UNIVERSIDAD DE EXTREMADURA

Escuela Politécnica

Grado en Ingeniería Informática – Ingeniería del Software

Trabajo Fin de Grado

Una propuesta de automatización de operaciones CRUD en IFML

Javier Sierra Blázquez

Septiembre, 2017

UNIVERSIDAD DE EXTREMADURA

Escuela Politécnica

Grado en Ingeniería Informática – Ingeniería del Software

Trabajo Fin de Grado

Una propuesta de automatización de operaciones CRUD en IFML

Autor: Javier Sierra Blázquez

Tutor: Juan Carlos Preciado Rodríguez

Co-Tutor: Roberto Rodríguez Echeverría

Tribunal Calificador

Presidente:

Secretario:

Vocal:

Agradecimientos

Agradezco a mi tutor, Juan Carlos, y a mi co-tutor, Roberto, la confianza depositada en mi desde el inicio del proyecto. Gracias por los consejos, por guiarme hasta el final de esta etapa y por utilizar un pedazo de vuestro preciado tiempo para ayudarme.

Agradecer también al grupo de compañeros y amigos, conocidos en esta etapa universitaria, por hacer más fácil y divertido el día a día fuera de casa, ayudándome a progresar como persona.

Por último y con mayor énfasis, espero que lo entiendan, GRACIAS a mis queridísimas madre y hermana por estar siempre ahí y ayudarme a superar todas las dificultades de la vida.

Sin el apoyo de todas estas personas no hubiera sido posible la experiencia vital tan gratificante como son los estudios universitarios.

Gracias.

ÍNDICE DE TABLAS 8
ÍNDICE DE FIGURAS 10
RESUMEN 13
1 Introducción 14
1.1 Motivación 15
1.2 Antecedentes 15
1.3 Objetivos 16
2 Estado del arte 19
2.1 Ingeniería Web 19
2.2 Ingeniería Web dirigida por Modelos 20
2.3 WebML 21
2.4 IFML y WebRatio 21
3 Metodología 22
3.1 Fases 22
3.2 Diagrama de Gantt 24
4 Punto de partida 25
5 Diseño de la solución 30
5.1 Patrones 30
5.1.1 Estructura interna 30
5.1.2 Elementos 34
5.1.3 Unicidad de elementos con relación 48
5.2 Analizador del patrón 50
5.2.1 Validación: XML Schema 51
5.2.1.1 schema.xsd 52
5.2.2 Interpretación y serialización de la información 59
5.3 Generación 61
5.4 Registro 61
5.5 Interfaz de usuario 64
5.5.1 Diálogo principal 64
5.5.2 Ventana del registro 66
5.5.3 Diálogo de generación 68
6 Implementación y desarrollo 69
6.1 Análisis y Arquitectura de la aplicación 69
6.1.1 Diagramas de flujo 69

6.1.1.1 Proceso 1: Carga y preselección de entidades y patrones 71
6.1.1.2 Proceso 2: Validación y análisis del patrón 72
6.1.1.3 Proceso 3: Visualizar registro 73
6.1.1.4 Proceso 4: Generación y registro de elementos 74
6.1.2 Clases 76
6.1.2.1 Modificación de clases del proyecto anterior 76
6.1.2.2 Clases utilizadas en la solución final 78
6.1.2.3 Diagramas de clases 83
7 Resultados 90
8 Conclusiones 93
8.1 Dificultades encontradas 93
8.2 Líneas de desarrollo futuro 94
REFERENCIAS BIBLIOGRÁFICAS 95
ANEXOS 97
1 ANEXO I: Manual de usuario 97
1.1 Instalación del plug-in 97
1.2 Localización de los archivos 98
1.3 Patrones de ejemplo 99
1.3.1 Create (default).xml 99
1.3.2 Read (default).xml 104
1.3.3 Update (default).xml 106
1.3.4 Delete (default).xml 112
1.3.5 AllInOne (default).xml 114
1.4 Ejemplo de ejecución del asistente 123
2 ANEXO II: Manual del Programador 132
2.1 Exportación del plug-in 132
2.2 Depuración en el desarrollo del plug-in 134
2.3 Ejemplo del archivo Registry 137
3 ANEXO III: Actas de reuniones 140
3.1 Reunión del día 31 de marzo 140
3.2 Reunión del día 21 de abril 141
3.3 Reunión del día 4 de mayo 142
3.4 Reunión del día 2 de junio 143
3.5 Reunión del día 3 de julio 144

3.6 Reunión del día 20 de julio 145
3.7 Cronología de reuniones..... 146
4 ANEXO IV: Contenido del CD..... 147

ÍNDICE DE TABLAS

Tabla 1 – Objetivo 1: Descripción externa de patrones 16
Tabla 2 - Objetivo 2: Simplificar la interacción..... 17
Tabla 3 - Objetivo 3: Registro 17
Tabla 4 - Objetivo 4: Generación paso a paso 18
Tabla 5 - Objetivo 5: Refactorización del código fuente 18
Tabla 6 - Objetivo 6: Identificación de patrones existentes 18
Tabla 7 - Elementos generables..... 31
Tabla 8 - Atributos comunes..... 35
Tabla 9 - Elemento page 36
Tabla 10 - Elemento xor 36
Tabla 11 - Elemento powerIndexUnit 37
Tabla 12 - Elemento dataUnit..... 37
Tabla 13 - Elemento multiMessageUnit..... 38
Tabla 14 - Elemento noOpContentUnit 38
Tabla 15 - Elemento entryUnit..... 39
Tabla 16 - Elemento createUnit..... 39
Tabla 17 - Elemento deleteUnit..... 40
Tabla 18 - Elemento updateUnit..... 40
Tabla 19 - Elemento connectUnit 41
Tabla 20 - Elemento disconnectUnit..... 41
Tabla 21 - Elemento reconnectUnit 42
Tabla 22 - Elemento normalNavigationFlow..... 43
Tabla 23 - Elemento dataFlow..... 45
Tabla 24 - Elemento okLink..... 45
Tabla 25 - Elemento koLink..... 46
Tabla 26 - Elemento isNotNullUnit 47
Tabla 27 - Elemento selectorUnit 47
Tabla 28 - Ejemplos Unicidad de elementos con relación 49
Tabla 29 - Leyenda de objetivos 90
Tabla 30 – Grado de consecución del Objetivo 1: Descripción externa de patrones 90

Tabla 31 - Grado de consecución del Objetivo 2: Simplificar la interacción 91
Tabla 32 - Grado de consecución del Objetivo 3: Registro 91
Tabla 33 - Grado de consecución del Objetivo 4: Generación paso a paso 92
Tabla 34 - Grado de consecución del Objetivo 5: Refactorización del código fuente 92
Tabla 35 - Grado de consecución del Objetivo 6: Identificación de patrones existentes 92

ÍNDICE DE FIGURAS

Ilustración 1: Diagrama de Gantt 24
Ilustración 2: Wizard CRUD Assistant v2. Página de selección de entidad 25
Ilustración 3: Wizard CRUD Assistant v2. Página de creación de SiteView/Area 26
Ilustración 4: Wizard CRUD Assistant v2. Página de selección de patrón. Pestaña AllInOne 26
Ilustración 5: Wizard CRUD Assistant v2. Página de selección de patrón. Pestaña Create 27
Ilustración 6: Wizard CRUD Assistant v2. Página de selección de patrón. Pestaña Read 27
Ilustración 7: Wizard CRUD Assistant v2. Página de selección de patrón. Pestaña Update 28
Ilustración 8: Wizard CRUD Assistant v2. Página de selección de patrón. Pestaña Delete 28
Ilustración 9: Estructura interna del patrón 33
Ilustración 10: Proceso de generación (simple) 50
Ilustración 11: Diagrama de flujo. Procesamiento del patrón: sección de relaciones 60
Ilustración 12: Diagrama de la estructura del registro 62
Ilustración 13: Interfaz de usuario. Diálogo principal 65
Ilustración 14: Interfaz de usuario. Ventana del registro 67
Ilustración 15: Interfaz de usuario. Diálogo de generación 68
Ilustración 16: Diagrama de flujo. Vista completa del flujo de la aplicación 70
Ilustración 17: Diagrama de flujo. Proceso 1: Carga y preselección de entidades y patrones 71
Ilustración 18: Diagrama de flujo. Proceso 2: Validación y análisis del patrón 72
Ilustración 19: Diagrama de flujo. Proceso 3: Visualizar registro 73
Ilustración 20: Diagrama de flujo. Proceso 4: Generación y registro de elementos 75

Ilustración 21: Diagrama de paquetes 83
Ilustración 22: Diagrama de clases. Paquete principal 83
Ilustración 23: Diagrama de clases. Jerarquía del paquete element . 84
Ilustración 24: Diagrama de clases. Paquete exceptions 85
Ilustración 25: Diagrama de clases. Paquete generation 85
Ilustración 26: Diagrama de clases. Paquete parser 86
Ilustración 27: Diagrama de clases. Paquete registry 86
Ilustración 28: Diagrama de clases. Paquete webratio 87
Ilustración 29: Diagrama de clases. Paquete webratioaux 88
Ilustración 30: Diagrama de clases. Paquete wizards 89
Ilustración 31: Localización del directorio patterns 98
Ilustración 32: Diagrama IFML del patrón Create sin relaciones 100
Ilustración 33: Diagrama IFML del patrón Create con relaciones 101
Ilustración 34: Diagrama IFML del patrón Read 104
Ilustración 35: Diagrama IFML del patrón Update sin relaciones 108
Ilustración 36: Diagrama IFML del patrón Update con relaciones ... 108
Ilustración 37: Diagrama IFML del patrón Delete 112
Ilustración 38: Diagrama IFML del patrón AllInOne sin relaciones .. 117
Ilustración 39: Diagrama IFML del patrón AllInOne con relaciones . 118
Ilustración 40: Ejecución del asistente. 1- Inicio del asistente 123
Ilustración 41: Ejecución del asistente. 2- Diálogo principal 124
Ilustración 42: Ejecución del asistente. 3- Selección de atributos ... 125
Ilustración 43: Ejecución del asistente. 4- Diálogo de generación... 126
Ilustración 44: Ejecución del asistente. 5- Generación (1) 127
Ilustración 45: Ejecución del asistente. 5- Generación (2) 127
Ilustración 46: Ejecución del asistente. 5- Generación (3) 128
Ilustración 47: Ejecución del asistente. 5- Generación (4) 129
Ilustración 48: Ejecución del asistente. 6- Registro (1)..... 130
Ilustración 49: Ejecución del asistente. 6- Registro (2)..... 131
Ilustración 50: Exportación del plug-in. Paso 1: proyecto en el workspace 132
Ilustración 51: Exportación del plug-in. Paso 2: File -> Export 132
Ilustración 52: Exportación del plug-in. Paso 3: selección de plug-in y directorio 133

Ilustración 53: Exportación del plug-in. Paso 4: ventana final 134
Ilustración 54: Ejemplo del archivo WebRatio.ini..... 135
Ilustración 55: Debug configurations 136
Ilustración 56: Cronología de reuniones y fases del proyecto 146

RESUMEN

Las tecnologías de desarrollo y despliegue de aplicaciones web con uso intensivo de datos han evolucionado considerablemente en los últimos años. El enfoque de Ingeniería Web dirigido por Modelos es un ejemplo de estas tecnologías. Propone la automatización de tareas de desarrollo redundante como el factor clave para su éxito. Sin embargo, la automatización de las operaciones CRUD[1] no se ha enfrentado adecuadamente.

Este es el objetivo principal de AutoCRUD, un plug-in de WebRatio que automatiza la generación de operaciones CRUD en IFML[2] (Interaction Flow Modeling Language). Fue desarrollado en 2013 por un alumno[3] de la Escuela Politécnica de la Universidad de Extremadura.

El trabajo que se expone a continuación se construye sobre estos cimientos. Se continúa insistiendo en la automatización de tareas de desarrollo, añadiendo nuevas funcionalidades al plug-in anterior que multiplican las posibilidades de generación y personalización de las operaciones CRUD.

Gracias a las mejoras desarrolladas, se permite al usuario definir el conjunto de elementos IFML que compone cada operación CRUD y generar automáticamente ese patrón de elementos en el modelo web. De esta forma, se elimina la necesidad de retocar manualmente el modelo web después de la generación automática. El proceso de generación permite al usuario controlar la velocidad de disposición de los elementos en el modelo.

Además, se puede consultar en todo momento la información de los patrones y elementos generados en un proyecto web determinado, junto con diversos datos estadísticos de interés.

El conjunto las funcionalidades y mejoras añadidas en este proyecto tienen un objetivo claro: la mejora de la productividad del usuario.

El proyecto se ha desarrollado poniendo especial énfasis en la calidad, asegurando su mantenibilidad y posibilitando el desarrollo de mejoras futuras.

[1] CRUD del inglés: Create, Read, Update, Delete
[2] http://www.ifml.org/
[3] Ver el punto *1.2 Antecedentes*

1 Introducción

De acuerdo con Rodríguez-Echeverría, *et al*. (2016)[4], los enfoques de la Ingeniería Web dirigida por modelos proporcionan metodologías y herramientas para el diseño y desarrollo de la mayoría de los tipos de aplicaciones web. Tratan diferentes aspectos utilizando modelos separados (navegación, presentación, datos, etc.), y son apoyados por compiladores de modelos que producen automáticamente la mayor parte de las páginas Web de la aplicación y el código lógico. Los beneficios de la utilización de estos enfoques son claros desde diferentes puntos de vista, tales como la productividad del equipo, la calidad del software o la adaptación a las tecnologías en evolución.

Entre los diferentes enfoques, vale la pena mencionar IFML (Interacting Flow Modeling Language), un estándar adoptado por la OMG [5] (Object Management Group) para el desarrollo de aplicaciones de datos intensivos que se ha convertido en una referencia en los desarrollos industriales. Su exitosa herramienta de desarrollo, WebRatio[6], permite la edición y validación de modelos IFML, pero también, y lo que es más importante, permite la generación del código de aplicación final para una plataforma de implementación tecnológica específica, reduciendo el tiempo de lanzamiento al mercado y el esfuerzo de desarrollo de estas aplicaciones.

Centrándose en el esfuerzo de desarrollo, una de las tareas más redundantes en el desarrollo de aplicaciones web con uso intensivo de datos es la implementación de operaciones CRUD. IFML / WebRatio no han proporcionado todavía una herramienta automática oficial para realizar estas tareas. Por ello se desarrolla AutoCRUD, permitiendo la generación automática de operaciones CRUD.

Pero ¿cuál es la motivación de la herramienta?

[4] RODRÍGUEZ-ECHEVERRÍA, R., CONEJERO, J., PRECIADO, J. y SÁNCHEZ-FIGUEROA, F. “AutoCRUD – Automating IFML Specification of CRUD Operations”. Proceedings of the 12th International Conference on Web Information Systems and Technologies (WEBIST 2016) - Volume 1, pages 307-314 ISBN: 978-989-758-186-1

[5] http://www.omg.org/

[6] http://www.webratio.com/

1.1 Motivación

La gran mayoría de los proyectos de aplicaciones web que se desarrollan, llevan a cabo numerosas implementaciones de las operaciones CRUD. El desarrollo de estas operaciones es muy reiterativo, cada operación suele seguir un patrón que se repite. Como es sabido, la implementación de tareas redundantes suele llevar a fallos, aumentando los costes del proyecto.

La necesidad de una herramienta como AutoCRUD para proyectos de Ingeniería Web dirigida por Modelos es clara. Se obtiene una mejora aproximadamente del 95% en el tiempo dedicado a la especificación de estas operaciones, además del ahorro de tiempo en la detección y depuración de errores.

Estos datos están respaldados por Homeria Open Solutions[7], una empresa externa especializada en el desarrollo de aplicaciones web intensivas en datos mediante WebRatio que ha colaborado con el desarrollo del plugin.

¿Pero qué ocurre si la operación CRUD se define utilizando otros elementos? El usuario tiene que modificar manualmente la cadena de operaciones para satisfacer sus necesidades. El grado de personalización de AutoCRUD no admite la modificación de elementos en un patrón.

Esta es la mejora principal que se plantea en este trabajo, permitir al usuario modificar los patrones que definen las operaciones CRUD y crear sus propios patrones personalizados. De esta forma, se amplía considerablemente el rango de acción y se aumentan las posibilidades de utilización de la herramienta, aumentando la productividad del usuario final.

1.2 Antecedentes

Este proyecto se propone como una extensión del Proyecto Fin de Carrera: "*Generación automática de la arquitectura de una aplicación web en WebML a partir de la especificación de requisitos*", realizado y presentado por Carlos Aguado Fuentes, alumno de Ingeniería Informática de la Escuela Politécnica de la Universidad de Extremadura en Julio de 2013.

[7] http://www.homeria.com/

1.3 Objetivos

Tomando como base Wizard CRUD Assistant v2.0.1[8] y siguiendo la línea de investigación que este dejó abierta, se continúa incidiendo en la sistematización y productividad del desarrollo de modelos Web para WebRatio.

Para poder cumplir esta meta global, se definen una serie de objetivos con una prioridad de cumplimiento asociada a cada uno de ellos.

Objetivo 1: Descripción externa de patrones. Prioridad: Alta

Se pretende que la aplicación genere las unidades definidas en un archivo externo, dando la posibilidad al usuario de crear/modificar sus propios patrones siguiendo unas reglas establecidas.

❖ Listado de subobjetivos:

- Definir una estructura del patrón genérica que permita representar los patrones de la versión del plug-in anterior (CRUD y AllInOne) y los que el usuario defina.
- Definir la sintaxis y la semántica de los elementos y sus atributos.
- Dotar a la aplicación de la capacidad de interpretar los patrones, almacenar la información y generar sus unidades con las reglas establecidas.
- Validar los patrones en base a la estructura diseñada.

Tabla 1 – Objetivo 1: Descripción externa de patrones

[8] Wizard CRUD Assistant v2.0.1 (AutoCRUD): versión final del plugin de WebRatio desarrollado por el Proyecto Fin de Carrera sobre el que se basa este trabajo.

Objetivo 2: Simplificar la interacción. Prioridad: Alta

Se requiere modificar la interfaz gráfica de usuario para unificar las páginas del asistente y eliminar la creación de SV/Area, resultando ser más intuitivo y manejable.

❖ Listado de subobjetivos:

- Eliminar la funcionalidad que crea un SiteView/Area nuevo.
- Aunar las páginas restantes del asistente.
- Visualizar completamente el contenido de los componentes.

Tabla 2 - Objetivo 2: Simplificar la interacción

Objetivo 3: Registro. Prioridad: Alta

Se quiere añadir un registro que permita almacenar la información de los patrones generados y sus elementos. Además, utilizando la información registrada, se quiere poder ver/operar sobre un patrón directamente y generar datos estadísticos.

❖ Listado de subobjetivos:

- Diseñar la estructura de datos que almacenará el registro.
- Registrar los elementos generados en base a la estructura diseñada.
- Operar a nivel de patrón sobre el modelo.
- Identificar conflictos por cambios manuales en el modelo.
- Mostrar el registro.
- Generar datos estadísticos.
- Exportar los datos estadísticos, en CSV (comma-separated values) o similar.

Tabla 3 - Objetivo 3: Registro

Objetivo 4: Generación paso a paso. Prioridad: Media

Se quiere dar la posibilidad al usuario de controlar la velocidad de generación de elementos en el modelo.

❖ Listado de subobjetivos:

- Modificar la secuencia de generación.
- Mostrar información del progreso.

Tabla 4 - Objetivo 4: Generación paso a paso

Objetivo 5: Refactorización del código fuente. Prioridad: Media

Se quiere obtener una versión final profesional del producto. Para ello, se debe realizar una refactorización general del código fuente mejorando su calidad y mantenibilidad.

❖ Listado de subobjetivos:

- Mejorar la eficiencia general del código de la versión anterior.
- Traducir variables y documentación de la versión anterior.
- Realizar la documentación interna faltante de la versión anterior.

Tabla 5 - Objetivo 5: Refactorización del código fuente

Objetivo 6: Identificación de patrones existentes. Prioridad: Baja

Se quiere analizar el modelo en busca de una combinación de elementos que se asemeje a un patrón, identificarlo y registrarlo.

❖ Listado de subobjetivos:

- Crear un sistema de búsqueda.
- Recorrer los elementos del modelo aplicando el sistema de búsqueda.
- Identificar el patrón en el modelo web.
- Registrarlo.

Tabla 6 - Objetivo 6: Identificación de patrones existentes

2 Estado del arte

2.1 Ingeniería Web

El concepto general de Ingeniería Web nace a finales de los años 90. Aunque es en marzo de 2001 cuando se publica el número especial de la revista IEEE Multimedia[9], donde los autores[10] que se atribuyen la autoría del término *"Web Engineering"* escribieron una editorial y diversos artículos al respecto.

Sin embargo, el nacimiento del nuevo concepto no estuvo exento de polémica:

Sus detractores defendían que el elemento fundamental que compone la Ingeniería Web, al fin y al cabo, es software. Por lo que sería un concepto ya existente dentro de la disciplina de la Ingeniería del Software.

Los promotores del nuevo concepto planteaban que la Ingeniería Web incluye conocimientos que están lejos de lo que se conoce como Ingeniería del Software tradicional como, por ejemplo, temas relacionados con el diseño gráfico y diseño multimedia, que en el desarrollo clásico de aplicaciones de gestión en Ingeniería del Software no se les daba mayor importancia. Tampoco había herramientas o técnicas para modelar desde el punto de vista del diseño gráfico, documentos y contenidos multimedia.

Además, se encontraba un tercer grupo de expertos que proponían que el nuevo concepto se enmarcase en una disciplina más general como la Ingeniería Documental, porque lo que se construyen son documentos. Recordemos que los documentos de lenguajes de marcado nacieron en el ámbito editorial, en el de la Ingeniería Documental con el formato SGML (Standard Generalized Markup Language) que luego daría lugar al formato XML (Extensible Markup Language), más ligado al ámbito informático, pero ya existían estas ideas de marcado en el ámbito editorial.

En definitiva, lo que suele aceptarse hoy en día es que, la Ingeniería Web, es una disciplina informática con identidad propia que está ligada a otros

[9] IEEE MultiMedia Volume 8, Number 1, January / March, 2001: ftp://ftp.math.utah.edu/pub/tex/bib/toc/ieeemultimedia.html#8(1):January/March:2001

[10] Athula Ginige and San Murugesan. *University of Western Sydney, Australia*

tipos de ingeniería, aprovechando muchos conocimientos y conceptos de otros campos como son la Ingeniería Documental, Ingeniería Hipermedia y la Ingeniería del Software.

Uno de los problemas que surgen en el ámbito del desarrollo de aplicaciones web, es la multitud de metodologías que han publicado diferentes autores desde mediados de los años 90. Obteniéndose así demasiados enfoques y vocabularios diferentes, distintos métodos y técnicas propuestas para las fases de desarrollo. A todo ello había que añadir la existencia de pocas herramientas CASE (Computer Aided Software Engineering) que soportasen completamente las propuestas.

2.2 Ingeniería Web dirigida por Modelos

En general, todas estas metodologías proponen que se realicen modelos durante las fases de desarrollo del proyecto.

La idea es la de formalizar al máximo posible las especificaciones de un proyecto tanto a nivel de análisis como a nivel de diseño, de tal manera que se pueda pasar de los modelos de más alto nivel (modelos de la fase de análisis) a los modelos más ligados a la plataforma que se vaya a utilizar (Ej. JAVA, PHP…) a través de transformaciones. Transformaciones que incluso puedan ser automáticas, es decir, que se pueda obtener el código de la aplicación web a partir del modelo definido.

Se detecta la existencia de tres tipos de modelos que son comúnmente recomendados en estas metodologías:

- Modelo conceptual de información, que muestre la estructura de la información que se va a manejar en una aplicación web. Por ejemplo, diagrama de clases UML (Unified Modeling Language)
- Modelo navegacional, que muestre las posibilidades de navegación por las páginas de una aplicación web por parte del usuario. (Ej. Propuesta original de J. Conallen[11], en forma de perfil UML, aunque no está estandarizado)

[11] James Conallen: https://www.researchgate.net/profile/James_Conallen

- Modelo de presentación, que propone modelar la interfaz con el usuario en este caso en forma de páginas web (Ej. Estándar IFML)

2.3 WebML

Una de las metodologías que surgieron del nuevo concepto de Ingeniería Web comentadas en el punto anterior fue WebML (Web Modeling Language).

La especificación de una aplicación Web en WebML consiste en un conjunto de modelos ortogonales: el modelo de datos de la aplicación (un modelo Entidad-Relación extendido), uno o más modelos de hipertexto (es decir, diferentes vistas para diferentes tipos de usuarios), expresando las rutas de navegación y la composición de la página; Y el modelo de presentación, describiendo el aspecto visual de las páginas.

En sus orígenes, fue definido como un modelo conceptual para las aplicaciones web de datos intensivos. Su principal innovación fue la notación de modelos de hipertexto, que permite la especificación de páginas web consistentes en componentes conceptuales (unidades) interconectados por enlaces conceptuales. El modelo de hipertexto se dibuja en una notación visual simple y bastante intuitiva, pero tiene una semántica rigurosa, que permite la transformación automática de diagramas en el código completo de ejecución de una aplicación web de datos intensivos.

2.4 IFML y WebRatio

IFML se inspira en la extensa experiencia de 10 años de WebML y WebRatio. De hecho, IFML fue promovido por WebRatio, una empresa derivada de la Universidad Politécnica de Milán, nacida durante un proyecto de investigación cooperativa europea en un período de tiempo (1999-2001) en el que la Web estaba cambiando la forma en que se implementaba el software. La empresa desarrolló una herramienta (llamada WebRatio en sí) para el desarrollo dirigido por modelos de aplicaciones web empresariales, basado en el lenguaje de modelado WebML.

IFML fue adoptado como un estándar por OMG en marzo de 2013. En febrero de 2014, OMG emitió la versión Beta2 de la norma. En marzo de 2014,

la Junta de Arquitectura de OMG adoptó formalmente la especificación de IFML 1.0 y fue publicada en marzo de 2015.

Entre los objetivos de este estándar se encuentran los de aislar los aspectos de la implementación de los aspectos de las interfaces gráficas de usuario y realizar una especificación formal de los componentes que constituyen el *front-end* de una aplicación.

De esta forma se podrán distinguir componentes, eventos, acciones, flujos de navegación y flujos de datos.

3 Metodología

Como cualquier proyecto software, el desarrollo del trabajo ha sido un proceso largo y laborioso que no ha estado exento de complicaciones y de decisiones tomadas para su resolución. Sin embargo, el aprendizaje realizado en estos años de carrera universitaria ha ayudado enormemente en todas las fases del proyecto: desde la planificación hasta la documentación, pasando por la fase de desarrollo y el conjunto de decisiones tomadas a lo largo de todo el proyecto, el de mayor envergadura realizado hasta la fecha.

3.1 Fases

Se distinguen 4 grandes fases en la realización de este trabajo:

- Fase 1: Inicio del proyecto (principios de febrero – principios de abril)

Esta fase se dedicó al estudio del Proyecto Fin de Carrera anterior sobre el que se construiría este trabajo. Comenzó con la lectura de su memoria, seguido de las pruebas oportunas en WebRatio para entender la funcionalidad del plug-in desarrollado. Una vez familiarizado con su funcionamiento se inició el estudio del código fuente.

Tuvimos la primera reunión para discutir los objetivos y el alcance del proyecto. Los días siguientes se emplearon en planificar las tareas, colocando en primer lugar aquellas necesarias para cumplir los objetivos prioritarios. Para el final, se dejaron aquellas tareas correspondientes a objetivos menos

prioritarios y, algunos, con la marca de opcionalidad, cuyas implementaciones dependerían del tiempo empleado en las anteriores.

- Fase 2: Desarrollo del proyecto (principios de abril – finales de julio)

Es la fase que constituye el grueso del proyecto y donde se ha empleado mayor tiempo.

Su ciclo de vida se asemeja al desarrollo iterativo e incremental, donde una funcionalidad representa una iteración del ciclo de vida. En cada una de ellas se realiza: un análisis sobre lo que se quiere hacer, un diseño de cómo se llevaría a cabo, seguido de su implementación con las pruebas unitarias del componente desarrollado, finalizando con la integración del componente en el resto de la aplicación y realizando las pruebas de integración. El resultado de cada iteración produce un prototipo completamente funcional con funcionalidad añadida.

A lo largo de esta fase tuvieron lugar varias reuniones con la finalidad de informar del trabajo realizado y, sobre todo, para solventar las dudas surgidas durante el proceso de desarrollo.

La fase finaliza con una reunión en las instalaciones de Homeria Open Solutions, donde se expone el trabajo realizado con la intención de mejorar la aplicación con los comentarios que se realizasen durante la misma.

- Fase 3: Pruebas (mediados de julio – principios de agosto)

El tiempo destinado a esta fase se emplea en realizar pruebas globales de la aplicación: generando los patrones de ejemplo, revisando cada elemento y comprobando los atributos de las relaciones generadas.

En esta fase de pruebas se realizaron mejoras en la interfaz, realizando más sencilla e intuitiva la interacción del usuario con la misma. También se llevaron a cabo diversas modificaciones para corregir pequeños fallos descubiertos al realizar las pruebas.

- Fase 4: Documentación (finales de julio – finales de agosto)

Comienza la recopilación de toda la información generada durante todas las fases anteriores, redactándola en este documento final.

3.2 Diagrama de Gantt

A continuación, se muestra un diagrama cronológico con todas las fases del proyecto y sus tareas más relevantes:

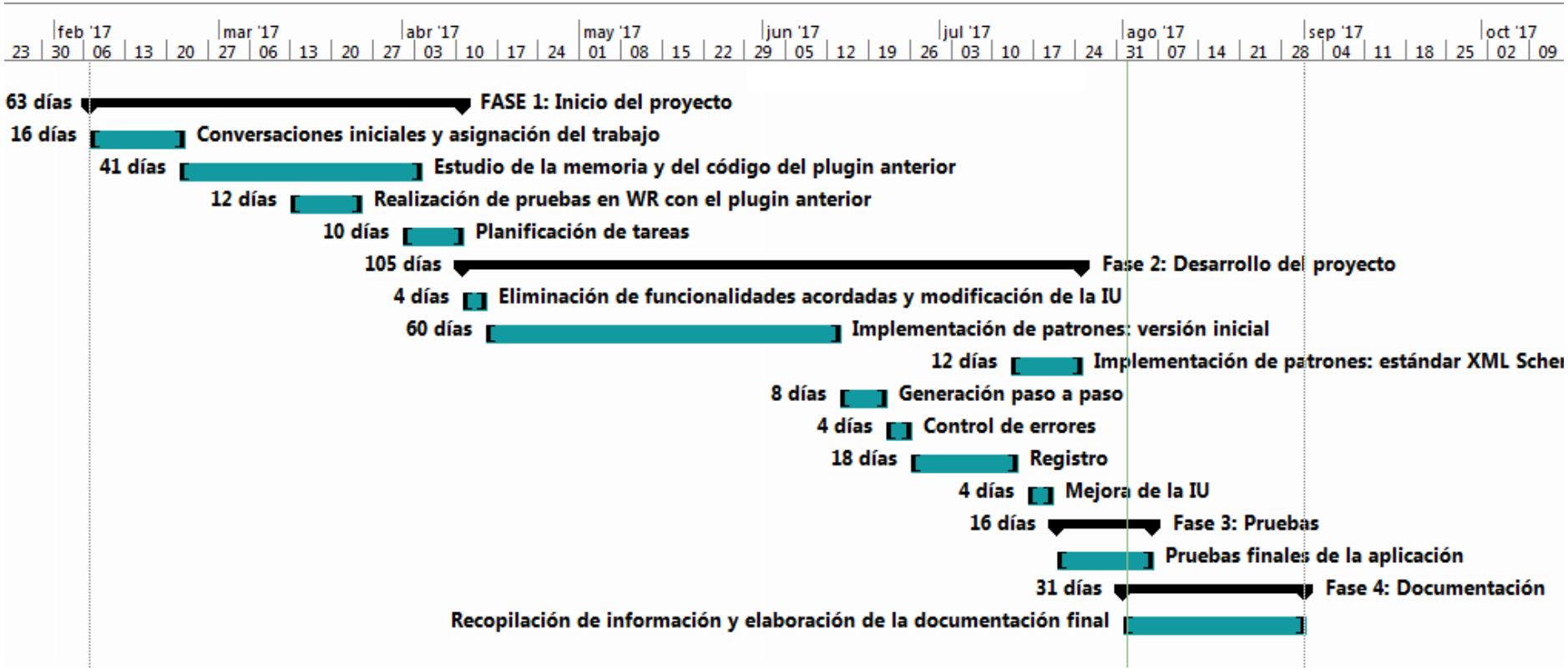

Ilustración 1: Diagrama de Gantt

4 Punto de partida

El trabajo se construye sobre la versión final del asistente desarrollado por el Proyecto Fin de Carrera anteriormente citado *(ver 1.2 Antecedentes)*.

El asistente mencionado tiene tres páginas:

1. Página de selección de entidad:

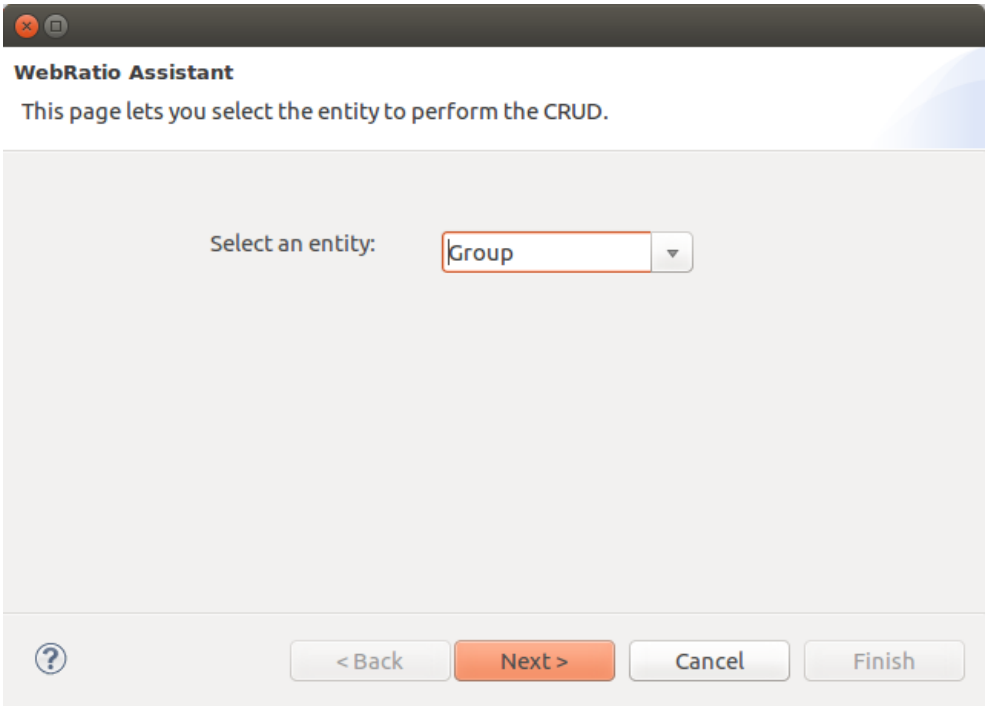

Ilustración 2: Wizard CRUD Assistant v2. Página de selección de entidad

2. Página de creación de SiteView o Area:

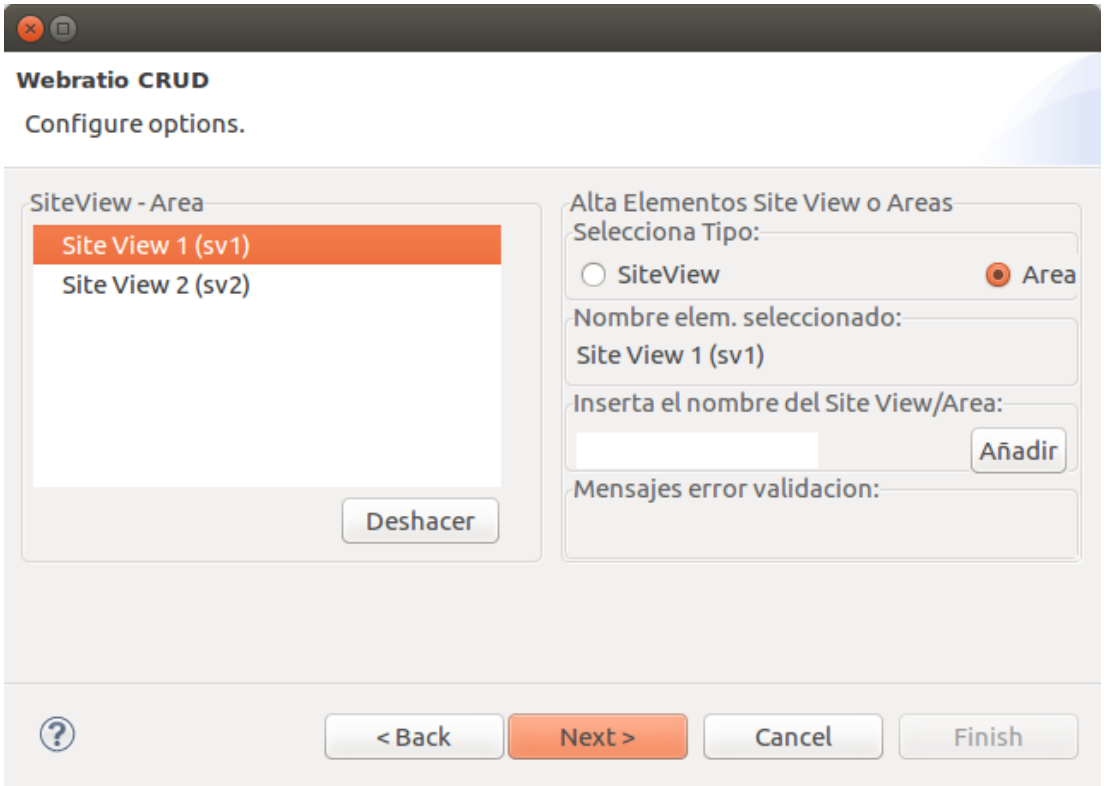

Ilustración 3: Wizard CRUD Assistant v2. Página de creación de SiteView/Area

3. Página de selección de patrón y atributos:

3.1. Pestaña AllInOne

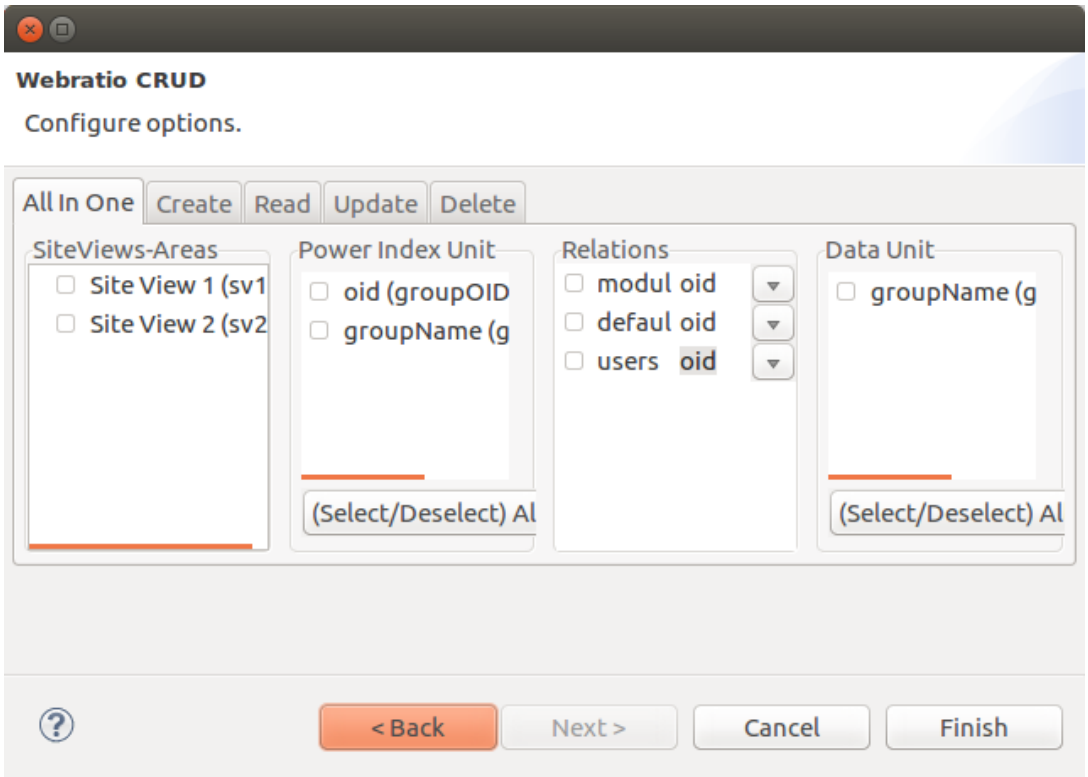

Ilustración 4: Wizard CRUD Assistant v2. Página de selección de patrón. Pestaña AllInOne

3.2. Pestaña Create

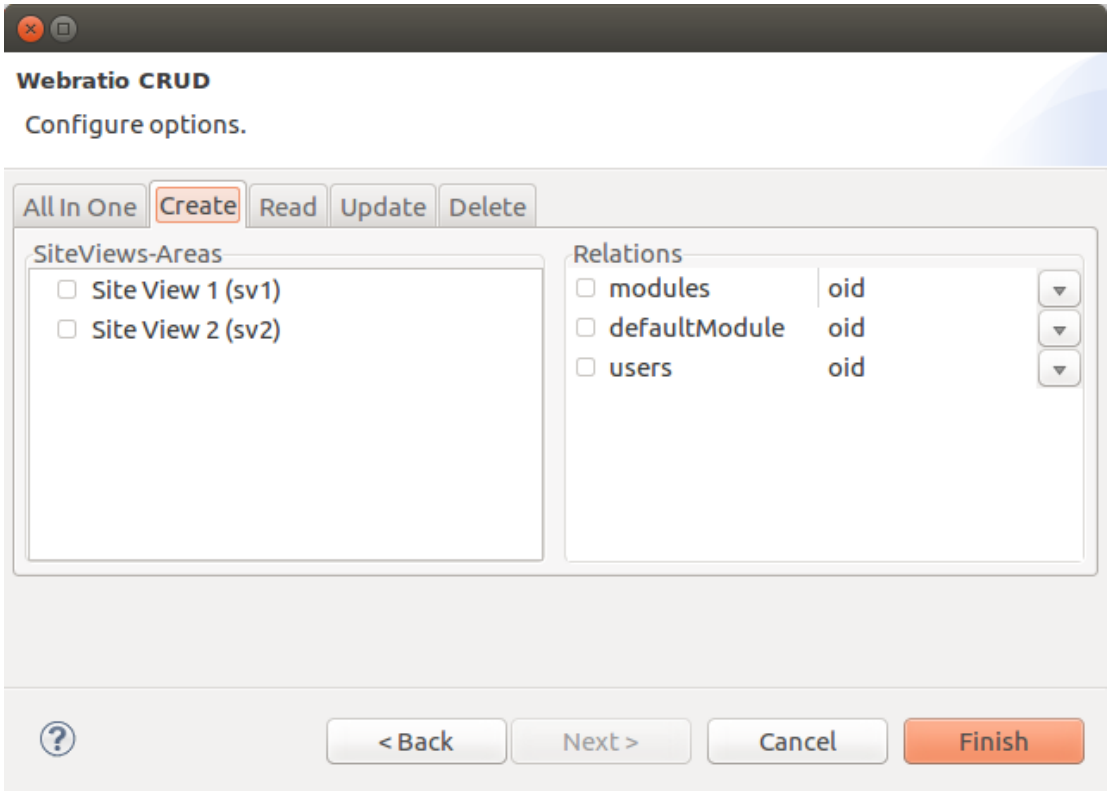

Ilustración 5: Wizard CRUD Assistant v2. Página de selección de patrón. Pestaña Create

3.3. Pestaña Read

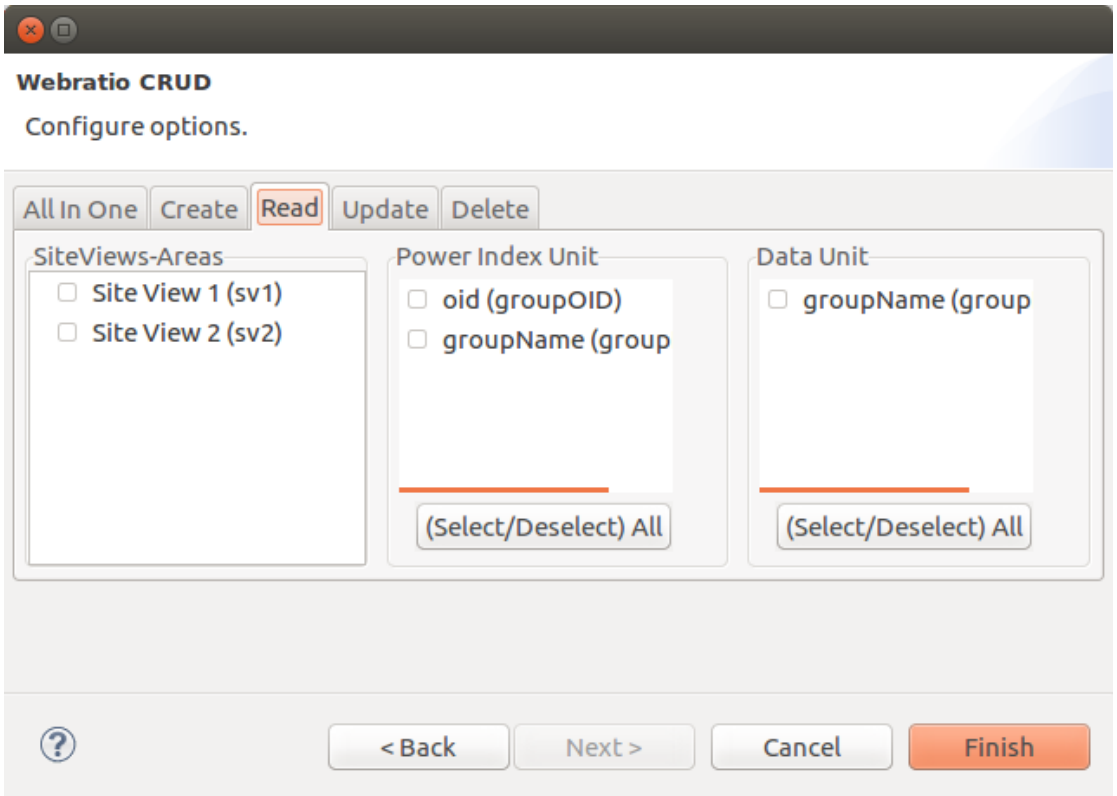

Ilustración 6: Wizard CRUD Assistant v2. Página de selección de patrón. Pestaña Read

3.4. Pestaña Update

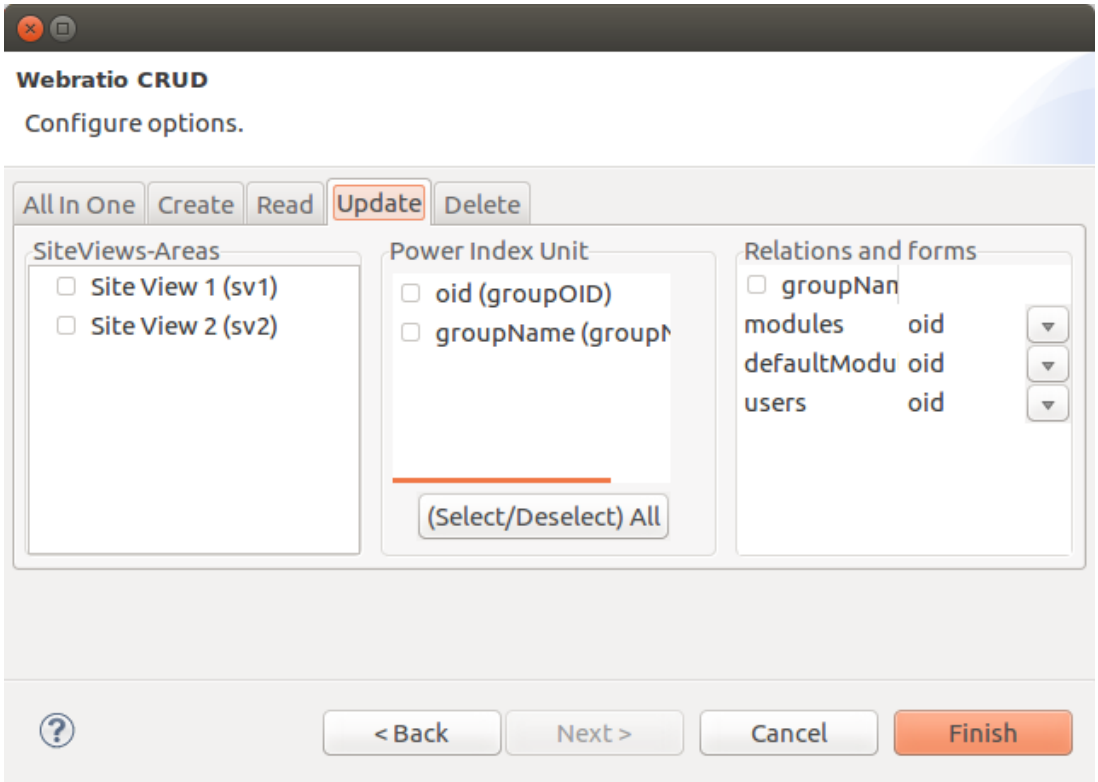

Ilustración 7: Wizard CRUD Assistant v2. Página de selección de patrón. Pestaña Update

3.5. Pestaña Delete

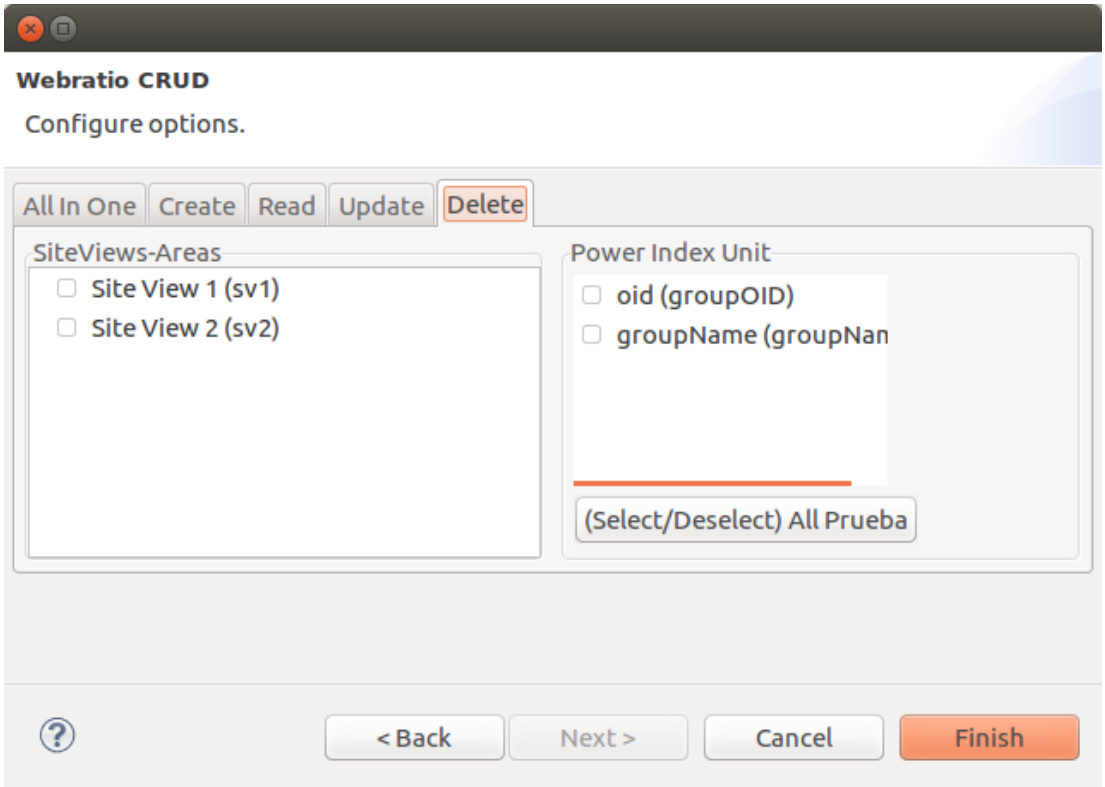

Ilustración 8: Wizard CRUD Assistant v2. Página de selección de patrón. Pestaña Delete

Los cinco patrones (Create, Read, Update, Delete y All In One) están definidos en el código interno de la aplicación. La modificación principal que se realiza en este proyecto consiste en posibilitar al usuario la definición de esos mismos patrones en un archivo externo. De esta forma, el usuario puede modificar estos archivos o crear nuevos patrones personalizados. En el apartado *5.1 Patrones* y sucesivos se describe la solución completa a esta modificación.

Por otra parte, la interfaz de usuario necesita una serie de modificaciones:

La *página de selección de entidad* contiene únicamente un selector. Dado la escasa ocupación espacial del mismo, se elimina esta página y se recoloca el componente en otra, facilitando la interacción con el usuario eliminando una página del asistente que muestra poca información.

La *página de creación de SiteView/Area* se elimina porque la funcionalidad de creación de un SiteView se sale del objetivo del asistente. Por lo que quedaría una página principal con toda la funcionalidad en su interior.

Se puede ver en la *página de selección de patrón y atributos* que la información, en ocasiones, se muestra incompleta y con espacio reducido.

Todos estos aspectos y algunos más que tienen que ver con la modificación de la interfaz gráfica, se describen en el punto *5.5 Interfaz de usuario.*

Además, se añaden funcionalidades nuevas, como el registro de elementos que se puede consultar en el punto *5.4 Registro* y mejoras en el código fuente, realizando una refactorización general como se explica en el punto *6.1.2 Clases.*

5 Diseño de la solución

En este apartado se detallan todos los aspectos de diseño y las decisiones tomadas de la solución a los objetivos propuestos.

5.1 Patrones

Un patrón es un conjunto de elementos relacionados entre sí con un propósito común.

Por ejemplo, el patrón “Create” se define mediante una serie de flujos y unidades, formando la cadena de operaciones que permite al usuario final introducir nuevas entradas en la base de datos.

Un mismo patrón se puede definir de varias formas diferentes. Aunque la finalidad objetiva sea la misma, un usuario puede requerir la inclusión de alguna operación intermedia, fusionar patrones, etc. Aquí es donde surge la necesidad de permitir al usuario la creación de sus propios patrones permitiendo la generación automática de elementos IFML con el asistente.

Para hacer esto posible, se crea una estructura general del patrón, se definen los elementos y se establecen una serie de reglas para que el usuario pueda definir patrones válidos para su correcta generación.

5.1.1 Estructura interna

Debido a que WebRatio utiliza XML como lenguaje de marcado de documentos para almacenar los modelos web, se decide seguir la misma línea utilizando el mismo lenguaje para la definición del patrón.

Se comienza con el análisis de los elementos que se pueden generar con Wizard CRUD Assistant v2.0.1, obteniéndose el siguiente listado:

Contenedores	Unidades de contenido	Unidades de operación
Page Alternative (XOR)	PowerIndexUnit DataUnit MultiMessageUnit NoOpContentUnit EntryUnit	CreateUnit DeleteUnit UpdateUnit ConnectUnit DisconnectUnit ReconnectUnit[12]
Flujos	**Control de flujo**	**Componentes de utilidad**
NormalNavigationFlow DataFlow OKLink KOLink	IsNotNullUnit	SelectorUnit

Tabla 7 - Elementos generables

Seguidamente se analiza de forma independiente cómo se genera cada elemento. Para ello, se estudian las clases existentes que realizan la generación: *Create.java, Read.java, Update.java, Delete.java* y *AllInOne.java (ver 6.1.2.1 Modificación de clases del proyecto anterior)*

Se realiza un listado de los **parámetros** que se debe proporcionar a cada unidad para su generación. Hay que tener en cuenta que una misma unidad se puede comportar en el Modelo Web de manera diferente dependiendo de los parámetros que se proporcionen en el momento de la generación.

Además de los parámetros, hay que poner atención a la **dependencia** entre unidades, una característica esencial con mucho significado. Esta dependencia puede ser:

- Relaciones de contención: una unidad está contenida en una página, por lo que su padre es la propia página.

[12] La unidad se ha añadido, posibilitando su generación en esta nueva versión.

- Relacionada por los flujos: el origen y el destino de un flujo son unidades previamente definidas en el patrón.

Sea como fuere, esta dependencia necesita ser resuelta por parte del usuario.

Otro apartado (opcional) debe ser el de las **relaciones**. ¿Por qué opcional? Porque depende íntegramente de la selección de relaciones en la interfaz de usuario. Se pueden seleccionar ninguna, una o varias relaciones. También hay que tener en cuenta la cardinalidad de la relación seleccionada, porque se necesita especificar un comportamiento diferente que depende de ella. Por lo que esta sección se dividirá en subsecciones que permiten especificar el comportamiento deseado en cada alternativa.

Se han llegado a desarrollar tres versiones de la estructura del patrón. La tercera y última es la que se explica a continuación, combinando las mejores cualidades y solucionando los fallos de sus predecesoras.

El archivo XML donde se define el patrón debe tener la siguiente estructura:

```xml
<?xml version="1.0"?>

<PATTERN id="" name=""
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:noNamespaceSchemaLocation="schema.xsd">

    <!-- PAGES SECTION-->
    <PAGES>

    </PAGES>


    <!-- UNITS OUTSIDE OF ANY PAGE -->
    <OUTSIDEUNITS>

    </OUTSIDEUNITS>


    <!-- LINKS SECTION -->
    <LINKS>

    </LINKS>


    <!-- RELATIONS SECTION -->
    <RELATIONS> <!-- Common section for 1:1, 1:N and N:M relations -->
        <ALL> <!-- All relations -->

        </ALL>

        <LAST> <!-- Last relation (used to close operations chain) -->

        </LAST>
    </RELATIONS>

    <NMRELATIONS> <!-- Specific section only for N:M relations (Replaces RELATIONS)-->
        <ALL> <!-- All N:M relations -->

        </ALL>

        <FIRST> <!-- First N:M relation (used to start operations chain) -->

        </FIRST>

        <REMAINING> <!-- Second and subsequent N:M relations (used to continue operations chain) -->

        </REMAINING>

        <LAST> <!-- Last relation (used to close operations chain) -->

        </LAST>
    </NMRELATIONS>
</PATTERN>
```

Ilustración 9: Estructura interna del patrón

Obligatoriamente, debe tiene un nodo raíz (*PATTERN*) con dos **atributos**:

1. id: identificador único del patrón. No puede haber otro patrón con el mismo identificador. Atributo obligatorio.
2. name: nombre del patrón a mostrar. Atributo opcional, aunque recomendable.

Este nodo raíz contiene cinco grandes **secciones**:

1. Sección para definir las páginas y su contenido (PAGES)

2. Sección para definir las unidades que no son contenidas por ninguna página (OUTSIDEUNITS)
3. Sección para definir los flujos (LINKS)
4. Sección que debe procesarse por cada relación seleccionada en la interfaz de usuario (RELATIONS)
5. Sección que debe procesarse por cada relación N:M seleccionada en la interfaz de usuario. (NMRELATIONS). En caso de definirse y sólo en el caso de ser una relación N:M la que se está procesando, esta sección reemplazaría a la anterior (RELATIONS)

Dentro de las secciones mencionadas se definen los elementos (se explican en el siguiente apartado). Todas las secciones son opcionales por lo que puede omitirse su definición si no va a contener ningún elemento.

5.1.2 Elementos

Nuevamente, se analizan las clases del proyecto anterior que generan los elementos en el modelo de WebRatio. Esta vez se focaliza en cada unidad de forma independiente, conociendo las diferentes formas que adopta en cada patrón dependiendo de los parámetros que se le proporcione.

Como resultado de este análisis, se obtiene, por cada elemento, una lista de parámetros. Hay parámetros que son intrínsecos de la generación y no tiene sentido que el usuario siquiera conozca su existencia así que se indicarán en el código interno. Los que son susceptibles de personalización por parte del usuario se han extraído para su definición en el patrón en forma de atributos.

Cada elemento web que generar se representará mediante una etiqueta XML con su nombre identificativo, seguido de una serie de atributos.

Todos los elementos tienen atributos que son comunes a otros elementos. Sin embargo, ciertos elementos tienen atributos específicos porque necesitan definir un comportamiento único.

Todos los elementos que se pueden generar y sus atributos se detallan a continuación.

Antes de definir los elementos que se pueden definir en un patrón, se explican los atributos que son comunes en ciertos elementos.

Atributos comunes

Nombre	Obligatorio	Descripción
id	Sí	Identificador único del elemento.
name	No	Nombre que se mostrará en el modelo.
x	Sí	Coordenada X del elemento.
y	Sí	Coordenada Y del elemento.
sourceId	Sí	Identificador de la unidad origen del flujo
targetId	Sí	Identificador de la unidad destino del flujo

Tabla 8 - Atributos comunes

A continuación, se detalla toda la información necesaria para definir los elementos en el patrón. Se clasifican por tipo: contendores, unidades de contenido, unidades de operación, flujos, control de flujo y componente de utilidad.

- **Contenedores:**

Etiqueta:	page
Elemento de WebRatio:	Page
Atributos comunes**:**	id, name, x, y
Atributos específicos opcionales: ❖ landmark: establece la propiedad landmark de la página. o Valores permitidos: "*true*" o "*false*" ❖ default: establece la propiedad default de la página. o Valores permitidos: "*true*" o "*false*"	
Localización: Se puede definir dentro de: - La sección *PAGES* - El elemento *page* - El elemento *xor*	

Tabla 9 - Elemento page

Etiqueta:	xor
Elemento de WebRatio:	Alternative (XOR Page)
Atributos comunes**:**	id, name, x, y
Atributos específicos opcionales: No tiene	
Localización: Se puede definir dentro de: - El elemento *page*	

Tabla 10 - Elemento xor

- **Unidades de contenido:**

Etiqueta:	powerIndexUnit
Elemento de WebRatio:	List
Atributos comunes**:**	id, name, x, y
Atributos específicos opcionales: No tiene	
Localización: Se puede definir dentro de: - El elemento *page*	

Tabla 11 - Elemento powerIndexUnit

Etiqueta:	dataUnit
Elemento de WebRatio:	Detail
Atributos comunes**:**	id, name, x, y
Atributos específicos opcionales: No tiene	
Localización: Se puede definir dentro de: - El elemento *page*	

Tabla 12 - Elemento dataUnit

Etiqueta:	multiMessageUnit
Elemento de WebRatio:	Message
Atributos comunes**:**	id, name, x, y
Atributos específicos opcionales: No tiene	
Localización: Se puede definir dentro de: - El elemento *page*	

Tabla 13 - Elemento multiMessageUnit

Etiqueta:	noOpContentUnit
Elemento de WebRatio:	View Component
Atributos comunes**:**	id, name, x, y
Atributos específicos opcionales: No tiene	
Localización: Se puede definir dentro de: - El elemento *page*	

Tabla 14 - Elemento noOpContentUnit

Etiqueta:	entryUnit
Elemento de WebRatio:	Form
Atributos comunes**:**	id, name, x, y
Atributo específico opcional: ❖ type: modifica el comportamiento de la unidad. o Valor permitido "*preloaded*": activa la propiedad *preloaded* para todos los campos del formulario.	
Localización: Se puede definir dentro de: - El elemento *page*	

Tabla 15 - Elemento entryUnit

- **Unidades de operación:**

Etiqueta:	createUnit
Elemento de WebRatio:	Create
Atributos comunes**:**	id, name, x, y
Atributos específicos opcionales: No tiene	
Localización: Se puede definir dentro de: - La sección *OUTSIDEUNITS*	

Tabla 16 - Elemento createUnit

Etiqueta:	deleteUnit
Elemento de WebRatio:	Delete
Atributos comunes**:**	id, name, x, y
Atributos específicos opcionales: No tiene	
Localización: Se puede definir dentro de: - La sección *OUTSIDEUNITS*	

Tabla 17 - Elemento deleteUnit

Etiqueta:	updateUnit
Elemento de WebRatio:	Update
Atributos comunes**:**	id, name, x, y
Atributos específicos opcionales: No tiene	
Localización: Se puede definir dentro de: - La sección *OUTSIDEUNITS*	

Tabla 18 - Elemento updateUnit

Etiqueta:	connectUnit
Elemento de WebRatio:	Connect
Atributos comunes**:**	id, name, x, y
Atributos específicos opcionales: No tiene	
Localización: Se puede definir dentro de: - La subsección *ALL* situada en la sección *RELATIONS* - La subsección *ALL* situada en la sección *NMRELATIONS*	

Tabla 19 - Elemento connectUnit

Etiqueta:	disconnectUnit
Elemento de WebRatio:	Disconnect
Atributos comunes**:**	id, name, x, y
Atributos específicos opcionales: No tiene	
Localización: Se puede definir dentro de: - La subsección *ALL* situada en la sección *RELATIONS* - La subsección *ALL* situada en la sección *NMRELATIONS*	

Tabla 20 - Elemento disconnectUnit

Etiqueta:	reconnectUnit
Elemento de WebRatio:	Reconnect
Atributos comunes**:**	id, name, x, y
Atributos específicos opcionales: No tiene	
Localización: Se puede definir dentro de: - La subsección *ALL* situada en la sección *RELATIONS* - La subsección *ALL* situada en la sección *NMRELATIONS*	

Tabla 21 - Elemento reconnectUnit

- **Flujos:**

Etiqueta:	normalNavigationFlow
Elemento de WebRatio:	Normal Navigation Flow
Atributos comunes**:**	id, name, sourceId, targetId

Atributo específico opcional:

- ❖ type: modifica el comportamiento del flujo.
  - Valores permitidos:
    - *"isNotNull"*: utilizado para realizar un acoplamiento con una isNotNullUnit.
    - *"fixedValue"*: realiza el enlace utilizando un valor fijo (0)
    - *"entryToCreate"*: utilizado para realizar un acoplamiento entre una entryUnit y una createUnit.
    - *"entryToUpdate"*: utilizado para realizar un acoplamiento entre una entryUnit y una updateUnit.
  - Valor permitido "*preloaded"*. Activa la propiedad *preloaded* para todos los campos del formulario, salvo los campos de relación.

Localización:

Se puede definir dentro de:

- La sección *LINKS*
- La subsección *ALL* situada en la sección *RELATIONS*
- La subsección *LAST* situada en la sección *RELATIONS*
- La subsección *ALL* situada en la sección *NMRELATIONS*
- La subsección *FIRST* situada en la sección *NMRELATIONS*
- La subsección *REMAINING* situada en la sección *NMRELATIONS*
- La subsección *LAST* situada en la sección *NMRELATIONS*

Tabla 22 - Elemento normalNavigationFlow

Etiqueta:	dataFlow
Elemento de WebRatio:	Data Flow
Atributos comunes**:**	id, name, sourceId, targetId

Atributo específico opcional:

- ❖ type: modifica el comportamiento del flujo.
  - o Valores permitidos:
    - ▪ *"preload"*: Se utiliza precargar los campos de relación en la entryUnit (destino).
    - ▪ *"unitToEntry"*: utilizado para realizar un acoplamiento entre cualquier unidad sin relación y una entryUnit.
    - ▪ "*unitToEntryRole*": utilizado para realizar un acoplamiento entre cualquier unidad con relación y una entryUnit.
    - ▪ *"entryToCreate"*: utilizado para realizar un acoplamiento entre una entryUnit y una createUnit.
    - ▪ *"entryToUpdate"*: utilizado para realizar un acoplamiento entre una entryUnit y una updateUnit.
    - ▪ *"entryToConnect"*: utilizado para realizar un acoplamiento entre una entryUnit y una connectUnit.
    - ▪ *"entryToReconnect"*: utilizado para realizar un acoplamiento entre una entryUnit y una reconnectUnit.

Localización:

Se puede definir dentro de:

- La sección *LINKS*
- La subsección *ALL* situada en la sección *RELATIONS*
- La subsección *LAST* situada en la sección *RELATIONS*
- La subsección *ALL* situada en la sección *NMRELATIONS*
- La subsección *FIRST* situada en la sección *NMRELATIONS*
- La subsección *REMAINING* situada en la sección *NMRELATIONS*
- La subsección *LAST* situada en la sección *NMRELATIONS*

Tabla 23 - Elemento dataFlow

Etiqueta:	okLink
Elemento de WebRatio:	OK Flow
Atributos comunes**:**	id, name, sourceId, targetId

Atributo específico opcional:

- ❖ type: modifica el comportamiento del flujo.
  - o Valores permitidos:
    - ▪ *"noCoupling"*: desactiva el acoplamiento automático.

Localización:

Se puede definir dentro de:

- La sección *LINKS*
- La subsección *ALL* situada en la sección *RELATIONS*
- La subsección *LAST* situada en la sección *RELATIONS*
- La subsección *ALL* situada en la sección *NMRELATIONS*
- La subsección *FIRST* situada en la sección *NMRELATIONS*
- La subsección *REMAINING* situada en la sección *NMRELATIONS*
- La subsección *LAST* situada en la sección *NMRELATIONS*

Tabla 24 - Elemento okLink

Etiqueta:	koLink
Elemento de WebRatio:	KO Flow
Atributos comunes**:**	id, name, sourceId, targetId

Atributo específico opcional:

- ❖ type: modifica el comportamiento del flujo.
  - o Valores permitidos:
    - ▪ *"noCoupling"*: desactiva el acoplamiento automático.

Localización:

Se puede definir dentro de:

- La sección *LINKS*
- La subsección *ALL* situada en la sección *RELATIONS*
- La subsección *LAST* situada en la sección *RELATIONS*
- La subsección *ALL* situada en la sección *NMRELATIONS*
- La subsección *FIRST* situada en la sección *NMRELATIONS*
- La subsección *REMAINING* situada en la sección *NMRELATIONS*
- La subsección *LAST* situada en la sección *NMRELATIONS*

Tabla 25 - Elemento koLink

- **Control de flujo:**

Etiqueta:	isNotNullUnit
Elemento de WebRatio:	Is Not Null
Atributos comunes**:**	id, name, x, y
Atributos específicos opcionales: No tiene	
Localización: Se puede definir dentro de: - La sección *OUTSIDEUNITS*	

Tabla 26 - Elemento isNotNullUnit

- **Componentes de utilidad:**

Etiqueta:	selectorUnit
Elemento de WebRatio:	Selector
Atributos comunes**:**	id, name, sourceId, targetId
Atributo específico opcional: ❖ type: modifica el comportamiento de la unidad. o Valores permitidos: ▪ *"keyCondition"*: crea una *keyCondition* en la unidad. ▪ *"roleCondition"*: crea una *roleCondition* en la unidad.	
Localización: Se puede definir dentro de: - El elemento *page* - La sección *OUTSIDEUNITS* - La subsección *ALL* situada en la sección *RELATIONS* - La subsección *ALL* situada en la sección *NMRELATIONS*	

Tabla 27 - Elemento selectorUnit

5.1.3 Unicidad de elementos con relación

Como se ha explicado en *un apartado anterior*, las secciones y subsecciones referentes a relaciones se ejecutarán una vez por cada relación seleccionada en la Interfaz de Usuario[13]. Es decir, se crearán los elementos definidos una vez por cada relación.

Para resolver la repetición de los valores definidos en cada atributo de estos elementos y evitar problemas de duplicidad, se definen una serie de marcadores (contadores). De esta forma se asegura la unicidad de cada elemento dentro del patrón.

- **Tipos:**

- Contador de relación (#): el contador de relación está identificado por el símbolo de la almohadilla #. Representa el número de la relación que se está procesando, es decir, se sustituirá por un número iterativo que comienza en 0 e irá aumentando con cada relación, sea del tipo que sea. Se puede utilizar en todas las subsecciones tanto de la sección *RELATIONS* como de la sección *NMRELATIONS*.

- Contador de relación N:M *(%)*: el contador de relación N:M está identificado por el símbolo de tanto por ciento %. Representa el número de la relación N:M que se está procesando, es decir, se sustituirá por un número iterativo que comienza en 0 e irá aumentando con cada relación de tipo N:M que se procese. Se puede utilizar en todas las subsecciones de la sección *NMRELATIONS*.

Una relación N:M aumenta ambos contadores (# y %).

- **Reglas:**

1. Cada contador se utiliza dentro de las secciones indicadas en su definición (ver apartado anterior).
2. Los contadores se deben situar al final del valor del atributo.

[13] Exceptuando la subsección *LAST que* se ejecutará una única vez al final de todas las relaciones

3. Los contadores se pueden utilizar únicamente en los atributos comunes *(ver Tabla 8 - Atributos comunes)*.

   a. Para los atributos *id*, *name*, *sourceId* y *targetId* se puede utilizar la expresión "*valorAtributo#-1*" o "*valorAtributo%-1*" para hacer referencia al anterior elemento procesado, o bien la expresión "*valorAtributo#+1*" o "*valorAtributo%+1*" para hacer referencia al siguiente elemento.

   b. Para los atributos *x* e *y* que representan las coordenadas no hace falta representar ninguna operación como en el apartado anterior. Únicamente se escribe el contador al final, seguido del número de la coordenada. La operación se realiza internamente, se suma el valor proporcionado al contador multiplicado por una separación estándar que se ha fijado con un valor de 120. ($Valor\ final\ =\ valor\ indicado\ +\ 120 * contador$)

- **Ejemplos:**

Estado actual	**Significado**
Segunda relación	# = 1
Primera relación N:M	% = 0
Sintaxis	**Significado**
id="selector#"	id="selector1"
name="selector%"	name="selector0"
sourceId="selector#-1"	sourceId="selector0"
targetId="selector%+1"	targetId="selector1"
x="300#"	x="420"
y="0%"	y="0"

Tabla 28 - Ejemplos Unicidad de elementos con relación

Se recomienda revisar la sección *1.3 Patrones de ejemplo* en la sección *ANEXO I: Manual de usuario*, poniendo especial atención en las relaciones entre unidades que utilizan los identificadores y los contadores.

5.2 Analizador del patrón

Una vez que se ha descrito el formato y la información que se desea guardar, se planea desarrollar un intérprete que procese dicha información con la finalidad de generar en el modelo web aquellos elementos que han sido descritos en el archivo externo que representa el patrón.

Se necesita detallar el proceso que seguirá la información, así el intérprete se diseña de la forma más óptima, preparando los datos para su posterior utilización.

Siguiendo con esta idea, se realiza un análisis del proceso de generación de la aplicación. El siguiente esquema gráfico lo representa:

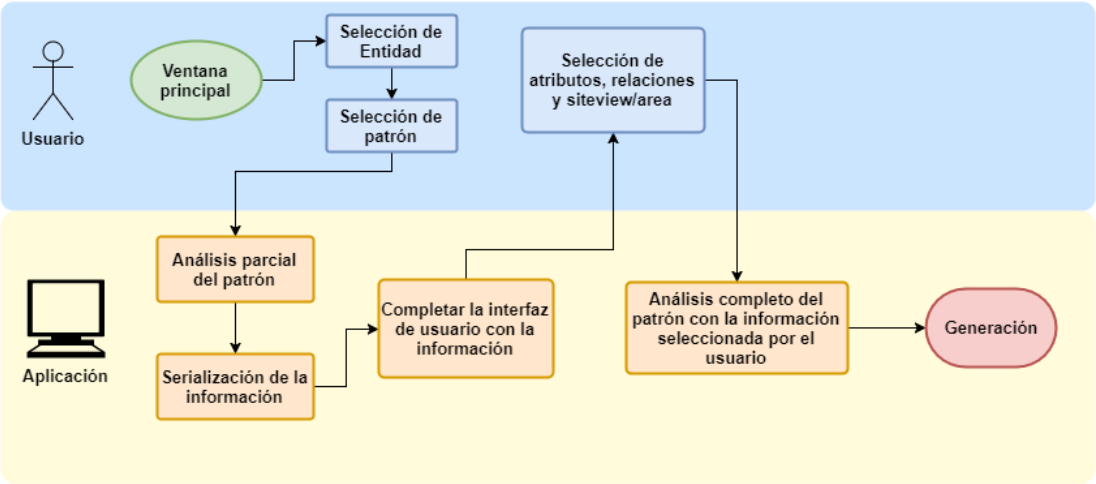

Ilustración 10: Proceso de generación (simple)

Se obtiene la conclusión de que no se puede analizar el patrón y generarlo directamente porque se necesita que el usuario complete la información seleccionando los atributos y las relaciones en la interfaz de usuario. Por lo que el análisis del patrón se debe realizar en dos fases:

- La primera con el fin de conocer los elementos definidos y rellenar la interfaz con su información.
- La segunda para analizar el apartado de relaciones con la información seleccionada en la interfaz de usuario.

En este punto entra en juego otro proceso clave además del análisis del patrón: la serialización de la información. Se necesita almacenar la información del patrón entre las dos fases anteriores y prepararla para el proceso final: la generación. Este proceso se explica en uno de los siguientes subapartados *(ver 5.2.2 Interpretación y serialización de la información).* Sin embargo, antes de realizar el proceso mencionado, hay que asegurarse de que el usuario ha confeccionado el patrón correctamente, sin errores sintácticos, con los campos obligatorios completos, etc. Esta validación del patrón es previa a su procesamiento e interpretación y se describe en el siguiente subapartado.

5.2.1 Validación: XML Schema

Durante la primera lectura del archivo que representa el patrón, se realiza un proceso de validación utilizando el estándar *XML Schema*[14].

XML Schema es un lenguaje de esquema utilizado para describir la estructura y las restricciones de los contenidos de los documentos XML de una forma muy precisa, más allá de las normas sintácticas impuestas por el propio lenguaje XML. Fue desarrollado por el *World Wide Web Consortium*[15] y alcanzó el nivel de recomendación en mayo de 2001.

La complejidad de utilizar esta tecnología radica únicamente en la definición del esquema. A cambio, se alivia al desarrollador del primer preprocesamiento del archivo en busca de fallos en su definición, cuya tarea suele resultar bastante tediosa y, en ocasiones, compleja.

El esquema definido para la validación mediante el método explicado anteriormente se encuentra en el archivo *schema.xsd* que se adjunta en el CD entregado con esta memoria y que también podrá ver en el siguiente subapartado.

[14] www.w3.org/TR/xmlschema-0/
[15] www.w3.org/

5.2.1.1 schema.xsd

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<!-- Declaration -->

    <!-- Elements -->
    <xs:element name="page" type="pageType"/>

    <xs:element name="selectorUnit">
        <xs:complexType>
            <xs:attributeGroup ref="stdUnitAttributes"/>
            <xs:attribute name="type">
                <xs:simpleType>
                    <xs:restriction base="xs:token">
                        <xs:enumeration value="keyCondition"/>
                        <xs:enumeration value="roleCondition"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="parentId" type="xs:token"/>
        </xs:complexType>
    </xs:element>

    <xs:group name="contentUnits">
        <xs:choice>
            <xs:element name="powerIndexUnit" type="stdUnitAttributesType"/>
            <xs:element name="dataUnit" type="stdUnitAttributesType"/>
            <xs:element name="entryUnit">
                <xs:complexType>
                    <xs:attributeGroup ref="stdUnitAttributes"/>
                    <xs:attribute name="type">
```

```
                    <xs:simpleType>
                        <xs:restriction base="xs:token">
                            <xs:enumeration value="preloaded"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:attribute>
            </xs:complexType>
        </xs:element>
        <xs:element name="multiMessageUnit" type="stdUnitAttributesType"/>
        <xs:element name="noOpContentUnit" type="stdUnitAttributesType"/>
        <xs:element ref="selectorUnit"/>
    </xs:choice>
</xs:group>

<xs:group name="links">
    <xs:choice>
        <xs:element name="okLink" type="okKoType"/>
        <xs:element name="koLink" type="okKoType"/>
        <xs:element name="normalNavigationFlow">
            <xs:complexType>
                <xs:attributeGroup ref="stdLinkAttributes"/>
                <xs:attribute name="validate" type="xs:boolean"/>
                <xs:attribute name="type">
                    <xs:simpleType>
                        <xs:restriction base="xs:token">
                            <xs:enumeration value="isNotNull"/>
                            <xs:enumeration value="fixedValue"/>
                            <xs:enumeration value="entryToCreate"/>
                            <xs:enumeration value="entryToUpdate"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:attribute>
            </xs:complexType>
        </xs:element>
        <xs:element name="dataFlow">
            <xs:complexType>
```

```
                    <xs:attributeGroup ref="stdLinkAttributes"/>
                    <xs:attribute name="type">
                        <xs:simpleType>
                            <xs:restriction base="xs:token">
                                <xs:enumeration value="preload"/>
                                <xs:enumeration value="entryToConnect"/>
                                <xs:enumeration value="entryToReconnect"/>
                                <xs:enumeration value="unitToEntry"/>
                                <xs:enumeration value="unitToEntryRole"/>
                                <xs:enumeration value="entryToCreate"/>
                                <xs:enumeration value="entryToUpdate"/>
                            </xs:restriction>
                        </xs:simpleType>
                    </xs:attribute>
                </xs:complexType>
            </xs:element>
        </xs:choice>
    </xs:group>

    <!-- Attributes -->

    <xs:attribute name="id">
        <xs:simpleType>
            <xs:restriction base="xs:token">
                <xs:minLength value="1"/>
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="name" type="xs:token"/>

    <!-- AttributeGroups -->
    <xs:attributeGroup name="stdLinkAttributes">
        <xs:attribute ref="id" use="required"/>
        <xs:attribute ref="name"/>
        <xs:attribute name="sourceId" type="xs:token" use="required"/>
        <xs:attribute name="targetId" type="xs:token" use="required"/>
```

```
</xs:attributeGroup>

<xs:attributeGroup name="stdUnitAttributes">
    <xs:attribute ref="id" use="required"/>
    <xs:attribute ref="name"/>
    <xs:attributeGroup ref="coordinates"/>
</xs:attributeGroup>

<xs:attributeGroup name="coordinates">
    <xs:attribute name="x" type="xs:token" use="required"/>
    <xs:attribute name="y" type="xs:token" use="required"/>
</xs:attributeGroup>

<!-- Complex types -->
<xs:complexType name="stdUnitAttributesType">
    <xs:attributeGroup ref="stdUnitAttributes"/>
</xs:complexType>

<xs:complexType name="stdLinkAttributesType">
    <xs:attributeGroup ref="stdLinkAttributes"/>
</xs:complexType>

<xs:complexType name="pagesSectionType">
    <xs:sequence>
        <xs:element ref="page" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="pageType">
    <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element ref="page"/>
        <xs:element name="xor" type="xorType"/>
        <xs:group ref="contentUnits"/>
    </xs:choice>
    <xs:attributeGroup ref="stdUnitAttributes"/>
    <xs:attribute name="landmark" type="xs:boolean"/>
```

```
        <xs:attribute name="default" type="xs:boolean"/>
    </xs:complexType>

    <xs:complexType name="xorType">
        <xs:choice minOccurs="0" maxOccurs="unbounded">
            <xs:element ref="page"/>
            <xs:group ref="contentUnits"/>
        </xs:choice>
        <xs:attributeGroup ref="stdUnitAttributes"/>
    </xs:complexType>

    <xs:complexType name="outsideunitsType">
        <xs:choice minOccurs="0" maxOccurs="unbounded">
            <xs:element name="createUnit" type="stdUnitAttributesType"/>
            <xs:element name="deleteUnit" type="stdUnitAttributesType"/>
            <xs:element name="updateUnit" type="stdUnitAttributesType"/>
            <xs:element name="isNotNullUnit" type="stdUnitAttributesType"/>
            <xs:element ref="selectorUnit"/>
        </xs:choice>
    </xs:complexType>

    <xs:complexType name="okKoType">
        <xs:attributeGroup ref="stdLinkAttributes"/>
        <xs:attribute name="message" type="xs:string"/>
        <xs:attribute name="type">
            <xs:simpleType>
                <xs:restriction base="xs:token">
                    <xs:enumeration value="noCoupling"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
    </xs:complexType>

    <xs:complexType name="linksType">
        <xs:choice minOccurs="0" maxOccurs="unbounded">
            <xs:group ref="links"/>
```

```
        </xs:choice>
    </xs:complexType>

    <xs:complexType name="relSubsectionType">
        <xs:choice minOccurs="0" maxOccurs="unbounded">
            <xs:group ref="links"/>
            <xs:element ref="selectorUnit"/>
            <xs:element name="connectUnit" type="stdUnitAttributesType"/>
            <xs:element name="disconnectUnit" type="stdUnitAttributesType"/>
            <xs:element name="reconnectUnit" type="stdUnitAttributesType"/>
        </xs:choice>
    </xs:complexType>

    <xs:complexType name="relationsType">
        <xs:sequence>
            <xs:element name="ALL" type="relSubsectionType" minOccurs="0"/>
            <xs:element name="LAST" type="linksType" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="nmrelationsType">
        <xs:sequence>
            <xs:element name="ALL" type="relSubsectionType" minOccurs="0"/>
            <xs:element name="FIRST" type="linksType" minOccurs="0"/>
            <xs:element name="REMAINING" type="linksType" minOccurs="0"/>
            <xs:element name="LAST" type="linksType" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>

<!-- Root definition -->

    <xs:element name="PATTERN">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="PAGES" type="pagesSectionType" minOccurs="0"/>
                <xs:element name="OUTSIDEUNITS" type="outsideunitsType" minOccurs="0"/>
```

```
                <xs:element name="LINKS" type="linksType" minOccurs="0"/>
                <xs:element name="RELATIONS" type="relationsType" minOccurs="0"/>
                <xs:element name="NMRELATIONS" type="nmrelationsType" minOccurs="0"/>
            </xs:sequence>
            <xs:attribute ref="id" use="required"/>
            <xs:attribute ref="name" use="required"/>
        </xs:complexType>

        <xs:unique name="uniqueID">
            <xs:selector xpath=".//*" />
            <xs:field xpath="@id" />
        </xs:unique>
    </xs:element>

</xs:schema>
```

5.2.2 Interpretación y serialización de la información

Una vez validado el patrón, la información se almacena en una estructura en árbol de tipo DOM (Document Object Model)[16], lista para su interpretación.

La interpretación consiste en: para cada elemento procesado del árbol, conocer el tipo de elemento que estamos tratando, obtener los atributos de ese elemento y resolver la dependencia con el padre, si la hubiera, guardando su identificador. Este proceso se complica en el apartado de relaciones, debido a la elevada casuística que se puede producir dependiendo de las relaciones que ha seleccionado el usuario en la interfaz y de la cardinalidad de estas relaciones:

[16] https://www.w3.org/DOM/

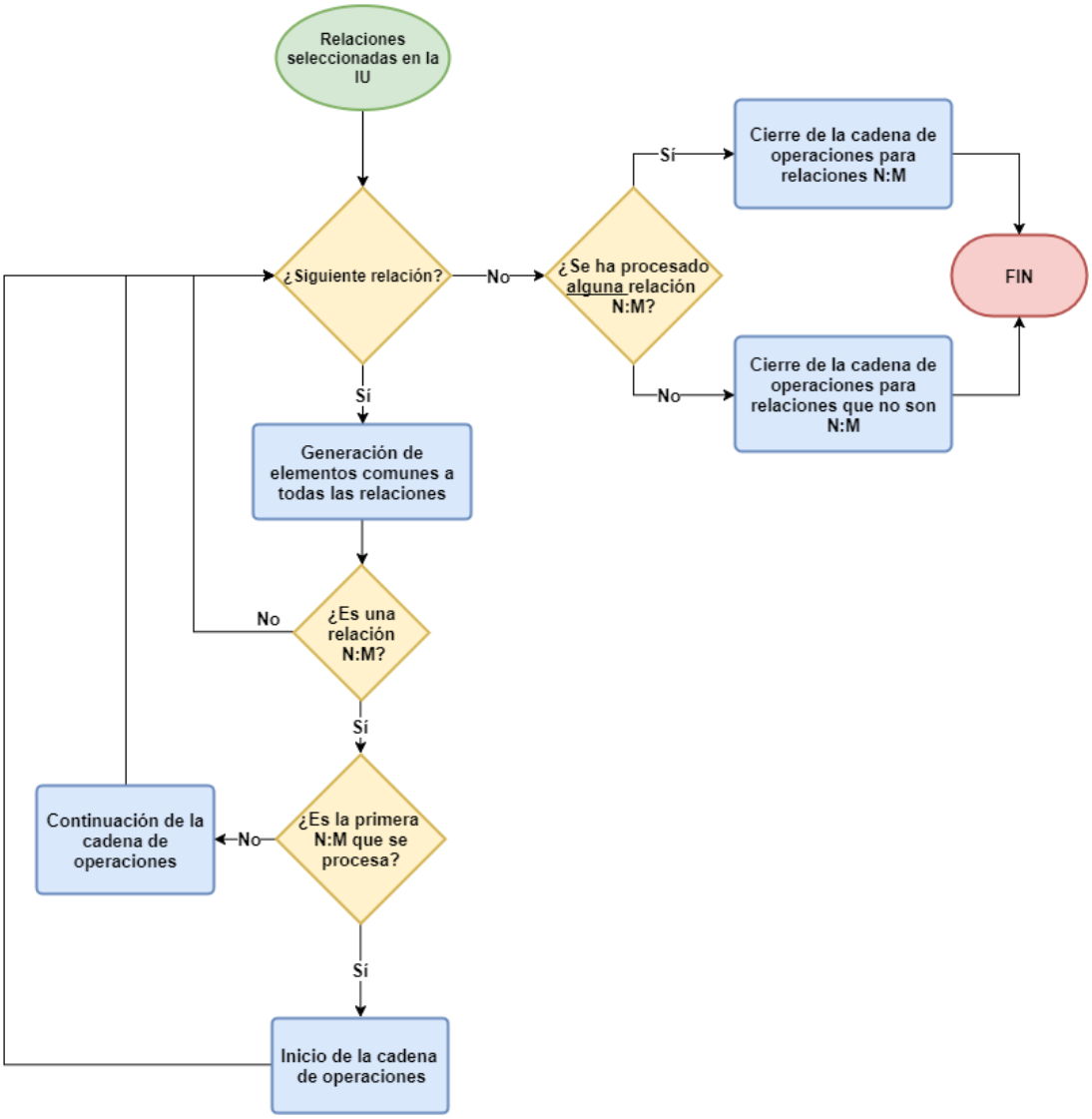

Ilustración 11: Diagrama de flujo. Procesamiento del patrón: sección de relaciones

Toda esa información se debe clasificar y guardar para utilizarla en el proceso de generación.

En el apartado Elementos, se realizó un análisis de los elementos y atributos y se comprobó que existen elementos con atributos comunes. Esta información, además de ser válida para definir la estructura del patrón, resulta útil para diseñar cómo almacenar la información.

Se debe tener en cuenta la acción común que realizan todos los elementos: la generación, aunque cada elemento la realiza de una forma diferente.

Finalmente, se llega a una conclusión acorde con las características de la información y al proceso que se realiza: implementar una jerarquía de

clases para almacenar la información. Los nodos hoja de esta jerarquía son los elementos que se instanciarán para almacenar la información. Las clases intermedias son clases abstractas utilizadas para definir los atributos comunes y crear grupos de elementos con el mismo tipo de información. La raíz también es una clase abstracta que define los atributos que son comunes a todos los elementos, además de especificar la acción común que deben definir sus hijos: la generación. *(Ver: Ilustración 23: Diagrama de clases. Jerarquía del paquete element)*

5.3 Generación

La generación consiste en crear los elementos IFML en el modelo de WebRatio con la información que hemos obtenido del patrón y que se ha explicado en los apartados anteriores. Estos elementos se sitúan en el modelo siguiendo las reglas de asociación descritas en el mismo patrón.

El usuario controla este proceso. Puede elegir entre generar un elemento cada vez que pulse un botón o generar todos los elementos restantes de manera consecutiva si pulsa otro botón. En cualquier caso, se muestra información del progreso realizado en la misma ventana.

5.4 Registro

Todos los elementos que han sido generados por el asistente se registran un archivo llamado "*Registry"* situado en el directorio donde se encuentran los patrones. Este archivo no requiere de la intervención del usuario, se crea y se actualiza automáticamente en el momento de la generación.

Siguiendo la línea de desarrollo del proyecto, el archivo utiliza el lenguaje XML para almacenar la información del registro.

La estructura del archivo es la siguiente:

Cada patrón generado se representa con un nodo nuevo, almacenando su identificador y su nombre. En el interior, se crea un nodo por cada *SiteView* sobre el que opera el patrón, almacenando también su nombre e identificador. Es en este tercer nivel, en el interior del nodo que representa el *SiteView*, donde se crea una nueva entrada por cada elemento generado en el patrón. Se almacena el tipo de elemento y su identificador final.

La siguiente ilustración muestra un diagrama de la estructura del registro:

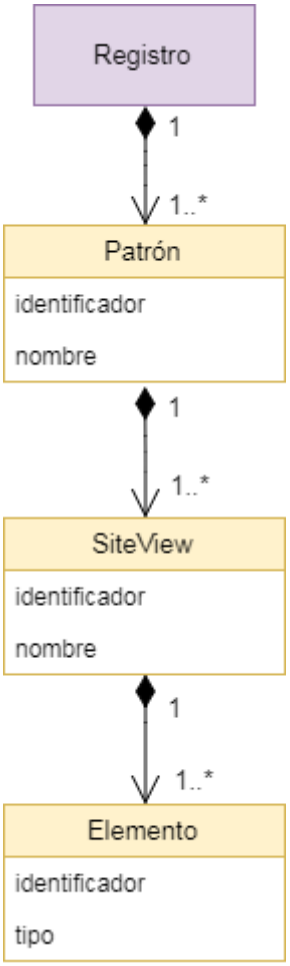

Ilustración 12: Diagrama de la estructura del registro

Al final de la memoria, se proporciona un ejemplo del archivo de registro *(ver: 0*

Ejemplo del archivo Registry en la sección

ANEXO II: Manual del Programador*)*

Toda la información almacenada en el registro se muestra en una nueva ventana cuando el usuario pulsa el botón correspondiente de la interfaz. Esta información no sólo se compone de los datos del registro sino de una serie de datos estadísticos generados a partir de dicha información.

Los datos estadísticos permiten al usuario hacerse una idea del número de patrones que se han generado en el proyecto web actual, el número de elementos que contiene el proyecto entero o sólo de un patrón generado, además de los SiteViews utilizados.

5.5 Interfaz de usuario

La interfaz de usuario ha sufrido diversas modificaciones respecto a la versión anterior, tal y como se ha enunciado en el punto *4. Punto de partida*.

En este apartado se profundiza en los aspectos de diseño que, junto con las decisiones tomadas, dan lugar a la interfaz de usuario final de la aplicación.

5.5.1 Diálogo principal

En la siguiente ilustración se marcan con un recuadro numerado los cuatro componentes fundamentales del diálogo principal de la aplicación:

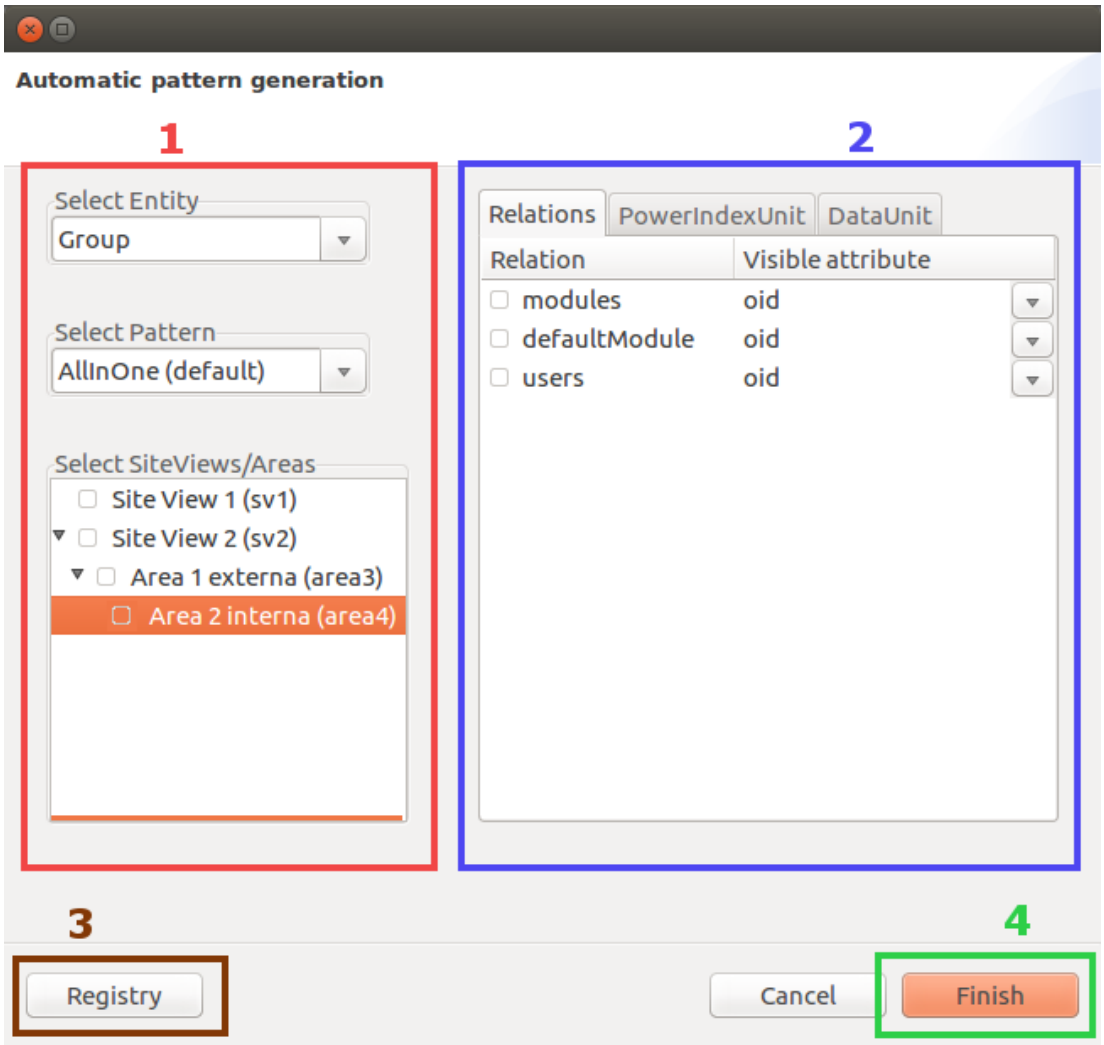

Ilustración 13: Interfaz de usuario. Diálogo principal

1) El recuadro número uno es un grupo estático de componentes. Su contenido es el siguiente:
   a) Selector de entidad: contiene todas las entidades del modelo de dominio.
   b) Selector de patrón: contiene todos los patrones que se localizan en el directorio destinado a ellos. Cada entrada representa un patrón diferente. (Ver 1.2 Localización de los archivos en el ANEXO I: Manual de usuario)
   c) Árbol de *SiteViews* y *Areas*: donde el usuario puede elegir dónde se generan los elementos. Se pueden elegir varias opciones simultáneas.

2) El segundo es un conjunto dinámico de componentes. Su contenido varía dependiendo de los elementos definidos en el patrón elegido, aunque siempre sigue la misma estructura de pestañas. Posibilidades de contenido:
   a) Una pestaña que contiene las relaciones de la entidad seleccionada. Este componente se muestra cuando se definen en el patrón unidades que necesiten las relaciones (p.e. CreateUnit o UpdateUnit)
   b) Una pestaña por cada unidad PowerIndexUnit o DataUnit definida en el patrón. Cada una de ellas contiene los atributos de la entidad seleccionada.

3) Botón que permite visualizar el registro. Al pulsar en él se abre una nueva ventana con los datos del registro y las estadísticas. En el *apartado siguiente* se detallan sus componentes gráficos.

4) Botón con el que se finaliza la adquisición de requisitos por parte del usuario y se comienza la fase de generación. Se cierra el diálogo principal y se abre uno nuevo en el que el usuario puede controlar el proceso. A continuación, en el apartado *5.5.3 Diálogo de generación,* se describen sus componentes gráficos.

5.5.2 Ventana del registro

La ventana del registro es la que muestra toda la información registrada durante la generación de los patrones.

La siguiente ilustración es un ejemplo tras generar varios patrones:

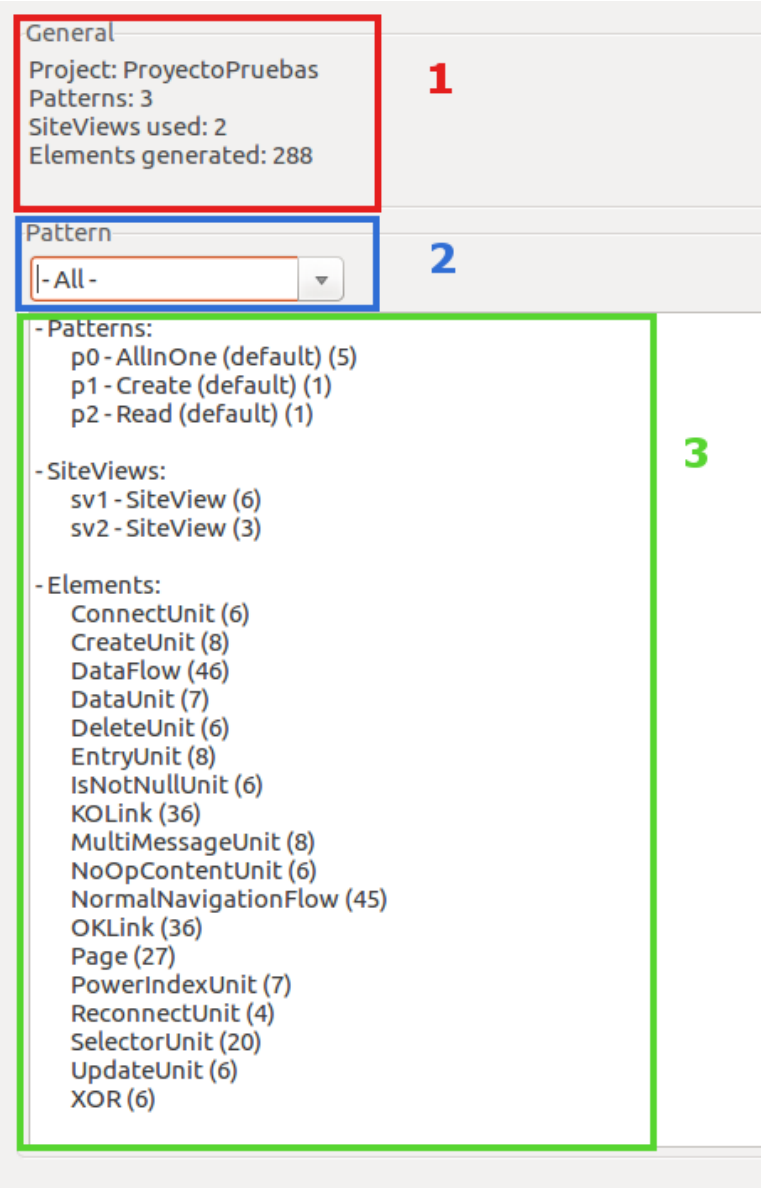

Ilustración 14: Interfaz de usuario. Ventana del registro

1) La primera sección contiene información general del proyecto junto con las estadísticas generales producidas.

2) El segundo recuadro contiene el selector que permite elegir entre todos los patrones registrados o alguno específico.

3) El tercer y último cuadro es un componente de tipo textual. Su contenido es modificado dinámicamente al cambiar la elección del selector descrito anteriormente. Contiene información más detallada sobre la selección realizada.

5.5.3 Diálogo de generación

El diálogo de generación es una ventana que permite al usuario controlar la velocidad de generación y visualizar el progreso.

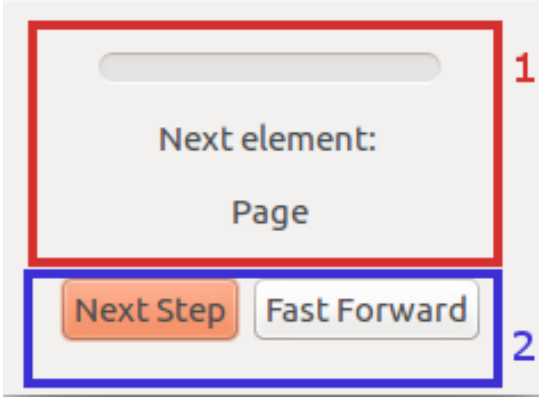

Ilustración 15: Interfaz de usuario. Diálogo de generación

1) El primer grupo de componentes tiene la función de mantener informado al usuario en todo momento del proceso de generación. Contiene una barra de progreso y una etiqueta que muestra la siguiente unidad que se va a generar.

2) El segundo grupo contiene los botones que permiten al usuario interactuar con la aplicación, indicando la velocidad de generación.
    a. Botón *Next Step*: genera el siguiente elemento.
    b. Botón *Fast Forward:* genera todos los elementos restantes.

En caso de no haber elementos restantes que generar, esta ventana de diálogo se cierra poniendo fin a la ejecución de la aplicación.

6 Implementación y desarrollo

Durante todo el desarrollo del proyecto se ha utilizado la herramienta de control de versiones Git[17]. El repositorio está disponible en GitHub en la siguiente URL: https://github.com/JavierSierraBlazquez/AutoCRUD

6.1 Análisis y Arquitectura de la aplicación

6.1.1 Diagramas de flujo

A continuación, se muestran los diagramas de flujo de trabajo de la aplicación. Abarca todas las acciones que el usuario puede realizar y el proceso que desarrolla la aplicación para cada una de ellas.

Para facilitar la lectura y el entendimiento de esta sección, primero se muestra un diagrama de alto nivel donde se han agrupado las acciones comunes que constituyen un proceso. Posteriormente, se analiza cada proceso de forma individual.

[17] https://git-scm.com/

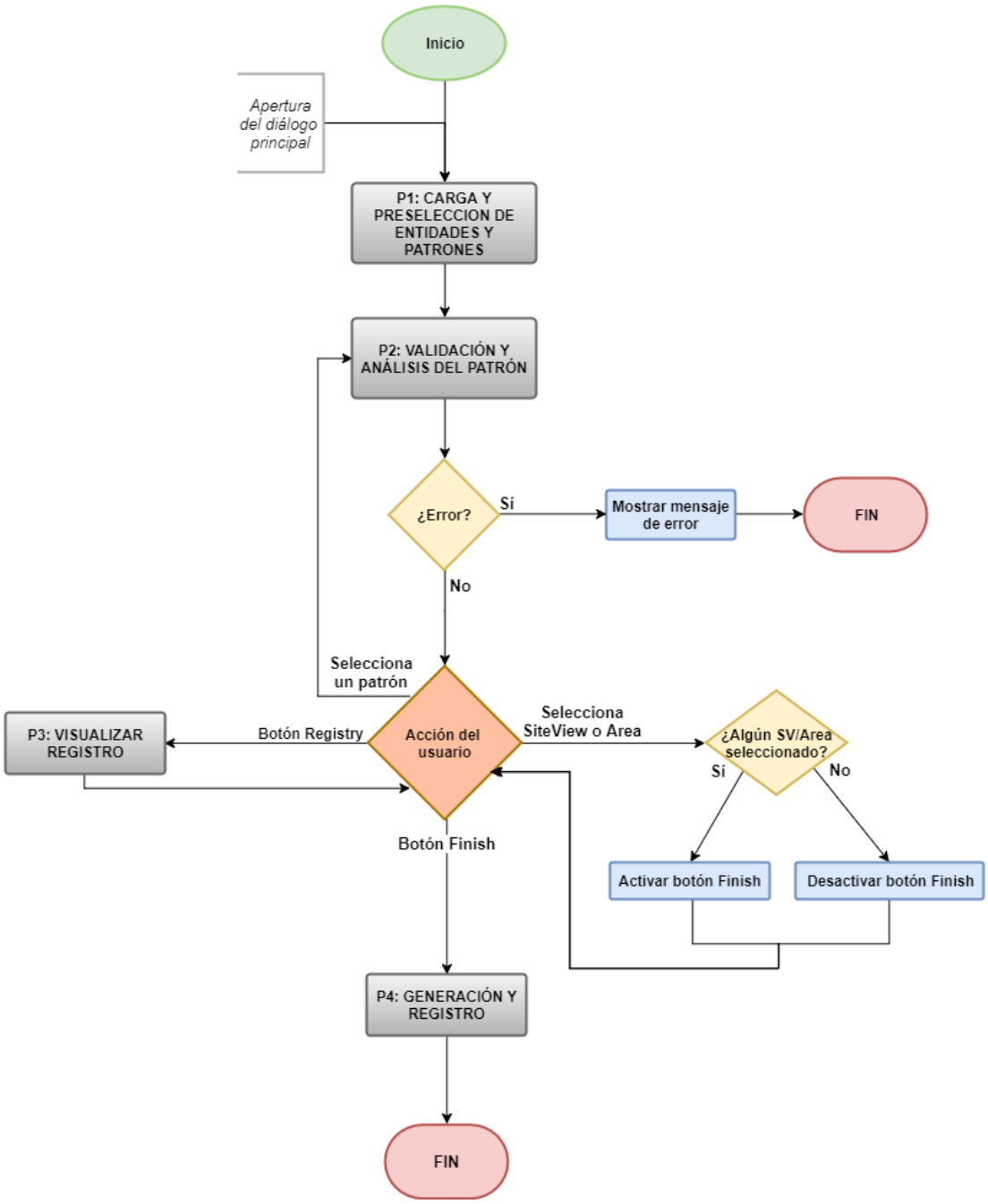

Ilustración 16: Diagrama de flujo. Vista completa del flujo de la aplicación

6.1.1.1 Proceso 1: Carga y preselección de entidades y patrones

Proceso que se ejecuta cuando se inicia el plug-in. La aplicación automáticamente carga las entidades del modelo que corresponden al proyecto web sobre el que estamos trabajando actualmente. Seguidamente, se localizan los patrones y disponibles y se preselecciona el primer patrón encontrado.

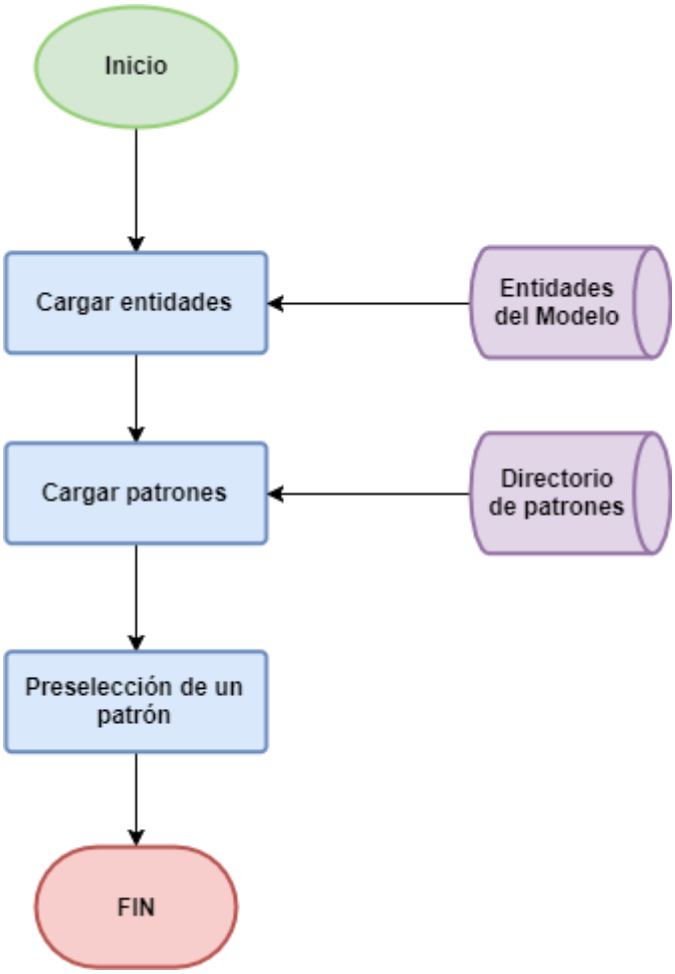

Ilustración 17: Diagrama de flujo. Proceso 1: Carga y preselección de entidades y patrones

6.1.1.2 Proceso 2: Validación y análisis del patrón

Este proceso recibe el patrón que se ha seleccionado previamente. El patrón se valida en cuanto a su estructura y contenido. Se procesa su contenido, se interpreta y se almacena la información para ser utilizada en procesos posteriores.

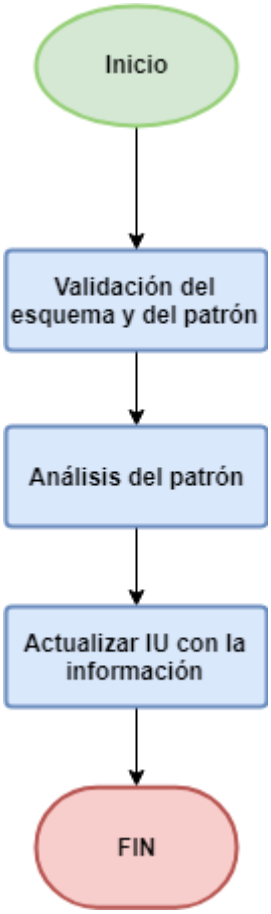

Ilustración 18: Diagrama de flujo. Proceso 2: Validación y análisis del patrón

6.1.1.3 Proceso 3: Visualizar registro

Todas las acciones que se realizan en este proceso se desarrollan sobre una nueva ventana de diálogo que se abre en su inicio. Esta ventana se completa con la información del registro que ha sido convenientemente analizada. El usuario puede aplicar filtros en la información del registro o cerrar la ventana de diálogo volviendo al diálogo principal.

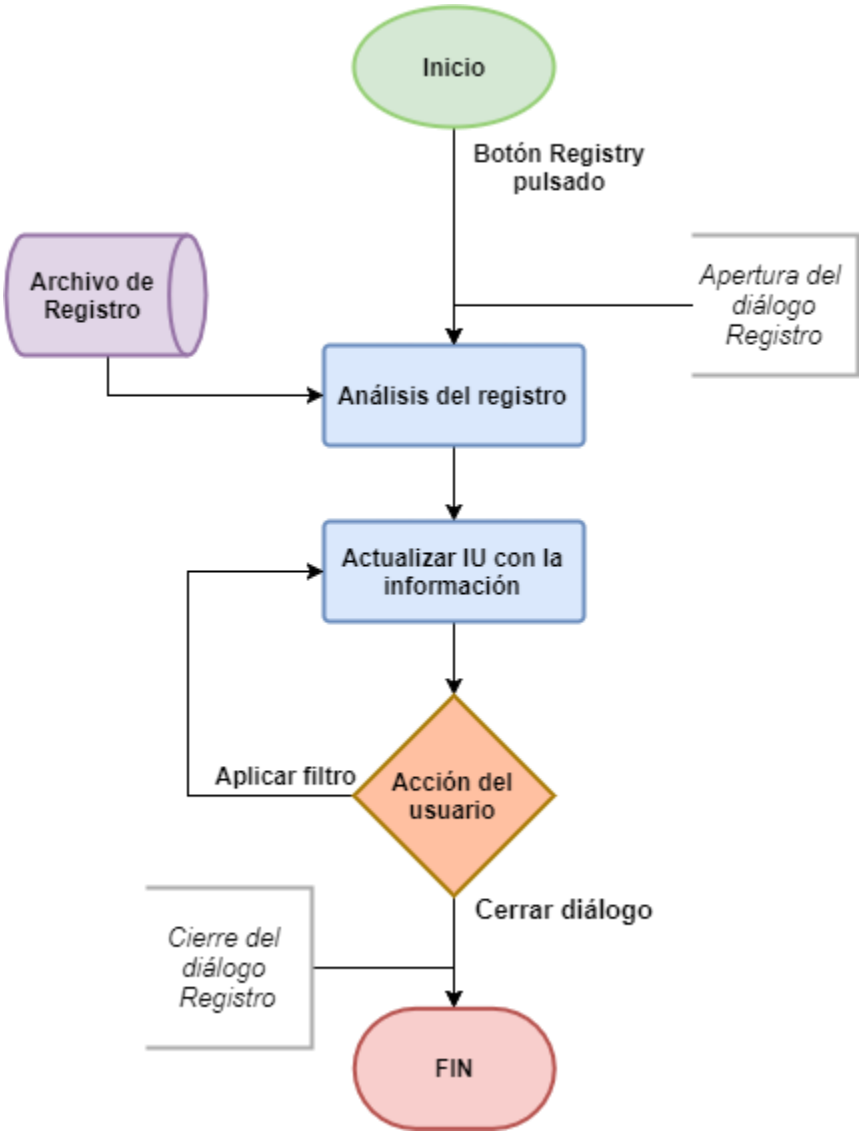

Ilustración 19: Diagrama de flujo. Proceso 3: Visualizar registro

6.1.1.4 Proceso 4: Generación y registro de elementos

Proceso que se inicia cuando el usuario pulsa el botón de finalización en el diálogo principal.

Se obtienen las opciones seleccionadas en la interfaz de usuario. Esta información se procesa junto con la información previamente almacenada del patrón. Se cierra la ventana de diálogo principal y se abre una nueva ventana que muestra el proceso de generación, ofreciendo al usuario dos opciones:

1. Botón Next: genera el siguiente elemento, situándolo en el modelo y actualiza el progreso en la interfaz de usuario, esperando una nueva acción del usuario.
2. Botón Fast forward: genera todos los elementos restantes, los coloca en el modelo según se van generando mientras se actualiza la interfaz de usuario con el progreso.

Una vez generados todos los elementos restantes, se ha completado el proceso y se cierra la ventana de diálogo poniendo fin a la ejecución de la aplicación.

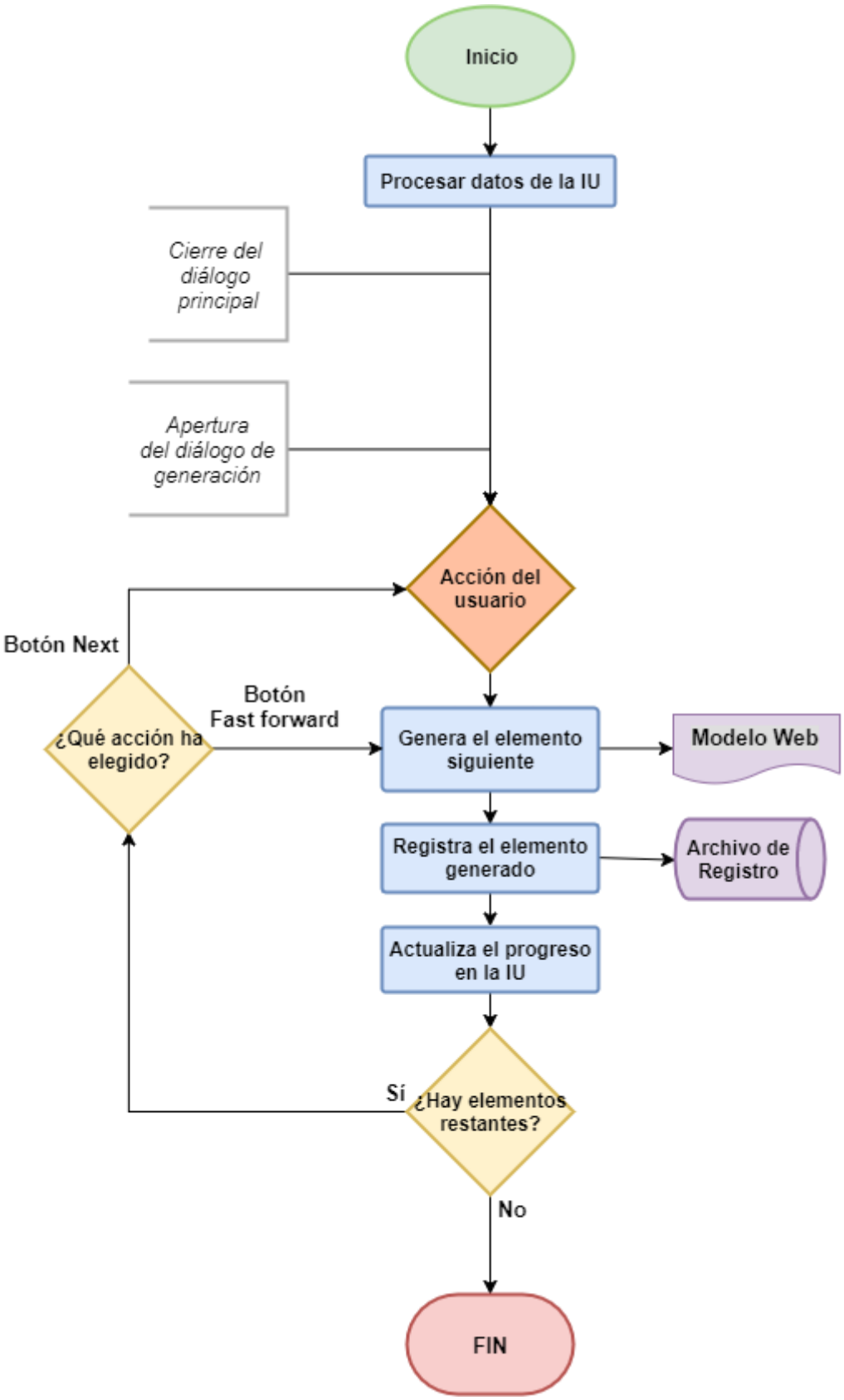

Ilustración 20: Diagrama de flujo. Proceso 4: Generación y registro de elementos

6.1.2 Clases

Para desarrollar la solución a los objetivos propuestos, se continúa utilizando el paradigma de Programación Orientada a Objetos, tal y como se venía realizando en el proyecto anterior.

Aunque se han creado nuevas clases para implementar las funcionalidades nuevas, se siguen utilizando algunas de la solución anterior con o sin modificaciones. Otras han sido eliminadas porque ya no son necesarias o se ha reutilizado su contenido en otras partes de la aplicación.

Sea como fuere, en los siguientes apartados se explican las modificaciones realizadas junto con la especificación de todo el conjunto de clases utilizadas en la solución final.

6.1.2.1 Modificación de clases del proyecto anterior

- **org.homeria.webratioassistant.handler**

El paquete *handler* ha sido eliminado por la refactorización general del proyecto agrupando funcionalidades.

Su contenido:

- lanzarCRUD.java se renombra a Launcher.java y se mueve al paquete principal.

- **org.homeria.webratioassistant.crud**

El paquete *crud* se ha eliminado. Todas sus clases contienen los comandos para generar los elementos en la secuencia que formaban los diferentes patrones. Todas ellas se han eliminado y su código se ha fragmentado en las clases del paquete elements, otorgando la capacidad de generar elementos de forma aislada y no en secuencia.

Su contenido:

- AllInOne.java, Create.java, CRUD.java, Delete.java, Retrieve.java y Update.java han sido eliminadas por lo explicado anteriormente.

- **org.homeria.webratioassistant.plug-in**

El paquete *plug-in* se ha renombrado a *webratio* con el fin de identificar mejor su funcionalidad, puesto que su contenido contiene llamadas a las librerías de WebRatio. Las clases han sido renombradas cumpliendo así parte del *Objetivo 5*, concretamente el subobjetivo: *Traducir variables y documentación de la versión anterior.*

Su contenido:

- EventoNuevaAlternantiva.java se ha renombrado a: NewAlternative.java
- EventoNuevaArea.java se ha renombrado a: NewArea.java
- EventoNuevoLink.java se ha renombrado a: NewLink.java
- EventoNuevaPagina.java se ha renombrado a: NewPage.java
- EventoNuevaSelector.java se ha renombrado a: NewSelector.java
- EventoNuevaUnit.java se ha renombrado a: NewUnit.java
- Evento.java se ha renombrado a: WebRatioCalls.java
- EventoNuevaSiteView.java se ha eliminado por la supresión de la funcionalidad que crea un nuevo SiteView
- ObjStViewArea.java se ha eliminado porque no era necesario. El cometido que cumple se puede realizar de una forma más eficiente.

- **org.homeria.webratioassistant.wizards**

Se mantiene el paquete, pero las clases son modificadas.

Su contenido:

- WizardCRUD.java se modifica el objetivo final de la clase, ahora se dedica a crear las páginas, administrar el asistente gráfico y almacenar la información recogida en la interfaz de usuario para que el nivel superior (Launcher.java) la recoja y comience la generación.
- WizardCRUDPreviaPage.java se ha eliminado por la supresión de la funcionalidad que crea un nuevo SiteView
- WizardSelectEntityPage.java se ha eliminado por la unificación de la selección de entidad con el resto de opciones en una misma página.

- WizardCRUDPage.java se ha renombrado a WizardPatternPage.java. Parte de su código se sigue utilizando con modificaciones, el resto se ha eliminado dando paso a código nuevo.

6.1.2.2 Clases utilizadas en la solución final

A continuación, se muestran todas las clases que constituyen la implementación de la aplicación final.

Las clases nuevas que han sido creadas en este proyecto son todas las contenidas en los paquetes *elements*, *exceptions*, *generation*, *parser* y *registry,* junto con *WizardDialogWithRegistryButton.java* y *WizardPatternPage.java* que pertenecen al paquete *wizards*.

El resto de clases han sido heredadas del proyecto anterior, algunas han sido modificadas tal y como se ha explicado en el *apartado anterior*.

- **org.homeria.webratioassistant**
  - **Activator.java:** inicia el ciclo de vida del plug-in.
  - **Launcher.java:** clase maestra que alberga y ejecuta el ciclo de vida completo del plug-in

- **org.homeria.webratioassistant.elements**
  - **ConnectUnit.java:** almacena los datos que permiten la generación de la ConnectUnit en el modelo web.
  - **CreateUnit.java:** almacena los datos que permiten la generación de la CreateUnit en el modelo web.
  - **DataFlow.java:** almacena los datos que permiten la generación del DataFlow en el modelo web.
  - **DataUnit.java:** almacena los datos que permiten la generación de la DataUnit en el modelo web.
  - **DeleteUnit.java:** almacena los datos que permiten la generación de la DeleteUnit en el modelo web.
  - **DisconnectUnit.java:** almacena los datos que permiten la generación de la DisconnectUnit en el modelo web.

- **ElementTypes.java:** contiene los tipos de elementos que se pueden generar.
- **EntryUnit.java:** almacena los datos que permiten la generación de la EntryUnit en el modelo web.
- **IsNotNullUnit.java:** almacena los datos que permiten la generación de la IsNotNullUnit en el modelo web.
- **KOLink.java:** almacena los datos que permiten la generación del KOLink en el modelo web.
- **Link.java:** superclase que contiene información común para todos los flujos.
- **MultiMessageUnit.java:** almacena los datos que permiten la generación de la MultiMessageUnit en el modelo web.
- **NoOpContentUnit.java:** almacena los datos que permiten la generación de la NoOpContentUnit en el modelo web.
- **NormalNavigationFlow.java:** almacena los datos que permiten la generación del NormalNavigationFlow en el modelo web.
- **OKLink.java:** almacena los datos que permiten la generación del OKLink en el modelo web.
- **Page.java:** almacena los datos que permiten la generación de la Page en el modelo web.
- **PowerIndexUnit.java:** almacena los datos que permiten la generación de la PowerIndexUnit en el modelo web.
- **ReconnectUnit.java:** almacena los datos que permiten la generación de la ReconnectUnit en el modelo web.
- **SelectorUnit.java:** almacena los datos que permiten la generación de la SelectorUnit en el modelo web.
- **Unit.java:** superclase que contiene información común para todas las unidades.
- **UnitOutsidePage.java:** superclase que contiene información común para todas las unidades que no están contenidas en ninguna página.
- **UpdateUnit.java:** almacena los datos que permiten la generación de la UpdateUnit en el modelo web.

- **WebRatioElement.java:** superclase que contiene información común a todos los elementos generables.
- **XOR.java:** almacena los datos que permiten la generación de la Alternative Page (XOR Page) en el modelo web.

- **org.homeria.webratioassistant.exceptions**
  - **ExceptionHandler.java:** se utiliza para administrar todas las excepciones se producen en el proyecto. Muestra un mensaje informativo al usuario cuando se produce.
  - **IdNotUniqueException.java:** excepción personalizada que se utiliza para alertar de que un identificador no es único.
  - **NoPatternFileFoundException.java:** excepción personalizada que se utiliza para alertar de la inexistencia de patrones en el directorio.
  - **NoPatternsFolderFoundException.java:** excepción personalizada que se utiliza para alertar de la inexistencia del directorio de patrones

- **org.homeria.webratioassistant.generation**
  - **Generate.java:** Esta clase permite que los elementos WebRatio se generen con los datos proporcionados. También registra los elementos generados.
  - **StepGenerationAppWindow.java:** contiene la información para crear la ventana de diálogo que permite al usuario dirigir la generación.

- **org.homeria.webratioassistant.parser**
  - **PatternParser.java:** clase responsable de leer y analizar los patrones, serializando la información en elementos más manejables utilizando la jerarquía definida en el paquete elements.

- **org.homeria.webratioassistant.registry**
  - **PatternRegisteredPOJO.java:** almacena la información que se obtiene del registro.
  - **Registry.java:** responsable de leer, analizar y mantener el registro.
  - **RegistryDialog.java:** contiene la información para crear la ventana de diálogo que permite al usuario visualizar el registro.

- **org.homeria.webratioassistant.webratio**
  - **Debug.java:** Clase estática que muestra mensajes por la consola en el modo debug.
  - **MyEntityComparator.java:** comparador personalizado utilizado para comparar entidades.
  - **NewAlternative.java:** conjunto de llamadas a las librerías de WebRatio que permiten añadir una página alternativa a un SiteView del modelo web.
  - **NewArea.java:** conjunto de llamadas a las librerías de WebRatio que permiten añadir un Area a un SiteView u otro Area del modelo web.
  - **NewLink.java:** conjunto de llamadas a las librerías de WebRatio que permiten crear links en el modelo web.
  - **NewPage.java:** conjunto de llamadas a las librerías de WebRatio que permiten añadir una página nueva a un SiteView del modelo web.
  - **NewSelector.java:** conjunto de llamadas a las librerías de WebRatio que permiten añadir una SelectorUnit a un SiteView del modelo web.
  - **NewUnit.java:** conjunto de llamadas a las librerías de WebRatio que permiten añadir una nueva unidad a un SiteView del modelo web.
  - **ProjectParameters.java:** clase que almacena variables utilizadas por WebRatio para su correcto funcionamiento.
  - **Utilities.java:** clase estática que reúne varios métodos de diversa utilidad que no tienen cabida en otro lugar.

- **WebRatioCalls.java:** superclase que contiene información común para la generación de los elementos.

- **org.homeria.webratioassistant.webratioaux**
  - **AddLinkCommand.java:** clase auxiliar que forma parte de la generación de un flujo. Crea el comando que será utilizado para la generación del elemento.
  - **AddSelectorConditionCommand.java:** clase auxiliar que crea una condición en la SelectorUnit.
  - **AddUnitCommand.java:** clase auxiliar que forma parte de la generación de una unidad. Crea el comando que será utilizado para la generación del elemento.
  - **CompositeMFCommand.java**
  - **SiteViewProperty.java:** clase que sirve de soporte interno a la implementación del SiteView, utilizada para modular la información del mismo.

- **org.homeria.webratioassistant.wizards**
  - **WizardCRUD.java:** administra el asistente gráfico, creando las páginas y controlando el ciclo de vida.
  - **WizardDialogWithRegistryButton.java:** extensión de WizardDialog permitiendo situar el botón para visualizar el Registro en la parte inferior, como si fuera un botón predeterminado.
  - **WizardPatternPage.java:** página del diálogo principal. En ella se encuentran los elementos gráficos que permiten al usuario interactuar con la aplicación y lanzar los diferentes procesos.

6.1.2.3 Diagramas de clases

Debido a la elevada extensión del diagrama de clases completo, se ha dividido por paquetes.

Primero se muestra el diagrama general con todos los paquetes sin clases y posteriormente cada diagrama de clases del paquete de forma individual.

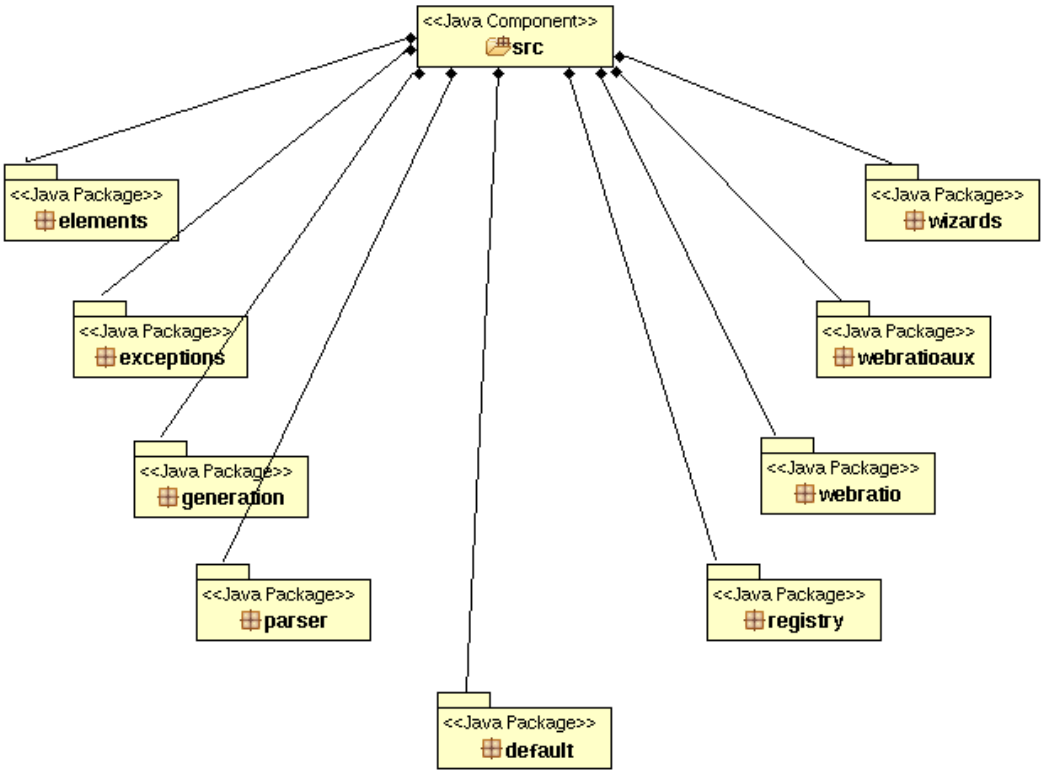

Ilustración 21: Diagrama de paquetes

- **org.homeria.webratioassistant**

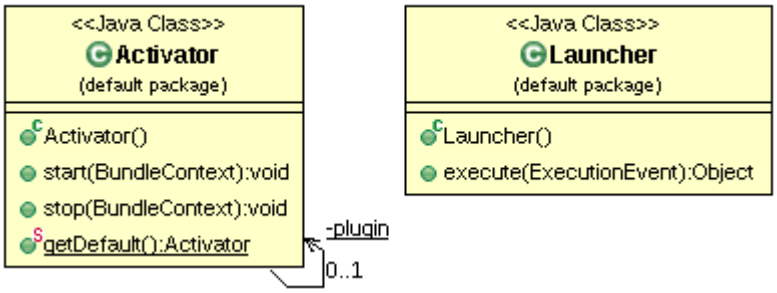

Ilustración 22: Diagrama de clases. Paquete principal

- **org.homeria.webratioassistant.elements**

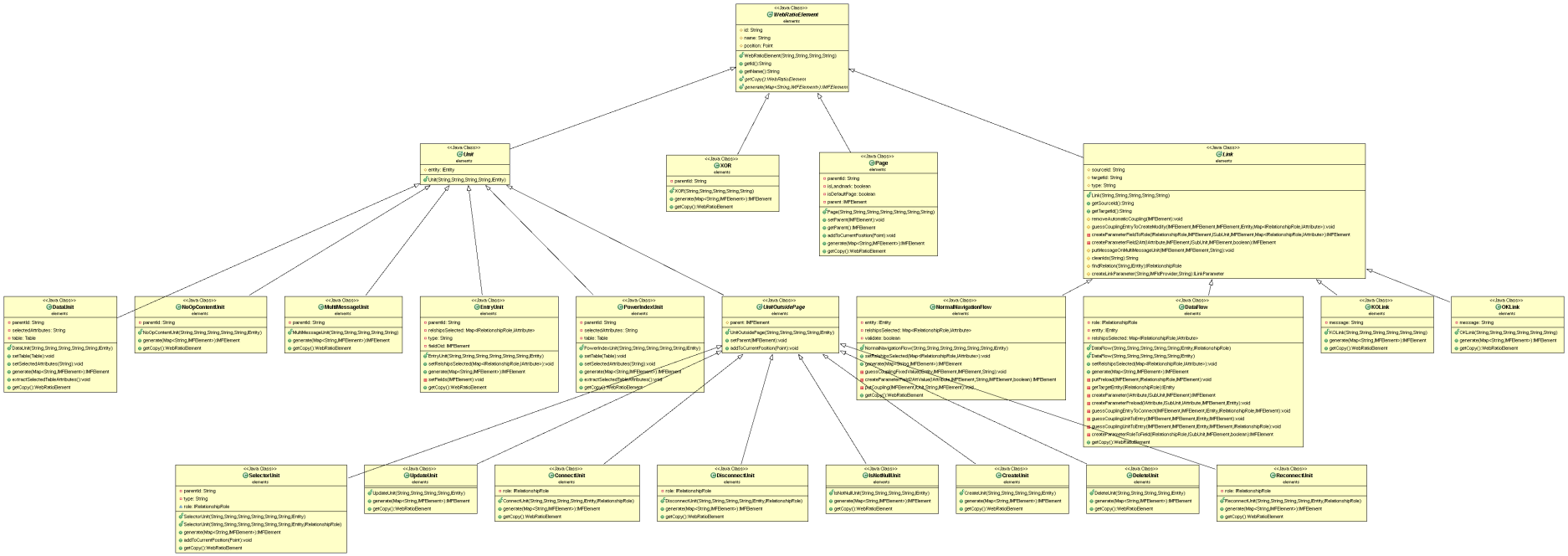

Ilustración 23: Diagrama de clases. Jerarquía del paquete element

- **org.homeria.webratioassistant.exceptions**

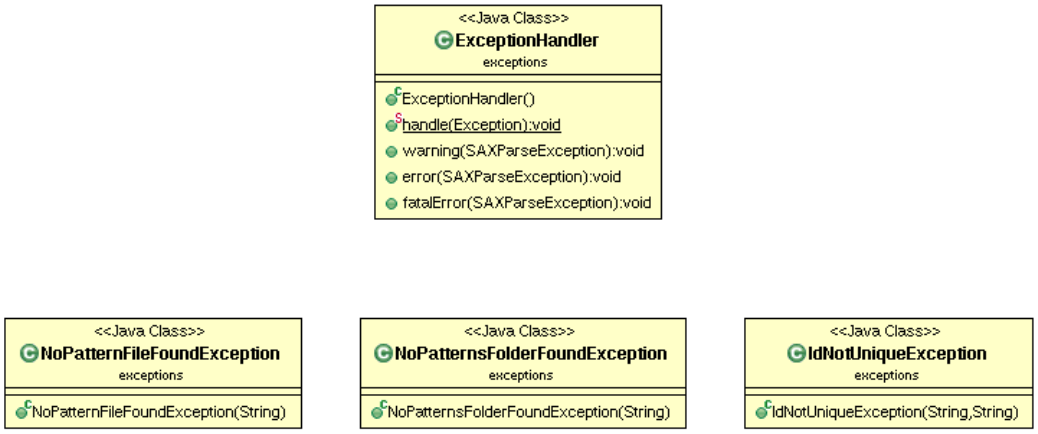

Ilustración 24: Diagrama de clases. Paquete exceptions

- **org.homeria.webratioassistant.generation**

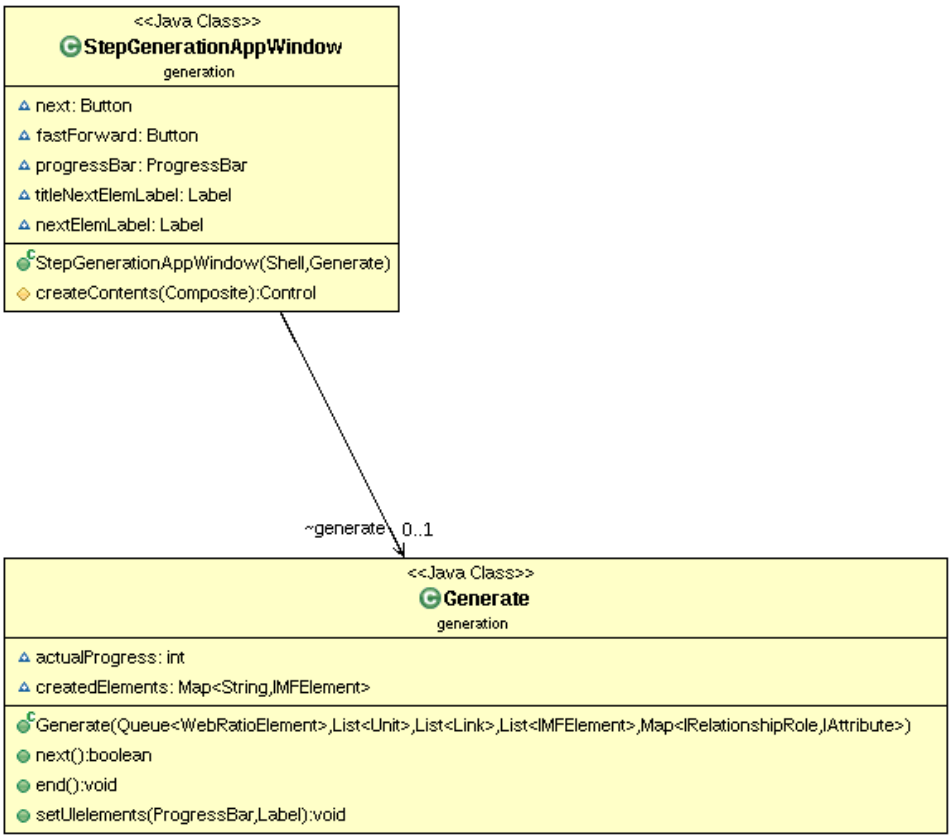

Ilustración 25: Diagrama de clases. Paquete generation

- **org.homeria.webratioassistant.parser**

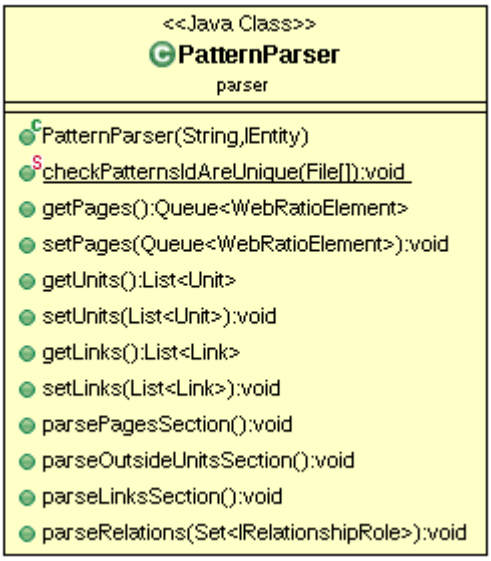

Ilustración 26: Diagrama de clases. Paquete parser

- **org.homeria.webratioassistant.registry**

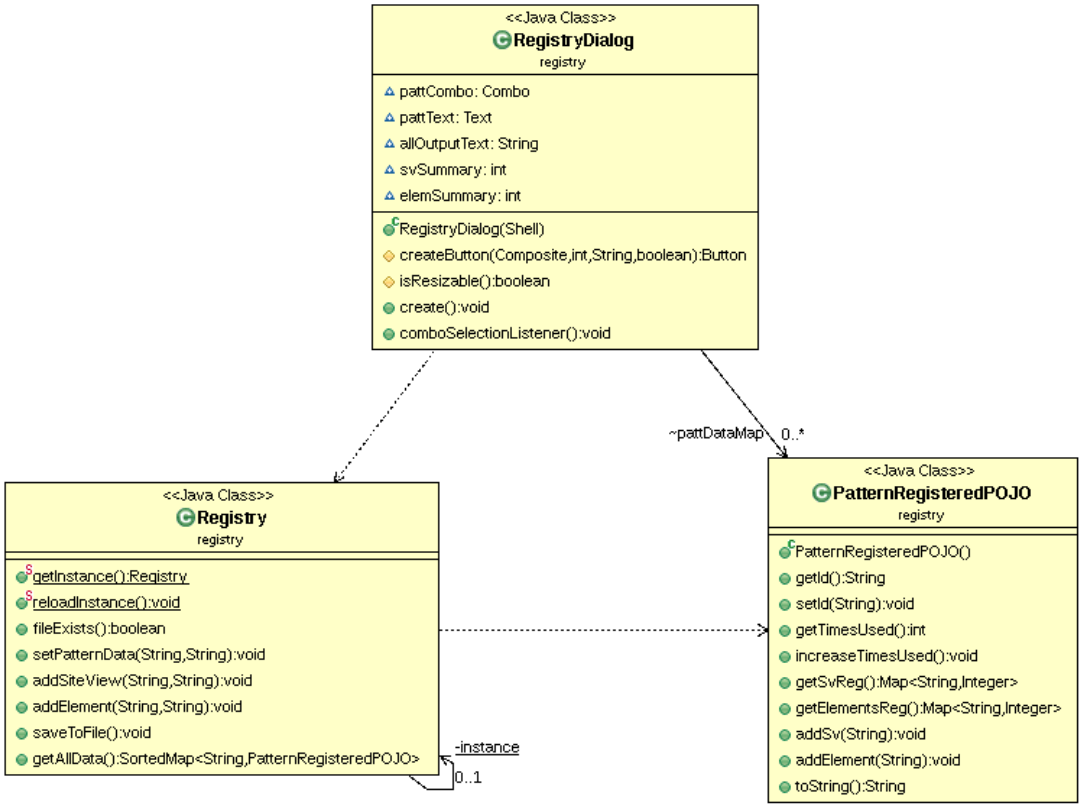

Ilustración 27: Diagrama de clases. Paquete registry

- **org.homeria.webratioassistant.webratio**

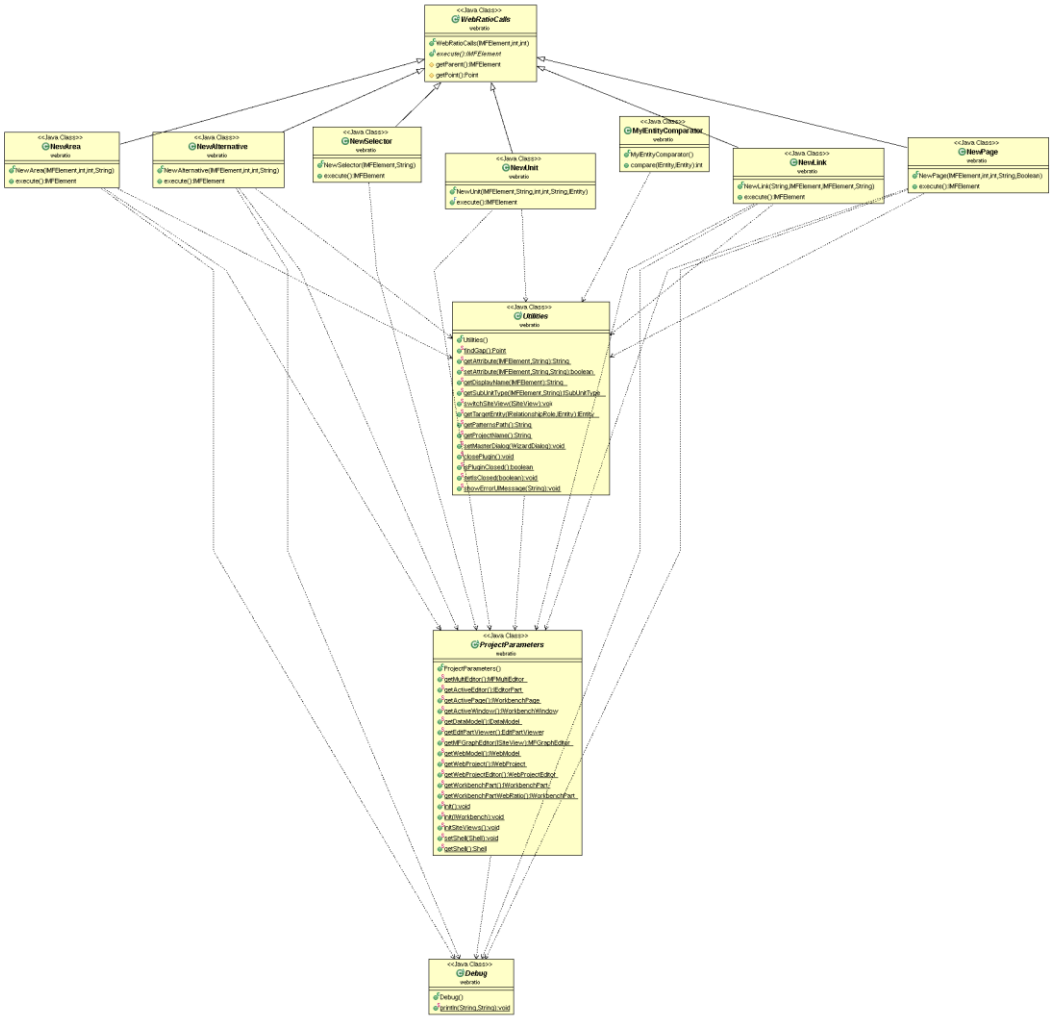

Ilustración 28: Diagrama de clases. Paquete webratio

- **org.homeria.webratioassistant.webratioaux**

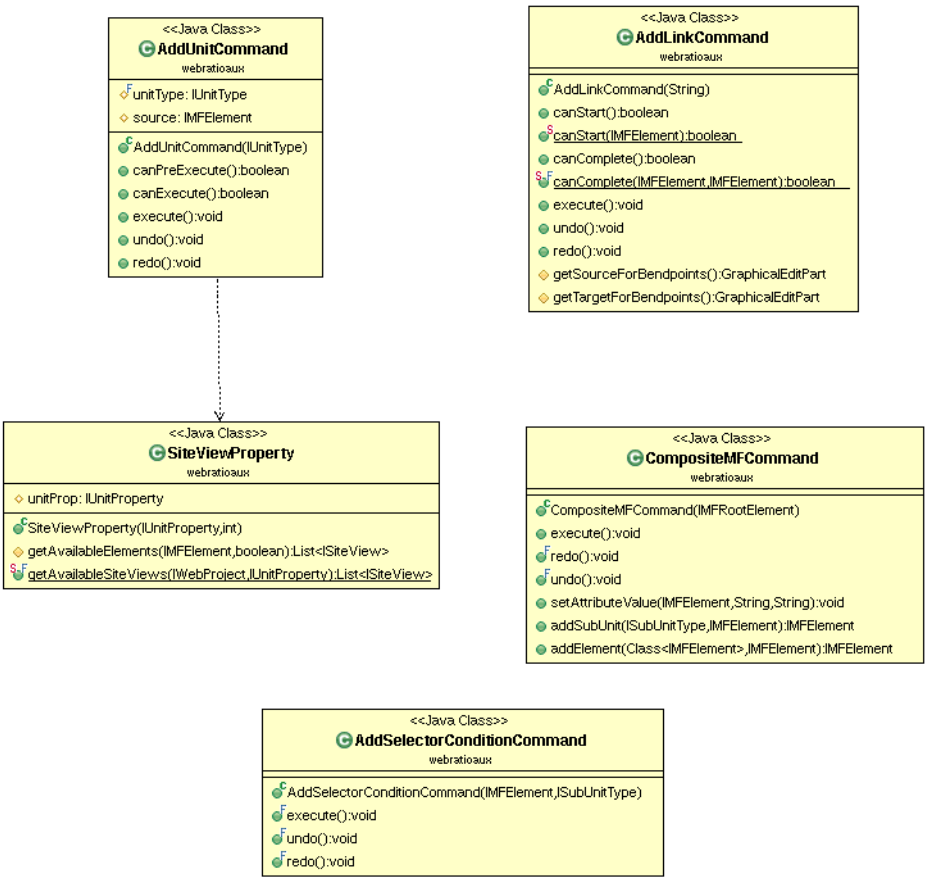

Ilustración 29: Diagrama de clases. Paquete webratioaux

- **org.homeria.webratioassistant.wizards**

Ilustración 30: Diagrama de clases. Paquete wizards

7 Resultados

Se ha conseguido cumplir la mayor parte de los objetivos propuestos al inicio del proyecto. Se ha tenido en cuenta la importancia de la obtención de cada objetivo para el resultado final y se ha seguido la prioridad establecida.

Seguidamente se puede ver el grado de consecución de cada objetivo y sus objetivos de forma individual, con un breve comentario debajo de aquellos objetivos que no se han conseguido:

Leyenda de objetivos:

	Verde	Completado
	Amarillo	Completado parcialmente
	Rojo	No completado / Descartado

Tabla 29 - Leyenda de objetivos

Objetivo 1: Descripción externa de patrones. Prioridad: Alta

❖ Listado de subobjetivos:

- Definir una estructura del patrón genérica que permita representar los patrones de la versión del plug-in anterior (CRUD y AllInOne) y los que el usuario defina.
- Definir la sintaxis y la semántica de los elementos y sus atributos.
- Dotar a la aplicación de la capacidad de interpretar los patrones, almacenar la información y generar sus unidades con las reglas establecidas.
- Validar los patrones en base a la estructura diseñada.

Tabla 30 – Grado de consecución del Objetivo 1: Descripción externa de patrones

Objetivo 2: Simplificar la interacción. Prioridad: Alta

❖ Listado de subobjetivos:

- Eliminar la funcionalidad que crea un SiteView/Area nuevo.
- Aunar las páginas restantes del asistente.
- Visualizar completamente el contenido de los componentes.

Tabla 31 - Grado de consecución del Objetivo 2: Simplificar la interacción

Objetivo 3: Registro. Prioridad: Alta

❖ Listado de subobjetivos:

- Diseñar la estructura de datos que almacenará el registro.
- Registrar los elementos generados en base a la estructura diseñada.
- Operar a nivel de patrón sobre el modelo.
- Identificar conflictos por cambios manuales en el modelo.
- Mostrar el registro.
- Generar datos estadísticos.
- Exportar los datos estadísticos (en CSV o similar).

Tabla 32 - Grado de consecución del Objetivo 3: Registro

El motivo de no haber podido cumplir los subobjetivos anteriores (excepto el último), tiene que ver con la dificultad de utilizar las librerías internas de WebRatio. Se explica en el punto *8.1 Dificultades encontradas.*

La causa de no cumplir el subobjetivo de exportación de datos estadísticos tiene que ver básicamente con el tiempo invertido en las demás funcionalidades, impidiendo su consecución en los plazos establecidos.

Objetivo 4: Generación paso a paso. Prioridad: Media

❖ Listado de subobjetivos:

- Modificar la secuencia de generación.
- Mostrar información del progreso.

Tabla 33 - Grado de consecución del Objetivo 4: Generación paso a paso

Objetivo 5: Refactorización del código fuente. Prioridad: Media

❖ Listado de subobjetivos:

- Mejorar la eficiencia general del código de la versión anterior.
- Traducir variables y documentación de la versión anterior.
- Realizar la documentación interna faltante de la versión anterior.

Tabla 34 - Grado de consecución del Objetivo 5: Refactorización del código fuente

Objetivo 6: Identificación de patrones existentes. Prioridad: Baja

❖ Listado de subobjetivos:

- Crear un sistema de búsqueda.
- Recorrer los elementos del modelo aplicando el sistema de búsqueda.
- Identificar el patrón en el modelo web.
- Registrarlo.

Tabla 35 - Grado de consecución del Objetivo 6: Identificación de patrones existentes

El motivo del descarte completo de este objetivo tiene que ver con la ambición puesta en la fase de análisis para conseguir una aplicación final excelente.

Conociendo la probabilidad de que este objetivo se saliera de los plazos previstos del proyecto, se le asignó una prioridad baja y su logro dependería de la consecución de los objetivos anteriores con más prioridad, junto con la

existencia de un período de tiempo razonable para su realización. Situación a la que no se ha podido llegar.

8 Conclusiones

Se ha continuado la línea del *Proyecto de Fin de Carrera anterior*, incidiendo en la sistematización y productividad del desarrollo de modelos Web para WebRatio, cumpliendo los nuevos requisitos del usuario final para tal efecto.

Se ha conseguido introducir una funcionalidad muy importante en la aplicación que permite al usuario personalizar el conjunto de elementos que desea generar mediante patrones de generación. Se han añadido nuevas funcionalidades, como el registro, además de modificar las que ya existían en el proyecto anterior, mejorando su calidad, eficiencia y mantenibilidad.

En definitiva, se ha logrado la gran mayoría de los objetivos que se propusieron en el inicio del proyecto, superándose todas las dificultades encontradas y obteniendo un resultado muy satisfactorio.

8.1 Dificultades encontradas

Como no podía ser de otra manera, el desarrollo del proyecto no ha estado exento de dificultades de todo tipo. A continuación, se describen dos de las dificultades más importantes que se han encontrado:

La primera tiene que ver con la fase inicial y el estudio del código de la aplicación del proyecto anterior. La dificultad radica en las dimensiones del proyecto y del número de clases. Clases que necesitan ser comprendidas perfectamente para su posterior modificación.

La segunda es sobre la dificultad de utilización de las librerías internas de WebRatio. WebRatio no tiene una API (Application Programming Interface) externa y no se puede consultar ninguna documentación al respecto. Resulta muy complicado encontrar las funcionalidades deseadas cuando tan sólo una librería (de las muchas que tiene el entorno) contiene más de 1580 clases. Esto ha marcado parte del desarrollo del proyecto al no haberse podido cumplir algunos de los subobjetivos marcados.

8.2 Líneas de desarrollo futuro

Se puede continuar la línea de trabajo seguida hasta ahora y añadir nueva funcionalidad a la aplicación o mejorar la actual.

Una de las tareas que se pueden llevar a cabo es desarrollar el *objetivo 6* de este proyecto que no se ha llegado cumplir.

Por otra parte, se pueden investigar las librerías de WebRatio con el fin de:

- Obtener los comandos de generación de unidades adicionales que no se contemplan en este proyecto.
- Obtener los comandos de manipulación del modelo web, permitiendo modificar elementos del modelo, siguiendo la línea de los subobjetivos que no se han podido cumplir del *objetivo 3.*

REFERENCIAS BIBLIOGRÁFICAS

AGUADO FUENTES, Carlos. “Generación automática de la arquitectura de una aplicación web en WebML a partir de la especificación de requisitos”, Proyecto Fin de Carrera, Escuela Politécnica - Universidad de Extremadura, Cáceres, julio 2013.

BRAMBILLA, Marco. “Interaction Flow Modeling Language (IFML) adopted as a standard by OMG” (23 mar. 2013). [En línea] <https://marco-brambilla.com/2013/03/23/interaction-flow-modeling-language-ifml-adopted-as-a-standard-by-omg/> [Consulta: 9 ago. 2017]

BRAMBILLA, Marco; BUTTI, Stefano. “Fifteen Years of Industrial Model-Driven Development in Software Front-End: from WebML to WebRatio and IFML”. *Novática,* abril - junio 2014, nº228, 36. ISSN: 0211-2124. Disponible en: <http://dbgroup.como.polimi.it/brambilla/sites/dbgroup.como.polimi.it.brambilla/files/referencematerials/WebML-IFML-WebRatio-Novatica-EN.pdf> [Consulta: 6 ago. 2017]

DOCUMENT OBJECT MODEL (DOM) [En línea] <https://www.w3.org/DOM/> [Consulta: 1 sept. 2017]

GIT, sistema de control de versiones. [En línea] <https://git-scm.com/> [Consulta: 1 sept. 2017]

HILERA GONZÁLEZ, José Ramón [Jose R. H.]. “Metodologías de Ingeniería Web dirigida por modelos” (11 nov. 2013). [Youtube] <https://www.youtube.com/watch?v=BVxr4EfozMU> [Consulta: 5 ago. 2017]

HOMERIA OPEN SOLUTIONS [En línea] <http://www.homeria.com/> [Consulta: 1 sept. 2017]

IEEE MultiMedia Volume 8, Number 1, January / March, 2001 [En línea] <ftp://ftp.math.utah.edu/pub/tex/bib/toc/ieeemultimedia.html#8(1):January/March:2001> [Consulta: 1 sept. 2017]

IFML: The Interaction Flow Modeling Language | The OMG standard for front-end design. Object Management Group. [En línea] <http://www.ifml.org/> [Consulta: 7 ago. 2017]

OBJECT MANAGEMENT GROUP (OMG) [En línea] <http://www.omg.org/> [Consulta: 1 sept. 2017]

RODRÍGUEZ-ECHEVERRÍA, R., CONEJERO, J., PRECIADO, J. y SÁNCHEZ-FIGUEROA, F. "AutoCRUD – Automating IFML Specification of CRUD Operations". Proceedings of the 12th International Conference on Web Information Systems and Technologies (WEBIST 2016) - Volume 1, pages 307-314 ISBN: 978-989-758-186-1

TOMASSETTI, Gabriele. "Parsing in Java: Tools and Libraries" (31 mayo 2017). [En línea] <https://tomassetti.me/parsing-in-java/> [Consulta 10 jul. 2017]

WEBRATIO [En línea] < http://www.webratio.com/> [Consulta: 1 sept. 2017]

WEBRATIO [WebRatio] "IFML, el nuevo estándar para el modelado | WebRatio Webinar" (4 jul. 2013). [Youtube] <https://www.youtube.com/watch?v=Wb87UfqZuFA> [Consulta: 8 ago. 2017]

WORLD WIDE WEB CONSORTIUM (W3C) [En línea] <https://www.w3.org/> [Consulta: 1 sept. 2017]

XML Schema Part 0: Primer Second Edition. W3C Recommendation. (28 oct. 2004) [En línea] <https://www.w3.org/TR/xmlschema-0/> [Consulta: 10 jul. 2017]

ANEXOS

1 ANEXO I: Manual de usuario

En esta sección se explica la instalación del plug-in desarrollado junto con la localización de los archivos para un correcto funcionamiento de la aplicación. Seguidamente, se desarrollan los patrones de ejemplo que se adjuntan a esta memoria en el CD. Finalizando con un ejemplo de ejecución del asistente que le guiará a través de la aplicación.

Si necesita información sobre los patrones, su estructura, elementos disponibles, atributos, sintaxis, etc. le recomiendo leer el apartado *5.1 Patrones* de esta memoria.

1.1 Instalación del plug-in

El requisito fundamental para instalar el plug-in es el de tener instalado el entorno de desarrollo WebRatio que se puede obtener en http://www.webratio.com. Además, para poder ejecutar el programa, hay que instalar la versión de Java (www.java.com) correspondiente a la versión de WebRatio obtenida.

Una vez instalados los requisitos, se localiza el archivo "**org.homeria.webratioassistant_3.0.0.jar**", que se adjunta en el CD entregado con esta memoria, y se copia en el directorio *"plug-ins"* situado dentro del directorio de WebRatio.

Hay que reiniciar WebRatio, si estuviese abierto, porque los plug-ins se cargan en memoria durante el arranque del entorno.

1.2 Localización de los archivos

Los archivos xml que representan los patrones definidos por el usuario deben introducirse en un nuevo directorio llamado "*patterns*". Este directorio debe situarse en la raíz del Proyecto Web donde se desee utilizar.

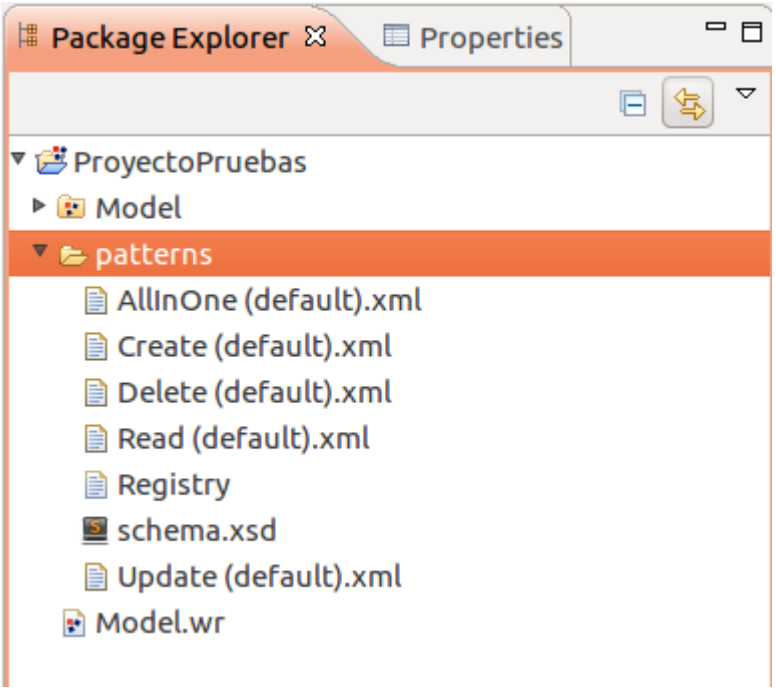

Ilustración 31: Localización del directorio patterns

Como se puede comprobar en la ilustración anterior, en este directorio también se debe situar el archivo *schema.xsd* que contiene la definición de la estructura del patrón y es utilizada por la aplicación.

1.3 Patrones de ejemplo

En el CD adjunto a esta memoria se proporcionan 5 patrones de ejemplo para que el usuario los pueda utilizar y le sirvan de guía para crear sus propios patrones personalizados.

En los subapartados sucesivos se detalla cada uno de ellos proporcionando un ejemplo de ejecución junto a su código.

1.3.1 Create (default).xml

El patrón Create describe el conjunto de elementos IFML que constituye la cadena de operaciones. Esta secuencia permite al usuario final de la aplicación web crear nuevas tuplas en una entidad de la base de datos.

1.3.1.1 Estructura

Contenedores:

Se define una página contenedora que alberga un formulario *(EntryUnit)* y una unidad de mensajes *(MultiMessageUnit)*.

Unidades fuera de contenedores:

Fuera de la página, se encuentra la unidad de creación (*CreateUnit)*.

Flujos:

El formulario es el origen del flujo de carga de datos hacia la unidad de creación. En caso de error en esta unidad, se muestra un mensaje en la unidad de mensajes, tal y como indica el flujo KO definido.

Relaciones*:*

Dependiendo de las relaciones seleccionadas en la interfaz de usuario se generan unos elementos u otros:

Todas las relaciones:

Por cada relación que se seleccione, sin importar su cardinalidad, se define una *SelectorUnit* de la relación dentro de la página definida anteriormente, además de un flujo de transporte que une esta unidad con el formulario.

Relaciones N:M:

Además de lo anterior, por cada relación seleccionada con cardinalidad N:M, se define una *ConnectUnit*, un flujo de datos que conecta el formulario con esta unidad y un flujo KO que conecta la unidad con la unidad de mensajes.

La primera relación N:M que se procese crea un flujo OK desde la *CreateUnit* a la *ConnectUnit*. Las sucesivas relaciones N:M crean un flujo OK entre la anterior *ConnectUnit* y la nueva.

Cierre de la cadena de operaciones:

Si se ha seleccionado alguna relación N:M, se crea un flujo OK entre la última *ConnectUnit* y la unidad de mensajes, tal y como se indica en el apartado *LAST* de *NMRELATIONS.*

Si no se ha seleccionado ninguna relación N:M, se crea un flujo OK entre la *CreateUnit* y la unidad de mensajes, tal y como se indica en el apartado *LAST* de *RELATIONS*.

1.3.1.2 Diagrama IFML

Con la entidad *Group* y ninguna relación:

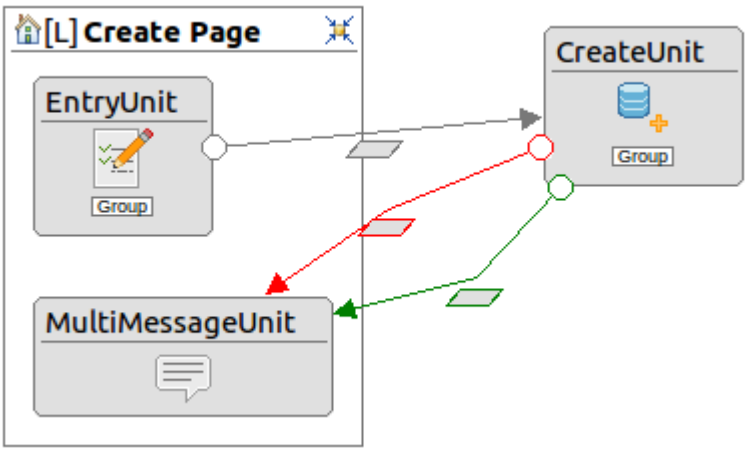

Ilustración 32: Diagrama IFML del patrón Create sin relaciones

Con la entidad *Group* y dos relaciones:

1. La relación N:M entre *User* y *Group.*
2. La relación 1:N entre *Module* y *Group.*

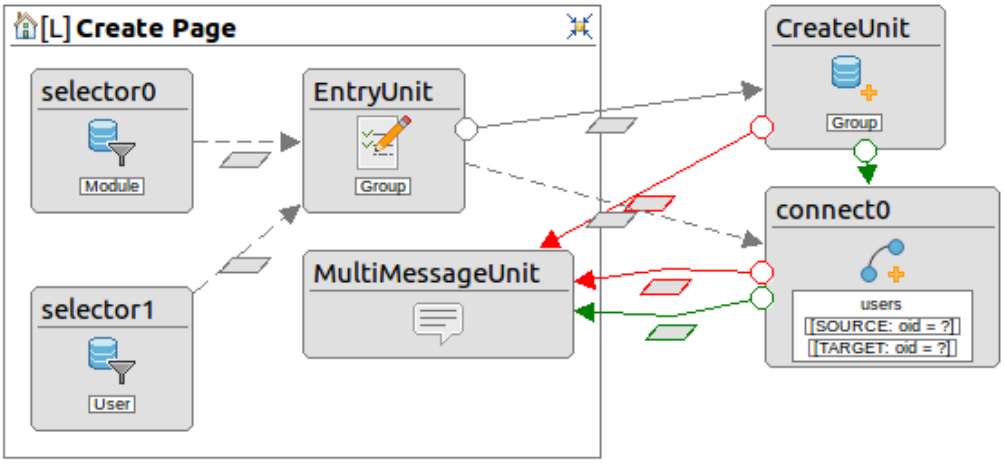

Ilustración 33: Diagrama IFML del patrón Create con relaciones

1.3.1.3 Código

```
<?xml version="1.0"?>

<PATTERN id="p1" name="Create (default)"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:noNamespaceSchemaLocation="schema.xsd">

    <!-- PAGES SECTION-->
    <PAGES>
        <page id="CreatePage" name="Create Page" landmark="true" x="30" y="30">
            <entryUnit id="entryUnit" name="EntryUnit" x="150" y="0"/>
            <multiMessageUnit id="multiMessageUnit" name="MultiMessageUnit" x="150" y="100"/>
        </page>
    </PAGES>

    <!-- UNITS OUTSIDE OF ANY PAGE -->
    <OUTSIDEUNITS>
        <createUnit id="createUnit" name="CreateUnit" x="450" y="30"/>
    </OUTSIDEUNITS>

    <!-- LINKS SECTION -->
    <LINKS>
        <normalNavigationFlow id="nnfLoad" name="Load" sourceId="entryUnit" targetId="createUnit"
type="entryToCreate"/>
        <koLink id="koCreate" sourceId="createUnit" targetId="multiMessageUnit" message="Failed creating data"/>
    </LINKS>

    <!-- RELATIONS SECTION -->
    <RELATIONS> <!-- Common section for 1:1, 1:N and N:M relations -->
        <ALL> <!-- All relations -->
            <selectorUnit id="selector#" name="selector#" parentId="CreatePage" x="0" y="0#"/>
            <dataFlow id="flowSelectorForm#" name="Load" sourceId="selector#" targetId="entryUnit" type="preload"/>
        </ALL>

        <LAST> <!-- Last relation (used to close operations chain) -->
```

```
            <okLink id="okConnect" sourceId="createUnit" targetId="multiMessageUnit" message="Data created
correctly"/>
        </LAST>
    </RELATIONS>

    <NMRELATIONS> <!-- Specific section only for N:M relations (Replaces RELATIONS)-->
        <ALL> <!-- All N:M relations -->
            <selectorUnit id="selector #" name="selector#" parentId="CreatePage" x="0" y="0#"/>
            <dataFlow id="flowSelectorForm #" name="Load" sourceId="selector #" targetId="entryUnit"
type="preload"/>

            <connectUnit id="connect%" name="connect%" x="450" y="130%"/>
            <dataFlow id="flowFormConnect%" name="Load" sourceId="entryUnit" targetId="connect%"
type="entryToConnect"/>

            <koLink id="koConnect%" sourceId="connect%" targetId="multiMessageUnit" message="Failed creating data"/>
        </ALL>

        <FIRST> <!-- First N:M relation (used to start operations chain) -->
            <okLink id="okFirstConnect%" sourceId="createUnit" targetId="connect%" message=""/>
        </FIRST>

        <REMAINING> <!-- Second and subsequent N:M relations (used to continue operations chain) -->
            <okLink id="okConnect%" sourceId="connect%-1" targetId="connect%" message=""/>
        </REMAINING>

        <LAST> <!-- Last relation (used to close operations chain) -->
            <okLink id="okConnect%+1" sourceId="connect%" targetId="multiMessageUnit" message="Data created
correctly"/>
        </LAST>
    </NMRELATIONS>

</PATTERN>

    <!-- # = iterative number for each relation -->
    <!-- % = iterative number for each NM relation -->
```

1.3.2 Read (default).xml

El patrón Read describe el conjunto de elementos IFML que constituye la cadena de operaciones. Esta secuencia permite al usuario final de la aplicación web visualizar las tuplas de una entidad de la base de datos.

1.3.2.1 Estructura

Contenedores:

Se define una página contenedora que alberga una lista *(PowerIndexUnit)* y una unidad de detalle *(DataUnit)*.

Flujos:

La lista es el origen de un flujo de navegación hacia la unidad de detalle.

1.3.2.2 Diagrama IFML

Con la entidad *Group*:

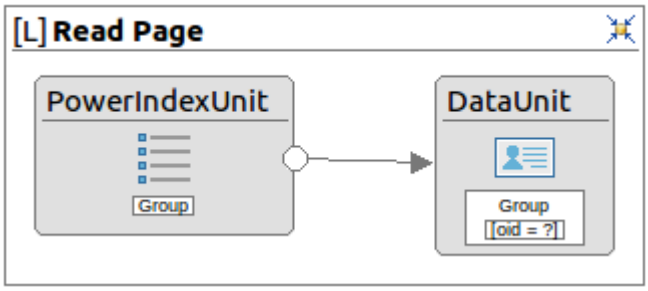

Ilustración 34: Diagrama IFML del patrón Read

1.3.2.3 Código

```
<?xml version="1.0"?>

<PATTERN id="p2" name="Read (default)"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:noNamespaceSchemaLocation="schema.xsd">

    <!-- PAGES SECTION-->
    <PAGES>
        <page id="ReadPage" name="Read Page" landmark="true" x="30" y="30">
            <powerIndexUnit id="powerIndexUnit" name="PowerIndexUnit" x="0" y="0"/>
            <dataUnit id="dataUnit" name="DataUnit" x="200" y="0"/>
        </page>
    </PAGES>


    <!-- LINKS SECTION -->
    <LINKS>
        <normalNavigationFlow id= "nnf1" name="Details" sourceId="powerIndexUnit" targetId="dataUnit"/>
    </LINKS>

</PATTERN>
```

1.3.3 Update (default).xml

El patrón Update describe el conjunto de elementos IFML que constituye la cadena de operaciones. Esta secuencia permite al usuario final de la aplicación web modificar las tuplas de una entidad de la base de datos.

1.3.3.1 Estructura

Contenedores:

Se define una página contenedora que alberga un formulario *(EntryUnit)*, una lista *(PowerIndexUnit),* una unidad de mensajes *(MultiMessageUnit)* y una *SelectorUnit*.

Unidades fuera de contenedores:

Fuera de la página, se encuentra la unidad de actualización (*UpdateUnit)*.

Flujos:

Existe un flujo de navegación normal entre la lista y el selector de la entidad.

El formulario es el origen del flujo de carga de datos hacia la unidad de actualización. En caso de error en esta unidad de actualización, se muestra un mensaje en la unidad de mensajes, tal y como indica el flujo KO definido.

Además, se define un flujo de transporte entre el selector de la entidad y el formulario.

Relaciones*:*

Dependiendo de las relaciones seleccionadas en la interfaz de usuario se generan unos elementos u otros:

Todas las relaciones:

Por cada relación que se seleccione, sin importar su cardinalidad, se define una *SelectorUnit* de la relación dentro de la página definida anteriormente, junto a un flujo de transporte que relaciona esta unidad con el formulario.

Relaciones N:M:

Además de lo anterior, por cada relación seleccionada con cardinalidad N:M, se define:

Una *SelectorUnit* de la relación dentro de la página definida anteriormente.

Un flujo de datos entre la *SelectorUnit* de la entidad y la *SelectorUnit* de la relación.

Otro flujo de datos entre la *SelectorUnit* de la relación y el formulario.

Una *ReconnectUnit.*

Un flujo de datos que conecta el formulario con la *ReconnectUnit*

Un flujo KO que conecta la *ReconnectUnit* con la unidad de mensajes.

La primera relación N:M que se procese crea un flujo OK desde la *UpdateUnit* a la *ReconnectUnit*. Las sucesivas relaciones N:M crean un flujo OK entre la anterior *ReconnectUnit* y la nueva.

Cierre de la cadena de operaciones:

Si se ha seleccionado alguna relación N:M, se crea un flujo OK entre la última *ReconnectUnit* y la unidad de mensajes, tal y como se indica en el apartado *LAST* de *NMRELATIONS.*

Si no se ha seleccionado ninguna relación N:M, se crea un flujo OK entre la *UpdateUnit* y la unidad de mensajes, tal y como se indica en el apartado *LAST* de *RELATIONS*.

1.3.3.2 Diagrama IFML

Con la entidad *Group* y ninguna relación:

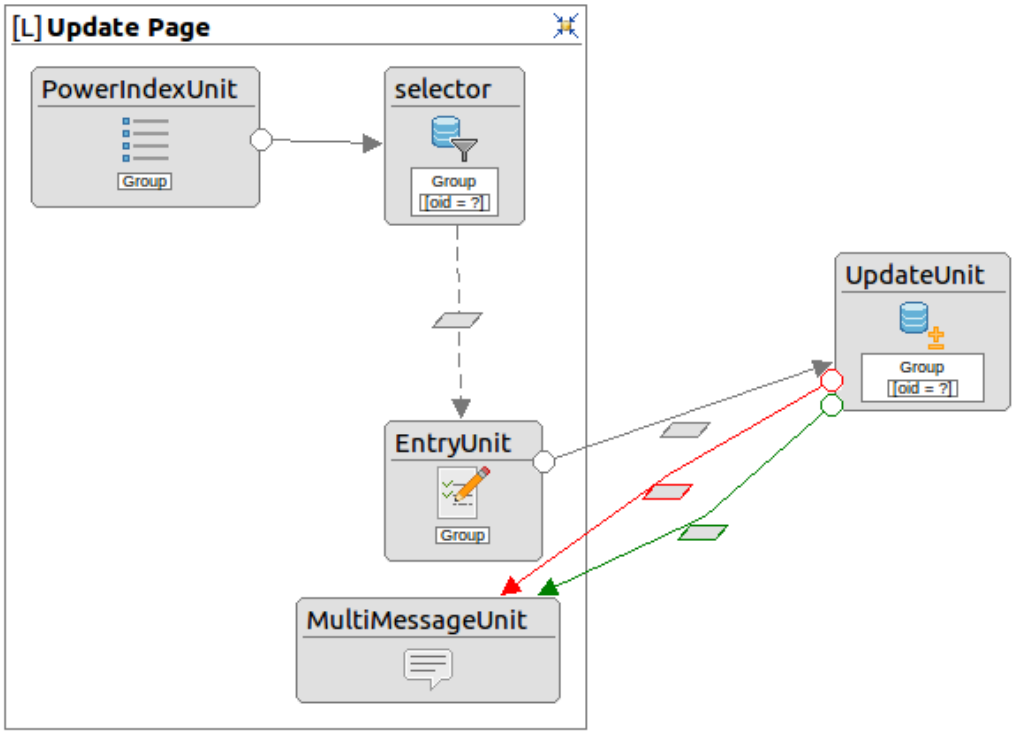

Ilustración 35: Diagrama IFML del patrón Update sin relaciones

Con la entidad *Group* y dos relaciones:

1. La relación N:M entre *User* y *Group.*
2. La relación 1:N entre *Module* y *Group.*

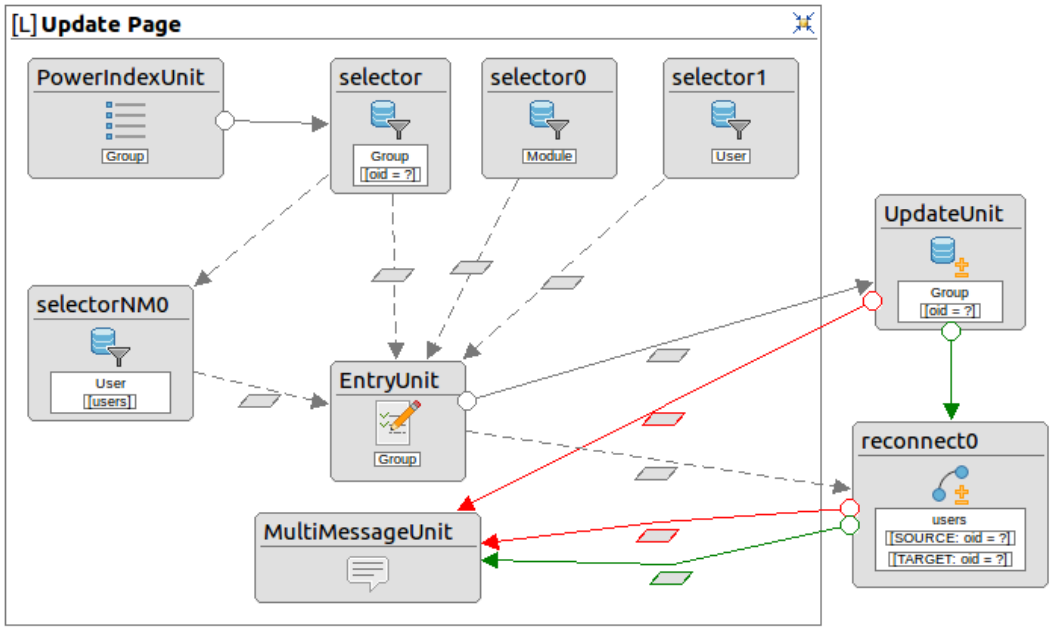

Ilustración 36: Diagrama IFML del patrón Update con relaciones

1.3.3.3 Código

```
<?xml version="1.0"?>

<PATTERN id="p3" name="Update (default)"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:noNamespaceSchemaLocation="schema.xsd">

    <!-- PAGES SECTION-->
    <PAGES>
        <page id="UpdatePage" name="Update Page" landmark="true" x="30" y="30">
            <entryUnit id="entryUnit" name="EntryUnit" type="preloaded" x="200" y="200"/>
            <powerIndexUnit id="powerIndexUnit" name="PowerIndexUnit" x="0" y="0"/>
            <multiMessageUnit id="multiMessageUnit" name="MultiMessageUnit" x="150" y="300"/>
            <selectorUnit id="selectorEntidad" name="selector" type="keyCondition" x="200" y="0"/>
        </page>
    </PAGES>


    <!-- UNITS OUTSIDE OF ANY PAGE -->
    <OUTSIDEUNITS>
        <updateUnit id="updateUnit" name="UpdateUnit"  x="500" y="170"/>
    </OUTSIDEUNITS>


    <!-- LINKS SECTION -->
    <LINKS>
        <normalNavigationFlow id="nnfUpdate" name="Modify" sourceId="powerIndexUnit" targetId="selectorEntidad" />
        <normalNavigationFlow id="nnfEntryUpdate" name="Load" sourceId="entryUnit" targetId="updateUnit"
type="entryToUpdate"/>
        <dataFlow id="dfSelectorEntry" name="Load" sourceId="selectorEntidad" targetId="entryUnit"
type="unitToEntry"/>
        <koLink id="koUpdate" sourceId="updateUnit" targetId="multiMessageUnit" message="Failed modifying data"/>
    </LINKS>
```

```
    <!-- RELATIONS SECTION -->
    <RELATIONS> <!-- Common section for 1:1, 1:N and N:M relations -->
        <ALL> <!-- All relations -->
            <selectorUnit id="selector#" name="selector#" parentId="UpdatePage" x="300#" y="0"/>
            <dataFlow id="flowSelectorForm#" name="Load" sourceId="selector#" targetId="entryUnit" type="preload"/>
        </ALL>

        <LAST> <!-- Last relation (used to close operations chain) -->
            <okLink id="okReconnect" sourceId="updateUnit" targetId="multiMessageUnit" message="Data modified
correctly"/>
        </LAST>
    </RELATIONS>

    <NMRELATIONS> <!-- Specific section only for N:M relations (Replaces RELATIONS)-->
        <ALL> <!-- All N:M relations -->
            <selectorUnit id="selector #" name="selector#" parentId="UpdatePage" x="300#" y="0"/>
            <dataFlow id="flowSelectorForm #" name="Load" sourceId="selector #" targetId="entryUnit"
type="preload"/>

            <selectorUnit id="selectorNM%" name="selectorNM%" parentId="UpdatePage" type="roleCondition" x="0"
y="150%"/>
            <dataFlow id="flowSelectSelect%" name="Load" sourceId="selectorEntidad" targetId="selectorNM%"/>
            <dataFlow id="flowSelectEntry%" name="Load" sourceId="selectorNM%" targetId="entryUnit"
type="unitToEntryRole"/>

            <reconnectUnit id="reconnect%" name="reconnect%" x="500" y="300%"/>
            <dataFlow id="flowFormReconnect%" name="Load" sourceId="entryUnit" targetId="reconnect%"
type="entryToReconnect"/>
            <koLink id="koReconnect%" sourceId="reconnect%" targetId="multiMessageUnit" message="Failed modifying
data"/>
        </ALL>

        <FIRST> <!-- First N:M relation (used to start operations chain) -->
            <okLink id="okFirstReconnect%" sourceId="updateUnit" targetId="reconnect%" message=""/>
```

```
        </FIRST>

        <REMAINING> <!-- Second and subsequent N:M relations (used to continue operations chain) -->
            <okLink id="okReconnect%" sourceId="reconnect%-1" targetId="reconnect%" message=""/>
        </REMAINING>

        <LAST> <!-- Last relation (used to close operations chain) -->
            <okLink id="okReconnect%+1" sourceId="reconnect%" targetId="multiMessageUnit" message="Data modified
correctly"/>
        </LAST>
    </NMRELATIONS>

</PATTERN>


    <!-- # = iterative number for each relation -->
    <!-- % = iterative number for each NM relation -->
```

1.3.4 Delete (default).xml

El patrón Delete describe el conjunto de elementos IFML que constituye la cadena de operaciones. Esta secuencia permite al usuario final de la aplicación web eliminar las tuplas de una entidad de la base de datos.

1.3.4.1 Estructura

Contenedores:

Se define una página contenedora que alberga una lista *(PowerIndexUnit) y* una unidad de mensajes *(MultiMessageUnit).*

Unidades fuera de contenedores:

Fuera de la página, se encuentra la unidad de eliminación (*DeleteUnit)*.

Flujos:

Existe un flujo de navegación normal entre la lista y la unidad de eliminación.

Además, se define un flujo OK y otro KO que comparten el origen (*DeleteUnit)* y el destino *(MultiMessageUnit)*

1.3.4.2 Diagrama IFML

Con la entidad *Group* y ninguna relación:

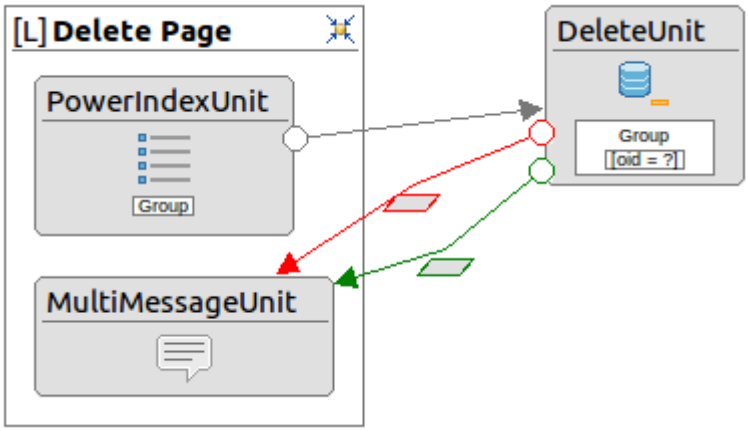

Ilustración 37: Diagrama IFML del patrón Delete

1.3.4.3 Código

```
<?xml version="1.0"?>

<PATTERN id="p4" name="Delete (default)"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:noNamespaceSchemaLocation="schema.xsd">

    <!-- PAGES SECTION-->
    <PAGES>
        <page id="DeletePage" name="Delete Page" landmark="true" x="30" y="30">
            <powerIndexUnit id="powerIndexUnit" name="PowerIndexUnit" x="0" y="0"/>
            <multiMessageUnit id="multiMessageUnit" name="MultiMessageUnit" x="0" y="100"/>
        </page>
    </PAGES>


    <!-- UNITS OUTSIDE OF ANY PAGE -->
    <OUTSIDEUNITS>
        <deleteUnit id="deleteUnit" name="DeleteUnit" x="300" y="30"/>
    </OUTSIDEUNITS>


    <!-- LINKS SECTION -->
    <LINKS>
        <normalNavigationFlow id="nnfDelete" name="Delete" sourceId="powerIndexUnit" targetId="deleteUnit"/>
        <koLink id="koDelete" sourceId="deleteUnit" targetId="multiMessageUnit" message="Failed deleting data"/>
        <okLink id="okDelete" sourceId="deleteUnit" targetId="multiMessageUnit" message="Data deleted correctly"/>
    </LINKS>

</PATTERN>
```

1.3.5 AllInOne (default).xml

El patrón AllInOne es un patrón especial que agrupa todos los patrones anteriores (Create, Read, Update y Delete). Describe el conjunto de elementos IFML que constituye la cadena de operaciones. Esta secuencia permite al usuario final de la aplicación web dar de alta, visualizar, modificar y eliminar las tuplas de una entidad de la base de datos.

1.3.5.1 Estructura

Contenedores:

Se define una página principal que contiene una página alternativa *(XOR)* que, a su vez, contiene tres páginas contenedoras:

1) Una página marcada por defecto que contiene un enlace *(NoOpContentUnit)*, una lista *(PowerIndexUnit),* una unidad de mensajes *(MultiMessageUnit)* y una *SelectorUnit*.
2) Una segunda página que contiene la unidad de detalle *(DataUnit)*
3) Una tercera página que alberga el formulario *(EntryUnit)* y la *SelectorUnit* de la entidad.

Unidades fuera de contenedores:

Fuera de la página, se encuentra la unidad de creación *(CreateUnit)*, la unidad de actualización (*UpdateUnit),* la unidad de eliminación *(DeleteUnit)* además de una unidad *IsNotNullUnit.*

Flujos:

Se definen una serie de flujos de navegación normal entre:

- La unidad de detalle y la unidad de mensajes.
- El formulario y la unidad de mensajes.
- La unidad *NoOpContentUnit* y el selector de la entidad.
- La lista y la unidad de detalle.
- La lista y el selector de la entidad.
- La lista y la unidad de eliminación.
- El formulario y la unidad *IsNotNullUnit*.

Un par de flujos OK entre:

- La unidad de eliminación y la unidad de mensajes.
- La unidad *IsNotNullUnit* y la unidad de actualización.

Un conjunto de flujos KO entre:

- La unidad de eliminación y la unidad de mensajes.
- La unidad de creación y la unidad de mensajes.
- La unidad de actualización y la unidad de mensajes.
- La unidad *IsNotNullUnit* y la unidad de creación.

Una sucesión de flujos de datos entre:

- El selector de la entidad y el formulario.
- El formulario y la unidad de creación.
- El formulario y la unidad de actualización.

Relaciones*:*

Dependiendo de las relaciones seleccionadas en la interfaz de usuario se generan unos elementos u otros:

Todas las relaciones:

Por cada relación que se seleccione, sin importar su cardinalidad, se define una *SelectorUnit* de la relación dentro de la página del formulario definida anteriormente, junto a un flujo de transporte que relaciona esta unidad con el formulario.

Relaciones N:M:

Además de lo anterior, por cada relación seleccionada con cardinalidad N:M, se define:

Una *SelectorUnit* de la relación dentro de la página del formulario.

Un flujo de datos entre la *SelectorUnit* de la entidad y la *SelectorUnit* de la relación.

Otro flujo de datos entre la *SelectorUnit* de la relación y el formulario.

Una *ConnectUnit*.

Un flujo de datos que conecta el formulario con la *ConnectUnit*.

Un flujo KO que conecta la *ConnectUnit* con la unidad de mensajes.

Una *ReconnectUnit.*

Un flujo de datos que conecta el formulario con la *ReconnectUnit*

Un flujo KO que conecta la *ReconnectUnit* con la unidad de mensajes.

La primera relación N:M que se procese crea un flujo OK desde la *CreateUnit* a la *ConnectUnit*, junto con otro flujo OK desde la *UpdateUnit* a la *ReconnectUnit*.

Las sucesivas relaciones N:M crean un flujo OK entre la anterior *ConnectUnit* y la nueva, junto con otro flujo OK entre la anterior *ReconnectUnit* y la nueva.

Cierre de la cadena de operaciones:

Si se ha seleccionado alguna relación N:M, se crea un flujo OK entre la última *ConnectUnit* y la unidad de mensajes junto con otro flujo OK entre la última *ReconnectUnit* y la unidad de mensajes, tal y como se indica en el apartado *LAST* de *NMRELATIONS.*

Si no se ha seleccionado ninguna relación N:M, se crea un flujo OK entre la *CreateUnit* y la unidad de mensajes, junto con otro flujo OK entre la *UpdateUnit* y la unidad de mensajes, tal y como se indica en el apartado *LAST* de *RELATIONS*.

1.3.5.2 Diagrama IFML

Con la entidad *Group* y ninguna relación:

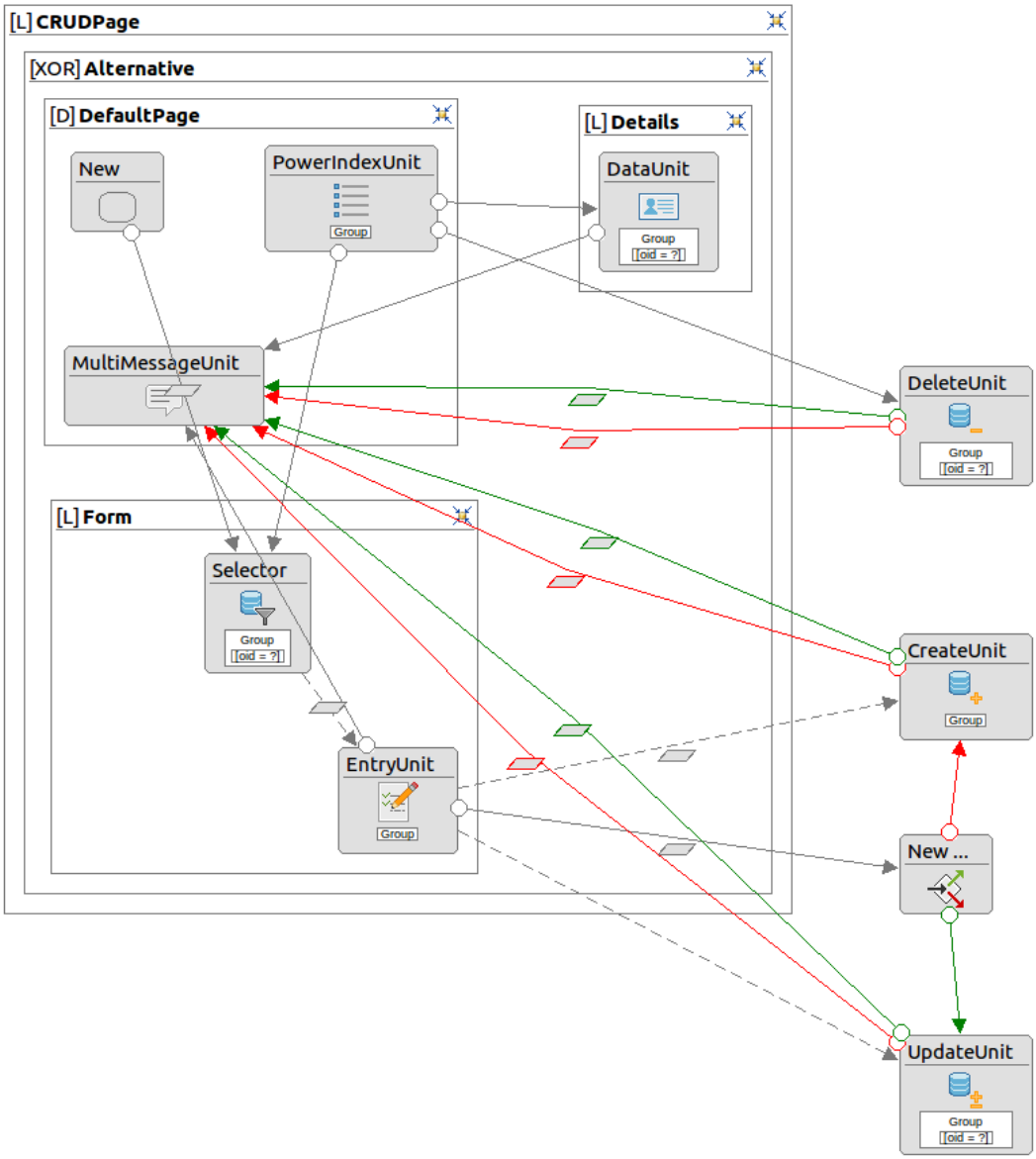

Ilustración 38: Diagrama IFML del patrón AllInOne sin relaciones

Con la entidad *Group* y dos relaciones:

3. La relación N:M entre *User* y *Group.*
4. La relación 1:N entre *Module* y *Group.*

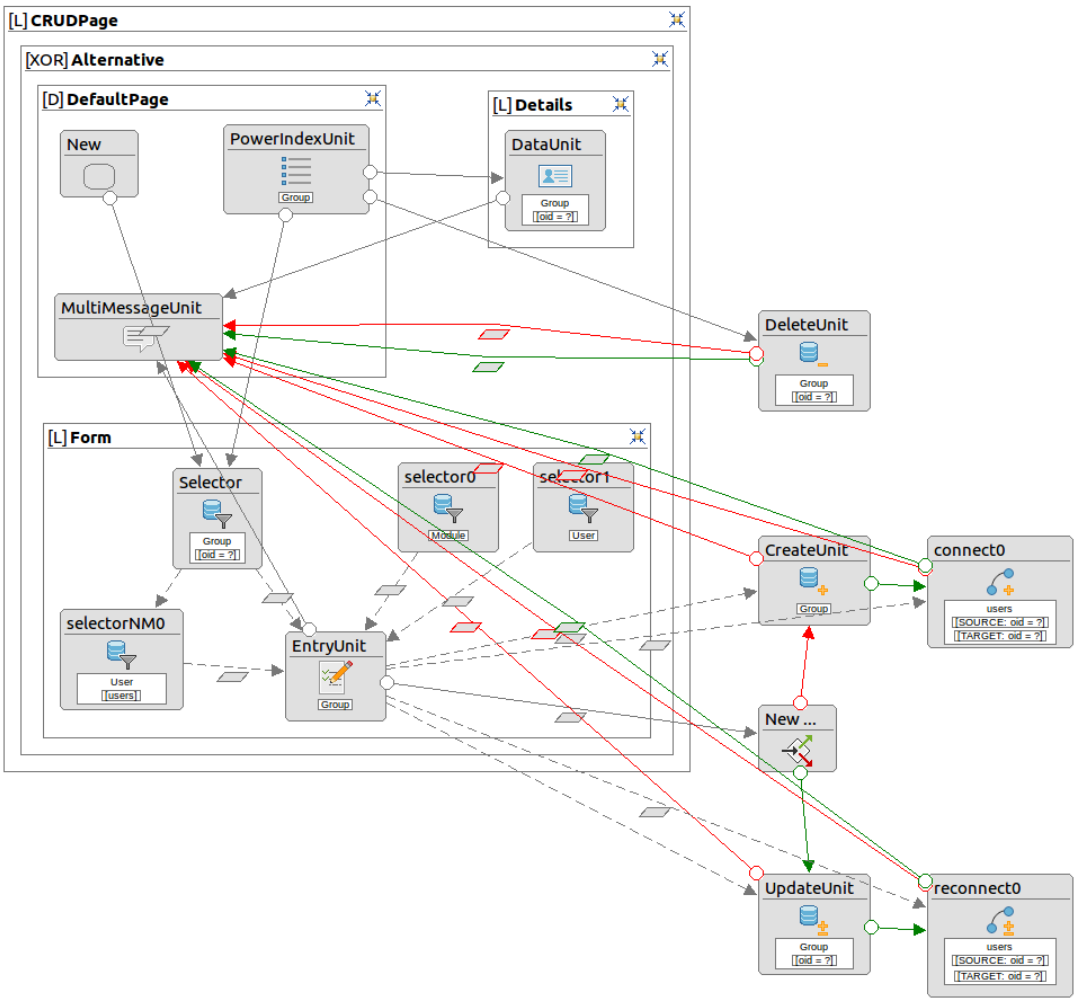

Ilustración 39: Diagrama IFML del patrón AllInOne con relaciones

1.3.5.3 Código

```
<?xml version="1.0"?>

<PATTERN id="p0" name="AllInOne (default)"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:noNamespaceSchemaLocation="schema.xsd">

    <!-- PAGES SECTION-->
    <PAGES>
        <page id="CRUDPage" name="CRUDPage" landmark="true" x="30" y="30">
            <xor id="XORPage" name="Alternative" x="0" y="0">
                <page id="DefaultPage" name="DefaultPage" default="true" x="30" y="30">
                    <noOpContentUnit id="noOpContentUnit" name="New" x="5" y="5"/>
                    <multiMessageUnit id="multiMessageUnit" name="MultiMessageUnit" x="0" y="150"/>
                    <powerIndexUnit id="powerIndexUnit" name="PowerIndexUnit" x="150" y="0"/>
                </page>

                <page id="DetailsPage" name="Details" landmark="true" x="400" y="5">
                    <dataUnit id="dataUnit" name="DataUnit" x="0" y="0"/>
                </page>

                <page id="FormPage" name="Form" landmark="true" x="5" y="300">
                    <entryUnit id="entryUnit" name="EntryUnit" x="200" y="150"/>
                    <selectorUnit id="selectorEntidad" name="Selector" type="keyCondition" x="100" y="5"/>
                </page>
            </xor>
        </page>
    </PAGES>


    <!-- UNITS OUTSIDE OF ANY PAGE -->
    <OUTSIDEUNITS>
        <createUnit id="createUnit" name="CreateUnit" x="700" y="500"/>
```

```
        <deleteUnit id="deleteUnit" name="DeleteUnit" x="700" y="300"/>
        <updateUnit id="updateUnit" name="UpdateUnit"  x="700" y="800"/>
        <isNotNullUnit id="isNotNullUnit" name="New ..." x="700" y="650"/>
    </OUTSIDEUNITS>


    <!-- LINKS SECTION -->
    <LINKS>
        <normalNavigationFlow id= "nnf1" name="Close" sourceId="dataUnit" targetId="multiMessageUnit"/>
        <normalNavigationFlow id= "nnf2" name="Cancel" sourceId="entryUnit" targetId="multiMessageUnit"
validate="false"/>
        <normalNavigationFlow id= "nnf3" name="New" sourceId="noOpContentUnit" targetId="selectorEntidad"
type="fixedValue"/>
        <normalNavigationFlow id= "nnf4" name="View" sourceId="powerIndexUnit" targetId="dataUnit"/>
        <normalNavigationFlow id= "nnf5" name="Modify" sourceId="powerIndexUnit" targetId="selectorEntidad"/>
        <normalNavigationFlow id= "nnf6" name="Delete" sourceId="powerIndexUnit" targetId="deleteUnit"/>
        <normalNavigationFlow id= "nnf7" name="Accept" sourceId="entryUnit" targetId="isNotNullUnit"
type="isNotNull"/>

        <okLink id="okDelete" sourceId="deleteUnit" targetId="multiMessageUnit" message="Data deleted correctly"/>
        <okLink id="okNN" sourceId="isNotNullUnit" targetId="updateUnit"/>

        <koLink id="koDelete" sourceId="deleteUnit" targetId="multiMessageUnit" message="Failed deleting data"/>
        <koLink id="koCreate" sourceId="createUnit" targetId="multiMessageUnit" message="Failed creating data"/>
        <koLink id="koUpdate" sourceId="updateUnit" targetId="multiMessageUnit" message="Failed modifying data"/>
        <koLink id="koNN" sourceId="isNotNullUnit" targetId="createUnit"/>


        <dataFlow id="df1" name="Load" sourceId="selectorEntidad" targetId="entryUnit" type="unitToEntry"/>
        <dataFlow id="df2" name="Load" sourceId="entryUnit" targetId="createUnit" type="entryToCreate"/>
        <dataFlow id="df3" name="Load" sourceId="entryUnit" targetId="updateUnit" type="entryToUpdate"/>
    </LINKS>


    <!-- RELATIONS SECTION -->
    <RELATIONS> <!-- Common section for 1:1, 1:N and N:M relations -->
```

```
        <ALL> <!-- All relations -->
            <selectorUnit id="selector#" name="selector#" parentId="FormPage" x="300#" y="0"/>
            <dataFlow id="flowSelectorForm#" name="Load" sourceId="selector#" targetId="entryUnit" type="preload"/>
        </ALL>

        <LAST> <!-- Last relation (used to close operations chain) -->
            <okLink id="okConnect" sourceId="createUnit" targetId="multiMessageUnit" message="Data created
correctly"/>
            <okLink id="okReconnect" sourceId="updateUnit" targetId="multiMessageUnit" message="Data modified
correctly"/>
        </LAST>
    </RELATIONS>

    <NMRELATIONS> <!-- Specific section only for N:M relations (Replaces RELATIONS)-->
        <ALL> <!-- All N:M relations -->
            <selectorUnit id="selector-#" name="selector#" parentId="FormPage" x="300#" y="0"/>
            <dataFlow id="flowSelectorForm-#" name="Load" sourceId="selector-#" targetId="entryUnit"
type="preload"/>

            <selectorUnit id="selectorNM%" name="selectorNM%" parentId="FormPage" type="roleCondition" x="0"
y="130%"/>
            <dataFlow id="flowSelectSelect%" name="Load" sourceId="selectorEntidad" targetId="selectorNM%"/>
            <dataFlow id="flowSelectEntry%" name="Load" sourceId="selectorNM%" targetId="entryUnit"
type="unitToEntryRole"/>

            <connectUnit id="connect%" name="connect%" x="850%" y="500"/>
            <dataFlow id="flowFormConnect%" name="Load" sourceId="entryUnit" targetId="connect%"
type="entryToConnect"/>
            <koLink id="koConnect%" sourceId="connect%" targetId="multiMessageUnit" message="Failed
creating/modifying data"/>

            <reconnectUnit id="reconnect%" name="reconnect%" x="850%" y="800"/>
            <dataFlow id="flowFormReconnect%" name="Load" sourceId="entryUnit" targetId="reconnect%"
type="entryToReconnect"/>
            <koLink id="koReconnect%" sourceId="reconnect%" targetId="multiMessageUnit" message="Failed modifying
data"/>
```

```
        </ALL>

        <FIRST> <!-- First N:M relation (used to start operations chain) -->
            <okLink id="okFirstConnect%" sourceId="createUnit" targetId="connect%"/>
            <okLink id="okFirstReconnect%" sourceId="updateUnit" targetId="reconnect%"/>
        </FIRST>

        <REMAINING> <!-- Second and subsequent N:M relations (used to continue operations chain) -->
            <okLink id="okConnect%" sourceId="connect%-1" targetId="connect%"/>
            <okLink id="okReconnect%" sourceId="reconnect%-1" targetId="reconnect%"/>
        </REMAINING>

        <LAST> <!-- Last relation (used to close operations chain) -->
            <okLink id="okConnect%+1" sourceId="connect%" targetId="multiMessageUnit" message="Data created
correctly"/>
            <okLink id="okReconnect%+1" sourceId="reconnect%" targetId="multiMessageUnit" message="Data modified
correctly"/>
        </LAST>
    </NMRELATIONS>

</PATTERN>


    <!-- # = iterative number for each relation -->
    <!-- % = iterative number for each NM relation -->
```

1.4 Ejemplo de ejecución del asistente

Se muestra un ejemplo completo de ejecución utilizando la entidad “Group” (entidad por defecto de WebRatio) y el patrón “AllInOne (default)” que es uno de los patrones por defecto que se entregan junto a la aplicación.

Para iniciar el asistente se debe iniciar WebRatio en primer lugar y abrir el modelo del proyecto web sobre el que se van a generar los elementos. Una vez abierto, se hace click sobre el icono marcado en rojo en la parte superior de la siguiente ilustración:

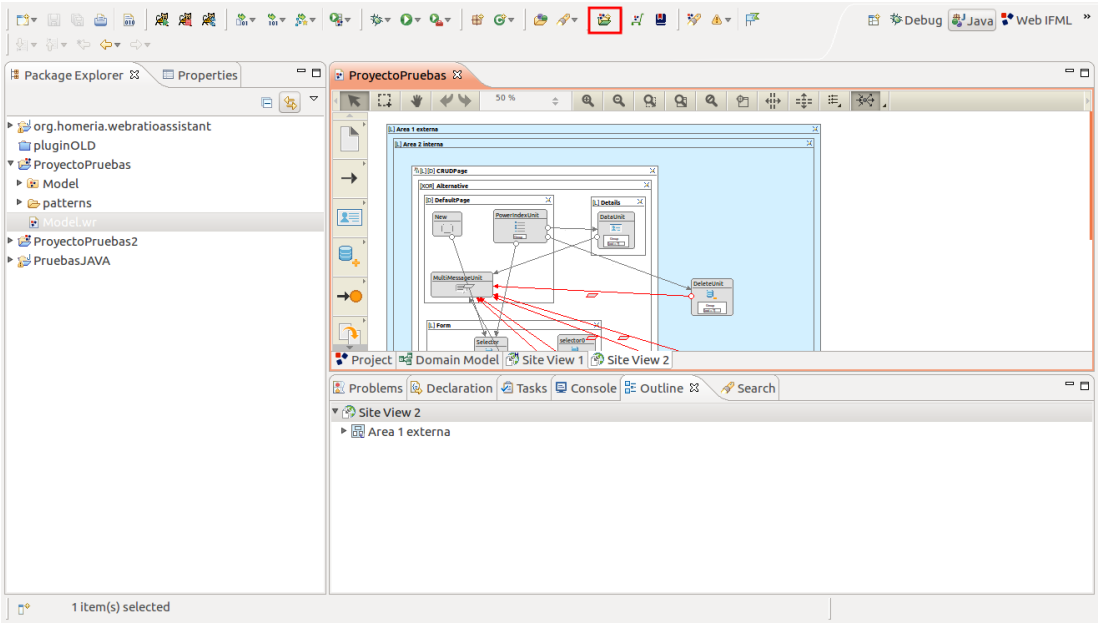

Ilustración 40: Ejecución del asistente. 1- Inicio del asistente

También se puede pulsar la combinación de teclas “Ctrl + F3” para iniciar el asistente de la misma forma.

Se abre el diálogo principal del asistente. En la parte izquierda se sitúan los selectores para elegir la entidad con la que queremos trabajar y el patrón que queremos generar en base a esa entidad. Previamente, se ha situado dicho patrón en la carpeta destinada a ello *(ver: 0*

Localización de los archivos*).* En la parte inferior se encuentra el árbol de *SiteViews* y *Areas* del proyecto donde se puede seleccionar una o más opciones.

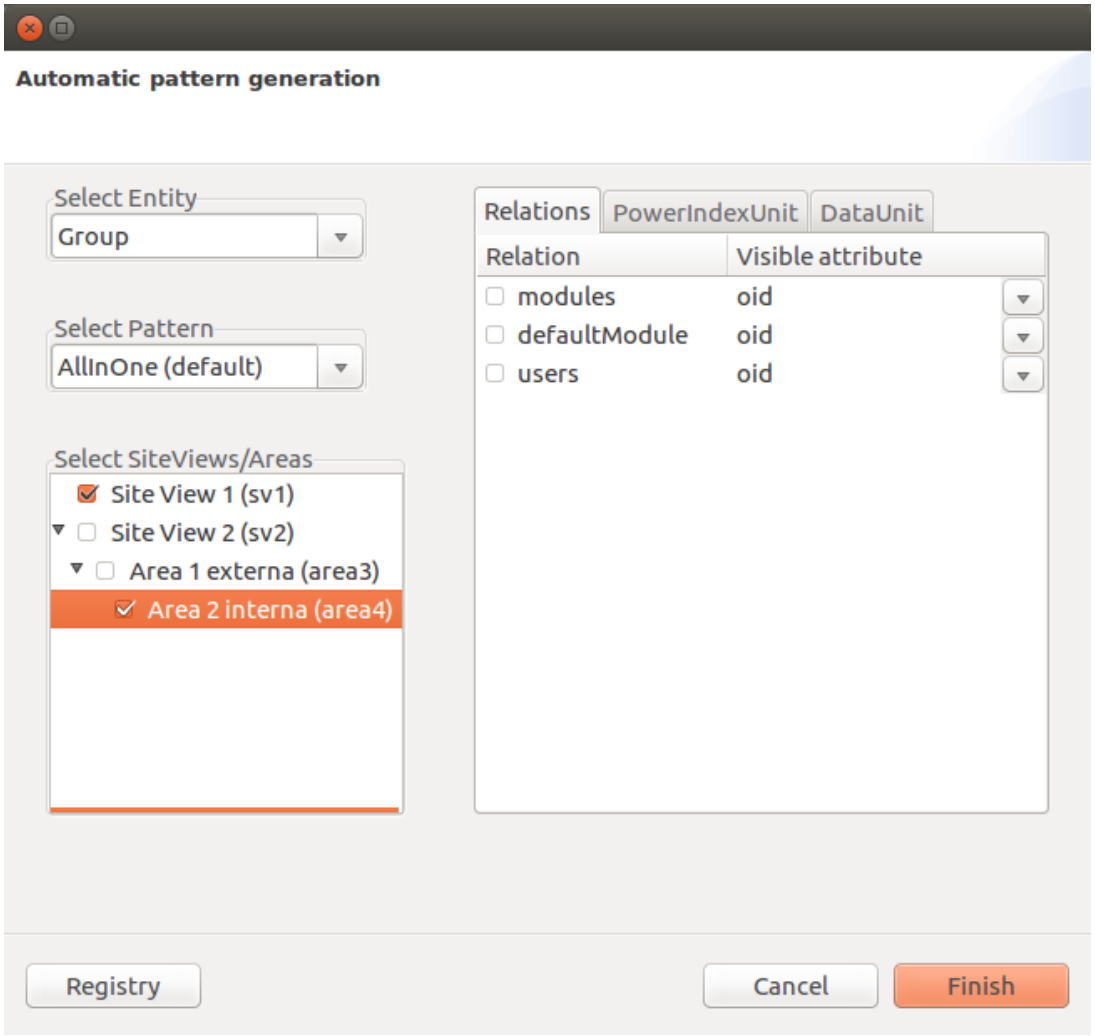

Ilustración 41: Ejecución del asistente. 2- Diálogo principal

Para este ejemplo, seleccionan tanto el *SiteView* (*sv1*) como el *Area* (*area4*), dicho *Area* está contenido dentro de otro que a su vez pertenece al *SiteView sv2.*

En la parte derecha del diálogo se puede encontrar un conjunto de pestañas que se generan dinámicamente, esto depende del patrón que se haya seleccionado y de las unidades contenidas en él. En este caso, se puede ver que hay tres pestañas:

- *Relations*: contiene las relaciones que posee la entidad seleccionada. Esta pestaña aparece cuando en el patrón seleccionado se definen unidades que dependen de relaciones (*p.e. CreateUnit* o *UpdateUnit*)

- *PowerIndexUnit*: contiene los atributos de la entidad que se desean mostrar en la lista. La pestaña aparece cuando se define una unidad del tipo powerIndexUnit.
- *DataUnit*: contiene los atributos de la entidad que se desean mostrar en la lista de detalle. La pestaña aparece cuando se define una unidad del tipo dataUnit.

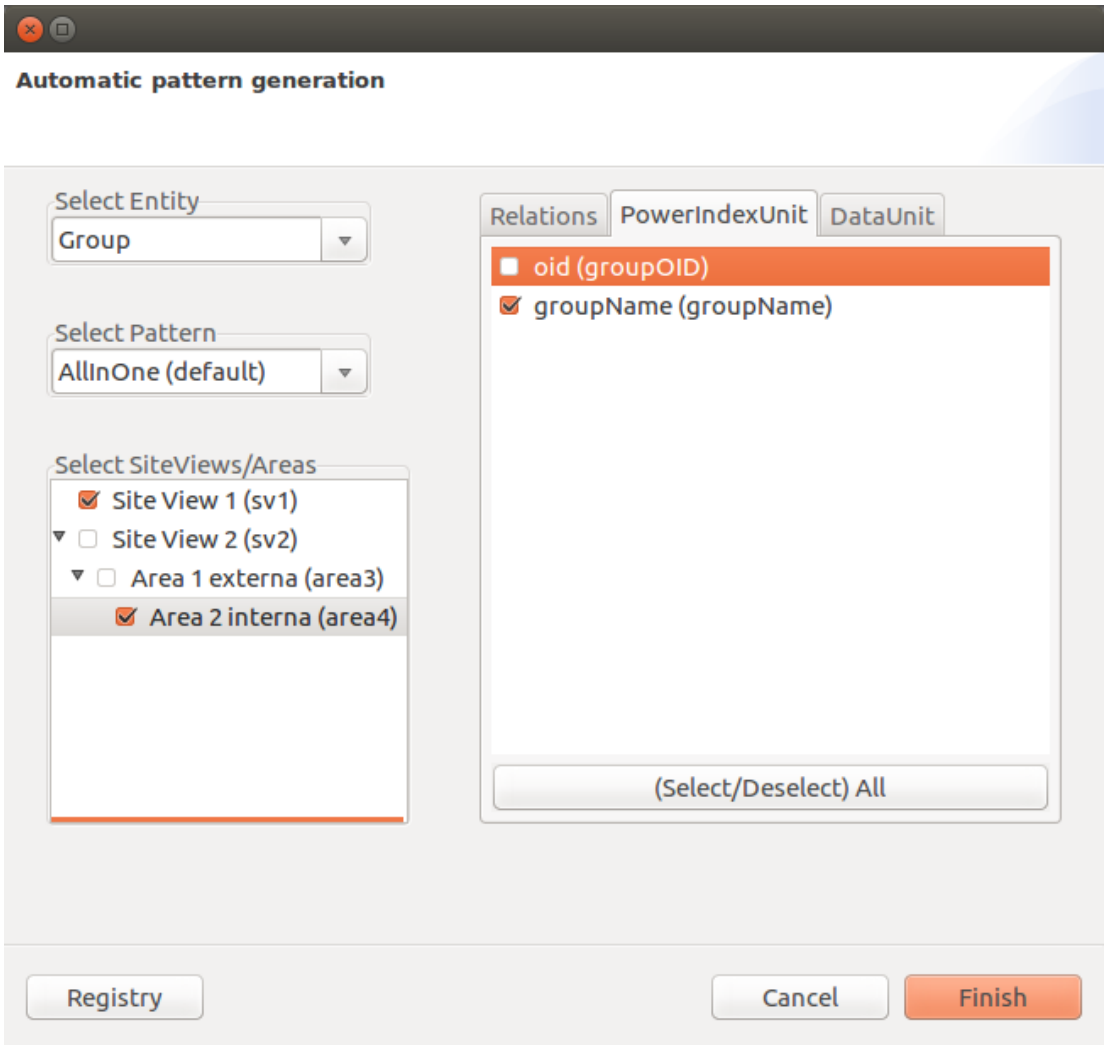

Ilustración 42: Ejecución del asistente. 3- Selección de atributos

Una vez se hayan seleccionado los atributos y las relaciones deseadas, se pulsa en el botón "*Finish*" para comenzar la generación. Al pulsar el botón se cerrará la ventana de diálogo actual y se abrirá otra más pequeña:

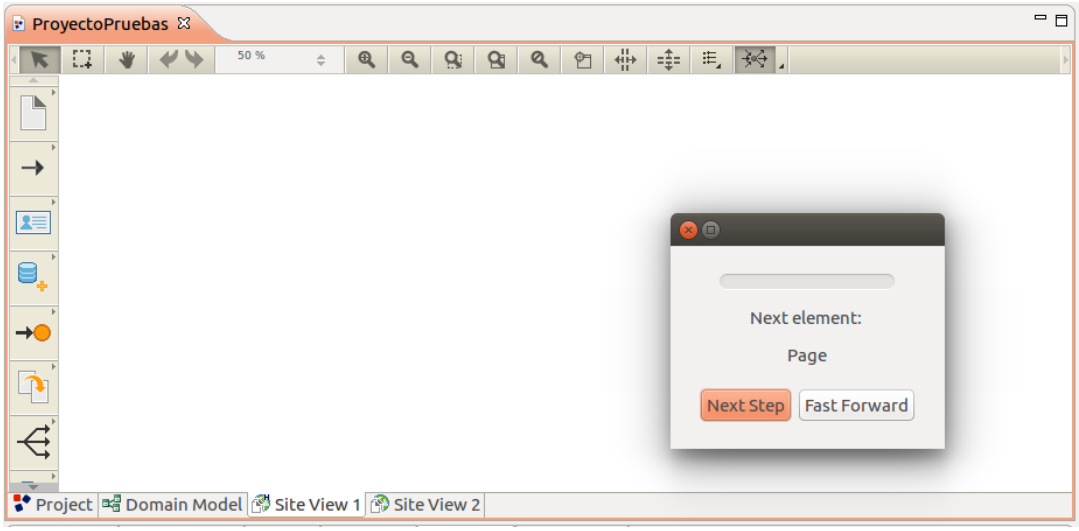

Ilustración 43: Ejecución del asistente. 4- Diálogo de generación.

Este diálogo muestra, en la parte superior, una barra de progreso para controlar el progreso de generación del patrón. Justo debajo se indica el siguiente elemento que se va a generar. Por último, dos botones donde el usuario elige si quiere generar solamente la siguiente unidad (*Next Step*) o, por el contrario, quiere generar todas las unidades restantes sin ningún tipo de pausa (*Fast Forward*).

Se muestran algunas imágenes donde se ve el proceso de generación paso a paso:

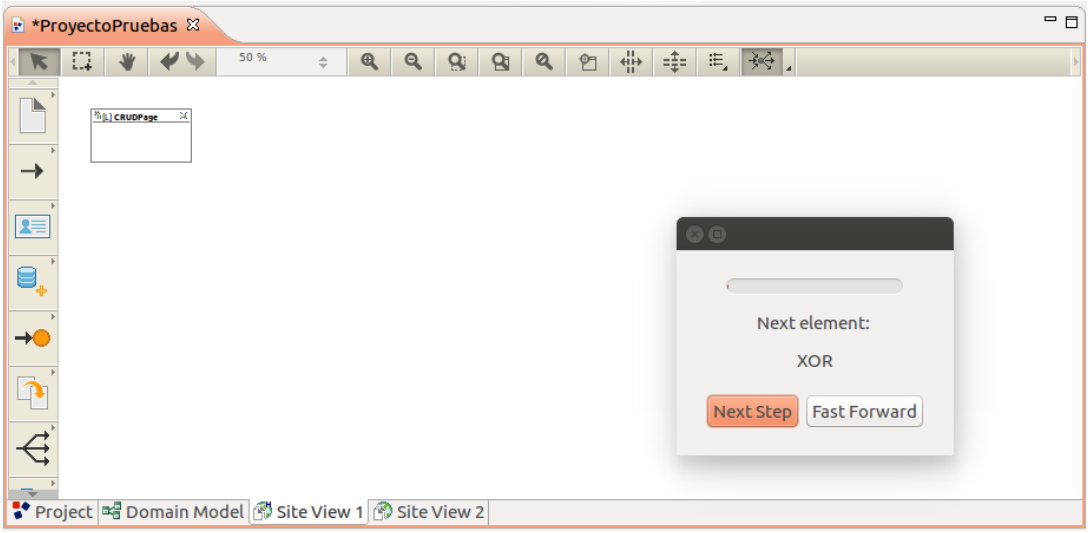

Ilustración 44: Ejecución del asistente. 5- Generación (1)

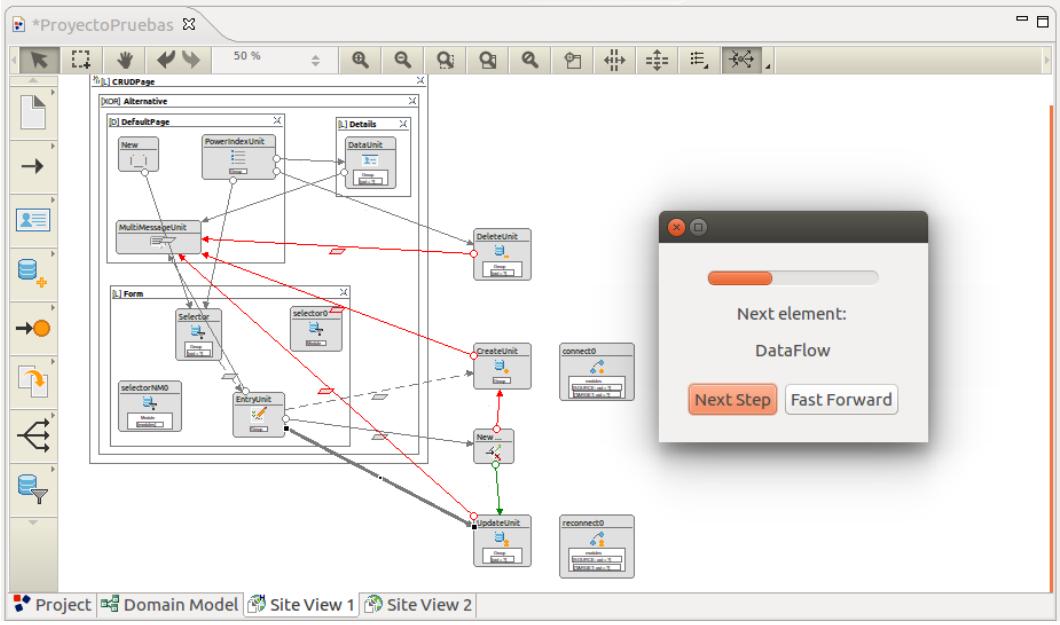

Ilustración 45: Ejecución del asistente. 5- Generación (2)

Una vez finalizada la generación en el primer *SiteView*, se cambia automáticamente al siguiente donde se generará dentro del *Area* interno, tal y como se ha elegido previamente en el asistente:

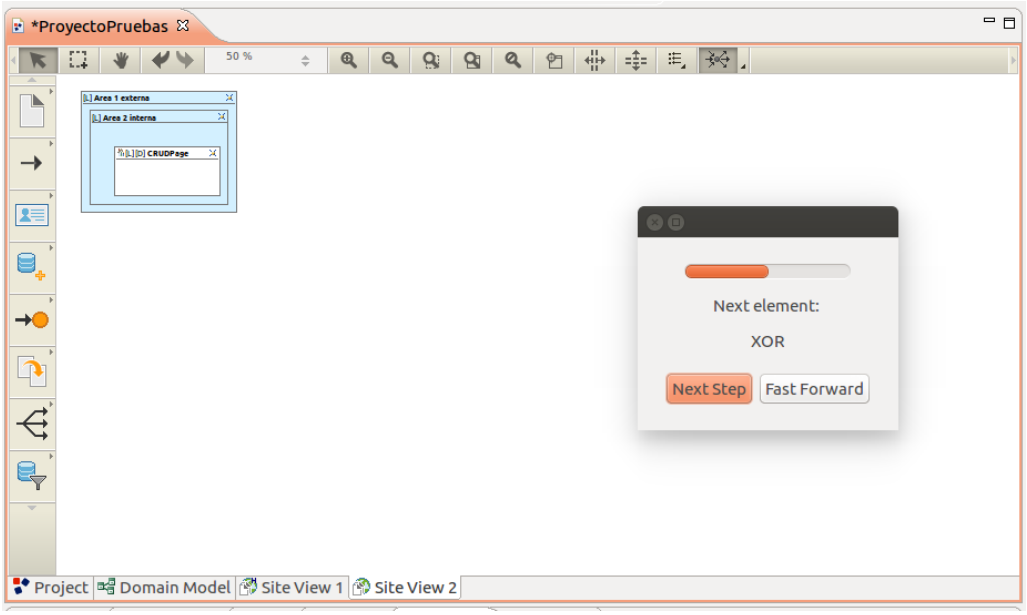

Ilustración 46: Ejecución del asistente. 5- Generación (3)

Una vez finaliza la generación, la ventana del asistente se cierra y podemos comprobar las unidades generadas en nuestro modelo.

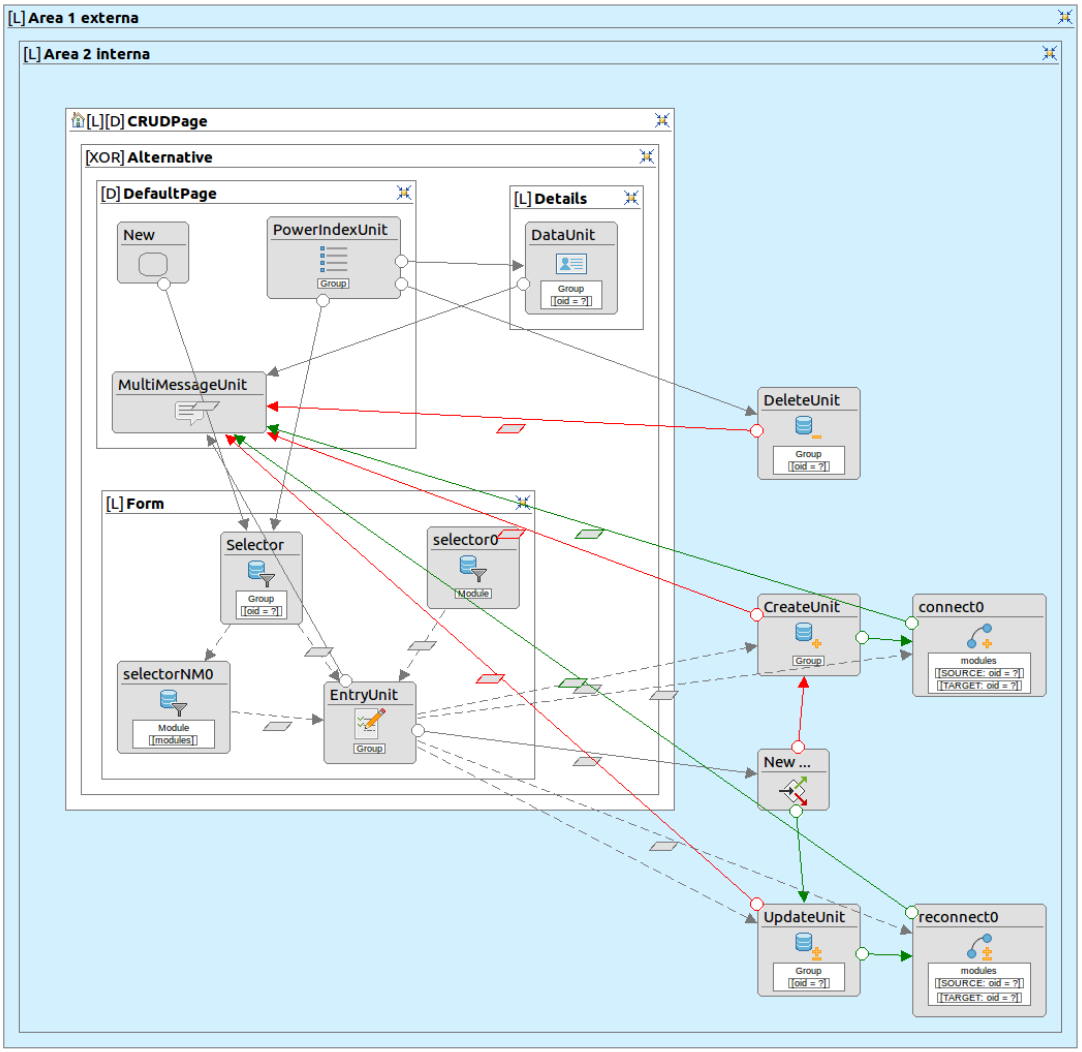

Ilustración 47: Ejecución del asistente. 5- Generación (4)

Para finalizar, se visualiza el Registro.

Se abre el asistente tal y como se explicó anteriormente y se pulsa en el botón "*Registry*" que se sitúa en la parte inferior izquierda del diálogo principal.

Al pulsar el botón, se abrirá una ventana con información estadística del proyecto en cuanto al uso de patrones. También se puede comprobar en qué SiteViews se han generado los patrones y qué unidades se han generado en cada patrón.

En la parte superior de la siguiente ilustración se visualiza la información general del proyecto. En la parte inferior, se muestran todos los patrones, las veces que se han generado y todos los elementos que se han generado utilizando la aplicación:

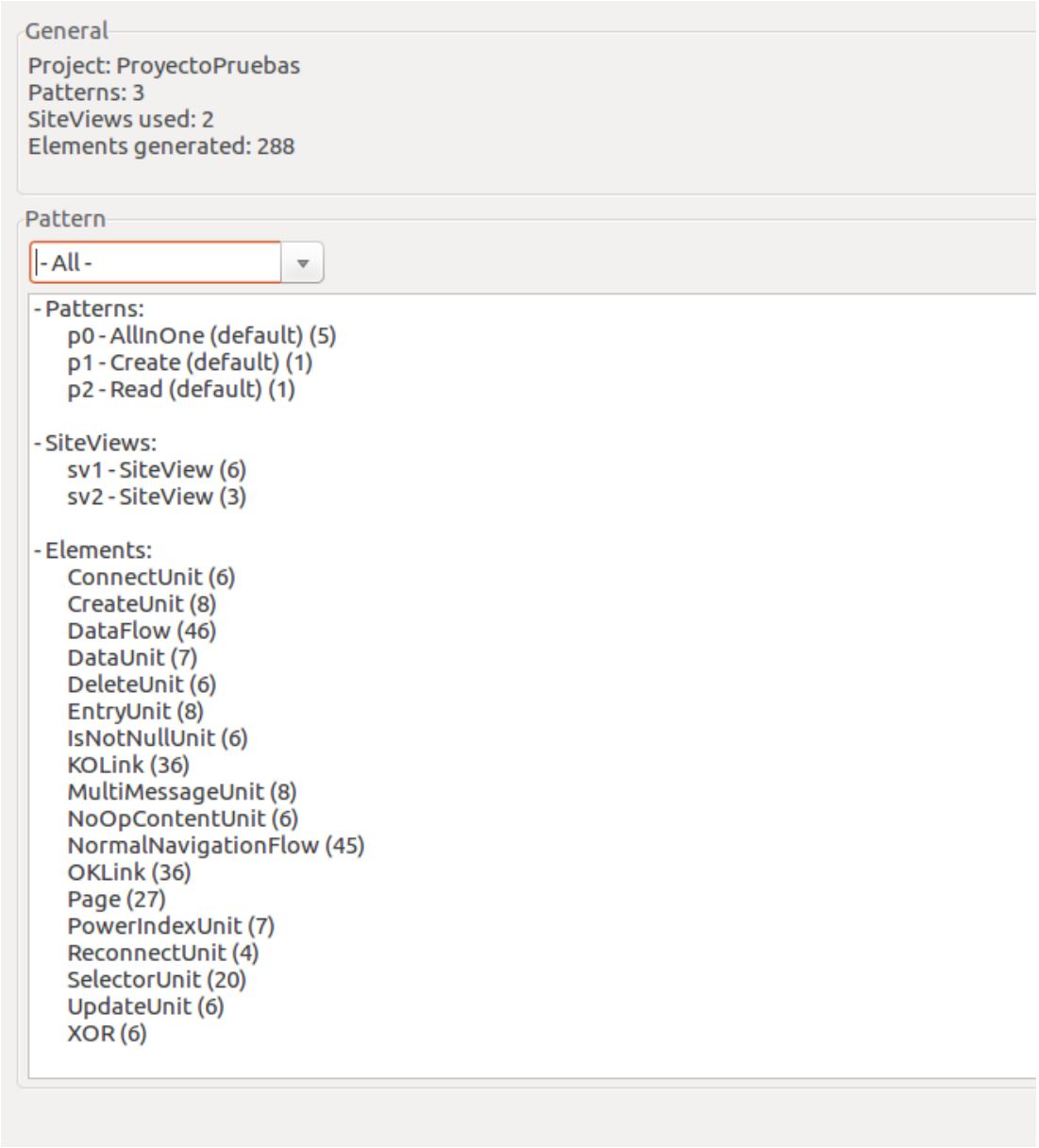

Ilustración 48: Ejecución del asistente. 6- Registro (1)

En la siguiente ilustración se muestra el patrón Create generado en otra ejecución:

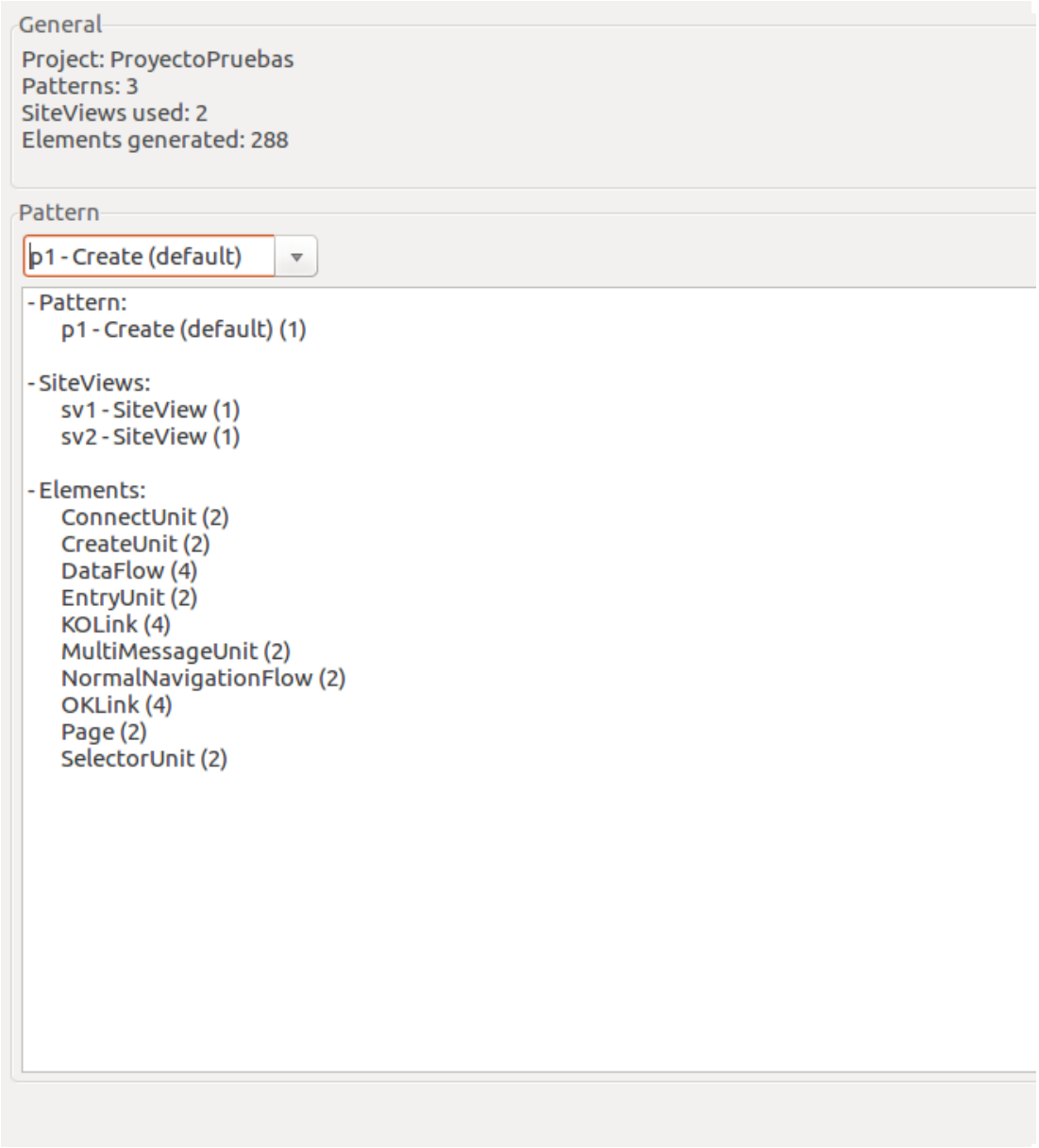

Ilustración 49: Ejecución del asistente. 6- Registro (2)

2 ANEXO II: Manual del Programador

2.1 Exportación del plug-in

Para empaquetar el plug-in y obtener el archivo apto para la ejecución en WebRatio se debe tener el proyecto en el workspace:

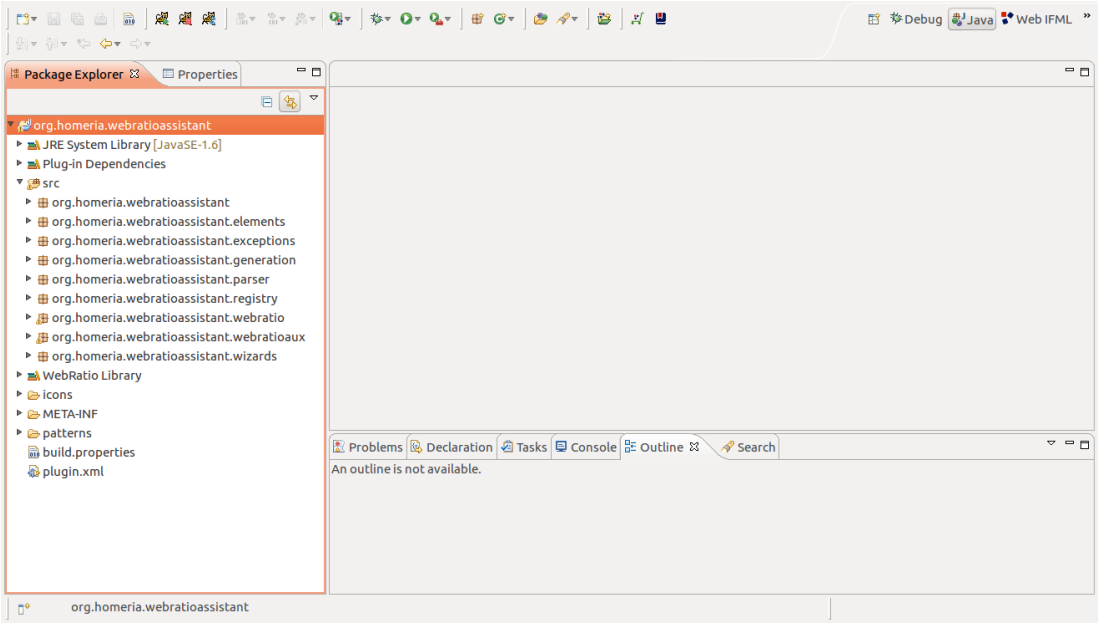

Ilustración 50: Exportación del plug-in. Paso 1: proyecto en el workspace

Seguidamente, se abre el menú **File -> Export** obteniéndose el siguiente diálogo:

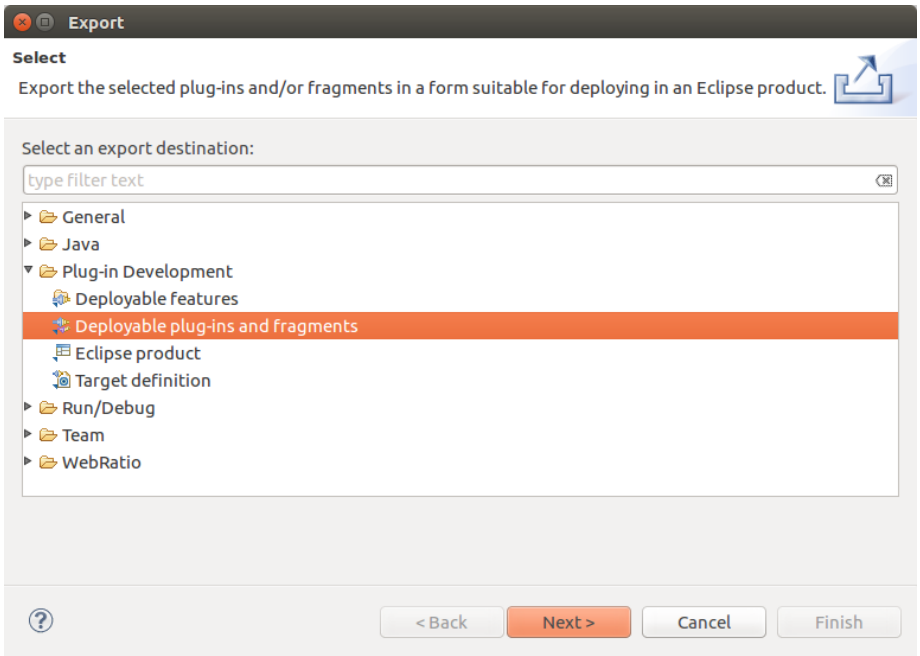

Ilustración 51: Exportación del plug-in. Paso 2: File -> Export

Seleccionándose "*Deployable plug-ins and fragments*" y pulsando en "*Next*" navegaremos hacia la ventana de exportación final:

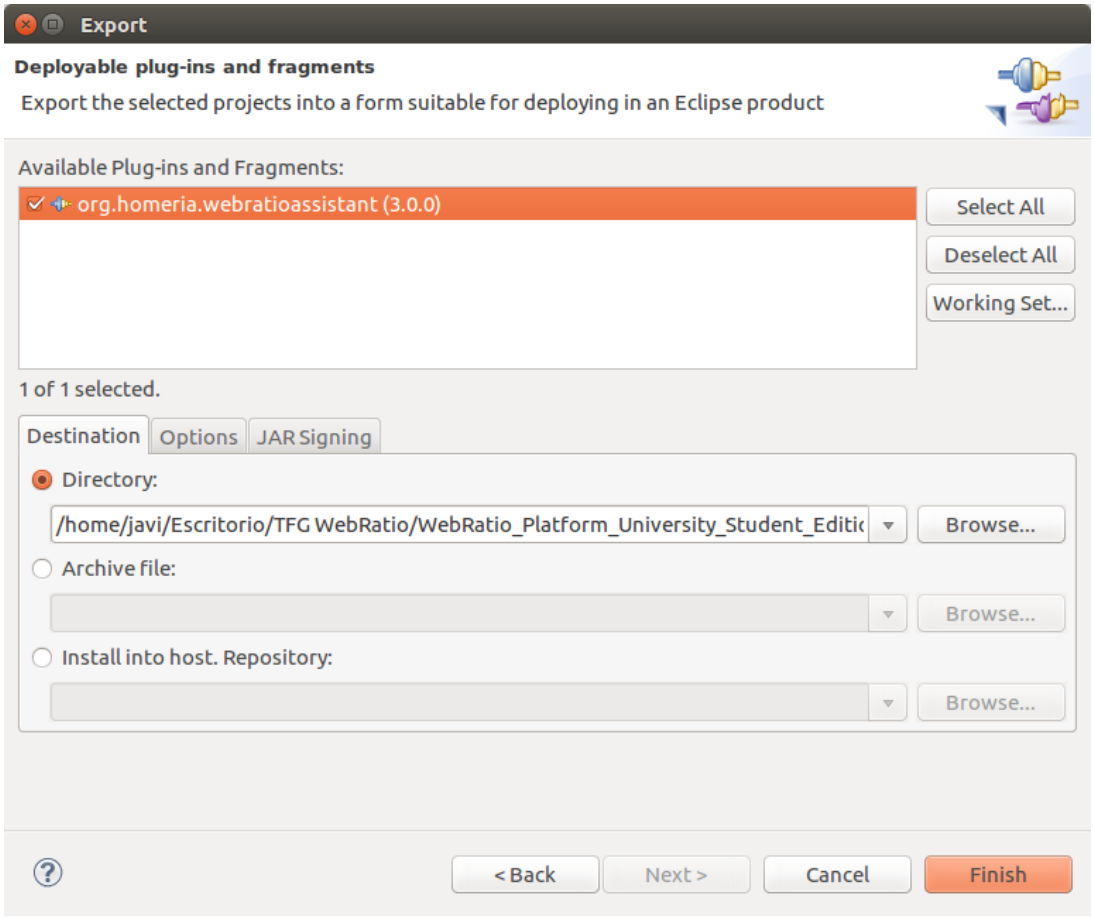

Ilustración 52: Exportación del plug-in. Paso 3: selección de plug-in y directorio

Dado que el proyecto está configurado para ser exportado como plug-in, se puede seleccionar el plug-in en la sección "*Available Plug-ins and Fragments*". No olvide elegir la localización donde será exportado, en la pestaña "*Destination*" y eligiendo el directorio con "*Directory*".

Pulsando el botón "*Finish*" se comenzará la exportación del plug-in. Se sabrá que ha terminado cuando se cierre la ventana que se abre una vez pulsado el botón, que tendrá un aspecto parecido a la siguiente imagen:

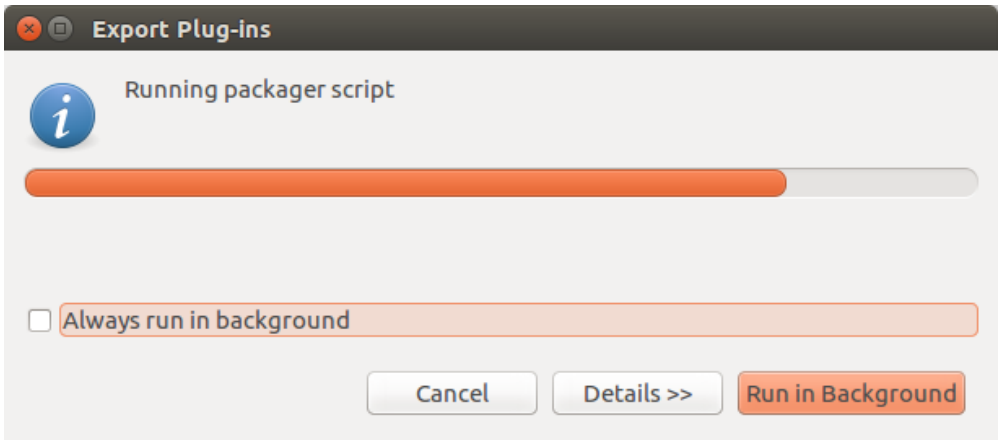

Ilustración 53: Exportación del plug-in. Paso 4: ventana final

2.2 Depuración en el desarrollo del plug-in

Para depurar durante el desarrollo del plug-in se necesitan dos instancias abiertas al mismo tiempo del entorno de desarrollo que estemos usando, en este caso WebRatio.

- **Instancia A:** en ella se ejecutará el plug-in, simulando las acciones del usuario necesarias hasta llegar al punto de interrupción.
- **Instancia B:** instancia desde la que se marcarán los puntos de interrupción y se realizará la depuración.

Importante: el workspace de la instancia B debe tener la misma versión del proyecto con la que se exportó el plug-in ejecutado por la instancia A.

- **Configuración de la Instancia A:**

Se debe modificar el archivo de configuración "**WebRatio.ini**" situado en el directorio principal de WebRatio añadiendo las siguientes líneas:

```
-vmargs
-Xdebug
-Xrunjdwp:transport=dt_socket,server=y,suspend=n,address=1044
```

Ejemplo del archivo WebRatio.ini:

```
WebRatio.ini (~/Escritorio/TFG WebRatio/WebRatio_Platform...ent_Edition_7.2.14.x64bit.Linu
Abrir                                                                          Guardar
1 -startup
2 plugins/org.eclipse.equinox.launcher_1.1.1.R36x_v20101122_1400.jar
3 --launcher.library
4 plugins/org.eclipse.equinox.launcher.gtk.linux.x86_64_1.1.2.R36x_v20101019_1345
5 -vmargs
6 -Xdebug
7 -Xrunjdwp:transport=dt_socket,server=y,suspend=n,address=1044
8 -XX:+UseConcMarkSweepGC
9 -XX:+UseFastAccessorMethods
10 -Xms40m
11 -Xmx512m
12 -XX:PermSize=64m
13 -XX:MaxPermSize=256m
14 -XX:+UseCompressedOops
15 -Duser.name=Javier Sierra Blázquez
.ini   Anchura de la pestaña: 8   Ln 5, Col 1   INS
```

Ilustración 54: Ejemplo del archivo WebRatio.ini

- **Configuración de la Instancia B:**

Con el proyecto importado, se abre el menú **Run → Debug configurations…**

Selecciona "**Java Remote Application"** y crea una nueva configuración. Como proyecto elige el proyecto del plug-in. En "**Connection properties**" debe introducir la dirección de la otra instancia (en este caso localhost) y el puerto indicado por address en la *configuración de la Instancia A* (en este caso 1044)

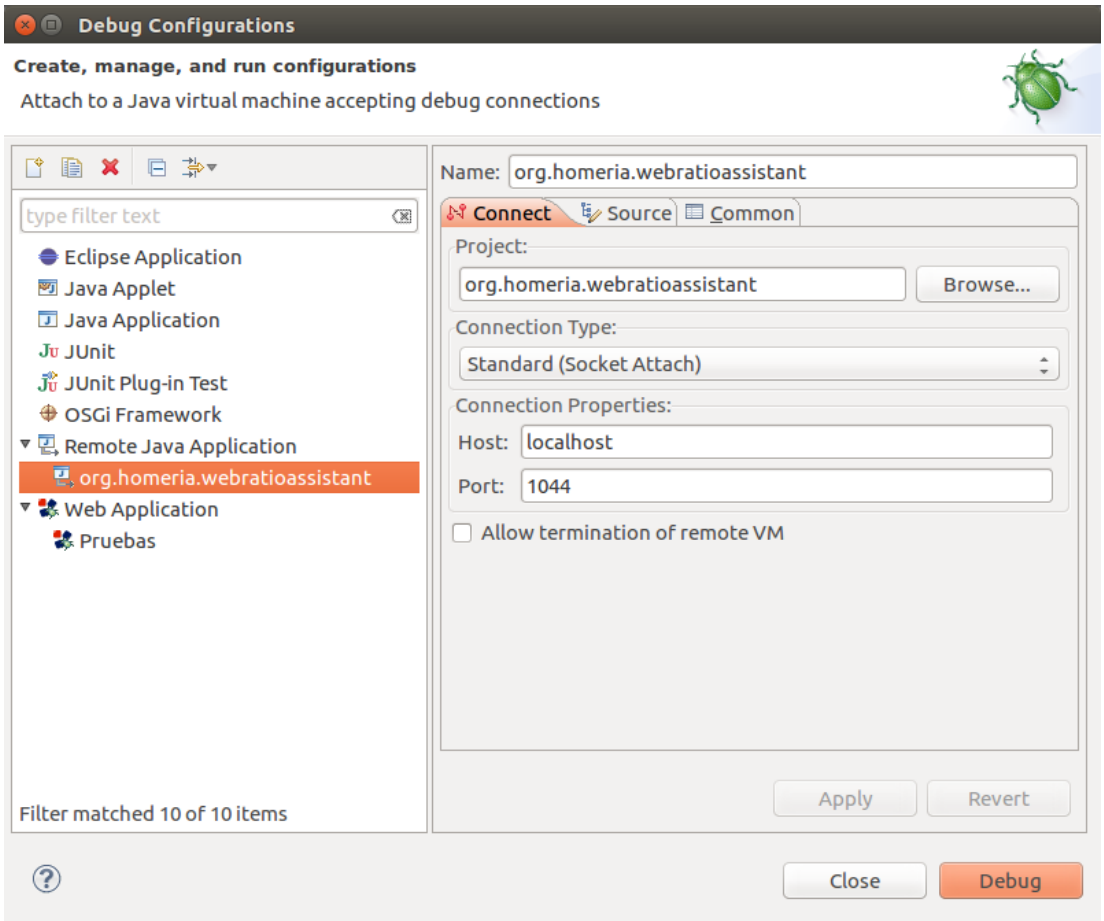

Ilustración 55: Debug configurations

Al iniciar Debug con esta configuración, se pondrá a la escucha en el puerto indicado esperando una conexión desde la otra instancia, que se producirá cuando se ejecute el plug-in.

En la instancia B es donde debe situar los puntos de interrupción y desde la Instancia A es desde donde navega por el plug-in hasta llegar a la interrupción. Cuando se produzca, el plug-in se congelará porque estará esperando que avance desde el menú de depuración de la Instancia B.

2.3 Ejemplo del archivo *Registry*

Se muestra un ejemplo del archivo *Registry* para aclarar las dudas que puedan surgir de su estructura. Se ha generado el patrón AllInOne (default) dos veces, una en cada SiteView y el patrón Create (default) una vez en un SiteView.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Registry>
  <Pattern id="p0" name="AllInOne (default)">
    <SiteView id="sv1" name="SiteView">
      <Page id="page9"/>
      <XOR id="alt3"/>
      <Page id="page10"/>
      <Page id="page11"/>
      <Page id="page12"/>
      <NoOpContentUnit id="gctu3"/>
      <MultiMessageUnit id="mssu3"/>
      <PowerIndexUnit id="pwu3"/>
      <DataUnit id="dau3"/>
      <EntryUnit id="enu3"/>
      <SelectorUnit id="seu7"/>
      <CreateUnit id="cru3"/>
      <DeleteUnit id="dlu3"/>
      <UpdateUnit id="mfu3"/>
      <IsNotNullUnit id="inn3"/>
      <SelectorUnit id="seu8"/>
      <SelectorUnit id="seu9"/>
      <ConnectUnit id="cnu3"/>
      <ReconnectUnit id="rcnu3"/>
      <NormalNavigationFlow id="ln31"/>
      <NormalNavigationFlow id="ln32"/>
      <NormalNavigationFlow id="ln33"/>
      <NormalNavigationFlow id="ln34"/>
      <NormalNavigationFlow id="ln35"/>
      <NormalNavigationFlow id="ln36"/>
      <NormalNavigationFlow id="ln37"/>
      <OKLink id="oln13"/>
      <OKLink id="oln14"/>
      <KOLink id="kln13"/>
      <KOLink id="kln14"/>
      <KOLink id="kln15"/>
      <KOLink id="kln16"/>
      <DataFlow id="ln38"/>
      <DataFlow id="ln39"/>
      <DataFlow id="ln40"/>
      <DataFlow id="ln41"/>
      <DataFlow id="ln42"/>
      <DataFlow id="ln43"/>
      <DataFlow id="ln44"/>
      <KOLink id="kln17"/>
      <DataFlow id="ln45"/>
      <KOLink id="kln18"/>
      <OKLink id="oln15"/>
      <OKLink id="oln16"/>
      <OKLink id="oln17"/>
      <OKLink id="oln18"/>
    </SiteView>
    <SiteView id="sv2" name="SiteView">
```

```
        <Page id="page13"/>
        <XOR id="alt4"/>
        <Page id="page14"/>
        <Page id="page15"/>
        <Page id="page16"/>
        <NoOpContentUnit id="gctu4"/>
        <MultiMessageUnit id="mssu4"/>
        <PowerIndexUnit id="pwu4"/>
        <DataUnit id="dau4"/>
        <EntryUnit id="enu4"/>
        <SelectorUnit id="seu10"/>
        <CreateUnit id="cru4"/>
        <DeleteUnit id="dlu4"/>
        <UpdateUnit id="mfu4"/>
        <IsNotNullUnit id="inn4"/>
        <SelectorUnit id="seu11"/>
        <SelectorUnit id="seu12"/>
        <ConnectUnit id="cnu4"/>
        <ReconnectUnit id="rcnu4"/>
        <NormalNavigationFlow id="ln46"/>
        <NormalNavigationFlow id="ln47"/>
        <NormalNavigationFlow id="ln48"/>
        <NormalNavigationFlow id="ln49"/>
        <NormalNavigationFlow id="ln50"/>
        <NormalNavigationFlow id="ln51"/>
        <NormalNavigationFlow id="ln52"/>
        <OKLink id="oln19"/>
        <OKLink id="oln20"/>
        <KOLink id="kln19"/>
        <KOLink id="kln20"/>
        <KOLink id="kln21"/>
        <KOLink id="kln22"/>
        <DataFlow id="ln53"/>
        <DataFlow id="ln54"/>
        <DataFlow id="ln55"/>
        <DataFlow id="ln56"/>
        <DataFlow id="ln57"/>
        <DataFlow id="ln58"/>
        <DataFlow id="ln59"/>
        <KOLink id="kln23"/>
        <DataFlow id="ln60"/>
        <KOLink id="kln24"/>
        <OKLink id="oln21"/>
        <OKLink id="oln22"/>
        <OKLink id="oln23"/>
        <OKLink id="oln24"/>
      </SiteView>
    </Pattern>
  <Pattern id="p1" name="Create (default)">
      <SiteView id="sv2" name="SiteView">
        <Page id="page17"/>
        <EntryUnit id="enu5"/>
        <MultiMessageUnit id="mssu5"/>
        <CreateUnit id="cru5"/>
        <SelectorUnit id="seu13"/>
        <ConnectUnit id="cnu5"/>
        <NormalNavigationFlow id="ln61"/>
        <KOLink id="kln25"/>
        <DataFlow id="ln62"/>
        <DataFlow id="ln63"/>
        <KOLink id="kln26"/>
```

```
      <OKLink id="oln25"/>
      <OKLink id="oln26"/>
    </SiteView>
  </Pattern>
</Registry>
```

3 ANEXO III: Actas de reuniones

Además de reuniones formales, se han intercambiado numerosos correos electrónicos entre los tutores y el alumno en materia de resolución de dudas puntuales y como informes de seguimiento.

3.1 Reunión del día 31 de marzo

- Lugar:

Sala i3 situada en el pabellón de informática, Escuela Politécnica de Cáceres.

- Asistentes:

Juan Carlos Preciado Rodríguez (Tutor)
Roberto Rodríguez Echeverría (Co-Tutor)
Javier Sierra Blázquez (Alumno)

- Contexto:

Siendo la primera reunión que se realiza, marca el inicio de la etapa de desarrollo del proyecto. Durante la misma se realiza el análisis de requisitos, poniendo encima de la mesa los requisitos funcionales y no funcionales de la aplicación, ordenando su prioridad y fijando las líneas generales y objetivos del desarrollo del proyecto.

- Decisiones tomadas:

Se necesita simplificar la interacción con el usuario, modificando la interfaz gráfica de la aplicación.

Se propone realizar una descripción de patrones externa, donde el usuario pueda definir los elementos (páginas, unidades, enlaces…) que componen un patrón. Actualmente, el plug-in genera los patrones CRUD (Create, Read, Update, Delete) y el patrón All-In-One que resulta en una combinación de los anteriores.

Modificar la velocidad de generación del patrón.

Implementar un registro con los patrones generados y toda la información asociada a ellos. Gracias a este registro se podrían obtener datos estadísticos y de uso del plug-in. Además, se podría interactuar con el patrón sobre el modelo, pudiendo modificar/eliminar a nivel de patrón.

A nivel de código, separar la Interfaz de Usuario del Negocio.

Registrar el plug-in con una licencia.

- Conclusiones:

Se acuerdan las siguientes acciones a realizar hasta la próxima reunión:

- Documentar los pasos que se van realizando.
- Generar un documento de requisitos inicial.
- Conseguir la descripción de patrones XML para implementar un pequeño prototipo del Registro.
- Modificar la interfaz de usuario.

Próxima reunión en 15 días, lunes después de Semana Santa.

3.2 Reunión del día 21 de abril

- Lugar:

Sala i3 situada en el pabellón de informática, Escuela Politécnica de Cáceres.

- Asistentes:

Juan Carlos Preciado Rodríguez (Tutor)

Roberto Rodríguez Echeverría (Co-Tutor)

Javier Sierra Blázquez (Alumno)

- Contexto:

Después de analizar todos los datos obtenidos en la primera reunión, se realiza esta segunda en la que se aclaran dudas surgidas por la reunión anterior y se profundiza en la modificación de la interfaz de usuario.

- Decisiones tomadas:

Se propone utilizar una ventana de dialogo con un solo paso, la selección de entidad quedaría en un selector en la esquina superior izquierda. El widget para la velocidad de generación estaría en la parte inferior de la ventana en posición horizontal. Añadir la opción de seleccionar modo interactivo. Este modo interactivo permite avanzar paso a paso en la generación. Los elementos se van creando mientras el usuario va haciendo click en siguiente.

Si se elige el modo interactivo reducir el tamaño de la ventana y que aparezca una especie de reproductor para pulsar en siguiente.

Si no se elige el modo interactivo minimizar la ventana.

- Conclusiones:

Tras aclarar las dudas sobre el repositorio de patrones y profundizado en la modificación de la interfaz de usuario, no se fija la próxima reunión en una fecha concreta sino cuando los objetivos fijados en esta reunión sean cumplidos.

3.3 Reunión del día 4 de mayo

- Lugar:

Sala i3 situada en el pabellón de informática, Escuela Politécnica de Cáceres.

- Asistentes:

Roberto Rodríguez Echeverría (Co-Tutor)

Javier Sierra Blázquez (Alumno)

- Contexto:

Tras haber comenzado el desarrollo del repositorio de patrones, se plantea el problema de la especificación de elementos que se generan dinámicamente con las relaciones que selecciona el usuario (SelectorUnit, ConnectUnit, etc.)

- Decisiones tomadas y conclusiones:

Se llega a la conclusión de crear unas etiquetas de control que definan comportamiento opcional. El condicionante son las relaciones que el usuario elija en la interfaz de usuario. De esta forma se define en el archivo XML las unidades y los enlaces que se tienen que generar en caso de que el usuario elija relaciones de un tipo o de otro.

3.4 Reunión del día 2 de junio

- Lugar:

Sala i3 situada en el pabellón de informática, Escuela Politécnica de Cáceres.

- Asistentes:

Juan Carlos Preciado Rodríguez (Tutor)
Roberto Rodríguez Echeverría (Co-Tutor)
Javier Sierra Blázquez (Alumno)

- Contexto:

Con la funcionalidad del repositorio de patrones casi desarrollada, se informa de los cambios realizados.

Se explica la estructura del patrón, atributos definidos en los elementos, la clase creada que funciona como *"parser"*, la jerarquía de clases implementada con la finalidad de serializar la información obtenida en el patrón para su posterior utilización durante la generación de los elementos. También se informa de pequeños cambios en la interfaz de usuario y la adición de una nueva unidad: ReconnectUnit.

- Decisiones tomadas y conclusiones:

Se desglosan las tareas que quedan por hacer, fijando las prioridades y eliminando de la hoja de ruta aquellas menos prioritarias que, por cuestión de tiempo, no se pueden realizar.

Tareas por hacer:

- Finalizar el desarrollo de patrones.

- Implementar el registro: guardar los identificadores de las unidades generadas y asociarlas con el patrón que las genera. De esta forma podemos llevar un control de dónde están generados los patrones y en un futuro poder borrar todas las unidades de un patrón de golpe.
- Control de errores.
- Generación paso a paso.
- Documentación interna en inglés (traducir el código del proyecto anterior y documentarlo)
- Simplificar la declaración del patrón.

3.5 Reunión del día 3 de julio

- Lugar:

Sala i3 situada en el pabellón de informática, Escuela Politécnica de Cáceres.

- Asistentes:

Juan Carlos Preciado Rodríguez (Tutor)
Roberto Rodríguez Echeverría (Co-Tutor)
Javier Sierra Blázquez (Alumno)

- Contexto:

Con la mayoría de las funcionalidades implementadas, se plantea la simplificación de la estructura del patrón, siendo más declarativa.

También se plantean ciertos problemas con el registro. Al no tener acceso a la API de WebRatio, no se conocen las llamadas para realizar las funcionalidades deseadas, tales como: seleccionar elementos en el modelo y/o borrarlos.

- Decisiones tomadas y conclusiones:

Se diseña un cambio en el apartado de relaciones de la estructura del patrón, siendo más declarativa, eliminando las estructuras de control “if” y “for” de la estructura del patrón.

Se toma la decisión de realizar el registro para almacenar los datos de los patrones generados, aunque no se puedan realizar más acciones que visualizar los datos con fines estadísticos

3.6 Reunión del día 20 de julio

- Lugar: Instalaciones de Homeria Open Solutions.

- Asistentes:

  Miguel Ángel Preciado Rodríguez (Anfitrión)

  Javier Sierra Blázquez (Alumno)

- Contexto:

Con la aplicación casi terminada, se propone una reunión con Homeria para presentar la nueva versión del plug-in y explicar las nuevas funcionalidades con la idea de obtener información de retroalimentación para mejorar el acabado final del proyecto.

- Decisiones tomadas y conclusiones:

La reunión fue satisfactoria con resultado positivo y buenas impresiones.

Además, se propuso una mejora en la interfaz de usuario para mostrar más información al usuario: añadir títulos en las columnas de los atributos cuando el patrón incluye unidades de contenido.

3.7 Cronología de reuniones

En la siguiente ilustración se puede ver una cronología de las reuniones enmarcadas dentro de las fases del proyecto.

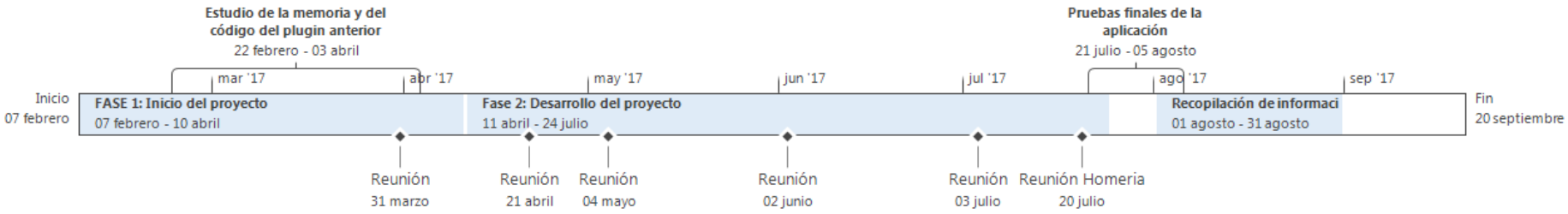

Ilustración 56: Cronología de reuniones y fases del proyecto

4 ANEXO IV: Contenido del CD

La presente documentación adjunta un CD con los siguientes directorios:

- Documentacion: contiene la documentación del proyecto en formato PDF con el nombre: **Memoria_TFG_Javier_Sierra_Blazquez.pdf**
- Codigo fuente: en su interior se encuentra todo el código fuente del plug-in desarrollado.
- Patrones y esquema: alberga los patrones de ejemplo y el esquema de la estructura. Los archivos son:
  - AllInOne (default).xml
  - Create (default).xml
  - Delete (default).xml
  - Read (default).xml
  - Update (default).xml
  - schema.xsd
- Plug-in: contiene el plug-in compilado. Archivo: **org.homeria.webratioassistant_3.0.0.jar**