción de WebPicture una herramienta para la generación de editores ad hoc basado en tecnologías web.

# Objetivos

* Generar un producto flexible, capaz de integrarse fácilmente con los demás componentes del ambiente.
* Crear un generador de editores ad hoc basado en tecnologías web.
* Crear una herramienta capaz de generar editores web utilizando como componentes de entrada metamodelos construidos con EMF y cuya representación grafica sea denotada con el lenguaje Picture.

# Motivación

En la actualidad la arquitectura empresarial se ha convertido en una practica formalizada que cuenta con metodologías, herramientas y *frameworks* formales alineados con estándares globales. No obstante los modelos establecidos por *frameworks* como TOGAF entre otros pueden quedarse cortos en algunas circunstancias, en las cuales el arquitecto requiere representar decisiones o aspectos de la realidad sin embargo no le es posible ya que con los metamodelos disponibles pueden no soportar dichos aspectos.

Existen tecnologías como GMF que permiten crear editores gráficos ad hoc basado en EMF [6]. Pese a ser altamente configurable GMF es complejo de utilizar y requiere primero la instalación de componentes adicionales sobre Eclipse y segundo que el usuario escriba las anotaciones sobre el metamodelo para generar el editor grafico. GMF y EMF son tecnologías ampliamente utilizadas en la industria, sin embargo para utilizarlas se requiere obligatoriamente contar con una instalación de Java.

El grupo TISCw de la universidad de los Andes ya ha realizado distintos aportes para cambiar el paradigma de la generación de editores en GMF. Desde esta perspectiva se propuso entonces la separación del modelo en su metamodelo construido sobre EMF y de su representación grafica.

Es por esto que se propuso la creación de Picture un lenguaje formal y estructurado construido sobre XText. Picture le permite al usuario describir formalmente la representación grafica de los elementos del metamodelo. Sin embargo los prototipos desarrollados aun dependían de la plataforma Eclipse.

WebPicture pretende transformar el proceso de generación de editores primero al tomar la aproximación de la separación del modelo en su metamodelo construido en EMF y su representación grafica descrita en el lenguaje Picture. Segundo agilizar el proceso al tomar el metamodelo y la representación grafica para generar editores web capaces de realizar validaciones ontológicas y lingüísticas.

# Resultados esperados

* Crear una herramienta capaz de generar editores web utilizando Picture para describir la representación grafica y metamodelos construidos sobre EMF.
* Generar editores capaces de realizar validaciones lingüísticas y ontológicas sobre los modelos generados.
* Crear una herramienta que soporte una interfaz web y requiera configuraciones simples.
* Generar una herramienta capaz de integrarse fácilmente con los componentes del ambiente integrado de arquitectura empresarial.

# Antecedentes

* Greta

Greta consiste en un framework construido sobre una arquitectura basada en Eclipse capaz de soportar operaciones de composición sobre modelos. Asimismo capaz de soportar operaciones de composición sobre elementos gráficos [7].

* Web Picture

Proyecto de grado de Edgar Sandoval, consiste en un editor web de modelos, capaz de permitir al usuario cargar metamodelos en formato .ecore y archivos .picture (contiene las relaciones entre elementos gráficos definidos y elementos del metamodelo) para construir y editar modelos.

* Picture Maker

Proyecto de grado de Oscar Martínez, consiste en un editor web de modelos capaz de realizar validación de modelos. Este editor permite la edición de elementos gráficos del modelo (archivos .picture) y asimismo genera el correspondiente código HTML para visualizar el modelo.

# Marcos de referencia

# Modelos y metamodelos

Un modelo es la representación abstracta y simplificada de una realidad o un sistema (SuS) [8]. Los metamodelos son una representación abstracta de un modelo en si. De acuerdo con la especificación de OMG y MOF *un metamodelo es un modelo que define el lenguaje y la estructura para construir un modelo* [9]*.* Un metamodelo define las propiedades estructurales, reglas, objetos y relaciones para la construcción de un modelo.

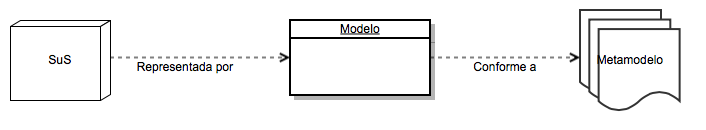


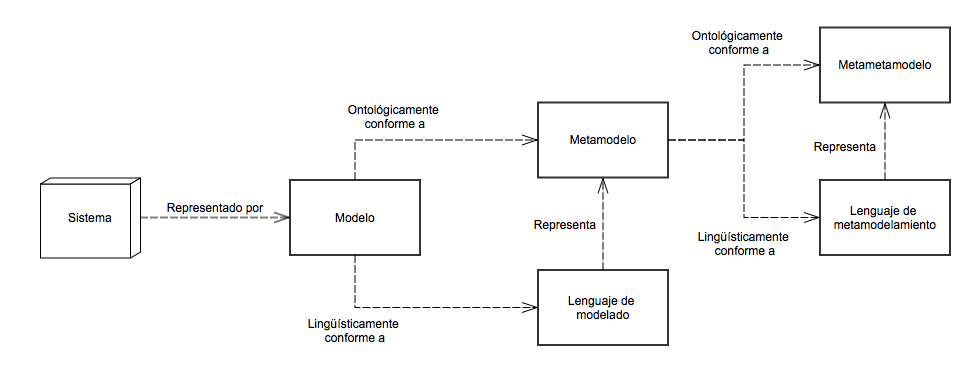
Figura 1: Estructura básica de los modelos y metamodelos

En la *Figura 1* se muestra la relación de conformidad entre un modelo y su metamodelo, entre un metamodelo se relacionan con los metamodelos. Por definición los modelos tienen una relación de conformidad a la especificación de un metamodelo definido, dicha relación de conformidad tiene dos partes, conformidad lingüística y conformidad ontológica.

La conformidad lingüística consiste implica que el modelo cumpla con la especificación lingüística del metamodelo. Por su parte la conformidad ontológica implica que el modelo cumpla las propiedades estructurales definidas por el metamodelo [9].

# Conformidad

Por definición los modelos son conformes lingüística y ontológicamente conformes a la especificación de un metamodelo.



*Figura 2: Modelos y conformidad [[1]](#footnote-1)*

La conformidad ontológica implica que los modelos son estructuralmente conformes a la definición de un metamodelo. Al mismo tiempo los modelos son lingüísticamente conformes a un lenguaje de modelado que representa al metamodelo. De la misma forma al ser un modelo per se, el metamodelo debe cumplir con las mismas propiedades de conformidad que un modelo.

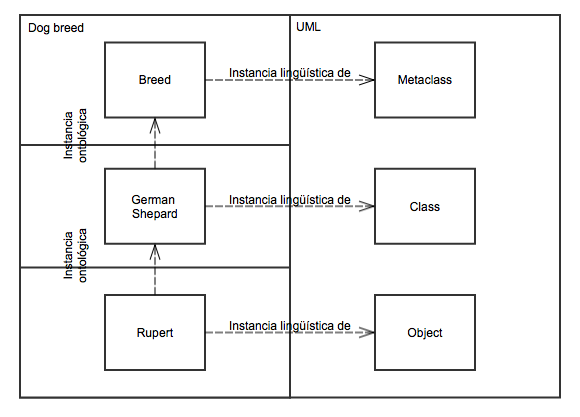


Figura 3: Conformidad lingüística y ontológica [[2]](#footnote-2)

Para explicar de manera practica la conformidad ontológica y lingüística de un modelo se propone el ejemplo de la *Figura 2*. Suponga que el lenguaje de modelamiento para este ejemplo es UML. En el modelo *Dog breed* el objeto UML *Rupert* es una instancia ontológica de la clase *German Shepard* y es ontológicamente conforme con esta. *German Shepard* es una instancia ontológica de la metaclase *Breed* y a su vez es ontológicamente conforme con esta. Adicionalmente en este ejemplo los elementos del modelo *Dog breed* son lingüísticamente conformes a la especificación del metamodelo de UML.

# Model driven engineering (MDE)

MDE consiste en una aproximación al desarrollo de software en la cual se sugiere primero desarrollar un modelo del sistema para luego transformar dicho modelo en ejecutables [9]. Los modelos juegan un papel clave en MDE ya que son el insumo clave para el proceso de transformación y generación de artefactos.

La idea básica detrás MDE consiste en utilizar modelos en diferentes niveles de abstracción y transformar dichos modelos en artefactos de software. No obstante para realizar el proceso de transformación se requieren lenguajes de dominio especifico (DSL) para especificar la notación de los modelos y herramientas capaces de realizar la transformación y generar los artefactos.

Los DSL son lenguajes especializados para describir modelos, estos lenguajes pueden ser gráficos y/o textuales. Por otra parte las herramientas responsables de hacer la transformación de modelos en artefactos de software requieren de modelos representados con un DSL que puedan consumir y transformar en código [10].

MDE tiene como premisa disminuir la complejidad del proceso de desarrollo, minimizar los errores potenciales sobre el código y aumentar la productividad.

# Eclipse Modeling Framework (EMF)

*Eclipse Modeling Framework* o EMF es un conjunto de herramientas desarrolladas por *Eclipse Foundation*. EMF brinda un framework de desarrollo orientado a MDE que permite crear modelos y transformarlos en código. Por medio de un modelo especificado en XMI/Ecore EMF puede generar el código Java base del modelo y así mismo generar un conjunto de editores básicos para manipular el modelo [11].

# Ecore

Ecore es el formato estándar aceptado por EMF para la creación de modelos. La sintaxis del lenguaje esta basada en XMI. Por otra parte los elementos estructurales del lenguaje provienen de elementos definidos de MOF. A continuación se describen los elementos básicos del lenguaje utilizados para describir la estructura de cualquier modelo [9].

* EClass: Se utiliza para representar una clase, este tiene un nombre y puede tener atributos (EAttributes).
* EAttribute: Es utilizado para representar un atributo, tiene un nombre y un tipo de dato.
* EReference: Se utiliza para representar una relación entre dos clases del modelo.
* EDataType: Se utiliza para representar un tipo de dato que puede pertenecer al conjunto de primitivas definidas o datos representados como objetos (Ej. Java.util.Date)

# XText

XText es una herramienta diseñada para la creación de lenguajes textuales de dominio especifico (DSL textuales). XText provee un conjunto de herramientas que a partir de la especificación de una gramática permiten generar artefactos para crear editores textuales para el lenguaje [12]. A continuación se describen los artefactos generados por Xtext a partir de la especificación de una gramática [13]:

* Metamodelo del lenguaje.
* Código generado a partir del metamodelo creado.
* Herramientas de validación del lenguaje (*Lexer* y *parser*)
* Editor textual del lenguaje.

XText es una herramienta sumamente poderosa que el agiliza el proceso de creación de lenguajes de dominio especifico y es capaz de generar herramientas de edición. Las herramientas generadas permiten ejecutar un editor basado en Eclipse validar la sintaxis escrita por con respecto al metamodelo del lenguaje.

# Picture

Picture es un lenguaje de dominio especifico creado para describir la representación grafica de un modelo de forma independiente a la especificación del metamodelo [14]. A continuación se presenta el metamodelo de la versión actual del lenguaje.

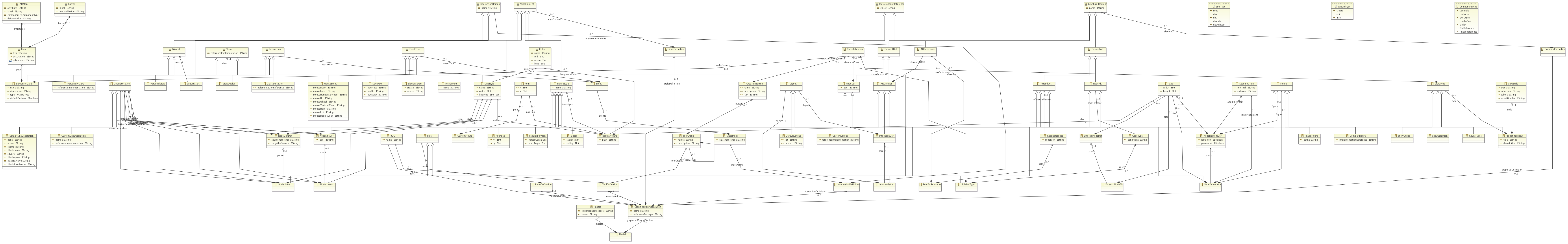


Figura 4: Metamodelo de Picture

Picture fue modelado y construido utilizando el framework de XText por lo que los elementos que se describan utilizando el lenguaje pueden ser extraídos a partir del metamodelo y ser validados utilizando las herramientas del framework.

# Acerca de WebPicture

WebPicture es una herramienta web diseñada para generar editores de modelos de arquitectura empresarial ad hoc. El objetivo de WebPicture es dar flexibilidad al ejercicio de arquitectura empresarial en el ámbito del modelamiento y al mismo tiempo mantener la formalidad al brindar al arquitecto la posibilidad de crear sus propios modelos.

WebPicture toma la especificación del metamodelo en formato *Ecore* y su representación grafica descrita en el lenguaje *Picture*. Estos elementos son directamente especificados por el usuario para generar un editor grafico. La herramienta genera editores construidos sobre tecnologías web, ligeros, fáciles de utilizar y compatibles con la mayoría de los navegadores web disponibles.

El proceso de generación de editores tiene tres partes. La primera consiste en la validación de la representación grafica con respecto al metamodelo del lenguaje y su validación con el metamodelo cargado. La segunda consiste en la transformación del metamodelo y su representación grafica en elementos web y la generación de reglas estructurales del modelo. Por ultimo el proceso reúne todos los elementos generados en un script web que puede ser reutilizado para la creación de distintos modelos.

# Arquitectura de la herramienta

WebPicture es una aplicación web distribuida en tres capas. Una capa de presentación, una capa de lógica y una de datos. En la *Figura 5* se muestra la arquitectura por capas de la herramienta.

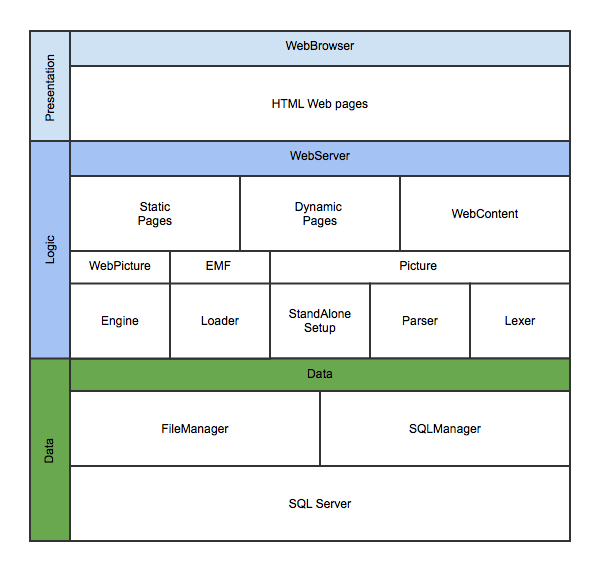


Figura 5: Vista por capas de la arquitectura de la aplicación

La capa de presentación contiene las paginas web en HTML que el usuario utiliza para interactuar con la aplicación.

La capa de lógica agrupa los componentes responsables de conectarse con el usuario vía HTML y los componentes responsables de manejar la lógica detrás de la validación de elementos y la generación de editores.

Por ultimo la capa de datos agrupa un manejador de archivos, un manejador transaccional y un servidor de base de datos.

# Capa de presentación

La capa de presentación contiene las paginas web utilizadas por el cliente para interactuar con la aplicación dichas paginas están construidas con tecnología HTML 5. En esta capa se muestra y recoge información utilizando contenedores web dinámicos y/o estáticos.

# Capa de lógica

La capa de lógica agrupa los componentes responsables de conectarse con el usuario vía HTML.

En esta capa también se encuentran los componentes de validar los elementos cargados por el usuario. Esta capa cuenta con tres componentes clave. Primero el componente *WebPicture Engine* se encarga de orquestar el proceso de generación de editores y validar los elementos cargados por el usuario. Segundo el componente *EMF Loader* es responsable de cargar el metamodelo y transformarlo en una estructura de datos intermedia. Por ultimo el componente *Picture StandAlone Setup* se ocupa de cargar y validar la implementación del lenguaje Picture cargada por el usuario.

En esta capa se hacen las transformaciones de información para generar los elementos gráficos y las reglas del editor.

# Capa de datos

En la capa de datos se agrupan tres componentes responsables de la persistencia de la información de la herramienta. El primer componente *FileManager* es el modulo responsable de administrar y almacenar la información real de los editores generados (scrips, recursos, contenido…) en el servidor. El segundo componente *SQL Sever* corresponde a un servidor de base de datos donde se almacena metainformación de los editores creados. Finalmente el componente *SQLManager* se encarga de conectarse la base de datos para extraer o insertar información de esta.

# Puntos de vista y modelos de arquitectura

A continuación se presentan distintos puntos de vista y modelos con el objetivo de explicar la arquitectura implementada.

# Vista de contexto

En la vista de contexto presentada en la *Figura 6* se muestra como el usuario interactúa con la herramienta y como interactúan los componentes de las capa de lógica y la capa de datos.

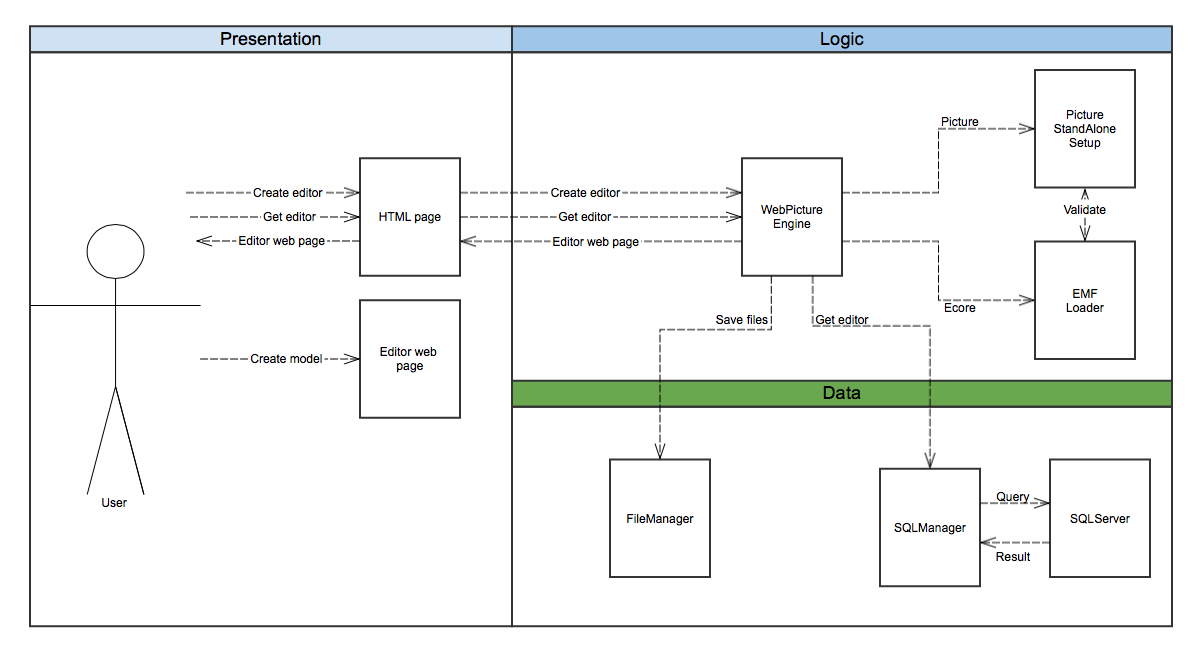


Figura 6: Vista de contexto de la herramienta

En esta vista se especifica como el usuario interactúa para crear un editor o utilizar un editor existente.

En el caso de que el usuario desee crear un nuevo editor debe cargar los archivos que describen el metamodelo (Ecore) y la representación grafica (Picture), estos archivos deben pasar por un proceso de validación y transformación para generar un editor.

En el caso de que el usuario desee utilizar un editor existente, la herramienta provee mecanismos para recuperar el editor.

Así mismo en el momento que el usuario obtiene el editor por cualquiera de los dos mecanismos el servidor deja de interactuar con el cliente y en este momento el editor es ejecutado en el navegador del cliente.

# Vista de despliegue

En la vista de despliegue de la Figura 7 se muestra la localización de los nodos de ejecución de la herramienta. En esta vista los distintos componentes se ejecutan en tres dispositivos, una maquina cliente que cuenta con un navegador web, una maquina servidor y un servidor de base de datos.

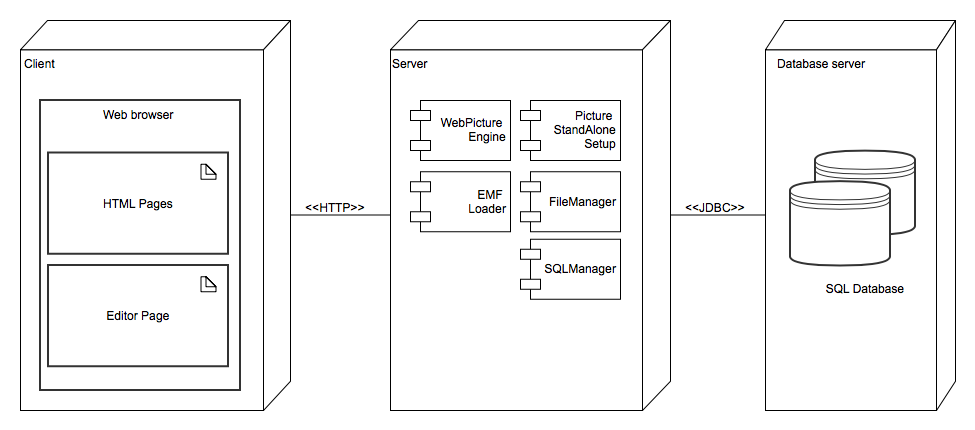


Figura 7: Vista de despliegue de la herramienta

La maquina del cliente visualiza y ejecuta las paginas web generadas en el servidor, esta se comunica con el servidor vía protocolo HTTP. Desde este nodo se envían las peticiones y reciben del servidor.

Por su parte en la maquina servidor se realizan las validaciones y la transformación de el metamodelo y la representación grafica para generar un editor web. Los editores web generados se son guardados en el sistema de archivos del servidor.

En el servidor de base de datos se almacena la metainformación de los editores generados. Este nodo se comunica con el servidor vía protocolo JDBC[[3]](#footnote-3). Este nodo permite realizar consultas SQL para manipular la metainformación de los editores.

# Vista de información

# Modelo estático de datos

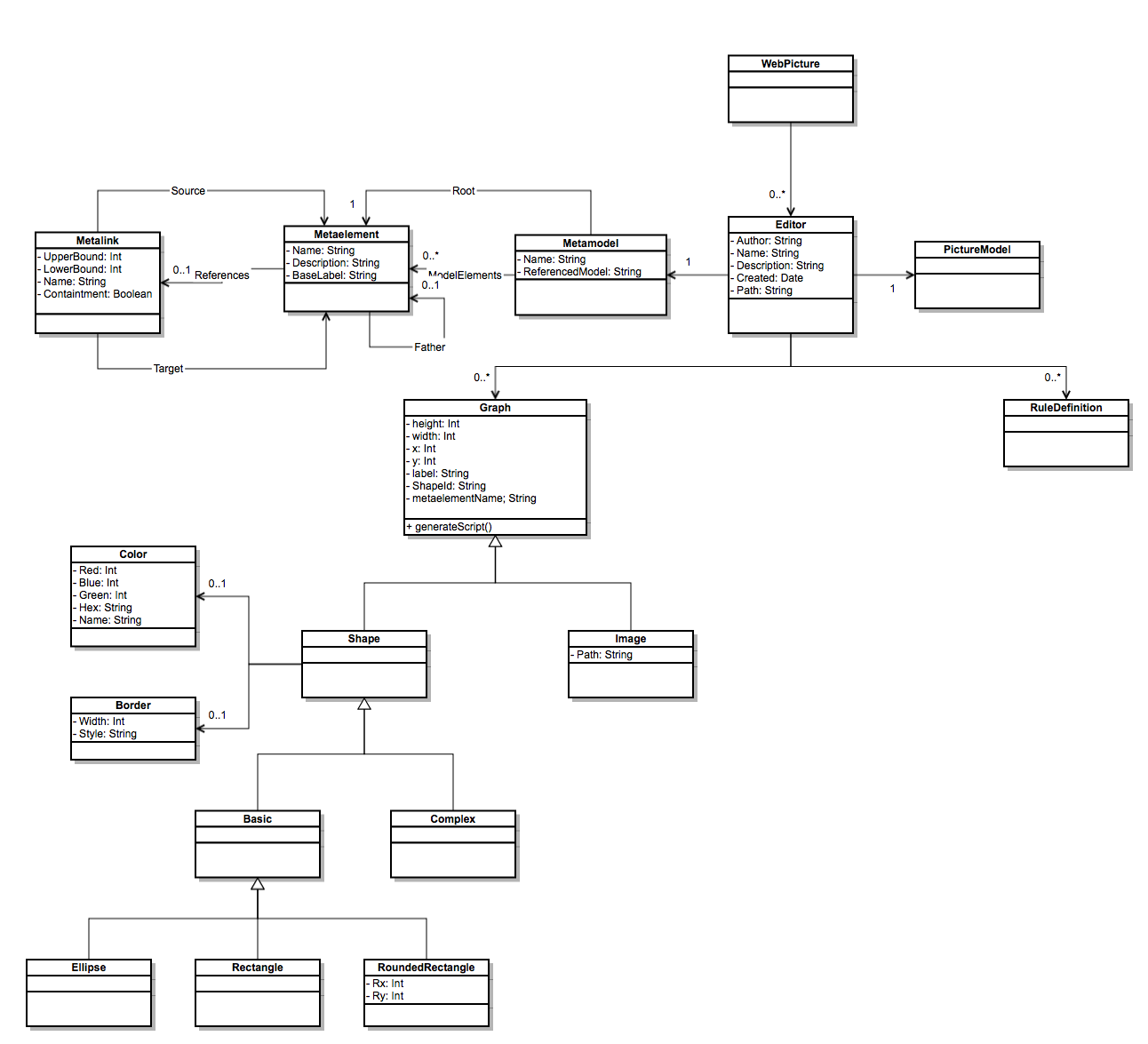


Figura 8: Modelo estático de datos

|  |  |
| --- | --- |
| Entidad | Descripción |
| WebPicture | Corresponde a la entidad principal del modelo de datos, es la responsable de administrar los editores generados |
| Editor | Representa a un editor web generado |
| Metamodel | Representa al metamodelo cargado por el usuario |
| Metaelement | Representa un elemento del metamodelo cargado por el usuario |
| Metalink | Representa una relación entre dos elementos del metamodelo cargado |
| PictureModel | Representa el modelo del lenguaje Picture |
| RuleDefinition | Representa una regla estructural generada |
| Graph | Representa un elemento grafico del modelo |
| Image | Representa un elemento grafico con una imagen de fondo |
| Shape | Representa una forma básica |
| Color | Representa un color de una forma básica |
| Border | Representa el borde de una forma básica |
| Complex | Representa una forma compleja |
| Basic | Representa una forma geométrica básica |
| Ellipse | Representa una forma circular |
| Rectangle | Representa una forma rectangular |
| RoundedRectangle | Representa una forma rectangular con bordes biselados |

Tabla 1: Catalogo de entidades de la herramienta

# Modelo de flujo de información

# Diagrama de secuencia: Creación de editores

# Tecnologías utilizadas

# HTML 5

# JavaScript

# Joint JS

# SVG

# MySQL Server

# Eclipse

# Transformación de datos y generación de editores

# Proceso de generación de editores

# Validación de elementos

# Transformación de elementos

# Del metamodelo al editor

# Detalles de la herramienta

# Requerimientos de instalación

# Requerimientos del cliente

# Requerimientos del servidor

# Java

# Eclipse

# Extensiones y complementos

# MySQL Server

# Apache Tomcat

# Requerimientos de uso y compatibilidad

# Generación de editores

# Requerimientos soportados

# Validaciones disponibles

# Limitaciones

# Mejoras propuestas para la siguiente versión

# Ejemplos disponibles

# UML

# BPMN

# Organizational structure

# Conclusiones

# Bibliografía

1. Naranjo D., Sánchez M., Villalobos J. PRIMROSe - A Tool for Enterprise Architecture Analysis  and Diagnosis. In: 16th International Conference on Enterprise Information Systems, Lisboa,  Portugal, 2014.
2. The Open group. TOGAF 9.1: Architecture content framework: Architectural artifacts, Estados Unidos, 2011.
3. Fill H., Karagiannis D. On the conceptualisation of modeling methods using the ADOxx metamodelling platform. In: Enterprise modelling Information Systems Architectures Vol 8, No 1, Viena, Austria, 2013.
4. MEGA. Enterprise architecture overview. Estados Unidos, 2014.
5. The Open group. TOGAF 9.1: Architectural artifacts by ADM phase, Estados Unidos, 2011.
6. Kolovos D., EuGENia: GMF for mortals. Nueva York, Estados Unidos, 2014.
7. Melo I., Sánchez M., Villalobos J. Composing Graphical Representations of Composed Models. In: International Workshop on The Globalization of Domain Specific Languages (GlobalDSL) at European Conference on Object-Oriented Programming (ECOOP2013). Montpellier, Francia, 2013.
8. Genova G. What is a metamodel: the OMG’s metamodeling infrastructure. In: Modeling and metamodeling in model driven development. Varsovia, Polonia, 2009.
9. Gašević D., Djurić D, Devedzic V. Model driven engineering. In: Model driven engineering and ontology development. Nueva York, Estados Unidos, 2009.
10. Schmidt D. Model driven engineering. In: Computer Vanderbilt University. Nashville Tennessee, Estados Unidos, 2006.
11. Eclipse Foundation, Eclipse Modeling Framework (EMF). Ottawa, Ontario, Canadá, 2014.
12. Eclipse Foundation, Textual Modeling Framework (TMF). Ottawa, Ontario, Canadá, 2014.
13. Efftinge S, Völter M. oAW XText: A framework for textual DSLs. Alemania, 2006.
14. Melo I. EnAr-Picture, Bogotá, Colombia, 2013.

# Agradecimientos

1. Adaptado de Genova G. What is a metamodel: the OMG’s metamodeling infrastructure. In: Modeling and metamodeling in model driven development, Varsovia, Polonia, 2009 [↑](#footnote-ref-1)
2. Adaptado de Gašević D., Djurić D, Devedzic V. Model driven engineering. In: Model driven engineering and ontology development capitulo 4. [↑](#footnote-ref-2)
3. What is JDBC: http://publib.boulder.ibm.com/infocenter/rbhelp/v6r3/index.jsp?topic=%2Fcom.ibm.redbrick.doc6.3%2Fciacg%2Fciacg33.htm [↑](#footnote-ref-3)