UNIVERSIDAD CATÓLICA "NUESTRA SEÑORA DE LA ASUNCIÓN"
FACULTAD DE CIENCIAS Y TECNOLOGÍA

PROYECTO FINAL DE CARRERA

INGENIERÍA INFORMÁTICA

Un Enfoque MDD para el desarrollo de RIA

Alumno:
Lic. Guido NUÑEZ

Tutores:
Ing. Magalí GONZÁLEZ
Ph.D. Luca CERNUZZI

Enero, 2017
Asunción, Paraguay

Agradecimientos

Este Proyecto es financiado por el CONACYT través del Programa PROCIENCIA con recursos del Fondo para la Excelencia de la Educación e Investigación – FEEI del FONACIDE. Este trabajo ha sido desarrollado bajo el proyecto “Mejorando el proceso de desarrollo de software: propuesta basada en MDD” (14-INV-056).

Mis agradecimientos personales a:

Dios, por haberme guiado, acompañado y fortalecido a lo largo de la carrera.

Mis padres Marina y Carlos, por el amor y apoyo incondicional de siempre, sus enseñanzas me permitieron llegar donde estoy hoy.

Mi pareja Ivette, mi compañera de vida, por los tantos momentos vividos, logros realizados y por realizar.

Mis tutores Magalí, Nathalie y Luca, por sus intereses en mi formación y la buena predisposición a lo largo del proyecto y la carrera.

Mis compañeros de la “La Pecera”, por la amistad y el buen compañerismo del día a día.

Tabla de contenido

1. Introducción

La web ha ido evolucionando continuamente con el pasar de los años. Anteriormente las aplicaciones web estaban caracterizadas por un procesamiento realizado del lado del servidor, donde el cliente solo se encargaba de solicitar y desplegar contenido estático. Esto generaba interacciones muy limitadas con el usuario, necesidad de recargar toda la página para realizar la navegación, tiempos de respuesta muy largos y dificultad de desplegar animaciones y contenidos multimedia.

Para enfrentarse a estas limitaciones han aparecido las Aplicaciones Enriquecidas de Internet o RIA (del inglés Rich Internet Applications) [1]. Este tipo de aplicaciones ha permitido mejorar notablemente la experiencia del usuario, principalmente en cuanto a aspectos de presentación e interacción. Las RIA poseen cuatro características principales que son la distribución de datos, la distribución de la lógica de negocios, la comunicación asíncrona entre cliente y servidor, y la mejora de la interfaz de usuario. Se han propuesto muchas tecnologías para facilitar el desarrollo de RIA y las mismas han logrado incrementar la productividad del desarrollador, sin embargo, presentan desventajas como la carencia de instrumentos de alto nivel que abstraigan detalles de implementación, la propensión a errores, la falta de consideración del ciclo de vida completo del software, entre otras más [2] [3].

Los enfoques de Desarrollo Dirigido por Modelos o MDD (del inglés Model Driven Development) [4] ofrecen soluciones a estas limitaciones, permitiendo crear aplicaciones a través de la especificación de modelos y la posterior generación de código a partir de ellos, logrando así un mayor nivel de abstracción. Es conveniente que estos enfoques adopten un estándar como la Arquitectura Dirigida por Modelos o MDA (del inglés Model Driven Architecture) [5] para dirigir el proceso de desarrollo y garantizar una mayor interoperabilidad y portabilidad entre sistemas.

En los últimos años, han surgido numerosos enfoques MDD para la implementación de RIA, muchos de ellos extendiendo enfoques para el desarrollo de aplicaciones web tradicionales. MoWebA [6] es un enfoque MDD para el desarrollo de aplicaciones web que adopta el estándar MDA y que permite la definición de un Modelo Específico de Arquitectura o ASM (del inglés Architecture Specific Model) para la implementación de extensiones. En [7] se ha elaborado una extensión para la arquitectura RIA contemplando aspectos de la capa de presentación. Otra extensión conveniente sería abarcar aspectos de las capas de datos, lógica de negocios y comunicación de las RIA.

El problema que se busca resolver en este trabajo es la necesidad de mejorar la experiencia del usuario en la web, potenciando el desarrollo de RIA y dotando a las aplicaciones web tradicionales con características propias de las RIA. Esta necesidad surge debido a que los usuarios de la web se han acostumbrado a mejores apariencias gráficas, patrones de interacción avanzados, contenidos multimedia y menores tiempos de respuesta. En consecuencia, hoy en día se solicita a los desarrolladores proporcionar interacciones ricas en casi cualquier aplicación web.

Para atacar el problema, este trabajo tiene como objetivo general definir un enfoque MDD que extienda el enfoque MoWebA y se centre en el desarrollo de RIA con características de distribución de datos, distribución de lógica de negocios y comunicación asíncrona entre cliente y servidor.

Este objetivo comprende los siguientes objetivos específicos:

- Analizar los principales enfoques MDD para el desarrollo de RIA que extienden enfoques para el diseño y desarrollo de aplicaciones web tradicionales.
- Proponer metamodelos para la generación de un ASM y reglas de transformación para la generación de código que contemplen el desarrollo de RIA con características de distribución de datos, distribución de lógica de negocios y comunicación asíncrona entre cliente y servidor.
- Realizar un análisis de la propuesta a partir de un caso de prueba.

Este proyecto de fin de carrera ha sido desarrollado con el soporte del CONACYT, en el contexto del proyecto "Mejorando el proceso de desarrollo de software: propuesta basada en MDD", que busca explorar los beneficios de la aplicación de MDD en el desarrollo de aplicaciones software de buena calidad, utilizando tecnologías actuales.

Este documento se encuentra organizado de la siguiente manera:

- El capítulo 2 introduce la definición de RIA, junto a sus características principales y tecnologías para su desarrollo. Posteriormente presenta fundamentos acerca de MDD y MDA. Finalmente presenta el enfoque MoWebA, detallando el proceso de modelado, el proceso de transformación y los metamodelos y perfiles a extender en esta propuesta.
- El capítulo 3 presenta un análisis de los enfoques MDD para el desarrollo de RIA existentes en la literatura. Para ello seguimos los pasos de un Mapeo Sistemático de la Literatura o SMS (del inglés Systematic Mapping Study), desglosando el proceso en las fases planificación, ejecución y resultados.
- El capítulo 4 describe la propuesta de este proyecto, describiendo la RIA a generar, el proceso de desarrollo seguido, las extensiones realizadas a los metamodelos y perfiles de MoWebA definiendo un nuevo ASM, y las reglas de transformación encargadas de generar el código final.
- El capítulo 5 describe la experiencia de validación realizada, que busca evaluar la usabilidad del enfoque MDD propuesto. Para ello, se ha solicitado la elaboración de una RIA funcional a los alumnos de último año de la carrera de Ingeniería Informática de la Universidad Católica "Nuestra Señora de la Asunción". Se siguieron los pasos de diseño de la experiencia, preparación y recolección de datos, y análisis e interpretación de los resultados obtenidos.
- El capítulo 6 finaliza el documento describiendo la conclusión y los posibles trabajos futuros a realizar a partir de este proyecto.

2. Introducción a RIA, MDD y MoWebA

Para comprender la propuesta de este proyecto de fin de carrera es necesario esclarecer ciertos conceptos y terminologías que permitirán definir el contexto y alcance del trabajo.

A continuación se presentarán a las Aplicaciones Enriquecidas de Internet o RIA, se hablará de sus características y tecnologías para su desarrollo. Posteriormente se definirá el Desarrollo Dirigido por Modelos o MDD, presentando las ventajas de seguir un paradigma de este tipo, junto a un subconjunto estándar de MDD denominado Arquitectura Dirigida por Modelos o MDA. Finalmente, se presentará un enfoque MDD para el desarrollo de aplicaciones web, denominado MoWebA, el que posteriormente será extendido para contemplar el desarrollo de RIA.

2.1. Las Aplicaciones Enriquecidas de Internet: RIA

En los inicios de la World Wide Web (WWW) las aplicaciones de internet poseían una arquitectura cliente-servidor, caracterizándose por una navegación a través de links, recarga entera de la página ante cualquier interacción del usuario, elementos de interfaz poco interactivos, almacenamiento de datos y lógica de negocios implementados totalmente del lado del servidor y una comunicación síncrona entre las partes. Estos factores afectaban notablemente a la experiencia del usuario en cuanto a aspectos de presentación e interacción y generaban tiempos de respuesta prolongados.

Para contrarrestar estas limitantes de las aplicaciones web tradicionales aparecen las Aplicaciones Enriquecidas de Internet. Brambilla et al. [8] definen a las RIA (del inglés Rich Internet Application) como "aplicaciones web que explotan el poder de clientes web para incrementar la adaptabilidad y usabilidad de las interfaces de usuario, ofreciendo funcionalidades similares a las de aplicaciones de escritorio. Las RIA siguen el paradigma cliente-servidor, pero a diferencia de las aplicaciones web tradicionales, ellas son capaces de transferir el procesamiento de la interfaz de usuario, lógica de negocio y manejo de datos al cliente, utilizando posiblemente comunicaciones asíncronas".

El término RIA apareció por primera vez en un documento de Macromedia, en marzo del año 2002, en el que se introdujo la tecnología Macromedia Flash MX [9]. A partir de allí empezaron a realizarse investigaciones y desarrollarse tecnologías que permitieron su implantación.

Las RIA son una combinación de la presentación e interactividad de aplicaciones de escritorio con la arquitectura y forma de distribución de las aplicaciones web [10]. La capa de presentación con su interacción asociada es trasladada al cliente, la computación de la página y datos de la misma se distribuyen entre servidor y cliente, y la comunicación se realiza mayormente en forma asíncrona. Este esquema permite obtener elementos de interfaz más interactivos, evitar la recarga innecesaria de la página, soportar contenidos multimedia, evitar transferencia de datos redundantes y minimizar tiempos de respuesta mejorando notablemente la experiencia del usuario.

Como ejemplos de RIA en la actualidad tenemos aplicaciones como Facebook[1], Flickr[2], Google Docs[3], Google Maps[4], SlideRocket[5] y Youtube[6].

[1] Facebook. https://www.facebook.com/
[2] Flickr. https://www.flickr.com/
[3] Google Docs. https://docs.google.com/

2.1.1. Características de las RIA

Las RIA poseen cuatro características principales [10] [11]:

Distribución de datos: además del almacenamiento de datos del lado del servidor, propio de las aplicaciones web tradicionales, las RIA soportan almacenamiento del lado del cliente. Los datos en el cliente pueden ser temporales o persistentes, permitiendo su manipulación y preparación antes de su envío al servidor. Esta característica de distribución de datos permite realizar una validación de datos local y el uso offline de la aplicación.

Distribución de lógica de negocios: las RIA permiten realizar operaciones complejas del lado del cliente. Estas operaciones incluyen ordenado y filtrado de datos antes de enviarlos al servidor, validación en vivo (acompañado de almacenamiento de datos local), uso offline, navegación y reordenamiento de la página.

Comunicación asíncrona: es posible utilizar comunicaciones síncronas y asíncronas. Tanto el cliente como el servidor son capaces de iniciar la comunicación. La comunicación asíncrona permite la recarga de partes de la página y operaciones del tipo *push* y *pull*.

Mejora de interfaz de usuario: la presentación y comportamiento de la interfaz de usuario de las RIA se ven altamente enriquecidos. Existen widgets y contenedores modernos y personalizables que mejoran la interfaz de usuario. Una página está compuesta de subpáginas donde cada una maneja la interacción del usuario de forma independiente, actualizando y desplegando elementos individuales de la interfaz de usuario. Estas mejoras permiten la implementación de animaciones, multimedia y operaciones del tipo *drag&drop*.

2.1.2. Tecnologías para el Desarrollo de RIA

Podemos desarrollar las RIA a partir de diversas tecnologías. Estas tecnologías se diferencian unas de otras en la forma en que la RIA es implementada y desplegada al usuario. La clasificación de estas tecnologías puede también ser utilizada para clasificar a las RIA en sí.

A continuación se presentan las diferentes tecnologías y su clasificación [10] [3] [12]:

Basadas en Scripting: las páginas web incluyen scripts (entendibles por navegadores modernos) que hacen referencia a elementos de presentación. HTML y CSS (Cascading Style Sheets) son utilizados para la definición y personalización de elementos de la interfaz, y Javascript se utiliza para la creación de los scripts que definen la lógica del lado del cliente, posibilitando solicitudes asíncronas al servidor. La técnica mayormente utilizada es AJAX (Asynchronous Javascript And XML). Una gran variedad de *frameworks* han sido desarrollados para superponer la complejidad de este tipo de aplicaciones y superar incompatibilidades con navegadores, entre ellos tenemos Backbase[7], Dojo[8], GWT[9], jQuery[10], Prototype[11] y la propuesta del W3C (World Wide Web Consortium), HTML5.

---

[4] Google Maps. https://www.google.com/maps/
[5] SlideRocket. http://www.sliderocket.com/
[6] Youtube. https://www.youtube.com/
[7] Backbase. https://www.backbase.com/
[8] Dojo. https://dojotoolkit.org/
[9] GWT. http://www.gwtproject.org/
[10] jQuery. https://jquery.com/
[11] Prototype. http://prototypejs.org/

Basadas en Plugin: se ejecutan en *plugins* de navegadores que pueden venir instalados previamente o requieran la instalación por parte del usuario. Este enfoque provee un mejor desempeño al de la utilización de Javascript y dota a la aplicación con procesamiento de eventos y renderizado avanzado. Las tecnologías para la creación de estas aplicaciones son Adobe Flash[12], Adobe Flex[13], OpenLaszlo[14], entre otras.

Ambientes de Ejecución Específicos: ejecutados en una aplicación independiente lanzada desde el navegador. Proponen lo mejor en cuanto a uso offline de la aplicación, acceso al sistema operativo y almacenamiento en el cliente. Las tecnologías más utilizadas son Adobe AIR[15], JavaFX[16] y Java Web Start[17].

Basadas en Navegador: especificadas en un lenguaje declarativo construido a partir de XML (eXtensible Markup Language) y reconocido por un navegador. A partir de este lenguaje los desarrolladores definen los distintos elementos e interacciones correspondientes. No requiere la extensión del navegador, sin embargo, es posible que las aplicaciones no puedan ser desplegadas en otros navegadores. Una tecnología utilizada para estas aplicaciones es Mozilla XUL[18].

Mediante estas tecnologías podemos implementar las características RIA presentadas en la sección anterior. Como podemos observar en la *Figura 2.1*, las tecnologías basadas en *plugin* y en ambientes de ejecución específicos cubren todas las características RIA, mientras que las basadas en scripting tienen limitaciones en cuanto a contenidos multimedia y persistencia de datos en el cliente. Las tecnologías basadas en navegador no son consideradas debido a que son raramente utilizadas en la práctica.

Características RIA / Tecnologías RIA	Presentación Enriquecida	Almacenamiento de Datos en el Cliente	Lógica de Negocios en el Cliente (y Distribuida)	Comunicación Cliente-Servidor
Basadas en Scripting	Limitada: Sin multimedia	Limitado: Sin persistencia	Si	Si
Basadas en Plugin	Si	Si (mediante *plugins* adicionales)	Si	Si
Ambientes de Ejecución Específicos	Si	Si	Si	Si

Figura 2.1: Características cubiertas por los tipos de tecnologías.

Si bien las tecnologías basadas en scripting no cubren todas las características, éstas son las más utilizadas para desarrollar RIA. Esto se da debido a que la técnica utilizada (AJAX) es bien aceptada en la comunidad de desarrolladores, ya que posee un standard bien establecido y es de código abierto. Además, no requiere instalación de software por parte de los usuarios, dejando el procesamiento de la aplicación en manos del navegador. Para sobreponerse a las limitaciones, estas tecnologías pueden combinarse con las tecnologías basadas en *plugin* y así obtener mejores representaciones de video y almacenamiento local.

[12] Adobe Flash. http://www.adobe.com/la/products/flash.html
[13] Adobe Flex. http://www.adobe.com/la/products/flex.html
[14] OpenLaszlo. http://openlaszlo.org/
[15] Adobe Air. http://www.adobe.com/la/products/air.html
[16] JavaFX. http://docs.oracle.com/javafx/
[17] Java Web Start. http://www.oracle.com/technetwork/java/javase/javawebstart/index.html
[18] Mozilla XUL. https://developer.mozilla.org/es/docs/XUL

2.2. El Desarrollo Dirigido por Modelos: MDD

Brambilla et al. [4] definen al Desarrollo Dirigido por Modelos o MDD (del inglés Model Driven Development) como "un paradigma de desarrollo que utiliza modelos como artefactos primarios en el proceso de desarrollo". Además destacan: "por lo general, en MDD la implementación es generada de forma (semi)automática a partir de los modelos".

MDD posee tres pilares fundamentales: los modelos, las transformaciones y las herramientas. Los modelos constituyen el núcleo dentro del proceso de desarrollo, éstos dejan de ser solo documentación para ahora formar parte del proceso de implementación del sistema, están escritos en lenguajes bien definidos y especificados en diferentes niveles de abstracción. Las transformaciones son aplicadas sobre los modelos de modo a obtener modelos más concretos o código. Las herramientas facilitan la creación de modelos y la especificación y posterior ejecución de transformaciones.

La idea es partir de modelos bien abstractos, realizar una serie de transformaciones modelo a modelo para obtener modelos más concretos, y finalmente, realizar una transformación modelo a texto para la obtención de la implementación del sistema. La iniciativa MDD busca obtener un mayor nivel de abstracción en la solución de problemas, aumentar la confianza en la automatización de tareas y el uso de estándares para facilitar las actividades de desarrollo [13].

Si bien las RIA pueden desarrollarse siguiendo un enfoque basado en código o basado en *frameworks* (más utilizado comúnmente), realizar un trabajo en donde se adopte el enfoque MDD resulta interesante, considerando de que según sus propulsores el mismo aporta las siguientes ventajas [13] [14]:

- Incremento de la productividad de desarrolladores.
- Mejora de la calidad del software generado.
- Mayor reusabilidad de modelos y transformaciones.
- Contempla más fases del ciclo de vida del software.
- Separación de la especificación e implementación de un sistema.
- Manejo de complejidad a través de abstracciones de alto nivel.
- Adaptación a cambios de requerimientos y cambios tecnológicos.
- Mejora de la comunicación con los usuarios y entre desarrolladores
- Mejor interoperabilidad entre sistemas.
- Mayor portabilidad entre plataformas.

2.2.1. Arquitectura Dirigida por Modelos

La Arquitectura Dirigida por Modelos o MDA[19] (del inglés Model Driven Architecture) es la propuesta de desarrollo dirigido por modelos de la OMG (Object Management Group). Consiste en una arquitectura general que provee lenguajes de modelado y transformaciones estándares para un proceso MDD.

MDA especifica tres niveles de abstracción [4]:

CIM (Computation-Independent Model): modelo de la aplicación sin tener en cuenta aspectos computacionales.

PIM (Platform-Independet Model): definición de la estructura y comportamiento de la

[19] MDA. http://www.omg.org/mda/

aplicación sin tener en cuenta la plataforma destino.

PSM (Platform-Specific Model): definición de la estructura y comportamiento de la aplicación para una plataforma determinada.

Para un CIM dado es posible tener más de un PIM asociado o "mapeado". A su vez, cada PIM puede mapearse a más de un PSM.

Entre los estándares asociados a MDA se encuentran MOF[20] (MetaObject Facilities), UML[21] (Unified Modeling Language), CWM[22] (Common Warehouse Metamodel), XMI[23] (XML Metadata Interchange) y CORBA[24] (Common Object Request Broker Architecture).

MDA establece que el proceso de desarrollo de software puede empezar con la especificación del PIM, mediante un lenguaje de modelado basado en MOF. A través de herramientas de desarrollo MDA, se transforma el PIM en PSM y luego se transforma el PSM en la implementación deseada. Esta implementación puede ser C#.NET, CORBA, EJB, XML/SOAP, entre otras [5]. También es posible iniciar el proceso con la especificación del CIM, derivando posteriormente el PIM asociado y a partir de éste seguir el procedimiento anterior [4].

MDA busca derivar valor a partir de los modelos, establecer una mejor interoperabilidad y portabilidad entre sistemas y separar el diseño de una aplicación en diferentes fases y modelos que permitan estandarizar y mejorar el proceso de desarrollo de software.

2.3. MoWebA

MoWebA (Model Oriented Web Approach) [6] es un enfoque MDD para el desarrollo de aplicaciones web. El enfoque adopta el estándar MDA, define diferentes aspectos metodológicos, se basa en una arquitectura por capas, propone metamodelos, y se complementa con herramientas de modelado, transformación y generación de código.

A diferencia de la mayoría de los enfoques que proveen un modelado basado en una navegación orientada a datos partiendo desde un modelo conceptual o de datos, MoWebA concibe una navegación orientada a funciones, permitiendo que la navegación se encuentre directamente relacionada con las funcionalidades definidas a través de los casos de uso.

MoWebA se basa fuertemente en una separación de conceptos, buscando proponer un modelado orientado a la navegación, automatización de tareas, adopción de estándares y el manejo de la evolución de ambientes web considerando las nuevas tendencias tecnológicas. Mediante esto se busca lograr una mayor interoperabilidad y extensibilidad de sistemas web.

El enfoque define dos tipos de procesos complementarios: el proceso de modelado y el proceso de transformación, que serán explicados en las secciones continuas. Posteriormente, se describirán los metamodelos y perfiles de los diagramas de MoWebA a extender en este trabajo.

2.3.1. Proceso de Modelado

El proceso de modelado consiste en la especificación de los diferentes diagramas que representan a la aplicación web. Se consideran las abstracciones CIM, PIM y PSM descriptas

[20] MOF. http://www.omg.org/mof/
[21] UML. http://www.uml.org/
[22] CWM. http://www.omg.org/cwm/
[23] XMI. http://www.omg.org/spec/XMI/
[24] CORBA. http://www.corba.org/

por MDA. Además se presenta una abstracción más llamada ASM (Architecture Specific Model).

El CIM es utilizado para la especificación de los requerimientos funcionales. El PIM define cinco tipos de modelos generales: modelo de entidad, modelo navegacional, modelo de comportamiento, modelo de presentación y modelo de usuario. A su vez, cada uno de estos modelos define uno o más diagramas para la especificación de sus funcionalidades. El modelo de entidad define el diagrama de entidades, utilizado para definir la estructura y relaciones estáticas entre clases identificadas en el dominio del problema. El modelo navegacional define el diagrama de árbol navegacional, utilizado para definir las funcionalidades básicas del sistema en una forma jerárquica, y el diagrama de nodos, utilizado para definir la navegación de cada nodo del árbol navegacional. El modelo de comportamiento define el diagrama lógico, utilizado para definir la composición estructural de procesos de negocios y procedimientos transaccionales, y el diagrama de servicios, utilizado para proveer una definición detallada de cada servicio o acción del diagrama lógico. El modelo de presentación define el diagrama de contenido, utilizado para definir los elementos a desplegar en cada página de presentación, y el diagrama de estructura, utilizado para definir la posición de encabezados, menús, pies de página y otros elementos relacionados a la estructura de la página. Finalmente, el modelo de usuario define el diagrama de roles y zonas, utilizado para definir los usuarios potenciales de la aplicación, y el diagrama de adaptación, utilizado para personalizar los modelos a través de fuentes y reglas que representan diferentes tipos de adaptaciones.

El ASM introducido por MoWebA sirve para enriquecer a los modelos con información acerca de una arquitectura específica, como por ejemplo REST, RIA, SOA (Service Oriented Applications). El PSM se utiliza para añadir a los modelos consideraciones específicas para cierta plataforma, como por ejemplo un lenguaje de programación específico o un *framework*.

La especificación de estas abstracciones se realiza de manera progresiva y sistemática en siete etapas. Las etapas de la 1 a la 6 contemplan el CIM y el PIM, mientras que la etapa 7 se refiere al ASM y PSM. Cada etapa define una serie de diagramas definidos según las dependencias entre modelos, nivel de granularidad de la tarea a realizar y el tipo de modelado. La *Figura 2.2* presenta gráficamente el proceso de modelado propuesto.

2.3.2. Proceso de Transformación

El proceso de transformación implica las actividades y el uso de herramientas necesarias para realizar la transformación de la especificación inicial en la implementación final a lo largo del proceso de desarrollo.

Existen dos fases esenciales: la transformación del PIM al ASM/PSM y la transformación del ASM/PSM al ISM (Implementation Specific Model). El ISM consiste en la implementación generada automáticamente sin intervención manual del desarrollador. La fase de PIM-ASM/PSM consiste en una transformación de modelo a modelo (M2M) realizada de forma semiautomática, mientras que la fase de ASM/PSM-ISM es una transformación de modelo a texto (M2T) realizada de forma automática. En la *Figura 2.3* podemos observar gráficamente el proceso de transformación descripto.

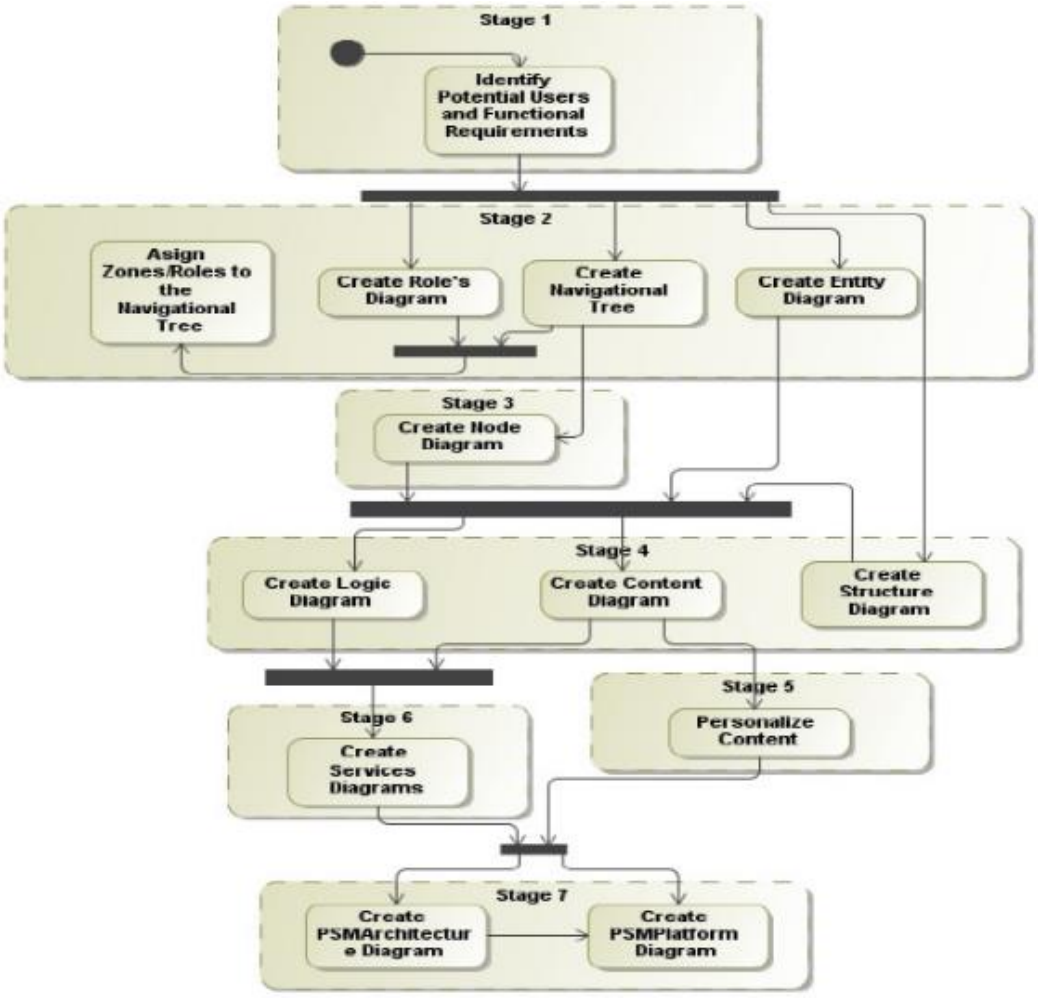


Figura 2.2: Proceso de modelado en MoWebA

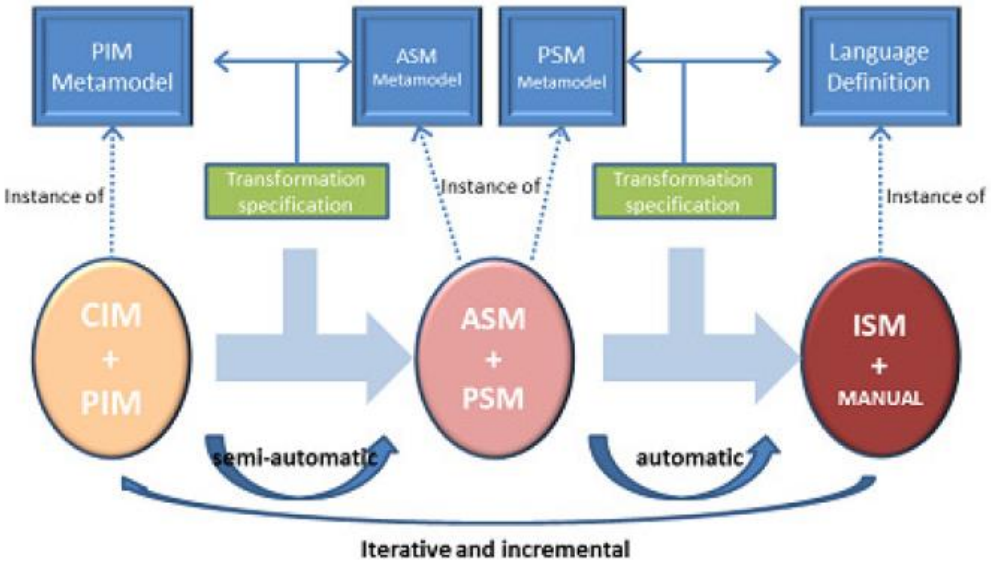


Figura 2.3: Proceso de transformación en MoWebA.

2.3.3. Metamodelos y Perfiles de MoWebA

MoWebA propone metamodelos y perfiles para la definición de los diagramas de los modelos del PIM mencionados anteriormente. A continuación se describirán los metamodelos y perfiles de los diagramas lógico y de contenido debido a que éstos serán utilizados posteriormente por la propuesta.

2.3.3.1. Metamodelos y Perfiles del Diagrama Lógico

El diagrama lógico define los procesos de negocios y procedimientos transaccionales que afectan al sistema. En la *Figura 2.4* se observa al metamodelo y perfil que especifican a este diagrama, el cual se halla compuesto por objetos de valor (clases con el estereotipo <<*valueObject*>>) que contienen datos encapsulados, y procesos (clases con el estereotipo <<*tprocess*>>) que representan procesos de negocios. Los objetos de valor proveen visibilidad del dominio a la capa de presentación y consisten en subconjuntos de atributos de clases del diagrama de entidades, especificados a través de relaciones de dependencia con estas últimas. Los procesos están compuestos por servicios, que representan transacciones complejas y se asocian mediante relaciones de dependencia con clases del diagrama de entidades para indicar el acceso a los datos por parte de los servicios.

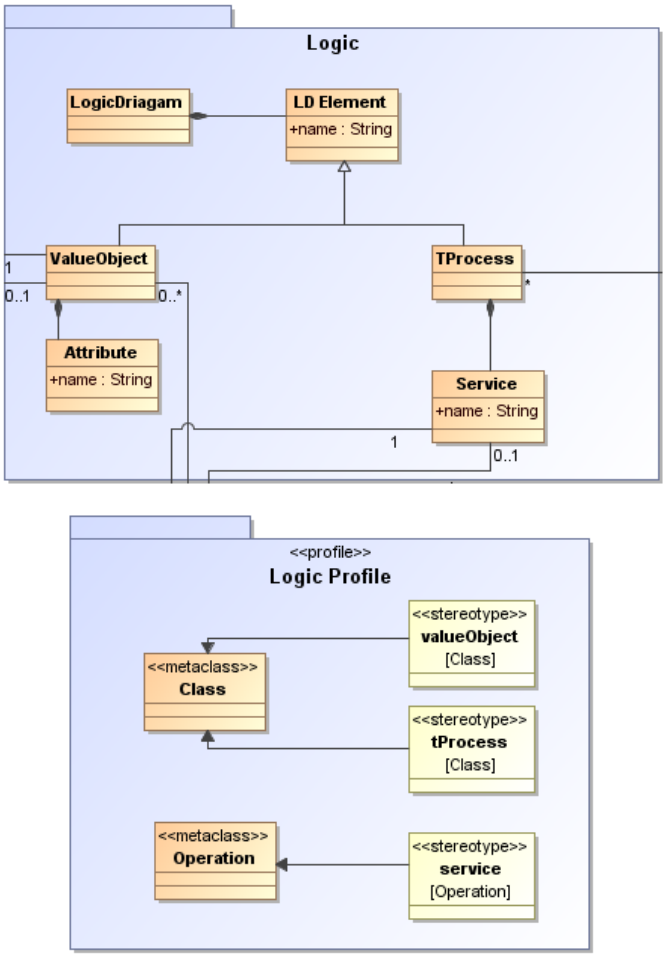


Figura 2.4: Metamodelo y perfil lógico de MoWebA.

2.3.3.2. Metamodelos y Perfiles del Diagrama de Contenido

El diagrama de contenido define los elementos a presentarse en cada página de la aplicación. El metamodelo y perfil de la *Figura 2.5* describen al diagrama como un conjunto de páginas de presentación (clases con estereotipo <<*presentationPage*>>), que contienen contenedores denominados elementos compuestos de interfaz de usuario (clases con el estereotipo <<*compositeUIElement*>>) que pueden especializarse en dos tipos de contenedores más, la tabla (clase con estereotipo <<*table*>>) y el formulario (clase con estereotipo <<*form*>>).

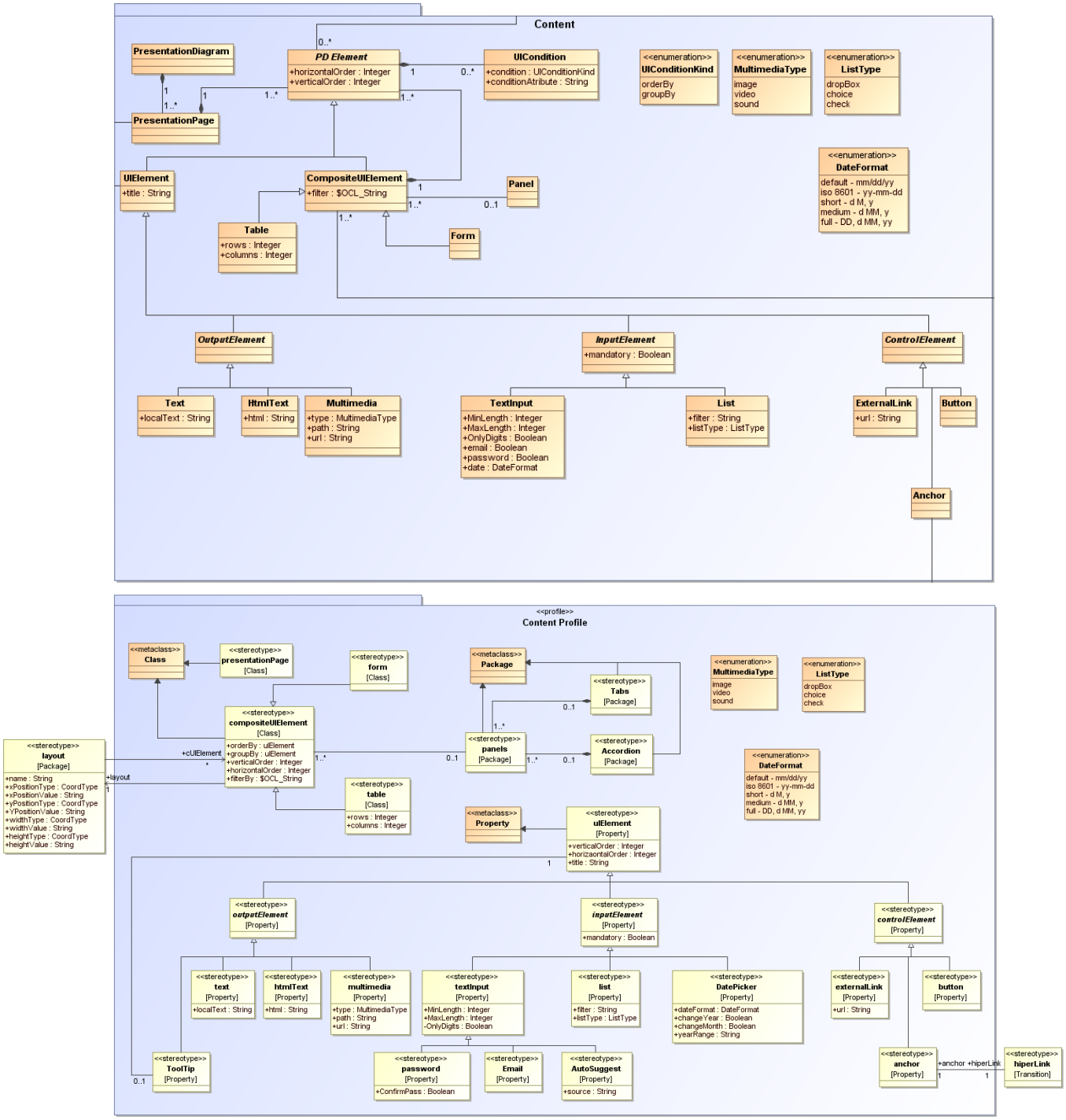


Figura 2.5: Metamodelo y perfil de contenido de MoWebA.

Estos contenedores contienen elementos de interfaz de usuario de tres tipos: i) los elementos de salida, que pueden ser los elementos texto (propiedad con estereotipo <<*text*>>), texto html (propiedad con estereotipo <<*htmlText*>>) y multimedia (propiedad con estereotipo <<*multimedia*>>); ii) los elementos de entrada, que pueden ser la entrada de texto (propiedad con estereotipo <<*textInput*>>), y lista (propiedad con estereotipo <<*list*>>); y iii) los elementos de control, que pueden ser el link externo (propiedad con estereotipo <<*externalLink*>>), enlace o ancla (propiedad con estereotipo <<*anchor*>) y el botón (propiedad con estereotipo <<*button*>>). Cada uno de estos elementos posee propiedades que modelan aspectos como restricciones, limitaciones, valores posibles, entre otros.

2.4. Resumen del Capítulo

En este capítulo hemos presentado a las RIA, describiendo sus características distintivas de distribución de datos, distribución de lógica de negocios, comunicación asíncrona entre cliente y servidor, y mejora de interfaz de usuario. También presentamos las tecnologías para su desarrollo, entre las cuales distinguimos que las tecnologías basadas en scripting constituyen la opción común para las prácticas de desarrollo de RIA.

Hemos expuesto los fundamentos de MDD, exhibiendo las ventajas de llevar a cabo un proceso de desarrollo de RIA siguiendo un enfoque de este tipo. Dentro de MDD, vimos que MDA surge como la propuesta de la OMG para seguir un proceso de desarrollo estándar.

Presentamos el enfoque MoWebA, una propuesta MDD que sigue el estándar MDA para el desarrollo de aplicaciones web. Hemos expuesto el proceso de modelado y de transformación a código del enfoque. Luego realizamos una descripción de los metamodelos y pefiles de los diagramas lógico y de contenido de MoWebA que posteriormente serán utilizados por la propuesta.

3. Análisis de Enfoques MDD para RIA

Para llevar a cabo nuestra propuesta de solución y con el fin de proveer una solución auténtica y novedosa, es necesario recopilar y analizar los estudios previos comprendidos en la comunidad científica acerca de enfoques MDD para el desarrollo de RIA que contemplen las características de distribución de datos, distribución de lógica de negocios y comunicación asíncrona entre cliente y servidor.

Para ello, hemos realizado un mapeo sistemático de la literatura, siguiendo las indicaciones establecidas por Kitchenham et al. [15] y Genero et al. [16]. Hemos optado por un mapeo debido a que éste permite resumir la evidencia existente acerca de un área de interés, identificar brechas en la investigación actual y proveer un marco de trabajo que guíe la investigación. Entonces, mediante el mapeo seguimos un procedimiento formal para la recopilación de información y para el posterior análisis de los estudios encontrados.

A continuación se presentará el mapeo sistemático de la literatura propuesto, desglosado en sus tres partes principales: la planificación, la ejecución, y los resultados.

3.1. Planificación del Mapeo Sistemático de la Literatura

Como punto de partida, se ha elaborado la planificación del mapeo sistemático de la literatura. Para ello, se ha definido un protocolo utilizado como guía para la ejecución del mapeo. El protocolo incluye las secciones de identificación de la necesidad, preguntas de investigación, estrategia de búsqueda, criterios de selección, proceso de selección de estudios y esquema de clasificación. Este protocolo se presenta a continuación.

3.1.1. Identificación de la Necesidad

Este proyecto busca atacar el problema consistente en la necesidad del usuario de una mejor experiencia en la web. Una mejor experiencia se traduce en mejor adaptabilidad, capacidades interactivas mejoradas, menor tiempo de respuesta e interfaz de usuario enriquecida. Esto es logrado mediante tecnologías del lado del cliente y comunicaciones asíncronas.

Las RIA ofrecen este tipo de soluciones, por lo que son elegidas como objeto de desarrollo para este proyecto. Mediante enfoques MDD podemos abstraer y agilizar el desarrollo de estas aplicaciones debido a las diferentes ventajas que otorgan y que fueron descriptas en el capítulo anterior.

Como punto de partida, es necesario realizar una búsqueda de propuestas de solución RIA basadas en MDD en la comunidad científica. Se han identificados trabajos como [11] y [3] que presentan un resumen de los distintos enfoques MDD que proveen soluciones para la generación de RIA. Si bien estos trabajos fueron rigurosos, no siguieron un procedimiento estrictamente formal como los de una revisión sistemática de la literatura o SLR (del inglés Systematic Literature Review) o un mapeo sistemático de la literatura o SMS (del inglés Systematic Mapping Study). Casteleyn et al. [17] siguen un procedimiento más formal y presentan un SMS, que resume toda la investigación realizada por la comunidad científica con respecto al desarrollo de RIA, entre los años 2002 y 2011. Este trabajo sigue un enfoque más general, considerando todas las propuestas de desarrollo existentes de RIA, sin centrarse en las propuestas basadas en MDD.

Como consecuencia a lo anterior, hemos optado por desarrollar un SMS, que resuma las distintas propuestas de solución basadas en MDD para el desarrollo de RIA, centrándonos en

aquellas RIA que dan soporte a las características de distribución de datos, distribución de lógica de negocios y comunicación asíncrona entre cliente y servidor. Además, con el fin de encontrar trabajos posteriores a los identificados por Casteleyn et al. se tendrá en cuenta un periodo más extenso, considerando publicaciones hasta el año 2015.

3.1.2. Preguntas de Investigación

El objetivo de este estudio consiste en recopilar y analizar todos los trabajos que proveen un enfoque MDD para el desarrollo de RIA, que tienen en cuenta las características de distribución de datos, distribución de lógica de negocios y comunicación asíncrona entre cliente y servidor. Se busca identificar cuáles de estas propuestas adoptan el estándar MDA y qué lenguajes de modelado, herramientas y tecnologías de implementación utilizan para su propuesta de solución.

El objetivo anterior puede reformularse en las siguientes preguntas de investigación:

- **PI1:** ¿Qué enfoques MDD existen para el desarrollo de RIA con características de distribución de datos, distribución de lógica de negocios y comunicación asíncrona entre cliente y servidor?
- **PI2:** ¿Los enfoques identificados siguen el estándar propuesto por MDA?
- **PI3:** ¿Qué lenguajes de modelado, herramientas, tecnologías utilizan los enfoques identificados?

3.1.3. Estrategia de Búsqueda

Para la búsqueda de trabajos, primeramente se define la cadena de búsqueda presentada en la sección 3.1.3.1. Posteriormente, se introduce la cadena de búsqueda en los motores de búsqueda de las fuentes de información indicadas en la sección 3.1.3.2. Finalmente, se analizan los 150 primeros resultados devueltos por cada fuente de información ordenados por relevancia. Se recopilan artículos de conferencias, revistas y capítulos de libros, en idioma español o inglés, publicados hasta agosto de 2015.

3.1.3.1. Cadena de Búsqueda

Se ha establecido una cadena de búsqueda simple y genérica que permita obtener todos los enfoques MDD para el desarrollo de RIA. Se han utilizado los términos principales *Model Driven* y *Rich Internet Application* con sus acrónimos *MDD* y *RIA* correspondientes. La cadena de búsqueda resultante es la siguiente:

(Model Driven OR MDD) AND (Rich Internet Applications OR RIA)

Posteriormente, mediante los criterios de inclusión y exclusión se van filtrando aquellos enfoques que cumplen con las características RIA en cuestión.

3.1.3.2. Fuentes de Información

Se ha consultado en las siguientes fuentes de información, seleccionadas debido a su gran relevancia en el área de ingeniería de software, éstas han sido consultadas en el orden que aparecen:

- Librerías digitales: IEEExplore[25], ACM Digital Library[26], ScienceDirect[27] y SpringerLink[28].

[25] IEEExplore. http://ieeexplore.ieee.org/
[26] ACM Digital Library. http://dl.acm.org/
[27] ScienceDirect. http://www.sciencedirect.com/

- Servicios de indexación: Google Scholar[29].

3.1.4. Criterios de Selección

Una vez obtenidos los resultados de la consulta en las fuentes de información, se deben filtrar los estudios de acuerdo a los criterios de inclusión y exclusión.

3.1.4.1. Criterios de Inclusión

Se incluyen los estudios que contemplan los siguientes criterios:

1. Enfoques o extensiones de enfoques MDD de RIA.
2. Aplicaciones o usos de enfoques MDD de RIA.
3. Estados del arte, mapeos sistemáticos de literatura, estudios comparativos y de evaluación de distintos enfoques MDD de RIA.
4. Herramientas o extensiones de herramientas utilizadas por enfoques MDD de RIA.

3.1.4.2. Criterios de Exclusión

Se excluyen los estudios que cumplen con los siguientes criterios:

1. Duplicados.
2. No contempla las características de distribución de datos, distribución de lógica de negocios y comunicación asíncrona entre cliente y servidor.

3.1.5. Proceso de Selección de Estudios

Los criterios de inclusión y exclusión deben aplicarse sobre los resultados obtenidos de las fuentes de información en las siguientes etapas:

1. Primeramente se aplican los criterios analizando el título y resumen de los trabajos obtenidos. El criterio de exclusión Nº 2 no debe aplicarse en esta etapa.
2. Luego de filtrar los resultados, se aplica el criterio de exclusión Nº 2 analizando la introducción y conclusión de los trabajos. Si al analizar estos elementos, los trabajos siguen cumpliendo con el criterio de exclusión, se realizará una búsqueda de palabras clave por el trabajo completo, y se analizará el contexto de estas palabras para finalmente rechazar los trabajos que siguen cumpliendo el criterio de exclusión. Las palabras clave a buscar aplicadas a estudios en idioma inglés son: *client side*, *client-side*, *distribution* (analizando si se habla de datos, procesos, lógica de negocios), *asynchronous* (analizando si se habla de comunicación). Para estudios en idioma español se buscan las traducciones al español de las palabras citadas.
3. Finalmente, se procederá a la lectura completa de los trabajos resultantes, de modo a verificar si en realidad cumplen con todos los criterios.

En caso de encontrar trabajos del tipo estado del arte, mapeo sistemático de literatura, estudios comparativos o de evaluación se procederá a la lectura completa de estos trabajos, de modo a identificar trabajos que puedan cumplir los criterios y que no hayan sido encontrados mediante el proceso de selección aplicado. En caso de identificar otros trabajos, éstos serán conseguidos y se aplicarán los criterios de inclusión y exclusión sobre ellos siguiendo el proceso de selección.

---

[28] SpringerLink. http://link.springer.com/
[29] Google Scholar. https://scholar.google.com/

3.2. Ejecución del Mapeo Sistemático de la Literatura

Una vez concluida la planificación, se procedió a la ejecución de los procedimientos establecidos. Siguiendo la estrategia de búsqueda, se aplicó la cadena de búsqueda en las fuentes de información y se obtuvieron 2300 artículos por analizar. Posteriormente, se aplicaron los criterios de selección según lo estipulado en el proceso de selección de estudios. Tras analizar el título y resumen de los trabajos se obtuvieron 48 artículos. Tras analizar la introducción, conclusión y buscar palabras clave a través de los documentos resultantes se obtuvieron 15 artículos. Finalmente, tras la lectura completa de los documentos, se obtuvo una cantidad final de 13 trabajos. La *Figura 3.1* presenta un esquema gráfico del proceso realizado. Los trabajos resultantes pueden observarse en el Anexo 1.

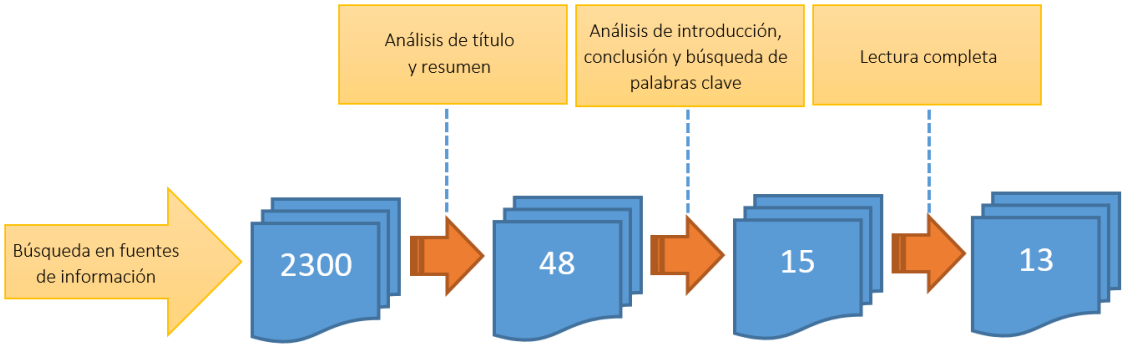


Figura 3.1: Cantidad de artículos obtenidos en cada etapa del proceso de selección de estudios.

3.3. Resultados del Mapeo Sistemático de la Literatura

Tras un análisis de los estudios seleccionados se han respondido las preguntas de investigación de la siguiente manera:

PI1: ¿Qué enfoques MDD existen para el desarrollo de RIA con características de distribución de datos, distribución de lógica de negocios y comunicación asíncrona entre cliente y servidor?

A partir de la lectura de los 13 estudios seleccionados, se han identificado 8 enfoques para desarrollar RIA que contemplan las características de distribución de datos, distribución de lógica de negocios, y comunicación asíncrona entre cliente y servidor.

La *Figura 3.2* presenta una tabla comparativa de estos enfoques. En las columnas 2, 3 y 4 podemos visualizar las características soportadas por cada enfoque. Podemos observar que cuatro de los ocho enfoques consideran todas las características en cuestión. Además se puede apreciar que la característica de comunicación asíncrona se encuentra presente en siete enfoques, siendo la más contemplada. A continuación describimos brevemente cada enfoque:

OOWS Extension [18] extiende el enfoque OOWS Method [19] para contemplar el desarrollo de RIA. El enfoque propone un metamodelo para especificar la interfaz de usuario en un nivel abstracto y en otro nivel concreto para definir widgets y reacciones ante eventos. Dado esto, el enfoque está centrado en la característica de mejora de la interfaz de usuario de las RIA. Sin embargo permite el modelado de comunicaciones asíncronas mediante reglas de eventos especificadas mediante XText[30], permitiendo que un servicio sea ejecutado asíncronamente ante la ocurrencia de un evento.

[30] Xtext. https://eclipse.org/Xtext/

Criterios / Enfoques	Distribución de Datos	Distribución de Lógica de Negocios	Comunicación Asíncrona	MDA	Lenguaje de Modelado	Herramienta de Desarrollo	Tecnología de Implementación
OOWS Extension [18]	✗	✗	✓	No	XText	OOWS model compiler	Adobe Flex
Enfoque MDD para aplicaciones web de alta calidad [20]	✗	✗	✓	Si	Extensión de UML	Eclipse	Javascript
RUX-Model [21]	✗	✗	✓	Si	DSL visual	RUX-Tool	Adobe Flex, AJAX, OpenLaszlo
WebML for RIA [22] [23] [3]	✓	✓	✗	No	Extensión de WebML	WebRatio	OpenLaszlo
WebML for RIA + RUX-Model [24] [25]	✓	✓	✓	No	Extensión de WebML, DSL visual de RUX-Model	WebRatio, RUX-Tool	OpenLaszlo, Adobe Flex, AJAX, XAML
WebML for RIA para aplicaciones web colaborativas [2] [11]	✓	✓	✓	No	Extensión de WebML	WebRatio	OpenLaszlo
Enfoque para diseño conceptual de RIA basado en procesos de negocios [8] [11]	✓	✓	✓	No	BPMN, WEBML, DSL visual de RUX-Model	WebRatio, RUX-Tool	No especificado
OOH4RIA Extension [26] [27] [3]	✓	✓	✓	Si	Czarnecki notation, UML	OOH4RIA Tool	RichFaces

Figura 3.2: Comparación de enfoques para desarrollo de RIA.

El enfoque MDD para aplicaciones web de alta calidad [20] presenta un perfil para extender UML con conceptos de aplicaciones web definiendo nuevos estereotipos y valores etiquetados, presentando además un *framework* de soporte y reglas de transformación para generación de código. Mediante nuevos estereotipos, se modelan solicitudes de ejecución de servicios web, distinguiéndolas en dos tipos, síncronas y asíncronas. Las asíncronas se pueden extender en los tipos *callback* (que incluye una operación a ejecutar) y *pull* (que permite almacenar respuestas del servidor).

RUX-Model [21] permite la especificación de la interfaz de usuario siguiendo un proceso de cuatro etapas: conexión con un modelo hipertextual, definición de la interfaz abstracta, definición de la interfaz concreta y especificación de la interfaz final que termina en generación de código. Al igual que OOWS Extension, RUX-Model está orientado a la especificación de la interfaz de usuario, pero permite modelar comunicación asíncrona a través de eventos y acciones asociadas mediante extensiones de UiML [28] contemplando métodos de solicitud del tipo *GET* y *POST*.

WebML for RIA [22] extiende los modelos de datos e hipertextual de WebML [29] para abarcar características RIA. Las entidades del modelo de datos son marcadas con niveles de persistencia que pueden ser *database* (permanente en el servidor) o *client* (temporal en el cliente) logrando así distribución de datos entre cliente y servidor. El modelo hipertextual expresa la estructura del *frontend* distinguiendo en páginas del tipo cliente o servidor. Además, en este modelo se define el contenido de las páginas, los enlaces y las operaciones lanzadas por el usuario. Estas operaciones son marcadas con la localización en la que se encuentran pudiendo ser en el servidor, cliente o una combinación de ambos obteniendo así distribución de lógica de negocios entre cliente y servidor.

WebML for RIA + RUX-Model [24] combina las ventajas de las dos últimas propuestas mencionadas, utilizando WebML for RIA para definir los modelos de datos e hipertextual para abarcar las características de distribución de datos y lógica de negocios entre cliente y servidor, y RUX-Model para definir la interfaz de usuario y soportar comunicación asíncrona entre las partes.

WebML for RIA para aplicaciones web colaborativas [2] extiende WebML for RIA con soporte para comunicación asíncrona definido sobre aplicaciones colaborativas que utilizan tecnologías de empuje. El modelo de datos es extendido con entidades que representan eventos y tipos eventos, mientras que el modelo hipertextual agrega dos nuevos tipos de operaciones, *send event* y *receive event* para la notificación de eventos y lanzamiento de reacciones. Las reacciones son modeladas a través de un *event view* especificado a través de una cadena de operaciones.

El enfoque para diseño conceptual de RIA basado en procesos de negocios [8] propone especificar una RIA a través de procesos de negocios descritos mediante BPMN, transformar esta especificación en los modelos de datos e hipertextual de WebML y aplicar RUX-Model para obtener la interfaz de usuario. Se define un modelo BPMN como instancia de un metamodelo de procesos propuesto. Mediante este modelo se logra la distribución de datos agrupando objetos de datos en *lanes* (objeto BPMN para categorizar actividades de un proceso) del tipo cliente o servidor y anotaciones sobre los objetos indicando su nivel de persistencia que puede ser persistente o volátil. La lógica de negocios es representada mediante actividades y flechas de flujo de control agrupadas en *lanes* del tipo cliente, servidor o cliente-servidor indicando la posible distribución. Flechas de flujo de control indican la comunicación entre

actividades y procesos y permiten anotaciones que especifican el tipo de transmisión, síncrona o asíncrona.

OOH4RIA Extension [26] dota a OOH4RIA [30] con dos nuevos modelos, el modelo de características y el modelo de componentes. El modelo de características está formalizado por un metamodelo MOF y especifica las diferentes características que se pueden considerar en el diseño de una RIA por medio de artefactos (entidades, componentes, atributos). Estos artefactos permiten representar almacenamiento de datos del lado del cliente del tipo persistente y volátil, lógica de negocios con localización en el cliente, servidor o cliente-servidor y estilos de comunicación que pueden ser RPC o asíncrono basado en mensajes. El modelo de componentes es una instancia de un perfil UML y representa la estructura visual de la aplicación y una topología de componentes.

PI2: ¿Los enfoques identificados siguen el estándar propuesto por MDA?

La columna 5 de la *Figura 3.2* especifica los enfoques que adoptan MDA. Se puede observar que solo tres de los ocho enfoques siguen con esta propuesta. La propuesta de José Luis Herrero y Pablo Carmona no especifica haber definido un PIM o PSM, pero presenta un modelo basado en UML y reglas de transformación. RUX-Model y OOH4RIA Extension definen un PIM y un PSM y reglas de transformación asociadas. Los modelos de esta última propuesta son también especificados mediante extensiones a UML.

PI3: ¿Qué lenguajes de modelado, herramientas, tecnologías utilizan los enfoques identificados?

Los lenguajes de modelado, herramientas de desarrollo y tecnologías de implementación utilizados por los enfoques identificados pueden visualizarse en las columnas 6, 7 y 8 respectivamente de la *Figura 3.2*. La mitad de los enfoques se basan en WebML, un lenguaje de modelado concebido originalmente para el desarrollo de aplicaciones web tradicionales, junto con su herramienta de soporte, WebRatio[31]. Además de WebML, la notación visual propuesta por RUX-Model es también requerida por otros enfoques y desarrollada haciendo uso de la herramienta RUX-Tool[32] [21]. Dada la utilización de estos lenguajes de modelado, el empleo de UML se ve limitado solo a dos enfoques de desarrollo. En cuanto a tecnologías de implementación, la mayormente utilizada es OpenLaszlo, seguida por Adobe Flex.

A partir de las respuestas a las preguntas de investigación planteadas, hemos observado los siguientes hechos:

- Cuatro enfoques cubren todas las características.
- Tres enfoques adoptan MDA.
- Un enfoque cubre todas las características y adopta MDA.
- Seis enfoques parten de un lenguaje de modelado existente.
- Cuatro enfoques emplean la herramienta WebRatio.
- Tres enfoques emplean la herramienta RUX-Tool.
- Cuatro enfoques generan una implementación en OpenLaszlo.
- Tres enfoques generan una implementación en Adobe Flex.

[31] WebRatio. http://www.webratio.com/
[32] RUX-Tool. http://www.homeria.com/

Lo anterior nos permite notar que los enfoques encontrados presentan debilidades como la falta de inclusión de todas las características RIA en cuestión y la falta de adopción de MDA (especialmente sus estándares UML y MOF). La no inclusión de alguna característica limita las funcionalidades de la RIA a generar, mientras que la no adopción de MDA hace que el proceso de desarrollo carezca de un modelo estándar a seguir, aumentando la complejidad del uso y desarrollo del enfoque.

También notamos que existen enfoques que proponen una metodología desde cero, mientras otros parten de una propuesta existente, combinándola con otra o agregando nuevas funcionalidades. Partir de un lenguaje de modelado existente resulta conveniente en términos de aprendizaje, sin embargo, si se parte de una combinación de varios lenguajes o se desarrolla un lenguaje desde sus inicios, la curva de aprendizaje podría aumentar notablemente.

Además se observa que la mayoría de las herramientas de desarrollo se encuentran muy atadas a un lenguaje de modelado, no pudiendo ser utilizadas de forma universal con otros lenguajes (a excepción de Eclipse).

En cuanto a tecnologías de implementación, podemos ver que la mayoría utiliza tecnologías basadas en *plugin* (OpenLaszlo y Adobe Flex) dejando de lado a las tecnologías basadas en scripting, que corresponden a las tecnologías mayormente utilizadas en la comunidad de desarrolladores.

Para superar estas debilidades, este proyecto de fin de carrera busca proveer una solución al desarrollo de RIA mediante un enfoque MDD que contemple todas las características en cuestión, adopte el estándar MDA, parta de un lenguaje de modelado existente, utilice herramientas universales y genere una implementación con tecnologías basadas en scripting.

3.4. Resumen del Capítulo

En este capítulo hemos realizado una búsqueda y análisis de los enfoques MDD para el desarrollo de RIA que tienen en cuenta las características de distribución de datos, distribución de lógica de negocios y comunicación asíncrona entre cliente y servidor. Para ellos hemos seguido los pasos de un SMS, realizando una planificación y posterior ejecución del proceso de búsqueda y selección de trabajos.

Tras la ejecución del proceso, encontramos 13 estudios de los cuales identificamos 8 enfoques finales. Entre estos enfoques, recolectamos datos acerca de las características soportadas, la adopción de MDA, los lenguajes de modelado utilizados, las herramientas de desarrollo empleadas y las tecnologías de implementación aprovechadas. Pudimos notar la falta de inclusión de todas las características RIA en cuestión, la falta de adopción de MDA, la propuesta de un lenguaje de modelado desde cero o como una combinación de varios lenguajes generando un aumento de la curva de aprendizaje, herramientas limitadas, y la falta de consideración de tecnologías basadas en scripting. A partir de lo señalado, vimos conveniente la propuesta de un enfoque MDD que supere las limitantes anteriores y provea una solución eficaz para el desarrollo de RIA.

4. Una Extensión RIA de MoWebA

Este proyecto de fin de carrera busca proveer un enfoque MDD que facilite el proceso de desarrollo de una RIA. Para ello se parte de MoWebA, un enfoque MDD concebido para el desarrollo de aplicaciones web, el cual es extendido mediante metamodelos y perfiles de modo a proporcionar funcionalidades RIA.

La RIA a generar tiene en cuenta las características de almacenamiento de datos en el cliente, lógica de negocios en el cliente y comunicación asíncrona entre cliente y servidor. Se han introducido elementos que representan funcionalidades RIA que abarcan estas características.

Se trabajó sobre los diagramas lógico y de contenido de MoWebA, modificando o agregando elementos correspondientes al PIM que faciliten el posterior modelado de elementos RIA en el ASM. Seguidamente, se elaboraron estos elementos RIA abarcando las características recientemente mencionadas. Por último, se desarrollaron reglas de transformación que convierten los modelos especificados en código correspondiente a la aplicación final.

En este capítulo se presentará primeramente la RIA a generar, detallando las funcionalidades a ser implementadas. Seguidamente se presentará el proceso a seguir para desarrollar la RIA. Luego se introducirán las extensiones realizadas al PIM y el ASM. Posteriormente se detallarán las reglas de transformación que generan la RIA. Finalmente se mostrará un ejemplo de modelado y generación de código haciendo uso de las extensiones propuestas.

4.1. RIA a Generar

El enfoque propuesto en este trabajo de fin de carrera busca generar una RIA que contemple puntualmente las características de almacenamiento de datos en el cliente, lógica de negocios en el cliente y comunicación asíncrona entre cliente y servidor.

Se tienen en cuenta estas características debido a que ellas son de gran importancia para el despliegue de la aplicación, ya que representan las operaciones ejecutadas por debajo para la correcta presentación de la interfaz de usuario.

Considerando el almacenamiento de datos en el cliente, se introducen elementos que representan variables almacenadas del lado del cliente. El *objeto de valor en el cliente* (*ClientValueObject*) consiste en una variable que obtiene sus datos de una base de datos en el servidor, mientras que el *objeto estático en el cliente* (*ClientStaticObject)* es una variable que obtiene datos especificados estáticamente en el modelo de la aplicación. Ambos elementos permiten especificar el nivel de persistencia de los datos, que pueden ser del tipo temporal o permanente.

Para la lógica de negocios en el cliente, se añaden elementos que permiten especificar elementos de interfaz de usuario con operaciones de lógica de negocios que serán ejecutadas en el cliente. La *tabla enriquecida* (*RichTable*) consiste en una tabla con operaciones de paginación, ordenamiento y búsqueda ejecutadas en el cliente (aunque también puede ser del lado del servidor); el *formulario enriquecido* (*RichForm*) es un formulario con opción de autocompletado de sus campos basados en valores previamente ingresados; y la *entrada de texto enriquecida* (*RichTextInput*) representa una entrada de texto que posee también la opción de autocompletado anterior, pero además permite especificar una lista de valores a sugerir.

En cuanto a la comunicación asíncrona entre cliente y servidor, se adhiere el elemento *llamada asíncrona* (*AsynchronousCall*) que consiste en una invocación asíncrona de un servicio desde

un elemento de presentación, permitiendo especificar el evento ocurrido sobre el elemento, el tipo de solicitud, tipo de respuesta y parámetros a enviar al servicio.

4.2. Proceso de Desarrollo de RIA

Con el fin de modelar y posteriormente generar una RIA se parte haciendo uso de la herramienta MagicDraw[33]. Mediante la importación en la herramienta de los perfiles de MoWebA y del perfil RIA propuesto es posible realizar el modelo de una RIA que aproveche las diversas funcionalidades proporcionadas por este trabajo. Este modelo es exportado a un archivo en formato XMI, el cual a su vez es importado en la herramienta Acceleo[34]. Esta última se vale de plantillas con reglas de transformación propuestas que permiten realizar transformaciones modelo a texto generando el código final correspondiente a la implementación de la RIA.

La implementación contiene código HTML5 para elementos de interfaz de usuario y código Javascript para el comportamiento de dichos elementos. Además, las funcionalidades de HTML5 LocalStorage[35] y SessionStorage[36] son aprovechadas para el almacenamiento de datos del lado del cliente. JQuery[37] con sus *plugins* Datatables[38] y jQuery UI[39] son utilizados para la lógica de negocios del lado del cliente, generando tablas enriquecidas y autocompletado de entradas de texto. JQuery también es utilizado para implementar llamadas asíncronas entre cliente y servidor. El producto final consiste en una RIA que puede ser fácilmente alojada y desplegada en un servidor web. El proceso propuesto puede observarse en la *Figura 4.1*.

Figura 4.1: Proceso de modelado y generación de RIA.

4.3. Extensiones al PIM

A fin de facilitar el modelado de funcionalidades RIA, se han implementado extensiones al metamodelo y perfiles originales de MoWebA. Estas extensiones corresponden a funcionalidades que no se incluían en la versión inicial de MoWebA y que permiten facilitar el posterior modelado de características RIA. Las extensiones han sido aplicadas a los diagramas lógico y de contenido.

Para facilitar la comprensión, las clases en los metamodelos y perfiles han sido coloreadas y distinguidas, usando el color salmón para aquellas que no han sufrido modificación en su forma original, el azul para las que han sido modificadas ya sea mediante la agregación, actualización o eliminación de alguna de sus propiedades y el verde para las nuevas clases adheridas.

[33] MagicDraw. http://www.nomagic.com/products/magicdraw.html
[34] Acceleo. https://eclipse.org/acceleo/
[35] LocalStorage. https://www.w3.org/TR/webstorage/#the-localstorage-attribute
[36] SessionStorage. https://www.w3.org/TR/webstorage/#the-sessionstorage-attribute
[37] JQuery. https://jquery.com/
[38] Datatables. https://datatables.net/
[39] JQuery UI. https://jqueryui.com/

4.3.1. Extensiones al Diagrama Lógico

El diagrama lógico de MoWebA permite la definición de objetos de valor (*ValueObjects*) que proveen visibilidad y acceso al dominio a la capa de presentación. Estos objetos de valor encapsulan datos que dependen de una o más entidades, conteniendo un conjunto de atributos de las entidades dependientes.

La extensión propuesta en la *Figura 4.2* presenta una nueva clase que representa a un objeto estático (*StaticObject*). Este objeto es también utilizado para encapsular datos, pero a diferencia de los objetos de valor no dependen de alguna entidad. Un objeto estático es una clase compuesta por un conjunto de valores, los cuales son definidos de forma estática como propiedades de la clase. Estos valores poseen un nombre (*name*) utilizado como el valor en sí, un título (*title*) utilizado como texto a desplegar y un campo para pre-seleccionar el valor cuando es desplegado en una lista (*checked*).

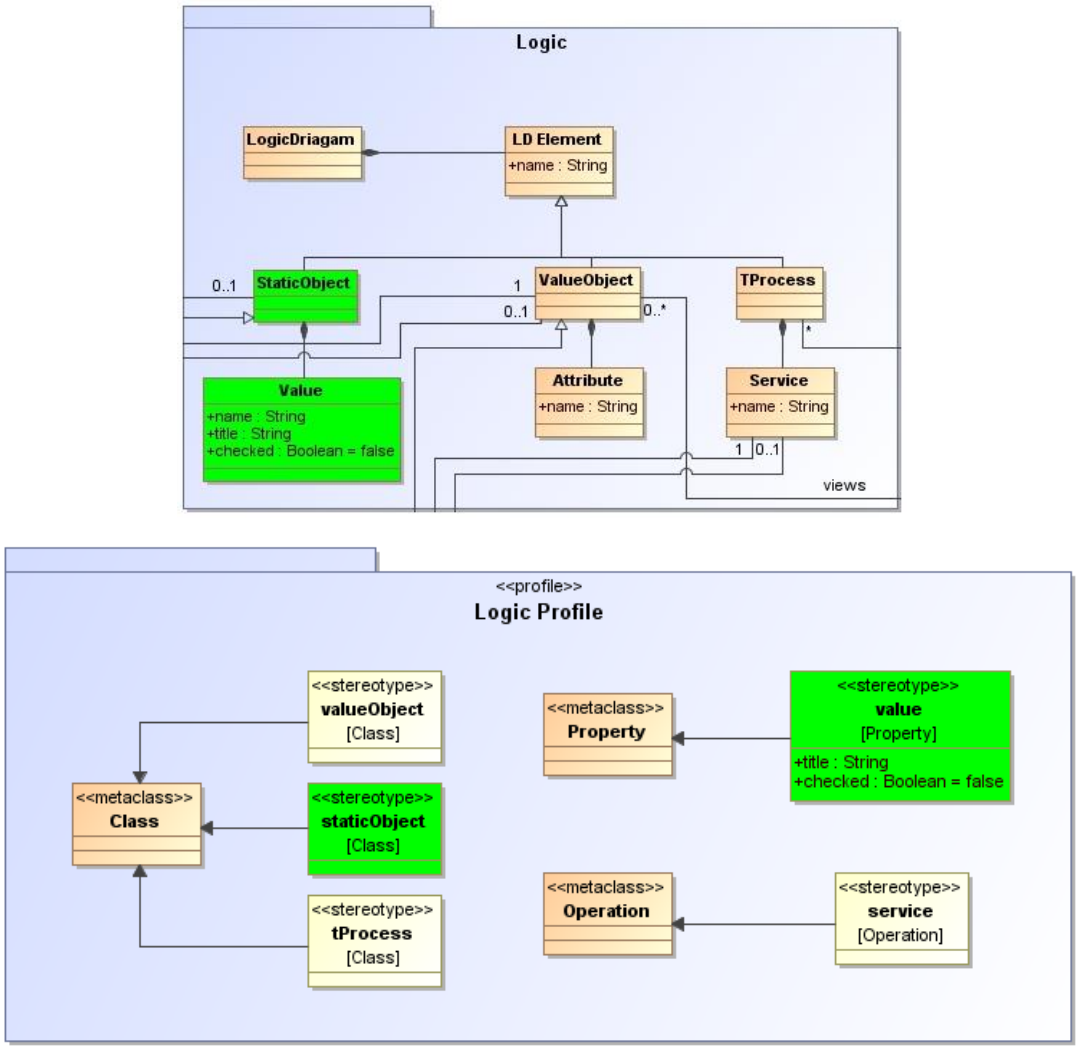


Figura 4.2: Metamodelo y perfil lógico extendido.

4.3.2. Extensiones al Diagrama de Contenido

El diagrama de contenido de MoWebA, utilizado para presentar a los usuarios los distintos elementos de cada página de la aplicación, también sufrió ligeras modificaciones. Éstas pueden observarse en la *Figura 4.3*.

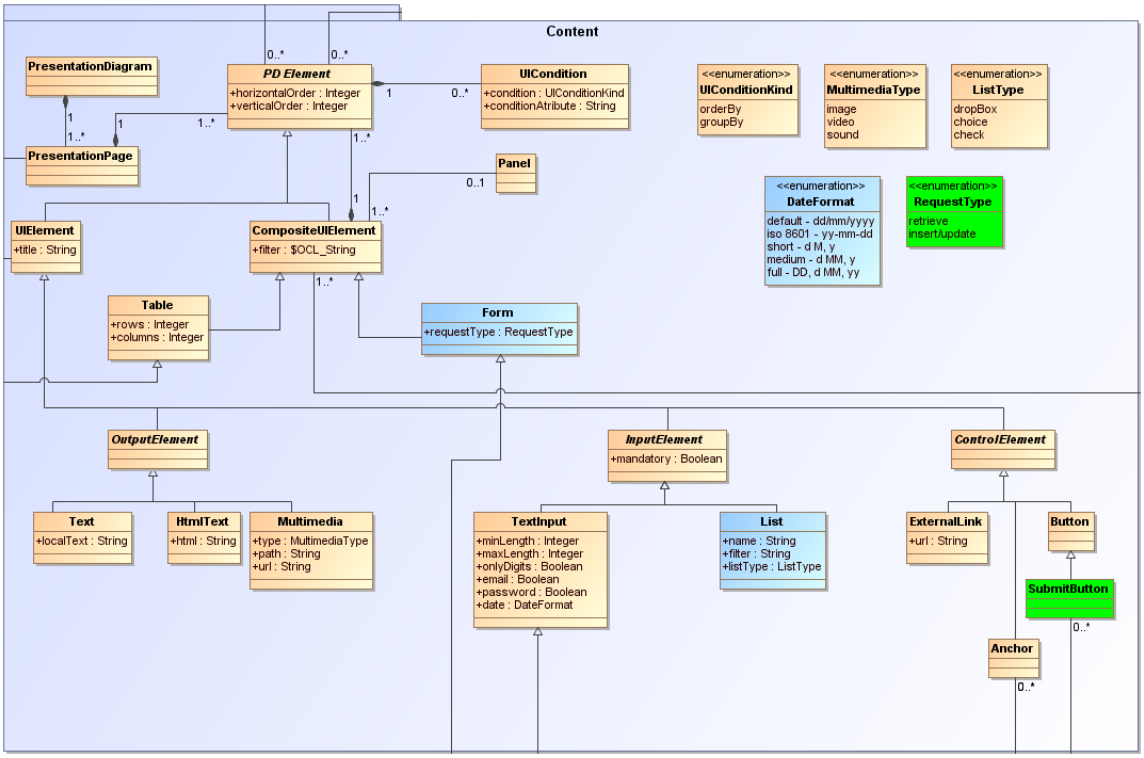
Primeramente, la clase formulario (*Form*) ahora comprende un atributo que especifica la forma de envío de datos o tipo de solicitud (*requestType*) realizada al servicio encargado del procesamiento del formulario. Este atributo puede utilizarse para recuperar datos remotos (*retrieve*) o para insertar o actualizar datos remotos (*insert*/*update*). A nivel de implementación estos valores serán mapeados con los métodos GET y POST de formularios HTML.

Al elemento lista (*List*) se le añadió el atributo nombre (*name*), utilizado para agrupar sus elementos. Los elementos de lista del mismo grupo deben tener el mismo nombre.

El campo *default* de la enumeración formato de fecha (*DateFormat*) ha sido cambiado al formato *dd/mm/yyyy* que se corresponde al formato por defecto de las entradas de fechas en HTML.

Se ha eliminado la relación entre el elemento enlace o ancla (*Anchor*) con el hiperenlace (*Hyperlink*) del diagrama de nodos. En su lugar, el *Anchor* ahora se relaciona directamente con el estado virtual (*VirtualState*) o el estado de servicio (*ServiceState*) del diagrama de nodos al cual redirige el enlace (este cambio no se encuentra coloreado debido a que afecta a las relaciones de la clase y no a sus atributos).

Por último, se agregó un botón de envío (*SubmitButton*) como especialización de la clase botón presente (*Button*), utilizado dentro de un formulario para enviar los datos de él a un servicio. Este botón se relaciona con un servicio del diagrama de nodos para indicar el servicio encargado de procesar los datos enviados por el formulario.



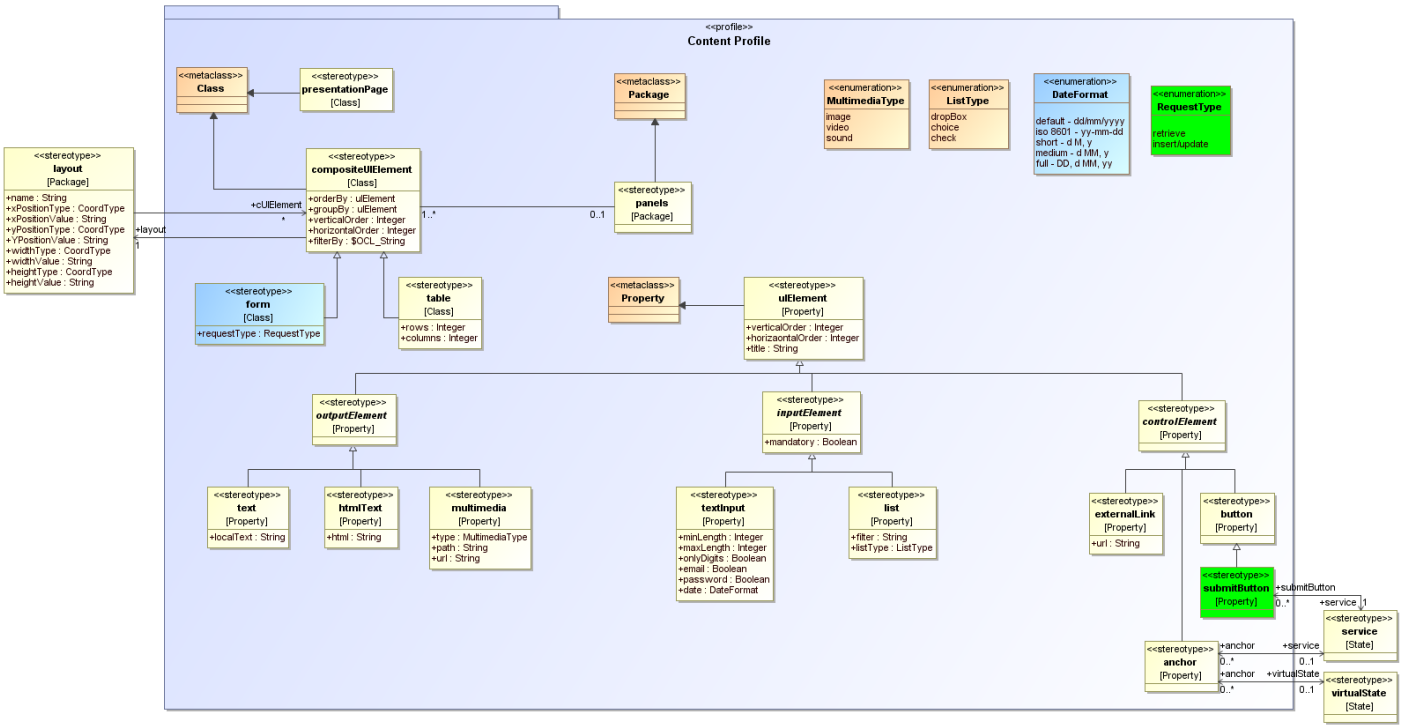


Figura 4.3: Metamodelo y perfil de contenido extendido.

4.4. ASM RIA

A partir de las extensiones propuestas en las secciones anteriores, ahora es posible presentar las extensiones que definen las funcionalidades RIA para MoWebA. Las funcionalidades abarcan las características de almacenamiento de datos en el cliente, lógica de negocios en el cliente y comunicación asíncrona entre cliente y servidor, y se presentan en el metamodelo y perfil RIA de la *Figura 4.4*. Los nuevos elementos especializan o se relacionan con elementos presentes en los diagramas lógico y de contenido.

A continuación se presentarán los nuevos elementos según la característica RIA que describen.

4.4.1. Almacenamiento de Datos en el Cliente

Para guardar datos en un cliente web, específicamente en un navegador web, se han creado dos nuevas clases, un objeto estático en el cliente (*ClientStaticObject*) y un objeto de valor en el cliente (*ClientValueObject*), ellos extienden al objeto estático y al objeto de valor del diagrama lógico respectivamente.

Los nuevos objetos se diferencian de los originales a nivel de implementación, ya que tanto el *ClientStaticObject* como el *ClientValueObject* serán mapeados a variables almacenadas en el navegador. El nombre de la clase corresponde al nombre de la variable y los valores de la clase con sus valores etiquetados asociados (*title* y *checked*) dan lugar a los valores de la variable. Además, estas nuevas clases permiten especificar un nivel de persistencia (*persistence*) que puede ser permanente (*permanent*) o temporal (*temporary*), permitiendo que la variable persista o no al término de una sesión o cierre del navegador. Si el nivel de persistencia es permanente se generará el objeto Localstorage de HTML, mientras que si la persistencia es temporal se generará el objeto Sessionstorage de HTML. Estos objetos serán generados tanto para el *ClientStaticObject* como para el *ClientValueObject*, siendo la propiedad de persistencia la que decida cuál será el objeto a generar.

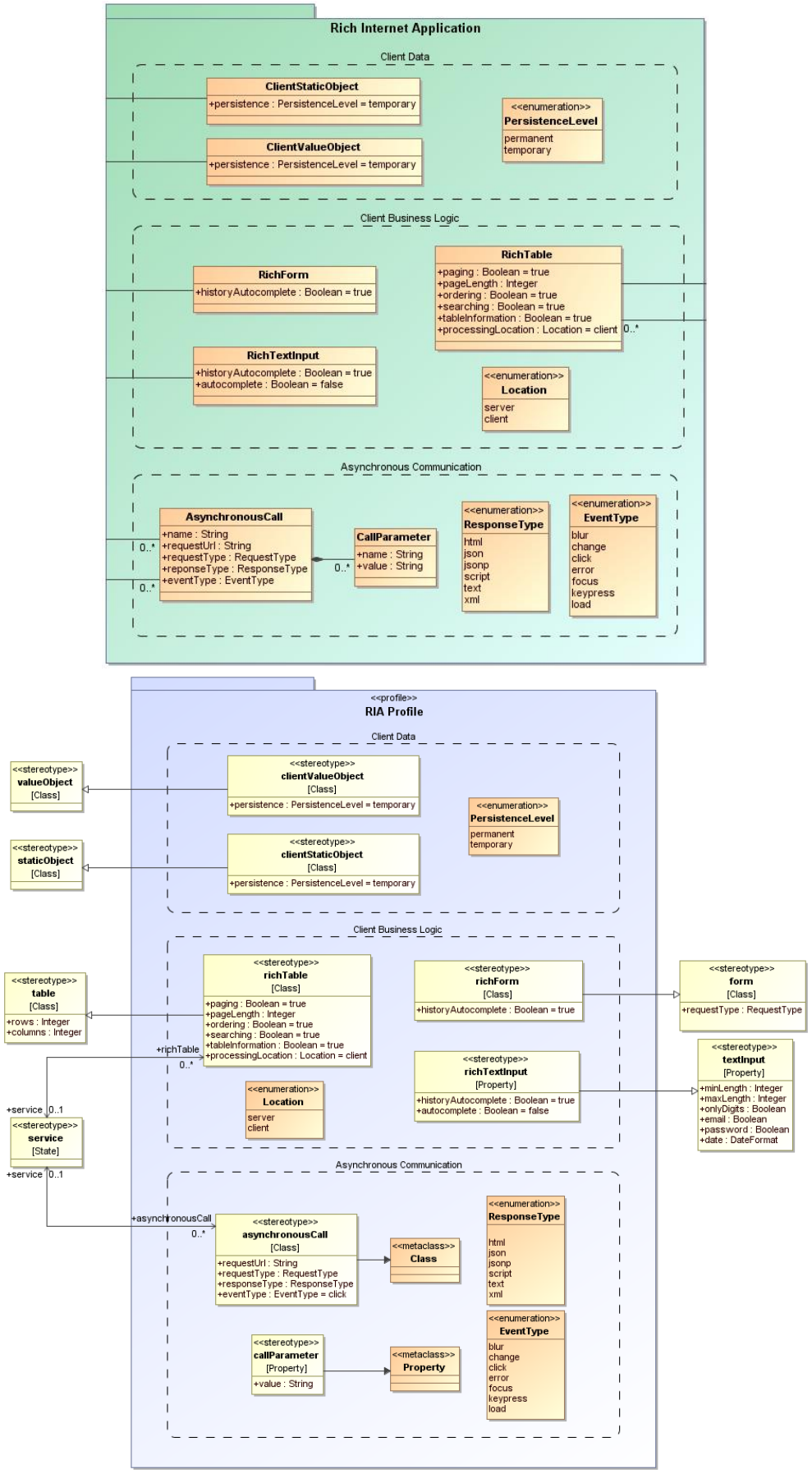


Figura 4.4: Metamodelo y perfil RIA.

4.4.2. Lógica de Negocios en el Cliente

En cuanto a procesos ejecutados en el cliente, se han creado tres elementos que se ejecutan en el navegador web, el formulario enriquecido (*RichForm*), la entrada de texto enriquecida (*RichTextInput*) y la tabla enriquecida (*RichTable*).

El formulario enriquecido consiste en una especialización del elemento formulario (*Form*) del diagrama de contenido. Contiene el atributo de autocompletado basado en historia (*historyAutocomplete*), que permite al navegador sugerir valores para completar las entradas del formulario, basados en valores previamente ingresados. A partir de cada instancia de un formulario enriquecido se generará un formulario HTML con la propiedad de autocompletado habilitada o no según lo establecido en el modelo.

La entrada de texto enriquecida extiende a la entrada de texto (*TextInput*) del diagrama de contenido. Así como el formulario enriquecido, la entrada de texto enriquecida posee el atributo de autocompletado basado en historia, pero además introduce el atributo de autocompletado (*autocomplete*) que se diferencia del anterior en que los valores a sugerir los obtiene a partir de una asociación con objetos de valor u objetos estáticos y sus especializaciones. Posteriormente, haciendo uso de jQuery UI se generará la asociación entre entradas y valores y se desplegará la opción de autocompletado.

La tabla enriquecida especializa a la clase tabla (*Table*) del diagrama de contenido. Permite especificar el servicio del diagrama de nodos encargado de poblar la tabla. Además, agrega nuevas propiedades a la tabla como permitir paginación (*paging*), especificar cantidad de registros por página (*pageLength*), permitir ordenamiento de columnas (*ordering*), buscar palabras en registros (*searching*), desplegar información de la tabla (*tableInformation*) y especificar dónde se procesarán estas funciones (*processingLocation*), ya sea en el servidor (*server*) o en el cliente (*client*). El *plugin* de jQuery, Datatables será utilizado para la implementación de una tabla enriquecida.

4.4.3. Comunicación Asíncrona entre Cliente y Servidor

Para permitir comunicaciones asíncronas entre cliente y servidor se ha creado la clase denominada llamada asíncrona (*AsynchronousCall*). Esta clase es utilizada para realizar una solicitud AJAX (HTTP asíncrona) de un servicio.

La llamada asíncrona se asocia a uno o muchos elementos de interfaz de usuario del diagrama de contenido y a cero o un servicio del diagrama de nodos. Cuando ocurre un evento sobre algún elemento, se ejecuta un servicio. El servicio puede ser alguno especificado en el diagrama de nodos, o un servicio ejecutándose desde una URL ajena al sistema (por ejemplo un web service). La instancia de la llamada asíncrona puede ubicarse en el diagrama de contenido junto a sus elementos de interfaz de usuario asociados.

La clase posee un nombre (*name*), una URL (*requestUrl*) (utilizado en caso de que solicite un servicio externo), un tipo de solicitud (*requestType*) que funciona de igual manera al tipo de solicitud del formulario, un tipo de respuesta (*responseType*) que puede ser en formato *html*, *json*, *jsonp*, *script*, *text*, *xml* y un tipo de evento (*eventType*), que se aplica al elemento de interfaz de usuario y puede tomar los valores *blur*, *change*, *click*, *error*, *focus*, *keypress*, *load*. Además, la llamada asíncrona puede adjuntar ciertos parámetros a la solicitud, cada parámetro consiste en una propiedad de la clase y posee un nombre y un valor. Cada llamada asíncrona generará un método *ajax()* de jQuery.

4.5. Reglas de Transformación

Luego de la elaboración del ASM de la aplicación es posible generar código a partir de él. Para ello, se han definido reglas de transformación de modelo a texto haciendo uso de la herramienta Acceleo.

El modelo definido en la herramienta MagicDraw debe ser exportado en formato EMF UML2 (v2.x) XMI y posteriormente importado en la herramienta Acceleo. Luego se ejecuta la aplicación Acceleo que utiliza las reglas de transformación definidas generando el código correspondiente a la aplicación modelada.

En el Anexo 2 se presentan las reglas de transformación elaboradas. Éstas siguen un enfoque basado en plantillas en el cual se especifican plantillas de texto con entradas para datos a extraer de los diagramas del modelo. Se utiliza el lenguaje MTL para la definición de las plantillas y OCL para realizar las consultas al modelo. Además, se utilizan servicios definidos en Java para extender a MTL con mayores funcionalidades.

Las reglas de transformación se encargan de realizar un mapeo entre elementos definidos en los diagramas lógico y de contenido del modelo de la aplicación con código en los lenguajes HTML y Javascript y las librerías jQuery, jQueryUI, y Datatables.

A continuación se presentará la estructura de archivos generada. Posteriormente se detallarán las reglas de transformación que parten del diagrama lógico. Finalmente se describirán las reglas de transformación que se inician a partir del diagrama de contenido.

4.5.1. Estructura de Archivos Generada

El sistema generado a partir de las reglas de transformación está compuesto por una carpeta raíz de nombre *RIA*. En el interior de esta carpeta se encuentran directorios (uno por cada página de la aplicación), los cuales están compuestos a su vez por un archivo **.html* y un archivo **.js* (en el caso de que el archivo **.html* utilice scripts definidos en el archivo **.js*). Además, dentro de la misma carpeta se encuentran un archivo de nombre *variables.js* que define las variables locales del sistema y una carpeta de nombre *servicios* que contiene archivos **.php* (uno por cada servicio invocado en la aplicación) a completar manualmente por el desarrollador. En la *Figura 4.5* se puede observar la estructura de archivos generada para un sistema de marcación de empleados utilizado como ejemplo de modelado y generación de código para las secciones posteriores.

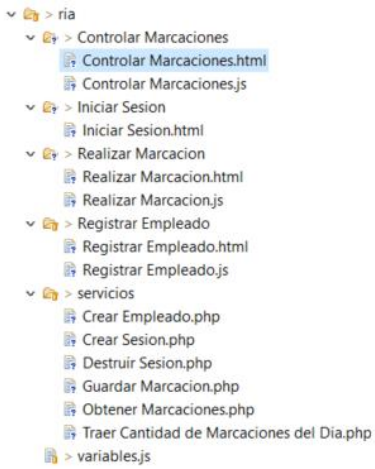


Figura 4.5: Estructura de archivos para el sistema de marcación de empleados.

4.5.2. Reglas de Transformación a partir del Diagrama Lógico

Se definieron reglas de transformación que toman como punto de partida el diagrama lógico. Específicamente se mapean los objetos estáticos en el cliente (*ClientStaticObject*) con variables almacenadas del lado del cliente especificadas mediante las funcionalidades LocalStorage y SessionStorage de HTML5 en el archivo *variables.js*.

Para cada objeto estático en el cliente especificado en el diagrama se crea un arreglo de objetos. Este arreglo posee el mismo nombre que el objeto estático en el cliente. Por cada valor del objeto estático se agrega un elemento al arreglo. Cada elemento consiste en un objeto con las propiedades *title*, *value* y *checked*. Los valores de las propiedades se obtienen del valor correspondiente del objeto estático en el cliente en cuestión.

De acuerdo al nivel de persistencia especificado en el objeto estático en el cliente se decide si utilizar un objeto SessionStorage o un objeto LocalStorage. Si el nivel de persistencia es temporal se crea un objeto SessionStorage, mientras que si el nivel de persistencia es permanente se crea un objeto LocalStorage.

4.5.3. Reglas de Transformación a partir del Diagrama de Contenido

Además de las reglas de transformación anteriores, se desarrollaron otras que parten de los diagramas de contenido. Por cada página de presentación de un diagrama de contenido se genera un directorio con el mismo nombre. Este directorio está compuesto por un archivo *.html y de ser necesario un archivo *.js, ambos con el mismo nombre que el directorio (y la página de presentación asociada).

La página de presentación puede estar compuesta por distintos tipos de contenedores. A partir de contenedores del tipo tabla (*Table*) se generan tablas HTML. Desde contenedores del tipo tabla enriquecida (*RichTable*) se generan tablas HTML con acciones dinámicas (paginación, filtrado, ordenamiento) implementadas mediante jQuery Datatables. Desde los contenedores del tipo formulario (*Form*) y formulario enriquecido (*RichForm*) se generan formularios HTML, con la diferencia de que este último permite el autocompletado basado en historia. Por último, a partir de elementos compuestos de interfaz de usuario (*CompositeUIElement*) se generan etiquetas *div* de HTML.

Estos contenedores pueden estar compuestos por diferentes elementos de interfaz de usuario (*UIElement*) y cada uno de éstos es mapeado a código de la siguiente forma:

- Los elementos texto (*Text*) y texto HTML (*HtmlText*) se mapean a texto HTML de forma directa.
- El elemento multimedia (*Multimedia*) es mapeado a código cuando el elemento es del tipo imagen (*image*), generando imágenes HTML a partir del URL (*url*) o camino (*path*) especificado.
- La entrada de texto (*TextInput*) y sus propiedades son mapeadas al elemento *input* de HTML junto a sus atributos *pattern*, *maxlength*, *required* y *type* para la implementación de restricciones.
- La entrada de texto enriquecida (*RichTextInput*) corresponde a una entrada de texto como la anterior, con la diferencia de que se vale de una función especificada mediante jQueryUI para proveer autocompletado de valores. Si los valores se obtienen de un objeto estático, éstos serán especificadas mediante Javascript, si se obtienen de un objeto estático en el cliente se obtendrán mediante Javascript haciendo uso de las variables del lado del cliente definidas en el archivo *variables.js*.

- El elemento lista (*List*) es mapeado a código según su tipo de lista (*listType*) correspondiente. Si el tipo de lista es *dropbox* se genera el elemento *select* de HTML, si el tipo de lista es *choice* se generan entradas del tipo *radio* de HTML, si el tipo de lista es *check* se generan entradas del tipo *checkbox* de HTML. Si la lista se encuentra asociada a un objeto estático, ésta será poblada mediante HTML, si se encuentra asociada a un objeto estático en el cliente, la lista será poblada mediante Javascript haciendo uso de las variables del lado del cliente generadas en el archivo *variables.js*.
- El elemento link externo (*ExternalLink*) es mapeado a un hiperenlace HTML apuntando a la URL (*url*) especificada.
- El elemento enlace o ancla (*Anchor*) genera también un hiperenlace HTML, pero apuntando a la página asociada del estado virtual (*VirtualState*) especificado.
- Los elementos botón y botón de envío son mapeados a sus correspondientes implementaciones en HTML.

Las llamadas asíncronas (*AsynchronousCall*) a servicios, junto a sus parámetros y valores etiquetados son mapeados a métodos *ajax()* de jQuery. La respuesta a la llamada es almacenada en una variable de nombre *result*, la cual puede ser posteriormente manipulada manualmente.

Finalmente, los servicios invocados, ya sea en un formulario, formulario enriquecido, tabla enriquecida o llamada asíncrona son contenidos como archivos *.php en un directorio de nombre *servicios*. Cada archivo tiene el mismo nombre que el servicio asociado y su contenido debe ser especificado de forma manual, debido a que los servicios implementados del lado del servidor se encuentran fuera del alcance de este enfoque.

4.6. Ejemplo de Modelado y Generación de Código

A continuación se presenta un ejemplo de modelado y generación de código de una aplicación haciendo uso de MoWebA con las extensiones RIA propuestas.

La aplicación consiste en un sistema de marcación de empleados en el cual un usuario invitado puede registrarse como empleado para posteriormente iniciar sesión y poder realizar marcaciones de entrada o salida. Si el usuario es un supervisor, es capaz de observar las marcaciones realizadas por los empleados.

Se siguió el proceso estipulado por MoWebA, el cual incluye la especificación del CIM, PIM, ASM y PSM de la aplicación, sistematizado en sus distintas etapas con sus diagramas correspondientes.

A continuación se presenta el proceso de modelado separado en dos fases, una para la elaboración de los diagramas del CIM/PIM y otra para los diagramas del ASM/PSM. Por cada diagrama del ASM/PSM, se exhibe el código generado junto a capturas de pantalla de la aplicación final desplegada. Se destacan los diagramas que incluyen los elementos recién definidos y/o elementos que sufrirán modificaciones al pasar del CIM/PIM al ASM/PSM, dejando los diagramas restantes en el Anexo 3.

4.6.1. CIM/PIM

Se han elaborado los distintos modelos y diagramas correspondientes al CIM (modelo de casos de uso) y PIM (modelos de entidades, navegacional, comportamiento, presentación, usuario) de MoWebA para el sistema de marcación de empleados.

La *Figura 4.6* presenta el diagrama lógico de la aplicación. Se definen procesos de negocios encapsulados en las clases *ManejoSesiones*, *AlmacenamientoDatos* y *RecuperacionDatos*.

También se definen los objetos de valor *EmpleadosRegistroVO* y *EmpleadosInicioVO* que dependen de la entidad *Empleados* y el objeto de valor *MarcacionesVO*. Se introducen cuatro objetos estáticos, *SexosSO*, *TipoMarcacionesSO*, *PaisesSO* y *CargosSO* definidas como clases con el estereotipo <<*staticObjetc*>> con propiedades representando valores. Las propiedades utilizan el estereotipo <<*value*>> y poseen valores etiquetados para el título de los valores (para desplegar en la interfaz) y para chequear el valor (para pre-seleccionar un valor en una lista).

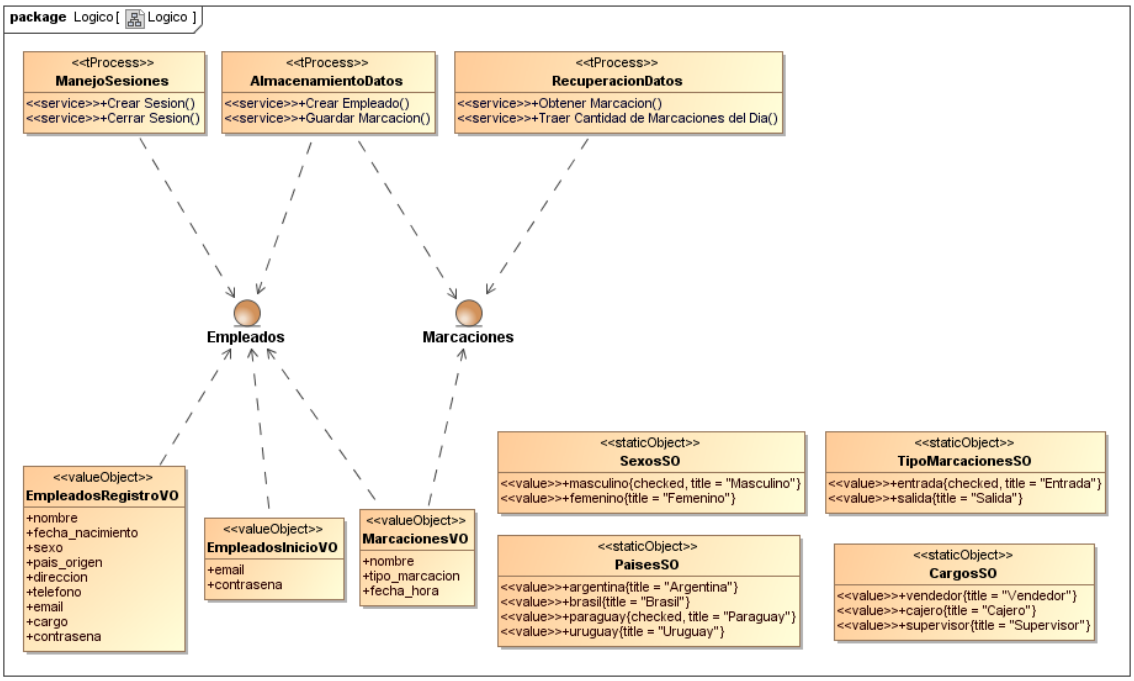


Figura 4.6: Diagrama lógico del sistema de marcación de empleados.

Se elaboraron diagramas de contenido para inicio de sesión, registro de empleados, realización de marcaciones y control de marcaciones. Los últimos tres serán explicados a continuación debido a que ellos abarcan la mayor cantidad de elementos de modelado que luego serán aprovechados para incluir funcionalidades RIA.

En la *Figura 4.7* se observa el diagrama de contenido para el registro de empleados. Contiene la página de presentación *Registrar Empleado* compuesto por el formulario *RegistroEmpleado*. Este formulario posee valores etiquetados para el tipo de solicitud, que en este caso toma el valor *insert*/*update*. Además, el formulario posee entradas de texto con valores etiquetados para las restricciones como *mandatory*, *minLength*, *maxLength*, formato *date* (*default – dd/mm/yyyy*), *onlyDigits*, *email* y *password*. También se incluyen dos listas con valores etiquetados para los tipos *choice* y *dropbox*, éstas se asocian a los objetos estáticos definidos en el diagrama lógico para desplegar los valores de los objetos en la lista. Se agregó un enlace con un valor etiquetado que permite volver a la página de presentación *Iniciar Sesión* (que se encuentra relacionada con el estado virtual del diagrama de nodos *Iniciar Sesión*). Por último, se define un botón para el envío del formulario haciendo uso del estereotipo <<*submitButton*>>, como valor etiquetado se especifica el servicio encargado del procesamiento del formulario, en este caso el servicio del diagrama de nodos *Crear Empleado*.

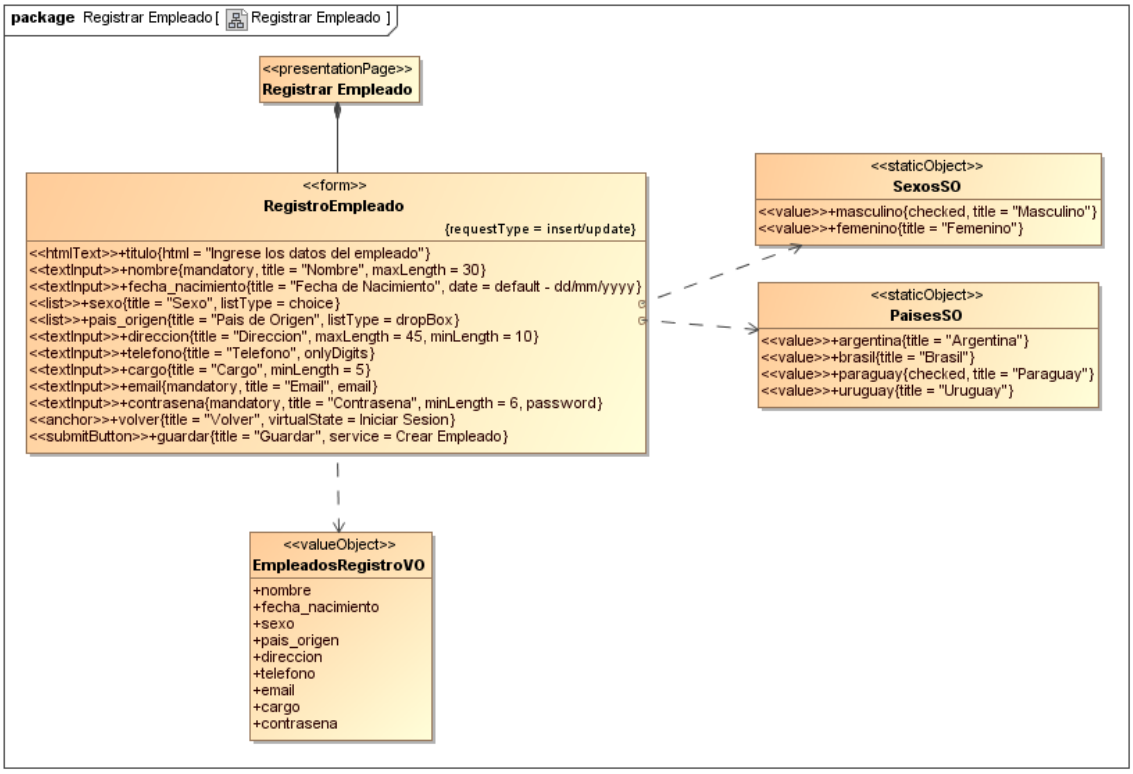


Figura 4.7: Diagrama de contenido para registro de empleados del sistema de marcación de empleados.

La *Figura 4.8* contiene el diagrama de contenido para el control de marcaciones. Se dispone de la página de presentación *Controlar Marcaciones*. Esta página está compuesta por la tabla *Marcaciones* y el elemento compuesto de interfaz de usuario *CerrarSesion*. La tabla posee los encabezados declarados como texto HTML *nombre*, *tipo_marcacion* y *fecha_hora*. El elemento compuesto de interfaz de usuario contiene un enlace para cerrar la sesión de usuario, para ello se especifica como valor etiquetado el servicio del diagrama de nodos *Destruir Sesion* encargado de eliminar la sesión y de redirigir al usuario a donde sea necesario.

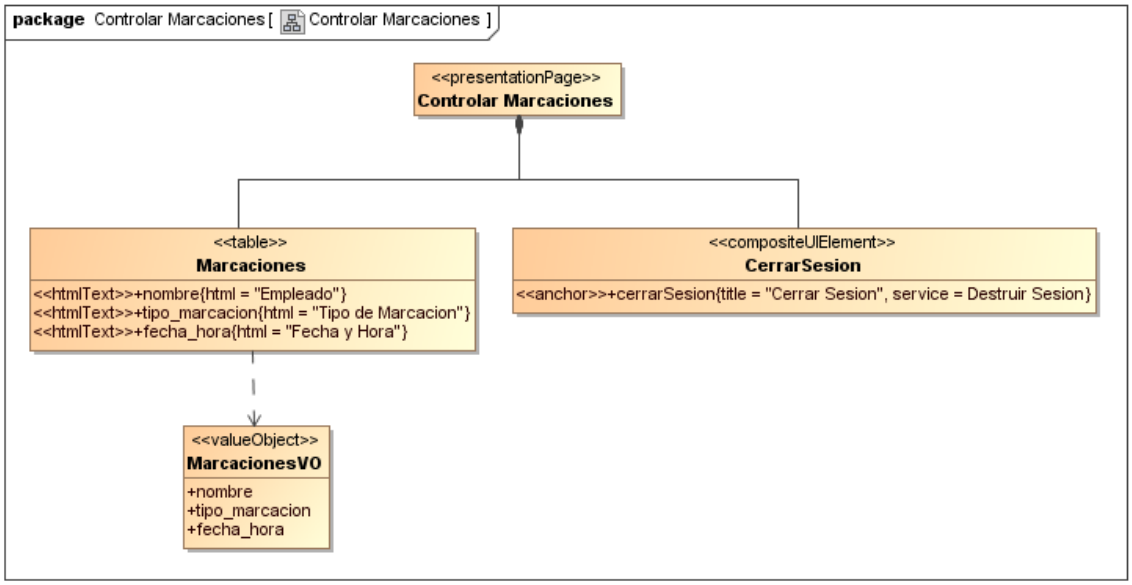


Figura 4.8: Diagrama de contenido para control de marcaciones del sistema de marcación de empleados.

La *Figura 4.9* presenta el diagrama de contenido para la realización de marcaciones. Posee la página de presentación *Realizar Marcación* que se encuentra a su vez compuesta por los elementos compuestos de interfaz de usuario *MarcacionesDelDia* y *CerrarSesion* y el formulario *IngresarMarcacion*. El elemento de interfaz de usuario *MarcacionesDelDia* contiene al enlace *actualizarMarcaciones*. Al presionar sobre este enlace se ejecuta el servicio *Traer Cantidad de Marcaciones del Dia*, que se encarga de actualizar el texto HTML *cantidadMarcaciones* de forma síncrona (con una recarga de la página entera). El formulario *IngresarMarcacion* y el elemento compuesto de interfaz de usuario *CerrarSesion* están definidos de la misma forma que en los diagramas de contenido anteriores.

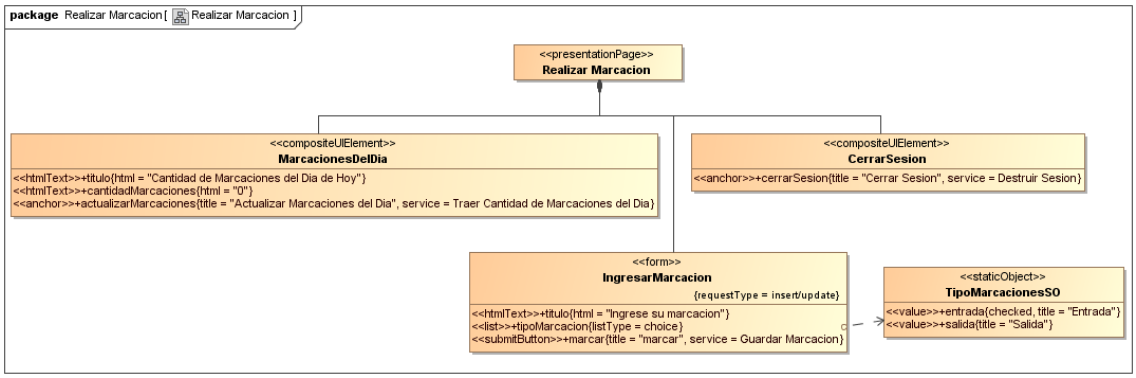


Figura 4.9: Diagrama de contenido para realización de marcaciones del sistema de marcación de empleados.

4.6.2. ASM/PSM

Posteriormente al modelado del CIM/PIM de la aplicación y siguiendo el proceso establecido por MoWebA, se procedió a hacer uso de las extensiones RIA propuestas para la definición de un ASM sobre el modelo elaborado. Cabe destacar que la propuesta no abarca la definición de un PSM por separado, debido a que el ASM ya contempla información de plataforma específica para tecnologías basadas en scripting (Javascript, jQuery, jQuery UI, Datatables).

A continuación se presentarán las variaciones que han sufrido los modelos del PIM introducidos en la sección anterior para aprovechar las ventajas ofrecidas por las RIA, junto al código generado y capturas de pantalla de la aplicación final. Se utiliza la misma convención de colores utilizada para las extensiones al PIM, el azul para las clases que han sido modificadas y el verde para las nuevas clases agregadas.

En primer lugar, se ha extendido el diagrama lógico de la aplicación con el fin de lograr almacenar datos en el cliente web (navegador). En la *Figura 4.10* podemos observar dos variaciones con respecto al diagrama de la *Figura 4.6*. Las clases *PaisesSO* y *CargosSO* ahora poseen el estereotipo *<<clientStaticObject>>* y corresponden a objetos estáticos almacenados en el cliente. Ambos poseen valores etiquetados para indicar su nivel de persistencia. *PaisesSO* tiene un nivel de persistencia temporal, mientras que *CargosSO* tiene un nivel de persistencia permanente.

A partir del diagrama lógico expuesto se genera el archivo *variables.js* de la *Figura 4.11*. Éste contiene las variables de lado del cliente *CargosSO* y *PaisesSO* definidas a partir de los objetos estáticos en el cliente de igual nombre, implementadas mediante los objetos LocalStorage y SessionStorage respectivamente.

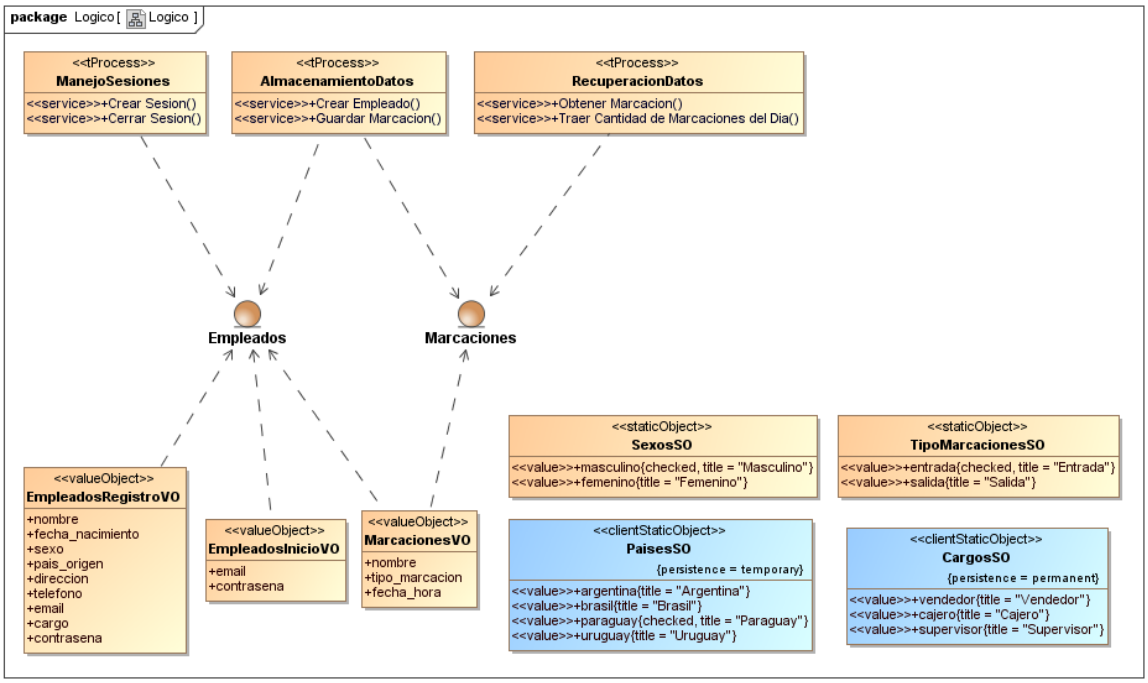


Figura 4.10: Diagrama lógico con extensiones RIA del sistema de marcación de empleados.

variables.js

```
var CargosSO = [];
CargosSO[0] = {title: "Vendedor", value:"vendedor", checked:false};
CargosSO[1] = {title: "Cajero", value:"cajero", checked:false};
CargosSO[2] = {title: "Supervisor", value:"supervisor", checked:false};
localStorage["CargosSO"] = JSON.stringify(CargosSO);
CargosSO = JSON.parse(localStorage["CargosSO"]);

var PaisesSO = [];
PaisesSO[0] = {title: "Argentina", value:"argentina", checked:false};
PaisesSO[1] = {title: "Brasil", value:"brasil", checked:false};
PaisesSO[2] = {title: "Paraguay", value:"paraguay", checked:true};
PaisesSO[3] = {title: "Uruguay", value:"uruguay", checked:false};
sessionStorage["PaisesSO"] = JSON.stringify(PaisesSO);
PaisesSO = JSON.parse(sessionStorage["PaisesSO"]);
```

Figura 4.11: Archivo *variables.js* para el sistema de marcación de empleados.

El diagrama de contenido para registro de empleados con extensiones para lógica de negocios en el cliente se observa en la *Figura 4.12*. A diferencia de la *Figura 4.7*, ahora el formulario corresponde a un formulario enriquecido gracias a la aplicación del estereotipo <<*richForm*>> sobre la clase. A partir de esto, se ha utilizado el valor etiquetado de autocompletado basado en historia para desactivar esta acción (si no se especifica nada, la opción de autocompletado basado en historia se encuentra habilitada por defecto en los formularios HTML). Las entradas de texto nombre y cargo ahora poseen el estereotipo <<*richTextInput*>>. La entrada nombre tiene el valor etiquetado para autocompletado basado en historia, permitiendo habilitar esta acción para la entrada a pesar de que se encuentre deshabilitada para el formulario. La entrada cargo posee el valor etiquetado de autocompletado permitiendo obtener los valores para autocompletado a través de la asociación de dependencia con el objeto estático en el cliente *CargosSO*. Por último, el elemento lista *país_origen* ahora obtiene sus valores a desplegar del objeto estático en el cliente *PaisesSO* gracias a la relación de dependencia entre ellos.

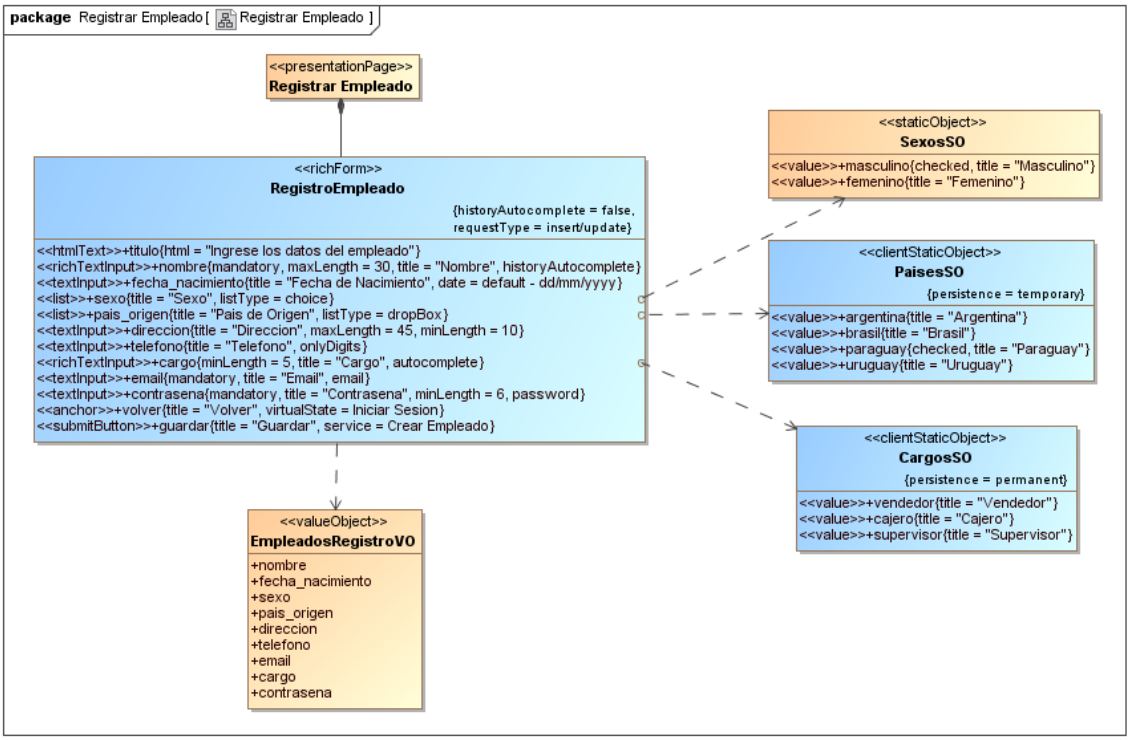


Figura 4.12: Diagrama de contenido para registro de empleados con extensiones RIA del sistema de marcación de empleados.

La *Figura 4.13* presenta el archivo *Registrar Empleado.js* generado a partir del diagrama de contenido anterior. En él es posible observar dos funciones, la primera representa código Javascript encargado de poblar la lista *pais_origen* con los valores de la variable *PaisesSO*, la segunda especifica código Javascript y jQueryUI encargado de obtener valores de la variable *CargosSO* y asignarlos a la entrada de texto *cargo* y así desplegarlos como autocompletado. La página generada para el registro de empleados puede observarse en la *Figura 4.14*.

Registrar Empleado.js

```
1  window.addEventListener('load', function(){
2      var list = document.getElementById("pais_origen");
3      for(var i = 0; i < PaisesSO.length; i++) {
4          var option = document.createElement("option");
5          option.textContent = PaisesSO[i].title;
6          option.value = PaisesSO[i].value;
7          option.selected = PaisesSO[i].checked;
8          list.appendChild(option);
9      }
10 });
11
12 window.addEventListener('load', function(){
13     $(function(){
14         var cargoTags = [
15         CargosSO[0].title,
16         CargosSO[1].title,
17         CargosSO[2].title
18         ];
19         $("#cargo").autocomplete({
20             source: cargoTags
21         });
22     });
23 });
24
```

Figura 4.13: Archivo *Registrar Empleado.js* para el sistema de marcación de empleados.

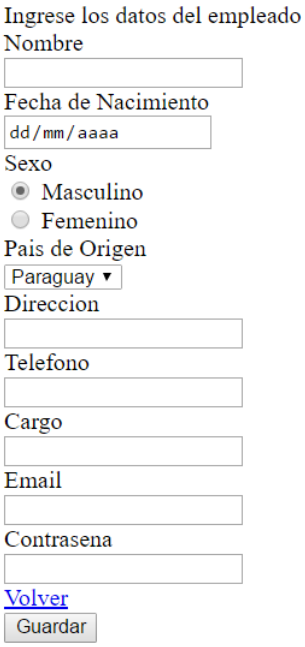


Figura 4.14: Página para registro de empleados del sistema de marcación de empleados.

El diagrama de contenido para control de marcaciones extendido para RIA se puede apreciar en la *Figura 4.15*. La lógica de negocios en el cliente también es aprovechada, en este caso haciendo uso de la tabla enriquecida. En lugar de la clase del tipo tabla de la *Figura 4.8*, ahora se tiene una clase con estereotipo *<<richTable>>*, representando una tabla enriquecida con operaciones que pueden realizarse del lado del cliente o del servidor. Se utilizan los valores etiquetados para habilitar ordenamiento, paginación y búsqueda del lado del cliente. Además, se especifica el máximo de registros por página (15) y se habilita la información de la tabla. Por último, se fija el servicio *Obtener Marcaciones* del diagrama de nodos como encargado de proveer registros a la tabla.

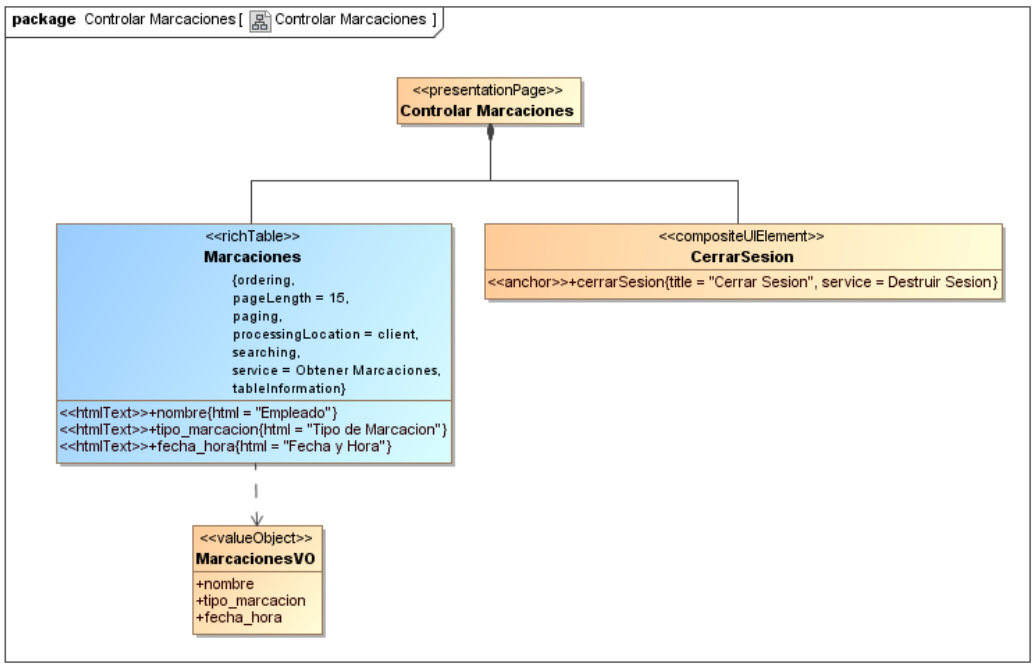


Figura 4.15: Diagrama de contenido para control de marcaciones con extensiones RIA del sistema de marcación de empleados.

En la *Figura 4.16* se observa el archivo *Controlar Marcaciones.js* generado a partir del diagrama de contenido recién descrito. Este archivo contiene código jQuery DataTables asociado a la tabla enriquecida *Marcaciones* del diagrama de contenido para el control de marcaciones, permitiendo proveer funcionalidades como paginación, ordenamiento y búsqueda del lado del cliente y la obtención de registros del servicio *Obtener Marcaciones.php*. La página generada para el control de marcaciones se presenta en la *Figura 4.17*.

Controlar Marcaciones.js

```
1  $(document).ready(function() {
2      $('#Marcaciones').DataTable({
3          "paging": true,
4          "pageLength": 15,
5          "ordering": true,
6          "searching": true,
7          "info": true,
8          "serverSide": false,
9          "ajax": "/ria/servicios/Obtener Marcaciones.php"
10     });
11 });
12
```

Figura 4.16: Archivo *Controlar Marcaciones.js* para el sistema de marcación de empleados.

Show 10 entries Search:

Empleado	Tipo de Marcacion	Fecha y Hora
esanchiz	Entrada	2016-08-15 08:02:00
esanchiz	Salida	2016-08-15 17:01:00
esanchiz	Entrada	2016-08-16 07:57:00
esanchiz	Salida	2016-08-16 17:00:00
esanchiz	Entrada	2016-08-17 08:07:00
esanchiz	Salida	2016-08-17 16:55:00
lriquelme	Entrada	2016-08-15 08:02:00
lriquelme	Salida	2016-08-15 17:01:00
lriquelme	Entrada	2016-08-16 07:57:00
lriquelme	Salida	2016-08-16 17:00:00

Showing 1 to 10 of 30 entries Previous 1 2 3 Next

Cerrar Sesion

Figura 4.17: Página para control de marcaciones del sistema de marcación de empleados.

La invocación asíncrona de servicios puede observarse en el diagrama de contenido para realización de marcaciones extendido de la *Figura 4.18*. Al diagrama de la *Figura 4.9* se le agregan dos nuevas clases que representan llamadas asíncronas de servicios. La primera llamada es especificada a través de la clase *traerCantidadDeMarcacionesDelDia* con el estereotipo <<*asynchronousCall*>>. Esta clase se asocia mediante una relación de dependencia con el elemento de interfaz de usuario que lanza la llamada, en este caso el enlace *actualizarMarcaciones*. Por medio de valores etiquetados se especifican el evento que ocurre sobre el elemento de interfaz de usuario para lanzar la llamada (*click*), el tipo de solicitud (*retrieve*), el tipo de respuesta (*html*) y el servicio del diagrama de nodos que es llamado (*Traer Cantidad de Marcaciones del Dia*). Los parámetros de la llamada se definen como propiedades

de la clase con el estereotipo <<*callParameter*>>. Esta llamada pasa como parámetro la variable de nombre *fechaActual* con valor *CURDATE()* definido como valor etiquetado. La segunda llamada asíncrona *guardarMarcacion* se asocia al botón de envío *marcar* del formulario enriquecido *IngresarMarcacion* y establece a través de valores etiquetados el tipo de evento sobre el botón de envío (*click*), el tipo de solicitud (*insert/update*), el tipo de respuesta (*html*) y el servicio a ejecutar (*Guardar Marcacion*). A diferencia de la llamada asíncrona anterior, ésta última no especifica ningún parámetro, sin embargo, como se encuentra relacionado con un formulario, los datos del formulario (tipo de marcación) serán enviados al servicio como parámetros.

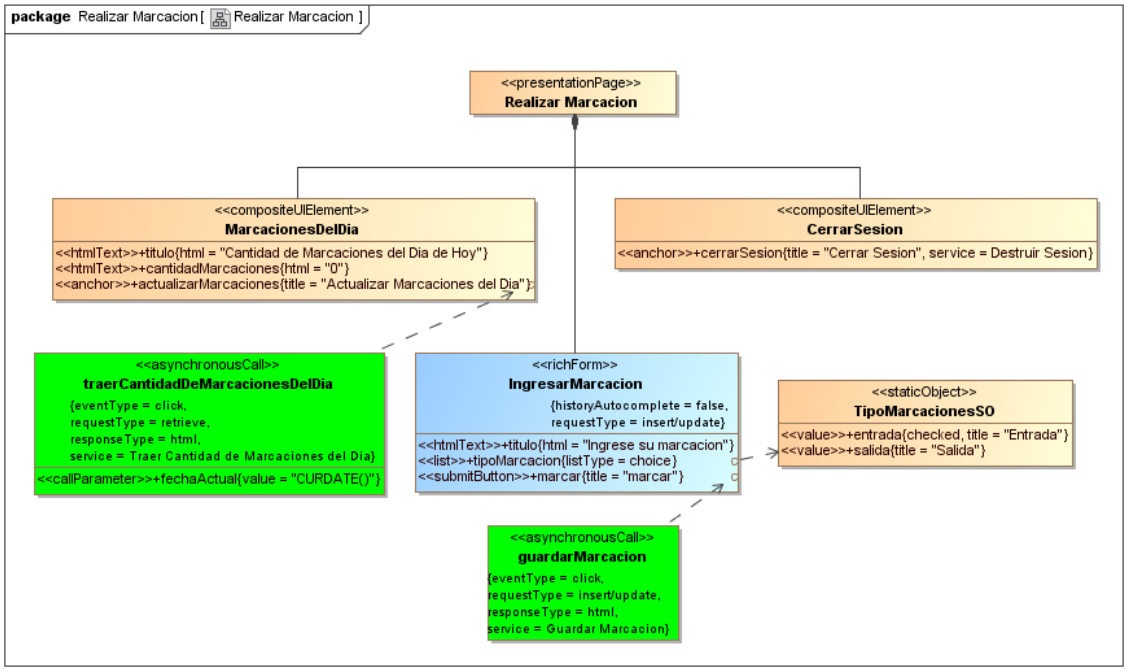


Figura 4.18: Diagrama de contenido para realización de marcaciones con extensiones RIA del sistema de marcación de empleados.

La *Figura 4.19* introduce el archivo *Realizar Marcacion.js* generado a partir del diagrama recién mencionado. Se pueden observar dos funciones con eventos *ajax()* de jQuery que implementan llamadas asíncronas a servicios. La primera realiza una llamada asíncrona al servicio *Traer Cantidad de Marcaciones del Dia.php* cuando ocurre un evento *click* sobre el enlace *actualizarMarcaciones*. La segunda realiza la llamada al servicio *Guardar Marcacion.php* cuando ocurre el evento *click* sobre el botón de envío *marcar*. La página generada para la realización de marcaciones se observa en la *Figura 4.20*.

4.7. Resumen del Capítulo

En este capítulo presentamos la propuesta de solución elaborada. Describimos el proceso de desarrollo a seguir, las extensiones realizadas a los metamodelos y perfiles UML de los diagramas lógico y de contenido de MoWebA, y el metamodelo y perfil UML para el ASM que contempla las funcionalidades RIA con las características de distribución de datos, lógica de negocios y comunicación asíncrona entre cliente y servidor.

A nivel de modelado, se introdujeron los elementos *ClientValueObject* y *ClientStaticObject* que representan variables almacenadas del lado del cliente; los elementos *RichTable*, *RichForm*, y *RichTextInput* que representan tablas, formularios y entradas de texto como elementos de interfaz de usuario con operaciones de lógica de negocios (paginación, ordenamiento, búsqueda,

autocompletado) ejecutadas del lado del cliente; y el elemento *AsynchronousCall* que representa la invocación asíncrona de un servicio desde un elemento de presentación.

A nivel de implementación, se ha generado código HTML5 para elementos de interfaz de usuario y código Javascript para el comportamiento de dichos elementos. Las funcionalidades de HTML5 LocalStorage y SessionStorage fueron aprovechadas para el almacenamiento de datos del lado del cliente. JQuery con sus *plugins* Datatables y jQuery UI fueron utilizados para la lógica de negocios del lado del cliente. JQuery también fue utilizado para implementar las invocaciones asíncronas.

A partir de un ejemplo de modelado y generación describimos como las extensiones propuestas son utilizadas para la generación de una RIA funcional.

Realizar Marcacion.js

```
window.addEventListener('load', function(){
    $("#actualizarMarcaciones").click(function traerCantidadDeMarcacionesDelDia(event){
        event.preventDefault();
        $.ajax({
            "url": "/ria/servicios/Traer Cantidad de Marcaciones del Dia.php",
            data: {
                fechaActual: "CURDATE()"
            },
            type: "GET",
            dataType: "html",
            success: function(result){

            }
        });
    });
});

window.addEventListener('load', function(){
    $("#marcar").click(function guardarMarcacion(event){
        event.preventDefault();
        $.ajax({
            "url": "/ria/servicios/Guardar Marcacion.php",
            data : $('#IngresarMarcacion').serialize(),
            type: "POST",
            dataType: "html",
            success: function(result){

            }
        });
    });
});
```

Figura 4.19: Archivo *Realizar Marcacion.js* para el sistema de marcación de empleados.

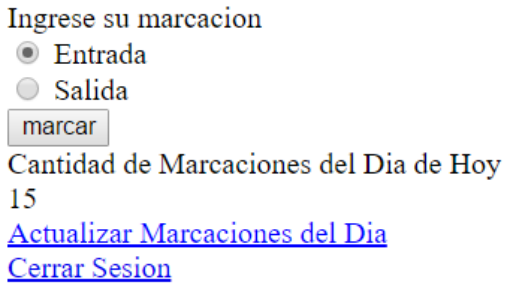


Figura 4.20: Página para realización de marcaciones del sistema de marcación de empleados.

5. Validación de la Propuesta

Dada la propuesta de solución presentada en el capítulo anterior, surge la necesidad de realizar un procedimiento para comprobar si nos encontramos con una propuesta válida y útil.

Es por ello que hemos realizado una experiencia de validación cuyo objetivo consiste en evaluar la usabilidad del enfoque MDD propuesto. Para ello, se ha solicitado la elaboración de una RIA funcional a los alumnos de último año de la carrera de Ingeniería Informática de la Universidad Católica "Nuestra Señora de la Asunción".

La ISO 9241-11 [31] provee una guía sobre usabilidad y la define como "la medida en que un producto puede ser utilizado por usuarios específicos para lograr objetivos específicos con eficacia, eficiencia y satisfacción en un contexto específico de uso". Además, la eficacia se define como "la exactitud y la integridad con la que los usuarios logran los objetivos específicos". La eficiencia se relaciona con "los recursos empleados en relación con la exactitud y la integridad con la que los usuarios alcanzan los objetivos". La satisfacción se define como "la ausencia de incomodidad, y las actitudes positivas hacia el uso del producto".

Nos enfocamos en evaluar la usabilidad del enfoque con el fin de identificar la experiencia del usuario final, correspondiente al desarrollador/modelador de la aplicación, y a partir de ésta realizar mejoras y correcciones futuras para incrementar la facilidad de uso y la confianza relacionada al enfoque propuesto.

La experiencia de validación sigue los lineamientos de un caso de estudio, pero no corresponde a un caso de estudio en sí, debido a que no es aplicado dentro de un contexto real. Optamos por este enfoque debido a que el mismo provee un procedimiento planeado y consistente, basado en una clara cadena de evidencias, para obtener conclusiones cualitativas como cuantitativas que añaden conocimiento al ya existente. Hemos seguido las guías establecidas por Wohlin et al. [32] y Genero et al. [16] para la elaboración de la experiencia.

A continuación presentamos las actividades que forman parte del proceso llevado a cabo para la realización de la experiencia: el diseño de la experiencia, la preparación y recolección de datos, y el análisis e interpretación de los resultados obtenidos.

5.1. Diseño de la Experiencia

Para la planificación de la experiencia, se describe primeramente el objetivo principal, seguido de las preguntas de investigación, el caso y sus unidades de análisis, y posteriormente los procedimientos o métodos a llevar a cabo.

5.1.1. Objetivo Principal

El objetivo principal de la experiencia ha sido elaborado haciendo uso de la plantilla GQM (Goal-Question-Metric) [33] y se expresa a continuación:

- **Analizar** el enfoque MoWebA para el desarrollo de RIA
- **con el propósito de** evaluar su usabilidad
- **con respecto a** la eficacia, eficiencia y satisfacción
- **desde el punto de vista del** desarrollador
- **en el contexto de** estudiantes de último año de Ingeniería Info**r**mática de la Universidad Católica "Nuestra Señora de la Asunción".

Cabe destacar que el enfoque a analizar contempla las características RIA de almacenamiento

de datos en el cliente, lógica de negocios en el cliente y comunicación asíncrona entre cliente y servidor.

5.1.2. Preguntas de Investigación

Debido a que el enfoque propuesto consiste en un proceso MDD, éste puede separarse en dos etapas principales: el proceso de modelado y el proceso de generación de código. A partir de esta observación y del objetivo principal definido, se establecen las siguientes preguntas de investigación:

- **PI1**: ¿Qué eficacia, eficiencia y satisfacción presenta el proceso de modelado del enfoque propuesto?
- **PI2**: ¿Qué eficacia, eficiencia y satisfacción presenta el proceso de generación de código del enfoque propuesto?
- **PI3**: ¿Qué percepción de satisfacción presenta el enfoque MDD propuesto?

5.1.3. Participantes

El contexto de la experiencia radica en alumnos del último año de Ingeniería Informática de la Universidad Católica "Nuestra Señora de la Asunción", más precisamente, en los alumnos de la materia Complemento de Informática del 10mo semestre de la carrera. Estos alumnos poseen experiencia en procesos basados en modelos desarrollada en una materia anterior y en procesos dirigidos por modelos desarrollados en la materia recién mencionada.

5.1.4. Caso y Unidades de Análisis

El caso consiste en un proyecto de desarrollo de una RIA haciendo uso del enfoque MDD propuesto. La RIA solicitada consiste en un sistema de inscripción de estudiantes a la carrera de Ingeniería Informática. Debe estar compuesta por las siguientes dos páginas:

- La primera página debe contener un formulario de inscripción a la carrera conteniendo los campos: *Nombre, CI, Fecha de Nacimiento, Lugar de Nacimiento, Nacionalidad, Dirección, Teléfono, Email*. El valor del campo *Lugar de Nacimiento* debe ser introducido de forma textual, sin embargo, deben desplegarse algunos posibles valores previamente definidos por el investigador. El valor del campo *Nacionalidad* debe seleccionarse de una lista de valores que también fue definida por el investigador. Estos valores deben obtenerse de una variable almacenada en el navegador. El formulario debe permitir autocompletado basado en valores previamente ingresados y debe ser enviado a un servicio indicado por el investigador.
- La segunda página debe contener un botón y una tabla. El botón debe contener el texto "Información de la Carrera" y al presionar sobre éste se debe desplegar un texto (en alguna parte de la página, sin recargar la página) que describa la carrera de Ingeniería Informática, éste texto debe obtenerse de un servicio ofrecido por el investigador. La tabla debe tener las columnas *Semestre* y *Materia* y debe contener registros obtenidos de un servicio proporcionado por el investigador. La tabla debe permitir paginación (con 10 registros por página), ordenamiento y búsqueda de registros del lado del cliente.

Se solicita a los alumnos modelar los diagramas de nodos, lógico y de contenido correspondientes, generar código a partir de ellos y realizar ajustes manuales al código generado. Los ajustes manuales consisten en agregar líneas de código necesarias para desplegar el texto obtenido del servicio ofrecido por el investigador en alguna parte de la página.

Las unidades de análisis corresponden a la fase de modelado y a la fase de generación de

código. Como podemos observar en la *Figura 5.1*, aplicando la clasificación de Yin [34], este caso puede caracterizarse como único y embebido.

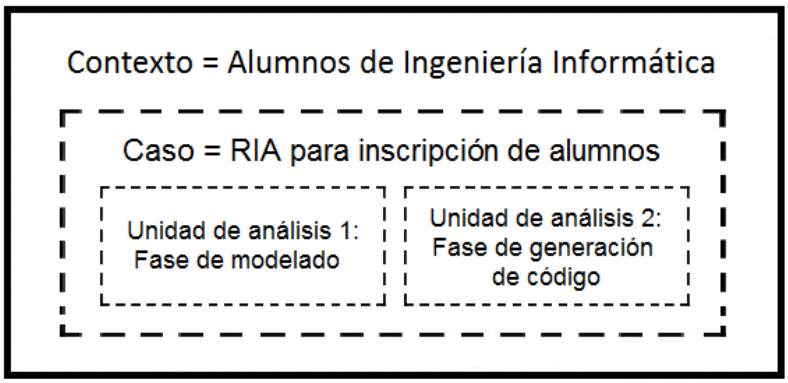


Figura 5.1: Caso de estudio único y embebido.

5.1.5. Procedimientos

Para la realización de la experiencia se desarrolló el siguiente procedimiento llevado a cabo en dos días de trabajo de 2 horas y 30 minutos cada uno, destacamos que las tareas de cada día fueron llevadas a cabo de manera secuencial:

Día 1: Explicación del enfoque

- El investigador expuso el enfoque propuesto a los alumnos.
- El investigador y los alumnos realizaron un ejemplo sencillo de modelado y generación de código de forma conjunta. El ejemplo consistió en el sistema de marcación de empleados descrito en el capítulo anterior.

Día 2: Desarrollo de la RIA

- El investigador presentó a los alumnos la aplicación a desarrollar y proveyó las indicaciones del trabajo.
- Los alumnos sin intervención del investigador elaboraron el modelo de la aplicación.
- Los alumnos con intervención del investigador realizaron correcciones al modelo elaborado. Las correcciones consistieron en agregar, modificar o eliminar elementos de modelado que han sido desatendidos por los alumnos y que el investigador ha considerado necesarios para la generación de código.
- Los alumnos completaron el cuestionario sobre el proceso de modelado del Anexo 4.
- Los alumnos sin intervención del investigador generaron código de forma automática a partir del modelo elaborado.
- Los alumnos con intervención del investigador realizaron ajustes manuales al código generado obteniendo la RIA final y funcional. Los ajustes manuales han consistido en agregar, modificar o eliminar líneas de código correspondientes a elementos desatendidos por los alumnos y no percibidos por el investigador en el modelo de la aplicación; y en agregar las líneas de código necesarias para desplegar el texto obtenido del servicio ofrecido por el investigador en alguna parte de la página.
- Los alumnos completaron el cuestionario sobre el proceso de generación de código del Anexo 4.
- Los alumnos completaron un cuestionario sobre el proceso completo del Anexo 4.

Durante ambos días, los alumnos dispusieron de un manual y tutorial del enfoque, así como también del modelo y código de la RIA de ejemplo.

5.2. Preparación y Recolección de Datos

Para la preparación y recolección de los datos se tuvieron en cuenta tres fuentes de información: la documentación del proyecto, planillas de medición de tiempos, y los cuestionarios. A continuación se detalla cada una de estas fuentes de información, indicando los datos obtenidos a partir de ellas. Posteriormente se describe como los datos obtenidos son utilizados para medir la usabilidad del enfoque.

5.2.1. Documentación del Proyecto

La documentación del proyecto incluye los modelos elaborados por los alumnos y el código generado automáticamente. De la documentación de cada alumno se obtuvieron:

- La cantidad de elementos modelados sin intervención del investigador.
- La cantidad de correcciones de modelado, correspondiente a la cantidad de elementos de modelado por agregar, modificar o eliminar para obtener un modelo completo.
- La cantidad de líneas de código generadas automáticamente.
- La cantidad de líneas de código por agregar para obtener el código completo.

A partir de las medidas anteriores, fue posible calcular las tasas de éxito para el proceso de modelado y para el proceso de generación de código:

- La tasa de éxito para el proceso de modelado, se obtuvo como resultado de la fórmula siguiente:

$$\left(1 - \frac{Cantidad\ de\ correcciones\ de\ modelado}{Cantidad\ de\ elementos\ de\ modelado}\right) * 100\%$$

- La tasa de éxito para el proceso de generación de código, se obtuvo como resultado de la fórmula siguiente:

$$\left(1 - \frac{Cantidad\ de\ líneas\ de\ código\ por\ agregar}{Cantidad\ de\ líneas\ de\ código\ generadas + Cantidad\ de\ líneas\ de\ código\ por\ agregar}\right) * 100\%$$

Posteriormente, se tomaron los resultados de todos los alumnos y se calcularon:

- La tasa de éxito promedio para el proceso de modelado
- La tasa de éxito promedio para el proceso de generación de código.

5.2.2. Planillas de Medición de Tiempos

Las planillas de medición de tiempos involucran los tiempos de desarrollo ejecutados por los alumnos, registrados por el investigador durante el desarrollo de la experiencia. Se registraron los siguientes tiempos por cada alumno presente:

- El tiempo de modelado (sin intervención del investigador) en minutos.
- El tiempo de generación automática de código en minutos.

Luego, a partir de los tiempos de todos los alumnos se calcularon:

- El tiempo promedio de modelado.
- El tiempo promedio de generación automática de código.

5.2.3. Cuestionarios

Los cuestionarios utilizados en el proceso se presentan en el Anexo 4. El primer cuestionario (sobre el proceso de modelado) y el segundo (sobre el proceso de generación de código) corresponden a cuestionarios ASQ (After Scenario Questionnaire) [35], mientras que el último cuestionario (sobre el proceso completo) corresponde a un cuestionario SUS (System Usability Scale) [36].

El ASQ consiste en un cuestionario de tres ítems, utilizado para evaluar la satisfacción del usuario tras finalizar las tareas de un determinado escenario. Los ítems abarcan tres aspectos de la satisfacción del usuario con respecto a la usabilidad del sistema: la facilidad de completitud de las tareas, el tiempo en completar las tareas y la información de soporte adecuada [37]. Como puede observarse en la *Figura 5.2*, cada ítem está acompañado de una escala de calificación de 7 puntos que van desde "Fuertemente de acuerdo" hasta "Fuertemente en desacuerdo". El puntaje general es la media aritmética de las tres puntuaciones de los ítems [38]. En [35] se ha demostrado que el ASQ es un cuestionario confiable, válido y sensible.

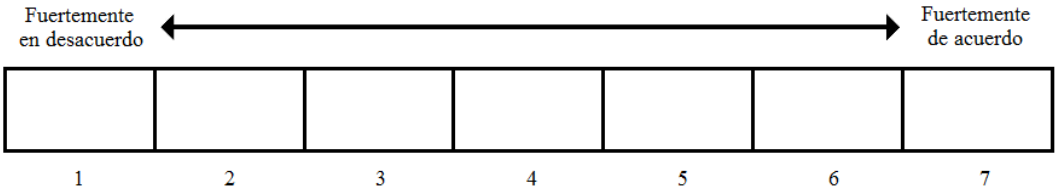


Figura 5.2: Escala de calificación para el ASQ.

El SUS consiste en un cuestionario de diez ítems utilizado para proveer una visión global de las evaluaciones subjetivas de la usabilidad. Como se observa en la *Figura 5.3*, cada ítem se acompaña de una escala de calificación de 5 puntos que van desde "Fuertemente en desacuerdo" a "Fuertemente de acuerdo". Para calcular el puntaje SUS primeramente se suma las contribuciones de puntaje de cada ítem. Para los ítems impares, la contribución de puntaje es la posición de la escala menos 1. Para los ítems pares, la contribución de puntaje es 5 menos la posición de la escala. Luego se multiplica la suma de las puntuaciones por 2.5 dando lugar al puntaje SUS final. SUS se ha convertido en el estándar de la industria, con referencias en más de 1300 artículos y publicaciones [36], y se ha demostrado que es una medida confiable y válida para la percepción de la usabilidad [39].

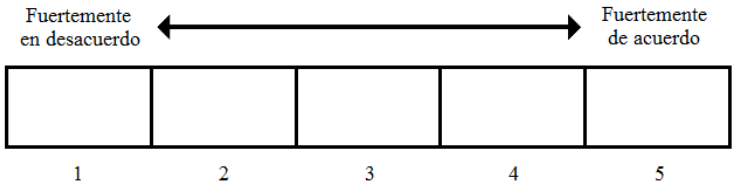


Figura 5.3: Escala de calificación para el SUS.

Distinguimos que al primer cuestionario (sobre el proceso de modelado) se le ha agregado una pregunta más (pregunta Nº 4), con el fin de identificar aquellos elementos que pudieron resultar difícil de modelar para los desarrolladores.

Por cada alumno, se analizaron las respuestas a los cuestionarios y se calcularon:

- El puntaje ASQ correspondiente al proceso de modelado.
- El puntaje ASQ correspondiente al proceso de generación de código.
- El puntaje SUS correspondiente al proceso completo.

Posteriormente, a partir de los puntajes de todos los alumnos se calcularon:

- El puntaje ASQ promedio para el proceso de modelado.
- El puntaje ASQ promedio para el proceso de generación de código.
- La media, mediana y desviación estándar de los puntajes SUS para el proceso completo.

Por último, se cuantificaron aquellos elementos difíciles de modelar para los desarrolladores.

5.2.4. Mediciones de Usabilidad a partir de los Datos Obtenidos

A continuación indicamos cuáles de los datos obtenidos en las secciones anteriores son utilizados para medir la eficacia, eficiencia y satisfacción de los procesos de modelado, de generación de código y del proceso completo:

- En el proceso de modelado:
    - Eficacia: se mide a través de la tasa de éxito promedio para el proceso de modelado.
    - Eficiencia: se mide a través del tiempo promedio de modelado.
    - Satisfacción: se mide a través de la media de los puntajes ASQ para el proceso de modelado.
- En el proceso de generación de código:
    - Eficacia: se mide a través de la tasa de éxito promedio para el proceso de generación de código.
    - Eficiencia: se mide a través del tiempo promedio de generación de código.
    - Satisfacción: se mide a través de la media de los puntajes ASQ para el proceso de generación de código.
- En el enfoque MDD en general:
    - Satisfacción: se mide a través de la media, mediana y desviación estándar de los puntajes SUS para el proceso completo.

5.2.5. Amenazas a la Validez

Se tienen en cuenta las siguientes amenazas a la validez:

Para la validez interna, que tiene que ver con el grado de confianza en una relación causa-efecto entre los factores de interés y los resultados observados, se desarrolló la experiencia sobre alumnos, los cuales, todos poseen el mismo nivel de experiencia en cuanto a un proceso MDD, evitando de esta forma participantes con conocimientos desbalanceados. Además, se les otorgó puntos extra en la materia por el desarrollo completo de la experiencia, con el fin de evitar la falta de participación en alguna de las sesiones. Por último, se supervisó a los alumnos y se les prohibió la comunicación entre pares para evitar plagios.

La validez externa, que representa el grado hasta el que los resultados alcanzados pueden generalizarse, podría verse afectada por la cantidad de alumnos involucrados en el proyecto (5 alumnos), que si bien, no corresponde a una cantidad suficiente para fines estadísticos, es suficiente para sacar primeros juicios, que luego deben ser corroborados con experimentos y/o experiencias con mayor cantidad de personas para poder generalizarlos con precisión. Además el caso desarrollado consistió en un caso pequeño, no obstante, se ha buscado un caso que contemple el desarrollo de una RIA que tenga en cuenta todas las características en cuestión.

En cuanto a la validez del constructo, que refleja hasta qué punto las medidas que se han realizado se adecuan a lo que el investigador tiene en mente y a lo que se está investigando, se han seleccionado datos que normalmente son utilizadas para medir los aspectos de calidad en cuestión. Además se han utilizado cuestionarios estándares, considerados confiables y válidos [35] [39].

En relación a la fiabilidad, que indica la dependencia de los datos y su análisis respecto de un investigador específico y la capacidad de replicar el mismo estudio y obtener los mismos resultados, se ha respetado la literalidad de los datos obtenidos, tanto de la documentación, de los tiempos medidos y de los cuestionarios, evitando la introducción de sesgos a través de la interpretación.

5.3. Análisis e Interpretación de Resultados

La experiencia realizada nos permitió responder las preguntas de investigación de la siguiente manera:

PI1: ¿Qué eficacia, eficiencia y satisfacción presenta el proceso de modelado del enfoque propuesto?

La *Figura 5.4* presenta las mediciones de usabilidad para el proceso de modelado realizado durante la experiencia. En promedio, se puede observar que los alumnos culminaron el modelo de la aplicación con una tasa de éxito del 85 %, en un tiempo de 44,2 minutos, con un puntaje ASQ de 2.67.

Escenario	Tasa de Éxito Promedio	Tiempo de Finalización Promedio	Satisfacción Promedio
Modelado	85 %	44.2 min.	2.67

Figura 5.4: Mediciones de usabilidad para el proceso de modelado.

Analizando estos resultados, podemos ver que se obtuvo una tasa de éxito satisfactoria. La mayor dificultad al momento de modelar se dio al desarrollar los diagramas de contenido, en los cuales los alumnos aplicaban de manera errónea algunos valores etiquetados y olvidaban de incluir el botón de envío en los formularios. Destacamos que no se registraron problemas en el modelado del diagrama de nodos y del diagrama lógico.

En cuanto al tiempo de modelado, no podríamos afirmar que corresponde a un tiempo favorable, debido a que para ello, deberíamos de compararlo contra otro enfoque o proceso que implemente la misma aplicación.

Con respecto al puntaje ASQ, apuntamos que valores más cercanos al 1, en una escala de 7 puntos, indican mayor nivel de satisfacción, por lo que el puntaje obtenido refleja un nivel de satisfacción bueno por parte de los alumnos con el proceso de modelado propuesto.

En la *Figura 5.5* presentamos la frecuencia con la que un elemento fue mencionado como difícil de modelar por los alumnos. Notamos que el botón no fue indicado como un elemento difícil de modelar por ningún alumno, mientras que el formulario, la tabla, las variables locales y los servicios invocados fueron difíciles de modelar por un alumno respectivamente. La mayoría de los alumnos indicaron que ningún elemento fue difícil de modelar.

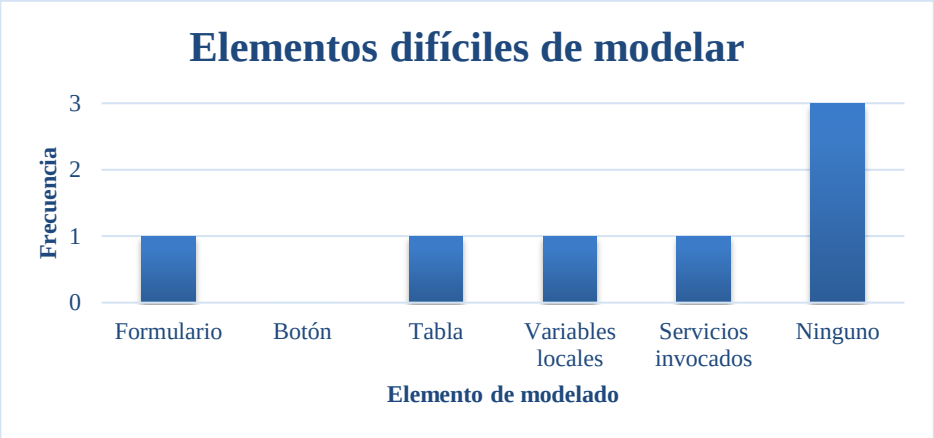


Figura 5.5: Elementos difíciles de modelar durante la experiencia.

PI2: ¿Qué eficacia, eficiencia y satisfacción presenta el proceso de generación de código del enfoque propuesto?

En la *Figura 5.6* presentamos las mediciones de usabilidad para el proceso de generación de código de la experiencia. En promedio, la generación automática de código fue culminada con una tasa de éxito del 85 %, en un tiempo de 3.6 minutos, y un puntaje ASQ de 2.20.

Escenario	Tasa de Éxito Promedio	Tiempo de Finalización Promedio	Satisfacción Promedio
Generación de Código	85 %	3.6 min.	2.20

Figura 5.6: Mediciones de usabilidad para el proceso de generación de código.

A partir de estos resultados, podemos notar que la tasa de éxito resulta igual de satisfactoria que la tasa de éxito obtenida para el proceso de modelado. Sin embargo, creemos que la tasa de éxito para el proceso de generación de código pudo haber obtenido un mejor porcentaje, ya que ésta se vio directamente afectada por el modelo de entrada utilizado. Si bien se generaron líneas de código a partir de todos los modelos, éstas no fueron completas, debido a las imperfecciones de los modelos desarrollados por los alumnos, utilizados como entrada para el generador de código. Destacamos que se realizaron correcciones in situ de los modelos elaborados, pero no se llegaron a detectar todas las correcciones necesarias por falta de tiempo.

En relación al tiempo de generación de código, podemos observar que se obtuvo un tiempo bastante reducido. Este tiempo podría generalizarse para la generación de todo tipo de aplicaciones ya que requiere de pasos estrictamente mecánicos y predefinidos, sin gran variación en la intervención del desarrollador.

En cuanto al puntaje ASQ obtenido, podemos apreciar que es superior al del proceso de modelado, indicando un mayor nivel de satisfacción para el proceso de generación de código.

PI3: ¿Qué percepción de satisfacción presenta el enfoque MDD propuesto?

El cuestionarios SUS sobre el enfoque completo fue utilizado para evaluar la usabilidad general del enfoque propuesto. Sauro y Lewis [40] han encontrado tras analizar 446 estudios y más de 5000 respuestas individuales del SUS, que un puntaje mayor a 68 es considerado por encima del promedio, mientras que un puntaje menor es considerado por debajo del promedio. Sin embargo, la mejor forma de interpretar los resultados implica normalizar los puntajes para producir un rango percentil. Este rango percentil indica que tan usable es la aplicación analizada relativa a los otros productos presentes en la base de datos total analizada por Sauro.

En la *Figura 5.7* presentamos los estadísticos descriptivos calculados para los puntajes SUS obtenidos. Los puntajes tuvieron una media de 50.5, una mediana de 50, con una desviación estándar de 4.81.

Media	50.5
Mediana	50
Desviación estándar	4.81

Figura 5.7: Estadísticos descriptivos de los puntajes SUS obtenidos.

Realizando una comparación de la media de 50.5 con el puntaje de referencia de 68, el puntaje SUS obtenido se encuentra por debajo del promedio de referencia. Convirtiendo el puntaje de 50.5 a través de la normalización nos da un rango percentil de 13 %, indicando que el enfoque

propuesto es más usable que el 13 % de los productos de la base de datos de Sauro.

Si bien se obtuvieron buenos resultados de usabilidad para el proceso de modelado y para el proceso de generación de código por separado, los resultados para el proceso completo no fueron muy alentadores. Se obtuvieron bajas puntuaciones en los ítems 1, 2, 4 y 10 del cuestionarios SUS del Anexo 4, correspondientes a la utilización frecuente, complejidad, necesidad de soporte de una persona técnica y necesidad de aprender muchas cosas del enfoque. La utilización frecuente del enfoque por parte de los alumnos podría verse afectada por el habitual trabajo de los alumnos con métodos ágiles y su preferencia hacia éstos. La complejidad podría parecer alta para ellos debido a la necesidad de incluir dos etapas diferentes (modelado y generación de código) en lugar de una implementación directa sin una fase formal de diseño. La necesidad de soporte y la de aprender muchas cosas estarían directamente relacionadas con el escaso tiempo dedicado para el aprendizaje del enfoque y la limitada experiencia en procesos MDD.

5.4. Resumen del Capítulo

En este capítulo presentamos la experiencia de validación realizada haciendo uso del enfoque propuesto. La experiencia fue desarrollada con los alumnos de último año de Ingeniería Informática de la Universidad Católica “Nuestra Señora de la Asunción”, a los cuales se les solicitó desarrollar una RIA funcional.

Buscando evaluar la usabilidad del enfoque, seguimos los pasos de un caso de estudio, elaborando el diseño de la experiencia, definiendo métodos para la preparación y recolección de datos y analizando e interpretando los resultados obtenidos.

Tras evaluar los procesos de modelado y de generación de código de la RIA resultante, obtuvimos resultados positivos en relación a las tasas de éxito de realización de estas tareas, los tiempos de desarrollo, y la satisfacción de los usuarios en relación a la propuesta.

6. Conclusiones y Trabajos Futuros

En este proyecto de fin de carrera hemos presentado una solución basada en MDD para el desarrollo de RIA incorporando las características de distribución de datos, distribución de lógica de negocios y comunicación asíncrona entre cliente y servidor.

Puntualmente se han obtenido las siguientes contribuciones:

1. Un estudio detallado acerca de las RIA, MDD y del enfoque MoWebA.
2. Una revisión a través de un SMS de las propuestas metodológicas para el desarrollo de RIA que abarcan las características de distribución de datos, distribución de lógica de negocios y comunicación asíncrona entre cliente y servidor.
3. Una extensión de MoWebA para aplicaciones RIA abarcando las características de distribución de datos, distribución de lógica de negocios y comunicación asíncrona entre cliente y servidor.
4. Una validación de la propuesta a través de una experiencia de desarrollo de una RIA con alumnos universitarios.

Hemos presentado a las RIA, describimos sus características principales y las diferentes tecnologías para su desarrollo. Entre las características de las RIA, dimos énfasis en las características de distribución de datos, distribución de lógica de negocios y comunicación asíncrona entre cliente y servidor, debido a que representan a las operaciones ejecutadas por debajo para la correcta presentación y despliegue de la interfaz de usuario. Además, hemos expuesto los fundamentos de MDD, exhibiendo las ventajas de llevar a cabo un proceso de desarrollo de RIA siguiendo un enfoque de este tipo. Dentro de MDD, vimos que MDA surge como la propuesta de la OMG para seguir un proceso de desarrollo estándar. Presentamos el enfoque MoWebA, una propuesta MDD que sigue el estándar MDA para el desarrollo de aplicaciones web. Hemos descripto el proceso de modelado y de transformación a código de MoWebA, destacando la introducción de un ASM para enriquecer a los modelos con información acerca de una arquitectura específica.

A través de un SMS realizamos una búsqueda y análisis de los enfoques MDD para el desarrollo de RIA que tienen en cuenta las características en cuestión. Encontramos 13 estudios de los cuales identificamos 8 enfoques finales. Entre estos enfoques pudimos notar la falta de inclusión de todas las características RIA en cuestión, la falta de adopción de MDA, la propuesta de un lenguaje de modelado desde cero o como una combinación de varios lenguajes generando un aumento de la curva de aprendizaje, herramientas limitadas, y la falta de consideración de tecnologías basadas en scripting que corresponden al tipo de tecnología mayormente utilizada para el desarrollo de RIA en la comunidad de desarrolladores.

Para superar las limitantes anteriores propusimos un enfoque MDD para el desarrollo de RIA que contempla las características en cuestión, adopta el estándar MDA, extiende un lenguaje ya existente, utiliza herramientas universales, y genera una implementación con tecnologías basadas en scripting. La propuesta parte del enfoque MoWebA, extendiendo los metamodelos y perfiles correspondientes a los diagramas lógico, y de contenido y proponiendo un metamodelo y perfil para el ASM correspondiente a las funcionalidades RIA con las características en cuestión. Además, se desarrollaron reglas de transformación, que a partir de instancias de estos metamodelos y perfiles, generan código correspondiente a la RIA final y funcional. A nivel de modelado, se introdujeron los elementos *ClientValueObject* y *ClientStaticObject* que representan variables almacenadas del lado del cliente; los elementos *RichTable*, *RichForm*, y

RichTextInput que representan tablas, formularios y entradas de texto como elementos de interfaz de usuario con operaciones de lógica de negocios (paginación, ordenamiento, búsqueda, autocompletado) ejecutadas del lado del cliente; y el elemento *AsynchronousCall* que representa la invocación asíncrona de un servicio desde un elemento de presentación. A nivel de implementación, se genera código HTML5 para elementos de interfaz de usuario y código Javascript para el comportamiento de dichos elementos. Las funcionalidades de HTML5 LocalStorage y SessionStorage son aprovechadas para el almacenamiento de datos del lado del cliente. JQuery con sus *plugins* Datatables y jQuery UI son utilizados para la lógica de negocios del lado del cliente. JQuery también es utilizado para implementar las invocaciones asíncronas.

Para comprobar si la propuesta es válida y útil realizamos una experiencia con los alumnos de último año de Ingeniería Informática de la Universidad Católica "Nuestra Señora de la Asunción", a los cuales se les solicitó desarrollar una RIA funcional haciendo uso del enfoque propuesto. Buscando evaluar la usabilidad del enfoque, seguimos los pasos de un caso de estudio, dando como resultado tasas de éxito ventajosas y buenos tiempos de desarrollo y niveles de satisfacción por parte de los alumnos para los procesos de modelado y de generación de código.

Concluimos con las siguientes actividades que podrían realizarse como trabajos futuros:

- Modelar operaciones CRUD (Create, Read, Update, Delete) desde la capa de presentación sobre los datos almacenados del lado del cliente.
- Abarcar más elementos de interfaz de usuario con lógica de negocios del lado del cliente, tales como objetos *drag&drop*, *slideshows*, *tooltips*, *ratings*, etc.
- Modelar acciones a ejecutar a partir de las respuestas de las llamadas asíncronas.
- Integrar el enfoque con la propuesta de López et al [7].
- Implementar reglas de transformación para otras plataformas destino.
- Realizar validaciones más estrictas de la propuesta, como experimentos formales o casos de uso en un contexto industrial o comercial.
- Comparar las medidas de usabilidad obtenidas utilizando el enfoque con las de otro enfoque o proceso desarrollando la misma aplicación.

7. Anexos y Apéndices

Anexo 1. Estudios seleccionados como resultado del Mapeo Sistemático de la Literatura

Fuente	Año	Título
IEEExplore	2010	S. Meliá, J. Gómez, S. Pérez, and O. Díaz, "Architectural and technological variability in rich internet applications," *IEEE Internet Computing*, vol. 14, no. 3, pp. 24–32, 2010.
ACM Digital Library	2007	M. Linaje, J. C. Preciado, and F. Sánchez-Figueroa, "A method for model based design of rich internet application interactive user interfaces," in *Web Engineering*. Springer, 2007, pp. 226–241.
	2009	F. Valverde and O. Pastor, "Facing the technological challenges of web 2.0: A ria model-driven engineering approach," in *International Conference on Web Information Systems Engineering*. Springer, 2009, pp. 131–144.
	2011	G. Toffetti, S. Comai, J. C. Preciado, and M. Linaje, "State-of-the-art and trends in the systematic development of rich internet applications," *Journal of Web Engineering*, vol. 10, no. 1, pp. 070–086, 2011.
	2013	J. L. H. Agustin and P. C. Del Barco, "A model-driven approach to develop high performance web applications," *Journal of Systems and Software*, vol. 86, no. 12, pp. 3013–3023, 2013.
SpringerLink	2007	G. T. Carughi, S. Comai, A. Bozzon, and P. Fraternali, "Modeling distributed events in data-intensive rich internet applications," in *Web Information Systems Engineering–WISE 2007*. Springer, 2007, pp. 593–602.
Google Scholar	2006	A. Bozzon, S. Comai, P. Fraternali, and G. T. Carughi, "Conceptual modeling and code generation for rich internet applications," in *Proceedings of the 6th international conference on Web engineering*. ACM, 2006, pp. 353–360.
	2007	J. C. Preciado, M. Linaje, S. Comai, and F. Sanchez-Figueroa, "Designing rich internet applications with web engineering methodologies," in *Web Site Evolution, 2007. WSE 2007. 9th IEEE International Workshop on*. IEEE, 2007, pp. 23–30.
	2008	A. Bozzon, S. Comai, P. Fraternali, and M. Tisi, "Current research on the design of web 2.0 applications based on model-driven approaches," in ICWE 2008 Workshops. Citeseer, p. 25.
	2008	M. Brambilla, J. C. Preciado, M. Linaje, and F. Sanchez-Figueroa, "Business process-based conceptual design of rich internet applications," in *Web Engineering, 2008. ICWE'08. Eighth International Conference on*. IEEE, 2008, pp. 155–161.
	2009	M. Brambilla, P. Fraternali, and E. Molteni, "A tool for model-driven design of rich internet applications based on ajax," Handbook of Research on Web, vol. 2, no. 3.0, pp. 96–118, 2009.
	2009	S. Meliá, J. J. M. Domene, Á. Pérez, and J. Gómez, "Ooh4ria tool: Una herramienta basada en el desarrollo dirigido por modelos para las rias." in JISBD, 2009, pp. 219–222.
	2009	M. Busch and N. Koch, "Rich internet applications. state-of-the-art," Ludwig-Maximilians-Universität München, München, Germany, techreport 0902, 2009.

Anexo 2. Reglas de Transformación

Acceleo Module File

```
[comment encoding = UTF-8 /]
[module generate('http://www.eclipse.org/uml2/5.0.0/UML')]
[import riaCodeGenerator::services::services]

[template public generateElement(c : Class)]
[comment @main/]
[if (c.hasStereotype('clientStaticObject'))]
	[clientStaticObject(c)/]
[/if]
[if(c.hasStereotype('presentationPage'))]
	[presentationPage(c)/]
[/if]
[/template]

[template public clientStaticObject(c : Class)]
[let	persistence	:	String	=	c.getValue(c.getAppliedStereotype('RIA
Profile::clientStaticObject'), 'persistence').oclAsType(EnumerationLiteral).name]
[file ('ria/'+'variables.js', true, 'UTF-8')]
[if(persistence = 'permanent')]
	[if (c.attribute->size() = 1)]
		[let p : Property = c.allAttributes()->first()]
		[if(p.hasStereotype('value'))]
			[let	title:	OclAny	=	p.getValue(p.getAppliedStereotype('Logic
Profile::value'), 'title')]
			[let	checked:	OclAny	=	p.getValue(p.getAppliedStereotype('Logic
Profile::value'), 'checked')]
var	[c.name/]	=	{title:	"[title/]",	value:"[p.name/]",
checked:[if(checked=true)]true[else]false[/if]};
localStorage['['/]"[c.name/]"[']'/] = JSON.stringify([c.name/]);
[c.name/] = JSON.parse(localStorage['['/]"[c.name/]"[']'/]);
			[/let][/let]
		[/if]
		[/let]
	[else]
var [c.name/] = ['['/][']'/];
		[for (p: Property | c.allAttributes())]
			[if(p.hasStereotype('value'))]
				[let	title:	OclAny	=	p.getValue(p.getAppliedStereotype('Logic
Profile::value'), 'title')]
				[let	checked:	OclAny	=	p.getValue(p.getAppliedStereotype('Logic
Profile::value'), 'checked')]
[c.name/]['['/][i-1/][']'/]	=	{title:	"[title/]",	value:"[p.name/]",
checked:[if(checked=true)]true[else]false[/if]};
				[/let][/let]
			[/if]
		[/for]
localStorage['['/]"[c.name/]"[']'/] = JSON.stringify([c.name/]);
[c.name/] = JSON.parse(localStorage['['/]"[c.name/]"[']'/]);
	[/if]
[else]
	[if (c.attribute->size() = 1)]
		[let p : Property = c.allAttributes()->first()]
		[if(p.hasStereotype('value'))]
			[let	title:	OclAny	=	p.getValue(p.getAppliedStereotype('Logic
Profile::value'), 'title')]
			[let	checked:	OclAny	=	p.getValue(p.getAppliedStereotype('Logic
Profile::value'), 'checked')]
var	[c.name/]	=	{title:	"[title/]",	value:"[p.name/]",
checked:[if(checked=true)]true[else]false[/if]};
sessionStorage['['/]"[c.name/]"[']'/] = JSON.stringify([c.name/]);
[c.name/] = JSON.parse(sessionStorage['['/]"[c.name/]"[']'/]);
			[/let][/let]
		[/if]
		[/let]
	[else]
var [c.name/] = ['['/][']'/];
		[for (p: Property | c.allAttributes())]
			[if(p.hasStereotype('value'))]
				[let	title:	OclAny	=	p.getValue(p.getAppliedStereotype('Logic
```

```
Profile::value'), 'title')]
                                [let checked: OclAny = p.getValue(p.getAppliedStereotype('Logic
Profile::value'), 'checked')]
[c.name/]['['/][i-1/][']'/]        =        {title:        "[title/]",        value:"[p.name/]",
checked:[if(checked=true)]true[else]false[/if]};
                                [/let][/let]
                        [/if]
                [/for]
sessionStorage['['/]"[c.name/]"[']'/] = JSON.stringify([c.name/]);
[c.name/] = JSON.parse(sessionStorage['['/]"[c.name/]"[']'/]);
        [/if]
[/if]
[/file]
[/let]
[/template]

[template public presentationPage(c : Class)]
[setPresentationPage(c)/]
[getAllAsociatedClasses(c)/]
[resetHasAsynchronousCall()/][resetHasRichTable()/][resetHasClientStaticObject()/][resetHasAutoc
omplete()/][clearCalledUIElements()/]
[for(cUIE: Class | getVisitedClasses().oclAsType(Class))]
        [if(cUIE.hasStereotype('richTable'))]
                [richTable(cUIE,c)/]
                [setHasRichTable()/]
        [/if]
        [for(p: Property | cUIE.getAllAttributes())]
                [for (d: Dependency | p.getRelationships().oclAsType(Dependency))]
                        [let client : Class = d.client->asSequence()->at(1).oclAsType(Class)]
                        [if(client.hasStereotype('asynchronousCall'))]
                                [asynchronousCall(client)/]
                                [setHasAsynchronousCall()/]
                        [/if]
                        [/let]
                        [let     supplier     :     Class     =     d.supplier->asSequence()-
>at(1).oclAsType(Class)]
                        [if(supplier.hasStereotype('clientStaticObject'))]
                                [setHasClientStaticObject()/]
                        [/if]
                        [/let]
                [/for]
                [if(p.hasStereotype('richTextInput'))]
                        [let  autocomplete:  OclAny  =  p.getValue(p.getAppliedStereotype('RIA
Profile::richTextInput'), 'autocomplete')]
                        [if(autocomplete = true)]
                                [setHasAutocomplete()/]
                        [/if]
                        [/let]
                [/if]
        [/for]
[/for]
[file ('ria/'+c.name+'/'+c.name+'.html', false, 'UTF-8')]
<!DOCTYPE html>
<html>
        <head>
        [if(getHasAutocomplete() = true)]
                <script src="//code.jquery.com/jquery-1.12.0.min.js"></script>
                <link   rel="stylesheet"   href="//code.jquery.com/ui/1.12.0/themes/base/jquery-
ui.css">
                <script src="https://code.jquery.com/ui/1.12.0/jquery-ui.js"></script>
        [/if]
        [if(getHasAsynchronousCall() = true)]
                <script src="//code.jquery.com/jquery-1.12.0.min.js"></script>
        [/if]
        [if(getHasRichTable() = true)]
                <script src="//code.jquery.com/jquery-1.12.0.min.js"></script>
                <link                    rel="stylesheet"                    type="text/css"
href="//cdn.datatables.net/1.10.11/css/jquery.dataTables.min.css">
                <script                  type="text/javascript"                  charset="utf8"
src="//cdn.datatables.net/1.10.11/js/jquery.dataTables.js"></script>
        [/if]
        [if(getHasClientStaticObject() = true)]
                <script src="/ria/variables.js"></script>
        [/if]
```

```
	[if(getHasAutocomplete() = true or getHasAsynchronousCall() = true or getHasRichTable()
= true or getHasClientStaticObject() = true)]
		<script src="[c.name/].js"></script>
	[/if]
	</head>
	<body>
	[for(cl: Class | getVisitedClasses().oclAsType(Class))]
		[if(cl.hasStereotype('table'))]
		<table id="[cl.name/]">
			<tr>
			[for(h: Property | cl.allAttributes())]
				[if(h.hasStereotype('htmlText'))]
					[let header: OclAny =
h.getValue(h.getAppliedStereotype('Content Profile::htmlText'), 'html')]
					<th>[header/]</th>
					[/let]
				[/if]
			[/for]
			</tr>
		</table>
		[/if]
		[if(cl.hasStereotype('richTable'))]
		<table id="[cl.name/]">
			<thead>
				<tr>
				[for(h: Property | cl.allAttributes())]
					[if(h.hasStereotype('htmlText'))]
						[let header: OclAny =
h.getValue(h.getAppliedStereotype('Content Profile::htmlText'), 'html')]
					<th>[header/]</th>
						[/let]
					[/if]
				[/for]
				</tr>
			</thead>
			<tbody>

			</tbody>
		</table>
		[/if]
		[if(cl.hasStereotype('richForm'))]
			[for(a : Property | cl.allAttributes())]
				[if(a.hasStereotype('submitButton'))]
					[let historyAutocomplete: OclAny =
cl.getValue(cl.getAppliedStereotype('RIA Profile::richForm'), 'historyAutocomplete')]
					[let requestType : String =
cl.getValue(cl.getAppliedStereotype('RIA
Profile::richForm'),'requestType').oclAsType(EnumerationLiteral).name]
					[let service: OclAny =
a.getValue(a.getAppliedStereotype('Content Profile::submitButton'), 'service')]
		<form id="[cl.name/]"[if(service<>null)]
action="/ria/servicios/[service.oclAsType(State).name/].php"[service(service.oclAsType(State))/]
[/if][if(requestType<>'')]  method="[if(requestType = 'retrieve')]GET[else]POST[/if]"[/if]
autocomplete="[if (historyAutocomplete = false)]off[else]on[/if]">
			[createUIElements(cl)/]
		</form>
					[/let][/let][/let]
				[/if]
			[/for]
		[/if]
		[if(cl.hasStereotype('form'))]
			[for(a : Property | cl.allAttributes())]
				[if(a.hasStereotype('submitButton'))]
					[let requestType : String =
cl.getValue(cl.getAppliedStereotype('Content
Profile::form'),'requestType').oclAsType(EnumerationLiteral).name]
					[let service: OclAny =
a.getValue(a.getAppliedStereotype('Content Profile::submitButton'), 'service')]
		<form id="[cl.name/]"[if(service<>null)]
action="/ria/servicios/[service.oclAsType(State).name/].php"[service(service.oclAsType(State))/]
[/if][if(requestType<>'')] method="[if(requestType = 'retrieve')]GET[else]POST[/if]"[/if]>
			[createUIElements(cl)/]
		</form>
					[/let][/let]
```

```
                        [/if]
                [/for]
        [/if]
        [if(cl.hasStereotype('compositeUIElement'))]
        <div id="[cl.name/]">
                [createUIElements(cl)/]
        </div>
        [/if]
    [/for]
    </body>
</html>
[/file]
[/template]

[template public asynchronousCall(c : Class)]
[let p : Class = getPresentationPage()]
[let requestUrl: OclAny = c.getValue(c.getAppliedStereotype('RIA Profile::asynchronousCall'),
'requestUrl')]
[let requestType : String = c.getValue(c.getAppliedStereotype('RIA
Profile::asynchronousCall'),'requestType').oclAsType(EnumerationLiteral).name]
[let responseType : String = c.getValue(c.getAppliedStereotype('RIA
Profile::asynchronousCall'),'responseType').oclAsType(EnumerationLiteral).name]
[let event : String = c.getValue(c.getAppliedStereotype('RIA
Profile::asynchronousCall'),'eventType').oclAsType(EnumerationLiteral).name]
[let uIElement : Property = c.clientDependency.supplier->asSequence()-
>first().oclAsType(Property)]
[addToCalledUIElements(uIElement)/]
[file ('ria/'+p.name+'/'+p.name+'.js', true, 'UTF-8')]
window.addEventListener('load', function(){
    $("#[uIElement.name/]").[event/](function
[c.name/]([if(uIElement.hasStereotype('anchor') or
uIElement.hasStereotype('submitButton'))]event[/if]){
        [if(uIElement.hasStereotype('anchor') or
uIElement.hasStereotype('submitButton'))]
        event.preventDefault();
        [/if]
        $.ajax({
        [if(requestUrl<>null)]
            url: "[requestUrl/]",
        [else]
            [let service: State = c.getValue(c.getAppliedStereotype('RIA
Profile::asynchronousCall'), 'service').oclAsType(State)]
            [if(service <> null)]
            "url": "/ria/servicios/[service.name/].php",[service(service)/]
            [/if]
            [/let]
        [/if]
        [let n : Integer = c.attribute->size()]
        [if(uIElement.hasStereotype('submitButton'))]
            [let form: Class = uIElement.ancestors()->at(1).oclAsType(Class)]
            data : $('#[form.name/]').serialize()[if(n>0)] + "[for (p: Property |
c.allAttributes())][if(p.hasStereotype('callParameter'))][let value : OclAny =
p.getValue(p.getAppliedStereotype('RIA Profile::callParameter'),
'value')]&[p.name/]=[value/][/let][/if][/for]"[/if],
            [/let]
        [else]
            [if(n>0)]
            data: {
            [for (p: Property | c.allAttributes())]
                [if(p.hasStereotype('callParameter'))]
                    [let value : OclAny =
p.getValue(p.getAppliedStereotype('RIA Profile::callParameter'), 'value')]
                [p.name/]: "[value/]"[if(i<>n)],[/if]
                    [/let]
                [/if]
            [/for]
            },
            [/if]
        [/if]
        [/let]
        [if(requestType<>'')]
            [if(requestType = 'retrieve')]
            type: "GET",
            [/if]
```

```
                        [if(requestType = 'insert/update')]
                        type: "POST",
                        [/if]
                [/if]
                [if(responseType<>'')]
                        dataType: "[responseType/]",
                [/if]
                        success: function(result){

                        }
                });
        });
});
[/file]
[/let][/let][/let][/let][/let][/let]
[/template]

[template public richTable(c : Class, p: Class)]
[let paging: OclAny = c.getValue(c.getAppliedStereotype('RIA Profile::richTable'), 'paging')]
[let pageLength: OclAny = c.getValue(c.getAppliedStereotype('RIA Profile::richTable'),
'pageLength')]
[let ordering: OclAny = c.getValue(c.getAppliedStereotype('RIA Profile::richTable'),
'ordering')]
[let searching: OclAny = c.getValue(c.getAppliedStereotype('RIA Profile::richTable'),
'searching')]
[let tableInformation: OclAny = c.getValue(c.getAppliedStereotype('RIA Profile::richTable'),
'tableInformation')]
[let processingLocation : String = c.getValue(c.getAppliedStereotype('RIA
Profile::richTable'),'processingLocation').oclAsType(EnumerationLiteral).name]
[file ('ria/'+p.name+'/'+p.name+'.js', true, 'UTF-8')]
$(document).ready(function() {
    $('#[c.name/]').DataTable({
        "paging": [paging/],
                [if(paging = true)]
                        [if(pageLength<>null)]
                "pageLength": [pageLength/],
                        [/if]
        [/if]
                "ordering": [ordering/],
                "searching": [searching/],
                "info": [tableInformation/],
                [if(processingLocation = 'server')]
                "serverSide": true,
                [else]
                "serverSide": false,
                [/if]
                [let service: State = c.getValue(c.getAppliedStereotype('RIA
Profile::richTable'), 'service').oclAsType(State)]
                [if(service <> null)]
                "ajax": "/ria/servicios/[service.name/].php"[service(service)/]
                [/if]
                [/let]
    });
});
[/file]
[/let][/let][/let][/let][/let][/let]
[/template]

[template public createUIElements(c: Class) post(trim())]
[for(a: Property | c.allAttributes())]
        [if(a.hasStereotype('text'))]
                [let localText: OclAny = a.getValue(a.getAppliedStereotype('Content
Profile::text'), 'localText')]
[localText/]<br>
                [/let]
        [/if]
        [if(a.hasStereotype('htmlText'))]
                [let htmlText: OclAny = a.getValue(a.getAppliedStereotype('Content
Profile::htmlText'), 'html')]
[htmlText/]<br>
                [/let]
        [/if]
        [if(a.hasStereotype('multimedia'))]
                [let multimediaType : String = a.getValue(a.getAppliedStereotype('Content
```

```
Profile::multimedia'),'type').oclAsType(EnumerationLiteral).name]
                [if(multimediaType = 'image')]
                        [let    path:    OclAny    =    a.getValue(a.getAppliedStereotype('Content
Profile::multimedia'), 'path')]
                        [let    title:    OclAny    =    a.getValue(a.getAppliedStereotype('Content
Profile::multimedia'), 'title')]
                        [if(path<>null)]
<img    [if(isInCalledUIElements(a.name))]id="[a.name/]"    [/if][if(title<>null)]alt="[title/]"
[/if]src="[path/]"><br>
                        [else]
                                [let  url:  OclAny  =  a.getValue(a.getAppliedStereotype('Content
Profile::multimedia'), 'url')]
<img    [if(isInCalledUIElements(a.name))]id="[a.name/]"    [/if][if(title<>null)]alt="[title/]"
[/if]src="[url/]"><br>
                                [/let]
                        [/if]
                        [/let][/let]
                [/if]
                [/let]
        [/if]
        [if(a.hasStereotype('textInput'))]
[createTextInput(a)/]
        [/if]
        [if(a.hasStereotype('list'))]
[createList(a)/]
        [/if]
        [if(a.hasStereotype('externalLink'))]
                [let     title:     OclAny     =     a.getValue(a.getAppliedStereotype('Content
Profile::externalLink'), 'title')]
                [let     url:     OclAny     =     a.getValue(a.getAppliedStereotype('Content
Profile::externalLink'), 'url')]
                [if(title<>null)]
<a [if(isInCalledUIElements(a.name))]id="[a.name/]" [/if]href="[url/]">[title/]</a><br>
                [/if]
                [/let]
                [/let]
        [/if]
        [if(a.hasStereotype('anchor'))]
                [let     title:     OclAny     =     a.getValue(a.getAppliedStereotype('Content
Profile::anchor'), 'title')]
                [if(title<>null)]
                        [let virtualState: OclAny = a.getValue(a.getAppliedStereotype('Content
Profile::anchor'), 'virtualState')]
                        [let   service:   OclAny   =   a.getValue(a.getAppliedStereotype('Content
Profile::anchor'), 'service')]
                        [if(virtualState <> null)]
<a                                              [if(isInCalledUIElements(a.name))]id="[a.name/]"
[/if]href="/ria/[virtualState.oclAsType(State).name/]/[virtualState.oclAsType(State).name/].html
">[title/]</a><br>
                        [elseif(service <> null)]
<a                                              [if(isInCalledUIElements(a.name))]id="[a.name/]"
[/if]href="/ria/servicios/[service.oclAsType(State).name/].php">[title/]</a><br>[service(service
.oclAsType(State))/]
                        [else]
<a [if(isInCalledUIElements(a.name))]id="[a.name/]" [/if]href="">[title/]</a><br>
                        [/if]
                        [/let][/let]
                [/if]
                [/let]
        [/if]
        [if(a.hasStereotype('button'))]
                [let     title:     OclAny     =     a.getValue(a.getAppliedStereotype('Content
Profile::button'), 'title')]
                [if(title<>null)]
<button                                         [if(isInCalledUIElements(a.name))]id="[a.name/]"
[/if]type="button">[title/]</button><br>
                [/if]
                [/let]
        [/if]
        [if(a.hasStereotype('submitButton'))]
                [let     title:     OclAny     =     a.getValue(a.getAppliedStereotype('Content
Profile::submitButton'), 'title')]
                [if(title<>null)]
<input [if(isInCalledUIElements(a.name))]id="[a.name/]" [/if]type="submit" value="[title/]"><br>
```

```
            [/if]
            [/let]
        [/if]
        [if(a.hasStereotype('richTextInput'))]
[createRichTextInput(a)/]
        [/if]
[/for]
[/template]

[template public createTextInput(a: Property) post(trim())]
[let mandatory: OclAny = a.getValue(a.getAppliedStereotype('Content Profile::textInput'),
'mandatory')]
[let minLength: OclAny = a.getValue(a.getAppliedStereotype('Content Profile::textInput'),
'minLength')]
[let maxLength: OclAny = a.getValue(a.getAppliedStereotype('Content Profile::textInput'),
'maxLength')]
[let onlyDigits: OclAny = a.getValue(a.getAppliedStereotype('Content Profile::textInput'),
'onlyDigits')]
[let email: OclAny = a.getValue(a.getAppliedStereotype('Content Profile::textInput'), 'email')]
[let password: OclAny = a.getValue(a.getAppliedStereotype('Content Profile::textInput'),
'password')]
[let date : String = a.getValue(a.getAppliedStereotype('Content
Profile::textInput'),'date').oclAsType(EnumerationLiteral).name]
[let title: OclAny = a.getValue(a.getAppliedStereotype('Content Profile::textInput'), 'title')]
[if(title<>null)][title/]<br>[/if]
[if(onlyDigits = true)]
<input [if(isInCalledUIElements(a.name))]id="[a.name/]" [/if]type="number"
name="[a.name/]"[if(minLength<>0)]
pattern=".{[minLength/],[if(maxLength<>0)][maxLength/][/if]}"[/if][if(maxLength<>0 and minLength
= 0)] maxlength="[maxLength/]"[/if][if(mandatory=true or minLength<>0)] required[/if]><br>
[elseif(email = true)]
<input [if(isInCalledUIElements(a.name))]id="[a.name/]" [/if]type="email"
name="[a.name/]"[if(minLength<>0)]
pattern=".{[minLength/],[if(maxLength<>0)][maxLength/][/if]}"[/if][if(maxLength<>0 and minLength
= 0)] maxlength="[maxLength/]"[/if][if(mandatory=true or minLength<>0)] required[/if]><br>
[elseif(password = true)]
<input [if(isInCalledUIElements(a.name))]id="[a.name/]" [/if]type="password"
name="[a.name/]"[if(minLength<>0)]
pattern=".{[minLength/],[if(maxLength<>0)][maxLength/][/if]}"[/if][if(maxLength<>0 and minLength
= 0)] maxlength="[maxLength/]"[/if][if(mandatory=true or minLength<>0)] required[/if]><br>
[elseif(date = 'default - dd/mm/yyyy')]
<input [if(isInCalledUIElements(a.name))]id="[a.name/]" [/if]type="date"
name="[a.name/]"[if(minLength<>0)]
pattern=".{[minLength/],[if(maxLength<>0)][maxLength/][/if]}"[/if][if(maxLength<>0 and minLength
= 0)] maxlength="[maxLength/]"[/if][if(mandatory=true or minLength<>0)] required[/if]><br>
[else]
<input [if(isInCalledUIElements(a.name))]id="[a.name/]" [/if]type="text"
name="[a.name/]"[if(minLength<>0)]
pattern=".{[minLength/],[if(maxLength<>0)][maxLength/][/if]}"[/if][if(maxLength<>0 and minLength
= 0)] maxlength="[maxLength/]"[/if][if(mandatory=true or minLength<>0)] required[/if]><br>
[/if]
[/let][/let][/let][/let][/let][/let][/let][/let]
[/template]

[template public createRichTextInput(a: Property) post(trim())]
[let historyAutocomplete: OclAny = a.getValue(a.getAppliedStereotype('RIA
Profile::richTextInput'), 'historyAutocomplete')]
[let autocomplete: OclAny = a.getValue(a.getAppliedStereotype('RIA Profile::richTextInput'),
'autocomplete')]
[let mandatory: OclAny = a.getValue(a.getAppliedStereotype('RIA Profile::richTextInput'),
'mandatory')]
[let minLength: OclAny = a.getValue(a.getAppliedStereotype('RIA Profile::richTextInput'),
'minLength')]
[let maxLength: OclAny = a.getValue(a.getAppliedStereotype('RIA Profile::richTextInput'),
'maxLength')]
[let onlyDigits: OclAny = a.getValue(a.getAppliedStereotype('RIA Profile::richTextInput'),
'onlyDigits')]
[let email: OclAny = a.getValue(a.getAppliedStereotype('RIA Profile::richTextInput'), 'email')]
[let password: OclAny = a.getValue(a.getAppliedStereotype('RIA Profile::richTextInput'),
'password')]
[let date : String = a.getValue(a.getAppliedStereotype('RIA
Profile::richTextInput'),'date').oclAsType(EnumerationLiteral).name]
[let title: OclAny = a.getValue(a.getAppliedStereotype('RIA Profile::richTextInput'), 'title')]
[if(title<>null)][title/]<br>[/if]
```

```
[if(onlyDigits = true)]
<input     [if(isInCalledUIElements(a.name)     or     autocomplete     =     true)]id="[a.name/]"
[/if]type="number"                                                    name="[a.name/]"[if(minLength<>0)]
pattern=".{[minLength/],[if(maxLength<>0)][maxLength/][/if]}"[/if][if(maxLength<>0 and minLength
=     0)]     maxlength="[maxLength/]"[/if][if(historyAutocomplete<>null)]     autocomplete="[if
(historyAutocomplete    =    false)]off[else]on[/if]"[/if][if(mandatory=true    or    minLength<>0)]
required[/if]><br>
[elseif(email = true)]
<input [if(isInCalledUIElements(a.name) or autocomplete = true)]id="[a.name/]" [/if]type="email"
name="[a.name/]"[if(minLength<>0)]
pattern=".{[minLength/],[if(maxLength<>0)][maxLength/][/if]}"[/if][if(maxLength<>0 and minLength
=     0)]     maxlength="[maxLength/]"[/if][if(historyAutocomplete<>null)]     autocomplete="[if
(historyAutocomplete    =    false)]off[else]on[/if]"[/if][if(mandatory=true    or    minLength<>0)]
required[/if]><br>
[elseif(password = true)]
<input     [if(isInCalledUIElements(a.name)     or     autocomplete     =     true)]id="[a.name/]"
[/if]type="password"                                                  name="[a.name/]"[if(minLength<>0)]
pattern=".{[minLength/],[if(maxLength<>0)][maxLength/][/if]}"[/if][if(maxLength<>0 and minLength
=     0)]     maxlength="[maxLength/]"[/if][if(historyAutocomplete<>null)]     autocomplete="[if
(historyAutocomplete    =    false)]off[else]on[/if]"[/if][if(mandatory=true    or    minLength<>0)]
required[/if]><br>
[elseif(date = 'default - dd/mm/yyyy')]
<input [if(isInCalledUIElements(a.name) or autocomplete = true)]id="[a.name/]" [/if]type="date"
name="[a.name/]"[if(minLength<>0)]
pattern=".{[minLength/],[if(maxLength<>0)][maxLength/][/if]}"[/if][if(maxLength<>0 and minLength
=     0)]     maxlength="[maxLength/]"[/if][if(historyAutocomplete<>null)]     autocomplete="[if
(historyAutocomplete    =    false)]off[else]on[/if]"[/if][if(mandatory=true    or    minLength<>0)]
required[/if]><br>
[else]
<input [if(isInCalledUIElements(a.name) or autocomplete = true)]id="[a.name/]" [/if]type="text"
name="[a.name/]"[if(minLength<>0)]
pattern=".{[minLength/],[if(maxLength<>0)][maxLength/][/if]}"[/if][if(maxLength<>0 and minLength
=     0)]     maxlength="[maxLength/]"[/if][if(historyAutocomplete<>null)]     autocomplete="[if
(historyAutocomplete    =    false)]off[else]on[/if]"[/if][if(mandatory=true    or    minLength<>0)]
required[/if]><br>
[/if]
[if(autocomplete = true)]
[populateAutocompleteTags(a)/]
[/if]
[/let][/let][/let][/let][/let][/let][/let][/let][/let][/let]
[/template]

[template public populateAutocompleteTags(a: Property)]
[let p : Class = getPresentationPage()]
[file ('ria/'+p.name+'/'+p.name+'.js', true, 'UTF-8')]
window.addEventListener('load', function(){
        $(function(){
                var [a.name/]Tags = ['['/]
        [let    lastDependetClass    :    Class    =    a.clientDependency.supplier->asSequence()-
>last().oclAsType(Class)]
        [for       (dependentClass       :       Class       |       a.clientDependency.supplier-
>asSequence().oclAsType(Class))]
                [if(dependentClass.hasStereotype('staticObject'))]
                        [let n : Integer = dependentClass.attribute->size()]
                        [for(v: Property | dependentClass.allAttributes())]
                                [if(v.hasStereotype('value'))]
                                        [let            title:            OclAny            =
v.getValue(v.getAppliedStereotype('Logic Profile::value'), 'title')]
                "[title/]"[if(i<>n or dependentClass.name <> lastDependetClass.name)],[/if]
                                        [/let]
                                [/if]
                        [/for]
                        [/let]
                [/if]
                [if(dependentClass.hasStereotype('clientStaticObject'))]
                        [let n : Integer = dependentClass.attribute->size()]
                        [if(n=1)]
                                [let v: Property = dependentClass.allAttributes()->first()]
                                [if(v.hasStereotype('value'))]
                [dependentClass.name/].title[if(dependentClass.name                          <>
lastDependetClass.name)],[/if]
                                [/if]
                                [/let]
                        [else]
```

```
                                [for(v: Property | dependentClass.allAttributes())]
                                        [if(v.hasStereotype('value'))]
                [dependentClass.name/]['['/][i-1/][']'/].title[if(i<>n or dependentClass.name <>
lastDependetClass.name)],[/if]
                                        [/if]
                                [/for]
                        [/if]
                        [/let]
                [/if]
        [/for]
        [/let]
                [']'/];
                $("#[a.name/]").autocomplete({
                        source: [a.name/]Tags
                });
        });
});
[/file]
[/let]
[/template]

[template public createList(a: Property) post(trim())]
[let       listType       :       String       =       a.getValue(a.getAppliedStereotype('Content
Profile::list'),'listType').oclAsType(EnumerationLiteral).name]
[let dependentClass : Class = a.clientDependency.supplier->asSequence()->first()]
[let title: OclAny = a.getValue(a.getAppliedStereotype('Content Profile::list'), 'title')]
[if(listType = 'dropBox')]
[if(title<>null)][title/]<br>[/if]
        [if(dependentClass.hasStereotype('staticObject'))]
<select>
                [for(v: Property | dependentClass.allAttributes())]
                        [if(v.hasStereotype('value'))]
                                [let  title:  OclAny  =  v.getValue(v.getAppliedStereotype('Logic
Profile::value'), 'title')]
                                [let checked: OclAny = v.getValue(v.getAppliedStereotype('Logic
Profile::value'), 'checked')]
        <option value="[v.name/]"[if(checked = true)] selected[/if]>[title/]</option>
                                [/let][/let]
                        [/if]
                [/for]
</select><br>
        [/if]
        [if(dependentClass.hasStereotype('clientStaticObject'))]
<select id="[a.name/]"></select><br>[populateListWithJavascript(a, dependentClass)/]
        [/if]
[/if]
[if(listType = 'choice')]
[if(title<>null)][title/]<br>[/if]
        [if(dependentClass.hasStereotype('staticObject'))]
                [for(v: Property | dependentClass.allAttributes())]
                        [if(v.hasStereotype('value'))]
                                [let  title:  OclAny  =  v.getValue(v.getAppliedStereotype('Logic
Profile::value'), 'title')]
                                [let checked: OclAny = v.getValue(v.getAppliedStereotype('Logic
Profile::value'), 'checked')]
<input  type="radio"  name="[a.name/]"  value="[v.name/]"[if(checked  =  true)]  checked[/if]>
[title/]<br>
                                [/let][/let]
                        [/if]
                [/for]
        [/if]
        [if(dependentClass.hasStereotype('clientStaticObject'))]
<div id="[a.name/]"></div>[populateListWithJavascript(a, dependentClass)/]
        [/if]
[/if]
[if(listType = 'check')]
[if(title<>null)][title/]<br>[/if]
        [if(dependentClass.hasStereotype('staticObject'))]
                [for(v: Property | dependentClass.allAttributes())]
                        [if(v.hasStereotype('value'))]
                                [let  title:  OclAny  =  v.getValue(v.getAppliedStereotype('Logic
Profile::value'), 'title')]
                                [let checked: OclAny = v.getValue(v.getAppliedStereotype('Logic
Profile::value'), 'checked')]
```

```
<input type="checkbox" name="[a.name/]" value="[v.name/]"[if(checked = true)] checked[/if]>
[title/]<br>
                                [/let][/let]
                        [/if]
                [/for]
        [/if]
        [if(dependentClass.hasStereotype('clientStaticObject'))]
<div id="[a.name/]"></div>[populateListWithJavascript(a, dependentClass)/]
        [/if]
[/if]
[/let][/let][/let]
[/template]

[template public populateListWithJavascript(list: Property, dependentClass : Class)]
[let p : Class = getPresentationPage()]
[let listType : String = list.getValue(list.getAppliedStereotype('Content
Profile::list'),'listType').oclAsType(EnumerationLiteral).name]
[let dCSize : Integer = dependentClass.allAttributes()->size()]
[file ('ria/'+p.name+'/'+p.name+'.js', true, 'UTF-8')]
window.addEventListener('load', function(){
        var list = document.getElementById("[list.name/]");
        [if(dCSize>1)]
        for(var i = 0; i < [dependentClass.name/].length; i++) {
                [javascriptListContent(list,dependentClass)/]
        }
        [else]
        [javascriptListContent(list,dependentClass)/]
        [/if]
});
[/file]
[/let][/let][/let]
[/template]

[template public javascriptListContent(list: Property, dependentClass : Class) post(trim())]
[let listType : String = list.getValue(list.getAppliedStereotype('Content
Profile::list'),'listType').oclAsType(EnumerationLiteral).name]
[let dCSize : Integer = dependentClass.allAttributes()->size()]
[if(listType = 'dropBox')]
var option = document.createElement("option");
option.textContent = [dependentClass.name/][if(dCSize>1)]['['/]i[']'/][/if].title;
option.value = [dependentClass.name/][if(dCSize>1)]['['/]i[']'/][/if].value;
option.selected = [dependentClass.name/][if(dCSize>1)]['['/]i[']'/][/if].checked;
list.appendChild(option);
[else]
var option = document.createElement("input");
var optionText = document.createElement("label");
[if(listType = 'choice')]
option.type = "radio";
[elseif(listType = 'check')]
option.type = "checkbox";
[/if]
option.name = "[list.name/]";
option.value = [dependentClass.name/][if(dCSize>1)]['['/]i[']'/][/if].value;
option.checked = [dependentClass.name/][if(dCSize>1)]['['/]i[']'/][/if].checked;
optionText.innerHTML = ' '+[dependentClass.name/][if(dCSize>1)]['['/]i[']'/][/if].title+'<br>';
list.appendChild(option);
list.appendChild(optionText);
[/if]
[/let][/let]
[/template]

[template public getAllAsociatedClasses(c: Class)]
[clearVisitedClasses()/]
[addToVisitedClasses(c)/]
[getAssociatedClasses(c)/]
[for (c1 : Class | getVisitedClasses().oclAsType(Class))]
        [getAssociatedClasses(c1)/]
[/for]
[for (c1 : Class | getVisitedClasses().oclAsType(Class))]
        [getAssociatedClasses(c1)/]
[/for]
[for (c1 : Class | getVisitedClasses().oclAsType(Class))]
        [getAssociatedClasses(c1)/]
[/for]
```

```
[/template]

[template public getAssociatedClasses(c: Class)]
[for(a: Association | c.getAssociations())]
        [let e1: Class = a.memberEnd->at(1).type.oclAsType(Class)][let e2: Class = a.memberEnd-
>at(2).type.oclAsType(Class)]
        [if(not(isInVisitedClasses(e1.name)))][addToVisitedClasses(e1)/][/if]
        [if(not(isInVisitedClasses(e2.name)))][addToVisitedClasses(e2)/][/if]
        [/let][/let]
[/for]
[/template]

[template public service(s: State)]
[file ('ria/servicios/'+s.name+'.php', false, 'UTF-8')]
<?php
//Coloque el codigo correspondiente al servicio
?>
[/file]
[/template]
```

Java Services

```
package riaCodeGenerator.services;

import java.util.ArrayList;
import java.util.List;

import org.eclipse.uml2.uml.Class;
import org.eclipse.uml2.uml.Property;
import org.eclipse.uml2.uml.State;
import org.eclipse.uml2.uml.Stereotype;

public class Services {

        public boolean hasAsynchronousCall = false;
        public boolean hasRichTable = false;
        public boolean hasClientStaticObject = false;
        public boolean hasAutocomplete = false;
        public Class presentationPage;
        public ArrayList<Class> visitedClasses = new ArrayList<Class>();
        public ArrayList<Property> calledUIElements = new ArrayList<Property>();

        public boolean hasStereotype(Class iclass, String stereotypeName) {
          List<Stereotype> stereotypes = iclass.getAppliedStereotypes();
            for (Stereotype stereotype : stereotypes) {
              if (stereotype.getName().equals(stereotypeName)) {
                return true;
              }
            }
            return false;
        }

        public boolean hasStereotype(Property iproperty, String stereotypeName) {
          List<Stereotype> stereotypes = iproperty.getAppliedStereotypes();
            for (Stereotype stereotype : stereotypes) {
              if (stereotype.getName().equals(stereotypeName)) {
                return true;
              }
            }
            return false;
        }

        public boolean hasStereotype(State istate, String stereotypeName) {
                  List<Stereotype> stereotypes = istate.getAppliedStereotypes();
                    for (Stereotype stereotype : stereotypes) {
                      if (stereotype.getName().equals(stereotypeName)) {
                        return true;
                      }
                    }
                    return false;
                }

        public void addToVisitedClasses(Class c){
                visitedClasses.add(c);
```

```
}

public boolean isInVisitedClasses(String cname){
        for(int i=0; i<visitedClasses.size();i++){
                if (visitedClasses.get(i).getName().equals(cname)){
                        return true;
                }
        }
        return false;
}

public void clearVisitedClasses(){
        visitedClasses.clear();
}

public ArrayList<Class> getVisitedClasses(){
        return visitedClasses;
}

public void addToCalledUIElements(Property p){
        calledUIElements.add(p);
}

public boolean isInCalledUIElements(String cname){
        for(int i=0; i<calledUIElements.size();i++){
                if (calledUIElements.get(i).getName().equals(cname)){
                        return true;
                }
        }
        return false;
}

public void clearCalledUIElements(){
        calledUIElements.clear();
}

public ArrayList<Property> getCalledUIElements(){
        return calledUIElements;
}

public void setHasAsynchronousCall(){
        hasAsynchronousCall = true;
}

public void resetHasAsynchronousCall(){
        hasAsynchronousCall = false;
}

public boolean getHasAsynchronousCall(){
        return hasAsynchronousCall;
}

public void setHasRichTable(){
        hasRichTable = true;
}

public void resetHasRichTable(){
        hasRichTable = false;
}

public boolean getHasRichTable(){
        return hasRichTable;
}

public void setHasClientStaticObject(){
        hasClientStaticObject = true;
}

public void resetHasClientStaticObject(){
        hasClientStaticObject = false;
}

public boolean getHasClientStaticObject(){
        return hasClientStaticObject;
```

```
	}

	public void setHasAutocomplete(){
		hasAutocomplete = true;
	}

	public void resetHasAutocomplete(){
		hasAutocomplete = false;
	}

	public boolean getHasAutocomplete(){
		return hasAutocomplete;
	}

	public void setPresentationPage(Class c){
		presentationPage = c;
	}

	public Class getPresentationPage(){
		return presentationPage;
	}

}
```

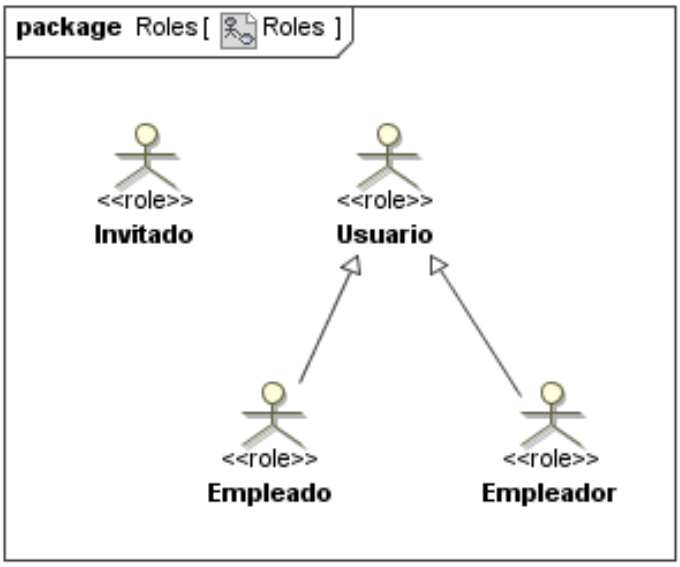
Java Services Wrapper

```
[comment encoding = Cp1252 /]
[module services('http://www.eclipse.org/uml2/5.0.0/UML')/]

[query public hasStereotype(arg0 : Class, arg1 : String) : Boolean
	=                                          invoke('riaCodeGenerator.services.Services',
'hasStereotype(org.eclipse.uml2.uml.Class, java.lang.String)', Sequence{arg0, arg1})
/]

[query public hasStereotype(arg0 : Property, arg1 : String) : Boolean
	=                                          invoke('riaCodeGenerator.services.Services',
'hasStereotype(org.eclipse.uml2.uml.Property, java.lang.String)', Sequence{arg0, arg1})
/]

[query public hasStereotype(arg0 : State, arg1 : String) : Boolean
	=                                          invoke('riaCodeGenerator.services.Services',
'hasStereotype(org.eclipse.uml2.uml.State, java.lang.String)', Sequence{arg0, arg1})
/]

[query public addToVisitedClasses(arg0 : Class) : OclVoid
	=                                          invoke('riaCodeGenerator.services.Services',
'addToVisitedClasses(org.eclipse.uml2.uml.Class)', Sequence{arg0})
/]

[query public isInVisitedClasses(arg0 : String) : Boolean
	=  invoke('riaCodeGenerator.services.Services',  'isInVisitedClasses(java.lang.String)',
Sequence{arg0})
/]

[query public clearVisitedClasses(anOclAny: OclAny) : OclVoid
	= invoke('riaCodeGenerator.services.Services', 'clearVisitedClasses()', Sequence{})
/]

[query public getVisitedClasses(anOclAny: OclAny) : Sequence(OclAny)
	= invoke('riaCodeGenerator.services.Services', 'getVisitedClasses()', Sequence{})
/]

[query public addToCalledUIElements(arg0 : Property) : OclVoid
	=                                          invoke('riaCodeGenerator.services.Services',
'addToCalledUIElements(org.eclipse.uml2.uml.Property)', Sequence{arg0})
/]

[query public isInCalledUIElements(arg0 : String) : Boolean
	= invoke('riaCodeGenerator.services.Services', 'isInCalledUIElements(java.lang.String)',
Sequence{arg0})
/]

[query public clearCalledUIElements(anOclAny: OclAny) : OclVoid
	= invoke('riaCodeGenerator.services.Services', 'clearCalledUIElements()', Sequence{})
```

```
/]

[query public getCalledUIElements(anOclAny: OclAny) : Sequence(OclAny)
        = invoke('riaCodeGenerator.services.Services', 'getCalledUIElements()', Sequence{})
/]

[query public setHasAsynchronousCall(anOclAny: OclAny) : OclVoid
        = invoke('riaCodeGenerator.services.Services', 'setHasAsynchronousCall()', Sequence{})
/]

[query public resetHasAsynchronousCall(anOclAny: OclAny) : OclVoid
        = invoke('riaCodeGenerator.services.Services', 'resetHasAsynchronousCall()', Sequence{})
/]

[query public getHasAsynchronousCall(anOclAny: OclAny) : Boolean
        = invoke('riaCodeGenerator.services.Services', 'getHasAsynchronousCall()', Sequence{})
/]

[query public setHasRichTable(anOclAny: OclAny) : OclVoid
        = invoke('riaCodeGenerator.services.Services', 'setHasRichTable()', Sequence{})
/]

[query public resetHasRichTable(anOclAny: OclAny) : OclVoid
        = invoke('riaCodeGenerator.services.Services', 'resetHasRichTable()', Sequence{})
/]

[query public getHasRichTable(anOclAny: OclAny) : Boolean
        = invoke('riaCodeGenerator.services.Services', 'getHasRichTable()', Sequence{})
/]

[query public setHasClientStaticObject(anOclAny: OclAny) : OclVoid
        = invoke('riaCodeGenerator.services.Services', 'setHasClientStaticObject()', Sequence{})
/]

[query public resetHasClientStaticObject(anOclAny: OclAny) : OclVoid
        =      invoke('riaCodeGenerator.services.Services',      'resetHasClientStaticObject()',
Sequence{})
/]

[query public getHasClientStaticObject(anOclAny: OclAny) : Boolean
        = invoke('riaCodeGenerator.services.Services', 'getHasClientStaticObject()', Sequence{})
/]

[query public setHasAutocomplete(anOclAny: OclAny) : OclVoid
        = invoke('riaCodeGenerator.services.Services', 'setHasAutocomplete()', Sequence{})
/]

[query public resetHasAutocomplete(anOclAny: OclAny) : OclVoid
        = invoke('riaCodeGenerator.services.Services', 'resetHasAutocomplete()', Sequence{})
/]

[query public getHasAutocomplete(anOclAny: OclAny) : Boolean
        = invoke('riaCodeGenerator.services.Services', 'getHasAutocomplete()', Sequence{})
/]

[query public setPresentationPage(arg0 : Class) : OclVoid
        =                                          invoke('riaCodeGenerator.services.Services',
'setPresentationPage(org.eclipse.uml2.uml.Class)', Sequence{arg0})
/]

[query public getPresentationPage(anOclAny: OclAny) : Class
        = invoke('riaCodeGenerator.services.Services', 'getPresentationPage()', Sequence{})
/]
```

Anexo 3. Diagramas complementarios del Sistema de Marcación de Empleados

Diagrama de Roles y Zonas



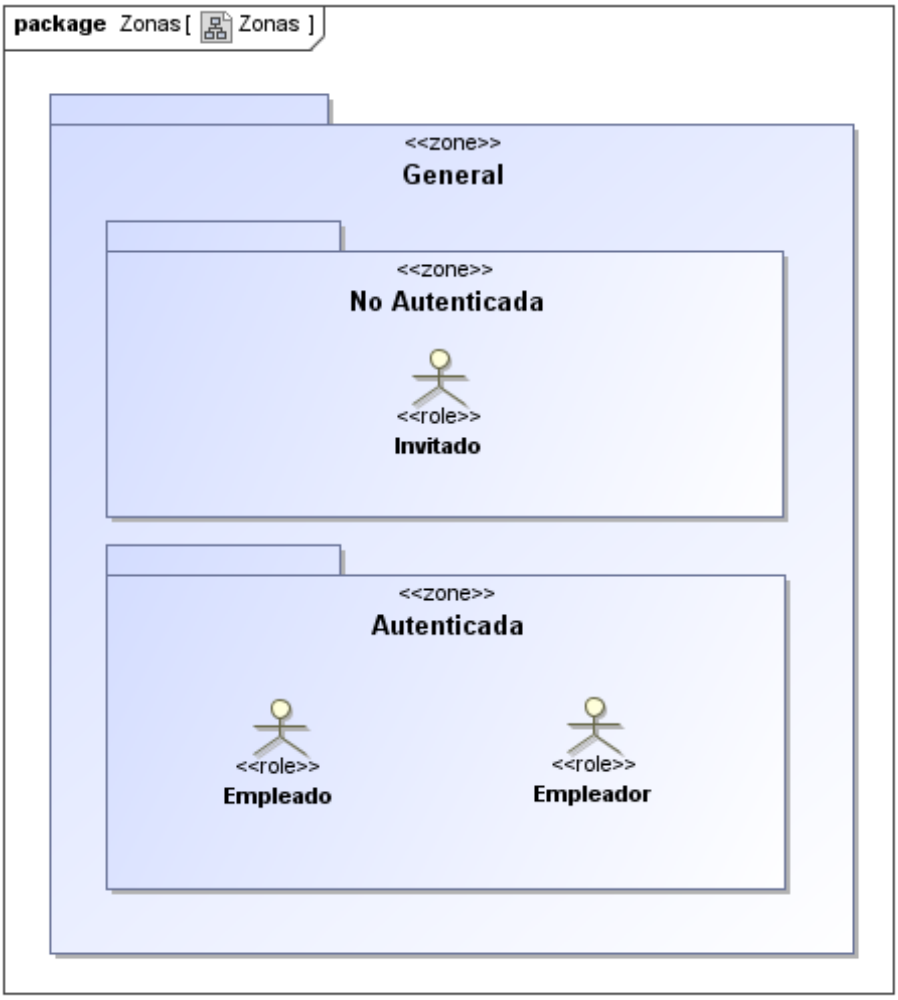
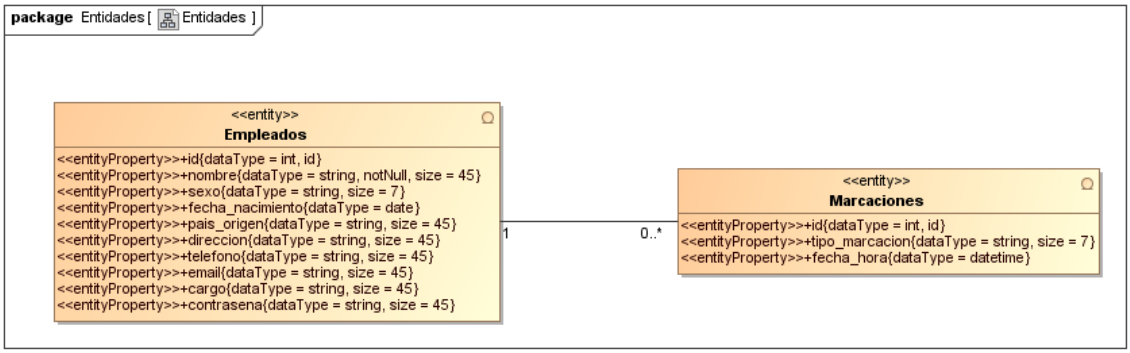
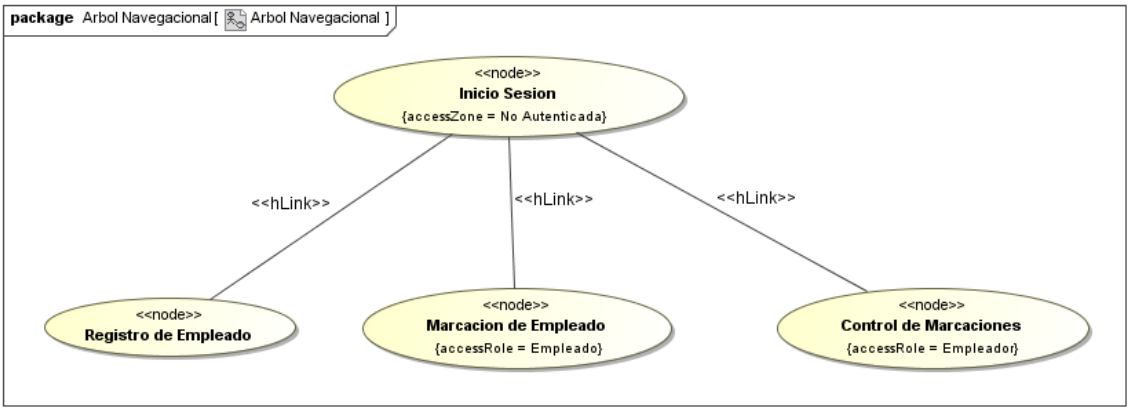


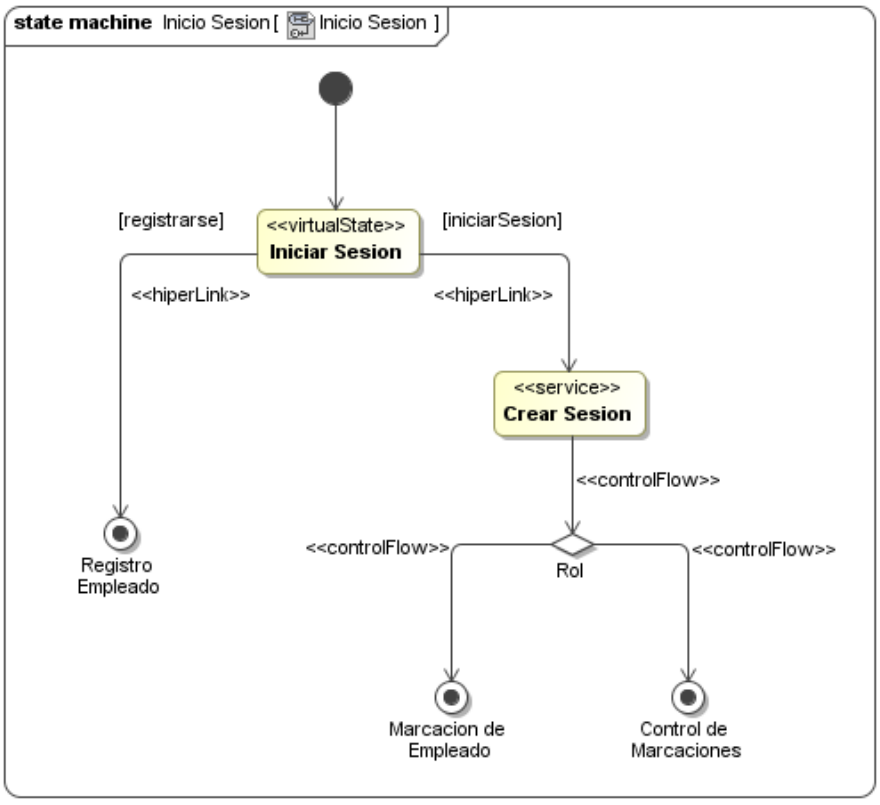
Diagrama de Entidades

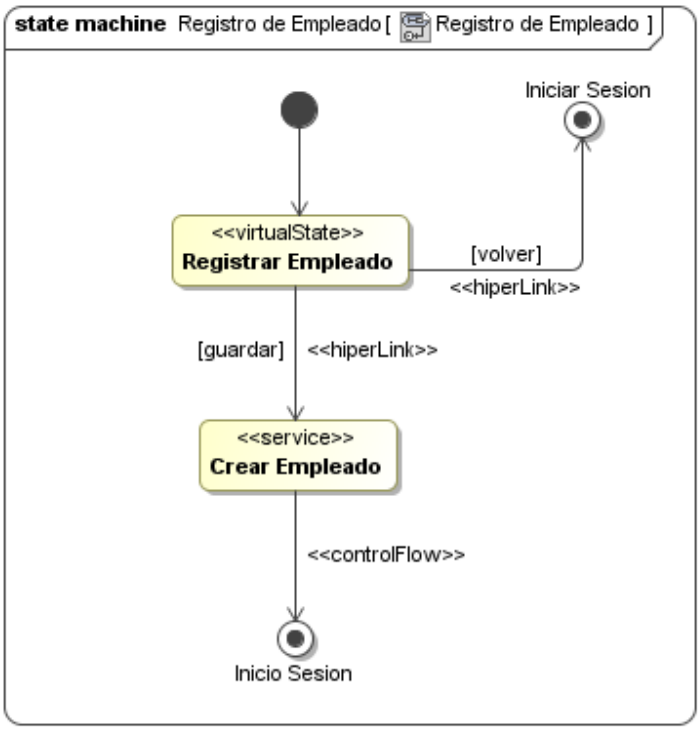


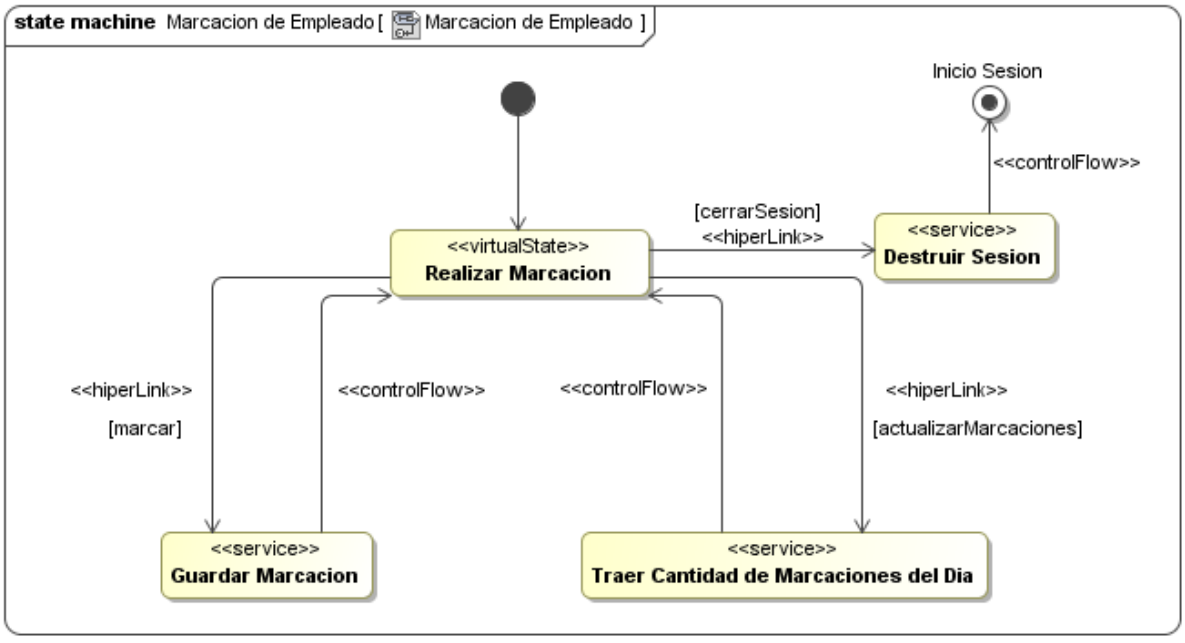
Árbol Navegacional

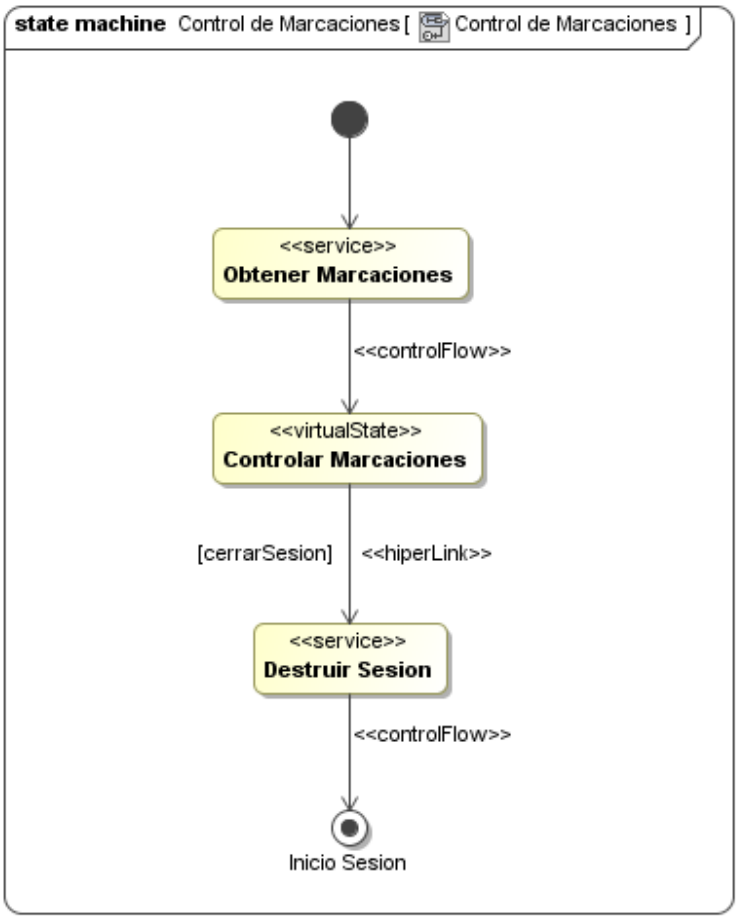


Diagramas de Nodos

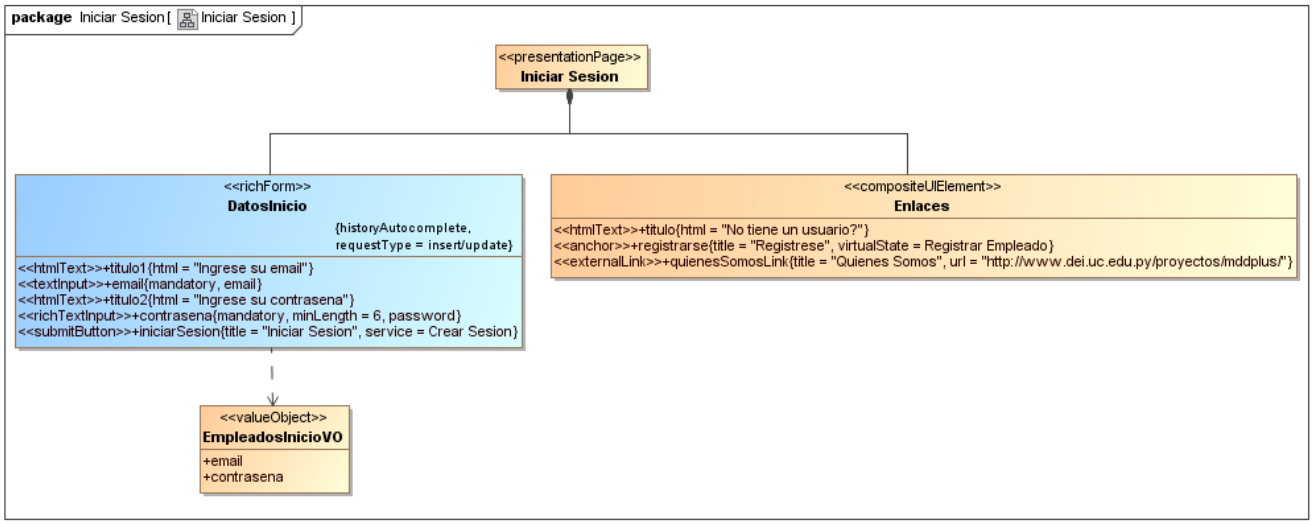



state machine Registro de Empleado [Registro de Empleado]
Iniciar Sesion
<<virtualState>>
Registrar Empleado
[volver]
<<hiperLink>>
[guardar]
<<hiperLink>>
<<service>>
Crear Empleado
<<controlFlow>>
Inicio Sesion


state machine Marcacion de Empleado [Marcacion de Empleado]
Inicio Sesion
<<controlFlow>>
[cerrarSesion]
<<hiperLink>>
<<virtualState>>
Realizar Marcacion
<<service>>
Destruir Sesion
<<hiperLink>>
[marcar]
<<controlFlow>>
<<controlFlow>>
<<hiperLink>>
[actualizarMarcaciones]
<<service>>
Guardar Marcacion
<<service>>
Traer Cantidad de Marcaciones del Dia



Diagramas de Contenido

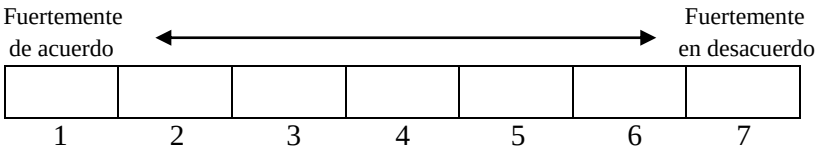


Anexo 4. Cuestionarios utilizados durante la Experiencia de Validación

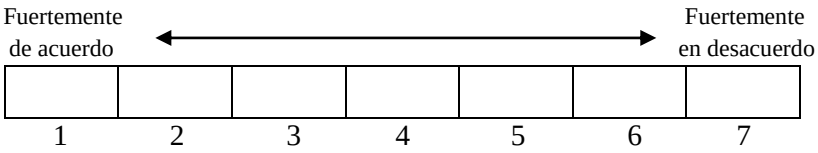
Cuestionario sobre el proceso de modelado

Escenario	**Tareas comprendidas**
Modelado	• Diseño de formulario. • Diseño de botón. • Diseño de tabla. • Diseño de variables locales. • Diseño de servicios invocados.

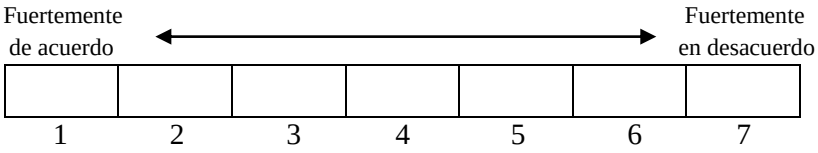
1. Estoy satisfecho con la facilidad de completar las tareas en este escenario.



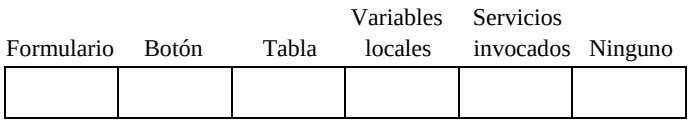
2. Estoy satisfecho con la cantidad de tiempo que tomó completar las tareas de este escenario.



3. Estoy satisfecho con la información de soporte al completar las tareas.



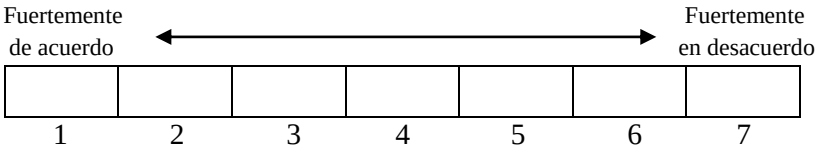
4. Me resultó difícil modelar el/los siguiente/s elemento/s:



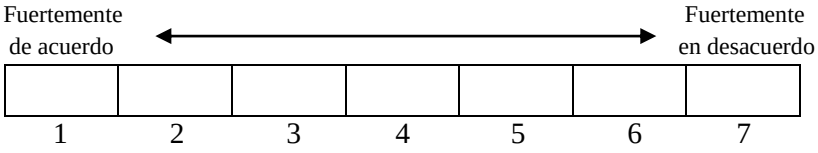
Cuestionario sobre el proceso de generación de código

Escenario	**Tareas comprendidas**
Generación de código	• Exportar el modelo desde magicDraw. • Importar el modelo en Eclipse. • Ejecutar el generador de código. • Realizar ajustes manuales al código.

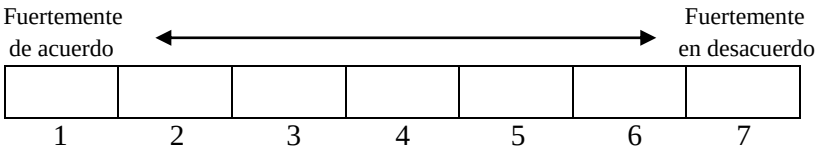
1. Estoy satisfecho con la facilidad de completar las tareas en este escenario.



2. Estoy satisfecho con la cantidad de tiempo que tomó completar las tareas de este escenario.

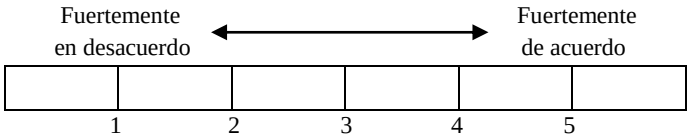


3. Estoy satisfecho con la información de soporte al completar las tareas.



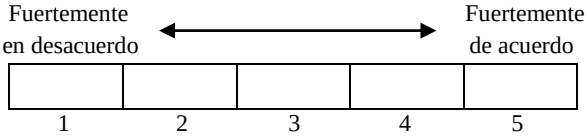
Cuestionario sobre el enfoque MDD para el desarrollo de RIA

1. Pienso que me gustaría utilizar el enfoque frecuentemente.

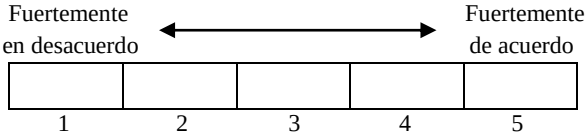


2. Creo que el enfoque es innecesariamente complejo.

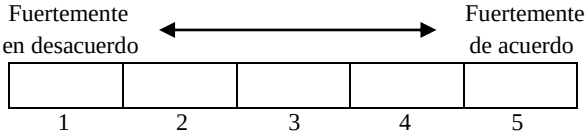
3. Pienso que el enfoque fue fácil de utilizar.



4. Pienso que necesitaría el soporte de una persona técnica para poder utilizar el enfoque.



5. Creo que las funciones del enfoque están bien integradas.



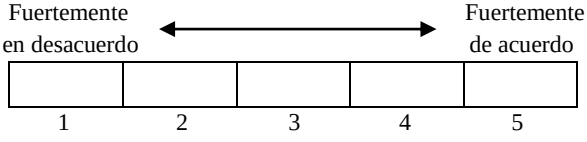
6. Pienso que hay muchas inconsistencias en el enfoque.

7. Imagino que la mayoría de los desarrolladores aprenderían a utilizar el enfoque rápidamente.

8. Creo que el enfoque es complicado de utilizar.

9. Me sentí muy seguro y confiado utilizando el enfoque.

10. Necesito aprender muchas cosas para poder manejarme con enfoque.



8. Referencias

[1] J. C. Preciado, M. Linaje, F. Sanchez, and S. Comai, "Necessity of methodologies to model rich internet applications," in *Web Site Evolution, 2005.(WSE 2005). Seventh IEEE International Symposium on*. IEEE, 2005, pp. 7–13.

[2] G. T. Carughi, S. Comai, A. Bozzon, and P. Fraternali, "Modeling distributed events in data-intensive rich internet applications," in *Web Information Systems Engineering–WISE 2007*. Springer, 2007, pp. 593–602.

[3] G. Toffetti, S. Comai, J. C. Preciado, and M. Linaje, "State-of-the-art and trends in the systematic development of rich internet applications," *Journal of Web Engineering*, vol. 10, no. 1, pp. 070–086, 2011.

[4] M. Brambilla, J. Cabot, and M. Wimmer, "Model-driven software engineering in practice," *Synthesis Lectures on Software Engineering*, vol. 1, no. 1, pp. 1–182, 2012.

[5] O. M. Group, "Mda specifications." [Online]. Available: http://www.omg.org/mda/specs.htm

[6] M. González, L. Cernuzzi, and O. Pastor, "A navigational role-centric model oriented web approach-moweba," *International Journal of Web Engineering and Technology*, vol. 11, no. 1, pp. 29–67, 2016.

[7] I. López, M. González, N. Aquino, and L. Cernuzzi, "Una propuesta basada en model driven architecture para el soporte de rich internet applications," in *Memorias de la XIX Conferencia Iberoamericana de Software Engineering*, Ecuador, Quito, Apr. 2016, pp. 25–38.

[8] M. Brambilla, J. C. Preciado, M. Linaje, and F. Sanchez-Figueroa, "Business process-based conceptual design of rich internet applications," in *Web Engineering, 2008. ICWE'08. Eighth International Conference on*. IEEE, 2008, pp. 155–161.

[9] J. Allaire, "Macromedia flash mx—a next-generation rich client," *Macromedia white paper*, pp. 1–2, 2002.

[10] P. Fraternali, G. Rossi, and F. Sánchez-Figueroa, "Rich internet applications," *Internet Computing, IEEE*, vol. 14, no. 3, pp. 9–12, 2010.

[11] M. Busch and N. Koch, "Rich internet applications. state-of-the-art," Ludwig-Maximilians-Universität München, München, Germany, techreport 0902, 2009.

[12] J. Farrell and G. S. Nezlek, "Rich internet applications the next stage of application development," in *Information Technology Interfaces, 2007. ITI 2007. 29th International Conference on*. IEEE, 2007, pp. 413–418.

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

[14] T. Stahl, M. Voelter, and K. Czarnecki, *Model-driven software development: technology, engineering, management*. John Wiley & Sons, 2006.

[15] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," Keele University and University of Durham, techreport EBSE-2007-01, 2007, version 2.3.

[16] M. Genero Bocco, J. A. Cruz-Lemus, and M. G. Piattini Velthuis, *Métodos de Investigación en Ingeniería de Software*, RA-MA, Ed., 2014.

[17] S. Casteleyn, I. Garrig'os, and J.-N. Maz'on, "Ten years of rich internet applications: A systematic mapping study, and beyond," *ACM Trans. Web*, vol. 8, no. 3, pp. 18:1–18:46, Jul. 2014. [Online]. Available: http://doi.acm.org/10.1145/2626369

[18] F. Valverde and O. Pastor, "Facing the technological challenges of web 2.0: A ria model-driven engineering approach," in *International Conference on Web Information Systems Engineering*. Springer, 2009, pp. 131–144.

[19] J. Fons, V. Pelechano, M. Albert, and O. Pastor, "Development of web applications from web enhanced conceptual schemas," in *International Conference on Conceptual Modeling*. Springer, 2003, pp.

232–245.

[20] J. L. H. Agustin and P. C. Del Barco, “A model-driven approach to develop high performance web applications,” *Journal of Systems and Software*, vol. 86, no. 12, pp. 3013–3023, 2013.

[21] M. Linaje, J. C. Preciado, and F. Sánchez-Figueroa, “A method for model based design of rich internet application interactive user interfaces,” in *Web Engineering*. Springer, 2007, pp. 226–241.

[22] A. Bozzon, S. Comai, P. Fraternali, and G. T. Carughi, “Conceptual modeling and code generation for rich internet applications,” in *Proceedings of the 6th international conference on Web engineering*. ACM, 2006, pp. 353–360.

[23] A. Bozzon, S. Comai, P. Fraternali, and M. Tisi, “Current research on the design of web 2.0 applications based on model-driven approaches,” in *ICWE 2008 Workshops*. Citeseer, p. 25.

[24] J. C. Preciado, M. Linaje, S. Comai, and F. Sanchez-Figueroa, “Designing rich internet applications with web engineering methodologies,” in *Web Site Evolution, 2007. WSE 2007. 9th IEEE International Workshop on*. IEEE, 2007, pp. 23–30.

[25] M. Brambilla, P. Fraternali, and E. Molteni, “A tool for model-driven design of rich internet applications based on ajax,” *Handbook of Research on Web*, vol. 2, no. 3.0, pp. 96–118, 2009.

[26] S. Meliá, J. Gómez, S. Pérez, and O. Díaz, “Architectural and technological variability in rich internet applications,” *IEEE Internet Computing*, vol. 14, no. 3, pp. 24–32, 2010.

[27] S. Meliá, J. J. M. Domene, Á. Pérez, and J. Gómez, “Ooh4ria tool: Una herramienta basada en el desarrollo dirigido por modelos para las rias.” in *JISBD*, 2009, pp. 219–222.

[28] M. Abrams, C. Phanouriou, A. L. Batongbacal, S. M. Williams, and J. E. Shuster, “Uiml: an appliance-independent xml user interface language,” *Computer Networks*, vol. 31, no. 11, pp. 1695–1708, 1999.

[29] S. Ceri, P. Fraternali, and A. Bongio, “Web modeling language (webml): a modeling language for designing web sites,” *Computer Networks*, vol. 33, no. 1, pp. 137–157, 2000.

[30] S. Meliá, J. Gomez, S. Perez, and O. Diaz, “A model-driven development for gwt-based rich internet applications with ooh4ria,” in *Web Engineering, 2008. ICWE’08. Eighth International Conference on*. IEEE, 2008, pp. 13–23.

[31] S. ISO, “9241-11. 1998,” *Ergonomic Requirements for Office Work with Visual Display Terminals (VDTs)–Part II Guidance on Usability*, 1998.

[32] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in software engineering*. Springer Science & Business Media, 2012.

[33] V. R. Basili and H. D. Rombach, “The tame project: Towards improvement-oriented software environments,” *IEEE Transactions on software engineering*, vol. 14, no. 6, pp. 758–773, 1988.

[34] R. K. Yin, *Case study research: Design and methods*. Sage publications, 2013.

[35] J. R. Lewis, “Psychometric evaluation of an after-scenario questionnaire for computer usability studies: the asq,” *ACM SIGCHI Bulletin*, vol. 23, no. 1, pp. 78–81, 1991.

[36] J. Brooke, “Sus—a quick and dirty usability scale. 1996,” *PW Jordan, B. Thomas, BA Weerdmeester and AL McClelland*, 1996.

[37] J. R. Lewis, “Ibm computer usability satisfaction questionnaires: psychometric evaluation and instructions for use,” *International Journal of Human-Computer Interaction*, vol. 7, no. 1, pp. 57–78, 1995.

[38] W. Albert and T. Tullis, *Measuring the user experience: collecting, analyzing, and presenting usability metrics*. Newnes, 2013.

[39] J. Sauro, “Measuring usability with the system usability scale (sus),” 2011.

[40] J. Sauro and J. R. Lewis, *Quantifying the user experience: Practical statistics for user research*. Elsevier, 2012.