Universidad Católica

“Nuestra Señora de la Asunción”

Facultad de Ciencias y Tecnología

***Departamento de Ingeniería Electrónica e Informática***



**Proyecto Final**

*“****Una propuesta MDA para el soporte de aplicaciones RIA****”*

**Alumno: Iván López.**

**Coordinadores: Ing. Magalí González.**

**Msc. Ing. Nathalie Aquino.**

**Asunción**

**2015**

**1- Introducción**

Hoy en día las aplicaciones web toman un rol protagónico, debido a que los usuarios demandan mejores aplicaciones, que sean más interactivas y que ofrezcan funcionalidades naturalmente intuitivas y ágiles. De alguna forma, esta demanda se ha podido lograr, gracias a la ingeniería web que define el uso de procesos científicos y principios de administración, acompañado de enfoques sistemáticos, con la meta de desarrollar, desplegar y mantener satisfactoriamente una alta calidad en los sistemas y aplicaciones basados en Web []. Es por eso que la esencia de la ingeniería Web se basa en administrar adecuadamente la diversidad y complejidad en el desarrollo de las aplicaciones Web evitando así, fallas potenciales que pueden llevar a tener serias implicancias.

Con la idea de que las aplicaciones Web se asemejen lo más posible a las aplicaciones de escritorio, nacieron las *Rich Internet Applicantions* (RIA). Estas representan todo un desafío para la ingeniería Web, ya que las RIAs han dado un cambio radical en la manera en que se comportan, desarrollan y despliegan las aplicaciones Web, ofreciendo mejoras substanciales con respecto a las aplicaciones Web tradicionales, con nuevas características referentes a la comunicación, la distribución de los datos y la computación en el lado cliente, acompañadas de interfaces mucho mas interactivas, en donde el usuario en ocasiones, no distingue si está utilizando la aplicación online o offline. Con estos avances propuestos por RIA, muchas de las metodologías Web tradicionales basadas en la Web 1.0, tales como; WebML[], UWE[], OOH[], OOHDM[] y OOWS, han tenido que evolucionar de cierta forma, agregando nuevos modelos o extendiendo los existentes, para dar cobertura a las diversas características sofisticadas propuestas por RIA. Muchas de las metodologías citadas han logrado una notable evolución en su afán de mantenerse vigentes con los avances propuestos por las RIA, sin embargo en la actualidad, ninguna de ellas, logra satisfacer todas las nuevas funcionalidades [] [] [] [].

He allí que resulta necesario crear nuevas metodologías de desarrollo Web o bien extender las actuales para satisfacer las nuevas características impuestas por la tendencia actual. Con ese propósito, un nuevo enfoque para el desarrollo de aplicaciones Web basado en modelos y fundamentado en los principios propuestos por la OMG[[1]](#footnote-1), se ha propuesto en el DEI[[2]](#footnote-2). Este enfoque está basado en los estándares MDA[[3]](#footnote-3) y ofrece un esquema de modelado en capas para la separación de conceptos. Dicho enfoque se denomina MoWebA[][], y en la actualidad cuenta con características de modelado a nivel de presentación, lógica de negocio, navegación y adaptabilidad de los usuarios, pudiendo generarse aplicaciones Web completas y funcionales con modelos independientes de la plataforma. MoWebA tiene la capacidad de llevar a cabo extensiones a sus metamodelos para cubrir nuevas características, lo cual la hace adaptable a los cambios actuales.

El objetivo de este trabajo de fin de carrera se enmarca en la idea de llevar a cabo extensiones a la propuesta Web MoWebA con respecto a la capa de Presentación, con el fin de abarcar a algunas de las principales características de las RIA. Para llevar a cabo esta propuesta de extensión, se divide el esquema de trabajo de la siguiente manera. Se definen primeramente las RIA, presentando sus principales características y los nuevos aportes a las aplicaciones Web tradicionales. Posteriormente se presenta el estado del arte de las metodologías de desarrollo basada en modelos MDD que dan cobertura a características de RIA. A continuación se presenta la aproximación de desarrollo Web MoWebA. Se extiende luego el metamodelo de contenido de MoWebA y se presenta una propuesta de transformación de modelo a texto (M2T) para la plataforma destino *jQueryUI[[4]](#footnote-4)* y *jQuery Validation Plugin[[5]](#footnote-5)*, para cubrir algunas características RIA de las presentaciones enriquecidas y de la lógica de negocios en el lado cliente. Como siguiente paso se evalúa el metamodelo de contenido extendido con una ilustración comparativa. Se finaliza el trabajo con un análisis de los resultados obtenidos elaborando la conclusión y los posibles trabajos futuros.

**2-Rich Internet Applications (RIA)**

El término RIA, fue introducido en marzo de 2002 por la empresa Macromedia (actualmente Adobe) que en ese entonces abordaba las limitaciones en cuanto a la riqueza de las interfaces, medios y contenidos de las aplicaciones []. En la Figura 1 se presentan los tres aspectos que en conjunción, dieron origen a las RIA. Las RIA son aplicaciones Web que exhiben widgets, comportamientos y características que están presentes en las aplicaciones de escritorio. Poseen una mayor capacidad de respuesta, son más seguras y presentan una interfaz más avanzada con respecto a las aplicaciones del modelo Web 1.0. Sus características principales incluyen, el paradigma de página única, un avanzado esquema de comunicación (con la inclusión de tecnologías push y comunicación asíncrona entre el cliente y el servidor y un manejo optimizado de los datos, reduciendo las solicitudes al servidor) y finalmente, la inclusión de un motor en el cliente (en la forma de máquina virtual o extensiones (plug-ins) en el navegador que administra la disposición gráfica de los elementos y la mayoría de las interacciones locales [].

Figura 1 Fundamento de las RIAs. Las aplicaciones Web, las aplicaciones de escritorio y las tecnologías de comunicación.

**2.1 Características principales de las RIA**

A continuación se presentan las cuatro características principales de las aplicaciones RIA.

**Almacenamiento de los datos:** en las RIA, es posible almacenar datos en el lado cliente, con diferentes niveles de persistencia (temporalmente, mientras la aplicación está en ejecución, o persistentemente). También, los datos pueden distribuirse entre ambos pares, cliente y servidor.

## Lógica de negocio: en las RIA es posible llevar a cabo operaciones complejas directamente en el cliente (por ejemplo: efectuar navegaciones, filtrados y ordenamiento de los datos con múltiples criterios; operaciones de dominio específico para sistemas complejos; y validación local de datos). También es factible distribuir la lógica de negocios entre el cliente y el servidor para, por ejemplo, validar algunos campos de un formulario en el cliente y otros en el servidor. Por lo tanto, el diseño conceptual debe responder a la decisión de cómo asignar la computación tanto de las páginas como así también de los componentes de éstas [14]

## Comunicación entre el cliente y el servidor: con las RIA se crean mecanismos para reducir al mínimo la transferencia de los datos migrando las capas de interacción y presentación del servidor al cliente. Las RIA soportan comunicaciones asíncronas entre el cliente y el servidor para la distribución de objetos de dominio, datos y la computación.

## Presentaciones enriquecidas: las interfaces de usuario ofrecen una mayor riqueza con el manejo de eventos en el lado del cliente y los widgets interactivos que son micro programas empotrados dentro de las páginas Web y administradas por un motor de widgets (que podría ser un plug-in instalado en el navegador). Los widgets presentan funciones bien específicas que por lo común resultan de utilidad a los usuarios tales como: presentar el estado del tiempo, la hora de diversos países, la cotización de las monedas extranjeras, calculadoras, entre otros. Los elementos multimedia dentro de las páginas como la intrusión de audio y video de alta calidad, a la par de animaciones también son características típicas de las RIA, como así también, la capacidad de arrastrar y soltar elementos dentro de la interfaz, las auto-sugerencias de datos a medida que se va escribiendo un patrón en un campo y el refrescado automático de las páginas (o porciones de esta), son otras de las características interesantes que pueden encontrarse.

**2.2 Herramientas para el desarrollo de las RIA**

Existen diferentes tecnologías para el desarrollo e implementación de las RIA. Las implementaciones basadas en *Javascript* o librerías Ajax son las más utilizadas en la actualidad, debido a que utiliza tecnologías de uso abierto estandarizado como; *Javascript*, HTML y CSS. Estas librerías tienen como objetivo abstraer a los desarrolladores de tener que lidiar directamente con el DOM (*Document Object Model*) para la disposición de los elementos en las páginas Web, ofreciendo capas de software amigable, reduciendo notablemente los tiempos de desarrollo y mejorando la productividad. En la Figura 2 se puede apreciar algunas librerías *Javascript* de uso extendido.



Figura 2 Algunas librerías *Javascript* de uso común

Estas librerías también buscan explotar el lado del cliente en las aplicaciones y minimizar las interacciones con el lado servidor, para que de esta forma se obtenga un mejor rendimiento. A la par de permitir a los desarrolladores implementar aplicaciones a un alto nivel de abstracción, las librerías ofrecen una gran variedad de *widgets* interactivos que son de uso común en las aplicaciones Web. Los *widgets* para las RIA representan elementos enriquecidos para la interfaz de usuario, que tienen como objetivo ofrecer una mayor interactividad, dada sus características dinámicas y un comportamiento general, similar a los patrones de comportamiento. Los *widgets* son microprogramas que cumplen una función predeterminada y que a la vez sus propiedades pueden ser modificadas para expresar comportamientos personalizados por el usuario. Una vez modificadas las propiedades del *widget*, éste es introducido dentro de la aplicación para cumplir una función en particular.

**3-Principales enfoques de desarrollo Web basado en modelos para las RIA**

En [] y [], se identifica la necesidad de metodologías sistemáticas para el desarrollo de las RIA y se llevan a cabo estudios presentando las diversas metodologías Web existentes para ese fin. El estudio más exhaustivo y reciente de comparativas se presenta en [].

Las metodologías tenidas en este análisis, son las que contribuyen a la investigación que proviene de la comunidad de ingeniería Web, y derivan de la evolución de los enfoques dirigidos por modelos, concebidos para el diseño y desarrollo de aplicaciones Web tradicionales en las que se incluyen a WebML-RIA[], OOHDM -RIA[], OOH4RIA[] , (UWE-R[]Patrones RIA con UWE[] y UWE+RUX []).

La Tabla 1 que se muestra un resumen de las propuestas, indicando las características RIA contempladas por las mismas.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Características versus metodologías Web | | OOHDM-RIA | OOH4RIA | WebML - RIA | Patrones RIA con UWE | UWE-R | UWE + RUX |
| Almacenamiento en el lado del cliente | | - |  | si | - | - | - |
| Lógica de negocio en el lado del cliente | Operaciones complejas | - | - | si | - | - | - |
| Operaciones específicas del dominio | - | - | - | - | si | - |
| Validación local | si | si | si | - | - | si |
| Presentaciones enriquecidas | Manejo de eventos en el lado cliente | - | - | si | si | si | si |
| Widgets | si | si | - | si | si | si |
| Paradigma de página única | si | si | si | si | - | si |
| Contenido multimedia | - | si | - | - | si | si |
| Comunicación cliente servidor | Sincronización de datos | - | - | si | - | si | si |
| Obtención de actualizaciones parciales de página | si | si | si | si | si | si |
| Push y Pull | - | - | si | - | si | - |

Tabla 1 Metodologías Web y sus alcances para RIA

**4-La Aproximación MoWebA (Model Oriented Web Approach)**

MoWebA[][11] es una propuesta creada en el DEI (Departamento de Electrónica e Informática) que adopta los principios de MDA. En la Figura 3 se muestran las dimensiones de MoWebA. Como puede observarse, consta de fases, niveles y aspectos, que se van describiendo a continuación.

Las fases se refieren a los procesos de modelado y transformación. Estas se encuentran claramente diferenciadas e incluyen a su vez una serie de modelos entre los que se citan:

a.  **Modelado del problema:** en el que se incluyen al CIM (*Computation Independent Model*), orientado al modelado de los requisitos funcionales*,* y al PIM (*Platform Independent Model*) orientado al modelado del problema sin considerar aspectos de la arquitectura o plataforma. De aquí es posible llevar a cabo transformaciones para obtener los modelos específicos de la plataforma de manera semi-automática por medio de reglas.

b.   **Modelado de la solución**: en donde forman parte el ASM (*Architectural Specific Model*) y el PSM (*Platform Specific Mode*l). Es en esta fase, en donde todos los detalles de la arquitectura y plataforma destino se definen, permitiendo generar a partir de aquí, el código de la aplicación de manera automática. En MoWebA se independiza esta fase, y esto hace que sea bastante prometedora para la implementación de las RIA, debido a que existen numerosas plataformas destino para desplegarlas. En las aproximaciones estudiadas, por lo general las extensiones RIA son definidas en el marco de los modelos conceptuales (PIMs), haciendo que los modelos que deberían ser independientes de la solución, adquieran elementos que ya son propios de una arquitectura específica.



Figura 3 Niveles y fases en el desarrollo de MOWEBA

c. **Código fuente:** incluye al ISM (*Implementation specific model*) que corresponde al código generado y el código manual a ser agregado (en caso de ser necesario) para generar la aplicación final. La aplicación puede refinarse, dado que todas las fases son iterativas e incrementales.

MoWebA también presenta distintos niveles de construcción separados en capas para representar a una aplicación Web. Se contemplan niveles para el contenido, la lógica del negocio, la navegación, la presentación, y los usuarios. Los aspectos están relacionados con la estructura y el comportamiento de la aplicación. Cada modelo es visto desde dos puntos de vista (estructura y comportamiento) por lo que existe una propuesta notacional para definirlos.

Definir una propuesta RIA para MoWebA resulta interesante ya que sería posible realizar un análisis para diferenciar el PIM del ASM, no contemplados en otras metodologías. Esto hace que al definir los modelos propios de las RIA, si hubiera necesidad de llevar a cabo una migración a otra arquitectura destino, probablemente deberán realizar muchos cambios sobre el modelo mismo. En MoWebA se plantea tener siempre el mismo PIM, y a partir de este adoptar la arquitectura correspondiente.

**4.1- El metamodelo de Contenido extendido para MoWebA .**

El metamodelo de Contenido (*Content*) con extensiones RIA junto al metamodelo de Estructura (*Layout*) de MoWebA se pueden apreciar en la Figura 5. En el metamodelo de Contenido se presentan los diversos elementos que permiten representar una interfaz de usuario enriquecida Como nuevos aportes al metamodelo de Contenido de MoWebA, se propuso la clasificación de los diferentes elementos de interfaz, en elementos de entrada, salida y control respectivamente. Esto fue necesario para establecer un orden dentro de los elementos de interfaz y para una mayor claridad dentro del metamodelo de Contenido. Los distintos elementos de interfaz de usuario se clasifican en:

* ***Elementos de salida* (*OutputElements*):** Comprende a los elementos de interfaz enriquecidos y tradicionales encargados de desplegar o mostrar información en las páginas de presentación. En esta categoría se engloba a los elementos *text*, *htmlText,* *multimedia* y *richToolTip.*
* **Elementos de entrada (*imputElements*):** Comprende a los elementos de interfaz enriquecidos y tradicionales encargadas de obtener una entrada desde la interfaz de usuario. En esta categoría, se engloban a los elementos *textInputs*, *list*, *richAutoSuggest*, *richDatePicker* y *richFieldLiveValidation*).
* ***Elementos de control* (*controlElements*)*:*** Comprende a los elementos de interfaz tradicionales encargados de obtener una orden de navegación o cambio de página. En esta categoría se engloban a los elementos *externalLink*, *anchor* y *button*.

También, como parte de las extensiones, se encuentran los *richAccordion* y los *richTabs*, que pueden tener uno o varios paneles (*Panels).* Cada uno de estos *Panels*, permite aglomerar a muchos elementos de interfaz de entrada, salida o control*.* El atributo *defaultPanelNumber*, permite establecer cuál de los *Panels* existentes en un *richAccordion* o *richTabs*, se desplegará primeramente por omisión.

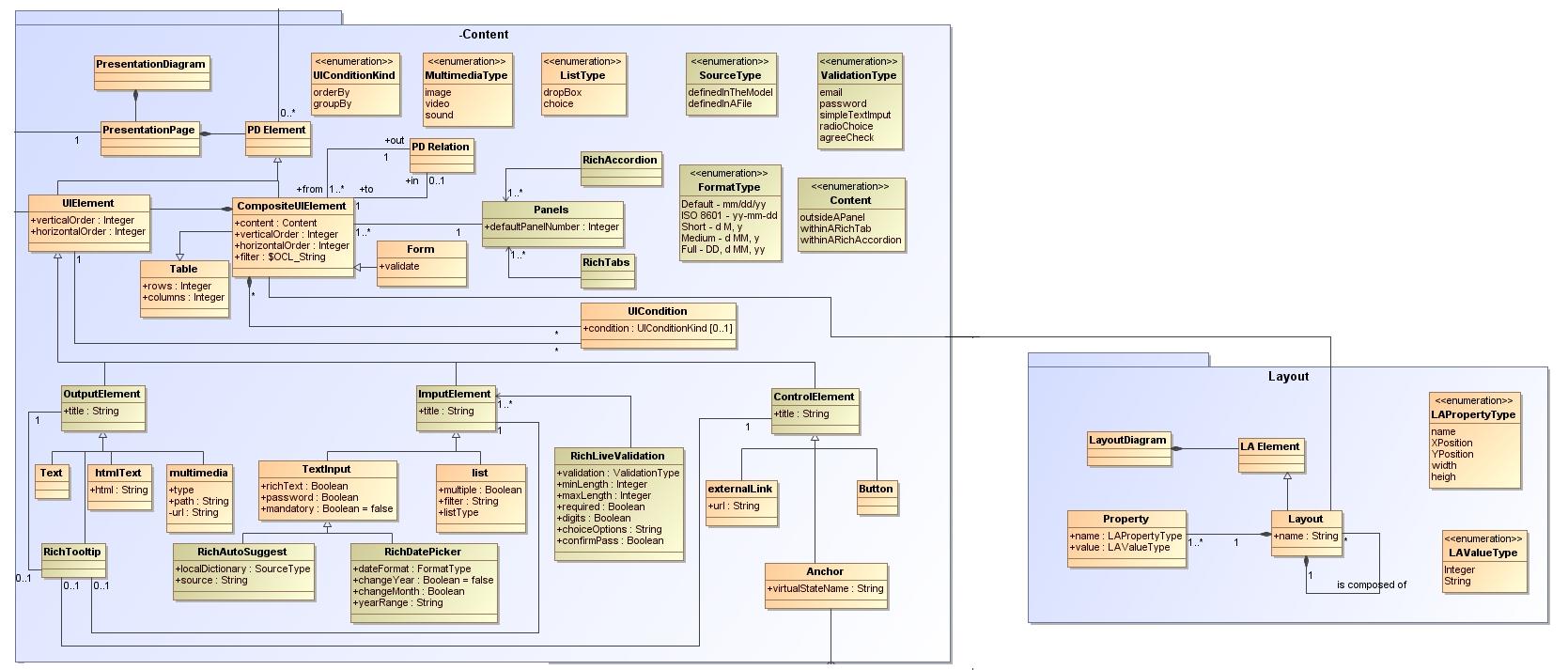


Figura 5 Metamodelo de contenido y estructura

A continuación se describen cada uno de los elementos que forman parte de la extensión al metamodelo de contenido de MoWebA.

**4.1.1 RichAutoSuggest**

El *widget* se presenta en la Figura 6. Este elemento de interfaz enriquecido, contiene los valores etiquetados *localDictionary* y *source*. El primero corresponde a una enumeración que permite definir al diccionario de sugerencias, que consiste en una lista en la cual están definidas todas las palabras sugeridas del *widget* *richAutoSuggest*, en un archivo en formato .*xml* o bien definir el listado de sugerencias en el mismo diagrama de clases en un formato de cadenas de sugerencias separado por comas. El valor etiquetado *source* será completado de acuerdo a lo definido en la enumeración anterior. En el caso de estar definido con el valor *definedInTheModel*, entonces *source* que es del tipo *string*, contendrá un conjunto de palabras correspondiente a las sugerencias separadas entre comas. Si en la enumeración está definido el valor *definedInAFile*, entonces el diccionario de palabras está definido en un archivo xml y en valor etiquetado *source* se colocará la ruta en la cual se encuentra el archivo de palabras.

Figura 6 El *widget* *RichAutoSuggest*

**4.1.2 RichDatePicker**

El *widget* se presenta en la Figura 7. Este elemento de interfaz enriquecido contiene a los valores etiquetados *dateFormat*, *changeYear*, *changeMonth* y *yearRange*. El *dateFormat* corresponde a una enumeración que contiene cinco formatos de fecha distintos que pueden ser:

\**Default - mm/dd/yy* por ejemplo 06/08/2015

\* *ISO 8601 - yy-mm-dd* por ejemplo 2015-06-08

\* *Short - d M, y* por ejemplo 8 Jun, 15

\**Medium - d MM, y por ejemplo 8 June, 15*

\* *Full - DD, d MM, yy por ejemplo Monday, 8 June, 2015*.

Figura 7 El *widget RichDatePicker*

El valor etiquetado *changeYear* es un valor booleano que indica la ausencia o presencia de un rango de años desplegable en una lista que formará parte del *richdatePicker.* Por omisión, si *changeYear* está configurado en verdadero, se mostrará en el *datePIcker* una lista desplegable presentando los diez años anteriores a partir de la fecha actual. También es posible asignar al valor etiquetado *yearRange* un rango de años para el *richDatePicker* que se define en el formato yyyy:yyyy; por ejemplo 1970:2015. Definir *yearRange* resulta ideal para la selección de fechas pasadas, como el año de nacimiento o fechas históricas.

Por último, el valor etiquetado booleano *changeMonth*  permite desplegar una lista con todos los meses del año para una rápida selección.

**4.1.3 RichToolTip**

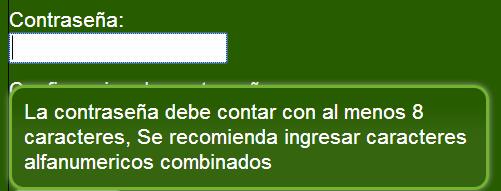
El *widget* se presenta en la Figura 8, en donde el mensaje “La contraseña debe contar con al menos 8 caracteres...” es desplegado al posicionar el puntero del mouse sobre el *textiImput* Contraseña. Este elemento de salida, tiene como objetivo enriquecer con mensajes personalizados a cualquiera de los elementos de entrada, salida y control. El *richToolTip* no cuenta con valores etiquetados. Al definirse este elemento en conjunción con algunos de los elementos simples de entrada, salida o de control, implica que un mensaje emergente será desplegado cuando el puntero del mouse se posicione sobre el elemento. Cada uno de los elementos de entrada, salida y control posee el valor etiquetado *title*, que corresponde al mensaje que será desplegado

Figura 8 El *widget RichToolTip*

**4.1.4 RichFieldLiveValidation**

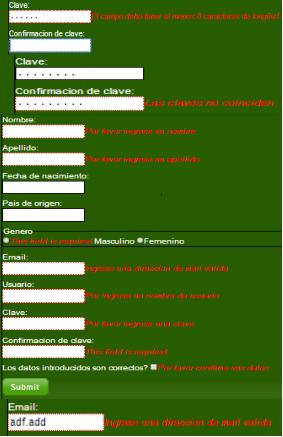
El *widget* se presenta en la Figura 9. Este elemento de interfaz enriquecido, permite llevar a cabo validaciones locales a diversos elementos pertenecientes a un formulario como se muestra en la Figura 9. El valor etiquetado *validation* corresponde a una enumeración que contiene todas las validaciones que son posibles de llevar a cabo. Estas validaciones pueden ser del tipo *email*, *password*, *simpleTextInput*, *radioChoice* y *agreeCheck*. Para las validaciones del tipo *email*, *password* y *simpleTextInput* es posible establecer la cantidad mínima de caracteres en un campo de entrada estableciendo el valor etiquetado *minLength*. Similarmente el valor etiquetado *maxLength* permite establecer la cantidad máxima de caracteres que es posible ingresar en un campo de entrada. Para el *simpleTextInput,*  el valor etiquetado *digits* establece que el campo de entrada debe tener estrictamente valores del cero al nueve. Las validaciones *email*, *password* y *simpleTextInput*, pueden configurarse como mandatorias, lo que implica que no pueden quedar vacías, estableciendo el valor etiquetado booleano *required* como verdadero. Al seleccionar el tipo de validación *password*, es posible establecer el valor booleano del valor etiquetado *confirmPass* en verdadero, para el caso en el que se necesite crear otro campo de entrada para la confirmación de contraseña.

Figura 9 El widget para validaciones locales *RichFieldLiveValidation*

Por otro lado, la validación del tipo *radioChoice*, permite establecer de un listado de opciones de selección, un valor obligatorio, que es establecido en el valor etiquetado *choiceOptions*, como cadenas separadas por comas. De todas las opciones existentes en *choiceOptions*, una de ellas se establece con la palabra reservada *Required*, separada con un espacio en blanco para indicar que esa opción por omisión es obligatoria y debe seleccionarse al menos esa opción. De igual forma, si se elige otra opción que no tenga la palabra reservada *Required* establecida, el campo será validado.

Finalmente, se encuentra el tipo de validación *agreeCheck*, que una vez seleccionado, permite desplegar un cuadro de selección del tipo *radio*, el cual es obligatorio y debe ser seleccionado en un formulario para que este pueda ser validado. Este campo es idóneo para acuerdos de conformidad.

El *richFieldLiveValidation* es activado en el momento en el que se establece como verdadero o el valor etiquetado booleano *valídate* dentro del elemento de interfaz *form*,que indica si el formulario en cuestión tendrá o no algún tipo de validación de campo.

**4.1.5 RichAccordion**

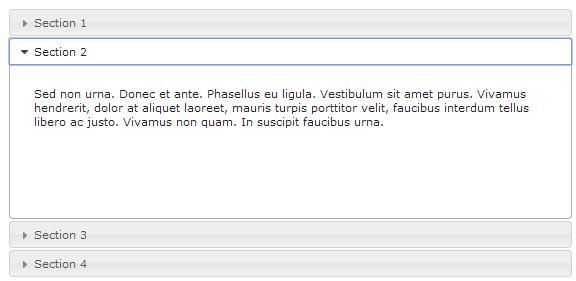
El *widget* se presenta en la Figura 10. Este elemento de interfaz enriquecido permite encapsular a varios elementos de interfaz dentro de sus paneles. Los elementos que pueden ser desplegados son los *compositeUIElements*, *table* y los *form*. Para establecer que un elemento de interfaz pertenece a un panel de un *richAccordion*, se debe seleccionar de la enumeración *content* que se encuentra dentro de un *compositeUIElement*, el valor *withinARichAccordion*. El valor por omisión establecido para la enumeración *content* es *independent* que especifica que el contenido no forma parte de un *ni de un richTabs*.

Figura 10 El *widget* *RichAccordion*

**4.1.6 RichTabs**

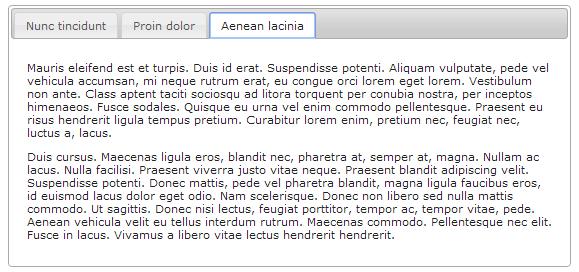
El *widget* se presenta en la Figura 11. Este elemento de interfaz enriquecido permite encapsular a varios elementos de interfaz dentro de sus paneles. Los elementos que pueden ser desplegados son los *compositeUIElements*, *table* y los *form*. Para establecer que un elemento de interfaz pertenece a un panel de un *richTabs*, se debe seleccionar de la enumeración *content* que se encuentra dentro de un *compositeUIElement*, el valor *withinARichTabs*. El valor por omisión establecido para la enumeración *content* es *independent* que especifica que el contenido no forma parte de un *richAccordion ni de un richTabs*.

Figura 11 El *widget* *RichTabs*

De los metamodelos de contenido y estructura presentados, se derivan los perfiles, que son extensiones al lenguaje UML, para agregar las características propias de MoWebA y se esa forma, hacer posible la representación de la sintaxis concreta de MoWebA. Con los perfiles extendidos, se modelan los PIM de las aplicaciones

**4.2- El enfoque utilizado con MoWebA para la generación de interfaces enriquecidas**

El objetivo de este trabajo de fin de carrera es agregar nuevos elementos al metamodelo original de contenido de MoWebA, precisamente elementos de interfaz propia de las RIA, que son los *widgets* interactivos y la validación en el lado del cliente en los formularios. Estos nuevos elementos serán modelados en primera instancia y luego traducidos a código por medio de una transformación M2T utilizando la herramienta Acceleo[[6]](#footnote-6).

Primeramente se modelan los PIM que representan a una aplicación en particular utilizando distintos perfiles UML de MoWebA. Estos perfiles representan extensiones a UML para agregar características específicas de MoWebA a los metamodelos, para que de esta forma sea posible representar la sintaxis concreta del DSL (*Domain Specific Lenguage*). Los modelos PIM y los perfiles están basados en el estándar MOF que forma parte del enfoque MDA. Los modelos PIM se crean utilizando la herramienta MagicDraw[[7]](#footnote-7). Posteriormente tanto los PIM como los perfiles son exportados al formato XMI del EMF[[8]](#footnote-8). Esto de por sí es llevado a cabo a fines de tener compatibilidad con la herramienta de transformación M2T Acceleo[[9]](#footnote-9), que toma como entrada modelos UML basados en el metamodelo Ecore[[10]](#footnote-10).

Una vez exportados los modelos (PIM y profile) al Acceleo, por medio de las plantillas de transformación y los módulos de servicio en Java (*Java Service Wrappers*), que forman parte de Acceleo, es posible llevar a cabo las transformaciones necesarias sobre los modelos de entrada para obtener los archivos fuentes (.html y .css) que representan a la aplicación en sí. Las plantillas de transformación, permiten establecer la estructura del código fuente que va a generarse, estableciendo las partes estáticas (código que va a generarse en ciertas condiciones y que no cambia) y dinámicas (código que es obtenido a partir de los modelos de entrada). Por medio de los metamarcadores de las plantillas de transformación de Acceleo, es posible definir expresiones OCL para la manipulación de los distintos elementos definidos en el modelo de entrada. Los módulos de servicio *Java*, permiten complementar a las plantillas de transformación, dando la posibilidad de agregar código *Java* para la manipulación de los elementos pertenecientes a los modelos.

En la Figura 4 se presenta el proceso para el modelado y generación de interfaces enriquecidas (también conocidos como los *front-ends* de las aplicaciones). El perfil de contenido de MoWebA, ha sido extendido con nuevos elementos de interfaz de usuario que son típicos de las RIA. También las plantillas de transformación, han sido adaptadas para generar el código correspondiente a cada uno de los nuevos elementos de interfaz RIA que han sido agregados.

A diferencia del enfoque de presentación de MoWebA en su forma original, que genera código HTML para los elementos de interfaz de la Web 1.0 y CSS para la estructura de cada uno de los elementos dentro de las páginas, en MoWebA extendido, se genera código HTML y *Javascript* para la plataforma *jQueryUI,* específicamente el código para los *widgets* *RichAccordion*, *RichTabs*, *RichDatePicker*, *RichTooltip*, y *RichAutoSuggest* y *jQuery Validation plug-in* para los diversos tipos de validación local. De igual manera, en su forma original, es posible generar el código CSS para estructurar cada uno de los elementos de interfaz enriquecidos (o no). Finalmente las librerías *Javascript jQueryUI* y *jQuery Validation Plugin* se invocan desde el código fuente generado para tener todas las funcionalidades enriquecidas de la aplicación a partir del código generado.



Figura 4 Fases de desarrollo para la propuesta de extensión a MoWebA

**5-iLUSTRACION DE UN CASO DE ESTUDIO**

Con la intensión de realizar un análisis de la propuesta se ha procedido a desarrollar una ilustración de un caso de estudio. Con este trabajo, se ha logrado recabar datos que permiten intuir que la propuesta de extensión presentada ofrece cobertura a algunas de las diversas características que contemplan las RIA que han sido analizadas. Puntualmente, estas características abarcan a la lógica de negocios en el lado del cliente, específicamente a las validaciones locales de campos en un formulario, y a las presentaciones enriquecidas que contemplan a ciertos eventos en el lado del cliente, *widgets* interactivos y el paradigma de una sola página o *single page paradigm*. El objetivo es analizar estas características por medio de la resolución de un *toy problem* denominado *Person Manager*. El *Person Manager* es una aplicación Web que contiene en sus especificaciones funcionales características de las RIA y resulta lo suficientemente expresiva para ilustrar la propuesta de extensión.

**52.3 El caso y las unidades de análisis**

El caso ilustrativo consistió en un sistema de administración de personas (*Person Manager*) en el dominio de las aplicaciones Web, que fue elegido entre varias otras opciones debido a que sus requerimientos funcionales ofrecen la posibilidad de representar a todas las características RIA que han sido agregadas al enfoque MoWebA, de una manera clara y sencilla.

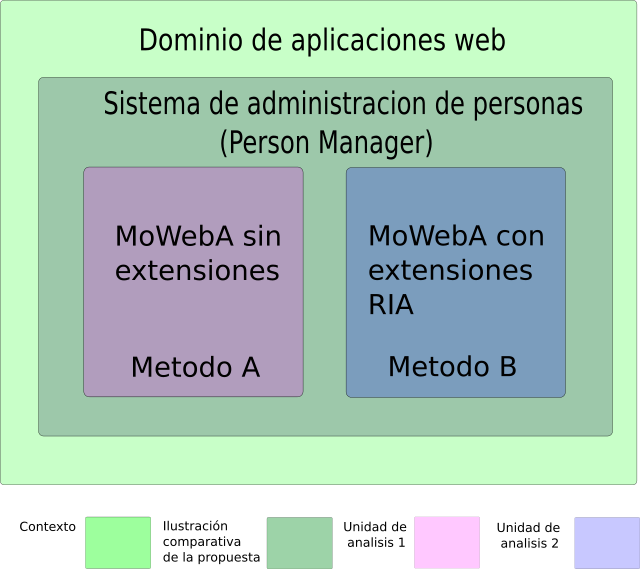


Figura 12 Ilustración del sistema *Person Manager* implementado con MoWebA desde dos enfoques distintos

El caso fue analizado desde dos unidades de análisis. La primera unidad de análisis se refiere a la implementación de la capa de presentación del *Person Manager* con MoWebA sin extensiones RIA (*método A*). La segunda unidad de análisis se refiere a la implementación de la misma capa de presentación del caso estudiado, con la nueva propuesta de extensión RIA a MoWebA (*método B*). Por lo tanto, se trata de una ilustración embebida, ya que se cuenta con más de una unidad de análisis para un mismo caso, como se puede apreciar en la Figura 12. El *Person Manager* está basado en el trabajo de Gharavi []

**6-Conclusiones y trabajos futuros**

En este trabajo de fin de de carrera se llevó a cabo una investigación acerca de las RIA, analizando sus características y las tecnologías utilizadas para su implementación. Se ha visto que dentro de las características más relevantes, se encuentra el uso del lado del cliente para mejorar la interactividad de las aplicaciones para con los usuarios y de esa forma, obtener aplicaciones Web similares a una aplicación de escritorio, con pocos retardos en las recargas de página. Dentro de las tecnologías RIA existentes, las basadas en librerías *Javascript* son las más populares y la más utilizada actualmente es *jQuery*. Con las extensiones a *jQuery*, *jQueryUI* y *JQuery validation plug-in* es posible explotar el lado del cliente agregando *widgets* interactivos como los *richAccordion*, *richTabs, richAutoSuggest, richToolTip, richDatePicker* y llevar a cabo validaciones locales de campos con el *richFieldLiveValidation*.

Hoy en día, la ingeniería de software basada en modelos, cumple un papel importante en el desarrollo del software. He allí que existen muchas metodologías Web con este enfoque y que dan cierta cobertura a características de las RIA. En base a una investigación del estado del arte de las principales metodologías Web basadas en modelos, se concluyó que ninguna de ellas logra dar cobertura a todas las características RIA. He allí la necesidad de extender a las metodologías Web existentes con nuevas características RIA o bien proponer nuevas metodologías que ofrezcan una mayor flexibilidad de extensión. En este trabajo se propuso a la aproximación MoWebA que está basada en los estándares propuestos por la OMG, como la alternativa apropiada, debido a su adecuada separación en capas y baja cohesión, que le brinda la posibilidad de llevar a cabo extensiones RIA a cualquiera de las capas sin afectar a las otras. En este trabajo se extendió con características RIA a la capa de Presentación de MoWebA que abarca a los metamodelos de Contenido y Estructura.

Precisamente con este trabajo de fin de carrera se alcanzó aportar lo siguiente:

* Un estudio detallado de las principales características y tecnologías de las RIA
* Una investigación del estado del arte de las principales metodologías Web basadas en MDD y MDA que ofrecen cobertura a las RIA.
* Un análisis de los elementos de interfaz enriquecidos (*widgets*) más utilizados.
* Una extensión al metamodelo de contenido de MoWebA agregando los siguiente:
  + Una reestructuración y clasificación de los elementos de interfaz con los que cuenta la aproximación MoWebA, separando a los distintos componentes de interfaz en elementos de entrada, salida y control.
  + *Widgets interactivos* comunes en las aplicaciones RIA, precisamente richAccordion, richTabs, richAutoSuggest, richDatePicker y richToolTip y el richFieldLiveValidation.
* Para la definición de la sintaxis concreta de la presentación, se agregaron los nuevos widgets al perfil de Contenido de MoWebA.
* Un análisis de las principales herramientas de transformación de modelo a texto (M2T) basado en plantillas.
* Con la herramienta de transformación (M2T) *Acceleo* se implementaron las siguientes plantillas:
  + **La plantilla de presentación**, la cual genera código para cada uno de los elementos definidos en el perfil de contenido de MoWebA a partir de los PIM de entrada. Para los *widgets* se genera código para la plataforma destino *jQueryUI* y *jQuery validation plug-in*.
  + **La plantilla de estructura**, la cual genera código CSS con las posiciones establecidas en el PIM de entrada
* Una ilustración evaluativa en la cual se presentan los aportes realizados a la capa de presentación de MoWebA.

**Bibliografía**

[1] Ginige A and Murugesan S. Guest editors’ introduction: The essence of web engineering— managing the diversity and complexity of web application development. *IEEE MultiMedia*, 8(2):22–25, Apr 2001.

[2] Martínez-Ruiz F J. *A Development Method for User Interfaces of Rich Internet Applications*. PhD thesis, Polytechnic University of Valencia, Spain, August 2010.

[3] Toffetti G, Comai S, Preciado J C, and Linaje M. State-of-the art and trends in the systematic development of rich internet applications. *J. Web Eng.*, 10(1):70–86, March 2011.

[4] Allaire J. Requirements for rich internet applications. http://download.macromedia.com/pub/flash/whitepapers/richclient.pdf, March 2002.

[5] Wright J and Dietrich J. Survey of existing languages to model interactive web applications. In *Proceedings of the fifth Asia-Pacific conference on Conceptual Modelling - Volume 79*, APCCM ’08, pages 113–123, Darlinghurst, Australia, Australia, 2008. Australian Computer Society, Inc.

[6] Preciado J C, Linaje M, Sanchez F, and Comai S. Necessity of methodologies to model rich internet applications. In *Proceedings of the Seventh IEEE International Symposium on Web Site Evolution*, WSE ’05, pages 7–13, Washington, DC, USA, 2005. IEEE Computer Society.

[7] Preciado J C, Linaje M, Morales-Chaparro R, Sanchez-Figueroa F, Zhang G, Kroi$β$ C, and Koch N. Designing rich internet applications combining uwe and rux-method. In *Proceedings of the 2008 Eighth International Conference on Web Engineering*, ICWE ’08, pages 148–154, Washington, DC, USA, 2008. IEEE Computer Society.

[8] Machado L, Filho O, and Ribeiro J. Uwe-r: an extension to a web engineering methodology for rich internet applications. *WSEAS Trans. Info. Sci. and App.*, 6(4):601–610, April 2009.

[9] Busch M and Koch N. Rich internet applications state-of-the-art. Technical report 0902, Programming and Software Engineering Unit (PST), Institute for Informatics, Ludwig-Maximilians-Universität München, Germany, December 2009.

[10] González M, Casariego J, Bareiro J, Cernuzzi L, and Pastor O. Una propuesta mda para las perspectivas navegacional y de usuarios. In *XXXVI Conferencia Latinoamericana de Informática (CLEI) - ISBN 978-99967-612-0-1*, page 58, Asunción, Paraguay, 2010.

[11] González M, Cernuzzi L, and Pastor O. Una aproximación para aplicaciones web: Moweba. In *XIV Congreso Iberoamericano en Software Engineering – CibSE*, Río de Janeiro, Brasil, 2011.

[12] Urbieta M, Rossi G, Ginzburg J, and D. Schwabe. Designing the interface of rich internet applications. In *Proceedings of the 2007 Latin American Web Conference*, LA-WEB ’07, pages 144–153, Washington, DC, USA, 2007. IEEE Computer Society.

[13] Koch N, Pigerl M, Zhang G, and Morozova T. Patterns for the model-based development of rias. In *Proceedings of the 9th International Conference on Web Engineering*, ICWE ’9, pages 283–291, Berlin, Heidelberg, 2009. Springer-Verlag.

[14] Fraternali P, Comai S, Bozzon A, and Carughi G T. Engineering rich internet applications with a model-driven approach. *ACM Trans. Web*, 4(2):7:1–7:47, April 2010.

[15] Meliá S, Gómez J, Pérez S, and Dáz O. A model-driven development for gwt-based rich internet applications with ooh4ria. In *Proceedings of the 2008 Eighth International Conference on Web Engineering*, ICWE ’08, pages 13–23, Washington, DC, USA, 2008. IEEE Computer Society.

[16] Vahid Gharavi S V. Model-driven development of ajax web applications. Master’s thesis, Faculty EEMCS, Delft University of Technology, September 2008.

1. **Object Management Group:** <http://www.omg.org/> 2015 [↑](#footnote-ref-1)
2. Departamento de Ingeniería Electrónica e Informática de la Universidad Católica Nuestra Señora de la Asunción [↑](#footnote-ref-2)
3. **Model Driven Architecture:** <http://www.omg.org/mda/> 2015 [↑](#footnote-ref-3)
4. **jQuery UI 1.11 API Documentation:** <http://api.jqueryui.com/> 2015 [↑](#footnote-ref-4)
5. **jQuery Validation Plugin:** <http://jqueryvalidation.org/> 2015 [↑](#footnote-ref-5)
6. **Acceleo:** [www.acceleo.org/](http://www.acceleo.org/) 2015 [↑](#footnote-ref-6)
7. **No Magic:** <http://www.nomagic.com/products/magicdraw.html> 2015 [↑](#footnote-ref-7)
8. **Eclipse Modelling Framwwork:** <https://www.eclipse.org/modeling/emf> 2015 [↑](#footnote-ref-8)
9. **Acceleo:** <https://eclipse.org/acceleo> 2015 [↑](#footnote-ref-9)
10. **Ecore:** Metamodelo nativo que forma parte del core del EMF para describir a los modelos [↑](#footnote-ref-10)