PA1710-3-Gengular

Gengular: Hacia la automatización de aplicaciones empresariales bajo el paradigma de arquitectura SPA y el enfoque MDE

DANIEL RAMIREZ ECHEVERRI

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERÍA
MAESTRÍA EN INGENIERÍA DE SISTEMAS Y COMPUTACIÓN
BOGOTÁ, D.C.
2017

PA1710-3-Gengular

Gengular: Hacia la automatización de aplicaciones empresariales bajo el paradigma de arquitectura SPA y el enfoque MDE

Autor:

Daniel Ramírez Echeverri

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

MAGÍSTER EN INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

Director

Ing. María Consuelo Franky de Toro Ph.D

Comité de Evaluación del Trabajo de Grado

Ing. Jaime Andrés Pavlich Mariscal Ph.D

Ing. María Catalina Acero Rozo M.Sc

Página web del Trabajo de Grado

http://pegasus.javeriana.edu.co/~PA1710-3-Gengular

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERÍA
MAESTRÍA EN INGENIERÍA DE SISTEMAS Y COMPUTACIÓN
BOGOTÁ, D.C.
Junio, 2017

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERÍA
MAESTRÍA EN INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

Rector Magnífico

Joaquín Emilio Sánchez García S.J.

Decano Académico Facultad de Ingeniería

Ingeniero Jorge Luis Sánchez Téllez

Decano del Medio Universitario Facultad de Ingeniería

Padre Antonio José Sarmiento Nova, S.J.

Director Maestría en Ingeniería de Sistemas y Computación

Ingeniera Ángela Cristina Carrillo Ramos

Director Departamento de Ingeniería de Sistemas

Ingeniero Efraín Ortiz Pabón

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

Agradezco a Dios, por su bondad y misericordia.

A mis padres, Carlos y Martha, por su incondicional amor y apoyo en todo lo que emprendo.

A mi hermano Sebastián, por sus invaluables consejos y su compañía.

Un agradecimiento especial a la profesora María Consuelo Franky PhD, por su dedicación, compromiso e intereses durante todo el proyecto, pero sobre todo por sus enseñanzas a lo largo de la maestría.

También quiero agradecer a la empresa Heinsohn Business Technology, por permitirme realizar las pruebas con sus componentes empresariales.

Daniel Ramirez Echeverri

Contenido

Lista de tablas

Lista de figuras

ABSTRACT

Companies that develop software projects need to automate the construction of applications in order to reduce costs and implementation times; However, the process to reuse and integrate their business technology assets into new projects is complex and they also face the problem of preserving their information systems from the constant evolution of platforms and technologies. In the present work, an alternative solution is proposed to these two problematics applying the approach of the Model-Driven Engineering (MDE), from models in the language ISML that are independent of any technology and that allow to generate web applications modern models that follow the SPA architecture paradigm, using a code transformer called Gengular. Additionally, in this work were modeled business components that are preserved in time because they are independent of any technology, and at the same time are easy to couple to SPA applications using the tools built into the project.

RESUMEN

Las empresas que desarrollan proyectos de software requieren automatizar la construcción de aplicaciones con el fin de reducir los costos y tiempos de implementación; sin embargo, el proceso para reutilizar e integrar sus activos tecnológicos empresariales a nuevos proyectos es complejo y además enfrentan la problemática de preservar en el tiempo sus sistemas de información frente a la constante evolución de plataformas y tecnologías. En el presente trabajo de grado, se propone una alternativa de solución a estas dos problemáticas aplicando el enfoque de la ingeniería dirigida por modelos (MDE), a partir de modelos en el lenguaje ISML que son independientes de cualquier tecnología y que permiten generar aplicaciones web modernas que siguen el paradigma de arquitectura SPA, mediante un transformador de código denominado Gengular. Adicionalmente en este trabajo se modelaron componentes empresariales que se preservan en el tiempo por ser independientes de cualquier tecnología, y que al mismo tiempo son fáciles de acoplar a aplicaciones SPA mediante las herramientas construidas en el proyecto.

.

RESUMEN EJECUTIVO

Durante los años 2012-2015 la Pontificia Universidad Javeriana y la empresa Heinsohn Business Technology (HBT) desarrollaron los proyectos Lion y Lion2, los cuales lograron incrementar la productividad en la organización mediante la automatización en el desarrollo de software aplicando ingeniería dirigida por modelos (MDE). Con ese proyecto, se construyó el lenguaje de modelado ISML (Information Systems Modeling Language) que al ser independiente de la plataforma permite crear aplicaciones para distintas tecnologías utilizando un conjunto de transformadores de código modulares. Sin embargo, en la actualidad estos transformadores se encuentran desactualizados a causa de la disrupción tecnológica en frameworks de capa de presentación donde tecnologías maduras y estables como JSF, que llevaban más de 10 años en el mercado, son reemplazados por el stack tecnológico HTML5-AngularJS-JEE7 impulsado por Google como el nuevo estilo de construcción de aplicaciones al promover la escalabilidad, desempeño, usabilidad y despliegue en la nube.

En el presente proyecto se construyó un transformador de código, que a partir de modelos definidos en lenguaje ISML, permite crear aplicaciones alineadas con el stack tecnológico HTML5-AngularJS-JEE7 y el paradigma de arquitectura de aplicaciones web SPA (Single-Page Application). Este paradigma ofrece una experiencia de usuario similar al de una aplicación nativa, mejorando sustancialmente el rendimiento de aplicaciones web mediante técnicas de diseño de cargue parcial de páginas, que además son compatibles tanto para PC como para dispositivos móviles.

El proyecto contribuyó adicionalmente con la recuperación de componentes que son valiosos para cualquier empresa y que para HBT (tomado como caso de estudio), representan activos tecnológicos y un medio de automatización en el desarrollo de software mediante la reutilización en los proyectos de componentes útiles de seguridad, manejo de menús, auditoría, notificaciones, procesamiento de archivos, etc.

En el proyecto se realizó en primer lugar, la especificación de las reglas de transformación asociadas a cada elemento de la arquitectura de referencia, posteriormente se efectuó la codificación e implementación de los artefactos de software. Finalmente, se validó el funcionamiento del transformador modelando componentes empresariales usuales como menús y notificaciones en lenguaje ISML, que luego fueron transformados al stack de tecnologías HTML5-AngularJS-JEE7.

Con este proyecto, la empresa caso de estudio puede disponer de un ambiente de generación de aplicaciones web y móviles que funcionan con la eficiencia de las últimas tecnologías SPA y adicionalmente, de modelos de componentes empresariales que se transforman en código fuente de esa misma tecnología para luego acoplarse a una aplicación previamente generada. Los modelos de componentes son estables especialmente en su lógica de negocio y en el futuro no será difícil construir nuevos transformadores para las nuevas tecnologías de la capa de presentación.

INTRODUCCIÓN

La industrialización del software plantea que la automatización es el mecanismo para producir software de alta calidad en menor tiempo y con costos más bajos. No obstante la constante evolución de tecnologías y plataformas a la que se enfrenta la industria TI, así como el acelerado ritmo en que sus herramientas se vuelven obsoletas en el mercado, dificulta alcanzar los niveles de productividad esperados. A partir de esta problemática, la ingeniería dirigida por modelos (MDE) [10] surge como alternativa de solución para permitir a las organizaciones preservar sus activos tecnológicos y ser más agiles en sus procesos de desarrollo. Para ello, propone un enfoque donde el modelo es el artefacto central en el desarrollo de software y se apoya en generadores y transformadores de código para convertir dichos modelos en código para una plataforma específica o incluso para otros tipos de modelos.

Con base en el enfoque MDE, en este trabajo de grado se construyó un generador de código denominado Gengular que permite crear aplicaciones de última generación que funcionan en cualquier dispositivo cliente incluidos celulares, a partir de modelos definidos en un lenguaje independiente de plataforma llamado ISML (Information Systems Modeling Language) [17], de manera que los activos tecnológicos típicos de una compañía de TI, como componentes empresariales, pueden ser modelados con un grado mayor de abstracción y de esa manera ser conservados a lo largo del tiempo.

El generador de código desarrollado, permite construir aplicaciones basadas en arquitecturas SPA (Single-Page Applications) [35], un paradigma orientado a mejorar el rendimiento de aplicaciones web y la experiencia de usuario, apoyado en tecnologías de presentación como HTML5 [22], última versión del estándar HTML establecido por el consorcio W3C [5], y framework de JavaScript AngularJS [20] desarrollado por Google Inc. [6], Además utiliza Java Empresarial en su versión siete [27], para soportar la lógica de negocio y en general los artefactos de *backend* en las aplicaciones SPA.

En la parte I del documento, se hace una presentación general del trabajo de grado, la problemática u oportunidad de mejora identificada, se formulan los objetivos, se describe la metodología utilizada y el potencial de innovación del proyecto.

En la parte II, se presenta el marco contextual del trabajo de grado, exponiendo el contexto de la industrialización de software en Colombia y la aplicación del enfoque MDE en la industria TI.

En la parte III, se muestra los referentes teóricos y conceptuales tenidos en cuenta en el desarrollo del trabajo de grado.

En la parte IV, se presenta las definiciones, artefactos y actividades relevantes realizadas en el desarrollo del proyecto. En primer lugar, la comparación y selección de tecnologías SPA, los requerimientos y funcionalidades que guiaron la construcción del transformador Gengular, el diseño en términos de reglas de transformación y arquitectura del mismo, para luego mostrar la validación y los resultados obtenidos.

En la parte V, se muestran las conclusiones donde se sintetiza y analiza el trabajo realizado, también se presentan los trabajos futuros. Finalmente, en la parte VI se muestra la bibliografía consultada, es decir cada una de las referencias utilizadas en el presente trabajo de grado.

I -PRESENTACIÓN DEL TRABAJO DE GRADO

En este capítulo se describe el problema de aplicación, los objetivos del proyecto y el potencial de innovación del transformador.

1. Problemática u oportunidad de mejora.

El desarrollo de software tradicional supone un conocimiento profundo de las tecnologías y lenguajes por parte de las organizaciones y sus equipos de trabajo. Sin embargo, las tecnologías son cada vez más cambiantes y evolucionan a un ritmo rápido ocasionando que las empresas se centren en aprender los detalles de plataformas y lenguajes en lugar de enfocarse en solucionar los problemas de negocio, que finalmente es lo que da un valor agregado e impulsa la iniciación de proyectos de software.

La ingeniería dirigida por modelos (MDE) surge como alternativa para hacer frente a esta problemática y a la incapacidad de los lenguajes de tercera generación de manejar la complejidad de las plataformas [10], mediante un enfoque donde el modelo es el artefacto central en el desarrollo de software permitiendo así expresar los conceptos de dominio de manera efectiva a través de generalización y abstracción.

Como estrategia para abordar la obsolescencia de tecnologías y plataformas, la Pontificia Universidad Javeriana y la empresa caso de estudio Heinsohn [1] desarrollaron los proyectos Lion y Lion2 [2] siguiendo un enfoque basado en MDE, en el cual se produjeron el ambiente y lenguaje de modelado ISML con transformadores hacia la tecnología vigente en el momento de los proyectos. Actualmente se tiene la problemática de que el ambiente quedó obsoleto con la llegada de tecnologías más eficientes como las SPA, dado que no se cuenta con transformadores que permitan generar aplicaciones en esas nuevas tecnologías a partir de los modelos ISML.

Por otra parte, la problemática asociada al constante cambio de tecnologías también afecta la preservación de los componentes empresariales de Heinsohn, debido a que se deben rehacer cada vez que se adopta una nueva arquitectura de referencia. En este proyecto se resuelve el problema conservando los componentes empresariales en forma de modelos ISML y construyendo el transformador para la tecnología deseada.

2. Objetivos

A partir de las problemáticas identificadas en este proyecto, se definió un objetivo general y seis objetivos específicos.

2.1. Objetivo General

Diseñar e implementar un transformador ISML denominado Gengular para el stack de tecnologías HTML5-AngularJS-JEE7 que facilite la creación de aplicaciones web bajo el paradigma SPA, contribuyendo a lograr una mayor reutilización y usabilidad en los productos de software.

2.2. Objetivos Específicos

1. Especificar los requerimientos del transformador.

2. Diseñar el transformador en términos de sus reglas de transformación.

3. Diseñar una estrategia de cómo expresar el modelo de un componente empresarial y cómo realizar su acople a una aplicación web generada con el transformador, reutilizando la lógica de negocio del componente.

4. Construir e implementar los artefactos de software que conforman el transformador.

5. Modelar dos componentes empresariales en el lenguaje ISML, tomando como caso de estudio la empresa HBT.

6. Validar el transformador utilizando un modelo ISML de una aplicación web sencilla más los modelos ISML de dos componentes empresariales, generándolos para el stack de tecnologías HTML5-AngularJS-JEE7 y realizando su acople.

3. Metodología

La construcción del transformador Gengular, se dividió en 6 fases: Estudio de las tecnologías relevantes para el paradigma SPA, especificación de requerimientos, diseño del transformador, implementación del transformador, modelado de componentes empresariales, pruebas y refinamiento.

Como estrategia para la implementación del transformador ISML, se utilizó la metodología de desarrollo de software Extreme Programming – XP formulada por Kent Beck [14], cuyo

enfoque ágil permitió la adaptación a los cambios y la producción de software de manera iterativa e incremental. A continuación, se presentan cada una de las fases desarrolladas.

Fase 1. Estudio de las tecnologías relevantes para el paradigma SPA: En esta fase se realizó la revisión y selección de bibliografía existente, así como de los trabajos de investigación relacionados con el proyecto. Dentro de esta fase se realizaron las siguientes actividades:

a. Revisión de material bibliográfico más significativo para el stack de tecnologías HTML5-AngularJS-JEE7 y el paradigma de aplicaciones SPA.
b. Análisis y organización de la bibliografía.
c. Realización de resumen de bibliografía significativa.

Fase 2. Especificación de requerimientos: Durante esta fase se estableció los requerimientos del sistema través de historias de usuario [15], para ello se identificó las principales funcionalidades y características del transformador. Las siguientes actividades fueron ejecutadas:

a. Definición de requerimientos de negocio.
b. Definición de requerimientos técnicos y restricciones.
c. Validación de requerimientos con el director del proyecto.
d. Priorización y selección de requerimientos a desarrollar.

Fase 3. Diseño del transformador: La fase 3 abordó tanto el diseño de la arquitectura de solución como el diseño detallado. Para ello se realizaron las siguientes actividades:

a. Definición de arquitectura de la aplicación web objetivo en consonancia con una arquitectura de referencia definida por la empresa caso de estudio.
b. Especificación de una aplicación web típica en ISML [17].
c. Especificación de las reglas de transformación para HTML5, AngularJS y JEE7.

Fase 4. Implementación del transformador: Durante esta fase se abordó la construcción de los componentes que conforman el transformador, a partir del diseño definido en la fase anterior. Las actividades realizadas fueron:

a. Definición de lineamientos y buenas prácticas para la construcción de software en las tecnologías seleccionadas.
b. Implementación de plantillas de generación de código.
c. Depuración y validación de código mediante pruebas.

Fase 5. Modelado de componentes empresariales: Esta fase implicó modelar dos componentes empresariales de la empresa HBT en ISML, lo cual permitió representar su funcionalidad independiente de cualquier tecnología. Los modelos se utilizaron como insumo para la generación de código fuente en las tecnologías destino HTML5-AngularJS-JEE7 por medio del transformador Gengular. Las actividades que incluyó la fase fueron:

a. Estrategia de modelado de componentes y acople a una aplicación web generada con el transformador mediante Maven [16] para resolver las dependencias de la aplicación respecto a los componentes.
b. Modelado de un componente de manejo de menús en ISML.
c. Modelado de un componente de notificaciones en ISML.
d. Transformación de los 2 componentes con acople a una aplicación web generada previamente.

Fase 6. Pruebas y refinamiento: La fase de pruebas y refinamiento permitió garantizar el correcto funcionamiento del transformador y la calidad de los artefactos generados. Para ello se realizaron durante la fase las siguientes actividades:

a. Diseño del plan de pruebas, teniendo en cuenta métricas de líneas de código (LOC) para validación de grado de reutilización en la aplicación y verificación de compatibilidad en múltiples dispositivos (PC, iPad, Smartphone) para promover usabilidad.
b. Ejecución del plan de pruebas.
c. Realizar ajustes y refinamiento a partir de hallazgos.
d. Encuesta de satisfacción a usuarios del transformador.

4. Potencial de Innovación

El potencial de innovación del presente proyecto es permitir la generación de aplicaciones web modernas bajo el paradigma SPA y al mismo tiempo la reutilización de componentes empresariales que a partir de modelos generan el código correspondiente para acoplarse a dichas aplicaciones, contribuyendo a que las empresas de tecnología sean más agiles en la construcción de sus productos mediante un transformador que les permita generar aplicaciones alineadas con el stack de tecnologías HTML5-AngularJS-JEE7. Adicionalmente, las aplicaciones generadas funcionan tanto en un navegador de PC como en dispositivos móviles.

II -MARCO CONTEXTUAL

Muchas empresas colombianas que desarrollan proyectos de software no reutilizan componentes ni automatizan parcialmente sus soluciones mediante generadores de código y MDE. Según el informe de caracterización del sector de software y tecnologías de la información en Colombia de Fedesoft, presentado en diciembre de 2015, la baja inversión en actividades de investigación, desarrollo e innovación no permite que se genere un impacto significativo en los niveles de productividad y competitividad, claves en el desarrollo de la industria TI del país [56].

De acuerdo a los indicadores presentados en 2015 por MinTIC (Ministerio de Tecnologías de la Información y las Comunicaciones de Colombia) y que corresponde a la inversión en I+D por parte de las empresas de software colombianas [57], el 91% invirtió menos de 294 millones para esta actividad como se muestra en la figura 1. Si además de este hecho, se tiene en cuenta que el área de Ingeniería de Sistemas es la que tiene el menor número de egresados de doctorados en el país (solamente 10 graduados en el 2013 frente a 139 graduados en ingeniería sin ningún énfasis específico [58]), se evidencia la escasez de mano de obra altamente calificada e idónea para realizar actividades de investigación e innovación.

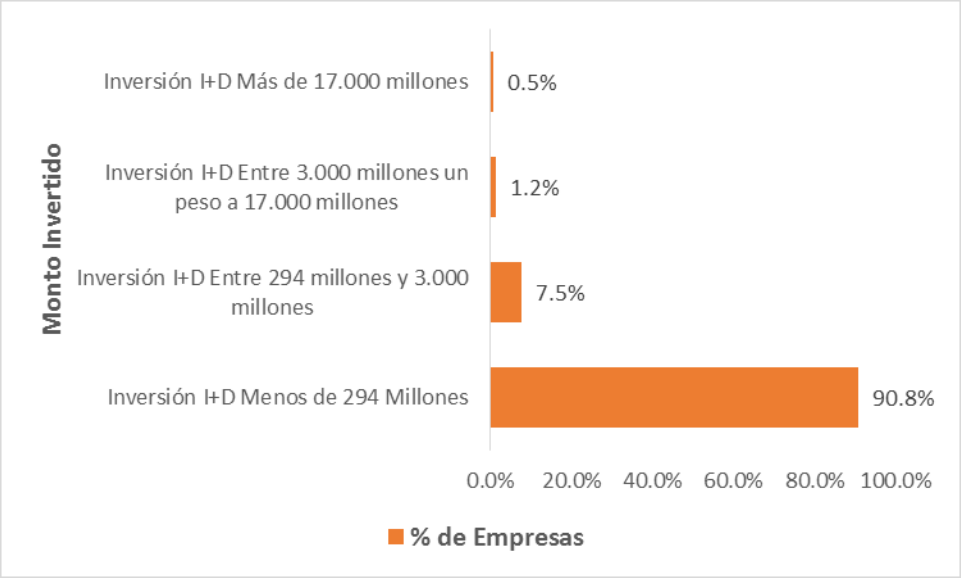

Figura 1: Inversión en I+D por parte de las empresas de software colombianas. Tomado de MinTIC [57]

Como consecuencia de la insuficiente inversión en I+D de la gran mayoría de empresas del sector TI en Colombia, y teniendo en cuenta que el gobierno nacional invirtió el 0.5% del PIB para actividades de ciencia, tecnología e innovación (ACTI) y la inversión únicamente en Investigación y Desarrollo (I+D) fue de 0,224% para el 2013 [59], se hace notorio los bajos niveles en la industrialización del software que se han alcanzado en el país.

Dentro de este contexto, la industrialización del software es un aspecto fundamental para el desarrollo y evolución del sector TI en Colombia, ya que determina la capacidad para producir software de mayor calidad, en menor tiempo y con costos más bajos, que pueda ser competitivo en mercados internacionales.

Los países que han alcanzado un alto grado de industrialización del software, como Estados Unidos, China, Rusia, India, Israel, Irlanda y Brasil, también han logrado incrementar sus ventas de software y posicionarse como exportadores importantes de servicios de TI en la economía internacional, con procesos optimizados y de mejora continua que permiten medir y controlar dichas actividades [60].

El eje central del presente trabajo de grado se enmarca en la automatización y reutilización del software, asumiendo que estos elementos son fundamentales para lograr que países en vía de desarrollo sean más competitivos dentro de la industria TI. Además, toma la propuesta de la Ingeniería Dirigida por Modelos (MDE) [10] como paradigma para alcanzar este propósito.

La Pontificia Universidad Javeriana es una de las universidades que se ha interesado en investigar el enfoque MDE para el desarrollo de software, adelantando trabajos de grado sobre esta temática en los niveles de pregrado, maestría y doctorado. Asimismo, imparte cursos sobre MDE y frameworks de generación. A continuación, se presentan algunos de los trabajos que han sido realizados y están en curso:

- **MIDAS:** Framework basado en el enfoque MDE para la construcción de sistemas adaptativos, el cual permite expresar los requerimientos para una aplicación en el lenguaje RSLAS (Requirements Specification Language for Adaptive Systems), que luego son convertidos a DMLAS (Design Modeling Language for Adaptive Systems) y posteriormente convertidos a ISML (Information Systems Modeling Language), para finalmente generar código automáticamente en alguna tecnología haciendo uso de transformadores [61].

- **LITYERSES:** Transformador para generar aplicaciones móviles nativas para el sistema operativo Android con componentes adaptativos a partir de un modelo independiente de plataforma ISML [62].

- **ANCHURUS-GEN:** Generador de código PHP bajo el marco de trabajo de Laravel, a partir de modelos ISML [63].

- **ZOE-GEN:** Transformador para generar aplicaciones en la plataforma Java EE bajo el marco de trabajo Primefaces, a partir de modelos ISML [64].

- **WAPP GENERATOR:** Transformador para Java EE 6 con páginas JSF utilizando plantillas Acceleo [65].

Por otra parte, en cuanto al sector empresarial Heinsohn Business Technology es una de las compañías colombianas que se destaca por desarrollar actividades que les permiten ser más ágiles y competitivas, innovando y automatizando los procesos de TI que ayudan a generar valor y disminuir costos a sus clientes. Para ello, Heinsohn ha adelantado proyectos universidad-empresa con cofinanciación de Colciencias en instituciones de educación superior como la Universidad Javeriana [2] y la Universidad de los Andes [66].

Por las razones presentadas anteriormente, en este trabajo de grado la empresa Heinsohn se toma como caso de estudio para los siguientes dos propósitos: En primer lugar, realizar la verificación y alineación de las tecnologías destino a utilizar por el transformador Gengular frente a las empleadas actualmente en una compañía de tecnología de talla mundial y valorada en el máximo nivel de CMMI [67]. En segundo lugar, la reutilización de componentes empresariales mediante la estrategia de modelos ISML se hace con base a los componentes que posee Heinsohn.

A continuación, se describe la empresa caso de estudio:

Heinsohn Business Technology ofrece soluciones que cubren el 100% de la cadena de valor en TI. Cuenta con un amplio conocimiento en desarrollo de software para todos los sectores de la industria que le ha permitido trabajar con las principales empresas de América Latina y EE.UU. Con casa matriz en Bogotá y sedes en las siete principales ciudades de Colombia; sucursales en América Latina y los Estados Unidos para brindarle mayor cobertura a sus clientes.

Esta compañía tiene el respaldo y aval de los más exigentes estándares de calidad como se muestra en la figura 2. Es el caso de las certificación de calidad: ISO 9001 Versión 2008, y de la valoración CMMI Nivel 5, para la adopción de mejores prácticas en Ingeniería de Software.

Figura 2: Descripción de la empresa caso de estudio, Heinsohn Business Technology.

III -MARCO TEÓRICO

En esta sección se describen los referentes teóricos y conceptuales tenidos en cuenta para el desarrollo de este trabajo.

1. Ingeniería Dirigida por Modelos (MDE)

La ingeniería dirigida por modelos (MDE) es un enfoque para el desarrollo de software, el cual tiene como objetivo permitir a las empresas proteger sus inversiones en software frente a la constante evolución de la tecnología [37]. Dado que las plataformas, lenguajes de programación y herramientas de desarrollo tienden a volverse obsoletas más rápidamente mientras que aumenta la complejidad de las mismas, la ingeniería dirigida por modelos surge como alternativa para hacer frente a esta problemática y a la incapacidad de los lenguajes de tercera generación de manejar la complejidad de las plataformas [10], mediante un enfoque donde el modelo es el artefacto central en el desarrollo de software permitiendo así expresar los conceptos de dominio de manera efectiva a través de generalización y abstracción.

Bajo esta visión para el desarrollo de sistemas, la metodología en MDE propone dos dimensiones fundamentales: conceptualización e implementación; la primera se ocupa de la definición de modelos conceptuales como medio para representar o describir la realidad, y la segunda dimensión se encarga de la transformación de los modelos a algún sistema de software que pueda ser ejecutable [11].

La dimensión de conceptualización está constituida por tres niveles principales: el nivel de aplicación donde los modelos, reglas de transformación y componentes ejecutables son generados; el nivel de dominio de aplicación donde son definidos los lenguajes de modelado y plataformas de implementación para el dominio específico, y el meta-nivel donde es definido conceptualmente los modelos y transformaciones (ver figura 3).

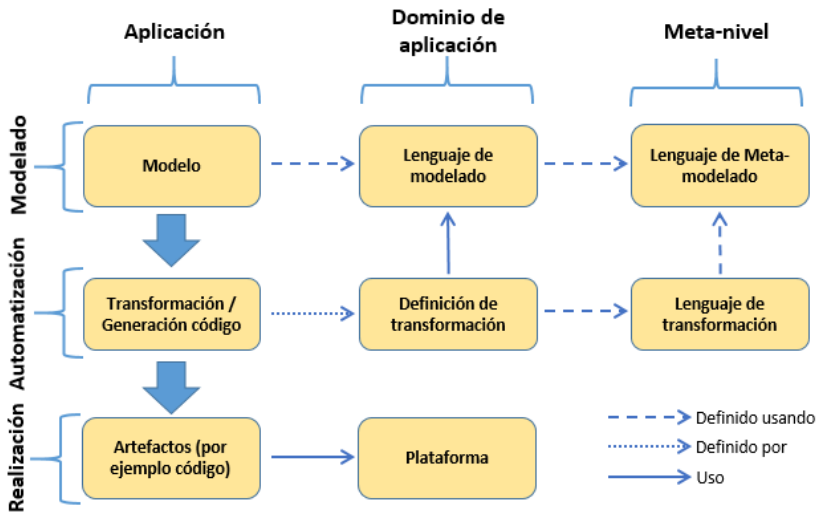

Figura 3: Metodología MDE. Adaptado de Brambilla, Cabot y Wimmer [11].

De la misma forma, la dimensión de implementación comprende tres niveles claves: el nivel de modelado donde evidentemente son definidos los modelos, el nivel de realización en el cual son implementadas las soluciones de software a través de artefactos que corren dentro de los sistemas ejecutables, y el nivel de automatización donde las transformaciones de los modelos al nivel de realización se ponen en marcha [11].

La arquitectura dirigida por modelos (MDA) es un estándar para MDE definido en el año 2001 por el Open Management Group (OMG) [12], con el objetivo de lograr portabilidad, interoperabilidad y reutilización a través del desarrollo basado en modelos que posteriormente son generados para múltiples plataformas [38].

MDA conduce hacia un entendimiento de los modelos como activos tecnológicos organizacionales, que perduran y evolucionan en el tiempo al ser independientes de implementaciones. Para lograr esta independencia de plataforma, MDA define varios modelos expresados en determinados lenguajes y con un grado de abstracción diferente, que son transformados entre sí mediante funciones de transformación [13]. Estos modelos son el CIM (Computation-Independent Model) el más abstracto utilizado para expresar requerimientos, PIM (Platform-Independent Model) que describe el comportamiento y estructura de la aplicación sin importar la plataforma de implementación, y el PSM (Platform-Specific Model) que contiene la información específica o de detalle de la plataforma.

En la figura 4 se presenta el ciclo de vida de desarrollo de software en MDA, que en gran medida es similar al ciclo de vida tradicional en sus fases, sin embargo, la mayor diferencia radica en los artefactos que son creados durante el proceso de desarrollo.

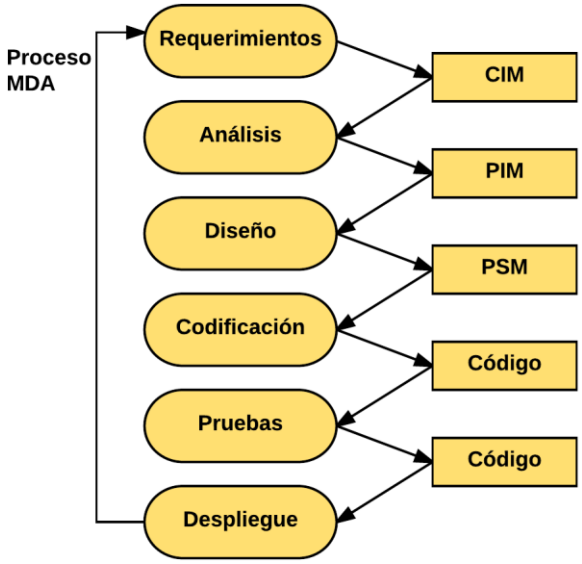

Figura 4: Ciclo de vida de desarrollo MDA. Adaptado de Kleppe, Warmer y Bast [39].

El ciclo de vida de desarrollo de software propuesto en MDA plantea dos importantes beneficios, por una parte, maximiza la productividad y portabilidad al permitir que el diseñador o desarrollador de aplicaciones trabaje independientemente de detalles y especificaciones de

las plataformas destino, ya que estos detalles técnicos serán automáticamente adicionados por las transformaciones PIM a PSM. Por otra parte, MDA contribuye a direccionar el foco del desarrollo de software en los modelos y por ende en resolver los problemas de negocio y necesidades de los usuarios [39].

2. Maven

Apache Maven es una herramienta de gestión de proyectos de software, enfocada en la automatización de tareas de construcción en el ciclo de vida del software. Maven se encarga tanto de aspectos de construcción (compilación, empaquetamiento, instalación y despliegue), como de la gestión de dependencias del proyecto, aquellos módulos o librerías que son requeridas por el mismo [40].

Liberado en el año 2002 por Jason van Zyl [41], Maven se enmarca dentro de una arquitectura basada en plugins, la cual provee un conjunto de funcionalidades *core* que pueden ser extendidas a través de un conjunto de plugins descargables desde diferentes repositorios [42].

Cada proyecto en Maven utiliza un archivo *XML* llamado POM (*Project Object Model*), donde son definidas las características del proyecto y son asignadas un conjunto de coordenadas que permiten identificarlo de manera única. Contar con un modelo en cada proyecto implica importantes ventajas, debido a que toda la configuración y personalización ocurre en un nivel de abstracción elevado, que además facilita la integración y portabilidad entre entornos de desarrollo integrado (IDE), al permitir estandarizar el trabajo mediante un lenguaje común [16].

Maven también introduce el concepto de arquetipo, como un patrón o modelo original sobre el cual pueden definirse nuevos proyectos, definiendo la estructura, esqueleto o plantilla que se utiliza de base para la creación de nuevos proyectos. El uso de arquetipos Maven contribuye a generar proyectos consistentes, configurados correctamente y que responden a las practicas empleadas por una organización o grupo, reduciendo los tiempos invertidos en definir la estructura de directorios de un proyecto, las dependencias del mismo, convenciones, nomenclatura, entre otros [43].

3. Single-Page Application (SPA)

Single-Page Application (SPA) es un paradigma de arquitectura de aplicaciones web cuyo propósito es llevar el poder de las aplicaciones de escritorio al entorno delgado y multiplataforma de un navegador web. SPA ofrece una experiencia de usuario similar al de una aplicación nativa, mejorando sustancialmente el rendimiento de aplicaciones web mediante técnicas de diseño de cargue parcial de páginas, que además son compatibles tanto para PC como para dispositivos móviles [9].

SPA es el resultado de múltiples esfuerzos previos en la industria de alcanzar el aspecto y desempeño que poseen las aplicaciones de escritorio; aunque el termino SPA fue utilizado por primera vez en el año 2005 por Steve Yen [50], tecnologías como los Applets de Java [51], Adobe Flash [52] y Microsoft Silverlight [53] representan el esfuerzo de importantes compañías de TI para lograr este objetivo.

A pesar que las soluciones mencionadas anteriormente lograron en cierta medida aplicaciones web que se asemejan a las nativas, el SPA basado en JavaScript ofrece importantes ventajas frente a éstas que lo convierten en la alternativa más popular y de mayor difusión, ya que no requiere de plugins para su ejecución que deban ser instalados, ni lenguajes o plataformas complementarias [35]. A continuación, se presenta la arquitectura de una aplicación SPA:

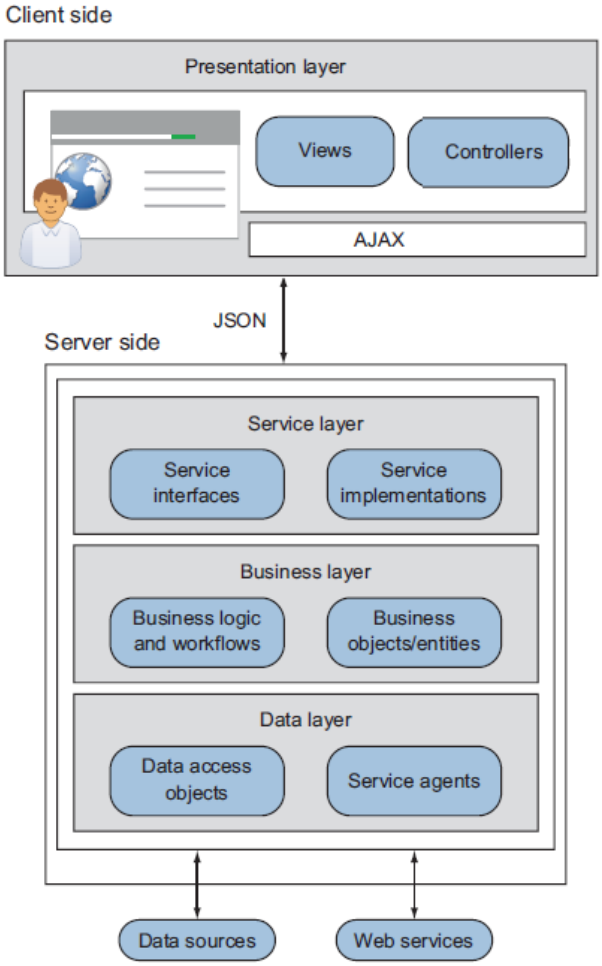

Figura 5: Arquitectura de una aplicación SPA. Adaptado de Scott [35].

Como se observa, la capa de presentación (vistas y controladores) se ejecutan en el lado del cliente y las transacciones que ocurren vía AJAX [54] no implican recargar el navegador. Asimismo, en esta arquitectura la comunicación con el lado del servidor se da para el envío/recepción de información mediante formatos ligeros (principalmente JSON [55]), pero el procesamiento y renderización de la información pasa a ser responsabilidad del cliente en lugar del servidor.

3.1. AngularJS

AngularJS es un framework MVC [33] escrito en JavaScript [34], de propósito general, para la construcción de la capa de presentación en aplicaciones web tipo SPA Single-Page Applications [35]. Soportado y licenciado por Google Inc, AngularJS se destaca por su innovador sistema de plantillas, facilidad de desarrollo y por estar construido bajo prácticas de ingeniería muy sólidas [19].

Aunque la versión 1.0 fue liberada en junio de 2012, el inicio de este framework empezó en 2009 como un proyecto personal de Miško Hevery [36], líder técnico y manager en Google. El auge y popularidad que ha alcanzado AngularJS en los últimos años se debe en gran medida a su capacidad de producir automáticamente el enlace de datos de dos sentidos (*Two-way Data Binding*), el cual se encarga de actualizar la vista cuando el modelo cambia y viceversa, facilitando el esfuerzo en implementación al ahorrar la escritura de cientos de líneas de código que son requeridas cuando se utiliza frameworks como jQuery [20].

Adicionalmente, AngularJS ofrece importantes ventajas frente a JSF (Framework de presentación del estándar Java EE 7) [4], debido a que el controlador y la vista se ejecutan en el navegador web aprovechando los recursos computacionales de las máquinas del lado cliente, de manera que se facilita la escalabilidad y rendimiento de las aplicaciones [8]. En la figura 6 se muestra la arquitectura en que se basa AngularJS:

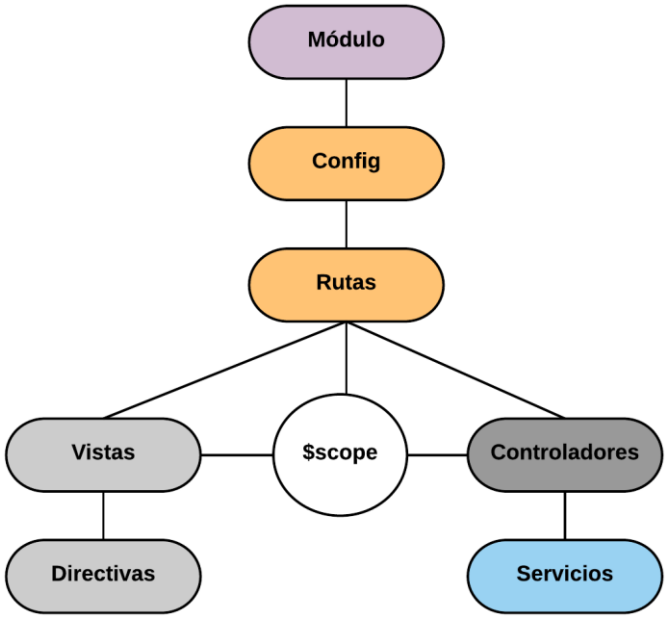

Figura 6: Arquitectura de AngularJS. Adaptado de Ruebbelke y Ford [20].

Cada uno de los componentes que conforman la arquitectura de AngularJS tiene un propósito específico que es detallado en la tabla 1.

Componente	Propósito
Módulos	Los módulos sirven como contenedores para ayudar a organizar el código dentro de la aplicación. Estos a su vez pueden contener sub-módulos, lo que facilita la composición de la funcionalidad de capa de presentación según sea necesario.
Rutas	Las rutas permiten establecer los flujos de navegación de páginas dentro de la aplicación. También permiten definir opciones de configuración para cada ruta, como la plantilla o controlador a utilizar.
Directivas	Una directiva es una extensión de una vista en AngularJS que permite crear elementos personalizados, reutilizables y que encapsulan el comportamiento. Las directivas se utilizan para ampliar las capacidades de las vistas y para hacer que estas extensiones estén disponibles para su uso en varios lugares de la aplicación.
Vistas	Las vistas en AngularJS son el resultado de compilar y procesar el DOM (Document Object Model) con todo el código JavaScript asociado.
$scope	El *scope* es un objeto de AngularJS que sirve como "pegamento" o enlace entre la vista y el controlador. El scope opera como un contexto de variables de ejecución, encargado de contener, transformar y hacer visible la información entre la vista y el controlador.
Controladores	Un controlador es el responsable de definir las operaciones y propiedades con los que una vista puede interactuar.
Servicios	Los servicios proveen funcionalidades que son comunes en toda la aplicación. Permiten extender los controladores y hacen estos más accesibles de forma global.

Tabla 1: Componentes de AngularJS. Adaptado de Ruebbelke y Ford [20].

3.2. HTML5

HTML5 es la versión más reciente de HTML (*HyperText Markup Language*), lenguaje de marcado para la creación de páginas Web. HTML5 surge como el primer intento de documentar formalmente muchos de los estándares de facto que los navegadores han soportado durante años, además incorpora nuevos elementos semánticos y multimedia que permiten estructurar de manera más adecuada los contenidos web [21].

Publicado oficialmente en el año 2014, la quinta versión del HTML está basado en HTML 4.01 Strict, pero a diferencia de su predecesor, HTML5 no utiliza el DTD (*Document Type Definition*); en su lugar hace uso del DOM (*Document Object Model*) para representar y estructurar los documentos [22].

HTML5, además de facilitar en gran medida el acceso y gestión de los elementos en los documentos mediante el DOM, define un nuevo conjunto de funcionalidades para permitir una mejor validación del contenido en los formularios al incluir 13 tipos de entradas y controles adicionales [23]. En la tabla 2 se muestran estas nuevas características.

Tipo de Entrada	Propósito
<input type="color">	Muestra un selector de colores, que envía un código hexadecimal para el color seleccionado.
<input type="date">	Muestra un selector de fechas, el cual envía la fecha seleccionada.
<input type="email">	Muestra un campo para introducir direcciones de correo electrónico.
<input type="number">	Muestra un campo restringido para introducir valores numéricos.
<input type="range">	Este tipo de entrada obliga al usuario a seleccionar números dentro de un rango. Normalmente, se mostrará como un deslizador (*slider*).
<input type="search">	Muestra un campo de búsqueda.
<input type="tel">	Muestra un campo que obliga al usuario a escribir un número de teléfono válido.
<input type="url">	Este tipo de entrada muestra un campo que obliga al usuario a escribir una URL válida.
<input type="time">	Muestra un selector de tiempo del tipo hora y minutos.
<input type="datetime">	Muestra un selector de fecha y hora.
<input type="datetime-local">	Muestra un selector de fecha y hora, que envía dichos valores sin zona horaria.
<input type="month">	Muestra un elemento que representa un control para seleccionar un mes en específico de un año.
<input type="week">	Muestra un control para selecciona una semana dentro de un año.

Tabla 2: Tipos de entrada y controles para formularios en HTML5.

Por otra parte, HTML5 también incluye un grupo significativo de APIs que contribuyen a estandarizar tareas que en versiones anteriores del lenguaje requerían de plugins adicionales o programación personalizada, en aspectos como manejo de video, audio, geolocalización, entre otros. En la tabla 3 se detallan las principales APIs definidas en la especificación de HTML5.

API	Propósito
Canvas API	HTML5 define el elemento canvas como un lienzo dependiente de la resolución que se puede utilizar para representar gráficos, imágenes de juegos u otros elementos visuales sobre la marcha. Un canvas es un rectángulo en una página web en la que se puede utilizar JavaScript para dibujar cualquier cosa [24].
Geolocation API	La API de geolocalización define una especificación para el uso de datos basados en la ubicación del usuario (longitud y latitud), con el fin de poder utilizarlos en las aplicaciones empresariales. Conocer dónde

	está el usuario puede ser útil para mostrar noticias y servicios relevantes a la posición del mismo [23].
Web Sockets API	Define una comunicación bidireccional entre un cliente y un servidor para la transmisión de datos en tiempo real, dando lugar a aplicaciones más ligeras, rápidas y eficientes. Esta API opera como el mecanismo para conectarse a un servidor local o remoto, enviar los datos y posteriormente cerrar dicha conexión [25].
Media API	Se encarga de la reproducción de archivos de video y audio con sincronización multimedia y subtítulos temporizados, junto a un grupo de eventos a gestionar como progreso, espera, cambio de duración, cambio de volumen, redimensión, entre otros [22].
File API	Da acceso a los archivos cargados desde una entrada de formulario. Permite mostrar vistas previas del archivo cargado y habilita la carga tipo *drap-and-drop* [22].
Web Workers API	Proporcionan una forma estándar para que los navegadores web ejecuten JavaScript en segundo plano, permitiendo generar múltiples subprocesos que se ejecutan al mismo tiempo [24].
Web Storage API	Provee un mecanismo para que los sitios web almacenen información en la computadora del cliente y la recuperen posteriormente. El concepto es similar a las cookies, pero están diseñados para soportar grandes cantidades de información: Las cookies son de tamaño limitado (4kb) y el navegador las envía al servidor web cada vez que solicita una página, aumentando los tiempos y consumos de ancho de banda, mientras que este nuevo mecanismo es mejor porque los datos son almacenados localmente, no es obligatorio enviarlos al servidor en cada petición y pueden almacenar hasta 5 Mb [24].
Session History API	Se encarga de exponer el historial del navegador del cliente con el fin de lograr una mejor experiencia de usuario mediante la personalización de sitios web [22].
Editing API	Permite manipular y editar el contenido de un elemento, facilitando la implementación de editores HTML del tipo WYSIWYG (*What You See Is What You Get*) [26].

Tabla 3: APIs más relevantes incorporadas en HTML5.

3.3. Bootstrap

Bootstrap es un framework de capa de presentación para crear sitios y aplicaciones web *responsive*, fue creado por Mark Otto [44] y Jacob Thornton [45] en el año 2011, quienes se desempeñaban como diseñadores en Twitter [46]. Surge por la necesidad de estandarizar todas las diferentes librerías y herramientas que utilizaban los ingenieros en la compañía.

Bootstrap incluye múltiples características que permiten la creación de diseños *responsive*: un método encargado de tomar todo el contenido existente en una página y optimizarlo para el dispositivo que lo está viendo [47]. A continuación, se presentan las principales ventajas que ofrece éste framework de código abierto:

- **Reusabilidad:** Incluye componentes preconstruidos, hojas de estilos CSS y plugins que pueden ser añadidos directamente en el código, lo que resulta en un desarrollo más rápido al reducir tiempos y esfuerzo de implementación. Asimismo, facilita el mantenimiento y organización del código [48].

- **Consistencia:** Las convenciones de nombres y semántica de clases en Bootstrap facilita la lectura del código y reduce la curva de aprendizaje para diseñadores o ingenieros. Además, es compatible con diferentes navegadores y ofrece un alto grado de uniformidad en el proceso de diseño [48].

- **Sistemas de rejillas flexibles:** Define un sistema de rejillas o retículas en el cual el diseño se realiza mediante filas y columnas que se adaptan al tamaño de la pantalla donde se visualiza. Por defecto el sistema utiliza 12 columnas para el contenedor de elementos como se muestra en la figura 7.

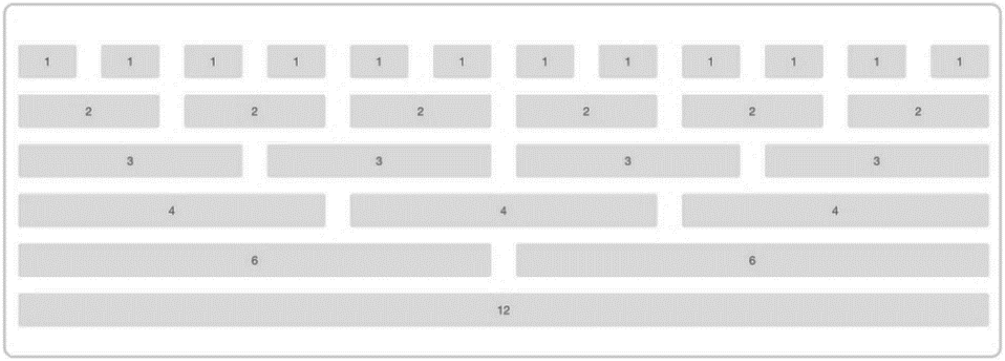

Figura 7: Sistema de Rejillas de Bootstrap. Tomado de Spurlock [47].

- **Personalización:** Al estar construido con bases muy sólidas de desarrollo, Bootstrap ofrece gran potencia y eficiencia que puede ser extendida o personalizada mediante LESS (Leaner CSS) [49].

4. Java Empresarial JEE7

Java en su edición empresarial (Java EE), proporciona una plataforma basada en estándares para el desarrollo de aplicaciones web y empresariales. Estas aplicaciones comúnmente son diseñadas bajo arquitecturas multicapa como se muestra en la figura 8, las cuales se encuentran constituidas por una capa de presentación (páginas web y *backing beans*), negocio (EJBs de sesión), y dominio (entidades respaldadas por una base de datos gracias a un framework de persistencia) [27].

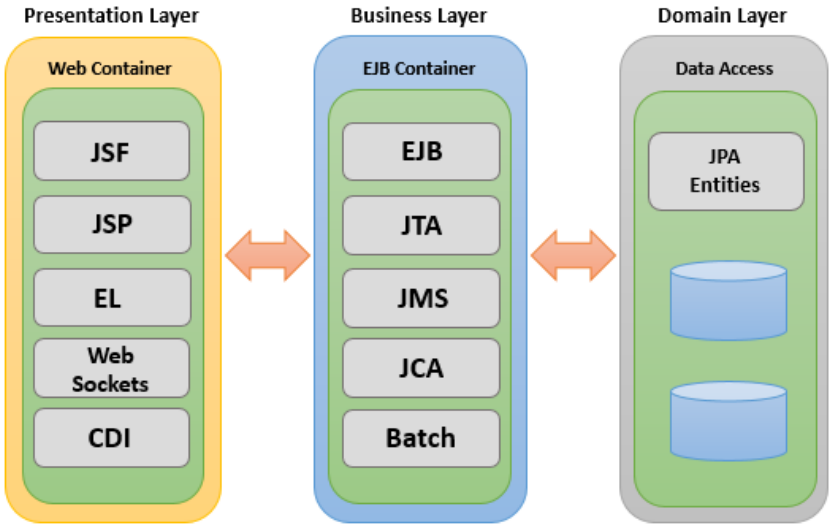

Figura 8: Arquitectura de Java EE.

Cada una de las capas mencionadas anteriormente, está soportada por un conjunto de APIs que define la plataforma Java EE además de un grupo de servicios transversales como invocación, inyección y administración de recursos que son desplegados en contenedores encargados de suplir soporte en tiempo de ejecución.

La especificación Java EE 7, definida en la JSR 342 [28] y liberada en junio de 2013, contiene otras 31 especificaciones entre actualizaciones a versiones anteriores (JSF 2.2, EJB 3.2, JAX-RS 2.0, etc) e incorporación de nuevas APIs entre las que se destaca WebSocket 1.0, para el desarrollo y despliegue de componentes que utilizan el protocolo WebSocket definido por HTML5 [29]. En la figura 9 se presentan los componentes que agrupan las especificaciones y que conforman la arquitectura de Java EE 7.

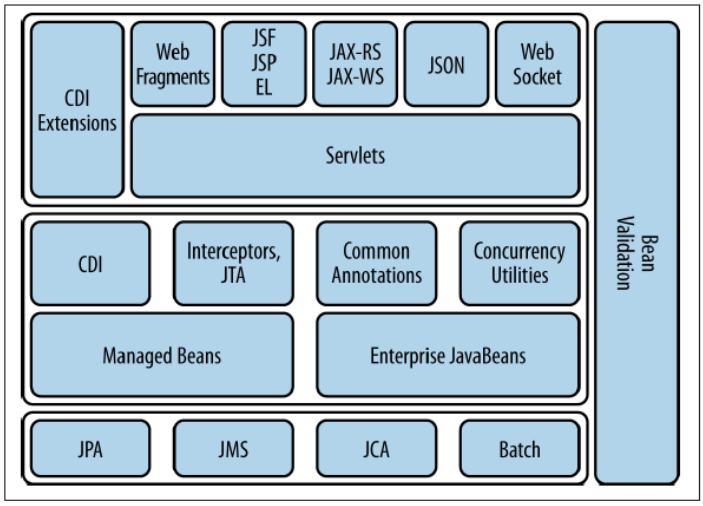

Figura 9: Componentes de Java EE 7. Tomado de Gupta [27].

Dado que el transformador MDE Gengular hace uso de tecnologías HTML5 y AngularJS en la capa de frontend en lugar de JSF, JSP y EL, se hará énfasis en los componentes relevantes para las capas intermedias (Web Services, JSON) y de backend (Enterprise Java, JPA) en el presente proyecto.

4.1. Invocación de servicios REST

Para la implementación de servicios web se hace uso del estilo arquitectónico REST (*Representation State Transfer*), donde los servicios son vistos como recursos que pueden ser identificados mediante URIs (*Uniform Resource Identifiers*). Los servicios web desarrollados bajo REST se conocen como RESTful web services [30].

Para realizar operaciones sobre los recursos en RESTful se utilizan los métodos que define HTTP, de esta manera se hace intuitivo y fácil de mantener las acciones sobre los servicios web. En la tabla 4 se describe cada una de estos métodos y su propósito.

Método	Propósito	Operación CRUD
GET	Una petición GET se utiliza para recuperar un recurso existente.	Read
POST	Una petición POST se utiliza para crear un nuevo recurso.	Create
PUT	Una petición PUT se utiliza para actualizar un recurso existente.	Update
DELETE	Una petición DELETE se utiliza para eliminar un recurso existente.	Delete

Tabla 4: Métodos HTTP en servicios web RESTful.

Al hacer uso de los métodos o verbos HTTP para realizar acciones u operaciones sobre los recursos, se promueve la estandarización en los servicios web RESTful y se evita definir operaciones arbitrarias en cada desarrollo [30].

4.2. Componentes de negocio EJB

En arquitecturas Java EE, típicamente los servicios web invocan a los Enterprise JavaBeans (EJBs), componentes del lado del servidor que encapsulan la lógica de negocio de la aplicación, es decir las reglas de negocio y aquella funcionalidad principal que da sentido al sistema [31]. Existen dos tipos de EJBs: Los de sesión (*Session Beans*) que pueden invocarse mediante programación por un cliente local, remoto, o servicio web, y los dirigidos por mensajes (*Message-Driven Beans*) que permiten procesar mensajes asíncronamente.

Los beans de sesión a su vez pueden ser tres tipos: con estado (*stateful*) donde se mantiene el estado conversación a través de las peticiones, sin estado (*stateless*) que no mantiene estado conversacional con el cliente, y de instancia única (*singleton*) que es creado una sola vez por aplicación y existe para todo el ciclo de vida de la misma [31].

4.3. Entidades persistentes JPA

En la mayoría de aplicaciones Java EE, es necesario conectarse a bases de datos para realizar operaciones como consulta, creación, actualización o eliminación de información; para ello, los EJBs hacen uso de la API JPA (*Java Persistence API*), especificación que establece como se puede aplicar persistencia a las entidades en Java: objetos de dominio de persistencia que comúnmente representan una tabla o vista de base datos [32].

Las entidades son administradas por el *EntityManager*, el cual provee un conjunto de métodos para interactuar con el contexto de persistencia (ver tabla 5).

Método	Propósito
persist	Permite insertar una entidad en la base de datos.
find	Permite localizar una entidad mediante su llave primaria.
merge	Permite sincronizar o actualizar una entidad en memoria con su versión en la base de datos.
refresh	Asigna a una entidad en memoria los valores almacenados en base de datos, cancelando posibles modificaciones en memoria durante la transacción.
remove	Permite eliminar una entidad de la base de datos.
flush	Permite aplicar inmediatamente a la base de datos modificaciones realizadas sobre las entidades.

Tabla 5: Métodos del EntityManager en JPA.

IV -DESARROLLO DEL PROYECTO

En este capítulo se presenta las definiciones, artefactos y actividades relevantes realizadas en la ejecución del proyecto.

1. Comparación y selección de tecnologías SPA

Aunque en el mercado existen más de 50 frameworks JavaScript con los que es posible construir aplicaciones web siguiendo el paradigma de arquitectura SPA, según la publicación "5 BEST JAVASCRIPT FRAMEWORKS IN 2017" de la compañía estadounidense de desarrollo de software DA-14 [67], los 5 mejores frameworks son AngularJS, ReactJS, Vue.js, Ember.js y Meteor.js respectivamente. En la tabla 6 se describe cada uno de estos y sus principales características.

Framework JavaScript	Descripción
	AngularJS es un framework JavaScript, de propósito general, para construir aplicaciones web modernas. Desarrollado y mantenido por Google, se destaca por su innovador sistema de plantillas, facilidad de desarrollo y por estar construido bajo prácticas de ingeniería muy sólidas [19].
	ReactJS es una librería JavaScript desarrollada y utilizada por Facebook e Instagram, para construir interfaces de usuario. Se caracteriza por ofrecer un rendimiento muy alto, mediante la estrategia de *Virtual DOM* donde los cambios al DOM se realizan en una copia en memoria que luego es sincronizada y renderizada con la original. [68].
	Vue.js se define como un framework JavaScript progresivo, incrementalmente adoptable, para construir interfaces web interactivas. Introduce el concepto de *Data Driven View*, característica que permite mantener en sincronía el modelo y la vista de forma transparente para el desarrollador [69].
	Ember.js es un framework para la creación de aplicaciones web ambiciosas. Basado en jQuery, se destaca por su sistema de convenciones que reduce la escrita de código, además posee uno de los componentes de *Routing* más poderosos. Es utilizado en portales web como LinkedIn, Groupon y Vine [70].
	Meteor.js es una plataforma de aplicaciones reactivas, sencillas y potentes, capaz de producir portales web robustos y sofisticados con solo unas pocas líneas de código. Dado que Meteor está construido sobre Node.js, se utiliza JavaScript tanto en el cliente como en el servidor [71].

Tabla 6: Frameworks JavaScript más utilizados.

1.1. Comparación de tecnologías SPA

Jaap Koetsier en su trabajo de grado "Evaluation of JavaScript frame-works for the development of a web-based user interface for Vampires" [72] de la Universidad de Amsterdam, realiza una comparación detallada de los cuatro frameworks más populares de la industria (AngularJS, ReactJS, Ember.js y Vue.js), utilizando el modelo para la calidad de productos de software del estándar ISO 25010:2011 [73], el cual define las siguientes 8 características:

- **Fiabilidad:** La fiabilidad de un producto de software se basa en la madurez, disponibilidad, tolerancia a fallos y capacidad de recuperación del producto.

- **Operabilidad:** El grado en que un producto de software es fácil de usar, fácil de aprender y atractivo de usar.

- **Desempeño:** El tiempo de respuesta, la utilización de recursos y la capacidad de un producto de software.

- **Seguridad:** La seguridad en el procesamiento de los datos y la gestión de los mismos en cuanto a confidencialidad, integridad, no repudio, responsabilidad y autenticidad.

- **Compatibilidad:** El grado en que un producto de software es capaz de coexistir con otro software, compartiendo los mismos recursos y para lo cual extiende sus capacidades de interoperabilidad.

- **Mantenibilidad:** La mantenibilidad se basa en la modularidad, reutilización, modificabilidad, capacidad de análisis y de prueba de un producto de software.

- **Transferibilidad:** El grado en que un producto de software puede ser transferido de un sistema a otro. Se basa en la portabilidad, adaptabilidad, instalación y despliegue del mismo.

- **Aptitud funcional:** El grado en que un producto de software cumple con los requisitos establecidos por los usuarios del producto.

Teniendo en cuenta que las características anteriores son demasiado amplias y aplican a cualquier producto de software, para que puedan ser tomados como criterios de evaluación en el ámbito de los frameworks JavaScript en el presente proyecto se realiza la siguiente personalización:

- **Madurez (Fiabilidad):** ¿Qué tan maduro es el framework evaluado en el mercado?

- **Facilidad de uso (Operabilidad):** ¿Cuál es la curva de aprendizaje del framework evaluado? ¿Su API, documentación y estructuras son claras y fáciles de entender?

- **Rendimiento (Desempeño):** ¿Cómo funciona el framework evaluado en términos de velocidad y uso de recursos?

- **Soporte de navegadores (Compatibilidad, Transferibilidad):** ¿Qué navegadores soporta el framework evaluado? ¿Funciona correctamente en una amplia gama de navegadores?

- **Modularidad / Reutilización (Mantenimiento):** ¿Es fácil construir software modular usando el framework evaluado? ¿Podemos construir fácilmente piezas reutilizables?

- **Testabilidad (Mantenibilidad):** ¿El framework evaluado viene con una suite de pruebas, y es fácil de usar?

- **Enrutamiento (Aptitud funcional):** ¿En aplicaciones de una sola página (SPA), el enrutamiento entre vistas es manejado por el framework evaluado?, ¿Incorpora el framework la funcionalidad de enrutamiento?

- **Sistemas de plantillas (Aptitud funcional):** ¿Cómo construye el framework evaluado las páginas HTML? ¿Este proceso es claro y conciso?

En la tabla 7 se presenta el resultado de la evaluación realizada, la puntuación está en una escala de 1 a 4, donde 4 es la calificación más alta y 1 la más baja.

	AngularJS	**ReactJS**	**Ember.js**	**Vue.js**
Madurez	4	2	4	1
Facilidad de uso	1	4	3	2
Rendimiento	3	4	1	2
Soporte navegadores	4	4	4	4
Modularidad	4	4	4	4
Testabilidad	4	3	2	1
Enrutamiento	4	3	4	4
Sistemas de plantillas	4	1	2	4
TOTAL	**28**	**25**	**24**	**22**

Tabla 7: Evaluación de frameworks JavaScript. Tomado de Koetsier [72]

Los resultados arrojados por la evaluación, demuestran que AngularJS es el framework que mejor satisface los criterios definidos al obtener el máximo puntaje en la mayoría de ítems, sin embargo, se debe tener en cuenta y es importante resaltar que también es el framework con la calificación más baja en facilidad de uso, esto debido a que la curva de aprendizaje es alta en cuanto al tiempo requerido para comprender y dominar el framework. Además, en rendimiento obtuvo 3 de 4, mientras que ReactJS la máxima calificación gracias a su estrategia de *Virtual DOM* (explicado en la tabla 6).

1.2. Arquetipo generador de la arquitectura SPA seleccionada

En esta actividad se construyó un arquetipo Maven llamado *gengular-archetype* en consonancia con la arquitectura de una aplicación SPA presentada en la sección 3.1 de la parte III (Marco Teórico), con el fin de tener una línea base que defina la estructura de los proyectos a construir y contenga los artefactos requeridos para el funcionamiento de las diferentes tecnologías involucradas, tales como librerías JavaScript, hojas de estilo CSS, archivos de configuración, entre otros.

El arquetipo Maven también es utilizado por el transformador MDE Gengular para indicar las rutas donde los artefactos generados serán colocados, de esta manera es posible tener una aplicación funcional en las tecnologías destino con muy pocos ajustes por parte del diseñador o desarrollador de software. La estructura del arquetipo consiste en 2 proyectos Maven: el *Frontend* que contiene los artefactos del lado del cliente (AngularJS, Bootstrap, páginas HTML5) y el proyecto *Backend* con los elementos de lado del servidor (servicios REST, entidades y lógica de negocio). En la figura 10 se presenta los elementos del arquetipo.

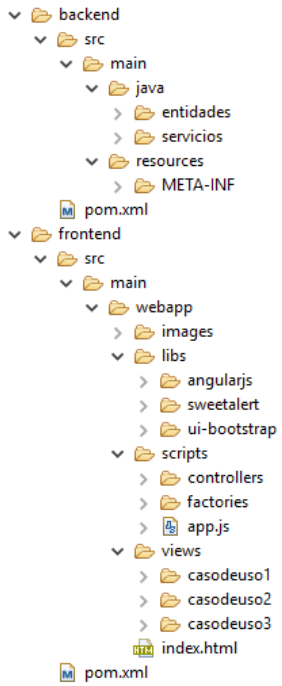

Figura 10: Estructura del arquetipo maven Gengular-archetype.

Descripción de artefactos proyecto ***Frontend***:

- **Images:** Contiene las imágenes del proyecto, como logotipos, iconos, etc.
- **Libs:** Contiene las librerías que utiliza el proyecto en capa de presentación, por ejemplo, AngularJS, Bootstrap, SweetAlert.
- **Scripts:** Contiene los artefactos JavaScript, como controladores y factorías. También el archivo *app.js* encargado de manejar las reglas de navegación de la aplicación.
- **Views:** Contiene las páginas HTML5 de cada caso de uso y la vista principal de la aplicación -*index.html*- que permite hacer el SPA.

Descripción de artefactos proyecto ***Backend***:

- **Java:** Contiene las entidades de negocio y servicios del proyecto.
- **Resources:** Contiene recursos del proyecto, como el archivo *persistence.xml* que define la unidad de persistencia a la base de datos.

El arquetipo ofrece importantes ventajas en el proceso de desarrollo de software: por una parte promueve la reutilización de código al evitar tener que crear y definir archivos que son comunes en todos los proyectos, además facilita la mantenibilidad al contar con una estructura estándar que indica donde debe ir cada elemento y separa las responsabilidades del lado cliente y del servidor. Finalmente, también contribuye a la productividad, pues una vez el arquetipo es instanciado y es ejecutado el proyecto generado, permite tener una aplicación funcional sin tener que escribir ninguna línea de código como se muestra en la figura 11.

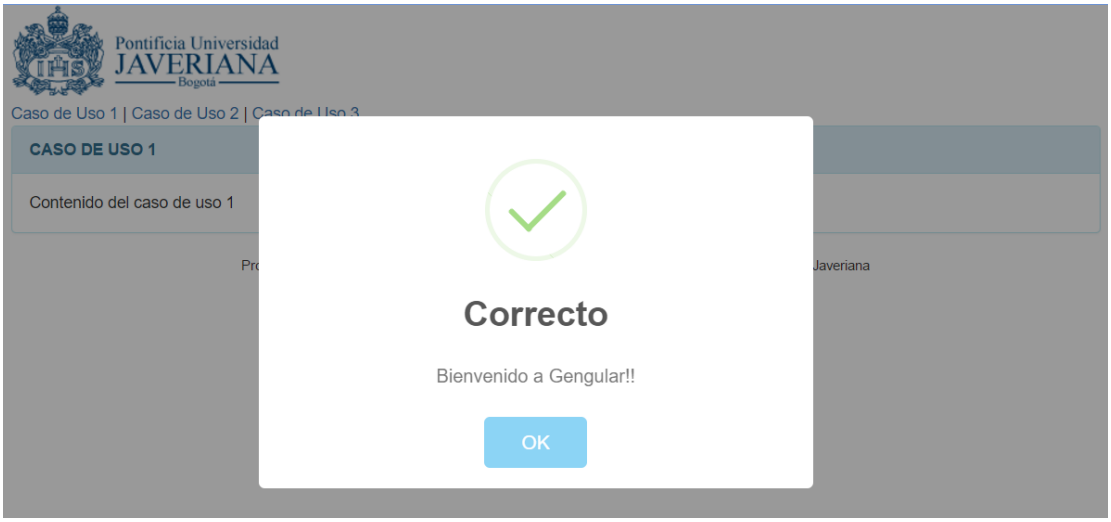

Figura 11: Instanciación de arquetipo Maven para crear un proyecto nuevo.

2. Requerimientos y funcionalidades del transformador

Con el fin de plasmar los requerimientos funcionales y no funcionales a satisfacer por el transformador MDE Gengular, se ha definido un conjunto de historias de usuario. Cada historia de usuario contiene un identificador, una descripción de la funcionalidad requerida, la justificación, clasificación de acuerdo al modelo FURPS [18], actor y nivel de riesgo asociado.

El modelo FURPS permite determinar el tipo de requisito que se está abordando en la historia de usuario según cinco grupos descritos a continuación: Funcionalidad (F), Usabilidad (U), Confiabilidad (R), Desempeño (P) y Soporte (S).

Como actores involucrados en las historias de usuario se ha identificado al Diseñador de Aplicaciones (DI), quien es el encargado de realizar la construcción de los modelos en ISML que luego son transformados en fuentes en las tecnologías destino. También se tienen como actor al Cliente Final (CI), quien será el usuario que hará uso de las aplicaciones generadas.

Por otra parte, las historias de usuario tienen asociada una complejidad de implementación medida en una escala de 1 a 3, donde 1 significa un nivel de complejidad bajo, 2 medio y 3 alto.

ID Hist.	Historia de Usuario	Justificación	Clasificación FURPS	Actor	Complejidad de implementación
H1	El transformador debe generar el código en las tecnologías destino HTML5-AngularJS-JEE7. Para ello deberá tomar como insumo modelos construidos en el lenguaje ISML.	El stack de tecnologías Java EE 7, AngularJS y HTML5 es impulsado por Google como el nuevo estilo de construcción de aplicaciones al promover la escalabilidad, desempeño, usabilidad y despliegue en la nube. Por lo anterior, los fuentes generados por el transformador corresponden a las tecnologías mencionadas.	F	DI	2
H2	El transformador debe generar aplicaciones con una arquitectura multimódulo haciendo uso de arquetipos Maven que contemple:	Para aprovechar las ventajas que ofrece Maven en la construcción y gestión de proyectos multi-capa, caracterizados por una mejor mantenibilidad y modificabilidad del código, se hará uso de la arquitectura multimódulo.	S	DI	2

	- Un módulo frontend en el cual irán los artefactos asociados a capa de presentación. - Un módulo backend que contenga los elementos de lógica de negocio.				
H3	El transformador debe generar aplicaciones que sean compatibles con múltiples dispositivos, es decir con *smartphones*, *tablets* y *laptop*. Para ello hacer uso de diseños adaptativos.	Las aplicaciones modernas demandan poder ser accedidas desde múltiples dispositivos y no únicamente en equipos de escritorio.	U	CI	1
H4	El transformador debe generar los artefactos en las tecnologías destino en un tiempo razonable para el Diseñador de Aplicaciones, no mayor a 1 minuto.	Si los tiempos de generación de los artefactos están por encima de 1 minuto, hay riesgo que el transformador no quiera ser usado por los Diseñadores de Aplicaciones.	P	DI	3
H5	Se debe crear la respectiva documentación asociada al transformador, como manuales técnicos y de usuario que detallen la manera en que puede ser utilizado y mantenido/actualizado el mismo, realizando la descripción de sus módulos y artefactos.	Es necesario contar con documentación a nivel técnico y de usuario que permita la evolución y mantenimiento del transformador, así como su correcta utilización por parte de los usuarios.	U	DI	2

H6	Se debe diseñar un modelo ISML que represente un componente de menú. El componente de menú deberá permitir mostrar los nodos y subnodos de un menú almacenados en una base de datos y representados mediante entidades, manteniendo un diseño adaptativo que pueda ser incorporado en aplicaciones AngularJS generadas previamente.	El componente de menú es necesario en las aplicaciones ya que ofrece la manera de consultar y acceder a las diferentes opciones que provee el sistema.	F	CI	2
H7	Se debe diseñar un modelo ISML que represente un componente de notificaciones que permita parametrizar, realizar el procesamiento y consultar las notificaciones enviadas.	El componente de notificaciones es requerido en la mayoría de proyectos, fue escogido por el alto impacto y nivel de reutilización que puede llegar a tener en los clientes de la empresa caso de estudio.	F	CI	2
H8	Se debe realizar las pruebas de software tanto del transformador como de los componentes generados.	Es necesario validar y verificar el correcto funcionamiento del transformador, así como de los componentes empresariales generados mediante los modelos ISML construidos en las historias de usuario 6 y 7.	F	CI	1
H9	El sistema debe contar con un wizard que permita acoplar los componentes empresariales.	Los componentes empresariales generados deberán poder ser acopladas a aplicaciones AngularJS generadas previamente mediante un	S	DI	2

		asistente o wizard, que facilite las tareas de integración y gestión de las dependencias.			
H10	Se debe crear un arquetipo en Maven que contenga la estructura de los proyectos destino.	El arquetipo a construir servirá como estructura base para los proyectos a generar en las tecnologías objetivo (HTML5, AngularJS, JEE7).	F	DI	1
H11	Se debe validar con los usuarios el transformador Gengular y el Wizard.	Es necesario conocer la opinión de los usuarios con respecto a los desarrollos realizados, para identificar si es factible utilizar el transformador, los componentes empresariales y el wizard dentro de un ambiente laboral. Para ello, se realizará un formato de encuesta que será diligenciado por ingenieros de la empresa caso de estudio.	R	CI	1

Tabla 8: Requerimientos y funcionalidades del transformador.

3. Arquitectura completa de la solución

A continuación, se presenta la arquitectura de solución propuesta para el transformador Gengular, estructurada a partir de un conjunto de vistas que describen diferentes aspectos del sistema, los elementos y tecnologías que la conforman así como las interacciones entre los mismos.

3.1. Vista de procesos

La figura 12 muestra el proceso a seguir para generar una aplicación utilizando el transformador, tomando como insumo los modelos definidos en ISML.

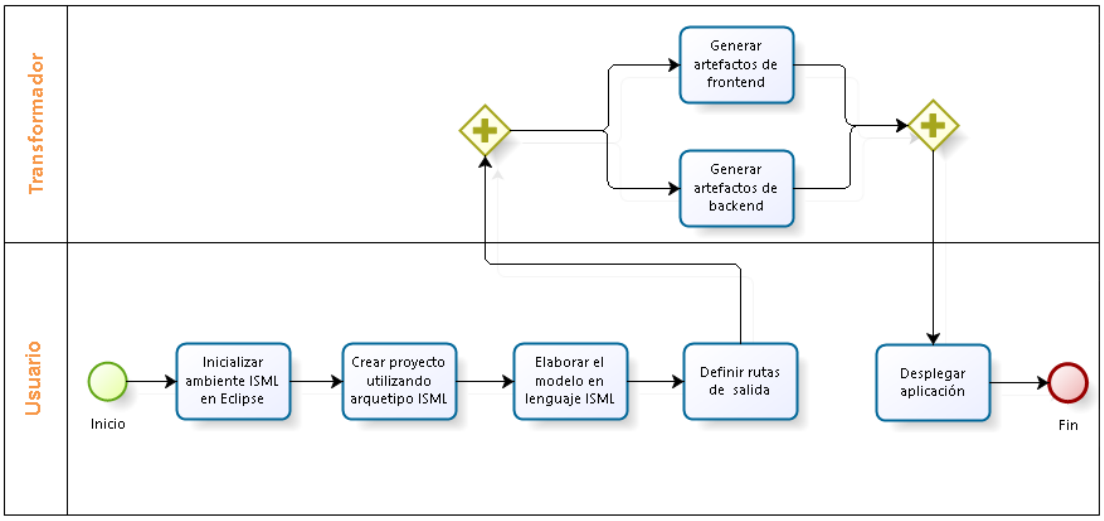

Figura 12: Vista de procesos de Gengular.

El proceso comienza cuando el usuario inicia el ambiente ISML en el entorno de desarrollo integrado Eclipse, donde se crea una nueva instancia utilizando el componente *org.eclipse.core.runtime*, el cual proporciona soporte para la plataforma de ejecución y brinda extensiones al ambiente.

Posteriormente el usuario crea un nuevo proyecto utilizando el arquetipo de ISML, que define la estructura de directorios y los archivos necesarios. Se elaboran los modelos que conforman la aplicación, es decir las páginas, controladores, servicios y entidades requeridos.

Luego el usuario edita el archivo *generation.conf.json* con el fin de definir las rutas de salida para el transformador y demás propiedades esenciales; después de esto el transformador toma los modelos ISML y realiza la creación de artefactos en las tecnologías destino para el *frontend* (HTML5, Bootstrap, AngularJS) y del *backend* (Servicios java y entidades de persistencia).

Finalmente, el usuario realiza el despliegue de la aplicación en el servidor de aplicaciones seleccionado (Wildfly 10 ha sido utilizado en el presente proyecto).

3.2. Vista lógica

La vista lógica describe las tecnologías y componentes que conforman la solución, presentando las principales decisiones arquitectónicas que soportan el sistema y las consideraciones de diseño.

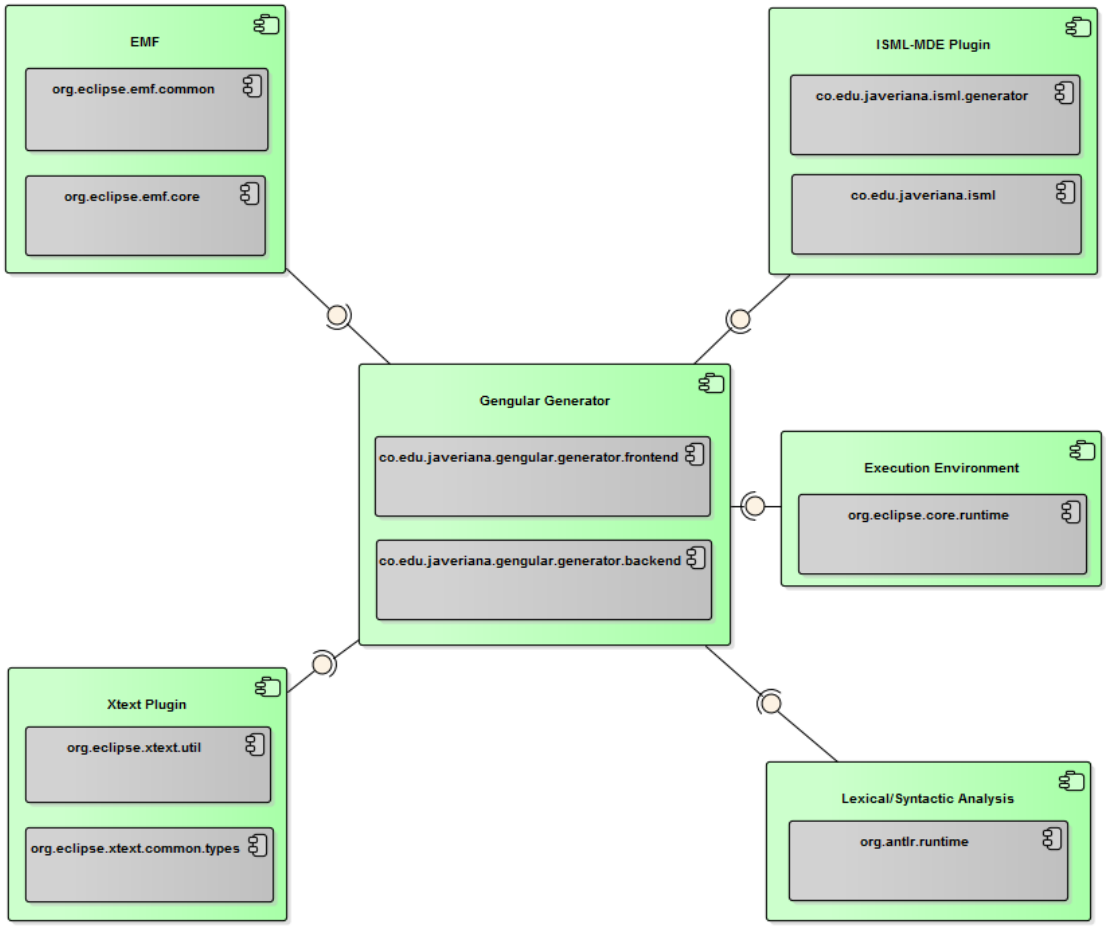

Figura 13: Vista lógica de Gengular.

El transformador Gengular se encuentra conformado por dos paquetes principales, los cuales contienen los elementos de configuración, generación y plantillas necesarias para producir los artefactos de salida. El primero, **co.edu.javeriana.gengular.generator.frontend** se encarga de las tecnologías de presentación que corresponden a HTML5, Bootstrap y AngularJS mientras que el segundo **co.edu.javeriana.gengular.generator.backend** tiene como objetivo las interfaces e implementación de servicios web y las entidades Java. En la tabla 9 se muestran las clases más relevantes que contienen cada uno de estos paquetes junto a su descripción.

Paquete co.edu.javeriana.gengular.generator.frontend	
Clase	**Descripción**
GengularFrontendGenerator	Esta clase instancia el conjunto de generadores que comprenden el *frontend*. Extiende a su vez de la clase GeneratorSuite de ISML.
GengularPagesGenerator GengularControllerGenerator GengularAppConfigGenerator	Estas clases corresponden a los generadores de páginas, controladores y reglas de navegación. Cada uno define aspectos como el nombre y extensión de los archivos, configuración de salida y la plantilla. Extienden a su vez de la clase SimpleGenerator de ISML.
GengularPagesTemplate GengularControllerTemplate GengularAppConfigTemplate ExpressionTemplate ReferenceTemplate StatementTemplate	Estas clases corresponden a las plantillas y permiten definir la estructura para los elementos de presentación como formularios, botones, paneles, tablas, campos de texto, contraseña, etc. Extienden a su vez de la clase SimpleTemplate de ISML.
Paquete co.edu.javeriana.gengular.generator.backend	
Clase	**Descripción**
GengularBackendGenerator	Esta clase instancia el conjunto de generadores que comprenden el *backend*. Extiende a su vez de la clase GeneratorSuite de ISML.
GengularEntityGenerator GengularServiceImplementationGenerator GengularServiceInterfaceGenerator	Estas clases corresponden a los generadores de servicios y entidades. Cada uno define aspectos como el nombre y extensión de los archivos, configuración de salida y la plantilla. Extienden a su vez de la clase SimpleGenerator de ISML.
GengularEntityTemplate GengularServiceImplementationTemplate GengularServiceInterfaceTemplate CommonTemplates ExpressionTemplate StatementTemplate ReferenceTemplate	Estas clases corresponden a las plantillas y permiten definir la estructura para las entidades Java y la interfaz e implementación de servicios los cuales contienen la lógica de negocio de la aplicación. Extienden a su vez de la clase SimpleTemplate de ISML.

Tabla 9: Clases que conforman el transformador Gengular.

3.3. Vista de Despliegue de una Aplicación Generada

En la figura 14 se muestran la distribución física para la solución, las tecnologías implicadas y la ubicación en nodos para los diferentes componentes. Como se observa, la aplicación generada está conformado por dos nodos que pueden ser desplegados en la misma máquina física o en máquinas diferentes. El nodo cliente puede ser tanto un navegador web que va a comunicarse con el proyecto *frontend* del servidor a través del protocolo HTTP o una aplicación java de escritorio/móvil que se comunica directamente con el proyecto *Backend* del servidor consumiendo servicios REST.

El nodo servidor contiene a Wildfly versión 10.1, servidor de aplicaciones Java donde son desplegadas las aplicaciones generadas en el presente proyecto (*frontend* y *backend*) y el sistema de administración de base de datos relacionales PostgreSQL en su versión 9.3 que almacena las entidades del modelo ISML.

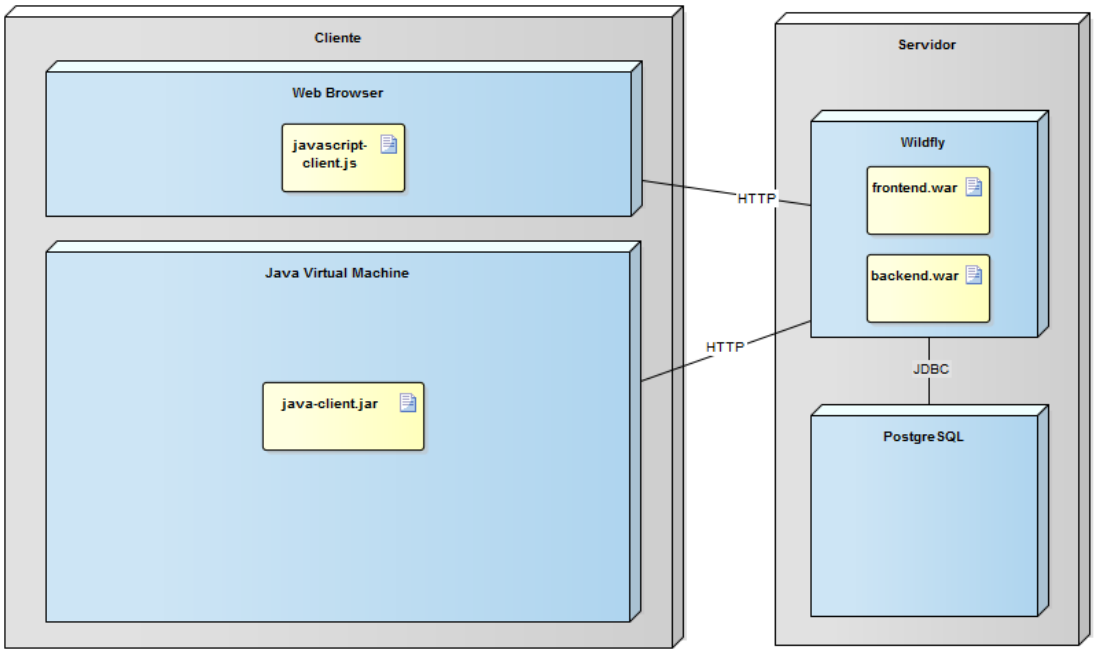

Figura 14: Vista despliegue de Gengular.

Cabe anotar que en el servidor Wildfly las aplicaciones Java pueden ser desplegadas independientemente, de esta manera se reduce el grado de acoplamiento y se facilita la mantenibilidad, además la comunicación entre aplicaciones es a través de servicios desacoplados tipo REST siguiendo las convenciones para métodos presentadas en la sección 4.1. Invocación de servicios REST.

4. Diseño: Reglas de transformación

Las reglas de transformación son un artefacto de diseño que permite ilustrar el inventario de los elementos de código fuente que hay que generar a partir del modelo ISML de una aplicación Web, para obtener una aplicación en el stack de tecnologías HTML5-AngularJS-JEE7.

Para ilustrar el inventario, se tomó como base para elaborar las reglas de transformación el demo GestionEstudiantes (https://bitbucket.org/gengular/gestion-estudiantes), el cual es una aplicación conformada por tres casos tipo CRUD y un caso de negocio para gestionar el proceso de inscripción de asignaturas en una institución de educación superior. Permite consultar, crear, editar y eliminar facultades, asignaturas y estudiantes, así como asociar uno o más asignaturas a un estudiante. Sin embargo, cabe aclarar que las reglas son generales y permitirán generar cualquier aplicación para la arquitectura destino AngularJS presentada en la figura 6 (sección 3 de la parte III Marco Teórico) tomando como insumo los elementos de la arquitectura de ISML descrita en la figura 15.

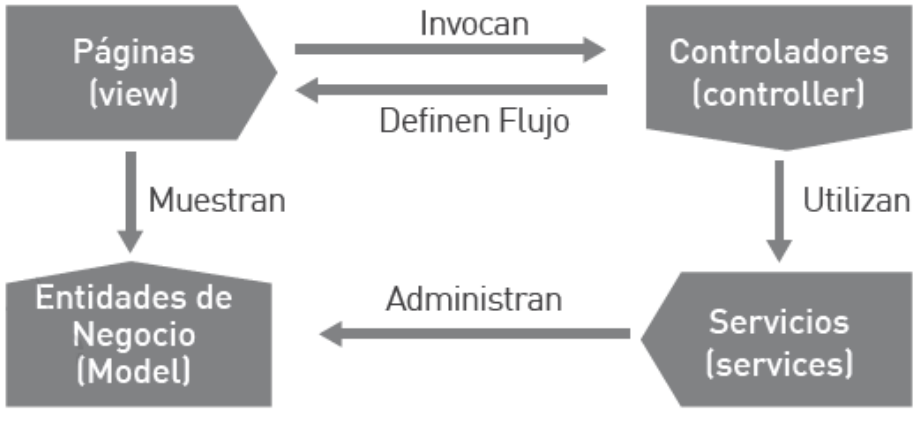

Figura 15: Arquitectura de ISML.

4.1. Reglas de transformación para páginas

Permiten especificar elementos visuales, como botones, cajas de texto, imágenes, enlaces, listas de selección, entre otros, sin entrar en excesivos detalles de un código HTML estándar.

Las reglas de transformación para páginas contemplan tres tecnologías destino que serán incluidas en la generación:

1) Bootstrap: etiquetas del framework Bootstrap como hoja de estilos CSS, que permiten crear diseños adaptativos, caracterizados por ser atractivos e intuitivos.

2) AngularJS: etiquetas del framework AngularJS para extender los atributos HTML por medio de directivas, habilitando la construcción de aplicaciones SPA Single-Page Applications.

3) HTML5: etiquetas HTML5 para estructurar y presentar el contenido de las páginas web mediante elementos como formularios, paneles, tablas, cajas de texto, cajas de contraseña, calendarios, radio botones, etc.

Elementos de página ISML	*Elementos de página en el proyecto frontend*	*Descripción*	*Tipo de tecnología destino*
`Button("nombreBoton") -> nombreAccion();`	`<a ng-click="nombreAccion()" class="btn btn-small btn warning">nombreBoton</a>`	Este elemento representa al componente botón, el cual permite ejecutar una acción asociada.	HTML5 BOOTSTRAP ANGULARJS
`Text( "labelComponente", entidad.atributo, máximo, mínimo);`	`<input type="text" id="labelComponente" name="labelComponente" ng-model ="entidad.atributo" minlength="mínimo" maxlength="máximo" />`	Este elemento representa una caja de texto de entrada editable. El valor mínimo y máximo indican la cantidad de caracteres permitidos.	HTML5 ANGULARJS
`Spinner( "labelComponente", entidad.atributo, incremento, mínimo, máximo, "prefijo")`	`prefijo <input type="number" id="labelComponente" name="labelComponente" min="mínimo" max="máximo" step="incremento" ng-model ="entidad.atributo" />`	Este elemento representa una caja de texto con dos botones triangulares que hacen que el valor aumente o disminuya en cada paso, según el número especificado en el incremento.	HTML5 ANGULARJS
`Password( "labelComponente", entidad.atributo, longitud)`	`<input type="password" id="labelComponente" name="labelComponente" ng-model ="entidad.atributo" minlength="longitud" maxlength="longitud" />`	Este elemento representa un componente de tipo contraseña, donde los caracteres ingresados no son visibles	HTML5 ANGULARJS

		de forma plana.	
`Label("Texto a mostrar")`	`<label>Texto a mostrar</label>`	Este elemento muestra en la página el texto recibido por parámetros.	HTML5
`Image("URL Imagen")`	`<img src="URL imagen">`	Este elemento permite mostrar una imagen en la página.	HTML5
`Link ("etiqueta del link" "URL del link")`	`<a href="URL del link">etiqueta del link</a>`	Este elemento representa un enlace hacia una página destino.	HTML5
`ComboChooser ("labelComponente", Array<Entidad> lista, Entidad valorSeleccionado, valor por defecto)`	`<select id="labelComponente" name="labelComponente" ngmodel="valorSeleccionado" ng-options="lista"> <option value="">valor por defecto</option> </select>`	Este elemento muestra una lista desplegable permitiendo al usuario seleccionar una sola opción.	HTML5 ANGULARJS
`RadioChooser ("labelComponente", Array<Entidad> lista, Entidad valorSeleccionado, valor por defecto)`	`<div ng-repeat="opcion in lista"> <input type="radio" name="labelComponente" ng-value="opcion" ng-model ="valorSeleccionado" ng-checked="valordefecto" /> <label>{{ opcion }}</label> </div>`	Este elemento representa un conjunto de radio botones que son excluyentes entre sí.	HTML5 ANGULARJS
`DataTable("Collection <Entidad>", null) { header : { Label("Atributo 1"); Label("Atributo 2"); Label("Atributo 3"); } body : for(Entidad entidad`	`<table class="table table-striped table-condensed"> <thead> <tr> <th>Atributo 1</th> <th>Atributo 2</th> <th>Atributo 3</th> </tr> </thead>`	Este elemento permite visualizar una tabla compuesta por un encabezado y un cuerpo. Cada fila	HTML5 BOOTSTRAP ANGULARJS

in lista) {
 	
Label(entidad.atributo1);
Label(entidad.atributo2);
Label(entidad.atributo3);
}
}
	<tbody>
	<tr ng-repeat="entidad in lista">
	<td>
	{{entidad.atributo1}}
	</td>
	<td>
	{{entidad.atributo2}}
	</td>
	<td>
	{{entidad.atributo3}}
	</td>
	</tr>
	</tbody>
	</table>
		corresponde a una entidad y se muestra en cada columna un atributo de dicha entidad.	

Calendar("labelComponente", entidad.valorFecha, null "dd/MM/yyyy", habilitada navegacion)	<input type="date" id="labelComponente" name="labelComponente" ng-model ="entidad.valorFecha" />	Este elemento muestra un componente de tipo calendario permitiendo escoger una fecha. El valor seleccionado se mostrará en la caja de texto.	HTML5 BOOTSTRAP ANGULARJS

Form {
Conjunto de elementos
que van dentro de la
forma
}
	<form
	novalidate="novalidate"
	class="form-horizontal">
	Conjunto de elementos que
	Van dentro de la forma
	</form>
		Este elemento representa un formulario el cual contiene un conjunto de elementos (botones, cajas de texto, cajas de contraseña, etc).	HTML5
			BOOTSTRAP

Panel("título") {
Conjunto de elementos
que van dentro del
panel
}
	
	título
	
	Conjunto de elementos que van
	dentro del panel
	
		Este elemento muestra un panel con su respectivo encabezado y cuerpo.	HTML5
			BOOTSTRAP

Media("URL video YouTube", ancho, alto);	<video width="ancho" height="alto" controls> <source src="URL video" type="video/mp4"> <source src="URL video" type="video/ogg"> </video>	Este elemento permite mostrar contenido multimedia en la página, como videos o animaciones.	HTML5

Tabla 10: Reglas de transformación para páginas.

4.1.1. Reglas de transformación relativas a flujo de navegación de páginas

Mediante el objeto $routeProvider de AngularJS, se define las reglas de navegación de páginas que permiten asociar una ruta a una página HTML y el respectivo controlador que atenderá la petición. A continuación, se ilustra las reglas de transformación.

Página ISML	*Archivos en el proyecto frontend*	*Descripción*
Page consultar_entidades() controlledBy EntidadCtrl {}	Creación de archivo: consultar_entidades.html Modificación de archivo **app.js**: $routeProvider .when('/consultar_entidades', {templateUrl: 'views/consultar_entidades.html' controller: 'EntidadCtrl'});	Por cada página en ISML, durante la transformación se debe crear un archivo HTML con los elementos de página descritos en la tabla anterior 3.1, Además se agregará la regla de navegación en el archivo **app.js** mostrada en la columna izquierda.

Tabla 11: Reglas de transformación relativas a flujo de navegación de páginas.

4.1.2. Ilustración de la aplicación de las reglas de transformación para páginas

Ejemplo de transformación de una página ISML que permite crear un estudiante, hacia un archivo con extensión .html conformado por las tecnologías HTML5, Bootstrap y AngularJS.

Página ISML

package co.edu.javeriana;
import co.edu.javeriana.entities.* ;
 	
page crear_estudiante(Any container, Collection<Estudiante> collection, Estudiante estudiante) controlledBy EstudianteCtrl {
Panel("CREAR ESTUDIANTE") {

```
    Form {
      Text("Número de Documento",
           estudiante.numeroDocumento,
           15,3);
      Text("Nombres",
           estudiante.nombres,
           25,3);
      Text("Apellidos",
           estudiante.apellidos,
           25,3);
      Text("Dirección",
           estudiante.direccion,
           50,10);
      Text("Teléfono",
           estudiante.telefono,
           20,7);
    Button("Aceptar") -> crearEstudiante();
    Button("Cancelar",false) -> cancelar();
    }
  }
}
```

Tabla 12: Aplicación de las reglas de transformación para páginas.

La página resultante crear_estudiante.html después de aplicar las reglas de transformación es la siguiente (tecnologías HTML5, Bootstrap y AngularJS):

Página Resultante

Tabla 13: Página resultante después de aplicar las reglas de transformación.

El código fuente de la página el siguiente:

Página en el proyecto frontend

```
<div class="panel panel-info">
 <div class="panel-heading">
   <strong>CREAR ESTUDIANTE</strong>
 </div>
 <div class="panel-body">
   <form novalidate="novalidate" class="form-horizontal"name="crearEstudianteForm">
     <div class="form-group">
      <label class="control-label col-sm-2" for="numeroDocumento">
      Número Documento:</label>
      <div class="col-sm-10">
        <input type="text"
               id="numeroDocumento"
               name="numeroDocumento"
               ng-model="estudiante.numeroDocumento"
               placeholder="Número de Documento"
               minlength="3"
               maxlength="15"
               required />
      </div>
     </div>
     <div class="form-group">
      <label class="control-label col-sm-2" for="nombres">Nombres:</label>
      <div class="col-sm-10">
        <input type="text"
               id="nombres"
               name="nombres"
               ng-model="estudiante.nombres"
               placeholder="Nombres"
               minlength="3"
               maxlength="25"
               required />
      </div>
     </div>
     <div class="form-group">
      <label class="control-label col-sm-2" for="apellidos">Apellidos:</label>
      <div class="col-sm-10">
        <input type="text"
               id="apellidos"
               name="apellidos"
               ng-model="estudiante.apellidos"
               placeholder="Apellidos"
               minlength="3"
               maxlength="25"
               required />
      </div>
     </div>
     <div class="form-group">
      <label class="control-label col-sm-2" for="direccion">Dirección:</label>
      <div class="col-sm-10">
        <input type="text"
               id="direccion"
               name="direccion"
               ng-model="estudiante.direccion"
               placeholder="Direccion"
               minlength="10"
               maxlength="50"
               required />
      </div>
     </div>
     <div class="form-group">
      <label class="control-label col-sm-2" for="telefono">Teléfono</label>
```

```
      <div class="col-sm-10">
        <input type="text"
               id="telefono"
               name="telefono"
               ng-model="estudiante.telefono"
               placeholder="Telefono"
               minlength="7"
               maxlength="20"
               required />
      </div>
     </div>
     <br />
     <div class="form-group">
      <div class="col-sm-offset-2 col-sm-10">
      <a ng-click="cancelar()" class="btn btn-small btn-danger">Cancelar</a>
      <a ng-click="crearEstudiante()" class="btn btn-small btn-success">Aceptar</a>
      </div>
     </div>
   </form>
 </div>
</div>
```

Tabla 14: Código fuente generado para página html crear estudiante.

4.2. Reglas de transformación para controladores

Permiten especificar acciones invocadas desde las páginas y el flujo de páginas. Una acción puede invocar servicios de negocio.

Elemento ISML	*Elemento controlador en el proyecto frontend*	*Descripción*

Tipo: encabezado de un controlador
 	
Ejemplo:
controller EntidadCtrl {
Contenido del
controlador...
}
	app.controller('EntidadCtrl', []
	function() {
	Contenido del controlador…
	}]);
		Por cada controlador en ISML se debe crear un archivo JavaScript con el nombre del controlador (EntidadCtrl.js).

Tipo: redirección hacia una página mediante la directiva show.
 	
Ejemplo:
irEditarEntidad(Integer idEntidad) {
Show
 	
editar_entidad(idEntidad);
}
	$scope.irEditarEntidad = function (idEntidad) {
	
	$location.path('editar_entidad/'
	+
	idEntidad);
	};
		Este método permite hacer la redirección a la página editar_entidad, enviando el identificador de la entidad que se desea editar.

Tipo: Condicional para la ejecución de bloque de código.	if ($scope.entidad.atributo) { $location.path('mostrar_detalle/'	Definición de estructura condicional IF. Si el valor del atributo de la

Ejemplo:
if (entidad.atributo) {
  show mostrar_detalle(
  entidad
  );
}
	+
	  entidad);
	}
		entidad es verdadero, se muestra el detalle.

Tipo: Invocación a método del controlador
 	
Ejemplo:
calcularTotal(Double valor, Double incremento)
{
  total = valor *
  incremento;
  ->
mostrarMensaje(total);
}
 	
mostrarMensaje(mensaje) {
  show(mensaje);
}
	$scope.calcularTotal = function(valor,incremento) {
	  total = valor * incremento;
	  $scope.mostrarMensaje(total);
	};
	
	$scope.mostrarMensaje = function (mensaje) {
	  sweetAlert(mensaje);
	};
		Muestra la invocación desde un método en el controlador a otro método en el mismo controlador.

Tipo: Ejecución de método al inicializarse controlador
 	
Ejemplo:
default init() {
  String mensaje =
  "Bienvenido";
  show (mensaje);
}
	$scope.init = function () {
	  mensaje = "Bienvenido";
	  sweetAlert(mensaje);
	};
	
	$scope.init();
		Realizar la ejecución de un método en un controlador cuando éste es inicializado.

Tabla 15: Reglas de transformación para controladores.

4.2.1. Ilustración de la aplicación de las reglas de transformación para controladores

Ejemplo de transformación de un elemento ISML que representa el controlador de la página para crear un Estudiante, hacia un controlador en AngularJS.

Elemento ISML
package co.edu.javeriana ; import co.edu.javeriana.entities.* ; controller EstudianteCtrl {

```
  has List<Estudiante> estudiantes;
  has Boolean valid;
  has Persistence persistence;

  crearEstudiante(Estudiante estudiante) {
    String mensaje;
    if(valid) {
      persistence.save(estudiante);
      mensaje = "El estudiante ha sido creado correctamente.";
      show Consultar_Estudiantes(mensaje);
    } else {
      mensaje = "La información se encuentra incompleta, por favor verifique.";
      show (mensaje);
    }
  }
  cancelar() {
    show Consultar_Estudiantes();
  }
}
```

Tabla 16: Aplicación de las reglas de transformación para controladores.

A partir del elemento ISML presentado anteriormente, el transformador generará el siguiente controlador AngularJS en el proyecto frontend:

Elemento controlador en el proyecto frontend

```
app.controller('EstudianteCtrl', ['$scope', '$routeParams', '$location', '$http',
'BASE',
 function ($scope, $routeParams, $location, $http, BASE) {

  $scope.estudiante;

  $scope.crearEstudiante = function () {
    if ($scope.crearEstudianteForm.$valid) {
      $http.post(BASE +
      '/resources/estudiante/agregarEstudiante',angular.toJson($scope.estudiante))
         .success(function (data) {
           sweetAlert("Correcto", "El estudiante ha sido creado correctamente.",
           "success");
           $location.path('/consultar_estudiantes');
         )
         .error(function (data) {
          sweetAlert("Error", "No fue posible crear el estudiante, por favor
          intente de nuevo.","error");
      });
    } else {
      sweetAlert("Error", "La información se encuentra incompleta, por favor
      verifique.","error");
    }
  };
  $scope.cancelar = function () {
    $location.path('/consultar_estudiantes');
  };
}]);
```

Tabla 17: Código fuente generado para controlador AngularJS.

4.3. Reglas de transformación de servicios y entidades de negocio

Para las reglas de transformación de Servicios (que permiten especificar la interfaz de servicios donde se concentra la lógica del negocio de la aplicación) y de Entidades de Negocio (que permiten especificar atributos, restricciones y asociaciones con otras entidades) se utilizarán submódulos del transformador ISML-JEE6-JavaFX-REST desarrollado en el proyecto Lion2 [2] debido a que en el stack tecnológico propuesto para el *Backend* de la aplicación, se mantiene el uso de Java Entities y EJBs expuestos como servicios REST.

5. Modelaje de componentes empresariales y estrategia de transformación/acople a una aplicación generada

Los componentes empresariales tienen una gran importancia en el contexto del desarrollo de software, debido a que resuelven requerimientos comunes de los proyectos como son la seguridad, auditoría, procesamiento de archivos, generación de notificaciones, administración de menús, entre otros. Además, impactan directamente en la calidad del software entregado, ya que al ser activos tecnológicos ampliamente probados contribuyen a reducir el número de errores o *bugs* asociados a la funcionalidad. Por otra parte, los componentes empresariales representan una ventaja competitiva en relación a los menores tiempos requeridos desde que un producto de software es definido hasta que pueda ser liberado o comercializado, generando valor al cliente (*time-to-market*).

Teniendo en cuenta la importancia que en el presente proyecto tienen los componentes empresariales, los dos principales retos identificados para alcanzar la automatización y reutilización de los mismos, fueron en primer lugar, lograr la preservación de los componentes en el tiempo y que su lógica de negocio se vea afectada lo menos posible por los cambios de plataformas o tecnologías y, en segundo lugar, permitir que el acople de los componentes a una aplicación nueva sea de manera rápida y con un alto grado de automatización. A continuación, se presenta la estrategia utilizada para solventar dichas problemáticas.

5.1. Modelaje de componentes empresariales

Los componentes empresariales que posee actualmente la empresa caso de estudio, se encuentran construidos en Java Empresarial versión 5 y 6 siguiendo la arquitectura de referencia que utilizaba anteriormente Heinsohn (ver figura 16), la cual consiste en el framework JSF para presentación, Richfaces en la versión 5 y Primefaces en la versión 6, así como controladores Backing Beans encargados de realizar invocaciones a componentes de negocio EJB, que a su vez contienen la lógica de negocio y realizan la persistencia mediante operaciones en entidades Java que abstraen el modelo de base de datos.

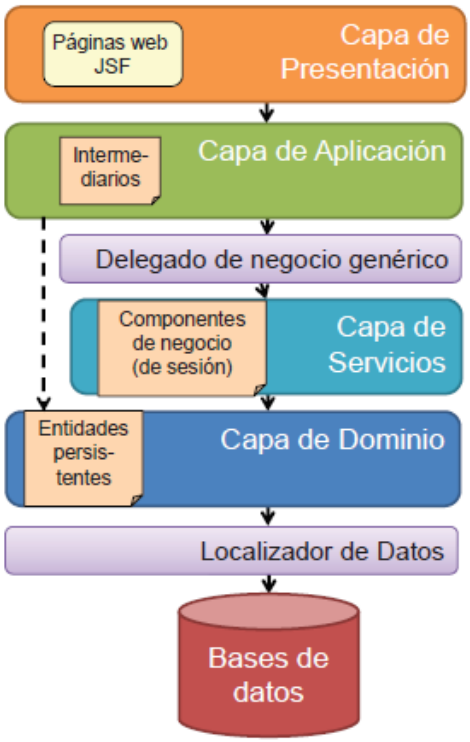

Figura 16: Arquitectura actual de Componentes Empresariales.

Dado que un componente empresarial típicamente está conformado por páginas, controladores, servicios y entidades, para resolver la problemática actual de su dependencia a la tecnología de interfaz-usuario, como JSF, se propone modelar la funcionalidad de cada componente siguiendo el enfoque de la ingeniería dirigida por modelos utilizando el lenguaje ISML, de manera que en lugar de volver a reconstruir cada componente con su capa de presentación en AngularJS (para responder a la arquitectura de referencia actual de la empresa caso de estudio) o en cualquier otra tecnología de presentación en el futuro, se deben seguir las etapas planteadas en la figura 17:

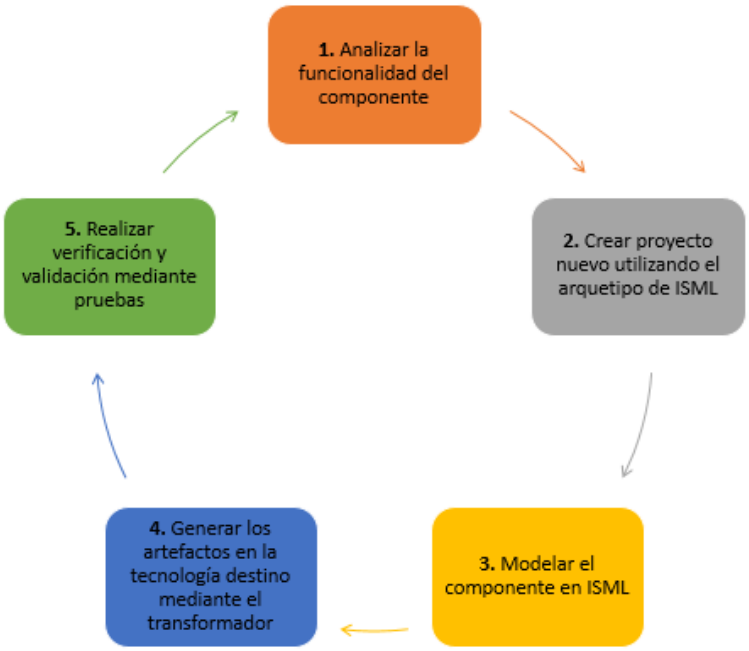

Figura 17: Estrategia para el modelado de Componentes Empresariales.

Como se muestra en la figura, en primer lugar se debe analizar la funcionalidad del componente con el fin de comprender qué tareas o actividades realiza, también identificar qué páginas lo conforman, qué servicios ofrece y qué entidades hacen parte de su modelo de dominio. Una vez se tiene claro aquellos elementos a modelar, es necesario generar un proyecto Maven utilizando el arquetipo de ISML, el cual provee la estructura de directorios donde serán alojadas los elementos del modelo ISML que creará el diseñador de aplicaciones (páginas, controladores, servicios y entidades).

Posteriormente, son generados los artefactos de software en la tecnología destino haciendo uso del generador Gengular; éste se encarga de transformar los modelos ISML en páginas HTML5, controladores AngularJS, servicios REST y entidades Java, colocando dichos artefactos que fueron generados según las reglas y rutas establecidas en el archivo de propiedades (generation.conf.json).

5.2. Estrategia de acople de componentes empresariales a una aplicación generada

Con el fin de resolver la segunda problemática que corresponde al acople de los componentes empresariales para alcanzar un alto grado de automatización, se construyó un asistente denominado **Gengular-Wizard**, cuyo propósito es reducir los tiempos necesarios para poder utilizar los componentes en aplicaciones nuevas y evitarle al arquitecto o desarrollador de software, tener que efectuar tareas manuales para la integración del componente a una aplicación nueva.

En primer lugar, el Gengular-wizard se encarga de crear un nuevo proyecto Maven utilizando el arquetipo generador de la arquitectura SPA y que fue desarrollado en el presente proyecto (Gengular-archetype); para ello el usuario del *wizard* define las coordenadas Maven del proyecto que desea crear y la ruta destino. Una vez el proyecto es generado, se procede a acoplar los componentes empresariales seleccionados por el usuario mediante expresiones regulares y modificando los siguientes archivos:

- **pom.xml**: En este archivo se agregan las dependencias del componente que será acoplado, tanto en el proyecto *frontend* como en el *backend*.

```
<dependency>
      <groupId>co.com.heinsohn.gengular</groupId>
      <artifactId>nombre-componente</artifactId>
      <version>version-componente</version>
      <type>war</type>
</dependency>
```

- **app.js**: En este archivo se agregan las reglas de navegación para el componente. En el caso de AngularJS corresponde a los proveedores de rutas encargados de definir para una ruta, cuál será la página que atenderá la petición y el controlador asociado.

```
$routeProvider.when('/ruta_pagina',{templateUrl:'views/casouso/pagin
a.html', controller: 'NombreControladorCtrl'});
```

- **index.html:** Este archivo consiste en la página principal de la aplicación, donde son agregados los recursos del componente a los que necesita tener acceso la aplicación generada.

```
<script
src="scripts/controllers/casouso/NombreControladorCtrl.js"></script>
```

En la figura 18 se muestra el diagrama de clases del asistente Gengular-wizard; como se observa se utilizó una interfaz que define el contrato para los componentes que se deseen acoplar; dentro del alcance del presente proyecto se realizó la implementación de acopladores para el componente de menú y de notificaciones de Heinsohn, sin embargo, la estrategia sería la misma si se requiere acoplar otros componentes como archivos, auditoría, seguridad, etc.

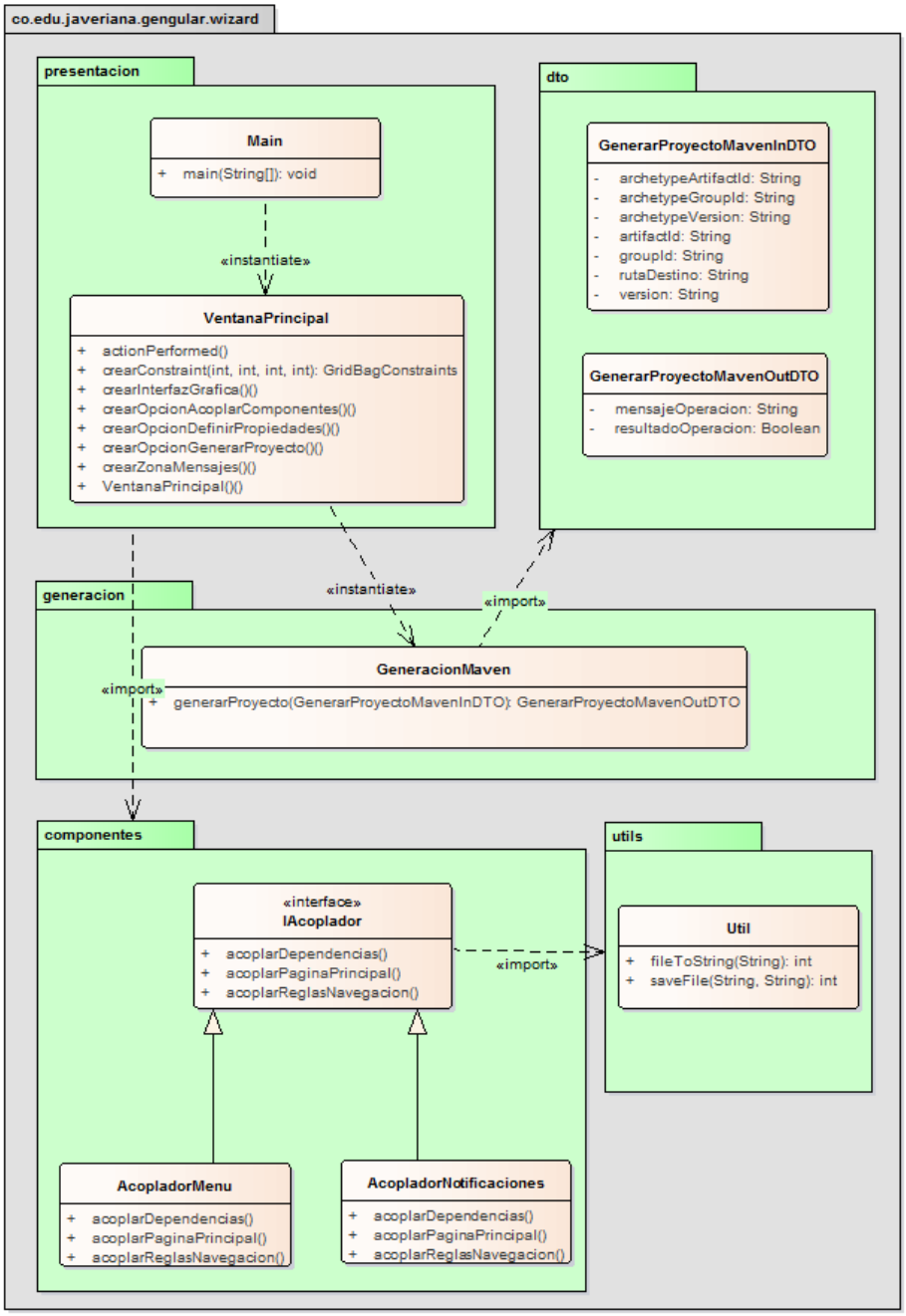

Figura 18: Diagrama de clases Gengular-Wizard.

El asistente Gengular-wizard fue desarrollador en Java SE 7 con framework de presentación Java Swing utilizando elementos gráficos como *JTextField*, *JLabel*, *JCheckBox*, *JButton y JTextArea*. Por otra parte, para la creación de los proyectos Maven desde el asistente, se utilizó las clases Process y ProcessBuilder del paquete java.lang que permiten ejecutar comandos

desde Java. En la figura 19 se presenta la interfaz gráfica del aplicativo y se describen las principales secciones:

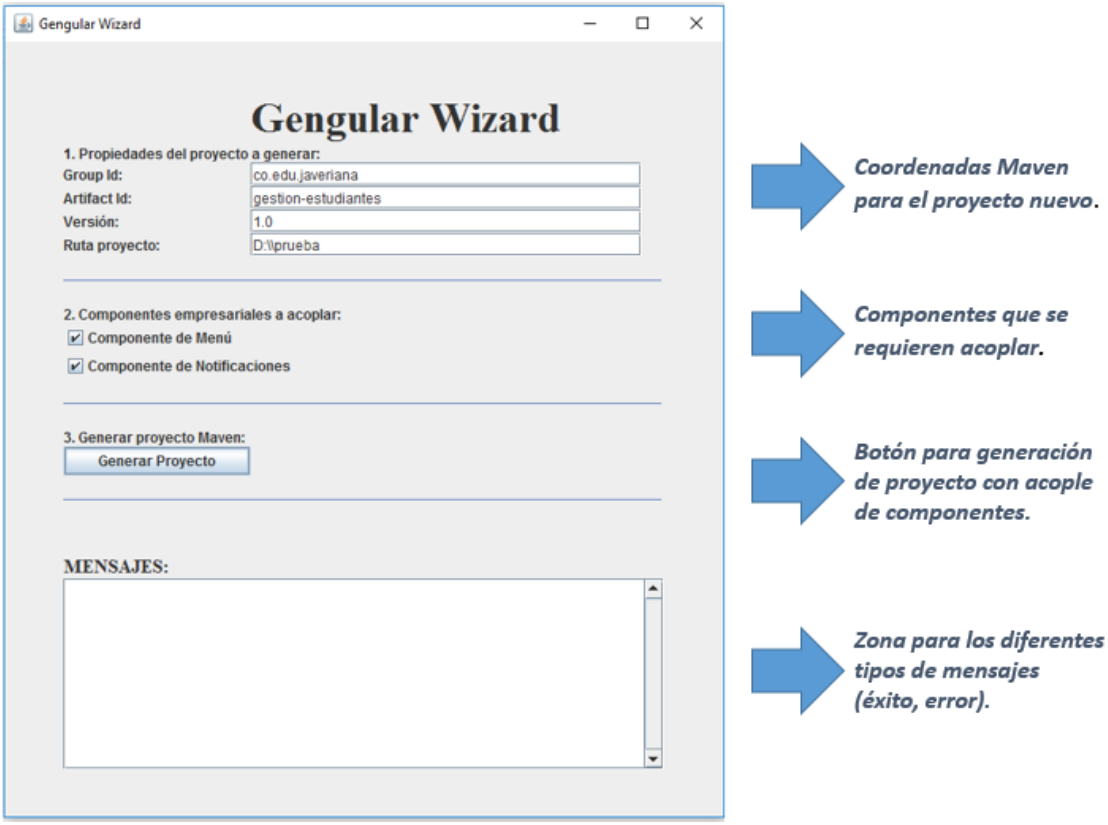

Figura 19: Interfaz gráfica del Gengular-Wizard.

En la sección 1 se definen las coordenadas Maven para el nuevo proyecto, que corresponden al identificador de grupo, de artefacto y la versión. También se establece la ruta donde el proyecto generado será colocado.

En la sección 2 el usuario indica los componentes empresariales que desea acoplar seleccionado los checkbox (botones de comprobación).

En la sección 3 se genera el proyecto Maven teniendo en cuenta la información ingresada por el usuario.

En la sección 4 está la zona de mensajes con el fin de presentar el resultado de las operaciones al usuario.

6. Validación del transformador

Para la validación del transformador se realizaron tres actividades principales; la primera corresponde a la correctitud al generar la aplicación demo Gestión de Estudiantes, la segunda está relacionada con la correctitud al generar el componente menú y el componente de notificaciones y a la capacidad del wizard para acoplar tales componentes al demo. Finalmente, la tercera actividad consistió en realizar la encuesta de satisfacción a usuarios para conocer el grado de cumplimiento de la solución frente a las expectativas de empleados en la empresa caso de estudio.

6.1. Validación al generar la aplicación demo Gestión de Estudiantes

Gestión de Estudiantes es una aplicación conformada por tres casos de uso tipo CRUD y un caso de negocio para gestionar el proceso de inscripción de asignaturas en una institución de educación superior. Permite consultar, crear, editar y eliminar facultades, asignaturas y estudiantes, así como asociar uno o más asignaturas a un estudiante.

A continuación, se muestra un ejemplo de modelo ISML en la aplicación Gestión de Estudiantes y que corresponde a la funcionalidad para consultar facultades. Como se aprecia, consiste en un panel que contiene una tabla de datos que a su vez tiene dos secciones: el encabezado con los títulos de la tabla y el cuerpo con cada una de las filas y los botones para realizar la edición o eliminación del registro. También incluye el botón para ir a la página de creación de una facultad.

```
page Consultar_Facultad(Collection<Facultad> listaFacultad) controlledBy FacultadCtrl  {

    Panel("GESTIONAR FACULTADES") {
        DataTable("Collection<Facultad>", null) {
            header : {
                Label("Codigo");
                Label("Nombre");
                Label("Descripcion");
                Label("Editar");
                Label("Eliminar");
            }
            body :
                for(Facultad facultad in listaFacultad) {
                    Label(facultad.codigo);
                    Label(facultad.nombre);
                    Label(facultad.descripcion);
                    Button("Editar") -> irEditarFacultad(facultad);
                    Button("Eliminar") -> eliminarFacultad(facultad);
                }
        }

        Button("Crear Facultad") -> irCrearFacultad(new Facultad);
    }

}
```

Figura 20: Página ISML de la aplicación Gestión Estudiantes.

La página ISML está soportada por un controlador ISML denominado FacultadCtrl que se encarga de invocar las operaciones de consulta, creación, edición y eliminación a través del servicio de persistencia.

```
controller FacultadCtrl {

    has Persistence<Facultad> persistence;

    irEditarFacultad(Facultad facultad) {
        show Editar_Facultad(facultad);
    }

    editarFacultad(Facultad facultad) {
        persistence.save(facultad);
    }

    irCrearFacultad(Facultad facultad) {
        show Crear_Facultad(facultad);
    }

    crearFacultad(Facultad facultad) {
        persistence.create(facultad);
    }

    eliminarFacultad(Facultad facultad) {
        persistence.remove(facultad);
    }
```

Figura 21: Controlador ISML de la aplicación Gestión Estudiantes.

El controlador ISML FacultadCtrl utiliza el servicio de persistencia el cual provee las operaciones requeridas para interactuar y gestionar cualquier entidad, pero dado que este servicio común define solamente la interfaz para dichas operaciones, fue necesario construir un servicio en AngularJS que realizara la implementación. En la figura 22 se muestra el servicio de persistencia desarrollado.

```
app.factory('persistence', ['$http', function($http) {

    var persistence = {};

    persistence.findAll = function(resource) {
        return $http.get(urlBase + resource);
    };

    persistence.find = function(resource, id) {
        return $http.get(urlBase + resource + '/' + id);
    };

    persistence.create = function(resource, object) {
        return $http.post(urlBase + resource, object);
    };

    persistence.save = function(resource, object) {
        return $http.put(urlBase + resource, object);
    };

    persistence.remove = function(resource, id) {
        return $http.delete(urlBase + resource + '/' + id);
    };
```

Figura 22: Servicio de persistencia para la aplicación Gestión Estudiantes.

Finalmente, también se realiza la definición de una entidad ISML llamada Facultad (ver figura 23), la cual contiene los atributos de código, nombre y descripción. Las entidades son ampliadas a través del transformador de entidades que se encarga de agregar métodos como los getters/setters, toString, equals y hashCode.

```
entity Facultad {

    String codigo;
    String nombre;
    String descripcion;

}
```

Figura 23: Entidad ISML de la aplicación Gestión Estudiantes.

Una vez que los modelos ISML son transformados a la tecnología destino mediante Gengular, utilizando los generadores de páginas, controladores, servicios y entidades se tendrá una aplicación como la que se muestra a continuación:

Figura 24: Aplicación generada por el transformador Gengular.

6.2. Validación al generar el componente menú y el componente notificaciones y realizar su acople al demo

El componente de Menú desarrollado agrega a una aplicación HTML5-AngularJS-JEE7 los elementos requeridos para poder manejar dinámicamente el menú del sistema. A partir de

entidades guardadas en la base de datos, permite configurar submenús, módulos y casos de uso que van a ser invocados como elementos de menú.

Figura 25: Componente de menú desarrollado

El componente de menú incluye 13 páginas de parametrización soportadas por 5 controladores que se encargan de la administración de módulos, casos de uso, permisos y perfiles. Dichas funcionalidades fueron modeladas en ISML y luego generadas a tecnología específica mediante el transformador Gengular.

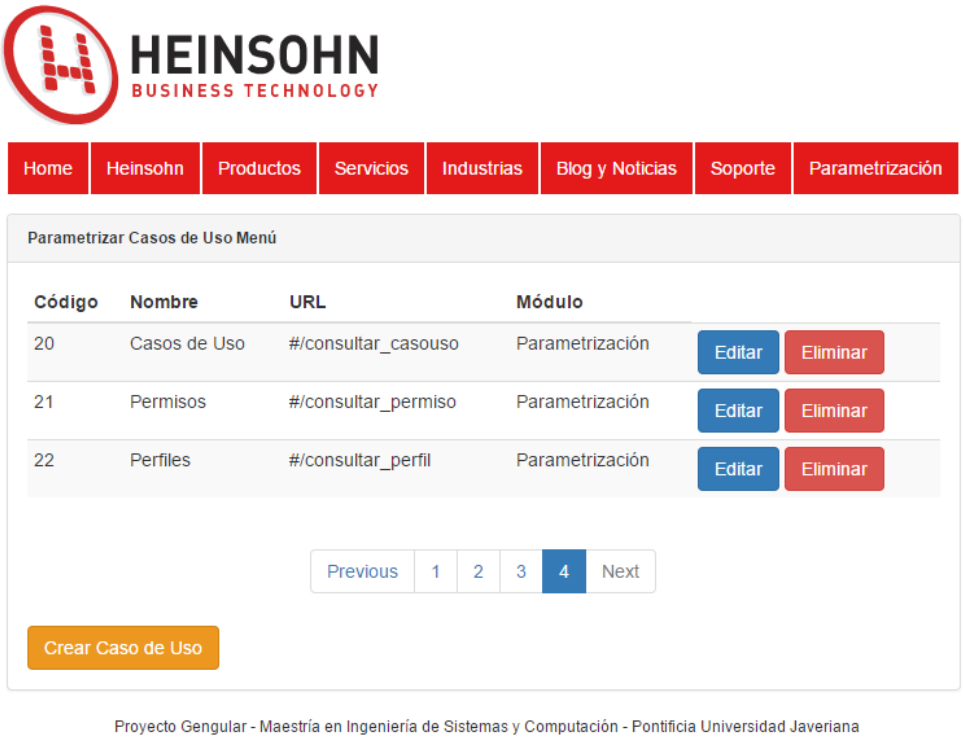

Figura 26: Parametrización Casos de Uso Menú.

El segundo componente desarrollado corresponde a notificaciones, permite desde una aplicación HTML5-AngularJS-JEE7 enviar emails masivos a grupos de destinatarios ofreciendo una consola web para poder indicar los mensajes a enviar, los destinatarios, configurar la autenticación ante servidores SMTP y de correo Microsoft Exchange.

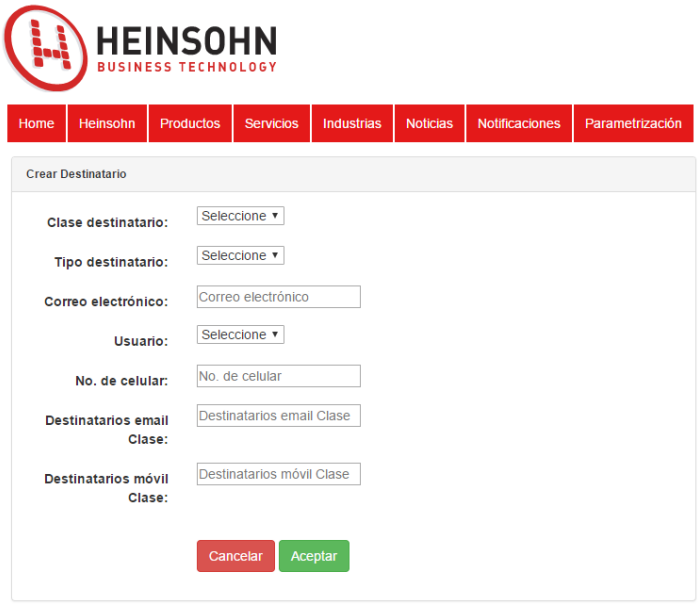

Figura 27: Componente de notificaciones desarrollado.

Para verificar el acople de los componentes empresariales menú y notificaciones a la aplicación demo Gestión de Estudiantes, se utiliza el asistente Gengular-Wizard con las configuraciones de la figura 28.

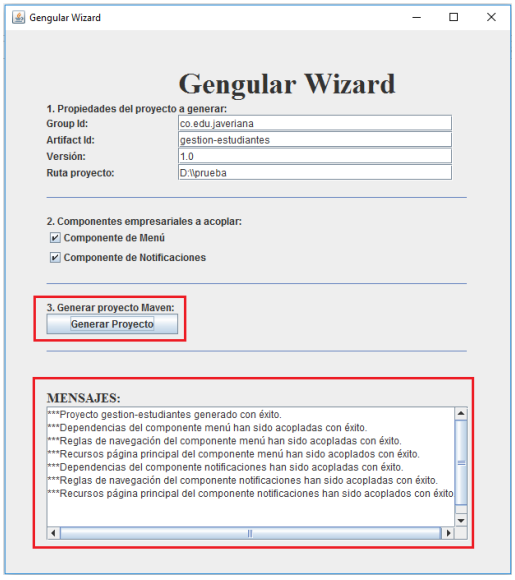

Figura 28: Acople de componentes utilizando el wizard.

6.3. Encuesta de validación

La tercera actividad de validación, corresponde a la definición de una encuesta para conocer la opinión de usuarios en la empresa caso de estudio en relación al potencial de automatización y reutilización del presente proyecto. La encuesta contiene 10 preguntas como se muestra en la tabla 18 y según consta en los anexos I y II, la validación arrojó muy buenos resultados por parte de dos Arquitectos de Software de Heinsohn, quienes consideran elementos de innovación en Gengular luego que conocer la solución.

Encuesta de satisfacción proyecto Gengular	**Encuesta I**	**Encuesta II**
1. ¿Qué tan útil le parece la propuesta del presente proyecto para la preservación de los componentes empresariales de Heinsohn?	Excelente	Excelente
2. ¿Considera que este proyecto puede contribuir a mejorar la productividad en los desarrollos de la compañía?	Excelente	Bueno
3. ¿Qué tan alineadas están las tecnologías utilizadas por el transformador (AngularJS-HTML5-JEE7) frente a las utilizadas en Heinsohn?	Excelente	Excelente
4. Respecto al asistente Gengular Wizard, ¿Considera que puede ayudar a reducir los tiempos de acople de los componentes empresariales a un nuevo proyecto?	Bueno	Excelente
5. ¿Considera que la estructura para un proyecto definida por el arquetipo Gengular Archetype es adecuada (frontend y backend)?	Excelente	Bueno
6. ¿Qué tan fácil le parece programar transformadores en Xtend?	Excelente	Bueno
7. ¿Qué tan fácil le parece modificar transformadores en Xtend?	Bueno	Excelente
8. ¿Qué tan fácil le parece modelar un componente utilizando el ambiente ISML?	Excelente	Excelente
9. ¿Qué tan fácil de parecer generar una aplicación a partir de modelos en ISML?	Bueno	Bueno
10. ¿Qué tal le parece la calidad de las interfaces generadas por el generador (Hoja de estilos Bootstrap)?	Excelente	Excelente

Tabla 18: Encuesta de satisfacción a usuarios.

De acuerdo con las respuestas de los usuarios, la estrategia para la preservación de componentes empresariales es adecuada y el proyecto puede contribuir a mejorar la productividad en la compañía. Además consideran que las tecnologías utilizadas por el transformador se encuentran alineadas con la arquitectura de referencia de la empresa (AngularJS-HTML5-JEE7).

Como sugerencias, proponen que sería interesante construir una versión del ambiente ISML y del transformador que funcione en la nube, para así tener entornos colaborativos de trabajo y evitar la instalación de las herramientas en cada equipo de los desarrolladores.

7. Resultados obtenidos

Para determinar el esfuerzo que el transformador Gengular permite minimizar, en términos del número de líneas del código (LOC) que se deben escribir en una aplicación SPA frente a las que componen el modelo ISML de la aplicación Gestión de Estudiantes, en la tabla 19 se muestra el total de líneas codificadas en ISML y el total que son generadas en las tecnologías destino (HTML5-AngularJS-JEE7) luego de utilizar el transformador.

Modelo ISML Gestión de Estudiantes		**Aplicación generada Gestión de Estudiantes**	
Componente ISML	**Total líneas de código por Componente ISML**	**Tecnología destino**	**Total líneas de código en tecnología destino**
Páginas	269	HTML5 + Bootstrap	633
Controladores	192	AngularJS	361
Servicios	116	Java Empresarial	1608
Entidades	45	Java Empresarial	652
TOTAL	**622**	**TOTAL**	**3254**

Tabla 19: Medición de código generado aplicación Gestión de Estudiantes.

A partir de los totales obtenidos en la tabla 19, se puede estimar que la proporción de líneas de código entre el modelo ISML y la aplicación generada es de 1:5, de manera que el transformador Gengular puede contribuir a reducir los tiempos y esfuerzos de desarrollo. En la figura 29 se presenta el análisis de los resultados obtenidos con la medición.

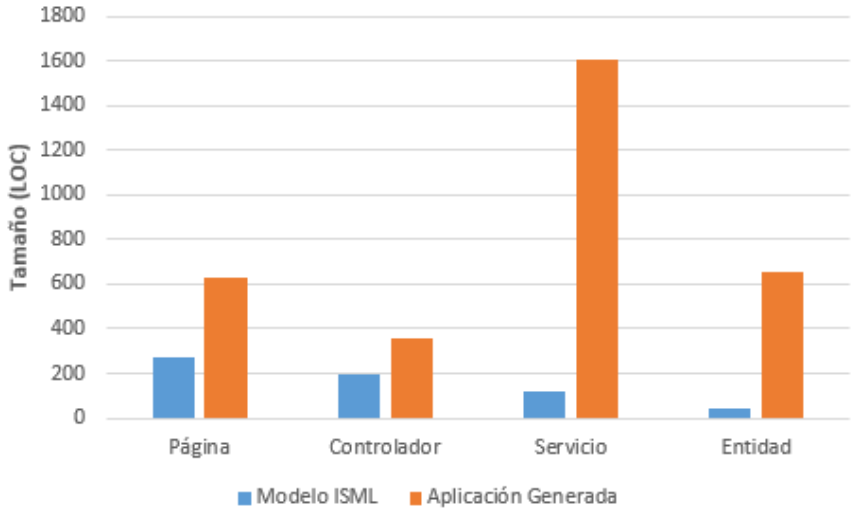

Figura 29: Análisis de código generado aplicación Gestión de Estudiantes.

V -CONCLUSIONES Y TRABAJOS FUTUROS

1. Conclusiones

En el presente proyecto se construyó el transformador Gengular el cual permite generar aplicaciones en las tecnologías HTML5, AngularJS y JEE7 a partir de modelos definidos en el lenguaje independiente de plataforma ISML. Asimismo, fueron generadas una aplicación demo y dos componentes empresariales que se acoplan a la aplicación utilizando el transformador. Gengular puede contribuir a reducir los tiempos y esfuerzo en el desarrollo de aplicaciones ya que permite crear código en una relación o proporción de 1:5.

Por otra parte, también se logró generar aplicaciones que se visualizan de manera adecuada en dispositivos móviles como *tablets* o *smartphones* al hacer uso de la hoja de estilos Bootstrap, encargada de adaptar los elementos de presentación al tamaño del navegador web donde de renderizan.

Adicionalmente, se definió e implementó una estrategia para lograr el acople de los componentes empresariales a una aplicación nueva mediante la creación de un asistente denominado Gengular-Wizard, el cual hace uso del arquetipo maven Gengular-Archetype y que corresponde a la arquitectura de referencia de la empresa caso de estudio. El asistente permitió acoplar los componentes mediante el uso de expresiones regulares que incorporan las dependencias de proyecto, reglas de navegación y recursos necesarios para la ejecución de cada componente.

Con el presente proyecto se puede contribuir a la automatización y reutilización en proyectos de software que realicen empresas de base tecnológica en Colombia, y además lograr la preservación de activos tecnológicos, como lo son componentes empresariales, frente a la constante evolución de tecnologías y plataformas.

2. Trabajos Futuros

Como trabajo futuro se propone modelar y generar los componentes empresariales pendientes para manejo de archivos, auditoría, elaboración de oficios, entre otros, siguiendo la estrategia propuesta en el presente proyecto. De la misma manera, ampliar las capacidades del asistente Gengular-Wizard para incorporar dichos componentes.

A partir de las tendencias de la industria TI para 2017 y 2018, se propone construir un transformador ISML para el framework Angular 2, dado que Google ha restructurado aspectos importantes del mismo en aras de mejorar el desempeño y la mantenibilidad en los desarrollos. También se propone evaluar la implementación de un transformador para el framework ReactJS, con el cual son construidas interfaces gráficas en aplicaciones como Facebook e Instagram. Con estos nuevos transformadores propuestos se probaría si realmente los modelos de los componentes empresariales se preservan, generando a partir de ellos, sin modificarlos, con los nuevos transformadores.

De acuerdo a los comentarios recibidos en las encuestas por parte de los usuarios, se evalúa construir una versión del ambiente ISML y del transformador que funcione en la nube, para así tener entornos colaborativos de trabajo.

Finalmente, se plantea modificar el lenguaje ISML para que permita definir los siguientes atributos:

- En la creación de un widget de botón, poder indicar el tipo de botón que corresponde en el formulario (botón cancelar, botón aceptar, botón eliminar, etc).

- En la creación de un widget de texto, indicar si es un campo de formulario obligatorio u opcional.

- Agregar un widget que represente un campo para cargar un archivo y las propiedades generales que puede tener el mismo, como extensiones permitidas, tamaño máximo habilitado, etc.

VI -REFERENCIAS

[1] Heinsohn Business Technology, Available at: http://www.heinsohn.com.co/. Last accessed on 2017-01-29.

[2] M. C. Franky, J. Pavlich-Mariscal, M. C. Acero, A. Zambrano, J. C. Olarte, J. L. Camargo and J. N. Pinzon, "ISML-MDE: A Practical Experience of Implementing a Model Driven Environment in a Software Development Organization", International Journal of Web Information Systems, 2016(04) issue.

[3] M. C. Franky, J. Pavlich-Mariscal, J. C. Olarte, M. C. Acero, A. Zambrano, J. L. Camargo and J. N. Pinzon, "ISML: A language and MDE environment to model and generate web applications with integration of existing components," 2015 10th Colombian Computing Conference, 10CCC 2015, pp. 100-107, / 11 / 20 /, 2015.

[4] Java Server Faces (JSF), Available at: http://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html/. Last accessed on 2017-01-29.

[5] W3C, HTML5, Available at: https://www.w3.org/TR/html5/. Last accessed on 2017-01-29.

[6] Google, AngularJS, Available at: https://angularjs.org/. Last accessed on 2017-01-29.

[7] Oracle, Java EE at a Glance, Available at: http://www.oracle.com/technetwork/java/javaee/overview/index.html Last accessed on 2017-01-29.

[8] M. Verano, L. Salamanca, R. Casallas, M. Villamizar, O. Garcés, L. Ochoa, H. Castro, A. Zambrano, C. Valencia and S. Gil, "Re-architecting a JEE on-premise web application to deploy it in the cloud," 2015 IEEE Globecom Workshops, GC Wkshps 2015 - Proceedings, / 02 / 18 /, 2016.

[9] M. Mikowski and J. Powell, Single Page Web Applications: JavaScript end-to-end. Manning Publications, 2013.

[10] D. C. Schmidt, Model-driven engineering, Computer-Ieee Computer Society-, vol. 39, pp. 25, 2006.

[11] M. Brambilla, J. Cabot and M. Wimmer, Model-Driven Software Engineering in Practice. San Rafael, California] Morgan & Claypool Publishers c2012, 2012.

[12] Object Management Group (OMG), Available at: http://www.omg.org/. Last accessed on 2017-01-29.

[13] S. J. Mellor and R. M. Soley, MDA Distilled. Principles of Model-Driven Architecture. Boston, Massachusetts; San Francisco, California; New York Addison-Wesley c2004, 2004.

[14] K. Beck and E. Gamma, Extreme Programming Explained. Embrace Change. Boston, Massachusetts Addison-Wesley c2000, 2000.

[15] K. Beck, M. Fowler, J. Kohnke and T. DeMarco, Planning Extreme Programming. Boston, Massachusetts Addison-Wesley c2001, 2001.

[16] Sonatype Company and M. K. Loukides, Maven. the Definitive Guide. Sebastopol, California; Beijing O'Reilly c2008; 1st ed, 2008.

[17] ISML-MDE, Available at: https://github.com/jpavlich/ISML-MDE/. Last accessed on 2017-01-29.

[18] R. Grady and D. Caswell, Software Metrics: Establishing a Company-Wide Program. Mountain View, California: Prentice Hall, 1987, pp. 275.

[19] P. B. Darwin and P. Kozlowski, AngularJS Web Application Development. Birmingham: Packt Publishing, 2013.

[20] L. Ruebbelke and B. Ford, AngularJS in Action. Manning Publications, 2015, pp. 5-10.

[21] E. Freeman and E. Robson, Head First HTML5 Programming: Building Web Apps with JavaScript. O'Reilly Media, Inc., 2011.

[22] J. N. Robbins, HTML5 Pocket Reference. O'Reilly Media, Inc., 2013.

[23] N. Shah and G. Balda, HTML5 Enterprise Application Development. Birmingham: Packt Publishing, 2013, pp. 23-24.

[24] M. Pilgrim, HTML5: Up and Running: Dive into the Future of Web Development. O'Reilly Media, Inc., 2010.

[25] V. Pterneas, Getting Started with HTML5 WebSocket Programming. Packt Publishing, 2013, pp. 15-16.

[26] WYSIHTML5: Open Source Rich Text Editor, Available at: http://xing.github.io/wysihtml5/. Last accessed on 2017-02-17.

[27] A. Gupta, Java EE 7 Essentials. Cambridge, Massachusetts O'Reilly c2013, 2013.

[28] Java Specification Request (JSR) 342, Available at: https://www.jcp.org/en/jsr/detail?id=342. Last accessed on 2017-02-19.

[29] A. Goncalves, Beginning Java EE 7. Berkeley, California Apress c2013, 2013.

[30] D. R. Heffelfinger, Java EE 7 with GlassFish 4 Application Server. Birmingham, UK: Packt Publishing, 2014.

[31] E. Jendrock, The Java EE 7 Tutorial. Upper Saddle River, New Jersey Addison-Wesley c2014; Fifth edition, 2014.

[32] P. A. Pilgrim, Java EE 7 Developer Handbook: Develop Professional Applications in Java EE 7 with this Essential Reference Guide. Birmingham: Packt Publishing, 2013, pp. 129-132.

[33] A. Leff and J. T. Rayfield, "Web-application development using the Model/View/Controller design pattern," Proceedings Fifth IEEE International Enterprise Distributed Object Computing Conference, pp. 118, 01, 2001.

[34] N. C. Zakas, Professional JavaScript for Web Developers. Indianapolis, IN: Wrox, 2009.

[35] E. Scott, SPA Design and Architecture: Understanding Single Page Web Applications. Manning Publications, 2015.

[36] Miško Hevery, Available at: http://misko.hevery.com/about/. Last accessed on 2017-03-02.

[37] O. El Beggar, B. Bousetta and T. Gadi, Code Generation by Applying MDA. Saarbrücken LAP Lambert Academic Publishing 2013, 2013.

[38] Ó. Pastor and J. C. Molina, Model-Driven Architecture in Practice. a Software Production Environment Based on Conceptual Modeling. Berlin New York Springer c2007, 2007.

[39] A. G. Kleppe, J. B. Warmer and W. Bast, MDA Explained. the Model Driven Architecture: Practice and Promise. New York Addison-Wesley c2003, 2003.

[40] R. Bharathan, Apache Maven Cookbook. Birmingham, UK: Packt Publishing Ltd, 2015.

[41] Jason van Zyl, Available at: https://www.crunchbase.com/person/jason-van-zyl#/entity. Last accessed on 2017-03-08.

[42] L. Anardu, Maven Build Customization. Birmingham, UK: Packt Publishing, 2014.

[43] Maven Archetype, Available at: https://maven.apache.org/guides/introduction/introduction-to-archetypes.html. Last accessed on 2017-03-08.

[44] Mark Otto, Available at: http://markdotto.com/about/. Last accessed on 2017-03-27.

[45] Jacob Thornton, Available at: https://github.com/fat. Last accessed on 2017-03-27.

[46] Twitter, Inc., Available at: https://twitter.com/. Last accessed on 2017-03-27.

[47] J. Spurlock, Bootstrap: Responsive Web Development. O'Reilly Media, 2013.

[48] A. Shenoy and U. Sossou, Learning Bootstrap. Birmingham, UK: Packt Publishing, 2014.

[49] D. Cochran and I. Whitley, Bootstrap Site Blueprints. Birmingham, UK: Packt Publishing, 2014.

[50] Steve Yen, Available at: http://connect15.couchbase.com/speaker/steve-yen-2/. Last accessed on 2017-03-27.

[51] Java Applets, Available at: https://docs.oracle.com/javase/tutorial/deployment/applet/. Last accessed on 2017-03-27.

[52] Adobe Flash, Available at: http://adobe.com/support/documentation/es/flashplayer/. Last accessed on 2017-03-27.

[53] Microsoft Silverlight, Available at: https://www.microsoft.com/silverlight/. Last accessed on 2017-03-27.

[54] N. Schutta and R. Asleson, Foundations of Ajax. Apress, 2006.

[55] B. Smith, Beginning JSON. Apress, 2015.

[56] Informe de caracterización del sector de software y tecnologías de la información en Colombia Fedesoft, Available at: http://fedesoft.org/noticias-fedesoft/disponible-estudio-de-caracterizacion-de-la-industria-del-software-colombiano/. Last accessed on 2017-04-06.

[57] Informes del Sector TI MinTIC, Available at: http://colombiatic.mintic.gov.co/602/w3-propertyvalue-715.html. Last accessed on 2017-04-06.

[58] Observatorio Laboral para la Educación, Graduados Colombia, Available at: http://www.graduadoscolombia.edu.co/html/1732/articles-334303_documento_tecnico_2013.pdf. Last accessed on 2017-04-06.

[59] Panorama TIC, Comportamiento del sector TIC en Colombia, Available at: https://colombiatic.mintic.gov.co/602/articles-8917_panoranatic.pdf. Last accessed on 2017-04-08.

[60] K. C. Palomino, Estudio del comportamiento de la industria del software en Colombia ante escenarios de capacidades de innovación y ventajas comparativas por medio de dinámica de sistemas, Universidad Nacional de Colombia, 2011.

[61] J. Bocanegra, J. A. Pavlich-Mariscal and A. C. Ramos, "MiDAS: A model-driven approach for adaptive software." in Webist, 2015, pp. 281-286.

[62] J. A. Andrade Mayorga, Lityerses: Transformador para generar aplicaciones móviles con componentes adaptativos a partir de un modelo independiente de plataforma, Bogotá: Pontificia Universidad Javeriana, 2016.

[63] F. S. Franco Hernández, Anchurus-GEN: Generador de código PHP a partir de modelos ISML, Bogotá: Pontificia Universidad Javeriana, 2015.

[64] J. C. Olarte Abello, ZOE-GEN: Un transformador para facilitar la generación de aplicaciones basado en modelos, Bogotá: Pontificia Universidad Javeriana, 2015.

[65] F.S. Aguillón Martínez and M.A. Mateus Gómez, Automatización del desarrollo de aplicaciones web mediante el enfoque MDA-MDE, Bogotá: Pontificia Universidad Javeriana, 2014.

[66] Retos y Consideraciones para llevar Aplicaciones Web / Java a la Nube, Available at: https://sistemas.uniandes.edu.co/images/forosisis/foros/fcc10/devOps.pdf. Last accessed on 2017-04-08

[67] 5 Best JavaScript Frameworks in 2017, Available at: https://da-14.com/blog/5-best-javascript-frameworks-2017. Last accessed on 2017-04-08.

[68] A. Fedosejev, React. Js Essentials. Birmingham, UK: Packt Publishing, 2015.

[69] Vue.js, Available at: https://vuejs.org/v2/guide/. Last accessed on 2017-04-08.

[70] J. Cravens and T. Q. Brady, Building Web Apps with Ember. O'Reilly Media, Inc., 2014.

[71] I. Strack, Getting Started with Meteor.Js JavaScript Framework. Packt Publishing Ltd, 2015.

[72] J. Koetsier, Evaluation of JavaScript frame-works for the development of a web-based user interface for Vampires, Universiteit Van Amsterdam, 2016.

[73] I. Iso, "IEC 25010: 2011" Systems and Software Engineering-Systems and Software Quality Requirements and Evaluation (SQuaRE)-System and Software Quality Models, 2011.

VII -ANEXOS

Anexo I - Encuesta de validación:

Trabajo de Grado	*Gengular: Hacia la automatización de aplicaciones empresariales bajo el paradigma de arquitectura SPA y el enfoque MDE.*	Pontificia Universidad JAVERIANA Bogotá

Pregunta	Excelente	Bueno	Regular	Deficiente	Malo
1. ¿Qué tan útil le parece la propuesta del presente proyecto para la preservación de los componentes empresariales de Heinsohn?	X				
2. ¿Considera que este proyecto puede contribuir a mejorar la productividad en los desarrollos de la compañía?	X				
3. ¿Qué tan alineadas están las tecnologías utilizadas por el transformador (AngularJS-HTML5-JEE7) frente a las utilizadas en Heinsohn?	X				
4. Respecto al asistente Gengular Wizard, ¿Considera que puede ayudar a reducir los tiempos de acople de los componentes empresariales a un nuevo proyecto?		X			
5. ¿Considera que la estructura para un proyecto definida por el arquetipo Gengular Archetype es adecuada (frontend y backend)?	X				
6. ¿Qué tan fácil le parece programar transformadores en Xtend?	X				
7. ¿Qué tan fácil le parece modificar transformadores en Xtend?		X			
8. ¿Qué tan fácil le parece modelar un componente utilizando el ambiente ISML?	X				
9. ¿Qué tan fácil le parece generar una aplicación a partir de modelos en ISML?		X			
10. ¿Qué tal le parece la calidad de las interfaces generadas por el generador (Hoja de estilos Bootstrap)?	X				

COMENTARIOS:

Seria interesante que el transformador se encontiara en la nube para entornos colaborativos

Nombre:	Luis Miguel Rojas
Empresa:	Heinsohn Business
Cargo:	Arquitecto
Email:	lrojas@heinsohn.com.co
Firma:	LuisMiguelRojas

Anexo II - Encuesta de validación:

Trabajo de Grado	***Gengular:** Hacia la automatización de aplicaciones empresariales bajo el paradigma de arquitectura SPA y el enfoque MDE.*	Pontificia Universidad JAVERIANA Bogotá

Pregunta	**Excelente**	**Bueno**	**Regular**	**Deficiente**	**Malo**
1. ¿Qué tan útil le parece la propuesta del presente proyecto para la preservación de los componentes empresariales de Heinsohn?	X				
2. ¿Considera que este proyecto puede contribuir a mejorar la productividad en los desarrollos de la compañía?		X			
3. ¿Qué tan alineadas están las tecnologías utilizadas por el transformador (AngularJS-HTML5-JEE7) frente a las utilizadas en Heinsohn?	X				
4. Respecto al asistente Gengular Wizard, ¿Considera que puede ayudar a reducir los tiempos de acople de los componentes empresariales a un nuevo proyecto?	X				
5. ¿Considera que la estructura para un proyecto definida por el arquetipo Gengular Archetype es adecuada (frontend y backend)?		X			
6. ¿Qué tan fácil le parece programar transformadores en Xtend?		X			
7. ¿Qué tan fácil le parece modificar transformadores en Xtend?	X				
8. ¿Qué tan fácil le parece modelar un componente utilizando el ambiente ISML?	X				
9. ¿Qué tan fácil de parecer generar una aplicación a partir de modelos en ISML?		X			
10. ¿Qué tal le parece la calidad de las interfaces generadas por el generador (Hoja de estilos Bootstrap)?	X				

COMENTARIOS:

Nombre:	Manuel Vargas
Empresa:	Heinsohn Business Technology
Cargo:	Arquitecto de Software
Email:	mvargas@heinsohn.com.co
Firma:	