Universidad de la República

Facultad de Ingeniería
Instituto de Computación

Informe de Proyecto de Grado

Prototipo de migración a plataforma web de herramienta gráfica para el modelado de aplicaciones y procesos de negocio.

Estudiantes
Daniela Andrea Varela Garrido
Oscar Flores Rodriguez

Tutor
Ariel Sabiguero Yawelak

Responsables
Ricardo Rezzano
Ernesto Rovan

Tribunal
Daniel Meerhoff
Leandro Scasso
Omar Viera

2020

Montevideo, Uruguay

Resumen del trabajo

A lo largo de este documento se presentará el proyecto de grado de fin de la carrera universitaria Ingeniero en Computación realizado por Daniela Varela y Oscar Flores. Durante el mismo se presentarán los pasos seguidos para el desarrollo de una nueva herramienta gráfica web de proceso de negocios, solicitada por la empresa KSIT. Esta solicitud se debe a la necesidad de una actualización tecnológica de la herramienta actual.

Se dará una gran importancia al análisis la herramienta de diseño gráfico pre-existente, AdaptorDesigner, como también a sus capacidades y elementos, para así poder determinar qué es lo que se debe migrar y de qué forma. También fue fundamental el análisis del manejo de los datos y la integración con el motor de base de datos. Este punto es uno de los principales durante el transcurso del proyecto, permitiendo generar los requisitos y las bases necesarias para el desarrollo del software.

Se mostrarán las tecnologías web y swing gráficas estudiadas antes de la selección final de JointJs (web) y los componentes necesarios para la comunicación con los motores de base de datos, y se describirá el desarrollo del frontend y el backend, así como las decisiones de diseño tomadas.

Al finalizar el documento presentaremos la gestión de proyecto, así como las conclusiones obtenidas. Se presentará también lo que se considera el trabajo futuro a realizar sobre el prototipo

Palabras claves

Ingeniería de software, actualización tecnológica, tecnologías web, tecnologías swing, herramienta de diseño gráfico, proceso de negocios, grafos, Adaptor, Java, JavaScript, Jquery, MyBatis, JointJS.

Índice de contenido

Índice de ilustraciones

Índice de tablas

1. Introducción

El siguiente capítulo describe los objetivos del proyecto, así como también la explicación de las diferentes fases en que se dividió el mismo. Además, especifica los conocimientos necesarios para su comprensión según el público objetivo.

1.1. Objetivos

El trabajo a realizar en este proyecto de grado consiste en la actualización tecnológica de una herramienta gráfica de diseño para el modelado de procesos de negocio que pertenece a la suite Adaptor, siempre manteniendo la compatibilidad con las demás herramientas pre-existentes.

La empresa KSIT actualmente ya posee una herramienta gráfica para el modelado de procesos de negocio, que permite a los usuarios desarrolladores generar de una forma simple y rápida, diferentes modelos que luego son ejecutados por otra plataforma. Si bien este dibujador es el que utilizan actualmente, el lenguaje en el que fue implementado (Visual Basic), en la actualidad se encuentra en desuso y hace difícil su mantenimiento, además solo funciona en plataformas Windows con base de datos Microsoft SQL-Server.

El objetivo principal es la de determinar si es posible la transición de una plataforma de escritorio en una web, manteniendo las características requeridas, abandonando el apego de las tecnologías usadas entre las cuales se presentan el motor de bases de datos y el lenguaje del desarrollo. Determinar si esta transición es manejable y fácil de implementar será estudiado durante el transcurro del documento.

Por estos motivos se investigaron varias tecnologías tanto web como de escritorio, se hizo un relevamiento y estudio del dibujador pre-existente. Se realizaron varios prototipos, seleccionando en un paso posterior el que permitiera implementar la mayor parte de los requerimientos relevados y resolver algunos de los problemas que la herramienta actual posee.

Al desarrollarse un prototipo solamente, varias de las funcionalidades del dibujador actual quedan fuera del alcance del proyecto, estas serán implementadas por el cliente, por eso fue fundamental que el prototipo se diseñara de tal forma que se pueda extender con facilidad y a su vez brindar una documentación clara y completa de como continuar con este trabajo.

Se espera poder obtener una clara idea del valor agregado que le puede generar este desarrollo a la empresa interesada. Así mismo, determinar si el producto generado es una mejora con respecto al actual.

1.2. Fases del proyecto

El proyecto tuvo varias etapas, se comenzó estudiando la herramienta pre-existente y realizando un relevamiento de los requerimientos del cliente, así como los problemas que esta posee. Paralelamente se realizó el estudio del arte de las posibles tecnologías que se podían utilizar para dibujo de procesos de negocio. Para poder seleccionar una tecnología para el prototipo final, se implementaron varios prototipos más básicos utilizando las diferentes bibliotecas (Java y JavaScript) que se adaptaban mejor a los requerimientos del proyecto.

Para la elección de la tecnología definitiva se fueron incorporando funcionalidades a los prototipos (uno web y uno de escritorio) y presentándoselas al cliente en sucesivas reuniones, hasta que se optó por la tecnología a utilizar.

Luego de la investigación realizada y de haber seleccionado una tecnología definitiva se procedió al diseño y desarrollo de un prototipo funcional que satisfaga los requerimientos que se mencionarán en el capítulo 3. Se suma a esto la implementación de varias de las actividades que actualmente posee la herramienta de diseño pre-existente.

El testeo del sistema y validación de datos generados se fue realizando a medida que se avanzaba con el desarrollo, ya que cada actividad implementada requería la validación de los datos que generaba.

Con respecto a la documentación de la nueva herramienta, a medida que se fueron completando las diferentes etapas del proyecto se fueron generando documentos intermedios que luego se utilizaron para la elaboración de este informe.

En todo el proceso de trabajo se tuvieron reuniones cada 2 o 3 semanas con el tutor y los responsables donde se decidían los nuevos requerimientos a agregar al prototipo.

1.3. Conocimientos previos y público objetivo

Para poder comprender en su totalidad este documento es recomendable tener conocimientos técnicos de base de datos (a nivel de usuario), programación web, servicios web y Java. Si bien se intentó explicar las tecnologías a utilizar, sin conocimientos previos de las mismas no se podrá comprender en totalidad las decisiones de diseño tomadas. No es necesario tener conocimiento sobre la herramienta para la cual se va a realizar una re-ingeniería, ya que en una sección posterior se dará un pantallazo general de la misma.

Por otro lado, para poder continuar con la implementación del prototipo desarrollado se recomiendan conocimientos de EJB, servicios RESTful, Java, JavaScript, HTML 5, MyBatis, JQuery, CSS, la biblioteca JointJS, formato de archivos JSON y también conocimientos sobre el dibujador pre-existente y algunas de las tablas de la base de datos utilizada.

1.4. Organización del Informe

Este documento está organizado de forma que sea accesible y fácil de leer. Se sigue una estructura de capítulos que permite agrupar y presentar de forma ordenada el trabajo realizado.

Durante el Capítulo 2 se realiza el análisis de la herramienta existente y la información técnica de la misma. También se exponen los objetivos del trabajo realizado y una presentación del estado del arte de las tecnologías existentes a utilizar durante el desarrollo de la herramienta.

El análisis del problema es formulado durante el Capítulo 3 donde se plantean los requerimientos del mismo. Se define el alcance del desarrollo de la herramienta durante este capítulo.

En el Capítulo 4 se presenta la solución planteada. Se describen los prototipos desarrollados para la selección de las tecnologías utilizadas, las decisiones de diseño pertinentes, así como la estructura y el diseño de la solución. También se exponen los casos de prueba y testeo para ver si se cumple con el alcancé planteado. Al finalizar el capítulo se describe cómo se debe agregar una nueva actividad al producto.

La gestión realizada durante todo el proyecto se expone durante el Capítulo 5, contemplando el cronograma y el manejo de los riesgos presentados durante el desarrollo del mismo.

Durante el Capítulo 6 se presentan las conclusiones obtenidas durante la duración del proyecto, como también los posibles trabajos a futuro realizables sobre el producto entregado.

Al finalizar el documento se encontrarán varios apéndices que complementan los datos presentados en los diferentes capítulos que los referencian. Dicha información es relevante para las decisiones tomadas, pero no se presenta en el cuerpo principal del informe debido a que no es imprescindible para la comprensión del mismo y apartarían al lector del foco de los temas relevantes.

El apéndice 9.1 describe con un poco más de detalle las bibliotecas estudiadas. Durante el apéndice 9.2 hace un comparativo sobre las ventajas y desventajas de implementar un prototipo web o de escritorio. Como se guardan los componentes de los grafos en la base de datos es explicado durante el apéndice 9.3 y mediante el apéndice 9.4 se detallan los ejemplos que se utilizaron como base del desarrollo y las pruebas realizadas.

2. Estado del Arte

Durante el siguiente capítulo se realiza un análisis de la herramienta que se utiliza actualmente por la empresa KSIT y se estudian las funcionalidades ofrecidas por la misma, con especial atención a la forma de integración del dibujador con las bases de datos utilizadas.

Se analizan las bases de datos presentes en la herramienta pre-existente, ya que las mismas representan un aspecto muy importante a tener en cuanta durante el desarrollo de la nueva herramienta, principalmente porque se desea realizar modificaciones al uso de las mismas.

También se exponen los resultados obtenidos por la investigación y prueba de algunas bibliotecas existentes para la manipulación de grafos y el estudio de nuevas tecnologías.

2.1. Descripción de herramienta pre-existente - AdaptorDesigner

AdaptorDesigner es la herramienta pre-existente de diseño que se utiliza en la empresa KSIT para el desarrollo de sus soluciones gráficas. Se presenta como "*una herramienta de modelado que ayuda a los desarrolladores a crear de una manera metódica, rápida y efectiva soluciones de Adaptor*" [11]. Se les denomina desarrolladores a los usuarios de la misma. Durante esta sección estudiaremos con más detalle la versión 7.1.1 que se está utilizando. Esta herramienta, a la que denominaremos dibujador, está incluida en la suite Adaptor.

"*Adaptor es una plataforma para el desarrollo de procesos de negocio especializada en el sector financiero. Dentro de sus características destacan el manejo de canales, la administración de archivos y documentos y la integración con las Aplicaciones Corporativas.*" [18]

Esta plataforma trabaja con un modelo de proceso abstracto, en un lenguaje sumamente computarizado, siendo sumamente complicado su comprensión y manejo por humanos. El dibujador soluciona este problema comportándose como intérprete del lenguaje, leyendo y escribiendo especificaciones en las bases de datos presentándolas en una interfaz gráfica amigable para su utilización. Dicha interfaz gráfica se presentará durante este capítulo.

2.2. Análisis general de herramienta pre-existente - AdaptorDesigner

AdaptorDesigner se presenta como un dibujador (herramienta de diseño gráfico) sobre la plataforma Adaptor.

El mismo ofrece las siguientes funcionalidades que se describirán de forma general y abstracta según la información obtenida del manual del usuario:

- Soluciones integradas
- Exportación en XML
- Ingeniería reversa
- Auto documentación
- Chequeo de errores

Las soluciones integradas definen que la herramienta AdaptorDesigner se centra en el concepto de proyecto mediante el cual "*se permite a los desarrolladores visualizar y trabajar con todas las partes de una solución en forma simultánea.*" [11]. Un proyecto es un agrupador de modelos, estos tienen diferentes funcionalidades, propósitos y permiten especificar un conjunto de funcionalidades de Adaptor.

Exportación en XML permite guardar los proyectos en archivos con formato XML, buscando que se logre la redistribución de las soluciones. Esto significa que el almacenamiento de las soluciones en un formato conocido, permitiendo que programas externos tengan acceso al manejo de los archivos.

La ingeniería reversa es una de las principales funcionalidades proporcionadas por la herramienta AdaptorDesigner. Mediante la misma se puede obtener un proyecto a partir de los datos de una solución implementada y guardada en base de datos.

Las soluciones implementadas con la herramienta AdaptorDesigner quedan documentadas automáticamente. Permitiendo a los desarrolladores tener acceso a sus proyectos y flujos de manera simple.

Al momento de compilar las soluciones desarrolladas con AdaptorDesigner se verifica la integridad y consistencia de los datos. El dibujador no debe permitir dejar inconsistente las bases de datos con su utilización.

Las mencionadas características son requeridas y por lo tanto se desarrollará el nuevo software con el objetivo de mantener y mejorar las mismas, sumando las que se consideren necesarias.

a) Información Técnica

AdaptorDesigner se presenta como una aplicación de escritorio desarrollada en Visual Basic, conectada a dos bases de datos Microsoft SQL Server, que mediante un dibujador de grafos permite la creación de distintos modelos que se presentarán más adelante.

En primera instancia se describirá la aplicación de escritorio, es decir el dibujador AdaptorDesigner. También se explicarán los conceptos con los cuales trabaja, sus características principales y los problemas que presenta. En segundo lugar, se describirán ambas bases de datos, explicando la funcionalidad de cada una de ellas, siguiendo el mismo mecanismo utilizado para el dibujador.

1 AdaptorDesigner

AdaptorDesigner es una aplicación de escritorio que permite a los desarrolladores que la utilizan manejar de forma gráfica los distintos modelos pertenecientes a un proyecto. La herramienta AdaptorDesigner fue desarrollada en Visual Basic cuya principal funcionalidad es permitir el desarrollo de los diferentes grafos que representan los modelos.

Una vez iniciada, se nos presentará la opción de conectarnos a diferentes servidores de base de datos, los cuales contendrán distintas versiones de la base de datos Designer y MMProdat, estas se detallan más adelante en el documento.

Ilustración 1: Pantalla de inicio AdaptorDesigner

Una vez seleccionada la base de datos, se pedirá el ingreso del usuario, contraseña y rol a utilizar en la misma, permitiendo seleccionar esto último si el usuario ingresado es correcto.

Inicializado el programa, se presenta la interfaz del mismo con el área de trabajo en blanco. Se dispondrá de las opciones de crear o abrir proyectos, o de realizar la operación Ingeniería reversa. Esta última opción permite seleccionar uno de los proyectos almacenados en la base de datos Designer y cargar todos los modelos asociados al mismo. Mediante el uso de varias ventanas y pestañas, se permitirá cambiar el modelo en el cual se trabaja. Existen distintas barras de herramientas de actividades y enlaces para poder dibujar el grafo y un explorador de modelos que presenta la estructura del proyecto actual y separa los modelos por tipo.

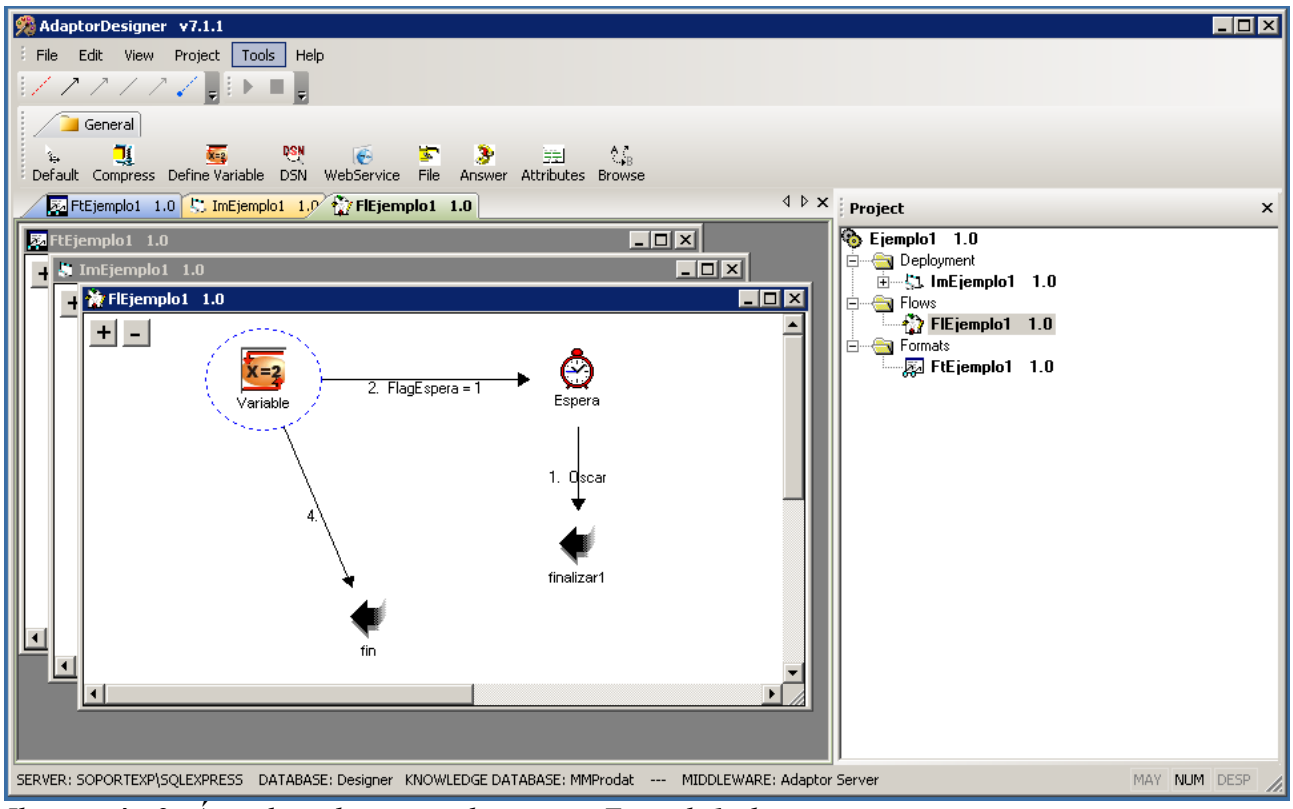


Ilustración 2: Área de trabajo con el proyecto Ejemplo1 abierto

Presentadas las generalidades de la herramienta AdaptorDesigner, se describen los componentes principales de la misma, centrando la atención en éstos a la hora de desarrollar una nueva versión del producto.

A continuación, se presentan las principales abstracciones del producto.

Proyecto

El proyecto es el concepto principal de trabajo del AdaptorDesigner. "*Un proyecto es un agrupador de modelos. Existen varios tipos de modelos con funcionalidades muy distintas. Cada uno de estos modelos tiene un propósito en particular y permite especificar en él un conjunto de funcionalidades de Adaptor. Cualquier solución de Adaptor que se quiera resolver mediante AdaptorDesigner, deberá comenzar por definir un proyecto. Luego a ese proyecto se le incorporaran modelos hasta lograr especificar la solución esperada.*" [11]

Modelo

Los modelos permiten, mediante las actividades y transiciones, desarrollar funcionalidades Adaptor. Algunas categorías de modelos son las siguientes:

- Implantación
- Flujo
- Formato

- Unidades conceptuales

El modelo de implantación contiene las entidades de alto nivel (DSNs, Entidades externas, petición, flujos, etc.).

Los flujos contendrán las soluciones desarrolladas.

Los formatos tendrán las características de los objetos de salida.

Actividad

"*Una actividad es una entidad especializada que cumple una función especifica en el tiempo. Existen diversos tipos de actividades como por ejemplo: actividades orientadas al manejo del sistema de archivos, interacción con bases de datos, formateo de mensajes, etc. Las actividades pueden ser independientes o utilizar información proveniente de otras actividades.*" [11]. Las actividades son un conjunto de entidades variables, es decir, hay una necesidad de actualizar e incorporar actividades durante la vida útil del producto, ya que las mismas pueden ser creadas, actualizadas o eliminadas a medida que los desarrolladores trabajen en distintos proyectos. Cada actividad cuenta con su ventana de propiedades personalizada.

Transición

"*Son entidades cuyo propósito es relacionar actividades*" [11]. Permitiendo determinar el orden y la forma en que las actividades se ejecutan. Existen varios tipos de transiciones:

- Evento: Indican la secuencia en que las actividades se ejecutan en el flujo.
- Error: Determinan el camino a seguir en caso de haber un error al ejecutar una actividad.
- Uso: Indican las dependencias en el uso de actividades.
- Relación: Permiten relacionar dos actividades de tipo "Tabla" en los diagramas de Unidades Conceptuales.
- Conceptual: Permiten poner etiquetas relacionadas a las distintas actividades.

2 Estudio de las bases de datos

En la siguiente sección se presenta uno de los componentes críticos del producto, las bases de datos. Originalmente existen dos bases de datos relacionadas, la base donde se almacenan los componentes lógicos de los flujos (MMProdat) y la base relacionada con los componentes gráficos de los mismos (Designer). Se prestó especial atención a los contenidos y a las funciones que cumple cada una de ellas ya que se debe mantener la base MMProdat y se espera poder eliminar la base Designer.

A continuación, se describen en detalle cada una de ellas:

I. **MMProdat**

La base de datos MMProdat es con la que se trabajará durante el desarrollo de la nueva herramienta. En la misma se guarda toda la información relacionada a los flujos implementados con la cual se trabajó fuertemente. Esta base no contiene datos gráficos de los flujos y tampoco información de los proyectos realizados con el AdaptorDesigner.

A continuación, se presentarán las tablas más relevantes de MMProdat, entre las 334 que la conforman.

LnFluMsg0172

Es la tabla de los Flujos compilados. Un flujo de mensajes representa la definición de un proceso, es decir, un conjunto de actividades vinculadas lógicamente, de acuerdo a lo especificado por el responsable de la configuración del flujo.

El motor de ejecución de flujos realiza, de acuerdo a lo especificado en su base de conocimientos, la evaluación de un conjunto de condiciones del mensaje y del contexto, y tiene la capacidad de elegir ejecutar diferentes acciones de acuerdo al resultado de ésta evaluación. Es posible ejecutar una secuencia tan compleja como sea necesario y evaluar paso a paso el resultado para decidir cómo proseguir el proceso.

Para esto se cuenta con una secuencia ordenada de líneas, equivalentes a un bloque de código, donde cada línea implica la ejecución de una acción, la cual pertenece a un grupo de acciones predefinidas. Cada línea tiene una condición de entrada o precondición y una de salida o pos-condición.

Las acciones que pueden ejecutarse desde una línea de flujo pueden agruparse en distintas categorías:

- Manipulación de campos a través de su Identificador financiero único, desde ahora IUFF (Identifier Unique Financial Field) (`AddIUFF, RemoveIUFF, etc`)
- Formateo de Mensajes (`Parser, Build`)
- Entrega y Recepción de Mensajes sobre Entidades Externas (`Send, Recive`)
- Manejo de Archivos (`Comprimir, Encriptar, Copiar, Borrar, etc`)
- Sincronización de Flujos (`WaitForValue, SelectValue, AddUpValue`)
- Varios (`Comentarios, Control de Flujo, Return, Meta-Data`)

La ejecución del flujo es secuencial descendente, a menos que explícitamente una línea redefina la próxima línea a ejecutar. Las líneas pueden ejecutar las acciones de forma sincrónica o asincrónica, dependiendo de la naturaleza de las acciones y las necesidades del proceso.

A continuación, los campos principales y la descripción de los mismos [14]:

Campo	Descripción
F1EnEx11800172	Entidad externa de origen
F1Mens01650172	Identificador del flujo
F1Vers01650172	Versión del flujo
Ordina0172	Ordinal del campo
F2EnEx11800172	Entidad externa de salida
F2Mens01650172	Identificador de mensaje
F2Vers01650172	Versión del flujo
IdMsOr0172	Cabezal ISO, solo en mensajes ISO
Descri0172	Línea de descripción
InCond0172	Condición de entrada
OutCon0172	Condición de salida
ErrCon0172	Condición de error
Accion0172	Acción a ejecutar
CpOrDto0172	Campo IUFF, Datos de origen y destino
Parame0172	Parámetros extras de la acción
EjVece0172	Números de veces a ejecutar la línea
IdLine0172	Identificador de línea
ProxLi0172	Próxima línea a ejecutar
ModEje0172	Modalidad de ejecución (Synchr. / Asynchr.)
LogEje0172	Trace True/False
LofErr0172	Error Trace True/False
TimOut0172	Time Out
F1DSND01790172	ID del Data Source Name (Desde ahora DSN). Datos orígenes de los receive o send de datos
F1Vers01790172	Versión DSN

Tabla 1: Descripción de la tabla LnFluMsg0172

EntExter1180

En esta tabla se guarda la información de las entidades externas a utilizar por las actividades.

Campo	Descripción
NombrePr1180	Entidad externa de origen
Version1180	Identificador del flujo
Direcc1180	Versión del flujo
Ciudad1180	Ordinal del campo
Pais__1180	Entidad externa de salida
Telefo1180	Identificador de mensaje
NumFax1180	Versión del flujo
E_mail1180	Cabezal ISO, solo en mensajes ISO
Notas_1180	Línea de descripción

Tabla 2: Descripción de la tabla EntExter1180

CbFluMsg0171

"*Contiene las características generales del flujo de mensajes. Contiene: el identificador de flujo y el canal que lo inicializa, una condición de inicio del flujo y una de fin del mismo.*" [14]:

Campo	Descripción
F1EnEx11800171	Entidad externa
F1Mens01650171	Versión de flujo
F1 Vers01650171	Descripción
IndCond0171	Condición de entrada
OutCon0171	Condición de salida

Tabla 3: Descripción de la tabla CbFluMsg0171

II. **Designer**

Es la segunda base de datos donde se guardan todos los datos relacionados a los proyectos, así como toda la información gráfica asociado al producto. Esta base de datos es la que se busca eliminar al desarrollar la nueva herramienta y al utilizar una tecnología diferente para el dibujado de los flujos, la información gráfica almacenada en la misma no es relevante para el nuevo desarrollo. La solución implementada para el trabajo con los proyectos se expone durante la sección de arquitectura y diseño de la solución.

Tablas relevantes:

Proyecto8010

Almacena la información de los proyectos registrados.

Modelo__8011

Almacena la información de los modelos registrados.

RelModel8015

Indica qué modelos pueden participar en qué proyectos.

StorProy8030

Guarda la información gráfica de los proyectos y sus claves para poder posteriormente efectuar la ingeniería reversa.

StorMode8031

Guarda la información gráfica de los modelos y sus claves para poder posteriormente efectuar la ingeniería reversa.

2.3. Tecnologías gráficas existentes para modelado de grafos.

En la sección anterior se realizó un análisis de la herramienta que ya posee la empresa KSIT para poder comprender mejor los requerimientos que se debían implementar y cuáles eran las tecnologías más adecuadas para el desarrollo del proyecto.

El objetivo principal de este proyecto fue la actualización de dicha herramienta tanto a nivel tecnológico como funcional, teniendo en cuenta las complicaciones con los que se enfrentaba el cliente al utilizándola. El principal problema fue que el dibujador que actualmente se utiliza solo funciona en plataformas Windows con base de datos Microsoft SQL Server, por lo tanto, se debía buscar una solución que se pudiera ejecutar sobre sobre cualquier sistema operativo y utilizar cualquier motor de base de datos.

Para poder presentar una nueva solución tecnológica se tuvo que investigar las bibliotecas existentes que permitieran modelar en forma gráfica modelos de procesos de negocios. Por eso mismo en la siguiente sección se hace un análisis de las tecnologías investigadas que fueron el punto de partida del diseño del prototipo.

Se investigaron bibliotecas tanto para aplicaciones web como para aplicaciones de escritorio, tratando de implementar con cada una de ellas un prototipo muy básico. Luego, de acuerdo a ciertos criterios se seleccionó una biblioteca para la web y otra de escritorio (JointJS y JGraphX).

El criterio que se utilizó para para poder seleccionar cada una de estas bibliotecas es que sean software libre (en su mayoria), tengan una comunidad activa y que sus proyectos sigan activos. Además, se busco que las mismas tengan varias funcionalidades importantes para el desarrollo del proyecto (ver Tabla 4: Comparativa entre bibliotecas)

Por último, se eligió una tecnología web, teniendo el cliente la decisión de elegir cuál de las dos opciones se adapta mejor a sus necesidades. A continuación, se da un pantallazo de cada una de las tecnologías investigadas, por más información de las mismas consultar el apéndice 8.1 – Bibliotecas estudiadas.

A continuación, se presentarán las bibliotecas estudiadas.

a) GOJS

Biblioteca JavaScript para implementar diagramas interactivos y complejos. Ofrece funcionalidades como *drag and drop*, *copy-paste*, menús, diseños automáticos, deshacer, paletas y un sistema de herramientas extensible para personalizar operaciones. La desventaja de esta biblioteca es que es paga. No se Implementó ningún prototipo con la misma ya que se necesitaría licencia pero se presenta porque soluciona la gran mayoría de los requerimientos presentados en el proyecto. [2]

b) JointJS

Biblioteca con licencia open source que permite la creación de diferentes tipos de diagramas, como diagramas de flujo HTML5 o BPMN. A su vez permite personalizar los nodos y enlaces. Además, provee las funcionalidades de *drag and drop* y manejo de eventos.

c) JavaFX

Biblioteca para aplicaciones Swing con la capacidad de crear las pantallas a partir de archivos tipo XML. Esta biblioteca no provee funcionalidades de grafos, por lo tanto, se deben implementar a mano todas dichas funcionalidades (*drag and drop*, conexión mediante flechas, tipos de flecha, etc.).

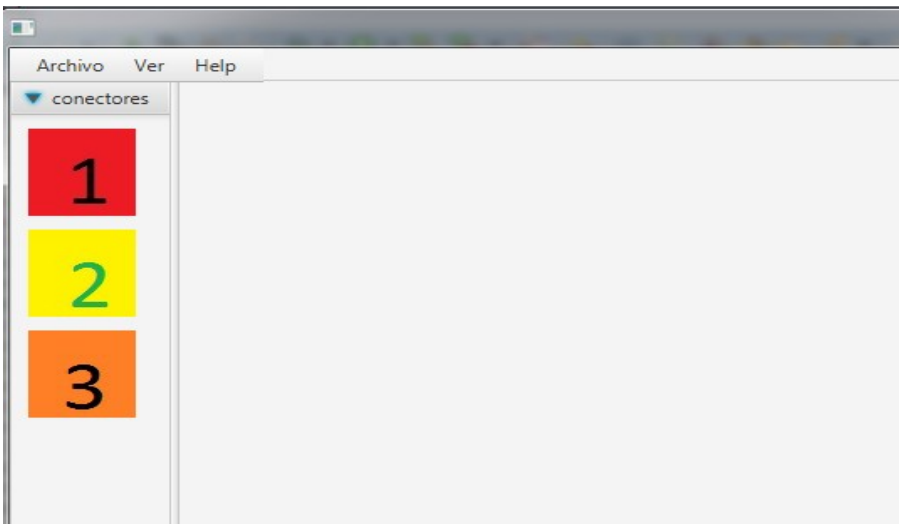


Ilustración 3: pruebas con JavaFX

d) CytoscapeJS

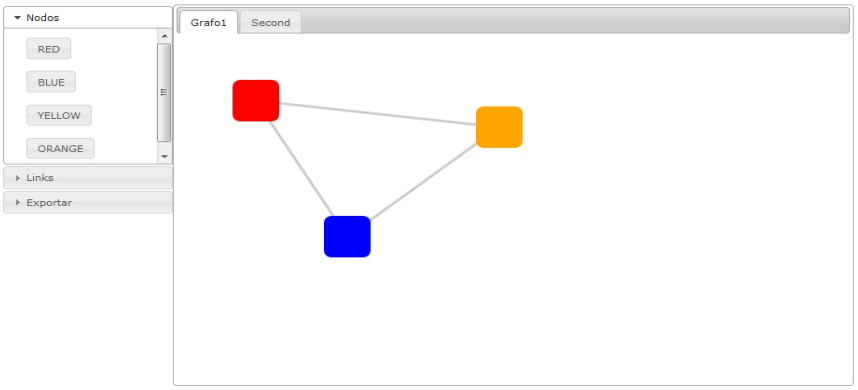


Ilustración 4: Prototipo con Cytoscape JS

Biblioteca JavaScript para manipulación de grafos, que provee diferentes *plugins* con nuevas funcionalidades, como paneles de propiedades o creación de enlaces mediante *drag and drop*. Se llegó a implementar un prototipo básico, pero se descartó ya que no se llegaron a resolver algunos de los requerimientos del problema (como ser implementar una barra de herramientas desde donde se permita crear nodos simplemente arrastrándolos desde esta barra hasta la pantalla de trabajo, o diferentes estilos para los enlaces) y se encontraron algunos errores en los *plugins* que esta biblioteca provee (el que permite crear enlaces entre nodos no tiene el funcionamiento esperado).

e) D3JS

Biblioteca JavaScript para visualización de datos en forma interactiva. Utiliza tecnologías como HTML5, CSS y SVG. Se pueden generar grafos interactivos con la misma, pero se enfoca más a la visualización y no tanto a la creación de grafos en forma interactiva. Se realizaron pruebas con la misma pero no se llegó a implementar un prototipo. [6]

f) JGraphX

Biblioteca Swing Java para implementar la diagramas interactivos y grafos. Es simple de usar, con una rápida curva de aprendizaje. Permite dibujar nodos y conectarlos entre sí mediante enlaces, asignándoles estilos a éstos.

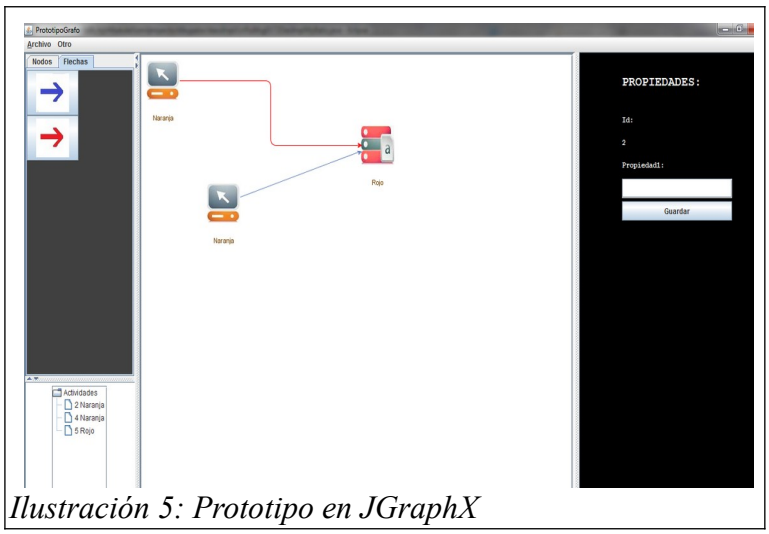


Ilustración 5: Prototipo en JGraphX

Para finalizar la sección se realizó una comparativa de las diferentes bibliotecas teniendo en cuenta varias funcionalidades o características que consideramos importantes para el desarrollo del proyecto.

	JointJS	JGraphX	CytoscapeJS	D3JS	JavaFX	GOJS
Plataforma	Web	Escritorio	Web	Web	Escritorio	Web
Software libre	SI	Si	Si	Si	Si	No
Biblioteca para dibujo Grafos	Si	Si	Si	Si	No	Si
Permite generar grafos interactivamente	Si	Si	Si	Enfocada en visualización	No	Si
Permite interactuar con grafos	Si	Si	Si	Si	No	Si
Diferentes estilos para los enlaces	Si	Si	No se logro implementar	Si	No	Si
Permite asignar imagenes a nodos	Si	Si	Si	Si	No	Si
Serialización y deserialización en archivo JSon	Si	Si	Si	Se debe implementar	No	SI

Tabla 5: Comparativa entre bibliotecas

Como se puede apreciar en la tabla JointJS y JgraphX permiten implementar todo el conjunto de funcionalidades (GOJS también los cumple pero es una biblioteca paga), mientras que las otras no cumplen algunos de los criterios.

2.4. Conclusiones del capítulo

Para poder realizar prototipos de la herramienta que se construyó primero se tuvo que investigar las funcionalidades de la herramienta pre-existente que se quisieran incorporar al prototipo desarrollado para el proyecto.

También se necesitó realizar un estudio de varias tecnologías existentes y con varias de ellas realizar prototipos básicos (que permitieran dibujar un flujo, asignar íconos a los nodos y asignar diferentes formatos de enlaces entre ellos) para poder decidir en un paso posterior el más indicado para utilizar en el proyecto.

3. Análisis del problema

Tomando como referencia el dibujador AdaptorDesigner, se necesitaba implementar una nueva herramienta con tecnologías actuales, para el modelado de aplicaciones y procesos de negocios, y que permita, a partir del dibujo de un grafo, generar líneas de código especializado a persistir en la base de datos que luego utilizará Adaptador.

Para poder definir e implementar la nueva herramienta se necesitó estudiar a fondo el AdaptorDesigner (capitulo 2) y a partir de la ejecución de una serie de ejemplos analizar cómo estos persistían la información en las Base de datos. También se realizaron pruebas de las diferentes funcionalidades para luego poder replicarlas en el nuevo prototipo.

A continuación, se detallan los requerimientos funcionales y los no funcionales del sistema a desarrollar.

También se presentará el alcance del proyecto generado a partir de las reuniones realizadas con los clientes.

3.1. Requerimientos Tecnológicos

El mayor requerimiento tecnológico que se nos planteó fue encontrar una herramienta para el modelado de grafos que sea sencilla y que resuelva el problema antes mencionado. Dentro de las prestaciones que debe proveer están:

- Poder modelar un grafo (nodos y aristas).
- Permitir asignar íconos a los nodos.
- Diferentes formatos (formas y colores) para las aristas.
- Grafos interactivos con el usuario mediante el uso del ratón.
- Persistencia y restauración del grafo.
- Poder seleccionar un nodo para luego editarle las propiedades.

Otro requerimiento tecnológico es posibilitar la persistencia en diferentes instancias de bases de datos, permitiendo al usuario, al comenzar a utilizar el sistema, seleccionar con qué Base de datos trabajará.

También fue necesario que la implementación de nuevas actividades sea lo más simple posible y fácil de extender.

Otra necesitad del cliente, que involucra un aspecto tecnológico, es permitir guardar localmente el grafo generado, en caso de falla de la aplicación o de la base de datos.

El cliente requiere, además, que no se use la Base de datos existente para guardar información referente al dibujo, por lo tanto, se debe buscar la mejor forma de guardar dicha información de otras maneras.

Que se contemple la posibilidad de cambio de tecnología de la base de datos durante el proceso de selección de tecnologías.

Era necesario también que las tecnologías utilizadas fueran con licencia libre, estén activas, bien documentadas y que se puedan depurar.

3.2. Requerimientos Funcionales

Desarrollaremos los requerimientos funcionales utilizando un enfoque *top-down*.

Los describiremos desde una vista global a una más específica, por cada componente de la herramienta a desarrollar.

a) Requerimientos Globales

Se espera que la herramienta funcione independientemente al estado de la conexión de la base de datos (*connection less*). Esto significa que se espera que siga brindando algunas funciones si se pierde conexión con la base de datos.

Darles a los proyectos la importancia que necesitan como agrupadores de modelos. Esta relación actualmente se ve reflejada en la base de datos Designer la cual se desea eliminar.

Que mantenga las principales funcionalidades del AdaptorDesigner explicadas en el capítulo 2.1

- Soluciones integradas
- Exportación en XML u otro formato.
- Ingeniería reversa
- Auto documentación
- Chequeo de errores

Que el producto sea multiplataforma.

Se espera que la forma de dibujar y controlar los grafos sea el punto focal del dibujador. Esto incluye un buen manejo de sus componentes: las actividades, de las transiciones entre las mismas, recorridos del grafo e ingreso de propiedades a las actividades.

Que se puedan almacenar grafos de forma local.

b) Requerimientos Base de Datos

Implementación manteniendo la base MMProdat con la posibilidad de actualización y cambio de tecnológica de la base de datos.

Posibilidad de trabajar cuando se presentan problemas con la conexión.

Guardado de proyectos y grafos en la base de datos.

Manejo de conexiones por servidor.

c) Requerimientos Web

Manejo de la posibilidad de pérdida de conexión y cerrado del navegado durante uso.

Descargar, cargar y exportar los proyectos mediante JSON.

Mantener los componentes visuales correspondientes a la aplicación, barras de herramientas, área de dibujo y explorador de proyecto.

Que no esté asociado a un solo navegador.

3.3. Alcance

Una vez determinado el tipo de tecnología a utilizar durante el desarrollo de la nueva herramienta se presentaron las características y funciones que debe presentar la misma.

El núcleo de la herramienta es la posibilidad de dibujar un grafo con nodos personalizados que permita modificar sus propiedades (actividades), con aristas numerables y que persista los flujos en la base de datos correspondiente, manteniendo la coherencia de la misma. También debe presentar las funcionalidades de ingeniara reversa, chequeo de errores simples y descarga de proyectos.

Se trabajará con un usuario y rol único para simplificar el proceso, debiéndose proyectar una futura actualización al mecanismo de inicio de sesión a la aplicación.

Para el dibujador es necesario que el área de trabajo dedicada a los grafos de los flujos sea importante, debe permitir ser ampliada para almacenar grafos de distintos tamaños considerando que tienden a ser más largos que altos. Se debe poder trabajar con varios flujos en la misma ventana, representando así un proyecto.

Con respecto a las actividades, se deben desarrollar un conjunto representativo que permita generar flujos de pruebas, se espera poder agregar nuevas actividades al producto en un futuro. Las propiedades de las mismas deben ser fácilmente modificables y de simple acceso.

En relación al entorno web, es necesario poder restaurar un grafo en caso de cerrado accidental o problemas de conexión. También debe ser posible descargar y cargar proyectos. El manejo de las conexiones con la base de datos se debe manejar desde el servidor y las operaciones realizadas sobre la misma deben ser atómicas. Esta característica es sumamente importante ya que no se deben permitir modificaciones que posibiliten estados inestables de la base de datos.

La herramienta debe presentar la funcionalidad de ingeniería reversa con el mismo comportamiento al del AdaptorDesigner. El formato para guardar y descargar los grafos debe ser JSON.

Se deberá entregar documentación que permita al cliente comprender las decisiones tomadas y poder continuar el trabajo entregado.

Para concluir, se espera que el producto desarrollado sea una versión simplificada totalmente funcional de la herramienta actual, permitiendo que en un futuro se pueda extender.

4. Dibujador Web

Para poder abarcar todos los requerimientos funcionales y no funcionales de la nueva aplicación se optó por un sistema Web basado en Java y JavaScript. A continuación, se analizará mucho más en detalle las tecnologías que se utilizaron, así como también las decisiones de diseño y arquitectura.

4.1. Prototipos iniciales

Seguidamente se describen las tecnologías utilizadas para los prototipos iniciales y también las razones por las que se optó por una para el prototipo definitivo.

a) Prototipos iniciales gráficos

A pesar de que las tecnologías usadas para implementar los prototipos se encuentran en el capítulo 2 sección 3, creemos relevante volver a incorporarlos a continuación, para poder comprender mejor la elección de la tecnología definitiva.

Antes de continuar con los prototipos creemos importante volver a presentar parte de la tabla en la que se muestra los puntos relevantes de comparación de cada una de las bibliotecas.

	JointJS	**JGraphX**	**CytoscapeJS**	**JavaFX**
Plataforma	Web	Escritorio	Web	Escritorio
Software libre	SI	Si	Si	Si
Biblioteca para dibujo Grafos	Si	Si	Si	No
Permite generar grafos interactivamente	Si	Si	Si	No
Permite interactuar con grafos	Si	Si	Si	No
Diferentes estilos para enlaces	Si	Si	No se logro implementar	No
Permite asignar imagenes a nodos	Si	Si	Si	No
Serialización y deserialización en archivo JSon	Si	Si	Si	No

Tabla 6: Tabla Comparativa entre bibliotecas de los prototipos

I. JointJS

Biblioteca con licencia libre que permite la creación de diferentes tipos de diagramas, como diagramas de flujo HTML5 o BPMN. A su vez permite personalizar los nodos y enlaces y provee las funcionalidades de *drag and drop* o manejo de eventos. Se eligió esta tecnología para el prototipo definitivo pues es la que mejor se adaptó al problema planteado. [3]

II. JavaFX

Biblioteca para aplicaciones de escritorio con la capacidad de crear las pantallas a partir de archivos tipo XML. Fue considerada pues, uno de los requerimientos del problema era la simplicidad para extender nuevas actividades (con sus pantallas para ingreso de propiedades). Se descartó esta herramienta por la dificultad que representa implementar, con dicho lenguaje, todas las funcionalidades de dibujo y manejo de grafos. Igualmente se implementó un prototipo muy básico. [4]

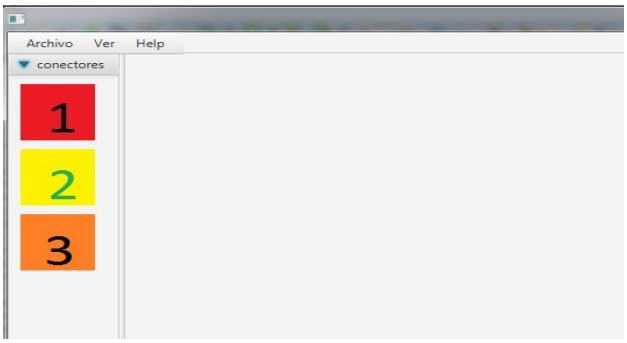


Ilustración 6: pruebas con JavaFX

III. CytoscapeJS

Biblioteca JavaScript para manipulación de grafos, que provee diferentes *plugins* con nuevas funcionalidades, como paneles de propiedades o creación de enlaces mediante *drag and drop*. Se llegó a implementar un prototipo básico, pero se descartó ya que no se llegaron a resolver algunos de los requerimientos del problema (como ser implementar una barra de herramientas desde donde se permita crear nodos simplemente arrastrándolos desde esta barra hasta la pantalla de trabajo, o diferentes estilos para los enlaces) y se encontraron algunos errores en los *plugins* que esta biblioteca provee (el que permite crear enlaces entre nodos no tiene un el funcionamiento esperado)

El prototipo implementado permite crear nodos mediante un panel de botones y enlaces entre estos, utilizando la funcionalidad de arrastrar desde un nodo hacia otro, también permite visualizar los paneles de propiedades de los mismos o eliminarlos ya que cada nodo despliega un menú con varias funcionalidades. [5]

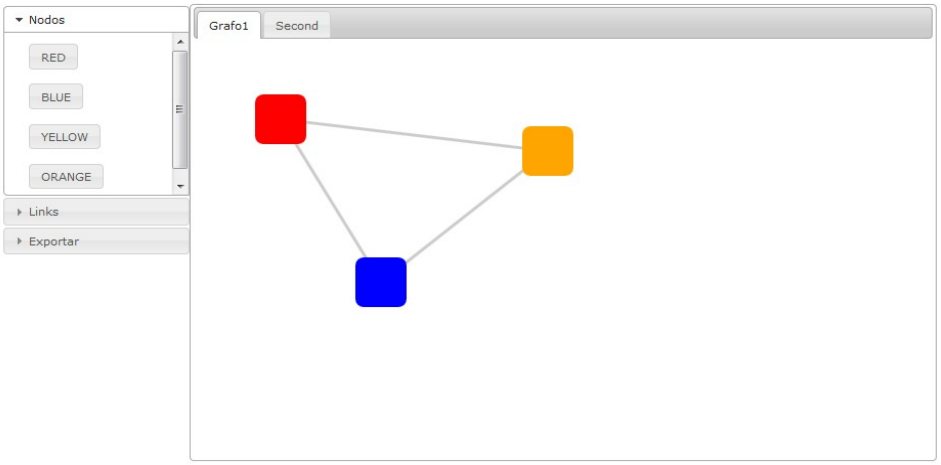


Ilustración 7: Prototipo con Cytoscape JS

IV. JGraphX

Biblioteca Java para implementar la diagramas interactivos y grafos. Es simple de usar, con una rápida curva de aprendizaje. Permite dibujar nodos y conectarlos entre sí mediante flechas, asignándoles estilos a éstos.

Se seleccionó esta tecnología para el prototipo Swing ya que brinda un abanico enorme de funcionalidades y permite implementar todas aquellas que exigía la solución del problema.

Con esta biblioteca se implementó un prototipo que permitía crear nodos y enlaces entre ellos, asignar nombres y panel de propiedades para los nodos, y también guardar y recuperar el flujo en un archivo. [7]

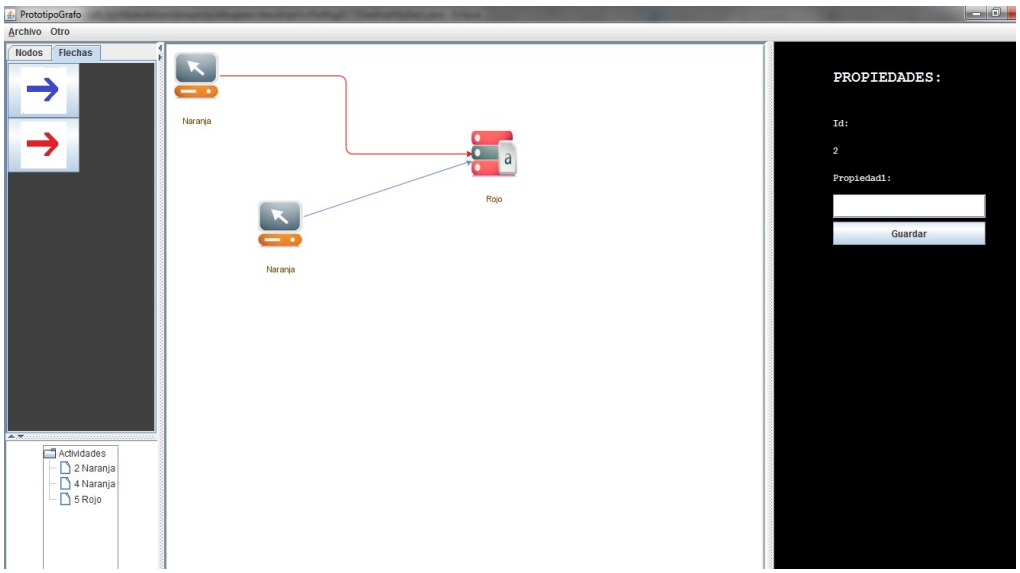


Ilustración 8: Prototipo JGraphX

b) Elección de tecnología gráfica

Para poder decidir cuál sería la tecnología a utilizar y si iba ésta a ser una aplicación web o de escritorio, se intentó resolver un problema sencillo utilizando algunas bibliotecas JavaScript y alguna tecnología Swing para dibujos, luego se eligió la mejor de cada una (JointJS como tecnología web y JGraphX como biblioteca swing) y se les incorporaron nuevas funcionalidades.

Por último, en una reunión con el cliente, se plantearon las ventajas y las desventajas de cada una de estas y el cliente se decidió por implementar una versión Web.

A continuación, se muestra una tabla que resume los beneficios y contras de una implementación web versus una implementación de escritorio. Por una explicación más detallada ver el apéndice 9.2 – Aplicación de escritorio Vs Aplicación web.

Web		**Escritorio**	
Ventajas	Desventajas	Ventajas	Desventajas
Facilidad de acceso	Compatibilidad con navegadores		Instalación
Fácil actualizar	Problemas de conexión con servidor	*Offline*	Más Dependientes del sistema operativo.
Datos centralizados	Menos Seguro	Más seguro	Actualización en todos los dispositivos

Tabla 7: Comparativo entre Aplicación Web y de Escritorio

Dejando de lado que la implementación sea web o swing se pueden apreciar algunas ventajas o desventajas entre los dos prototipos (JointJS vs JGraphX)

JointJS		**JGraphX**	
Ventajas	Desventajas	Ventajas	Desventajas
Más vistoso	Más complejo el manejo con ratón.	Más simple implementar *copy, paste*, halo de propiedades.	Interfaz de usuario más tosca.
Bien documentada			No existe tanta documentación para versión de escritorio.

Tabla 8: Comparativo de tecnología Web y tecnología Swing

Por otro lado, para la decisión de qué tecnología web utilizar, se muestra un cuadro comparando las ventajas y desventajas entre las dos tecnologías web utilizadas para los prototipos.

JointJS JS		CytoscapeJS JS	
Ventajas	Desventajas	Ventajas	Desventajas
Buena Documentación	Varias funcionalidades son pagas (Rappid)	Buena Documentación	Más complejo
Activo en github		Activo en github	Algunos plugins con errores
Simple		Halo de propiedades	
Fácil extender		Creación de Links mediante drag and drop	
Aristas se pueden doblar		Activo en stackoverflow	
Activo en stackoverflow			

Tabla 9: Comparativo entre tecnologías web

Luego de dicho estudio se decidió utilizar la tecnología web JointJs ya que fue la tecnología mejor se adaptaba al problema planteado.

4.2. Decisiones de diseño

A continuación, se presentan las decisiones que se tomaron con respecto a las tecnologías utilizadas.

Servicios: Para la comunicación entre la presentación y la capa lógica se utilizaron servicios RESTful, se tomó esta decisión pues estos son más simples de invocar desde JavaScript (ya que los servicios SOAP requieren la creación de un XML) pues permite utilizar los diversos métodos que proporciona HTTP para comunicarse, como lo son GET (método para obtener información del servidor) y POST (para modificar el estado del servidor), y permite trasmitir datos con formato JSON (esto es muy útil, ya que la biblioteca para dibujar grafos permite exportar los mismos a un archivo JSON) Con respecto a la seguridad (a pesar de que no se implementó en el prototipo, ya que el logueo, roles e usuarios quedaron fuera del alcance del proyecto) se investigó y se encontraron varias formas de implementación tokens, cookies, anotaciones.

Persistencia: Para la persistencia se optó por utilizar myBatis sobre tecnologías como Hibernate o uso directo de JDBC.

Se descartó utilizar directamente JDBC pues este es más difícil de mantener y el desarrollo es mucho más lento. Otra razón de utilizar un *framework* para la persistencia es que así se evitan errores habituales en el manejo de conexiones (como por ejemplo dejar conexiones abiertas)

Por otro lado, Hibernate permite el mapeo de tablas de la Base de Datos a objetos Java, generando automáticamente las sentencias SQL que si bien es muy práctico a la hora de programar se pierde el control del código SQL generado, pudiendo dar en algunos casos problemas de rendimiento.

Otra razón por la que no se optó por Hibernate es que la Base de Datos preexistente (MMProdat) es bastante compleja y contiene muchas tablas y a pesar de que el prototipo implementado utiliza una pequeña cantidad de estas, para continuar con el desarrollo del mismo se debería mapear cada una de las tablas a clases Java.

MyBatis sin embargo es idóneo para utilizar con Bases de Datos ya existentes, ya que éste mapea métodos a sentencias SQL, esto simplifica la programación ya que se elimina casi todo el código JDBC y el establecimiento de conexiones a mano, lo que ahorra tiempo y evita errores como dejar conexiones abiertas. Comparando con Hibernate, permite tener mucho más control del código SQL generado, permitiendo así optimizar las consultas.

MyBatis también permite realizar los mapeos mediante anotaciones, que fue uno de los requerimientos del cliente.

La tarea de realizar rollback y commit, así como manejo de los diferentes pools de conexiones las realiza el contenedor EJB.

Guardado de proyectos: Como se mencionó anteriormente, la agrupación de modelos se encontraba en la tabla Designer, y dado que ésta no se utilizará más, para guardar la estructura de los proyectos se presentó la opción de crear tablas nuevas en MProdat o guardar su estructura en algún tipo de archivo. Se optó por la segunda opción, en parte, por solicitud del cliente de no guardar información del dibujo en dicha base de datos. Para eso se creó una estructura de directorios donde se guardan archivos JSON que representan cada proyecto.

Transaccionalidad al invocar compilar: Cuando se compila un flujo existen 2 opciones de comportamiento:

- 1 - Flujo no existe: En este caso se insertan las líneas correspondientes en la Base de Datos.

- 2 - Flujo ya existe: En este caso se elimina el flujo anterior y se insertan las nuevas líneas. Dado que el servidor de aplicaciones se encarga del manejo de las conexiones, como también de realizar commit y rollback, se tuvo que indicar que el método es transaccional mediante la anotación `@TransactionAttribute(TransactionAttributeType.REQUIRED)`, y propagar la excepción desde el dao a dicho método. A su vez en el catch del método se le indica cómo comportarse mediante `sessionContext.setRollbackOnly()`.

Guardado Local de grafos: Para hacer frente a problemas de conexión y perdida del trabajo realizado, se optó por permitir descargar y cargar el grafo a partir de un archivo JSON y también guardar el JSON del grafo en el localStorage al cerrar el navegador, permitiendo restaurarlo nuevamente.

Extender herramienta: Uno de los requerimientos que el cliente le dio mayor importancia es que la nueva herramienta sea fácil de extender y la creación de nuevas actividades sea simple. Para esto, se estructuro tanto la capa de presentación, como los servicios, capa lógica, algoritmo para seteo de números de orden y siguiente actividad, de la forma más

genérica posible, así al agregar una nueva actividad solo se agrega en algunos lugares y no teniendo que modificar los puntos claves de la herramienta.

JointJS: Se utilizó la biblioteca JointJS para la implementación del dibujador y la interacción con el usuario. Esta herramienta facilita el poder dibujar los grafos en forma interactiva, así como también el manejo de diferentes eventos sobre sus componentes, permite arrastrar los nodos mediante el uso del mouse, y crear aristas entre ellos de forma interactiva.

EJB y JBoss: Se utilizó esta tecnología para poder simplificar y encapsular la lógica de negocios, eligiéndose además pues estábamos ya familiarizados con la tecnología. Otra razón por la que se usó la misma es para que se encargue del manejo transaccional y rollback de la base de datos, y también de manejar los diferentes pools de conexiones. Se utilizaron también los servicios RESTful que esta provee.

4.3. Tecnologías utilizadas

En esta sección se brinda una descripción de las herramientas, lenguajes y tecnologías utilizadas.

JointJS:

Biblioteca JavaScript con licencia *open source* que permite la creación de diferentes tipos de diagramas, como diagramas de flujo HTML5 o BPMN. A su vez permite personalizar los nodos y aristas y provee las funcionalidades de *drag and drop* o manejo de eventos. Esta tecnología fue la que se utilizó pues es la que se adaptó mejor al problema planteado. [3]

MyBatis:

"*MyBatis es un framework de persistencia que soporta SQL, procedimientos almacenados y mapeos avanzados, elimina casi todo el código JDBC, el establecimiento manual de los parámetros y la obtención de resultados, puede configurarse con XML o anotaciones y permite mapear mapas y POJOs (Plain Old Java Objects) con registros de base de datos.*" [8]

JQuery:

JQuery es una biblioteca multiplataforma de Javascript que permite simplificar la interacción con documentos HTML, manipular el árbol DOM, manejo de eventos, animaciones y utilización de Ajax en páginas web. A su vez simplifica la interacción con los diferentes navegadores y manipulación de estilos. [19]

Además, se utilizó JQuery UI que es un conjunto de interfaces, widgets y temas para la interfaz de usuario.

Gson:

Biblioteca que permite la serialización/deserialización de objetos java y su representación en objetos JSON de una forma sencilla.

EJB:

El *framework* EJB provee el ambiente para los componentes EJB, éstos son componentes del lado del servidor que encapsulan lógica de negocio, y se encargan del manejo transaccional y de la seguridad

También poseen un *stack* integrado para el manejo de mensajería, *scheduling*, acceso remoto, servicios web (RESTful / SOAP), inyección de dependencias, ciclo de vida y AOP (*Aspect oriented programming*). Un contenedor EJB provee servicios como ser el manejo transaccional, *pooling*, acceso a recursos, seguridad, soporte de concurrencia, interceptores, invocación asincrónica de métodos (sin usar mensajería)

JBoss:

Servidor de aplicaciones Java EE de código abierto implementado en Java que puede utilizarse en cualquier sistema operativo donde esté disponible una JVM. Tiene una gran comunidad y se actualiza constantemente.

4.4. Arquitectura y diseño de la solución

En esta sección se describe la arquitectura y diseño de la solución implementada.

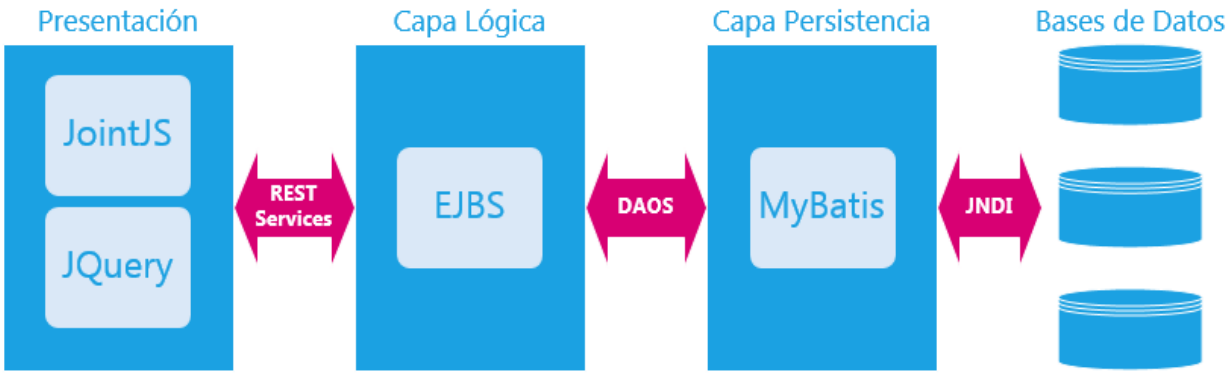


Ilustración 9: Diagrama de capas

Como se puede ver en la imagen se utilizó una arquitectura en capas.

La capa de presentación que es con la que interactúa el usuario y se desarrolló utilizando JavaScript, JQuery y JointJS (este último para el modelado gráfico de los procesos de negocio), ésta se comunica con la capa lógica mediante llamadas a servicios RESTful.

La capa de negocio recibe las peticiones de usuarios, las procesa y para los casos que sea necesario se comunica con la capa de persistencia para obtener o guardar datos en base de datos. Esta capa también contiene todos los algoritmos de recorrida, formateo de datos y asignación de orden de las líneas a persistir (esto es muy importante porque marca el orden de recorrida de los flujos)

La capa de persistencia engloba todos los métodos para guardar y obtener los datos en base de datos, para esto se utilizó MyBatis (mediante anotaciones). En esta capa se encuentran cada una de las sentencias SQL que se mapean a los métodos que son utilizados por los EJB.

Por otro lado, también se persiste información en archivos JSON, los datos de proyecto (nombre de actividades que la componen y nombre y versión del proyecto) se persisten en el servidor, mientras que los datos de un proyecto completo (dibujo e información relacionada a cada actividad) se persiste localmente en la máquina del cliente (Aunque esto último esta implementado en la capa de presentación y no necesita conexión con el servidor para ejecutarse)

En la siguiente sección se especifica en una forma más detallada lo que se resumió en esta sección.

4.5. Implementación de la solución

A continuación, se detalla cómo se implementó la solución planteada diferenciando las diferentes capas de la misma y como se realiza la comunicación entre dichas capas. También se presenta un diagrama de paquetes que especifica las agrupaciones lógicas de sus componentes.

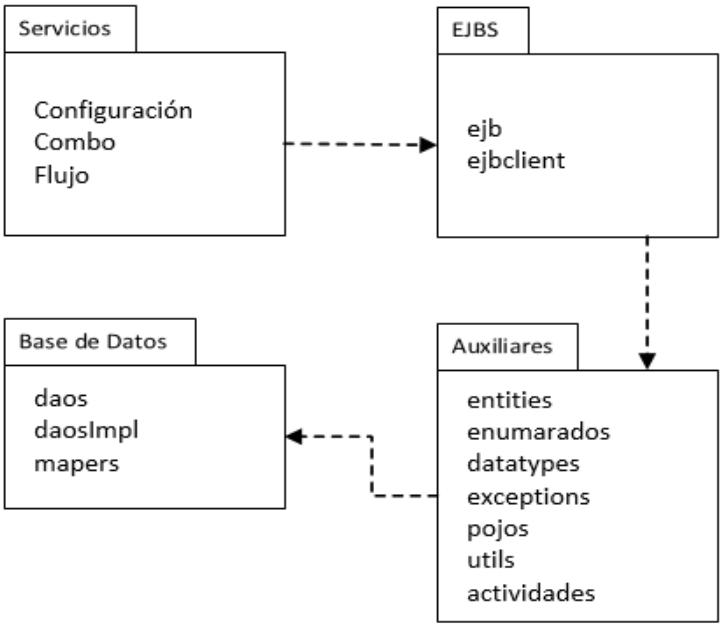


Ilustración 10: Diagrama de paquetes

a) Capa Persistencia

Persistencia en BD:

Se utiliza MyBatis para la persistencia con cada una de las Bases de datos que se encuentran en los *datasourse* de JBoss, para abrir una `session` se debe utilizar una instancia de `sqlSessionFactory` (que se obtiene a partir de `SqlSessionFactoryBuilder`). Para el manejo de cada Base de datos de los *datasource* se crea un `sqlSessionFactory` diferente y se van guardando en un hash. Si un usuario se conecta a un ambiente que no ha sido utilizado anteriormente, se guarda dicho `sqlSessionFactory` en dicho hash. De esta forma se manejan las diferentes conexiones a diferentes bases de datos. Mybatis no realiza el commit ni rollback, esto lo realiza el servidor de aplicaciones al terminar de ejecutarse el método EJB. Para delegar al servidor de aplicaciones estas tareas se debió configurar el *datasource* como JNDI.

"***JNDI*** *– Esta implementación de DataSource está pensada para ser usada en contenedores como Spring o los servidores de aplicaciones JEE en los que es posible configurar un DataSource de forma externa y alojarlo en el contexto JNDI" [6]*

FlujosMapper: Para la persistencia de todo lo relacionado con los flujos (`insert`,

`update`, `select` y `delete`), cada flujo se identifica por su nombre y versión.

CombosMapper: Para obtener los datos que se cargarán en los combos de la presentación.

ConfigMaper: Para obtener la información de usuarios, roles.

Persistencia en JSON:

1- Persistencia de proyectos localmente:

Si bien la persistencia de proyectos en forma local no se realiza en la capa de persistencia, sino que se implementa en la capa de presentación, utilizando JavaScript, se menciona en esta sección pues es otra forma de cómo se guardan los proyectos.

Uno de los requerimientos del cliente fue poder guardar el trabajo, aunque se perdiera conexión con el servidor, por ello se permite la persistencia del dibujo realizado junto con las propiedades ingresadas para cada actividad utilizada, en un archivo JSON, que el usuario del sistema podrá guardar en su disco local para luego poder restaurar su avance y así continuar trabajando.

Este archivo JSON contiene toda la información del dibujo junto con toda la información que se ingresó para cada uno de los flujos que conforman el proyecto.

2- Persistencia de estructura de proyectos:

Los nombres de los proyectos y los flujos que los componen se persistían en la base de datos Designer, pero como esta se dejó de utilizar en el nuevo prototipo, se tuvo que buscar una alternativa para guardar los nombres de los proyectos y los flujos que conforman cada uno de ellos. Para persistir la estructura de proyectos se utilizaron Archivos JSON dentro de una estructura de directorios, se crea un directorio por cada ambiente en el que se trabaje en el servidor, y los proyectos que pertenecen a dicho ambiente se persisten dentro del mismo.

El nombre del archivo JSON tiene el siguiente formato: `NombreProyecto_usuario_vs (nombre del proyecto, usuario que lo crea, versión del proyecto)` y este contiene la lista de flujos que conforman un proyecto:

```
{"nombre":"Proyectito","usuario":"danivare","vs":"1.0","ambiente":
"development","nombreFlujos":[{"nombre":"Grafo1","vs":"1.0"},
{"nombre":"grafo2","vs":"1.1"}]}
```

La estructura del JSON es la siguiente:

- nombre
- usuario
- vs
- ambiente
- nombreFlujos
  - nombre
  - vs

b) Servicios

Para la comunicación entre la presentación y la lógica se utilizaron servicios RESTful.

- Servicios manejo Flujos: Para compilar, obtener nombres de proyectos y obtener Flujos mediante la ingeniería reversa.

- Servicios Combos: Para la obtención de los datos de los combos que se muestran en la web, se creó un solo servicio que devuelve siempre una lista de String (con formato JSON) dependiendo de los parámetros de entrada.

- Servicios Configuración: Se utilizan para implementar el inicio de sesión, obtención de Ambientes, obtención de roles.

c) Presentación

Para simplificar la tarea de mantenimiento y actualización se separaron las diferentes funcionalidades en varios archivos JavaScript y se parametrizaron las distintas funciones para obtener los valores desde el archivo iconos.js.

A continuación, se presentarán los archivos JavaScript principales:

JavaScript Para manejo de formularios: Archivos JavaScript auxiliares para el manejo de componentes gráficos de diferentes actividades

DivPropiedades: Agrupa e importa los JavaScript de las diferentes paginas HTLM.

DivPropiedadesActividad: Para cada actividad existe un archivo cuyo nombre tiene el formato “DivPropiedadesNombreActividad”, estos contienen todos los métodos JavaScript para cada una de las páginas de propiedades de cada actividad.

TablasDivPropiedadesActDef: Define los métodos para manejo de la tabla de la actividad ActDef (agregar, editar y quitar elementos)

JavaScript para definir Actividades

Iconos: Contiene la información de los nodos de las actividades (id, nombre, ruta al ícono, id del div de propiedades que le corresponde, coordenadas en la paleta de elementos, ids de input de propiedades).

EntradasDiv: Se definen los campos (con su tipo de dato) que existen en cada panel de propiedades de cada una de las actividades.

JqueryComponentesLoadDiv: Se definen todos los paneles de propiedades existentes, para así no tener que definirlos en el archivo index.html.

Propiedades: parámetros y variables globales a todo el proyecto.

JavaScript de funcionalidades del sistema

Prototipov4: Definición del grafo e implementación del papel extensible.

Panelesv4: Implementa la visualización de los divs de propiedades dependiendo de en cual nodo se *clickea*.

SaveLoad: Están implementadas las funcionalidades de guardar/Abrir el grafo en un archivo (No es necesario modificarlo para el agregado de actividades), también guardar y restaurar del localStorage y la función para crear el archivo JSON que utilizan los servicios al compilar. Utiliza las funciones definidas en SaveLoadGenericFunction

SaveLoadGenericFunction: Función genérica que mapea el grafo a archivo JSON y función que a partir del archivo JSON dibuja el grafo en pantalla.

DecoratedRect: Se modela la estructura del nodo de paleta de elementos.

Actor: Se modela la estructura del nodo en el grafo luego de ser arrastrado desde la paleta de elementos.

StencilIconv4: Se implementa la funcionalidad de arrastrar el nodo desde paleta de elementos al papel.

ArbolData: Se especifica el código referente al árbol de contenido.

ManejoDialogProyectos: Implementa las diferentes ventanas para ingresar datos de los proyectos o modelos, Cambien para la selección de proyectos para ingeniería reversa.

Modelos: Funciones que implementan la funcionalidad de agregar una nueva pestaña al proyecto.

Modelos2: Para especificar los datos a ingresar al agregar un nuevo modelo al proyecto.

Services: Especifica las llamadas a los servicios RESTful

servicesLogin: Especifica las llamadas para el servicio de inicio de sesión.

KeyDetect: Funcionalidad para detectar la tecla presionada, se utiliza para poder eliminar nodos del dibujo.

CreateModelosLoadFile: Funciones para poder agregar las diferentes pestañas cuando se carga un proyecto desde un archivo.

Archivos HTML:

En la carpeta PropiedadesDiv se encuentran todos los paneles de propiedades para las diferentes actividades creadas y también para los enlaces.

d) Capa Lógica

La capa lógica se estructura de la siguiente forma:

com.proyecto.dibujador.actividades: Es donde se define cada una de las actividades

implementadas. La clase `ActividadGenerica` que contiene los atributos y métodos comunes de todas las actividades y es padre de cada una de las actividades concretas. En este paquete también se encuentra la clase `ActividadModeloGenerico` que sirve de planilla para crear nuevas actividades.

En cada una de las clases de actividades concretas se implementan los métodos para formatear los datos y obtener los mismos a partir de una o varias líneas de la Base de datos (representada como la entidad `LnFluMsg0172`). Detalles de que datos y como se guardan en la base de datos se pueden ver en el apéndice 9.3 – Persistencia de flujos.

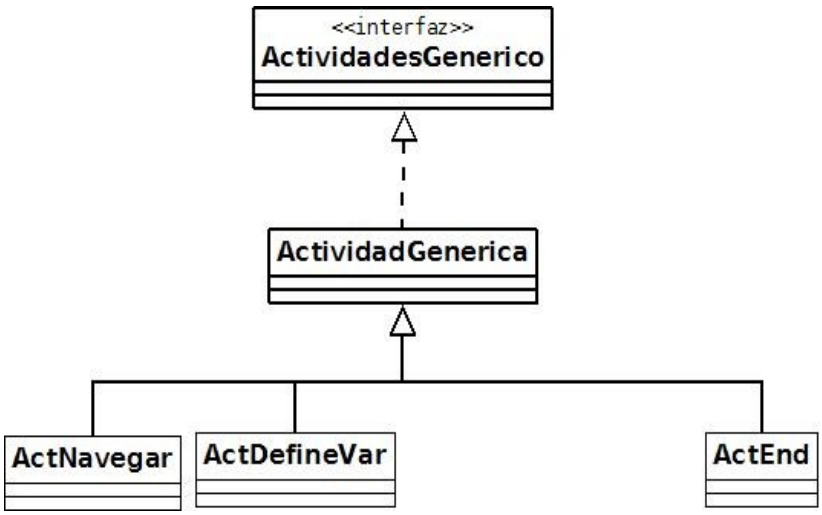


Ilustración 11: Actividades

com.proyecto.dibujador.daos: Interfaces de los daos.

com.proyecto.dibujador.daosImpl: Implementación de las interfaces de los daos utilizando MyBatis. Estos se dividen en 3 tipos:

- ConfigDaoImplMyBatis: Daos para la obtención de los datos de configuración (ambiente, usuarios, roles).
- CombosDaoImplMyBatis: Daos para la obtención de los diferentes datos de los combos presentes en la presentación.
- LnFluMsg0172DaoImplMyBatis: Daos para la obtención y guardado de flujos en la tabla LnFluMsg0172.

com.proyecto.dibujador.datatypes: *Datatypes* utilizados para la serializacion/ deserialización de los JSON y también los utilizados como respuesta de los servicios.

com.proyecto.dibujador.ejb: EJB que implementan los métodos que consumen los servicios RESTful. Se estructuran en 3 tipos:

- ConexionesEJBBean: Para el manejo de usuarios, roles, ambiente y inicio de sesión.
- FlujosEJBBean: Engloban los métodos para el manejo de flujos (parseo de JSON, compilar, obtener y guardar flujos de la base de datos)
- ProyectoEJBBean: Contiene los métodos para manejo de JSON con la estructura de los proyectos.

com.proyecto.dibujador.ejbclient: Interfaces para los EJB Remotos.

com.proyecto.dibujador.entities: Entidades que representan objetos a ser persistidos, tanto en base de datos como en objetos JSON.

com.proyecto.dibujador.enumerados: Enumerados que se utilizan en las diferentes actividades.

com.proyecto.dibujador.excepciones: Excepciones personalizadas utilizadas para formatear los mensajes de error.

com.proyecto.dibujador.mapers: Mapeos utilizados por MyBatis para la persistencia en las Bases de datos, implementados mediante el uso de anotaciones. Se dividen en 3 tipos:

- FlujosMaper: Para manejo (`select, insert, delete`) de datos en la tabla `LnFluMsg0172`
- CombosMaper: Para obtención de los diferentes combos a partir de sus tablas correspondientes.
- ConfigMaper: Para obtención de roles, ambientes y usuarios.

com.proyecto.dibujador.utils: Contiene los métodos utilitarios utilizados en los EJB para los algoritmos de manejo de los flujos, seteo de orden, siguientes flujos. A su vez contiene el archivo *propierties* donde se guarda la ruta donde se guarda la estructura de los diferentes proyectos.

MyBatis: Contiene archivos de configuración, clases para el manejo de sesiones, manejo de transacciones.

4.6. Casos de estudio y Testeo

Se cuenta con 5 proyectos de prueba otorgados por el cliente para realizar el estudio de las funcionalidades, elementos y comportamiento esperado del dibujador. Los mismos cuentan con un conjunto de actividades representativas, ejemplos de los flujos utilizados y su representación en la base de datos.

Como primer proyecto de pruebas tenemos un test extremadamente simple utilizando un timeout, el mismo se llama “Ejemplo1” y se encuentra en la versión 1.0.

El “Ejemplo2” en la versión 1.0 representa una ejecución básica de un procedimiento almacenado en una base de datos.

En la siguiente prueba “Ejemplo3” versión 1.0 se realizan distintas operaciones con correos electrónicos correspondientes a la salida de la ejecución de un procedimiento almacenado.

El ejemplo “TC003EjSp” versión 1.0 presenta la ejecución de una consulta SQL y da devolución de los valores obtenidos en la misma.

“EjemploVarios” versión 1.0 al contrario de los anteriores, nos presenta múltiples flujos (cinco) con distintas actividades. Las cuales incluyen navegar en un XML, trabajar con un XML simple, iterar un XML a mano (bucle), parsear un *string* desde archivo y ejecutar procedimientos almacenados.

De los casos presentados se selecciona el proyecto EjemploVarios 1.0 como caso de uso ya que es el único que presenta varios flujos e incluye a los otros casos de estudio.

En la siguiente sección describiremos las características del mismo y lo presentaremos como el caso de uso principal de nuestro proyecto.

a) Estudio del proyecto EjemploVarios versión 1.0

En este proyecto nos encontraremos con 5 modelos de flujo, uno de implementación y otro de flujo como los componentes del mismo.

Las actividades y transiciones elegidas para desarrollar son las pertenecientes a los modelos de flujo que componen al mismo. Las mismas serán utilizadas como modelo para el futuro desarrollo de la herramienta.

Se cuenta con el proyecto en la herramienta actual, es decir la representación gráfica del proyecto, su información almacenada en la base de datos y la posibilidad de editarlo para observar las modificaciones del mismo.

Los modelos de flujo son los siguientes:

- EjemploXML13 - El cual navega en un XML.
- EjemploXML1 - Trabajar en un simple XML.

- EjemploIteraXML - Itera un XML a mano mediante bucle (sin la actividad *foreach*)
- EjemploEjecutaSP - Ejemplo que ejecuta procedimientos almacenados.
- EjemploParser - El cual parsea un *string* desde archivo

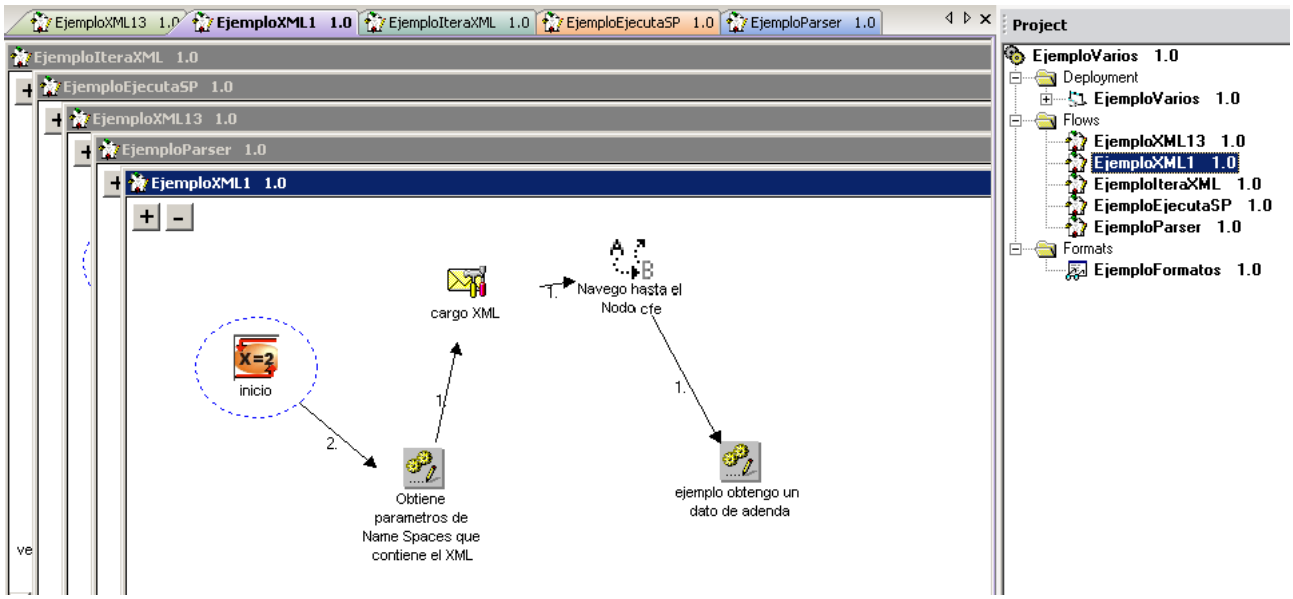


Ilustración 12: Proyecto EjemploVarios 1.0 en AdaptorDesigner v7.1.1

Todos en su versión 1.0.

El modelo de implementación tiene como nombre EjemploVarios 1.0 y el de formatos EjemploFormatos 1.0.

Se adjunta el SQL utilizado para insertar el proyecto en la base de datos MMProdat y Designer.

b) Actividades y transiciones presentes

Listaremos las actividades implementadas para cada uno de los modelos de flujo, indicando en cada caso en cual flujo están presentes.

1. `Define Variable`

Esta actividad está presente en todos los flujos implementados por el proyecto.

2. `End`

Presente en todos los flujos a excepción de EjemploXML1 y EjemploEjecutaSP.

3. `Parse/Build`

Presente en todos los flujos a excepción de EjemploEjecutaSP.

4. `Browse`
5. `Expression Evaluator`

Incluida en EjemploXML13, EjemploXML1 y EjemploIteraXML.

6. `Values`

En EjemploXML13 y EjemploIteraXML.

7. XPATH Query
8. Iterator For Each Flow

Presentes en EjemploXML13.

9. SQL
10. DSN

En EjemploEjecutaSP.

11. Managing Files

Incluido en EjemploParser.

En la sección Actividades se describen la funcionalidad de cada una de ellas y en Persistencia se especifican como dichas actividades son guardadas en la base de datos.

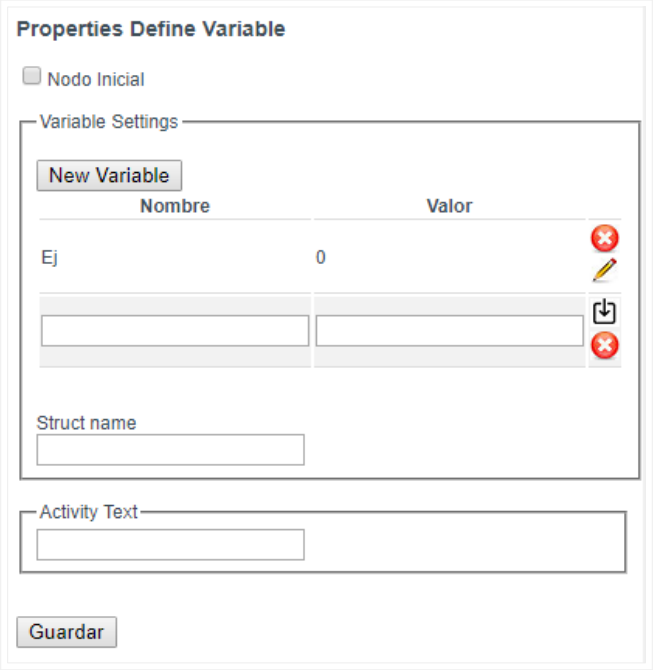
En las transiciones se observa la utilización de Transition y Uso. Implementando a su vez la transición de Error utilizada en los otros casos de estudio. Al igual que las actividades, las mismas están detalladas en la sección Transiciones y Persistencia.

Los datos de prueba se prestarán en el Apéndice 9.4 – Datos de Prueba.

4.7. Implementación de actividades

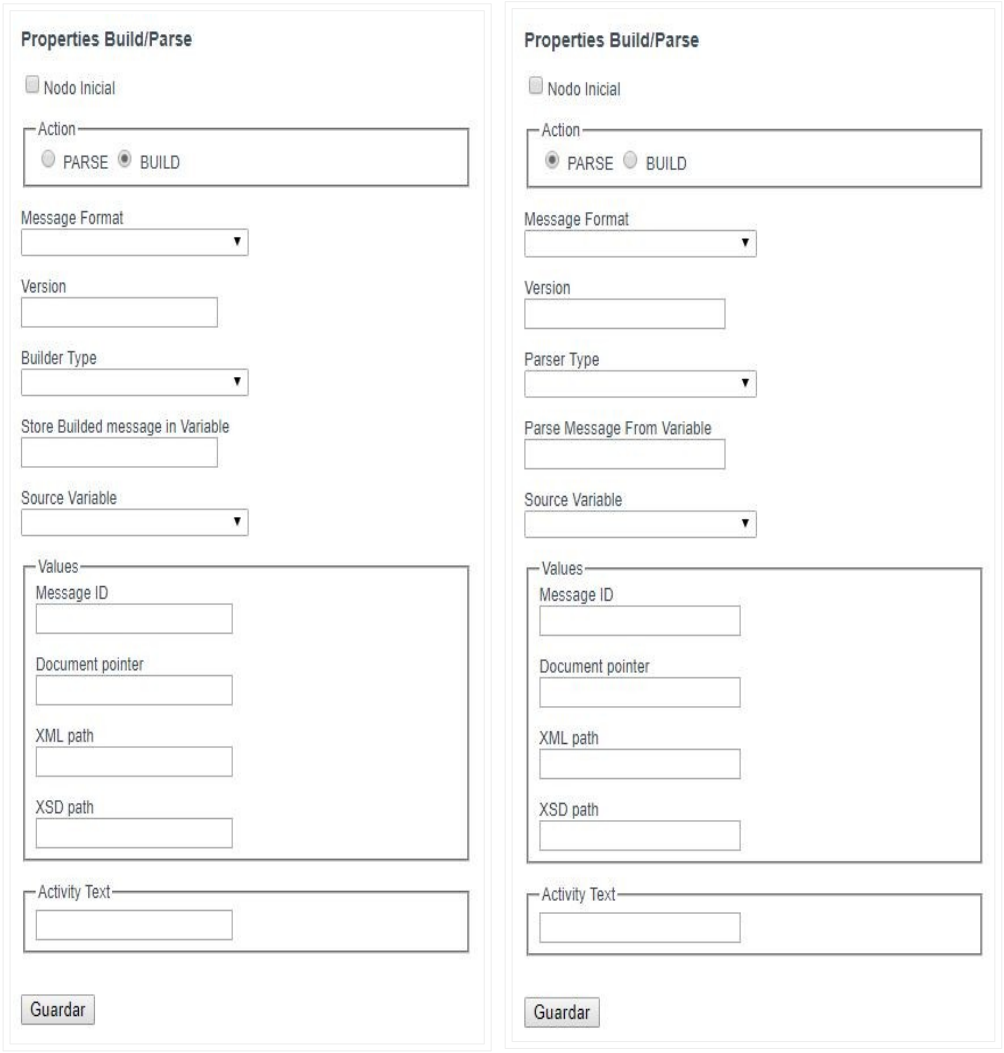
A continuación, se listarán la definición de las actividades en el actual dibujador. El formato y los datos guardados en la base de datos se especifican en la sección Persistencia.

a) Actividades implementadas

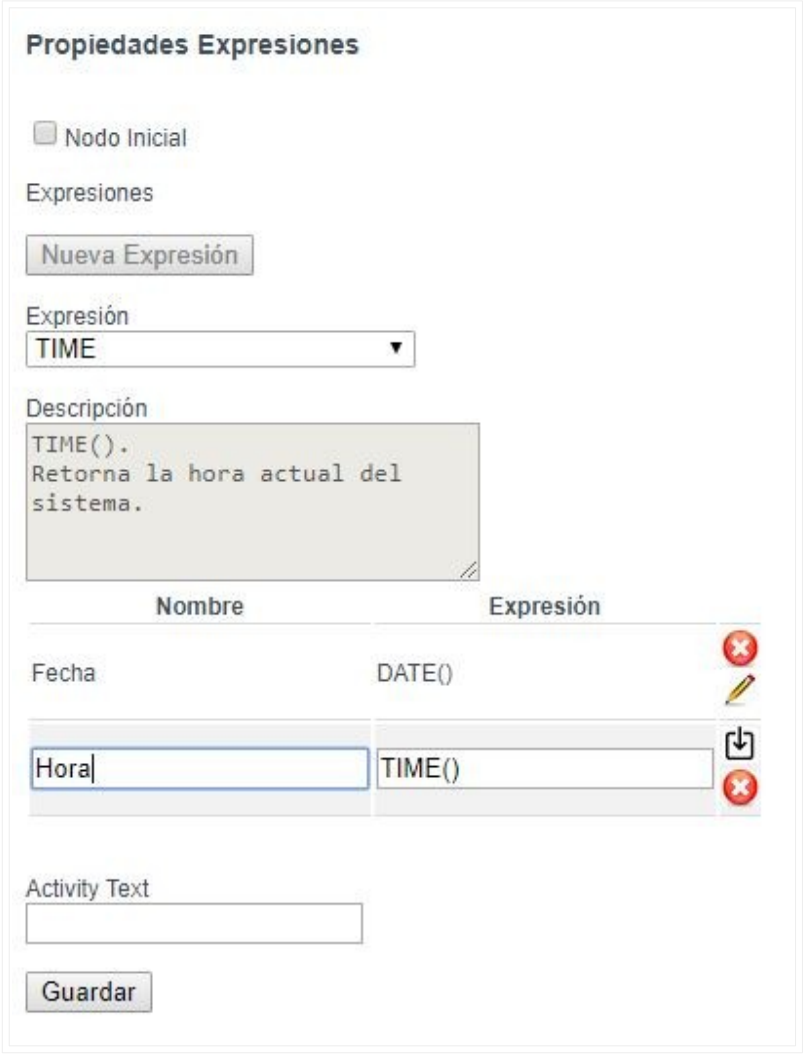
Nombre	`Define Variable`
Descripción	Esta actividad permite agregar/modificar variables, que luego serán utilizadas a lo largo del Proceso.
Icono	
Ventana	
Propiedades	New Variable Name (Text Box) Value (Text Box) Defined Variables (List Box) - Carga los valores Name:Value a la lista de variables definidas. - Carga los valores en los campos Name y Value para poder modificarlos. -Elimina la variable correspondiente. Struct name (Text Box) Activity Text (Text Box)

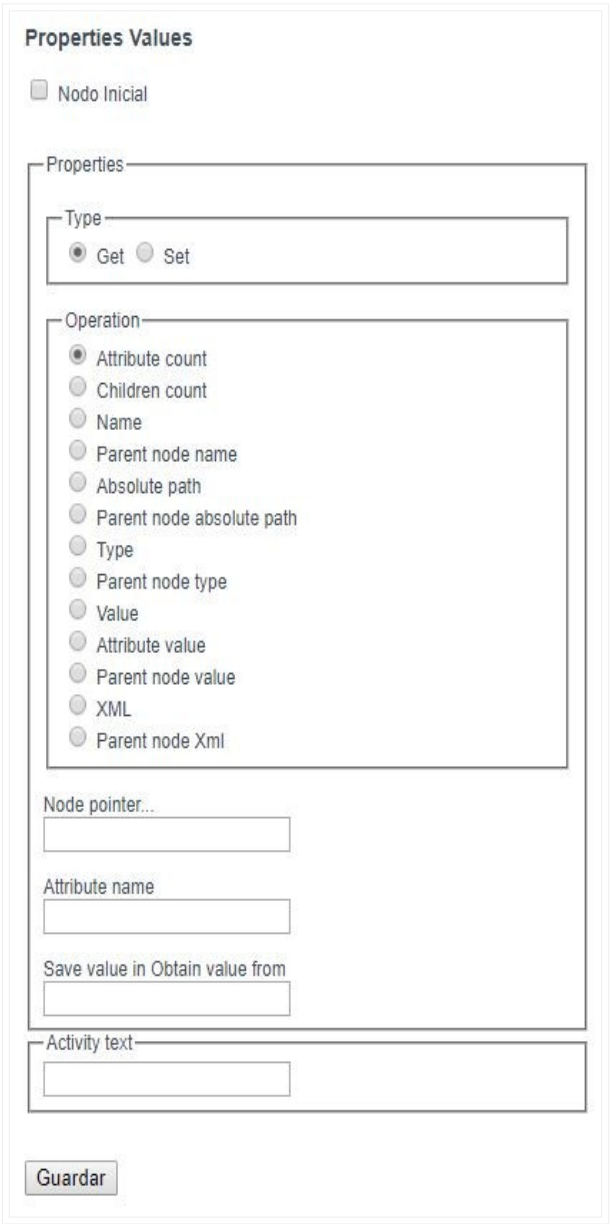
Nombre	`End Flow`
Descripción	Esta Actividad finaliza un flujo de actividades (un Procedimiento) con estado ***"OK"***.
Icono	
Ventana	
Propiedades	Activity Text (Text Box)

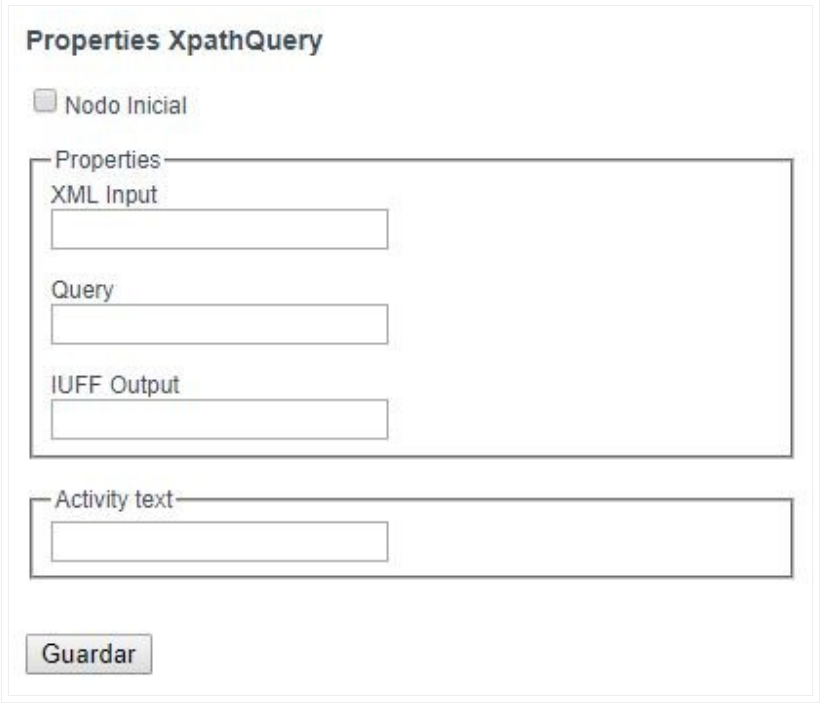
Nombre	`Parse/Build`
Descripción	Parsear o analizar un mensaje significa crear una lista de variables a partir de una cadena de caracteres de entrada (un mensaje), según un formato previamente especificado. Realizar un Build o construir un mensaje, significa generar una cadena de caracteres (un mensaje) a partir de una lista de variables, según un formato previamente especificado. Esta Actividad, permite analizar o construir un mensaje de acuerdo a un formato previamente creado y especificado en la configuración de dicha actividad.
Icono	

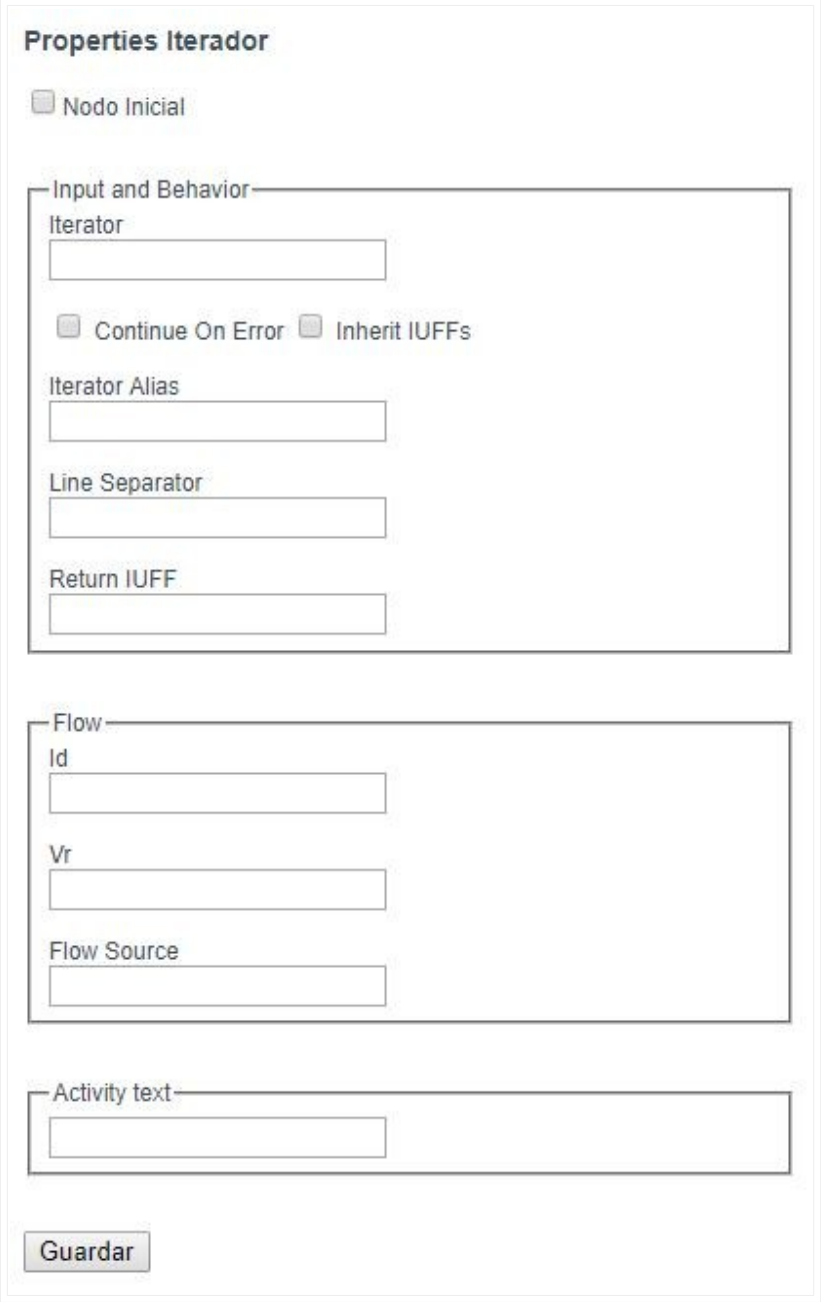
Ventana	
Propiedades	BUILD/PARSE (Check Box) – Los campos se adaptan según la selección. Message Format (Combo Box) Version (Text Box) – Versión del formato seleccionado. Builder/Parser Type (Combo Box) – Pre cargado con los tipos correspondientes. Source Variable (Combo Box) StoreBuided message in Variable (Text Box) Parse Message From Variable (Text Box) Document pointer (Text Box) XML path (Text Box) XSD path (Text Box) - Los anteriores cuatro Text Box se activan y desactivan según el tipo seleccionando. Activity Text (Text Box)

Nombre	`Browse`
Descripción	Actividad para navegación de un XML
Icono	
Ventana	
Propiedades	Parent/Child/Next/Previous/Jump to (Radio Button) Node pointer... (Text Box) Save pointer in... (Text Box) Xpath (Text Box) Activity Text (Text Box)

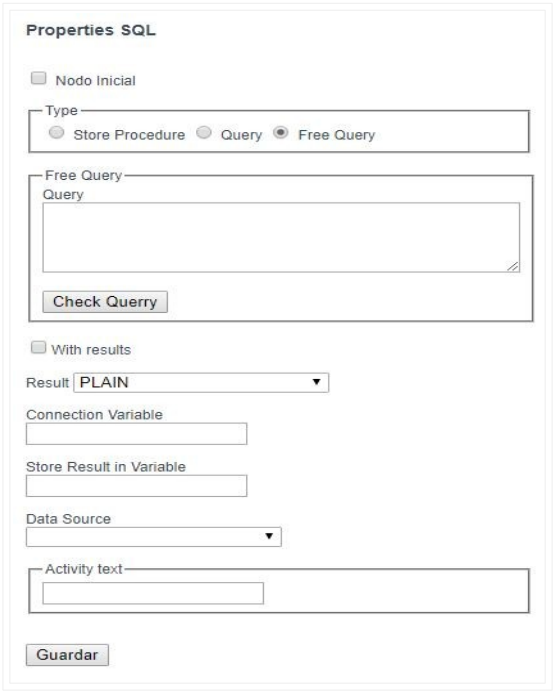
Nombre	`Expression Evaluator`
Descripción	Process Designer, incluye una lista de expresiones que podrán ser utilizadas en cualquier momento del proceso, por ejemplo, obtener campos de una cadena de caracteres, convertir un valor de tipo String en numérico, obtener la fecha y hora actual del sistema, etc. La Actividad “Expresión” asiste en la escritura de dichas expresiones, almacenando el resultado de las mismas en la variable especificada en nombre.
Icono	
Ventana	
Propiedades	Expression (Text Box) – Permite escribir la expresión Nueva Expresión (Button) – Agrega una nueva expresión a la lista. Expresión (Combo Box) – Agrega la plantilla de la expresión a Expresión Descripcion (Text Box – Solo lectura) Activity Text (Text Box)

Nombre	`Values`
Descripción	Esta actividad permite obtener o especificar información y propiedades de un Xml.
Icono	
Ventana	
Propiedades	Get/Set (Radio Button) Operation (Radio Button) Node pointer (Text Box) - Se especifica el nombre de iuff que contiene puntero al nodo. Attribute name (Text Box) Save value in (Text Box) - Se especifica iuff en el cual se guardará el resultado. Activity Text (Text Box)

Nombre	`Xpath Query`
Descripción	Esta actividad tiene como objetivo aplicar una consulta XPATH sobre un documento XML.
Icono	
Ventana	
Propiedades	XML Input (Text Box) - XML en forma plana contenido en un IUFF, o especificado explícitamente. Query (Text Box) - Es la consulta XPATH a ejecutar, puede ser especificado explícitamente o estar contenido en un IUFF. IUFF Output (Text Box) - Se especifica IUFF donde se almacenará el resultado de la actividad. El resultado es un Iterador. Descripcion (Text Box)

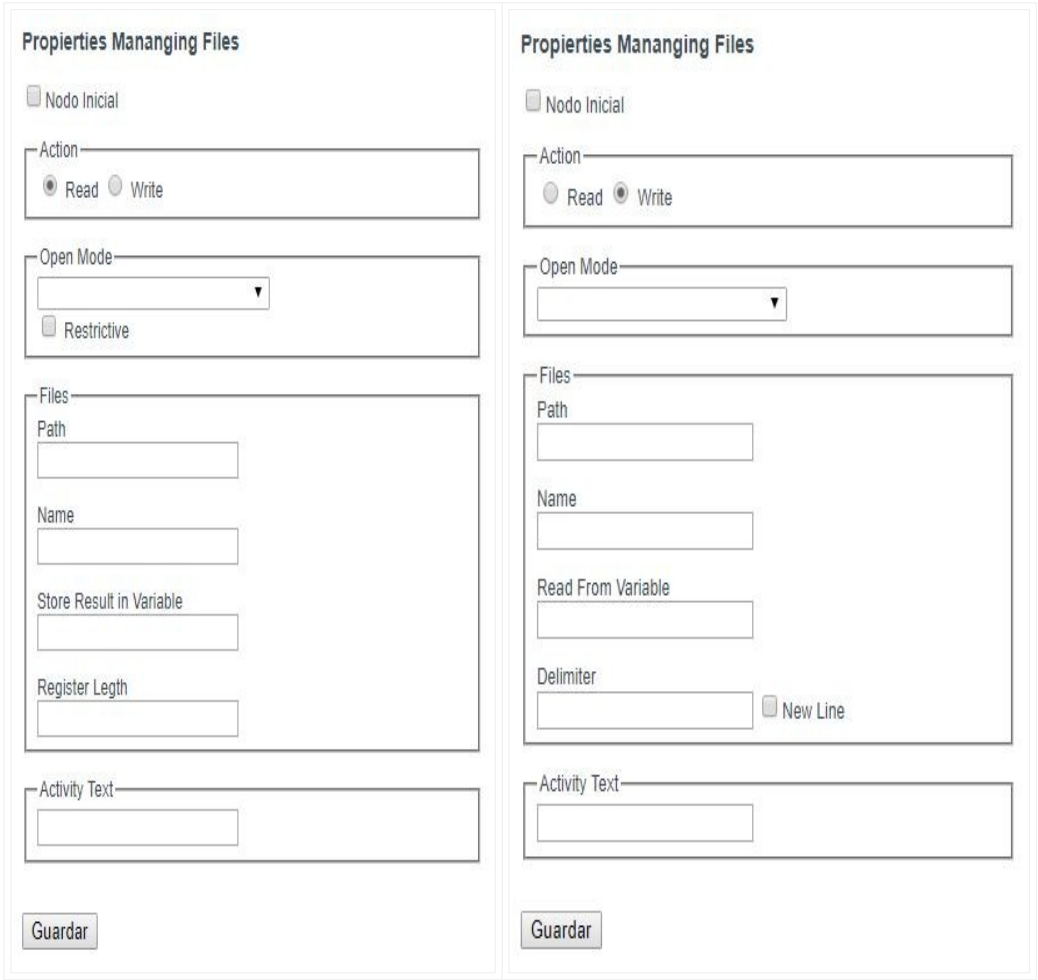
Nombre	`Iterator For Each Flow`
Descripción	Actividad que realiza una iteración sobre hijos de un nodo específico. Utiliza la expresión ITRForEachFlow(iterador, idflow, vrflow, source, continueOnError)
Icono	
Ventana	
Propiedades	Iterator (Text Box) -Es el nombre de un puntero iterador (puede ser creado previamente con la actividad XPATH) Iterator Alias (Text Box) - El nombre del puntero iterador disponible en cada iteración para realizar operaciones utilizando el XML.

	Return IUFF (Text Box) - Contiene el resultado de la ejecución, simplemente si fue exitosa o no (int). Line Separator (Text Box) Continue On Error (Check Box) - En caso de estar seleccionado continua la iteración aunque alguna de ellas haya dado error. Inherit iuffs (Check Box) - En caso de estar seleccionada esta opción los iuff creados en el flujo principal estarán disponibles en el flujo utilizado para cada iteración. Id (Text Box) - Nombre de flujo que se ejecutará en cada iteración. Vr (Text Box) - Versión de flujo que se ejecutará en cada iteración (debe estar especificado entre comillas dobles pues el punto es separador). Source (Text Box) - entidad externa definida para el flujo que se ejecutará en cada iteración. Activity Text (Text Box)

Nombre	SQL - Parcial
Descripción	Esta Actividad permite interactuar con múltiples manejadores de base de datos, permitiendo tanto la ejecución de Stored Procedures (si el origen de datos lo permite), como sentencias SQL. Esta Actividad debe estar vinculada al origen ODBC correspondiente, mediante el acceso directo del DSN creado en la hoja de recursos. El origen de datos se puede definir en una variable de conexión, será un IUFF que previamente será cargado campo " Variable de Conexión" o utilizando la actividad DSN,(este campo también puede contener el nombre exacto del DSN vinculando de esta forma las dos actividades).
Icono	
Ventana	

Propiedades	Store Procedure/Query/Free Query (Radio Button) Free Query (Text Box) With results (Check Box) Result (Combo Box) Connection variable (Text Box) Store Result in Variable (Text Box) Activity Text (Text Box)
NOTA	Dada la complejidad de la actividad se desarrollará solo el tipo Free Query.

Nombre	DNS
Descripción	Esta Actividad permite especificar todo lo relativo a un origen de datos para indicar con qué se tiene que conectar y cómo (cadena de conexión, usuario, etc), para tener acceso a datos de otras bases de datos, aplicaciones, canales. Los DSN serán usados por otras actividades, quienes serán generadoras de la entrega de los datos, proveyéndoles a las mismas de todos los datos necesarios para acceder al canal o entidad. Algunas de las actividades que harán uso de un DSN son: SQL, FTP, Mail.
NOTA	Esta actividad no se implementará. Se incorporara un combo box en cada actividad que lo requiera con los DNS de la base de datos.

Nombre	`Managing Files`
Descripción	Esta Actividad permite leer o escribir un archivo en formato texto (archivo plano).
Icono	
Ventanas	
Propiedades	Action (Radio Button) – Selecciona la acción a realizar. Open File (Combo Box) – Forma de abrir el archivo Restrictive (Check box) Path (Text Box) – Ruta del archivo Name (Text Box) – Nombre del archivo Store Result in/Read From Variable (Text Box) Register Length (Text Box) Delimiter (Text Box) – No colocar si no esta New Line activo New Line (Check box) Activity Text(Text Box)

b) Transiciones

En esta sección detallaremos los tres tipos de transiciones implementadas en la herramienta.

Nombre	`Transition` (Plugins.Evento)
Descripción	Su propósito es indicar la secuencia en que las distintas actividades se ejecutan a lo largo del flujo. Estas transiciones podrán estar condicionadas lógicamente, llevándose a cabo si la expresión lógica indicada evalúa verdadero.
Icono	
Propiedades	Condition (Text Box) Index (Text Box)

Nombre	`Uso` (Plugins.Uso)
Descripción	Su propósito es generar dependencias de uso, al relacionar dos actividades mediante una relación de uso, la actividad de origen puede hacer uso de ciertos datos de la actividad destino. Es muy utilizada para indicar que cierta Actividad hace “uso” de ciertos Recursos.
Icono	
Propiedades	Name (Text Box) Index (Text Box – No editable)

Nombre	`Error Handling` (Plugins.Error)
Descripción	Su propósito es indicar un curso alternativo en caso de que la actividad falle en su ejecución.
Icono	
Propiedades	Index (Text Box – No editable)

c) Agregar una nueva actividad

En esta sección se presentará como se debe agregar una nueva actividad al producto desarrollado, este procedimiento fue seguido durante la codificación. Esto permitió una simplificando y la generación de una guía la cual será entregada a los clientes para su utilización.

Para agregar una nueva actividad debemos generar código tanto en la lógica del producto como en la presentación del mismo. Se describirá donde se debe realizar los cambios requeridos de forma reducida para luego presentar el código necesario que se debe implementar.

1 Pasos a seguir

En el caso de la lógica se trabajará en `com.proyecto.dibujador.actividades.` Se seguirán los siguientes pasos para crear la actividad.

1. Se debe generar una clase para la actividad deseada que extienda `ActividadGenerica` con sus métodos. Los mismos se presentarán en la siguiente sección.

2. Una vez creada la clase debe agregarse en los métodos de `GeneradorActividades`.

Una vez generada la actividad en la lógica se deberá desarrollar sus componentes gráficos. Para ello se debe:

1. Generar el archivo HTML del formulario, así como los archivos de JavaScript necesarios para la utilización en los mismos.

2. En el archivo `Iconos.js` debajo de `JSPropiedadesActividades` se debe agregar una línea que indicara el icono a utilizar por la actividad, así como si posición en la barra de actividades. El icono será representado por un archivo de imagen de 30x30 pixeles.

3. Se debe incorporar el `div` generado en el paso 1 al archivo `index.html`.

4. Luego se generará una línea en el archivo `JqueryComponentesLoadDiv.js` debajo de `JSPropiedadesActividades.`

5. Para finalizar se debe agregar los atributos en `EntradasDiv.js` debajo de `JSPropiedadesActividades.`

A continuación, se desarrollarán de forma extensiva los pasos anteriormente comentados.

LÓGICA (ProyectoBL)

Cada actividad se representa mediante una clase java que extiende de la clase `ActividadGenerica`:

```
public class NombreActividad extends ActividadGenerica{
```

`ActividadGenerica` contiene varios de los atributos y métodos comunes para todas las actividades, y resuelve varias tareas, como la asignación de orden y actividades a las que se conectan.

`ActividadGenerica` contiene los siguientes atributos:

- `private String id;  //IdLine0172 en LnFluMsg0172`
- `private String orden;  //Ordina0172 en LnFluMsg0172`
- `private String vs;  //F1Vers01650172 en LnFluMsg0172`
- `private String nameFlow;  //F1Mens01650172 en LnFluMsg0172`
- `private String nameProy;  //F1EnEx11800172 en LnFluMsg0172`
- `private boolean inicial; // Si es una actividad inicial en un flujo`
- `private List<LinkActividad> siguientes; //Lista de Link a otras actividades`

La lista de Links se representa mediante `LinkActividad` y contiene

- `condición`: Condición en la web
- `SiguienteActividad`: Orden de actividad conectada
- `indexLink`: Índice ingresado en la web

En cada actividad que se crea nueva se deben implementar obligatoriamente los siguientes métodos

1. **`cantLineas()`**: Este método devuelve la cantidad de líneas que ingresa la actividad en la tabla LnFluMsg0172.
2. **`generarLineasFromActivity()`**: Genera las líneas que se guardarán en la base de datos. Tener especial cuidado en el caso que se genere más de una línea.
3. **`generateDataActivity(List<LnFluMsg0172>)`**: Se genera la actividad a

partir de las líneas de la base de datos.

4. **`getImageJson()`**: Devuelve la ruta al icono de la actividad en la presentación (ruta en `WebContent/JavaScript/JSPropiedadesActividades/Iconos.js`).

5. **`getProps()`**: Es la función inversa a `setProps`. Carga los valores de los distintos atributos en los campos asociados en el html de la actividad.

6. **`GetTipoJson()`**: Devuelve el tipo de actividad usado en la presentación (name en WebContent/JavaScript/JSPropiedadesActividades/Iconos.js)

7. **`isInicial()`**: Devuelve `true` si la actividad es inicial, para el caso de `ActEnd` devuelve `false`.

8. **`setProps(HashMap<String, String>)`**: Toma los valores de los distintos atributos del html asociado a las propiedades de la actividad.

El siguiente método es opcional dependiendo de que el formulario de la actividad tenga o no combos que deban ser cargados con datos de la base de datos:

1. **`obtenerDatosCombo(String, int, String, String)`** //Opcional

La clase `ActividadModeloGenerico` es una plantilla Genérica para crear nuevas actividades que generen una sola línea en la tabla `LnFluMsg0172` con todos los métodos antes mencionados y parte del código que debe haber en cada una de ellas.

También se deben agregar las actividades nuevas en la en los métodos de clase `GeneradorActividades`:

```
GenerarActividadPorTipo(String tipo,String
id,HashMap<String,ActProps> mapDataprop)

     if (tipo.equals("nombreActividad")){ //nombre act en
Iconos.js
               actividad=new NombreActividad();
               actividad.setId(id);
     }

GenerarActividadPorTipoJOSN(String tipo)

     if (tipo.equals("nombreActividad")){ //nombre act en
Iconos.js
          actividad=new nombreActividad();
     }

GenerarActividadXActName(String actName)
```

```
//comienzo de id de actividad en IdLine0172 en LnFluMsg0172
if (actName.startsWith("Plugins.IdActividad.")){
          actividad=new nombreActividad();

}
```

PRESENTACIÓN

Para simplificar la tarea de mantenimiento y actualización se separaron las diferentes funcionalidades en varios archivos JavaScript y se parametrizaron las distintas funciones para obtener los valores desde el archivo `iconos.js`.

A continuación, se detallarán los pasos necesarios para el agregado de una nueva actividad de la cual se cuenta con la lógica (Ej: `actividadN`):

1 - Crear un nuevo archivo html con el `div` para la nueva actividad en `PropiedadesDiv`.

```
<div id="propiedades">
   <div>
     <h3>Properties ActividadN</h3>
     <label style="display: none">id</label>
     <label style="display: none" id="idN"></label>
     <input type="checkbox" name="nodoInicialN" id="nodoInicialN"
          value="I"> nodoInicial<br>
     <br><br>
     <fieldset>
          <legend>Properties</legend>
          <label>Dato1</label><br>
          <input id="variableNameN"></input><br><br>
     </fieldset>
     <fieldset>
          <legend>Activity text</legend>
          <input id="actTextN"></input><br>
     </fieldset>
     <br><br>
     <button id="button1" onclick="GuardarDatos(5)">
Guardar</button>
   </div>
</div>
```

Como se puede observar se utilizan los siguientes `id` de campos:

1. `idN`: Identificador de la etiqueta que muestra el identificador del nodo

2. `nodoInicialN`: Identificador del checkbox para poder indicar que es una actividad inicial

3. `variableNameN`: Identificador del campo de ingreso de dato, para cada campo del formulario debe haber uno (puede ser `radio, select, textArea, input`, etc)

4. `actTextN`: id del campo para ingreso del texto de la actividad

Se asigna la función al onclick del botón: **`GuardarDatos(N)`**.

Si el formulario tiene comportamiento (ocultar o mostrar campos, cargar combos...) al seleccionar un componente (radio, select..), o contiene alguna tabla donde se ingresen dinámicamente datos estos deben ingresarse en un JavaScript propio del formulario (`WebContent/JavaScript/JSFormularios`).

Por cada nuevo JavaScript creado en esa ruta de debe ingresar una línea en `WebContent/JavaScript/JSFormularios/DivPropiedades.js`.

```
document.write("<script type='text/javascript'

 src='JavaScript/JSFormularios/DivPropiedadesActFlatFile.js'>   </
script"+">");
```

Siendo `N` el número de actividad agregada.

2 - Se debe agregar la actividad en `Iconos.js` de la siguiente manera:

```
Iconos[N]={name:'nombre',ruta:'images/icoN.jpg',
x:10,y:300,propiedad:"propiedadesN",id:"idN"};
```

Donde:

`name`: Es el nombre asignado a la actividad.

`ruta`: Ruta del ícono elegido para la actividad. (de debe también agregar el ícono a `/WebContent/images`).

`x,y`: Coordenadas en paleta de elementos ('y' varía sumando 80 al valor del elemento anterior).

`propiedadN`: Identificador del `div` en el archivo `index.html`.

3 - Agregar al `index.html` en propiedades el div generado

```
<div id="identificadorActividad"  style="display: none"></div>
```

4 - Agregar en `JqueryComponentesLoadDiv.js`:

```
$('#identificadorAct').load('PropiedadesDiv/
ActName.html#propiedades');
```

5- Agregar en `EntradasDiv.js`

```
Valores[N]=new Array();
Valores[N][1]={tipo:'R',valor:"atributo1N"};
Valores[N][2]={tipo:'I',valor:"atributo2N"};
Valores[N][3]={tipo:'I',valor:"atributo3N"};
Valores[N][4]={tipo:'TA',valor:"atributo4N"};
Valores[N][5]={tipo:'I',valor:"actTextN"};
Valores[N][6]={tipo:'C',valor:"nodoInicialN"};
```

Donde tipo puede ser:

- `R:Radio`
- `I: Input`
- `TA: TaxtArea`
- `S:Combo`
- `C:Check`

Se deben ingresar todos los `id` de atributos que se deben pasar a la logica, no se deben pasar `ids` que se utilicen para otros casos (identificadores de `divs` para mostrarlos u ocultarlos, por ejemplo).

5. Gestión del Proyecto

En este capítulo detallaremos las actividades realizadas durante la duración del proyecto, también como se planifico el manejo de los tiempos y las duraciones reales de cada etapa del mismo. Es importante destacar que el proyecto estaba planeado para un grupo de tres integrantes, pero por razones que no tienen importancia para el mismo uno de ellos no pudo iniciar el proyecto, el cual se completó con los integrantes restantes.

La idea inicial de la duración del proyecto se presenta en el siguiente diagrama de Gantt:

Id.	Nombre de tarea	Inicio	Finalizar	Duración
1	Análisis de tecnologías y análisis del producto	2/5/2016	4/11/2016	405h
2	Desarrollo del producto	7/11/2016	2/8/2018	1362h
3	Documentación y puesta a punto	6/8/2018	6/11/2018	201h
4	Finalización del proyecto	7/11/2018	1/2/2019	189h
5	Contratiempos y retrasos	4/2/2019	7/6/2019	270h

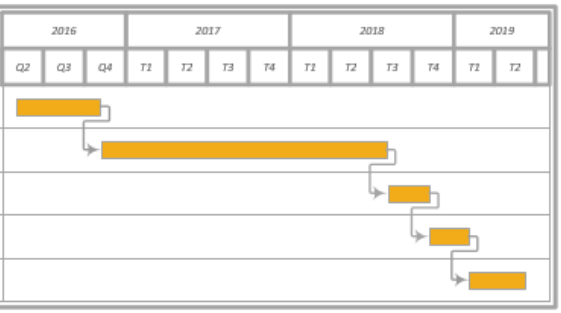


Ilustración 13: Diagrama de Gantt con estimación inicial

En dicha estimación, se platean semanas de 15hs de dedicación por cada integrante del grupo. Se plantearon aproximadamente nueve meses de trabajo para la finalización del proyecto además de dos meses para administrar los errores y contratiempos que se encontraran durante el desarrollo del mismo.

Para los nueve meses, se consideraron dos de los mismos para el análisis de las tecnologías y el estudio de la herramienta pre-existente. Cinco para el desarrollo del producto y los restantes para la puesta a punto, generar la documentación necesaria a presentar y la finalización del mismo.

Durante la primera etapa, se descubrió que la decisión de que tecnología utilizar, así como la negociación por del nuevo alcance del proyecto por la falta de un integrante resulto ser más complicada que lo se pretendido en un principio. Esto llevó a que finalización de la primera etapa se retrasara por aproximadamente dos meses. Durante este tiempo se desarrollaron distintos tipos de prototipos para las tecnologías seleccionadas, así como el estudio de la herramienta pre-existente y sus bases de datos. Se decidió un nuevo alcance, así como en qué áreas del producto a desarrollar serían las críticas.

Por la importancia del producto para la empresa, se mantuvieron reuniones con la misma aproximadamente dos veces por mes, lo cual permitía aclarar dudas y presentar lo que se tenía echo. Este tipo de modalidad ágil logro que los clientes tuvieran tempranamente la apariencia del producto y pudieran detallar sus necesidades con respecto al mismo.

Por problemas personales de ambos integrantes, el desarrollo del producto tuvo varias complicaciones, los que llevo que el mismo se extendiera mucho tiempo más. Durante estos periodos se continuó avanzando en los trabajos necesarios para la finalización del

mismo, pero a una menor velocidad.

Se generó variada documentación durante todo el desarrollo de prototipo. La misma se entregó a los clientes cuando se entregaron los fuentes del producto. La redacción de este informe se realizó utilizando varios de esos documentos, así como algunos se encuentran en los apéndices.

6. Conclusiones y trabajos Futuros

A continuación, se explican las conclusiones que se obtuvieron en el desarrollo del proyecto y como este se podría continuar por la empresa cliente.

6.1. Conclusiones

El objetivo de este proyecto era investigar las tecnologías existentes para posteriormente implementar una nueva versión, con tecnologías más actuales, del dibujador utilizado por la empresa KSIT.

La primera deducción que se puede obtener con el trabajo realizado es que nos encontramos con un entorno tecnológico lo suficientemente avanzado y especializado que permite el desarrollo de este tipo de software de forma web. Gracias a ello, se pudo generar un prototipo básico con algunas de las funcionalidades que presenta el dibujador, que se estudió en la sección de estado del arte, ya que implementarlo en su totalidad no entraba dentro del alcance del proyecto.

En segundo lugar, destacamos que la transición entre un entorno de escritorio a uno web es posible. Aunque parezca un resultado lógico, no fue una tarea sencilla. El análisis del aplicativo actual conllevó una gran porción de la carga total del trabajo realizado. La adaptación de las comunicaciones con la base de datos y el desarrollo de todo el ambiente web nos permite confirmar que es realizable esta transición que era uno de los principales objetivos a alcanzar a través del desarrollo de este trabajo.

Una vez terminado el análisis y familiarizados con las bibliotecas seleccionadas el desarrollo de la aplicación web no represento grandes complicaciones con relación a otros proyectos de desarrollo de software. Esto se puede explicar porque lo único que se reutilizó del aplicativo anterior fue la base de datos y la integración con la misma se logró satisfactoriamente en las primeras etapas del desarrollo.

Consideramos que la implementación del prototipo es de un gran valor para la empresa KSIT. Permitiendo una evolución tecnológica y logística de una herramienta altamente critica para el desarrollo de su negocio. Facilitando el acceso de los desarrolladores de la empresa al dibujador y permitiendo abrir distintas opciones para su utilización, como por ejemplo la facilidad de generar distintas instancias del sistema o la movilidad del entorno de desarrollo.

En cuanto si la herramienta desarrollada es una mejora a la actual, dependerá del esfuerzo en el desarrollo de la versión final producto. A partir del prototipo desarrollado podemos concluir que con los recursos necesarios se podría obtener un excelente producto que superaría en muchos aspectos al producto actual.

Para terminar, se concluye que, si bien el prototipo debe continuar implementándose para que este se pueda sustituir por el dibujador actual, este cubre las principales funcionalidades del mismo logrando una buena base para su futuro. Se logró el objetivo de desprenderse de la tecnología actual, la simplificación de las bases de datos con la

eliminación de la base designer y facilitando las posibilidades de mejoras de la herramienta.

6.2. Lecciones aprendidas

En la etapa de análisis del problema nos encontramos con varias dificultades al momento de crear los diferentes prototipos, ya que no todas las tecnologías permitían implementar todos los requerimientos, seleccionando así la que permitiera abordar la mayor cantidad de funcionalidades e implementando a un más bajo nivel las restantes.

También nos tuvimos que enfocar en comprender como se debería guardar la información en la base de datos, ya que cada actividad implementada, se persiste de diferente forma y no existe mucha documentación sobre cómo se debe guardar dicha información, ni cómo se debe recorrer el flujo para ser mapeado a las diferentes tablas de la base de datos.

6.3. Trabajos futuros

La solución propuesta permite extender las funcionalidades y creación de nuevas actividades.

A continuación, se detalla cómo se puede continuar con el desarrollo del mismo.

a) Agregar nuevas actividades

Para agregar nuevas actividades se creó una clase genérica donde tiene todos los encabezados de los métodos que se deben desarrollar, se podrá utilizar esta como modelo, pero para cada actividad de deberá implementar como se persisten y recuperan los datos de forma específica. Además de esto se debe crear el HTML para ingreso se las diferentes propiedades de cada una de las actividades, y seguir los pasos especificados en la sección de “Agregar una nueva actividad”.

b) Manejo de roles y usuarios:

Se crearon los métodos y la pagina para ingreso de estos datos para poder extenderlos en un futuro, ya que esto no entraba dentro del alcance del proyecto.

c) Cambio de bases de datos

El prototipo actual se implementó para que funcionará con Microsoft SQL Server, ya que fue la base de datos que nos proporcionó el cliente, pero al ser implementada la persistencia con MyBatis es muy simple extender para otros motores de base de datos.

d) Seguridad

La seguridad del producto no se encontraba dentro del alcance de este proyecto. Si bien el uso que se le dará a la herramienta será principalmente para el uso dentro de una LAN, al ser este un prototipo web se tuvo que tener en cuenta en la elección de tecnologías, que estas permitieran a futuro implementaciones para un uso seguro del sistema (como ser uso de HTTPS, seguridad en los servicios RESTful, manejo de roles).

Por otro lado, se seleccionaron tecnologías que no presentaban fallas de seguridad durante el desarrollo de la herramienta. Las comunidades activas de las mismas permitieron considerar que si alguna falla importante se encuentra, es muy probable que sea detectado y solucionado con bastante rapidez.

e) Perfiles por usuario

Otra funcionalidad que quedó por fuera del desarrollo fue la visualización de diferentes

menús y opciones para diferentes roles de usuario, esta tarea también quedo fuera del alcance del proyecto ya que el cliente lo consideró muy complejo y quería volver a analizarlo.

7. Glosario

Prototipo: Modelo del ciclo de vida de un software.

KSIT: Clientes del producto. Se especializan en desarrollo e implementaciones para la industria financiera.

Adaptor: es una infraestructura tecnológica conformada por un conjunto de servicios de software de base con el propósito de asistir la construcción de eficientes y robustos productos verticales orientados al manejo de mensajería, protocolos de comunicación, distribución física, administración de instancias, persistencia, seguridad, autenticación y ejecución de máquina de estado. Soluciones integradas.

Dibujador: Nombre dado a la herramienta de desarrollo gráfica integrada a la suite Adaptor.

Proyecto: Agrupador de modelos.

Modelo: Conjunto de actividades y transiciones que desarrollan funcionalidades Adaptor

Actividad: Entidad especializada que cumple una función específica en el tiempo.

Transacción: Entidades cuyo propósito es relacionar actividades.

Desarrollador: En contexto, usuario de AdaptorDesigne.

Swing: La API Swing es un conjunto de extensiones de componentes GUI que facilitara el desarrollo de aplicaciones *Front End/GUI* basadas en JAVA.

Java: Lenguaje de programación y plataforma informática.

EJB (Enterprise JavaBeans): Interfaces de programación de aplicaciones (APIs) basado en Java.

JavaScript: Lenguaje de programación interpretado basado en Java que se utiliza principalmente del lado del cliente, implementado como parte de un navegador web.

HTML: Lenguaje para el desarrollo de páginas web.

MyBatis: Herramienta de persistencia Java que se encarga de mapear sentencias SQL y procedimientos almacenados con objetos a partir de ficheros XML o anotaciones.

JQuery: Biblioteca multiplataforma de JavaScript que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web.

JointJS: Biblioteca JavaScript para diagramar.

JSON (JavaScript Objet Notation): Formato liviano de intercambio de información.

8. Bibliografía

[1] 10 bibliotecas JavaScript para dibujar tus propios diagramas, http://modeling-languages.com/javascript-drawing-libraries-diagrams/. Último acceso, febrero 2018.
[2] Diagramas JavaScript interactivos en HTML, https://gojs.net/latest/index.html. Último acceso, febrero 2018.
[3] Página principal Jointjs, https://www.jointjs.com/opensource. Último acceso, febrero 2018.
[4] Página principal de JavaFX, http://www.oracle.com/technetwork/java/javase/overview/javafx-overview-2158620.html. Último acceso, febrero 2018.
[5] Página principal de Cytoscape, www.cytoscape.org. Último acceso, febrero 2018.
[6] Página principal D3JS, https://d3js.org/. Último acceso, febrero 2018
[7] Manual de usuario JGraphX (JGraph 6), https://jgraph.github.io/mxgraph/docs/manual_javavis.html. Último acceso, febrero 2018.
[8] Página principal MyBatis, http://www.mybatis.org/mybatis-3/es/index.html. Último acceso, febrero 2018.
[9] Introducción a JSON, https://www.json.org/json-es.html. Último acceso, febrero 2018.
[10] Construyendo RESTful Web Services con JAX-RS, https://docs.oracle.com/javaee/6/tutorial/doc/giepu.html. Último acceso, febrero 2018.
[11] ADAPTOR Designer: Manual del usuario, KSIT, páginas: 4-12. 2016.
[12] Manual de mantenimientos K&S, KSIT, pagina: 1. 2016.
[13] Modelo de Datos de AdaptorDesigner, KSIT, páginas: 1-2, 2016.
[14] Agent of Formats and Message Flows, KSIT, 2016.
[15] ADAPTOR Diseñador de flujos de integración (Enterprise application integration), KSIT, 2016.
[16] Pro RESTful APIs: Design, Build and Integrate with REST, JSON, XML and JAX-RS, Sanjay Patni, marzo 2017.
[17] RESTful Java with JAX-RS 2.0, Bill Burke (O'Reilly 2nd Edition), 2013.
[18] Página principal Cliente, http://www.ksasociados.com/web/. Último acceso, febrero 2018.
[19] Fundamentos de jQuery, Rebecca Murphey, 2013
[20] Página principal Jboss, http://www.jboss.org/. Último acceso, febrero 2018.
[21] Java SWING. Java GUI library, Tutorials Point, 2016.

9. Apéndices

9.1. Apéndice 1 - Bibliotecas Estudiadas

a) GOJS	
Tipo	biblioteca JavaScript
Licencia	Paga
Pros	• Permite la implementación de diagramas interactivos con nodos complejos. • Multiplataforma (Browsers y SO) • Templates y layouts modificables. • JavaScript puro. • No depende de bibliotecas o frameworks. • Corre completamente en el navegador.
Contras:	• Licencia

b) JointJS Core	
Tipo	biblioteca HTML5 JavaScript
Licencia	Open Source
Pros	• Diagramas de flujo HTML5, BPMN y otros diagramas. • Permite crear diagramas estáticos y herramientas interactivas de diagramado como editores de flujos y herramientas. • Fácil de integrar a cualquier tecnología de backend
Contras:	• Rappid es la versión paga de JointJS que provee plugins con varias funcionalidades extra (y la licencia es bastante cara)

JointJS:

1. Permite la utilización de formas predefinidas (Rectángulo, Circulo, Elipse, Text...)
2. Permite la creación de formas personalizadas Vía SVG
3. Links y elementos interactivos
4. Permite conectar elementos mediante enlaces
5. Links (flechas) personalizados
6. Provee puntos de conexión para creación de enlaces
7. Serialización y deserialización con Formato JSON
8. Manejo eventos (click, dobleclick, move...)
9. Zoom In/Out
10. Filtros y Gradientes
11. Soporte para Node JS
12. Rápido: Puede renderizar cientos de elementos y enlaces instantáneamente.
13. Arquitectura MVC

Rappid provee:

1. Paneles de control
2. Paletas de elementos (drag and drop)
3. Deshacer/Rehacer
4. Selecciones
5. Copiar-Pegar
6. Validaciones

7. Pan/Zoom
8. Paneles de propiedades editables
9. Layouts
10. Tooltips
11. Algoritmos de grafos
12. Exportar a PNG/JPG/SVG
13. Imprimir

Implementación de algunas funcionalidades con Jointjs

1. Iconos con Imágenes:

Para implementar esto se tuvo que crear una forma personalizada utilizando tags SVG.

2. Stencil (paleta de elementos Drag and drop):

Se logró implementar esto creando un nuevo paper en otro div diferente, este paper tiene figuras más simples (rectángulos con íconos asignados como imágenes).

La implementación de esto consiste en crear un tercer paper en un nuevo div donde se copia el elemento que se quiere clonar, mediante eventos del mouse se obtiene el icono se dragueo y al soltarlo en el paper donde está el grafo se crea una "copia" de este elemento.

3. Icono de acceso rápido para eliminar elementos

La implementación de esto se solucionó utilizando tags SVG personalizados, la eliminación se realiza haciendo click con el ctrl presionado. Para esto se creó un archivo JS para escuchar el teclado, cuando se hace click en el icono se verifica si la tecla ctrl esta presionada, y solo elimina en ese caso.

4. Guardar y Abrir grafos:

El grafo se guarda en formato JSON al que se concatenan las propiedades asignadas a cada nodo y/o flecha (con su correspondiente id). Para cargar un grafo guardado deserializa el JSON del grafo lo que permite la generación automática del mismo y se cargan las propiedades en una estructura javascript.

5. Paneles de propiedades

Utilizadas para la visualización y edición de los atributos de los elementos y Links. Cuando se hace click en estos elementos se oculta o muestra el div correspondiente, y se setea el id de dicho elemento para luego poder identificarlo.

6. Paper extensible

Implementado mediante el desplazamiento de los elementos. A futuro sacar Scroll y mover solamente el paper (pan and zoom)

7. Spliter de barra de herramientas:

Para implementar esto se tuvo que marcar como rezizable uno de los divs, esto permite cambiarle el tamaño, al cambiar el tamaño del mismo se actualizan los anchos de ambos.

8. Varias pestañas:

Para implementar esto se utilizaron las “Tabs” de jquery, el grafo se guarda en un div fijo, y al hacer click en una pestaña diferente a la actual, se guardan los datos (JSON y datos) del grafo actual y se levanta (de memoria) los datos del grafo correspondiente a el tab clickeado. De esta forma solo se tiene 1 solo elemento “graph” de Jointjs.

Para guardar en Archivo todos los grafos, se utiliza la estructura que estaba en memoria con todos los JSON+Datos y otra estructura con el nombre de la pestaña ingresada por el usuario. Dicho archivo entonces contendrá el id,nombre e información (JSON+Datos) de cada uno de los grafos que forman el proyecto.

9. Árbol:
10. Alerta al cerrar navegador y guardado en local storage:
11. Divs de propiedades en Archivos independientes:
12. Cargado de información del formulario e íconos de cada actividad desde archivos:

Dificultadas encontradas:

Implementar algunas de las funcionalidades que provee rappid implica estar ampliamente familiarizado con SVG, Backbone y JointJs.

Se logró implementar la funcionalidad de panning para no tener que usar Scroll para moverse con la hoja, pero aún no se pudo ajustar lo suficiente a las otras funcionalidades ya logradas

Otra funcionalidad que se está trabajando es la creación de flechas mediante drag and drop entre nodos, para esto se utilizó una funcionalidad de JointJS llamada (ports), se creó el mismo en el centro del nodo y esto permitiría arrastrar hacia otro nodo para crear una flecha, aun no se pudo integrar a las otras funcionalidades y se está investigando como setear diferentes tipos de flechas de esta forma.

c) JavaFX	
Tipo	Plataforma de Interface de Usuario basada en Java.
Licencia	Open Source
Pros	• API clara. • Estilos CSS • FXML scriptable, lenguaje basado en XML para definir interfaces de usuario. • Motor web de rendering • Gran interacción con Swing. • Deploy global por Java Runtime Enviroment (JRE) • Importante esfuerzo e inversión de Oracle. Soporte.
Contras:	• Posibles bugs. • Sin acceso a estructuras en plataformas Windows por JNI. • Baja cantidad de bibliotecas de terceros. • Necesita instalación.

Se probó esta biblioteca por la capacidad de crear las pantallas a partir de archivos tipo XML. Esta biblioteca no provee funcionalidades de grafos, por lo tanto, se debe implementar a mano todas dichas funcionalidades (drag and drop, conexión mediante flechas, tipos de flechas…etc.). Se descartó esta herramienta por la dificultad que implica implantar con dicho lenguaje todas estas funcionalidades.

d) Cytoscape Web	
Tipo	Componentes Flash con una API Javascript
Licencia	Open Source
Pros	• Soporta GraphML, XGMML y SIF
Contras:	• Usuario debe utilizar un browser moderno con un plugin Flash instalado
NOTA:	En desuso, se sustituyó por Cytoscape JS

e) Cytoscape JS	
Tipo	biblioteca JavaScript
Licencia	Open Source
Pros	• biblioteca JS para manipulación de grafos • Plugings con nuevas funcionalidades -Halo de propiedades -Creación de enlaces mediante drag and drop
Contras:	• Algunos plugins tienen errores • Más complejo de utilizar que JointJS

Se probó la biblioteca creando un prototipo.

- Click en botones para crear nodos.
- Cada nodo tiene un punto rojo que si arrastra hacia otro nodo se crea una flecha que los conecta.
- Clic derecho sobre el nodo: Despliega un menú para eliminar los nodos (con sus flechas asociadas) o acceder al panel de propiedades del mismo.
- Click derecho/clean para eliminar todo.

Dificultades encontradas que no se lograron resolver:

- Barra de herramientas desde donde se arrastren los elementos.
- Generar diferentes tipos de flechas entre nodos que se seleccione del menú
- Menú con click derecho para las flechas (para eliminarlas o ver sus propiedades)
- El plugins para crear flechas entre nodos no funciona correctamente y tiene bugs.
- Flechas que se puedan doblar (similares a las que posee JointJS)

f) D3.js	
Tipo	biblioteca Javascript
Licencia	Open Source
Pros	• Enfatiza en los entandares de la web • Potentes componentes de visualización • Soporta gran variedad de browsers modernos (Firefox, Chrome, Safari, Opera, IE9+, Android y iOS). • Liviana. • Altamente modificable. • Gran comunidad.
Contras:	• Principal objetivo: visualización de datos. • No soporta IE8 o menores • Curva de aprendizaje complicada • No tiene exportación nativa de gráficos.

g) JGraphX	
Tipo	biblioteca Java swing
Licencia	Open Source
Pros	biblioteca para visualizas diagramas interactivos y grafos. Simple de usar, rápida curva de aprendizaje. Funcionalidades: • Dibujar nodos y conectarlos entre si mediante flechas. • Permite asignar estilos tanto a nodos como a las flechas. • Existen varios ejemplos y documentación de uso • Existen foro de dudas
Contras:	• No tiene documentado todo el alcance que tiene la biblioteca y algunas se deben aprender a utilizar a partir de códigos de ejemplo y de foros de preguntas

Dificultades Encontradas y como se solucionaron

Se encontraron varias dificultades en la creación de este prototipo más allá de las dificultades típicas de trabajar con swing (Manejo de paneles, layout...)

1- Panning:

Una de las funcionalidades a implementar fue aumentar el tamaño del grafo a medida que se movían los nodos del mismo, para esto se creó un ScrollPanel que contenga el panel con el grafo, también se le dio un tamaña grande a dicho panel y se habilito el panning del grafo mediante el método, esto permite moverse con el Mouse (Ctrl+Shift +botón derecho) por toda la hoja que contiene al grafo.

2- Iconos para los nodos:

Para solucionar esto se seteó el estilo de cada nodo, asignando una imagen en vez de un color de fondo

3- Diferentes tipos de flechas:

Esta funcionalidad se implementó seteando un estilo default de flecha (stylesheet) diferente cuando se presiona el botón con cada una de las flechas, al hacer esto todas las flechas creadas tendrán dicho estilo.

4- Guardado y respaldo de información

Para el guardado de la información se utiliza una funcionalidad de la biblioteca para serializado y deserializado del grafo en XML, concatenando con las propiedades ingresadas mediante los paneles de edición del grafo (identificadas por id.)

Dificultadas encontradas que aún no se pudieron solucionar:

Un problema encontrado fue el seteado de estilos de las flechas (Edges) que se realiza sustituyendo la hoja de estilos por defecto para flechas, pero al realizar esto no queda ningún estiló asignado a la flecha y no se puede identificar para el seteo de propiedades y serialización. Todavía no se pudo encontrar una solución adecuada al problema, una posible solución sería sobrescribir la creación automática de flechas que provee la biblioteca (mediante drag and drop) o identificar cuando se crea una flecha y guardar en otra estructura el id de la misma junto con el tipo.

Agregar una nueva actividad

Para simplificar esta tarea se creó un archivo de propiedades donde se guarda el nombre del estilo y el icono asignado a cada tipo de nodo:

cantidad=4

nodo1.style=ROUNDED1
nodo1.imagen=/images/ico1.jpg
nodo2.style=ROUNDED2
nodo2.imagen=/images/ico2.jpg
nodo3.style=ROUNDED3
nodo3.imagen=/images/ico3.jpg
nodo4.style=ROUNDED4
nodo4.imagen=/images/ico4.jpg

Estos datos se toman de este archivo al crear los botones para crear transacciones, como también el estilo de cada uno de ellos. Por esta razón para crear una nueva actividad, a nivel del dibujo solo basta con incrementar en uno la cantidad del archivo de propiedades y agregar estas 2 líneas al mismo:

nodo5.style=ROUNDED5
nodo5.imagen=/images/ico5.jpg

Actualmente el panel donde se ingresan valores para cada elemento es muy simple y siempre es el mismo por lo tanto no se cambia nada según la actividad seleccionada, en un futuro de debería guardar en algún archivo o BD los ítems de cada una de las actividades para crear cada uno de estos paneles dinámicamente.

9.2. Apéndice 2 - Aplicación de escritorio Vs Aplicación web

a) Web

- Facilidad de Acceso (Se ejecutan desde cualquier ordenador que se conecte al servidor.)
- Ligeras: el Usuario no necesita tener un ordenador de grandes prestaciones para trabajar con ellas.
- No se debe configurar ni instalar en cada máquina que se va a utilizar, aunque cada cliente va a tener que realizar la configuración en el servidor.
- Facilidad de actualizar u mantener (ya que solo se actualiza en el servidor donde están alojadas)
- No hay problemas de incompatibilidad entre versiones
- Datos centralizados

Desventajas:

- Visualización y compatibilidad con diferentes navegadores (no funciona con navegadores viejos)
- Menos seguridad: Si se levanta como aplicación en Internet está más expuesta a riesgos de seguridad y un gran número de usuarios no deseados.
- Problemas de conectividad con el servidor pueden dar problemas de performance y accesibilidad.

b) Escritorio

Ventajas:

- Se puede buscar la forma de trabajar *Offline* (sin necesidad de estar en red) y que sea necesario conectarse al servidor solo para compilar/obtener proyectos compilados.
- Seguridad
- Rendimiento: el tiempo de respuesta es muy rápido.
- Desventajas:
- Son dependientes del sistema operativo que utilice el ordenador y sus capacidades (video, memoria, etc.).

- Requieren instalación y actualización personalizadas (Se debe preparar el ambiente, instalar java, Base de Datos para cada máquina en la que se utilice.)

- Se puede buscar la forma de trabajar Offline (sin necesidad de estar en red) y que sea necesario conectarse al servidor solo para compilar/obtener proyectos compilados.

- Seguridad

- Rendimiento: el tiempo de respuesta es muy rápido.

Desventajas:

- Son dependientes del sistema operativo que utilice el ordenador y sus capacidades (video, memoria, etc.).

- Requieren instalación y actualización personalizadas (Se debe preparar el ambiente, instalar java, Base de Datos para cada máquina en la que se utilice.)

9.3. Apéndice 3 - Persistencia de flujos

II. **Guardado/Obtención de Flujos de Base de Datos**

Para poder levantar dibujos realizados por el “Dibujador Anterior ” se deben obtener los flujos de la Base de datos MMProdat.

Nombre del Flujo:

Columna “F1Mens01650172” - Tabla “[MMProdat].[dbo].[LnFluMsg0172]”

Reconocer Actividades

Para poder obtener un dibujo de un flujo determinado primero se obtienen las actividades de dicho flujo, para esto se seleccionan los valores que sean diferentes de la columna **idLine0172** para el dibujo dado. A partir de dicha columna se pueden obtener el “tipo de actividad” (para una actividad se puede repetir dicho nombre en más de una fila).

Cada Actividad tiene asociado un número ordinal (o más de uno) que se almacena en la columna **[Ordina0172]** para determinar el número de línea dentro del flujo. Utilizado para la ejecución (ejecución lineal y determinar saltos).

Reconocer Relaciones entre actividades

Para poder dibujar las flechas se obtienen los valores de las siguientes columnas:

[OutCon0172]: Indica el flujo principal mediante una sentencia “EJECUTAR:operación”. Ejemplos de operación:

- "GOTO X"
- IIF(LEN(Nombre) =0,"GOTO X",IIF(LEN(Nombre) <> 0,"GOTO Y","GOTO -1"))
- FINALIZAR

Donde X e Y son valores de Ordina0172.

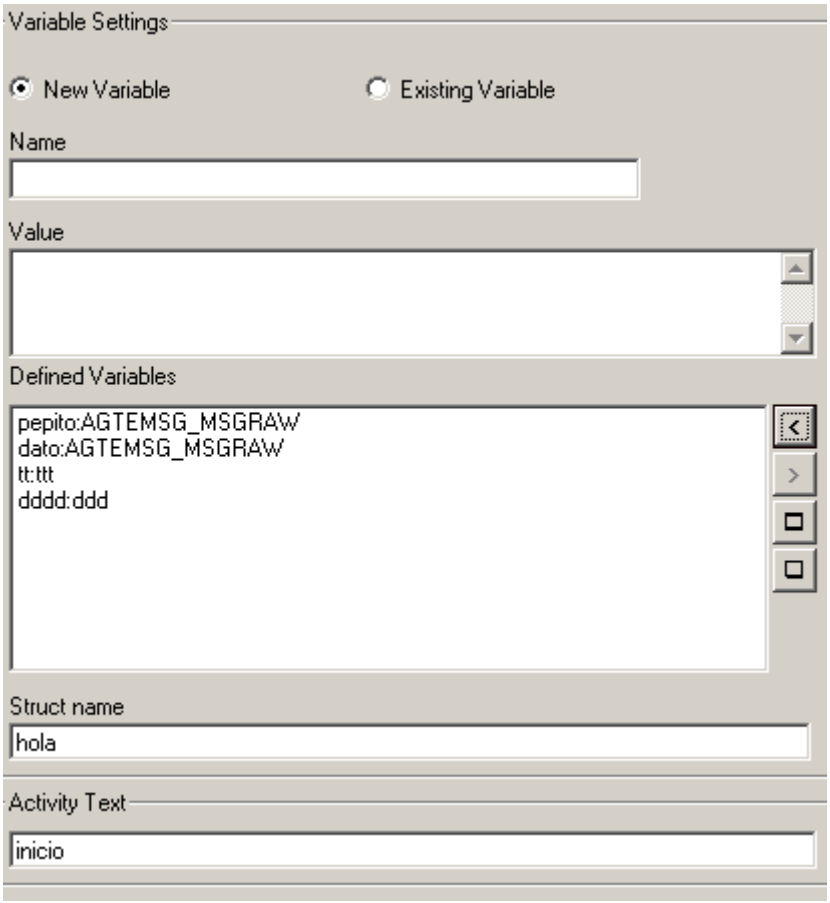
[ErrCon0172]: Indica el flujo en caso de error (por ejemplo: EJECUTAR:CANCELAR)

Para obtener las relaciones entre las actividades se recorre cada una de ella, y se dibuja una flecha entre dicha actividad y la actividad con el ordinal X.

III. **2- Persistencia de Actividades**

1 Define Variable

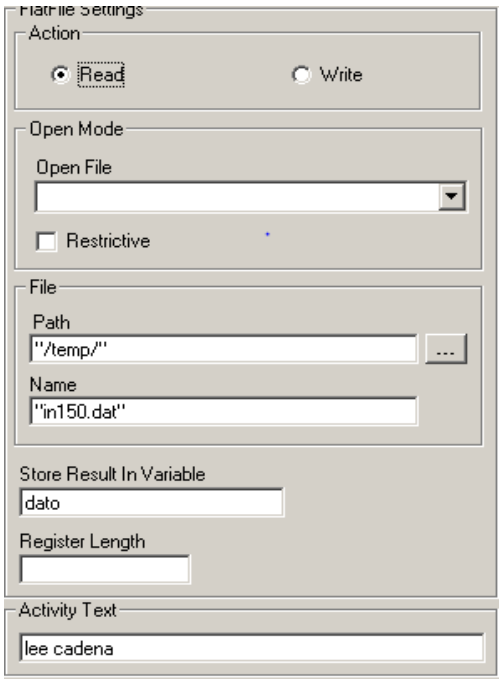
Nombre en BD: Plugins.ActDefineVar

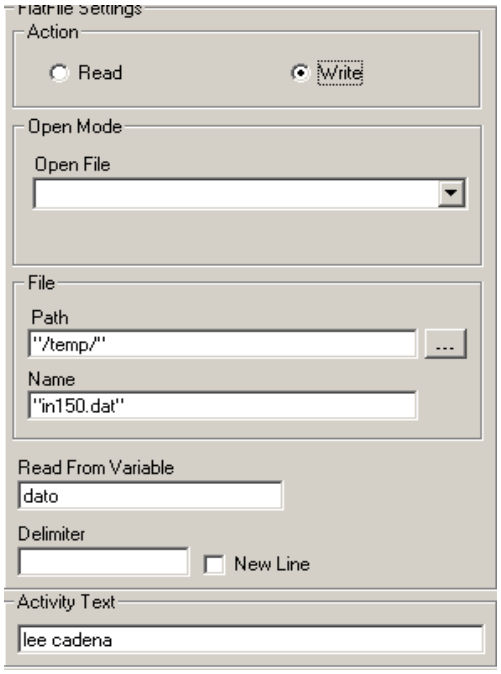


En la Base de datos se guardan los datos del campo Defined Variables (estos se agregan al Text Area a partir de input Name y Value.

Campo en formulario	Campo en la BD	Comentario
Defined Variables	[Parame0172]	Formato : name1:valor1;name2:valor2
Struct name	[CpOrDt0172]	Formato: Texto
Activity Text	NA	Se guarda en Designer, no se guarda en MMProdat

Nombre en BD: Plugins.ActFlatFile



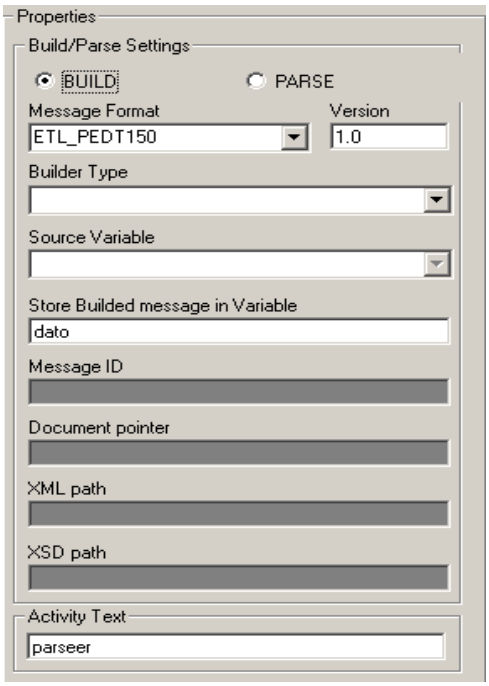


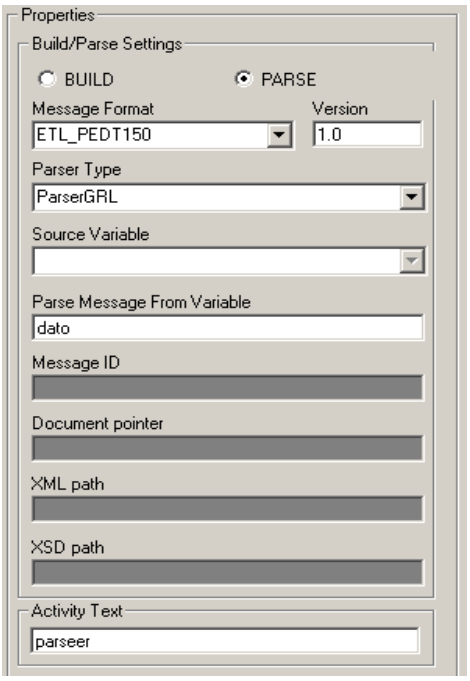
Esta actividad guarda en 2 líneas

Campo en formulario	Campo en la BD	Comentario
Read	Accion0172	RECEIBE (linea 2)
Open File	Parame0172	AGTEMSG_OPENFILE:"OUTPUT|"; (linea 1)
Restrictive	Parame0172	AGTEMSG_OPENFILE:"algo|**R**"; (linea 1)
Path	Parame0172	AGTEMSG_PATHFILE:"/temp/"; (linea 1)
Name	Parame0172	AGTEMSG_NAMEFILE:"in150.dat"; (linea 1)
Store Result in Variable	CpOrDt0172	(linea 2) y al final de Parame0172
Register Legth	Parame0172	AGTEMSG_LENREGFILE:; (linea 1)
Write	Accion0172	SEND (linea 2)
Read From Variable	CpOrDt0172	(linea 2)
Delimiter	Parame0172	AGTEMSG_DELREGFILE:"6" (linea 1)
New line	Parame0172	AGTEMSG_DELREGFILE:"NL" (linea 1)
Activity Text	NA	Se guarda en Designer, no se guarda en MMProdat
Open File		Ver de donde sale (Creo que son fijos)

3 Parse/Build

Nombre en BD: Plugins.ActBuildParse



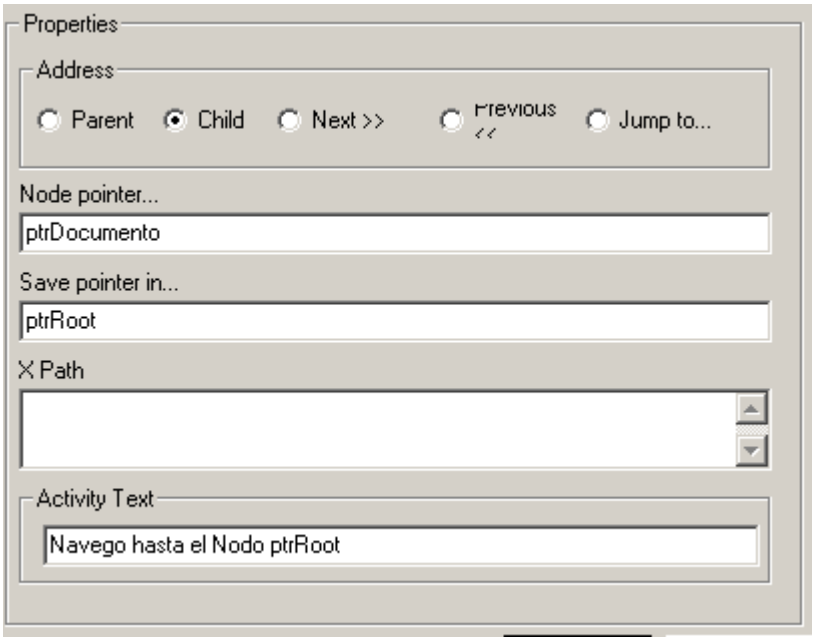


Esta actividad guarda en 1 linea o en 2.

Campo en formulario	Campo en la BD	Comentario
BUILD/PARSE	Accion0172	Valores: BUILD/PARSE
Message Format	F2Mens01650172	Combo cargado desde CbFtoMsg0165 – IdMens0165
Version	F2Vers01650172	Si se cambia se carga desde CbFtoMsg0165 Version0165
Builder/Parser Type	Parame0172	
Source Variable		
Store Builded message in Variable/Parser Message From Variable	CpOrDt0172	
Message ID	Parame0172	
Document pointer	Parame0172	
XML path	Parame0172	
XSD path	Parame0172	
Activity Text	NA	Se guarda en Designer, no se guarda en MMProdat

4 Browse

Nombre en BD: Plugins.ActNavegar



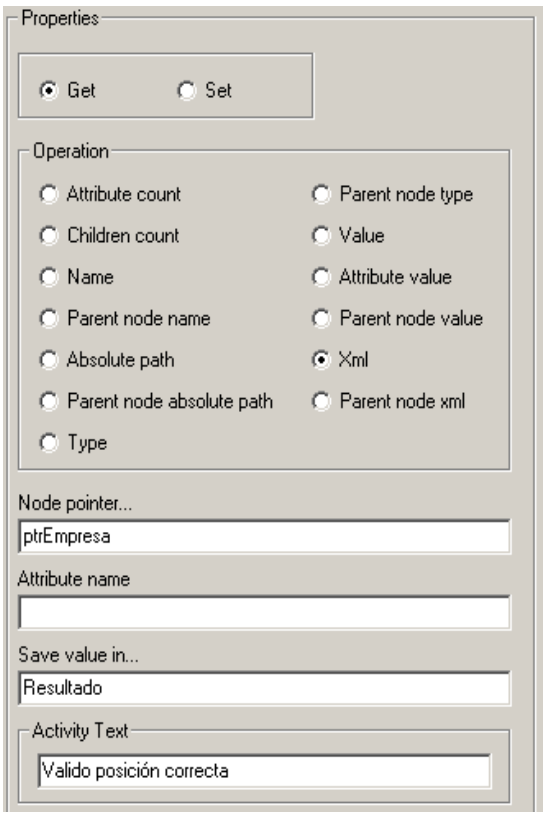
Campo en formulario	Campo en la BD	Comentario
Address	Parame0172	Formato : valor1:valor2:valor3:valor4. Valor2
Node pointer...	Parame0172	Valor1
Save pointer in...	Parame0172	Valor3
X Path	Parame0172	Valor4
Activity Text		

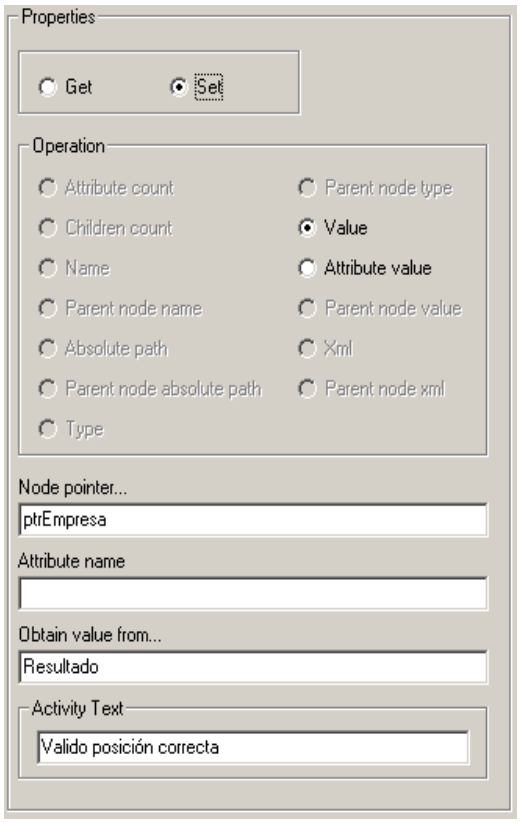
Address:

- XPATHCHILD : Jump To ...
- FIRSTCHILD : Child
- NEXTSIBLING : Next
- PARENT : Parent
- PREVIOUSSIBILING : Previous

5 Values

Nombre en BD: Plugins. ActValores

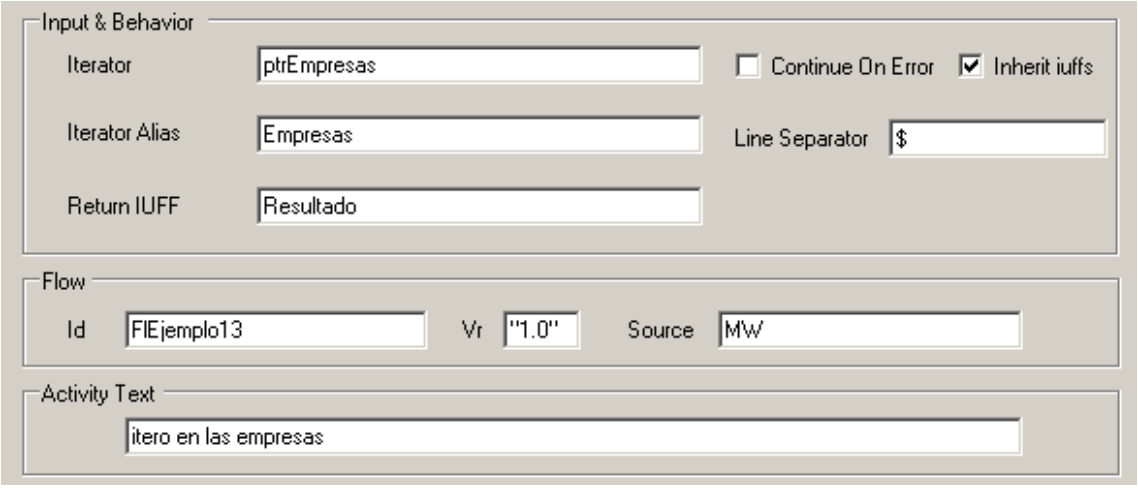




Campo en formulario	Campo en la BD	Comentario
Get/Set	Accion0172	GETPROPERTYNODEXML/ SETPROPERTYNODEXML
Operation	Parame0172	Formato : valor1:valor2:valor3:valor4. Valor2
Node pointer...	Parame0172	Valor1
Attribute name	Parame0172	Valor4
Obtain value from...	Parame0172	Valor3
Save value in...	CpOrDt0172	
Activity Text		

6 Iterator For Each Flow

Nombre en BD: Plugin.ITRForEachFlow

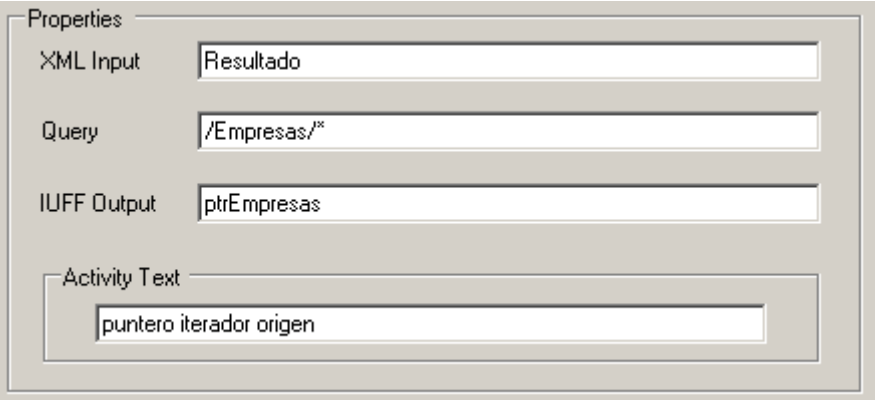


Esta actividad guarda en 3 lineas en LnFluMsg0172 (ADDIUFF, ADDIUFF y ERROR)

Campo en formulario	Campo en la BD	Comentario
Iterator	Parame0172	Formato : Parameter0:valor0;Parameter1:valor1;Parameter2:valor2;Parameter3:valor3;Parameter4:valor4;Parameter5:valor5;Parameter6:valor6;Parameter7:valor7; Parameter0 – Linea 1
Continue On Error	Parame0172	valor5 – Linea 1
Inherit iuffs	Parame0172	valor6 – Linea 1
Iterator Alias	Parame0172	valor1 – Linea 1
Line Separator	Parame0172	valor7 – Linea 1
Return IUFF	Parame0172	Valor:Resultado:ITRForEachFlow(p0,p1,p2,p3,p4,p5,p6,p7) Linea 2
Id	Parame0172	valor2 – Linea 1
Vr	Parame0172	valor3 – Linea 1
Source	Parame0172	valor4 – Linea 1
Activity Text		

7 Activity Text

Nombre en BD: Plugin.XpathQuery



Campo en formulario	Campo en la BD	Comentario
XML Input	Parame0172	Formato : valor1;valor2;valor3 valor1
Query	Parame0172	valor2
IUFF Putput	Parame0172	valor3
Activity Output		

8 FIN

Sin parámetros, en la columna [OutCon0172] tiene el valor EJECUTAR:FINALIZAR

9.4. Apéndice 4 - Datos de Prueba

Presentaremos los datos para cada modelo de flujo del proyecto EjemploVarios 1.0 y utilizaremos su representación en la base de datos para determinar si nuestro dibujador está realizando correctamente las inserciones en la base de datos MMProdat.

1 Ejemplo Parser

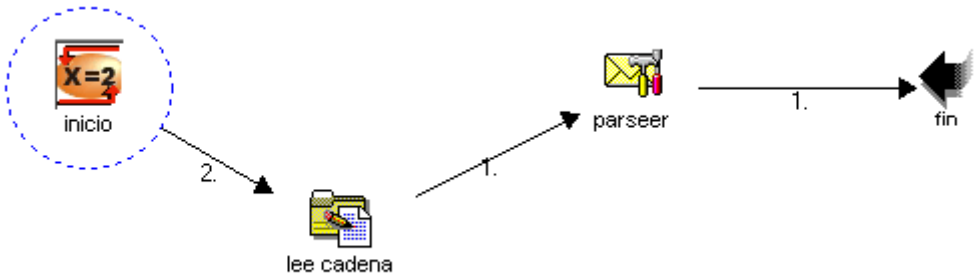


Ilustración 14: Flujo EjemploParser 1.0

Actividades

Actividad	Atributos	Datos
Define Variable	Name	dato
	Value	AGTEMSG_MSGRAW
	Activity Text	inicio
		NODO INICIAL
Managing Files	Action	Read
	Open File	INPUT
	Path	"/temp/"
	Name	"in150.dat"
	Store Result In Value	dato
	Register Length	99999999
	Activity text	lee cadena
Parse/Build	Operation	PARSE
	Message Format	ETL_PEDT150
	Version	01/01/00
	Parser Type	ParserGRL
	Parse Message From Variable	dato
	Activity Text	parseer
Flow End	Activity Text	fin

Transiciones

Tipo	Index	Origen	Destino
Transición	2	inicio	lee cadena
Transición	1	lee cadena	parseer
Transición	1	parseer	fin

2 EjemploXML13

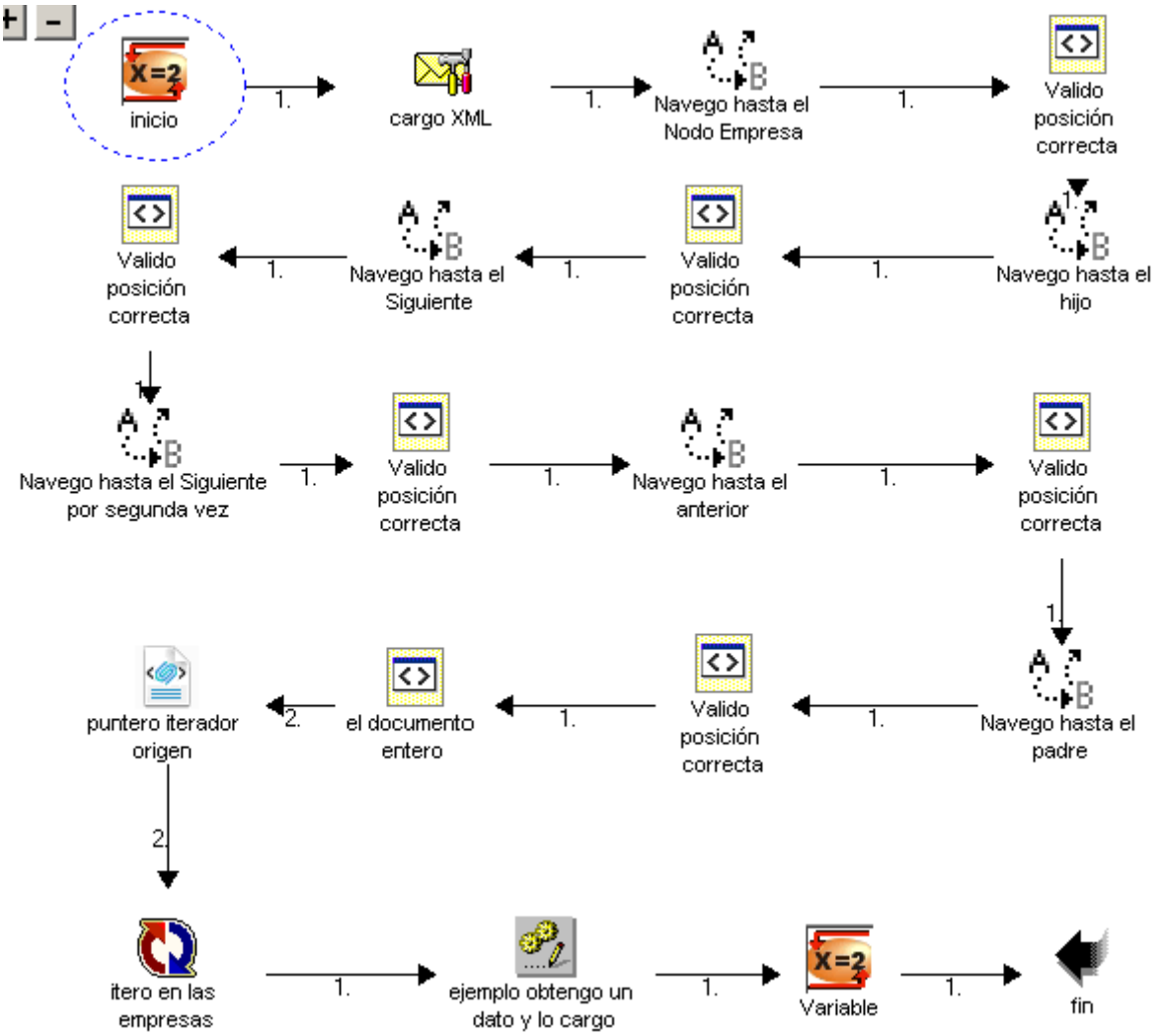


Ilustración 15: Flujo EjemploXML13 1.0

Actividades

Actividad	Atributos	Datos
Define Variable	Name	inicio
	Value	""
	Activity Text	inicio

		NODO INICIAL
Parse/Build	Operation	PARSE
	Message Format	formatoXML
	Version	1.0
	Parser Type	ParserXML
	Document pointer	ptrDocumento
	XML path	/temp/Ejxsd2.xml
	Activity Text	cargo XML
Browse	Address	Jump to...
	Node pointer...	ptrDocumento
	Save pointer in...	ptrEmpresa
	Xpath	"Empresas/Empresa"
	Activity Text	Navego hasta el Nodo Empresa
Values	Type	Get
	Operation	Xml
	Node pointer...	ptrEmpresa
	Save value in...	Resultado
	Activity Text	Valido posición correcta
Browse	Address	Child
	Node pointer...	ptrEmpresa
	Save pointer in...	ptrHijo
	Activity Text	Navego hasta el hijo
Values	Type	Get
	Operation	Xml
	Node pointer...	ptrHijo
	Save value in...	Resultado
	Activity Text	Valido posición correcta
Browse	Address	Next
	Node pointer...	ptrHijo
	Save pointer in...	ptrSiguiente
	Activity Text	Navego hasta el Siguiente
Values	Type	Get
	Operation	Xml
	Node pointer...	ptrSiguiente
	Save value in...	Resultado
	Activity Text	Valido posición correcta

Browse	Address	Next
	Node pointer...	ptrSiguiente
	Save pointer in...	ptrSiguiente
	Activity Text	Navego hasta el Siguiente por segunda vez
Values	Type	Get
	Operation	Xml
	Node pointer...	ptrSiguiente
	Save value in...	Resultado
	Activity Text	Valido posición correcta
Browse	Address	Parent
	Node pointer...	ptrSiguiente
	Save pointer in...	ptrAnterior
	Activity Text	Navego hasta el anterior
Values	Type	Get
	Operation	Xml
	Node pointer...	ptrAnterior
	Save value in...	Resultado
	Activity Text	Valido posición correcta
Browse	Address	Parent
	Node pointer...	ptrHijo
	Save pointer in...	ptrPadre
	Activity Text	Navego hasta el padre
Values	Type	Get
	Operation	Xml
	Node pointer...	ptrPadre
	Save value in...	Resultado
	Activity Text	Valido posición correcta
Values	Type:	Get
	Operation	Xml
	Node pointer...	ptrDocumento
	Save value in...	Resultado
	Activity Text	el documento entero
Xpath Query	XML Input	Resultado
	Query	/Empresa/*
	IUFF Output	ptrEmpresas

	Activity Text	Puntero iterador origen
Iterator For Each Flow	Iterator	ptrEmpresas
	Iterator Alias	Empresas
	Return IUFF	Resultado
	Inherit iuffs	TRUE
	Line Separator	$
	Id	FlEjemplo13
	Vr	"1.0"
	Source	MW
	Activity Text	itero en las empresas
Expression Evaluator	List of variables with expressions defined	CargoNombre1:XV(ptrEmpresas,'Nombre','VALUE')
	Activity Text	ejemplo obtengo un dato y lo cargo
Define Variable	Type	New Variable
	Defined Variables	Resultado:Resultado CargaNombre:CargaNombre CargaNombre1:CargaNombre1
	Activity Text	Variable
Flow End	Activity Text	fin

Transiciones

Tipo	Index	Origen	Destino
Transición	1	inicio	cargo xml
Transición	1	cargo xml	Navego hasta el Nodo Empresa
Transición	1	Navego hasta el Nodo Empresa	Valido posición correcta (1)
Transición	1	Valido posición correcta (1)	Navego hasta el hijo
Transición	1	Navego hasta el hijo	Valido posición correcta (2)
Transición	1	Valido posición correcta (2)	Navego hasta el Siguiente
Transición	1	Navego hasta el Siguiente	Valido posición correcta (3)
Transición	1	Valido posición correcta (3)	Navego hasta el Siguiente por segunda vez
Transición	1	Navego hasta el Siguiente por segunda vez	Valido posición correcta (4)
Transición	1	Valido posición correcta (4)	Navego hasta el anterior
Transición	1	Navego hasta el anterior	Valido posición correcta (5)
Transición	1	Valido posición correcta (5)	Navego hasta el padre
Transición	1	Navego hasta el padre	Valido posición correcta (6)
Transición	1	Valido posición correcta (6)	el documento entero

Transición	2	el documento entero	puntero iterador origen
Transición	2	puntero iterador origen	itero en las empresas
Transición	1	itero en las empresas	ejemplo obtengo un dato y lo cargo
Transición	1	ejemplo obtengo un dato y lo cargo	Variable
Transición	1	Variable	fin

3 EjemploXML1

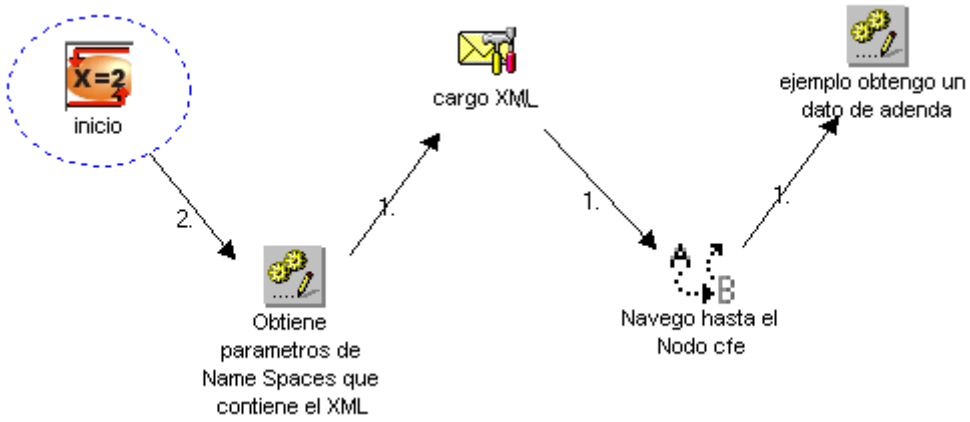


Ilustración 16: Flujo EjemploXML1 1.0

Actividades

Actividad	Atributos	Datos
Define Variable	Type	New Variable
	Defined Variables	inicio:"" vp1:"DGICFE:EnvioCFE/ns0:CFE"
	Activity Text	inicio
		NODO INICIAL
Expression Evaluator	List of variables with expressions defined	AGTEMENS_XPATH_NAMSESPACES:SELECTPARAMETER("FEDGI","XML","NAMESPACE")
	Activity Text	Obtiene parámetros de NameSpaces que contiene el XML
Parse/Build	Operation	PARSE
	Message Format	formatoXML
	Version	1.0
	Parser Type	ParserXML
	Document pointer	ptrDocumento

	XML path	/temp/Ejxml1.xml
	Activity Text	cargo XML
Browse	Address	Jump to...
	Node pointer...	ptrDocumento
	Save pointer in...	ptrcfe
	Xpath	vp1
	Activity Text	Navego hasta el Nodo cfe
Expression Evaluator	List of variables with expressions defined	EmisorElectronico:XV(ptrcfe, 'ADENDA/EmisorElectronico', 'VALUE') MailReceptor:XV(ptrcfe, 'ADENDA/MailReceptor', 'VALUE') NombreSobre:XV(ptrcfe, 'ADENDA/NombreSobre', 'VALUE')
	Activity Text	Ejemplo obtengo un dato de adenda

Transiciones

4 EjemploIteraXML

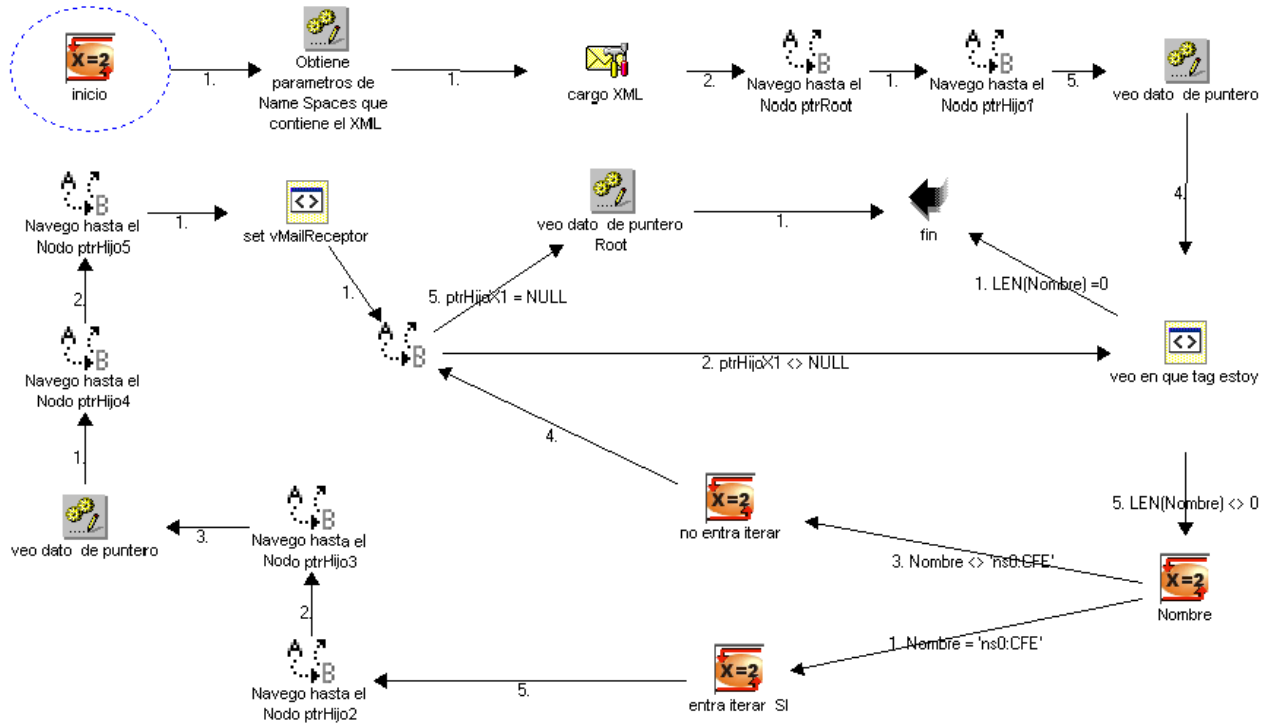


Ilustración 17: Flujo EjemploIteraXML 1.0

Actividades

Actividad	Atributos	Datos
Define Variable	Type	New Variable
	Defined Variables	inicio:"" vp1:"DGICFE:EnvioCFE" vMailReceptor:"dato@dato.com" vp2:"ADENDA/MailReceptor"
	Activity Text	inicio
		NODO INICIAL
Expression Evaluator	List of variables with expressions defined	AGTEMENS_XPATH_NAMSESPACES:SELECTPARAMETER("FEDGI","XML","NAMESPACE")
	Activity Text	Obtiene parámetros de Name Spaces que contiene el XML
Parse/Build	Operation	PARSE
	Message Format	formatoXML
	Version	1.0
	Parser Type	ParserXML
	Document pointer	ptrDocumento
	XML path	/temp/Ejxml2.xml
	Activity Text	cargo XML
Browse	Address	Child
	Node pointer...	ptrDocumento
	Save pointer in...	ptrRoot
	Activity Text	Navego hasta el Nodo ptrRoot
Browse	Address	Child
	Node pointer...	ptrRoot
	Save pointer in...	ptrHijoX1
	Activity Text	Navego hasta el Nodo ptrHijo1
Expression Evaluator	List of variables with expressions defined	VeoContenidoPuntero:XV(ptrHijoX1, './*', 'XML') NombreTagActual:XV(ptrHijoX1, '.', 'NAME')
	Activity Text	veo dato de puntero
Values	Type	Get
	Operation	Name
	Node pointer...	PtrHijoX1
	Salve value in...	Nombre
	Activity Text	veo en que tag estoy

Define Variable	Type	New Variable
	Defined Variables	Nombre:Nombre
	Activity Text	Nombre
Define Variable	Type	New Variable
	Defined Variables	entroltera:"SI"
	Activity Text	entra itera SI
Browse	Address	Child
	Node pointer...	ptrHijoX1
	Save pointer in...	ptrHijoX2
	Activity Text	Navego hasta el Nodo ptrHijo2
Browse	Address	Next
	Node pointer...	ptrHijoX2
	Save pointer in...	ptrHijoX3
	Activity Text	Navego hasta el Nodo ptrHijo3
Expression Evaluator	List of variables with expressions defined	VeoContenidoPuntero:XV(ptrHijoX3, './*', 'XML') VeoContenidoPuntero:XV(ptrHijoX3, '.', 'XML')
	Activity Text	veo dato de puntero
Browse	Address	Child
	Node pointer...	ptrHijoX3
	Save pointer in...	ptrHijoX4
	Activity Text	Navego hasta el Nodo ptrHijo4
Browse	Address	Next
	Node pointer...	ptrHijoX4
	Save pointer in...	ptrHijoX5
	Activity Text	Navego hasta el Nodo ptrHijo5
Values	Type	Set
	Operation	Value
	Node pointer...	ptrhijox5
	Obtain value from...	vMailReceptor
	Activity Text	set vMailReceptor
Browse	Address	Next
	Node pointer...	ptrHijoX1
	Save pointer in...	ptrHijoX1
Expression Evaluator	List of variables with expressions defined	VeoContenidoPuntero:XV(ptrRoot, '.', 'XML')
	Activity Text	veo dato de puntero Root

Define Variable	Type	New Variable
	Defined Variables	entroltera:"NO"
	Activity Text	no entra iterar
Flow End	Activity Text	fin

Transiciones

Tipo	Index	Condición	Origen	Destino
Transición	1		inicio	Obtiene parámetros de NameSpaces que contiene el XML
Transición	1		Obtiene parámetros de NameSpaces que contiene el XML	cargo XML
Transición	2		cargo XML	Navego hasta el Nodo ptrRoot
Transición	1		Navego hasta el Nodo ptrRoot	Navego hasta el Nodo ptrHijo1
Transición	5		Navego hasta el Nodo ptrHijo1	Veo dato de puntero
Transición	4		Veo dato de puntero	Veo en que tag estoy
Transición	1	LEN(Nombre)=0	Veo en que tag estoy	fin
Transición	5	LEN(Nombre)<>0	Veo en que tag estoy	Nombre
Transición	3	Nombre<>'ns0:CFE'	Nombre	No entra itera
Transición	1	Nombre='ns0:CFE'	Nombre	Entra itera SI
Transición	4		No entra iterar	
Transición	5		Entra itera SI	Navego hasta el Nodo ptrHijo2
Transición	2		Navego hasta el Nodo ptrHijo2	Navego hasta el Nodo ptrHijo3
Transición	3		Navego hasta el Nodo ptrHijo3	Veo dato de puntero
Transición	1		Veo dato de puntero	Navego hasta el Nodo ptrHijo4
Transición	2		Navego hasta el Nodo ptrHijo4	Navego hasta el Nodo ptrHijo5
Transición	1		Navego hasta el Nodo	Set vMailReceptor

			ptrHijo4	
Transición	1		Set vMailReceptor	
Transición	5	ptrHijoX1=NULL		Veo dato de puntero Root
Transición	2	ptrHijoX1<>NULL		Veo en que tag estoy
Transición	1		Veo dato de puntero Root	fin

5 EjemploEjecutaSP

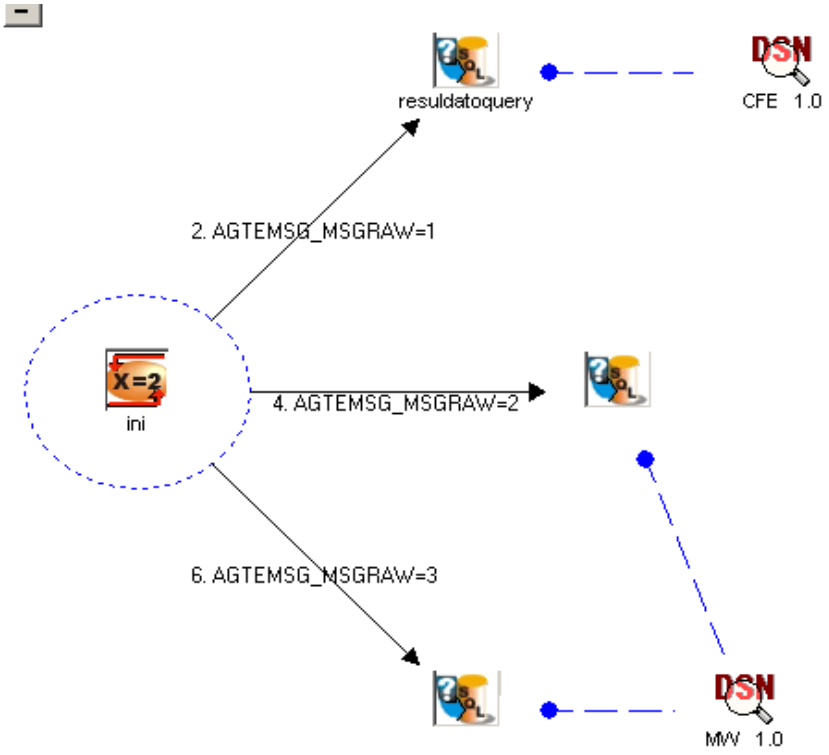


Ilustración 18: Flujo EjemploEjecutaSP 1.0

Actividades

Actividad	Atributos	Datos
Define Variable	Type	New Variable
	Defined Variables	ini:""
	Activity Text	ini
		NODO INICIAL
SQL	Type	Query
	Table	CAEData
	Type Query	SELECT
	With results	TRUE

	Result	PLAIN
	Store Result in Variable	Resuldatoquery
	Activity Text	resuldatoquery
SQL	Type	Query
	Table	Parametr0206
	Type Query	SELECT
	With results	TRUE
	Result	PLAIN
	Store Result in Variable	ResultadoQuery
SQL	Type	Store Procedure
	Table	Scantb
	Parameters	ini
	With results	TRUE
	Result	PLAIN
	Store Result in Variable	ResuldatoSP
DSN	Type	Use defined
	DSN name	CFE 1.0
	Description	DSN de uso Interno de MW
	User	sa
	Password	***
	Driver	ASPOOL
	Conection string	java:JBoss/CFE
	Server	192.168.15.8
	Database	CFE
	ReqPat	TRUE
DSN	Type	Use defined
	DSN name	MW 1.0
	Description	DSN de uso Interno de MW
	User	sa
	Password	***
	Driver	ASPOOL
	Conection string	java:JBoss/MMProDat
	Server	192.168.15.13
	Database	MMProdat
	ReqPat	TRUE

Transiciones

Tipo	Index	Condición	Origen	Destino
Transición	2	AGTEMSG_MSGRAW=1	ini	resultadoquery
Transición	4	AGTEMSG_MSGRAW=2	ini	(Query)
Transición	6	AGTEMSG_MSGRAW=3	ini	(Store Procedure)
Uso			resultadoquery	CFE 1.0
Uso			(Query)	MW 1.0
Uso			(Store Procedure)	MW 1.0

Se anexa las filas de la base de datos que almacenan cada uno de los flujos.

El ejemplo cuenta con el siguiente modelo de implementación:

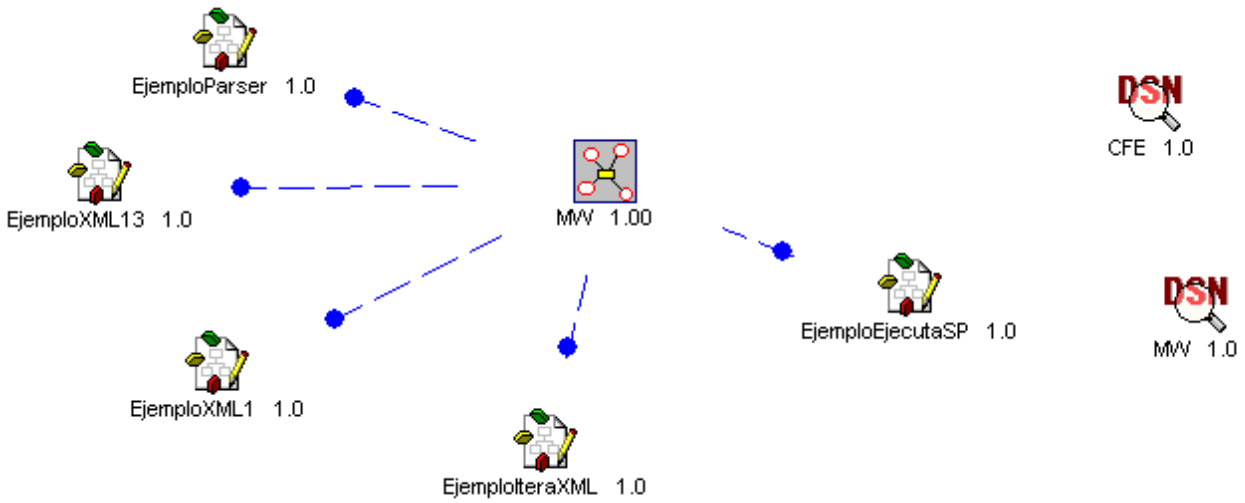


Ilustración 19: Implementación EjemploVarios 1.0