PONTIFICIA **UNIVERSIDAD CATÓLICA** DEL PERÚ

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA

Construcción y Pruebas de una herramienta de desarrollo de soluciones para Inteligencia de Negocios

Módulo de Extracción

Tesis para optar por el Título de Ingeniero Informático

Presentada por:

Luis Fernando Dall'Orto Gonzales del Valle

Raúl David Wu Yamashita

LIMA – PERÚ

2006

RESUMEN

La Tecnología de Información (TI) es, en la actualidad, un componente de gran importancia para cualquier organización. Sin embargo, son los datos y su adecuado manejo como transformaciones, búsqueda de patrones, y consolidaciones; lo que le da un carácter estratégico a la TI en la organización. En este contexto es donde aparecen conceptos como el de Inteligencia de Negocios, que apoyados en técnicas, estrategias, metodologías y herramientas buscan ofrecer información más adecuada para la toma de decisiones. Una solución de Inteligencia de Negocios puede, con gran posibilidad, cambiar el rumbo de una organización.

La implementación de soluciones de Inteligencia de Negocios se apoya necesariamente en un conjunto de herramientas informáticas que tienen que cubrir un ciclo de trabajo que comienza con la definición de un almacén de datos o Data Warehouse, la extracción y transformación de los datos desde diversas fuentes de información, y finalmente, la explotación de la información a través de diversos reportes tabulares y gráficos que permitan a la alta dirección de una organización la toma de decisiones.

El presente proyecto de tesis busca implementar el módulo de extracción de una herramienta básica para Soluciones de Inteligencia de Negocios que cubra todos los procesos del ciclo de trabajo. La arquitectura permitirá que una organización provea el servicio de Inteligencia de Negocios a múltiples organizaciones. Además, se toma en cuenta la escalabilidad del producto para soportar mayor número de fuentes de datos en futuras versiones.

FACULTAD
DE INGENIERÍA
Especialidad de Ingeniería Informática

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
90 AÑOS

TEMA DE TESIS PARA OPTAR EL TÍTULO DE INGENIERO INFORMÁTICO

TÍTULO: **"Construcción y pruebas de una herramienta de desarrollo de soluciones para inteligencia de negocios – Módulo de Extracción".**

ÁREA: Sistemas de Información

ASESOR: Ing. Luis Morales.
Ing. Carla Basurto
Ing. Luis Flores
Ing. Abraham Dávila

ESTUDIANTES	**CODIGO**	**NOMBRES**
	2000.0210.N.12	Raúl David Wu Yamashita
	2000.0354.7.12	Luis Fernando Dall'Orto Gonzales del Valle

TEMA N°: 207

FECHA: Lima, 04 de diciembre de 2006

DESCRIPCIÓN

La Tecnología de Información (TI) es, en la actualidad, un componente de gran importancia para cualquier organización, sin embargo, son los datos y su adecuado manejo como transformaciones, búsqueda de patrones, y consolidaciones; lo que le da un carácter estratégico a TI en la organización. En este contexto es donde aparecen conceptos, como el de Inteligencia de Negocios que apoyados en técnicas, estrategias, metodologías y herramientas buscan ofrecer información más adecuada para la toma de decisiones. Una decisión adecuada en una organización se traduce en mejoras significativas sea en dinero u otro beneficio. Una solución de Inteligencia de Negocios puede con gran posibilidad, cambiar el rumbo de una organización hacia escenarios más favorables y más beneficiosos.

En el mercado existen diversas herramientas que apoyan la implementación de Inteligencia de Negocios, pero son muy pocas las organizaciones que los utilizan en nuestro país, principalmente por el alto costo que implican la plataforma informática de este tipo de soluciones o porque las soluciones existentes cubren una parte y no todo el espectro de posibilidades.

La implementación de soluciones de Inteligencia de Negocios se apoya necesariamente en un conjunto de herramientas informáticas que tienen que cubrir un ciclo de trabajo que comienza con la extracción de los datos desde diversas fuentes de información como archivos de bases de datos de diferentes proveedores, hojas de cálculo, archivos planos, entre otros; continuando con un proceso de transformación de los mismos que puede ser tan simple como la homogeneización de los datos a conversiones complejas que se realizan en varias etapas; y, finalmente el análisis de información a través de diversos reportes tabulares y gráficos que permitan a la alta dirección de una organización tomar decisiones. Una solución básica de inteligencia de negocios utiliza grandes almacenes de datos (*data warehouse*) y herramientas que

ayuden al diseño de las transformaciones y la explotación de la información; cada cual con sus diversas complejidades.

El presente proyecto de tesis busca implementar una solución que permita el desarrollo de una herramienta básica para Soluciones de Inteligencia de Negocios. La arquitectura permitirá que una organización provea el servicio de Inteligencia de Negocios, en vez de pensar en un producto a ser vendido; por lo que será necesario definir una estrategia para el manejo de este servicio dentro de la solución que se va a implementar; asimismo esto implica que la solución debe ser escalable tanto a nivel de nuevo hardware a ser incorporado como inclusión de más componentes en la solución como parte de la evaluación misma del producto.

OBJETIVOS Y ALCANCE

Desarrollar una Herramienta Integral que permita desarrollar y explotar el Data Warehouse de una empresa o de varias de ellas a través del Internet.

La solución se construirá basada en la especificación J2EE e incorpora los siguientes módulos:

- Análisis Dimensional. Permite la administración del modelo estrella de un Data Warehouse.
- Extracción. Permite la administración y ejecución de los componentes que capturan los datos desde su origen hasta llevarlos al repositorio de Data Warehouse respectivo.
- Explotación: Permite la administración de los componentes que van a permitir visualizar la información del Data Warehouse.

Este proyecto de tesis cubrirá Construcción y Pruebas del módulo de Extracción, el cual cuenta con los siguientes sub-módulos:

- **Administración de Paquetes:** Permite la creación de paquetes para realizar la Extracción, Transformación y Carga de datos hacia el Data Warehouse.
- **Visualización de Componentes:** Permite visualizar de una manera amigable los componentes creados y las reglas de carga definidas por el usuario.
- **Extracción:** Permite realizar la conexión a los datos fuentes, el mapeo de los datos, la estandarización y el filtro de dichos datos, y la validación de dicha manipulación de datos.
- **Transformación y Carga:** Permite realizar la personalización de la manipulación de los datos extraídos y cargar dichos datos hacia el Data Warehouse. Dicha personalización también puede involucrar funciones predefinidas por el usuario.
- **Administración Jobs:** Permite la administración, control y supervisión de los procesos de ejecución de los diversos paquetes creados en el proceso de extracción.

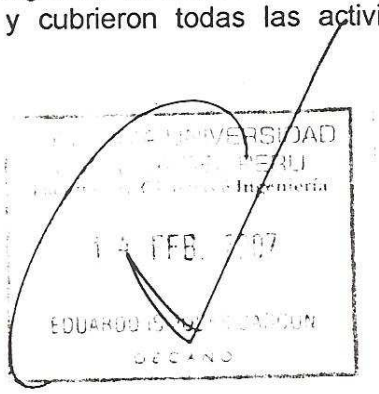
Adicionalmente, este módulo forma parte de la arquitectura integral de la herramienta de Inteligencia de Negocios.

Finalmente, se han desarrollado estos módulos de la siguiente manera:

SUBMÓDULOS	RESPONSABLES
Administración Paquetes	Raúl David Wu Yamashita
Visualización de Componentes	Raúl David Wu Yamashita
Extracción	Raúl David Wu Yamashita Luis Fernando Dall'Orto Gonzales del Valle
Transformación y Carga	Luis Fernando Dall'Orto Gonzales del Valle
Administración Jobs	Luis Fernando Dall'Orto Gonzales del Valle

Este documento de tesis se complementa con la tesis Análisis y Diseño de una herramienta de desarrollo de soluciones para inteligencia de negocios – Módulo de extracción. El equipo de trabajo fue uno sólo y cubrieron todas las actividades necesarias para la operatividad del producto.

Máximo: 100 páginas



"A mi mamá por estar desde cerca siempre apoyándome, a mi papá por estar desde lejos siempre velando por mí y a este grupo por ser mi mejor amigo."

-Luis Dall'orto

"A mis padres, Raúl y Kiomi, quienes me han dado todo en la vida. Gracias a ellos es que conocí y me interesé en lo que ahora es mi carrera."

-David Wu

De manera muy especial agradecemos, a la Pontificia Universidad Católica del Perú por la formación integral que nos brindó.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

En la actualidad, la Inteligencia de Negocios es dentro de las organizaciones una pieza clave para una adecuada y oportuna toma de decisiones. Una solución de Inteligencia de Negocios puede cambiar el rumbo de una organización hacia escenarios más favorables y más beneficiosos.

La herramienta propuesta es una herramienta integral que permite desarrollar y explotar el *Data Warehouse* de una empresa o de varias de ellas a través de Internet. Se desarrolló el módulo de Extracción, cuyo objetivo es el de permitir la administración y ejecución de los componentes que capturan los datos desde su origen hasta llevarlos al repositorio de *Data Warehouse* respectivo. Este módulo forma parte de la arquitectura integral de la herramienta de Inteligencia de Negocios, la cual consiste en tres módulos con funciones específicas. Estos módulos son: Análisis, Extracción y Explotación.

El módulo de Análisis es el que inicia el flujo de trabajo con la herramienta y define las estructuras de datos para el *Data Warehouse* en el que se colocarán los datos procesados.

El módulo de Extracción es el que se encarga de realizar los procesos de extracción, transformación y carga de datos, desde las fuentes de datos de la organización hacia el *Data Warehouse*.

El módulo de Explotación es el que se encarga de completar el flujo de trabajo, explotando los datos del *Data Warehouse* y mostrando reportes con la información requerida por el usuario final.

A lo largo del presente documento de tesis se describe el estudio realizado para la construcción y elaboración de pruebas del módulo de extracción de la herramienta propuesta.

Es importante mencionar que este trabajo de tesis tiene como punto de partida la tesis: “Análisis y Diseño de una herramienta de desarrollo de soluciones para inteligencia de negocios – Módulo de Extracción”, realizada por Pilar Infantas, César Mendoza y María Uribe, toma todo el análisis y diseño realizada en ésta para poder iniciar la construcción y pruebas de dicha herramienta.

En el primer capítulo, se presenta un análisis a la problemática existente y en base a la misma se plantea la solución. En este punto se considera importante comparar la herramienta creada con las ofrecidas actualmente en el mercado.

En el segundo capítulo, se presenta toda la información concerniente a la plataforma, el proceso de construcción y planeamiento, así como las pruebas que han sido completadas en el sistema.

Finalmente en el tercer capítulo, se presentan las observaciones, conclusiones y recomendaciones pertinentes.

1. MARCO TEÓRICO

1.1. Definición de la problemática

Actualmente la Tecnología de la Información se ha convertido en una herramienta clave en el proceso de desarrollo continuo dentro de las empresas. La competitividad del mercado y la globalización de la industria plantean un reto mayor dentro de toda organización pues se hace necesaria la innovación y el planeamiento estratégico que permita a la empresa trascender con un producto o servicio diferenciado.

Un componente indispensable en la toma de decisiones es el manejo eficaz y eficiente de los datos y la información que forma parte del conocimiento de la organización. En este contexto es donde aparecen conceptos como el de Inteligencia de Negocios que buscan ofrecer los resultados más adecuados para las organizaciones.

Una empresa que no sepa adaptar los nuevos conceptos de Inteligencia de Negocios en sus procesos de toma de decisión y manejo eficiente de la información, cae en el riesgo de quedar rezagada frente a un mercado cada vez más competitivo, que sabe aprovechar de buena forma los recursos tecnológicos y las tecnologías de información.

Uno de los activos valiosos que cada vez más se explotan en toda empresa es la información y los datos propios del negocio, almacenados durante el tiempo de vida de una organización. Sin importar el rubro en el cual se desarrolle, toda empresa genera durante sus operaciones diarias una cantidad de datos que se guardan en diferentes de formas (desde bases de datos especializadas de distintos proveedores hasta archivos en formatos de hoja de cálculo). El problema es que no siempre esta información está siendo explotada de forma inteligente. Lo óptimo es poder incorporar todos esos resultados en la toma de decisiones

futuras, y así formar un planeamiento más real. Todo negocio debe siempre estar aprendiendo de sí mismo.

El manejo de estos datos se sumerge ahora en toda una nueva forma de uso de la información, que requiere de un análisis no convencional. El hecho de implantar estas nuevas metodologías que permiten sacar provecho a los datos del negocio, es actualmente una labor complicada. Inicialmente, al no contar con una forma automatizada de lograr esa incorporación, eran necesarios muchos recursos, un análisis exhaustivo y migraciones complejas.

Actualmente, en el mercado existen algunas herramientas que apoyan la implementación de soluciones de Inteligencia de Negocios, pero son muy pocas las organizaciones que los utilizan en nuestro país, principalmente por el alto costo que implica implantar la plataforma informática de este tipo de soluciones o porque las existentes presentan limitaciones que no se adaptan al espectro de posibilidades que las empresas necesitan.

Para tener una mejor visión del alcance y lo que implica la elaboración de esta herramienta de Inteligencia de Negocios, presentamos la definición de los principales conceptos que están comprometidos con este tema.

a) Inteligencia de Negocios

Según Almeida [ALM 1999], se la puede definir como el uso de los datos recopilados con el fin de generar mejores decisiones de negocio, esto implica accesibilidad, análisis y revelar nuevas oportunidades.

Algunos conceptos de inteligencia de negocios no son nuevos, pero incluyen ahora la experiencia ganada desde los sistemas de información centrales hasta las aplicaciones de *data warehouse*.

La inteligencia de negocios busca proveer de un conjunto de tecnologías y productos para proporcionar a los usuarios la información que necesitan para resolver preguntas de negocios y tomar decisiones tácticas y estratégicas para el negocio.

b) Data Warehouse

Kimball [KIM 2002] lo define como la conglomeración de un conjunto de datos, los cuales se requieren almacenar y presentar de forma organizada. Los datos de la operación que se almacenan están estructurados de tal forma que puedan ser consultados con el fin de analizarlos.

Según Kimball, podemos distinguir algunos elementos básicos del *Data Warehouse*:

- Sistemas de Fuente Operacionales, la arquitectura en la cual se almacena los datos de la operación de la empresa.
- Área de arreglo de los Datos, donde se ejecutan la depuración, estandarización y

combinación de los datos de la fuente operacional. Además se almacena los datos y se ejecuta procesos de ordenamiento.

- Área de Presentación de los Datos, donde se realiza la carga de los datos que conforman un *Data Mart*, haciendo uso del modelamiento dimensional.
- Herramientas de Acceso a los Datos, que incluye aplicaciones para consulta específica, generadores de reportes, análisis de datos, modelamiento de proyecciones y estimación de resultados.

c) Data Mart

Kimball lo define como subconjunto lógico y físico del área de presentación de datos en un *Data Warehouse*. Originalmente, los *data mart* fueron definidos como un subconjunto altamente agregado de datos, normalmente usados para resolver preguntas específicas del negocio. Esta definición resultó no ser la más apropiada pues provocaba que los *data mart* sean inflexibles de combinarse con otros.

Esta primera concepción ha sido reemplazada, y el *data mart* es ahora definido como un conjunto flexible de datos, idealmente basado en los datos más atómicos posibles que se puedan extraer de una fuente operacional, y presentados en un modelo dimensional que es el que posee mayor capacidad de recuperación ante consultas inesperadas de los usuarios.

Los *data mart* pueden estar vinculados usando técnicas específicas al momento de conformar sus dimensiones. En este caso decimos que los *data mart* están conectados al bus del *data warehouse*.

En una forma simplificada, podemos decir que un *data mart* representa los datos de un proceso único de negocio.

d) Data Mining

Es una clase de consultas indirectas, normalmente sobre la data más atómica, que busca encontrar patrones inesperados en los datos. Los resultados más valiosos del *data mining* se obtienen agrupando, clasificando, estimando, prediciendo y encontrando acciones que ocurren juntas. Hay muchos tipos de herramientas que participan dentro del *data mining*.

La principal herramienta puede incluir árboles de decisiones, redes neuronales, herramientas de razonamiento basado en casos, herramientas de visualización, algoritmos genéticos y estadística clásica. Generalmente el *data mining* es un usuario del *data warehouse*.

e) Modelamiento Dimensional

Según Kimball, constituye una forma de modelamiento lógico de los datos orientado al rendimiento de las consultas y la facilidad de uso que se inicia de un conjunto de eventos de mediciones básicas. En el ámbito del modelo relacional de base de datos, una tabla *Fact* está acompañada de un conjunto de tablas de dimensión que le describen los atributos en el contexto de cada registro medidor. Por su estructura característica, al modelo dimensional se le conoce también como modelo de esquema estrella.

Un modelo dimensional busca con su diseño proveer claridad, predicción, escalabilidad y alta resistencia a una significativa cantidad de consultas, todo ello debido a su naturaleza simétrica. Dicho modelos dimensionales son la base de muchos componentes que agregan rendimiento en las bases de datos, incluyendo su considerable facilidad para poder vincular diferentes jerarquías de datos y aproximación por índices.

Los modelos dimensionales son la base para el desarrollo incremental y distribuido de los *data warehouse* a través del uso de dimensiones y Facts, además son la base lógica para los sistemas OLAP.

f) Fact Table / Tabla Fact

Kimball define que en un esquema tipo estrella (modelo dimensional), la tabla *Fact* representa la tabla central con medidores numéricos de rendimiento caracterizadas por una llave compuesta, cada una de ellas es una llave foránea en las tablas de dimensión.

g) Tabla de dimensión

Kimball define que en el modelo dimensional, una tabla de dimensión es una tabla con una única llave primaria y varias columnas de atributos descriptivos.

h) ETL (Extracción-transformación-carga)

Según Kimball el ETL es un conjunto de procesos por medio de los cuales los datos de la fuente operacional son preparados para colocarse en el *data warehouse*. El proceso primario de la preparación de los datos en el área de arreglo de datos de un *data warehouse*, antes de la presentación y la consulta.

El ETL consiste en extraer los datos de la fuente de origen, transformarla, cargarla e indexarla, asegurando su integridad, coherencia y disponibilidad en el destino.

i) Medida

Kimball lo define como una medición de rendimiento del negocio, típicamente numérico y aditivo, que es guardado en una Tabla Fact.

1.2. Objetivo principal

El objetivo principal de esta parte del proyecto es construir y realizar las pruebas de una herramienta ETL, la cual permita conectar el flujo de trabajo entre las otras dos partes del proyecto, el módulo de Análisis y el módulo de Explotación. El flujo de trabajo es el siguiente: luego de que se haya definido en el módulo de Análisis el esquema del *data mart* en el cual se almacenarán los datos históricos, se hace uso del módulo de Extracción para construir los flujos de transformación que cargarán los datos en el *data mart*. Finalmente, luego de haber ejecutado la carga de datos, mediante el módulo de Explotación se podrá obtener los reportes que mostrarán la información buscada.

Es importante notar que los datos de origen que serán cargados en el *data mart* pueden provenir de fuentes de datos heterogéneas. Esto hace que sea necesario pensar en una solución ETL que soporte el espectro de fuentes de datos existentes en el mercado en la actualidad, pero que también permita ser ampliada para no perder vigencia en el futuro.

Además, la herramienta a construir debe permitir trabajar de forma rápida y sencilla al usuario para que pueda definir el flujo de ETL sin complicaciones.

Finalmente, es importante que la herramienta trabaje de forma eficiente para asegurar que los recursos del servidor sean aprovechados correctamente.

1.3. Objetivos específicos

En base a lo expuesto en el punto 1.2 es que el equipo de trabajo encontró importante trabajar en base a los siguientes objetivos que dan forma al proyecto y posterior construcción de una solución integral.

a) Construir un componente compacto que permita una carga rápida de la herramienta en el explorador web. La carga inicial del componente no debe tomar más de 2 minutos usando una conexión de banda ancha (600kbps). La idea es evitar que el usuario tenga que esperar excesivamente para poder comenzar a trabajar con la herramienta.
b) Conseguir una integración total con los otros módulos de la herramienta de forma que el flujo de trabajo sea continuo y uniforme. El tiempo de adecuación para un usuario promedio, al uso básico de la herramienta, no debe superar los 15 minutos.
c) Implementar una solución escalable a futuro que permita agregar nuevos tipos de fuente de datos según sea necesario.

1.4. Solución propuesta: Características de la herramienta

La herramienta de Inteligencia de Negocios propuesta está apoyada no sólo en la idea de desarrollar una aplicación útil y funcional, sino en el resultado de una investigación y posterior análisis. De esta forma se logra definir un producto que se ajuste a la realidad empresarial, y que pueda calzar con las necesidades de innovación de la organización.

Se trata finalmente de una herramienta completa de extracción de datos, que se alimenta de distintos orígenes de datos, a los cuales les aplica funciones de transformación y limpieza para poder cargarlos a una base de datos destino alineado bajo el esquema dimensional de modelamiento, donde se podrá iniciar la explotación estratégica.

La aplicación cumple con un ciclo de trabajo que comienza con la extracción de los datos desde diversas fuentes de datos como archivos de bases de datos de diferentes proveedores, hojas de cálculo, archivos planos, entre otros. En base a éstos se ejecuta un proceso de transformación de los mismos mediante la aplicación de reglas predefinidas o personalizadas. Asimismo, la herramienta agrega control de los datos mediante filtros y estandarizaciones. Finalmente, los datos se cargarán en la fuente de destino elegida, donde se efectuarán las validaciones necesarias. En la Figura 1.1 se muestra gráficamente el flujo de procesos que ejecuta la herramienta.

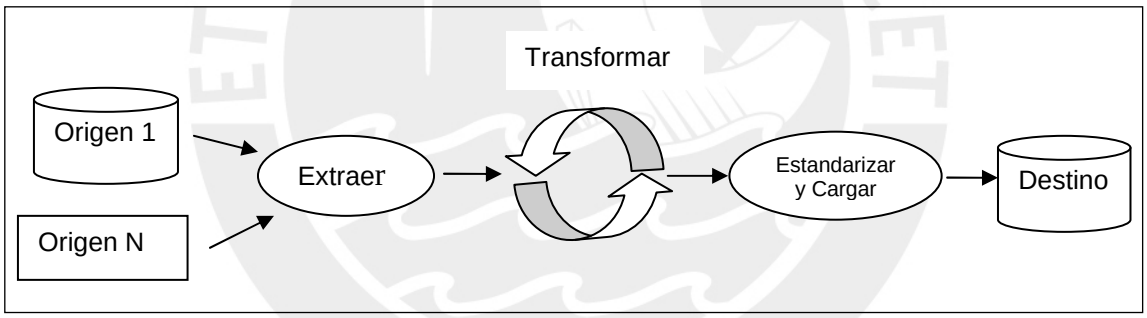


Figura 1.1 : Flujo de procesos de la herramienta

La aplicación es versátil en el sentido que no está sujeta a un estándar específico sino que da libertad a la empresa de poder aplicar sus propios análisis y estructura de orígenes de datos. Esta característica permite un manejo más independiente y personalizado de las funcionalidades. El modelo de ejecución y trabajo dentro de la aplicación está apoyado en criterios de simplicidad y facilidad de manipulación.

La herramienta hace uso de una arquitectura escalable tanto a nivel de *hardware* como en la inclusión de nuevos componentes que puedan formar parte de la solución. Esto representa una cualidad de alto potencial, pues deja la posibilidad y oportunidad de ser éste un proyecto abierto a la inclusión de nuevas formas tecnológicas de manejo de datos.

La herramienta de Extracción de Inteligencia de Negocios permite la creación de

proyectos de trabajo o *Jobs*, los cuales pueden guardarse y recuperarse durante el modelamiento de los flujos de extracción de datos. La ejecución de estos *Jobs* puede hacerse cuando uno se requiera, en forma manual, o también programarse para su ejecución posterior, mediante la calendarización de la ejecución. Esto permite ejecutar procesos de extracción complejos, en periodos donde el tráfico de las transacciones en las bases de datos de negocio sea bajo.

La herramienta puede alimentarse no sólo de bases de datos especializadas como por ejemplo las de Oracle 8, MySQL 5 o MS SQL Server 2000, sino que puede recibir también datos de archivos de texto, XML y archivos de hoja de cálculo Microsoft Excel [1].

Los algoritmos que forman parte del motor de filtrado, transformación y estandarización de datos desarrollan procesos ágiles pero consistentes. Todos los datos temporales que puedan generarse durante la ejecución de los flujos de extracción son trabajados de manera interna por el motor, sin necesidad de alterar las bases de datos de origen o destino.

1.5. Productos existentes

Dentro los productos existentes que ofrecen funcionalidades similares encontramos los siguientes:

- Sagent ETL

  Este sistema integrado extrae, transforma, mueve, distribuye y presenta la información clave para la toma de decisiones en la empresa. [SAG 2006]

- MicroStrategy

  Este *software* cubre los requerimientos de *reporting*, análisis y capacidades de envío en una sola plataforma, que permite, entre otras funciones, la gestión de seguridad centralizada, la administración, el desarrollo y la implementación de soluciones de *Business Intelligence* en forma centralizada. [MST 2006]

- Business Objects

  Esta herramienta permite a los usuarios el acceso, análisis y distribución de la información. Business Objects se caracteriza por ser una herramienta fácil de usar, segura, escalable y extensible. Incluye soluciones de consulta, generación de informes y análisis, un portal de BI con funcionalidad completa de *broadcasting* y potentes herramientas de administración. [BUS 2006]

[1] Oracle 8, MySQL 5, SQL Server y Microsoft Excel son productos y marcas registradas de Oracle Corp, MySQL AB y Microsoft Inc, respectivamente.

- Cognos

  Cognos ofrece todas las herramientas para Business Intelligence (BI): *reporting*, análisis, *scorecarding*, *dashboards*, administración de eventos de negocio así como la integración de los datos, en una arquitectura sola, probada. Fácil de integrar, desplegar y utilizar, Cognos entrega un ambiente simplificado de BI que facilita la adaptación del usuario, permite toma de decisiones. [COG 2006]

- Sunopsis

  Sunopsis integra en un solo producto las herramientas necesarias para abordar proyectos de Migración de Datos, Limpieza de datos, ETL en *Batch* o en línea, Replicación de Datos, Sincronización de Datos e Integración de Aplicaciones en línea. Sunopsis se caracteriza por un fácil entorno gráfico, flexibilidad y potencia. [SUN 2006]

- DataStage

  DataStage ofrece una solución ETL eficaz y altamente escalable. Presente tres características principales: provee de amplia conectividad para acceder fácil y rápidamente a cualquier sistema fuente o destino, ofrece herramientas avanzadas de desarrollo y mantenimiento que aceleran la implementación y simplifican la administración, también presenta una plataforma escalable que puede manejar fácilmente los volúmenes masivos de datos. [DSTG 2006]

Características	SAGENT ETL	COGNOS DECISIONSTREAM	BUSINESS OBJECTS INTELLIGENCE PLATAFORM	SUNOPSIS ETL v3	MICROSTRATEGY	DATASTAGE
1. Data Warehouse Escalable.	✓		✓	✓	✓	✓
2. Proceso de transformación en masa no registro por registro.	✓					
3. Reportes Web.	✓	✓				✓
4. Log de mensajes de error o advertencias.		✓	✓		✓	✓
5. Soporta sentencias SQL.		✓	✓			✓
6. Ejecución de scripts.			✓		✓	✓
7. Conexión con: JDBC, ODBC.			✓	✓		✓
8. Seguridad.	✓	✓	✓	✓	✓	✓
9. Asistente tipo wizard.	✓	✓	✓	✓	✓	✓
10. Administración de proyectos.	✓	✓		✓		
11. Plataforma independiente.				✓	✓	✓

Tabla 1.1: Comparación de productos existentes

2. CONSTRUCCIÓN DE LA PLATAFORMA DE INTEGRACIÓN

2.1. Consideraciones preliminares

Luego de terminado el análisis y diseño de la herramienta, se prosigue con la programación de las clases bases y sus principales métodos, y armar la estructura y relación entre estas, para así evitar clases con funciones diferentes. Es decir, una clase que es usada por más de un miembro del grupo, pero que para cada miembro tiene un significado diferente.

Esto puede causar algún conflicto en el futuro, mientras se programa, de manera que al dejar clara la función de cada clase se evita también la redundancia de líneas de código. Existen métodos comunes que podrían ser usados por otros módulos o serán desarrollados por éstos, por lo que el análisis de estos métodos en las etapas previas a la programación reduce el tiempo de desarrollo.

Luego de definir las clases que intervendrán en el proyecto, es necesario definir el *Framework Web* que se utilizará. Se presentaron dos opciones: *Struts* y JSF. Fueron seleccionados pues ambos cumplen con el patrón MVC (*Model View Controller*) que separa la lógica del control, la lógica del negocio y la presentación, el cual es un patrón que se adecuaba perfectamente al análisis y diseño que se había elaborado, y que además representa un patrón muy bien establecido en la actualidad para este tipo de desarrollos [SAL 2005] [REA 2005].

Se decidió usar el *Framework JSF* por las siguientes razones principales:

a) Facilita enormemente la realización de las páginas que interactuarán con el usuario, puesto que posee componentes mucho más sofisticados que *Struts*. Los componentes poseen muchas más propiedades que no se encuentran en *Struts*

y facilitan la codificación de las páginas con las que interactuará el usuario.

b) Los componentes *JSF* están hechos de manera que puedan ser manipulados fácilmente por el programador, es decir permite personalizar los componentes de una manera más intuitiva. Por ejemplo, las páginas muestran mensajes de error personalizados que dependen de lo ingresado por el usuario, todo esto se hace desde un control central sin necesidad de usar varias clases que enredarían el código y complicaría el mantenimiento de la herramienta.

c) Al ser *JSF* posterior a Struts, no posee las deficiencias de Struts y mejora muchos otros aspectos, por ejemplo la interacción con el usuario basado en eventos.

Se decidió hacer uso también de la internacionalización, por lo que se crearon dos archivos de recursos (*bundle*): uno para los títulos de las páginas y otro para el contenido de las páginas. De esta forma se puede adaptar el código de manera genérica y flexible, y en el momento que se tenga que extender el mercado al cual va dirigido la aplicación, se pueda agregar otros idiomas de manera más rápida y ordenada. Al hacer uso de la internacionalización también se redujo el tiempo que demora en cargar una página web en el navegador del usuario.

Para reducir aún más el tiempo que demora en cargar una página, se decidió colocar el código JavaScript en archivos separados de las páginas, con la ventaja adicional de que permite administrar el código de forma más óptima, pues no se mezcla JavaScript con lenguaje del *framework JSF*.

2.2. Metodología de desarrollo

Para el desarrollo del proyecto se aplicó una metodología basada en RUP, con iteraciones incrementales en la fase de construcción.

La metodología RUP divide en 4 fases el ciclo de vida del *software*: Concepción, Elaboración, Construcción y Transición.

A grandes rasgos, la fase de concepción se centra en definir el objetivo del proyecto y elaborar el modelo del negocio sobre el cual trabajará el producto. La fase de elaboración consiste en la planificación del proyecto, completar la definición de requerimientos del producto, definir la arquitectura del *software* y parte del análisis y diseño del *software*. La fase de construcción se centra en la implementación del *software*, acompañada de ajustes en el diseño. Finalmente, la fase de transición consiste en hacer la transición del producto construido a los nuevos usuarios. [KRU 2000]

La metodología de desarrollo por iteraciones consiste en tomar y agrupar casos de uso para ser desarrollados en orden, considerando las dependencias que puedan existir.

Dependiendo de la complejidad del producto se elige el número de iteraciones a realizar. En el caso del presente proyecto se realizaron 3 iteraciones en la fase de construcción, las cuales se centraron cada una en un grupo de funcionalidades.

La primera iteración tenía por objetivo completar las clases base que serían usadas para ofrecer las funcionalidades de la herramienta, implementar la creación de flujos de transformación e implementar la capa de almacenamiento de datos en archivos XML. Además, se completaría la estructura básica del applet con el cual interactúa el usuario.

Al finalizar la primera iteración, se podría crear *jobs*, crear paquetes, conectar a fuentes de datos, obtener las estructuras de la fuente de datos, formar el flujo de transformación, configurar el flujo de transformación a nivel de subcomponentes y programar la ejecución del *job*. Además todos los datos configurados en la herramienta se podrán almacenar en archivos XML para ser recuperados cuando sea necesario.

La segunda iteración tenía por objetivo implementar todos los algoritmos de ejecución de los subcomponentes del flujo de transformación. Los subcomponentes implementados son: *script*, filtro, transformación, estandarización, *lookup* y carga de datos. Además se implementó el sistema de ejecución automática de *jobs* programados, el manejo de archivos intermedios necesario para la ejecución de los subcomponentes y el ordenamiento automático de componentes para la ejecución, basado en precedencias.

Al finalizar la segunda iteración, se podría ejecutar un *job* programado automáticamente, ejecutando todos los subcomponentes que forman el flujo de transformación de datos.

La tercera iteración tenía por objetivo implementar las funcionalidades restantes del módulo de extracción, tales como: registro de *logs* de ejecución, verificación de *scripts* ingresados por el usuario, funciones definidas por el usuario, envío de correos de confirmación de ejecución de *jobs* y uso de parámetros en los *jobs*.

2.3. Estimación de esfuerzo para el proyecto

Como en todo proyecto de desarrollo, es importante contar con una estimación del tiempo que será necesario invertir en el desarrollo del mismo para poder hacer un planeamiento adecuado.

Las técnicas usadas para realizar la estimación del proyecto fueron las siguientes: Cocomo II [COC 2002] y Puntos de Función [IFP 1999], con estas se logró obtener la medida de meses-persona la cual indica cuánto esfuerzo se requiere por cada integrante en meses para poder cubrir el desarrollo de la herramienta.

Es importante resaltar que esta estimación se calculó para la segunda etapa del proyecto, la cual se desarrolló sobre plataforma web. Los conocimientos del equipo eran

mucho más sólidos así como la cohesión del mismo debido a la experiencia adquirida durante el desarrollo de la versión preliminar. Estos factores influyeron para que esa etapa se desarrollara con más rapidez y eficiencia, lo cual se corresponde con los resultados obtenidos con la estimación.

El documento de estimación de esfuerzo elaborado para el proyecto se encuentra en el anexo G.

2.4. Distribución de las tareas en el equipo

Como en cualquier proyecto, fue necesario distribuir las tareas del proyecto entre los miembros del equipo de desarrollo. Se buscó la mejor distribución de carga de trabajo considerando el tiempo disponible que cada integrante podría invertir en el proyecto. Además, para la distribución de funcionalidades para la programación se consideró la experiencia previa adquirida por miembros del equipo durante el desarrollo de la primera versión del producto durante el curso de Desarrollo de Programas 1.

El promedio de tiempo a dedicar al proyecto fue de 12 horas semanales, en algunos casos un poco más. Esto debido a que habiendo finalizado los estudios universitarios, ya no existía la carga académica que hacía más corto el tiempo disponible para avanzar el proyecto.

A continuación se presentan las tablas que muestran los temas que tocó cada miembro del equipo en el desarrollo del proyecto. Se consideran 4 categorías principales: Funcionalidades por caso de uso, otras funcionalidades programadas, documentación del proyecto y elaboración del documento de tesis.

Luis Dall'orto [Tesis de Construcción y Pruebas] ***(continúa)***			
Casos de uso	**Otras funcionalidades**	**Documentación**	**Documento de tesis**
Crear job	Bloqueo de paquetes para permitir modificación por un solo usuario	Documento de arquitectura	Capítulo 2 : Características del entorno, características de los usuarios
Configurar parámetros	Implementación de deshacer y rehacer acciones en el applet	Documento de pruebas	Capítulo 4: Construcción de la herramienta
Crear funciones definidas por el usuario	Implementación del control de parseo de funciones definidas por el usuario durante la ejecución.	Especificación de requisitos de software (para los casos de uso que le corresponden)	Capítulo 5: Observaciones, conclusiones y recomendaciones.
Programar job		Diagramas de secuencias (para los casos de uso que le corresponden)	

Luis Dall'orto [Tesis de Construcción y Pruebas] *(continuación)*			
Casos de uso	**Otras funcionalidades**	**Documentación**	**Documento de tesis**
Ejecutar job		Documento de algoritmos (para los casos de uso que le corresponden)	
Personalizar transformación usando objetos activos – Script			
Ejecutar job - Script			

Tabla 2.1: Tareas asignadas a Luis Dall'orto

Raúl Wu [Tesis de Construcción y Pruebas]			
Casos de uso	**Otras funcionalidades**	**Documentación**	**Documento de tesis**
Crear paquete	Implementación de la interfaz de usuario del applet	Documento de análisis	Capítulo 3 : Diseño de la herramienta.
Crear flujo de transformación	Implementación de objetos de dibujo del diagrama	Diagrama de clases	Capítulo 5: Observaciones, conclusiones y recomendaciones.
Configurar orden de ejecución de paquetes	Implementación de funcionalidad de zoom en el área de dibujo	Especificación de requisitos de software (para los casos de uso que le corresponden)	
Personalizar transformación usando objetos activos - Transformacion y Lookup	Implementación de control de envío de correos electrónicos	Diagramas de secuencias (para los casos de uso que le corresponden)	
Ejecutar job - Ejecutar paquetes y llamar a ejecución de subcomponentes	Implementación de control de parseo de parámetros en la ejecución de job	Documento de algoritmos (para los casos de uso que le corresponden)	
Ejecutar job - Ejecutar subcomponente Transformación y Lookup	Implementación de algoritmo para determinar orden de ejecución de componentes de un paquete		
Configurar correos de notificación	Implementación de grabación a xml de objetos de dibujo		
Visualizar log de ejecucion - Durante la ejecucion	Integración de las partes		

Tabla 2.2: Tareas asignadas a Raúl Wu

Pilar Infantas [Tesis de Análisis y Diseño]			
Casos de uso	**Otras funcionalidades**	**Documentación**	**Documento de tesis**
Configurar fuente de datos		Documento de estructura de XML	Capítulo 1 : Productos existentes
Conectar a fuente de datos		Especificación de requisitos de software (para los casos de uso que le corresponden)	Capítulo 5: Observaciones, conclusiones y recomendaciones.
Obtener estructura y objetos de fuente de datos		Diagramas de secuencias (para los casos de uso que le corresponden)	
Validar script mediante parser		Documento de algoritmos (para los casos de uso que le corresponden)	

Tabla 2.3: Tareas asignadas a Pilar Infantas

César Mendoza [Tesis de Análisis y Diseño]			
Casos de uso	**Otras funcionalidades**	**Documentación**	**Documento de tesis**
Personalizar transformación usando objetos activos – Estandarización	Implementación de grabación a xml de objetos lógicos	Especificación de requisitos de software (para los casos de uso que le corresponden)	Capítulo 1: Definición de la problemática, objetivos propuestos, características de la herramienta.
Ejecutar job – Subcomponente Estandarización	Implementación de manejo de archivos intermedios usados en la ejecución de job	Diagramas de secuencias (para los casos de uso que le corresponden)	Capítulo 5: Observaciones, conclusiones y recomendaciones.
Ejecutar job – Carga de datos al destino		Documento de algoritmos (para los casos de uso que le corresponden)	
Visualizar log de ejecución – Luego de la ejecución			

Tabla 2.4: Tareas asignadas a César Mendoza

María Uribe [Tesis de Análisis y Diseño]			
Casos de uso	**Otras funcionalidades**	**Documentación**	**Documento de tesis**
Configurar estructura de archivo plano	Mostrar contenido de archivos planos, según estructura configurada	Especificación de requisitos de software (para los casos de uso que le corresponden)	Capítulo 5: Observaciones, conclusiones y recomendaciones.
Personalizar transformación usando objetos activos – Filtro		Diagramas de secuencias (para los casos de uso que le corresponden)	
Ejecutar job - Ejecutar subcomponente Filtro		Documento de algoritmos (para los casos de uso que le corresponden)	

Tabla 2.5: Tareas asignadas a María Uribe

2.5. Consideraciones para Implementación de la herramienta

Para terminar el desarrollo de la herramienta y cumplir con las fechas establecidas, se eligió primero el mejor orden en el que se iban a programar los casos de uso de forma que todas las dependencias sean tomadas en cuenta y evitar retrasos. Se decidió hacer la programación de todas las funcionalidades en 4 iteraciones: en la primera iteración, se desarrollaría mayormente la interacción del usuario con la herramienta; en la segunda iteración, se desarrollaría principalmente las funcionalidades de ejecución de los diversos componentes de transformación que posee la herramienta (estandarización, filtro, transformación y *lookup*); en la tercera iteración, se implementaría principalmente las funcionalidades que complementaban la iteración dos, como son las funciones definidas por el usuario, los *logs* de ejecución, etc.; dejando al final en la cuarta iteración, el módulo de Seguridad del Sistema, pues para el módulo de Extracción constituye un extra y ninguna funcionalidad dependía directamente del módulo de Seguridad.

Durante el desarrollo de las iteraciones, se tomaron en cuenta algunas consideraciones más:

a) El componente 'proceso' de un flujo que es el eje principal de esta herramienta y que contiene la lógica de la transformación, se elaboró considerando la gran cantidad de datos que pueden tener actualmente las bases de datos transaccionales. Teniendo esto en cuenta, no se maneja toda la información en memoria puesto que esto podría recargar la memoria y hacer más lenta la ejecución. Además, según el análisis que se realice para completar un flujo este componente 'proceso' podría no filtrar los datos de entrada, de manera que el número de registros que maneja el proceso podría mantenerse elevado.
b) La ejecución de los flujos se puede hacer teniendo como origen diversas fuentes de datos, esto quiere decir que el componente 'proceso' debe verificar el tipo de motor de cada fuente, y de acuerdo a ello aplicar las operaciones y sentencias adecuadas. Se debe tener en cuenta que cada motor de base de datos maneja su metadata de manera distinta. Además, el origen de datos no sólo se limita al uso de bases de datos, sino que puede tratarse de archivos de texto o archivos de hoja de cálculo Excel, por lo que la lectura de datos también debe funcionar para estos casos.
c) En ciertos casos es necesario que la ejecución de un flujo se realice en diferentes fechas, es decir que puedan tener diversas programaciones y para cada programación las fuentes de datos origen puedan ser distintas. Con este fin, se implementó el que la herramienta permita el uso de parámetros y de acuerdo a los parámetros ingresados tomar la fuente de datos correcta en cada programación.
d) El analista puede necesitar estar al tanto del resultado de las ejecuciones que haya

programado o de los flujos de que es responsable, para esto la herramienta debe mandar notificaciones tanto en casos de error como de éxito de las ejecuciones. Esto se hará con el envío de correos electrónicos a la casilla que haya ingresado el analista.

e) El analista podría querer usar procedimientos almacenados que ya se encuentran en alguna base de datos para realizar la transformación. La herramienta toma en cuenta este requerimiento y permite al usuario poder configurar la ejecución de este procedimiento con el componente '*script'*, el cual no sólo permite escoger un determinado procedimiento existente, sino también permite al analista que tiene conocimientos del lenguaje SQL ingresar la sentencia por sí mismo. Los *scripts* pueden estar en el servidor origen o en el destino y ser ejecutados en cualquier parte del flujo de transformación.

f) El analista necesitará ver los *logs* de ejecución tanto en el momento que se estén ejecutando los *jobs*, o en cualquier momento luego de la ejecución. La herramienta por lo tanto guardará los *logs* de ejecución para poder ser visualizados por el analista en cualquier momento. Por supuesto, si el usuario se encuentra en uso de la herramienta y precisamente en ese instante se está ejecutando un *job*, si lo desea podrá seguir la ejecución actual de ese *job*.

2.6. Pruebas de desempeño

Se realizaron pruebas exhaustivas de las distintas partes que integran esta herramienta. Se buscó en las pruebas la correcta ejecución de la transformación en un flujo, para este fin se usó una base de datos Sql Server que hizo las veces de origen y destino de datos. Se probaron todas las posibles formas de transformación que se pueden realizar en la herramienta, usando todos los componentes posibles: filtro, estandarización, transformación y *lookup*. No sólo se hicieron pruebas usando estos componentes por separado sino también con combinaciones de los mismos. En algunos casos la transformación de un flujo puede hacer uso de archivos intermedios, se hicieron pruebas también en ese aspecto tanto para un número pequeño de registros como para una gran cantidad de datos. En la tabla 2.11 se muestran los resultados.

También se realizaron pruebas de la ejecución de los flujos en la fecha establecida por las programaciones, probando todos los tipos de programación que se pueden hacer: diaria, semanal, mensual y anual. Además se probó la ejecución de las programaciones en forma simultánea, es decir cuando los flujos han sido programados para una misma fecha.

Todo lo mencionado anteriormente se hizo planteando los casos de prueba de acuerdo a la funcionalidad que es vista por cada usuario. Las pruebas se derivaron usando "clase equivalente" y los casos de uso. La derivación de casos de prueba usando clase equivalente se basa en definir las distintas entradas para una funcionalidad del programa, esta entrada según el valor que toma en la prueba puede ser una clase válida que es cuando el valor es correcto para el sistema, o una clase inválida que representa una entrada incorrecta para el sistema. De esta forma se plantean para cada funcionalidad un conjunto de combinaciones de clases

válidas e inválidas las cuales bastarán para probar la funcionalidad.

2.6.1. Casos de prueba

A continuación se presentan algunos casos de pruebas planteados divididos por tipo de funcionalidad. La lista completa se encuentra en el anexo H.

Configurar y Conectar Fuente de Datos	
ID Prueba:	BE01
Objetivo :	Configurar una fuente de datos mediante el ingreso de datos válidos para una base de datos SQL Server.
Precondición:	El usuario ha ingresado al sistema. El usuario ha abierto un paquete.
Clases :	JSFServlet, BEConfigurarBDBean, BEGestorFuenteDatos, BEControlFuenteDatos, BEFuenteDatos, IdFactory, Empresa, Proyecto, BEPaquete.
Proceso:	1. En el menú seleccione la opción “Paquete” y “Agregar Fuente de Datos”. 2. Seleccione “Conexión a Base de datos”. 3. Ingrese en el campo “Nombre” el valor “Fuente de prueba”. 4. Ingrese en el campo “Descripción” el valor “Fuente creada para caso de prueba”. 5. Elija del combobox de “Tipo” el valor “SQL Server” de la lista. 6. Ingrese en el campo “Conexión” el valor “Inti”. 7. Ingrese en el campo “Base Datos” el valor “Bipucp”. 8. Ingrese en el campo “Usuario” el valor “biextrausr”. 9. Ingrese en el campo “Contraseña” el valor “2006biusr”. 10. Seleccione la opción “Grabar”.
Resultado Esperado:	Se regresa al área de trabajo inicial y se muestra la estructura de la Fuente de Datos.

Tabla 2.6: Caso de prueba para funcionalidad tipo comunicación

Crear flujo de transformación *(continúa)*	
ID Prueba:	BE13
Objetivo :	Crear el flujo de transformación mediante la selección de datos válidos.
Precondición:	El usuario ha ingresado al sistema. El usuario ha abierto un job. Las fuentes de datos a utilizar han sido configuradas correctamente.
Clases :	AppletExtraccion, DiagramaDibujo, BEControlDibXML, DiagramaDibujoBean, AppletServlet, BEControlPaquete, Empresa, Proyecto, BEJob, BEPaquete,
Proceso:	1. Seleccione del árbol de navegación de objetos el paquete con nombre “Paquete Prueba”. 2. Elija de la lista de tablas de la Fuente de Datos con nombre “Fuente Prueba” la tabla “Persona”. 3. Click derecho sobre la tabla y seleccione la opción “Agregar tabla al flujo”. 4. Elija de la lista de tablas de la Fuente de Datos con nombre “Fuente Destino - DW” la tabla “Fact_Persona”. 5. Click derecho sobre la tabla y seleccione la opción “Agregar tabla al flujo”. 6. Presione el botón que corresponde al componente “Proceso” 7. Presione en el punto donde se insertará el componente.

Crear flujo de transformación *(continuación)*	
Proceso:	8. Presione el botón que corresponde al componente “Conector” 9. Seleccione la tabla “Persona” para indicar el origen. 10. Seleccione el componente “Proceso” para indicar el destino. 11. Presione el botón que corresponde al componente “Conector” 12. Seleccione el componente “Proceso” para indicar el origen. 13. Seleccione la tabla “Fact_Persona” para indicar el destino. 14. Seleccione la opción “Grabar”.
Resultado Esperado:	Se graba exitosamente el flujo y se muestra el área de trabajo sin ningún mensaje de error.

Tabla 2.7: Caso de prueba para funcionalidad tipo extracción

Programar Jobs	
ID Prueba:	BE21
Objetivo :	Crear una Programación mediante el ingreso de datos válidos para una programación Diaria.
Precondición:	El usuario ha ingresado al sistema. El usuario ha abierto un proyecto.
Clases :	JSFServlet, BEProgramacionBean, BEProgramacion, BEControlProgramacion, BEJob, Empresa, Proyecto, IdFactory.
Proceso:	1. En el menú, seleccione la opción “Programar Jobs” y “Crear”. 2. Ingrese en el campo “Nombre” el valor “Programacion Prueba”. 3. Ingrese en el campo “Descripción” el valor “Programacion para el caso de prueba”. 4. Elija del combobox de “Job” el valor “JobPrueba”.de la lista. 5. Ingrese en el campo “Hora de ejecución” el valor “20:00”. 6. Elija del combobox de “Periodicidad” el valor “Diaria” de la lista. 7. Seleccione los días “Lunes” y “Miércoles”. 8. Seleccione la opción “Grabar”.
Resultado Esperado:	Se guarda exitosamente la programación y se muestra en el formulario de “Administración de Programaciones” la nueva programación creada.

Tabla 2.8: Caso de prueba para funcionalidad tipo ejecución

2.6.2. Reportes de pruebas

Los escenarios para los casos de prueba realizados, en los cuales se prueba la correcta ejecución de una funcionalidad, se detallan a continuación.

En cada escenario se indica el caso de prueba que se está ejecutando, los datos específicos del entorno de ejecución, el resultado que se obtuvo y la forma de verificación del resultado.

Para las pruebas que requieren procesamiento de datos, se detalla la estructura de las tablas o archivos con los que se trabaja. Se indican los campos que la componen y el tipo de dato al que pertenecen.

2.6.2.1. Pruebas de casos de uso

En la tabla 2.9 se detalla una lista de reportes de prueba para algunos casos de uso definidos para la herramienta. La lista completa se encuentra en el anexo I.

Caso de prueba	Escenario de prueba	Resultado obtenido	Verificación del resultado
BE01- Configurar y conectar fuente de datos	Existe una base de datos SQL Server, cuyos datos de conexión son: Conexion:inti, BaseDatos:Bipucp Usuario:biextrausr Contraseña:2006biusr	El área de trabajo contiene la estructura de la base de datos Bipucp	Se observa la estructura en el árbol de navegación de objetos.
BE12- Agregar Archivo y configurar Estructura Tipo TXT	Existe un archivo TXT válido en la carpeta del usuario, con los campos siguientes: id: Integer nombre:String apellido:String edad:Integer	La configuración del archivo TXT se configuró correctamente.	Se observa en la página de configurar estructura la información guardada.
BE18- Crear Job	Se ingresan los siguientes datos para el nuevo job: nombre: “Job Prueba” descripcion:”Job creado para caso de prueba”.	El job se guardó exitosamente.	Se observa en el formulario de Administración de jobs el nuevo job creado.
BE43- Personalizar transformación usando objetos activos – Transformación de Datos.	Se usa las siguientes sentencias de transformación: idPersona= “Persona.idPersona” Apellidos= “Persona.apellidoPaterno”+ ' '+”Persona.apellidoMaterno”	La transformación se guardó exitosamente.	Se puede abrir el formulario de Transformación y observar la información guardada.
BE59- Crear funciones definidas por el usuario tipo:Regla	Se agregan las siguientes reglas: Entrada: “M”,Salida:”1” Entrada: “F”, Salida:”0” Entrada:”Masc”, Salida:”1”	Se guardó correctamente la función	Se puede abrir la función nuevamente y observar la información guardada.
BE65- Programar Jobs	Se ingresan los siguientes datos para la programación: Job:”Job Prueba” Hora:”20:00” Periodicidad:”Diaria” Seleccionar Lunes y Miercoles	Se guardó correctamente la programación	Se observa en el formulario de Administración de programaciones la nueva programación.
BE79- Ejecutar Job Origen:TXT Destino: TXT	El archivo TXT origen contiene los siguientes campos: id:Integer, nombre: String apellidoPat:String apellidoMat:String edad:Integer telefono:String El archivo TXT destino contiene los siguientes campos: id: Integer, nombres: String, telefono: String	El proceso de ejecución se realizó correctamente.	Se observa en pantalla las tareas que realiza el job. El archivo destino TXT contiene la data correcta.

Caso de prueba	Escenario de prueba	Resultado obtenido	Verificación del resultado
BE84- Ejecutar Job Origen: Tabla Destino:Tabla	La tabla origen contiene los siguientes campos: id:Integer, nombre: String apellidoPat:String apellidoMat:String edad:Integer telefono:String La tabla destino contiene los siguientes campos: id: Integer, nombres: String, telefono: String	El proceso de ejecución se realizó correctamente.	Se observa en pantalla las tareas que realiza el job. La tabla destino contiene la data correcta.
BE87- Ejecutar Job Origen: Tabla Origen: TXT Destino: TXT	La tabla origen contiene los siguientes campos: id:Integer, nombre: String apellidoPat:String apellidoMat:String edad:Integer telefono:String El archivo TXT origen contiene los siguientes campos: id:Integer compras:double ventas:double El archivo TXT destino contiene los siguientes campos: id: Integer, nombres: String, telefono: String compras: Double	El proceso de ejecución se realizó correctamente.	Se observa en pantalla las tareas que realiza el job. El archivo destino TXT contiene la data correcta.

Tabla 2.9: Reporte de pruebas para casos de uso

2.6.2.2. Pruebas de otras funcionalidades

En la tabla 2.10 se detalla una lista de reportes de prueba para los requerimientos no funcionales de la herramienta que no encajan directamente en un caso de uso. El listado completo se encuentra en el Anexo I.

Funcionalidad	Escenario de prueba	Resultado obtenido	Verificación del resultado
Ejecución de jobs automáticos	El sistema tiene configurado un job para ejecutarse el Lunes a las 20:00.	El job se ejecutó a la hora programada.	El log muestra las tareas que realizó el job para ejecutarse.
Deshacer y rehacer acciones	El usuario se encuentra en proceso de creación de un flujo. El usuario se equivoca en un conector. El usuario selecciona Deshacer	La última modificación se deshace correctamente.	En el área de trabajo se observa la última modificación deshacida.

Funcionalidad	Escenario de prueba	Resultado obtenido	Verificación del resultado
Acercar y Alejar Zoom	El usuario se encuentra en proceso de creación de un flujo. El usuario selecciona Acercar.	El dibujo del flujo cambia de tamaño correctamente.	Se observa en el área de trabajo el flujo con mayor tamaño.
Drag and Drop del árbol al área de trabajo	El usuario se encuentra en proceso de creación de un flujo. El usuario presiona el mouse sobre un objeto pasivo del árbol y lo suelta en el área de trabajo	El dibujo del flujo se actualiza correctamente.	Se observa en el área de trabajo el flujo conteniendo el nuevo objeto pasivo.
Concurrencia - Bloqueo	Un usuario “A” abre un flujo “F”. Otro usuario “B” abre el mismo flujo “F”.	El usuario “B” solo tiene permisos de lectura sobre el flujo.	Se observa que en el momento que el usuario “B” abre el flujo “F” aparece una ventana que indica el permiso de lectura.
Concurrencia – Desbloqueo al cerrar ventana	Un usuario “A” abre un flujo “F”. Otro usuario “B” abre el mismo flujo “F”. El usuario “A” cierra la ventana del navegador web.	El flujo es liberado por el usuario “A” y puede ser bloqueado por otro usuario.	Se observa que el usuario “B” al intentar abrir nuevamente el flujo no le aparece ninguna ventana de permiso de lectura.
Concurrencia – Desbloqueo al cerrar sesión	Un usuario “A” abre un flujo “F”. Otro usuario “B” abre el mismo flujo “F”. El usuario “A” cierra la sesión de la aplicación	El flujo es liberado por el usuario “A” y puede ser bloqueado por otro usuario.	Se observa que el usuario “B” al intentar abrir nuevamente el flujo no le aparece ninguna ventana de permiso de lectura.
Concurrencia – Desbloqueo al perder conexión	Un usuario “A” abre un flujo “F”. Otro usuario “B” abre el mismo flujo “F”. El usuario “A” no utiliza la aplicación por 30 minutos.	El usuario “A” pierde la conexión al servidor y el flujo es liberado pudiendo ser bloqueado por otro usuario.	Se observa que el usuario “B” al intentar abrir nuevamente el flujo no le aparece ninguna ventana de permiso de lectura.
Compatibilidad Firefox - popups	El usuario se encuentra en proceso de creación de un flujo, usando un navegador Firefox. El usuario selecciona “Configurar Filtro”	El popup aparece correctamente en el navegador firefox del usuario.	Se observa la ventana popup en la pantalla del usuario.
Compatibilidad Firefox – correcta diagramación de páginas	El usuario se encuentra en proceso de creación de un flujo, usando un navegador Firefox. El usuario selecciona “Configurar Transformación”	Se abre la ventana popup, y los componentes de dicha página son dibujados correctamente.	Se observa la ventana popup con los componentes ordenados.

Tabla 2.10: Reporte de pruebas para otras funcionalidades

2.6.3. Pruebas de rendimiento

Un punto crucial de la aplicación es el trabajo que realiza con los registros que obtiene de las bases de datos. Por esta razón, es necesario medir los tiempos que toman los procesos de ejecución para diversos volúmenes de datos.

En esta etapa del proyecto las pruebas se realizaron en una computadora de escritorio y con un volumen de datos relativamente pequeño. Sin embargo, con los resultados de estas pruebas se puede estimar el comportamiento de la herramienta para volúmenes de datos mayores y en un equipo de mayor capacidad de procesamiento, como un servidor.

El escenario en el que se realizaron las pruebas de rendimiento es el siguiente:

Tabla origen “Persona” con la siguiente estructura:

- id : integer
- nombre: varchar(25)
- apellido: varchar(25)
- edad: integer

Tabla destino “Persona_D” con la siguiente estructura:

- id: integer
- nombreCompleto: varchar(50)
- edad: integer

El proceso de transformación se configuró de la siguiente forma:

Transformación de datos:

- Persona_D.id = Persona.id
- Persona_D.nombreCompleto = Persona.nombre + ‘ ‘ + Persona.apellido
- Persona_D.edad = Persona.edad

Estandarización de datos:

- Persona_D.nombreCompleto = minusculas(Persona_D.nombreCompleto)

Configuración de *hardware*:

- Procesador Pentium 4 de 2.4GHz
- 1 GB de memoria RAM
- Disco duro de 40GB y 5400 rpm

Motor de datos usado:

- SQL Server 2000

Se resume en la tabla 2.11 los resultados de las pruebas de rendimiento realizadas para volúmenes de datos variados:

Cantidad de Registros	Tiempo de ejecución (segundos)
1 000	4.55
5 000	12.77
10 000	23.59
20 000	44.64
50 000	115.38

Tabla 2.11: Tiempos de procesamiento por volumen de datos

Para poder apreciar mejor el comportamiento de la herramienta se muestra a continuación en la figura 2.1 la gráfica de registros versus tiempo de procesamiento.

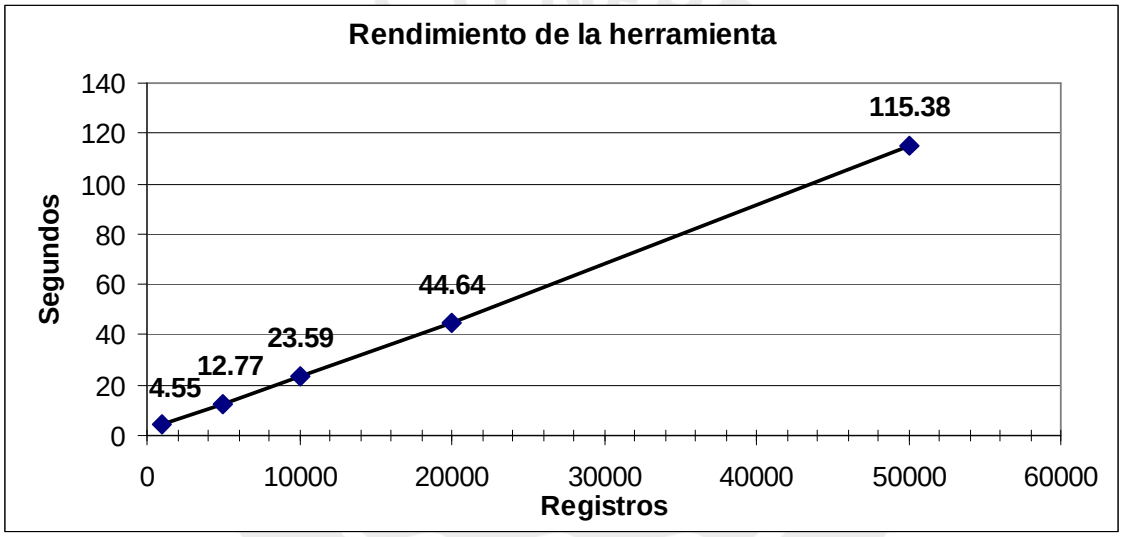


Figura 2.1: Gráfico de tiempo de procesamiento versus número de registros

Como se puede ver en el gráfico, el crecimiento es lineal para los tiempos, con una pendiente promedio de 383 registros/segundo.

Este resultado está dentro de lo esperado para la herramienta, pues no se obtuvieron resultados erráticos o incoherentes sino por el contrario una tasa constante de procesamiento de datos.

2.7. Ejemplo de funcionamiento de la herramienta

En esta sección se mostrará algunos ejemplos de uso de la herramienta para que el lector tenga una mejor apreciación de la capacidad y el potencial que tiene.

Primero se presentará un flujo de transformación sencillo que permita mostrar las funcionalidades básicas de la herramienta, como son: conectar a una fuente de datos, mostrar

contenido de tablas, formar flujo de transformación y configurar subcomponentes.

Luego se mostrará un flujo de transformación más complejo que involucre varias tablas de una fuente de datos y que permita cargar una serie de dimensiones y *facts* en el *data warehouse* destino.

La base de datos transaccional a conectarse cuenta con datos de clientes, productos y tiendas de la empresa.

Lo que se hará en el primer ejemplo es una transformación simple de datos, pasando datos de la tabla de clientes del origen a una tabla de clientes modificada en el destino. Se realizará un filtrado, transformación y estandarización de los datos.

- Se filtrarán los clientes con edades entre 20 y 25 años.
- Se combinarán en el campo destino 'nombreCompleto' los campos origen 'nombre' y 'apellidos'.
- Se estandarizará los nombres para que estén en mayúsculas.

Se muestan a continuación las pantallas de la herramienta durante el proceso de conexión a la fuente de datos (figura 2.2), creación del flujo de transformación (figura 2.3), vista de datos de la tabla origen (figura 2.4) y configuración de los subcomponentes: filtro (figura 2.5), transformación (figura 2.6) y estandarización. (figura 2.7)

Finalmente se muestra el resultado de la ejecución en la tabla destino. (figura 2.8)

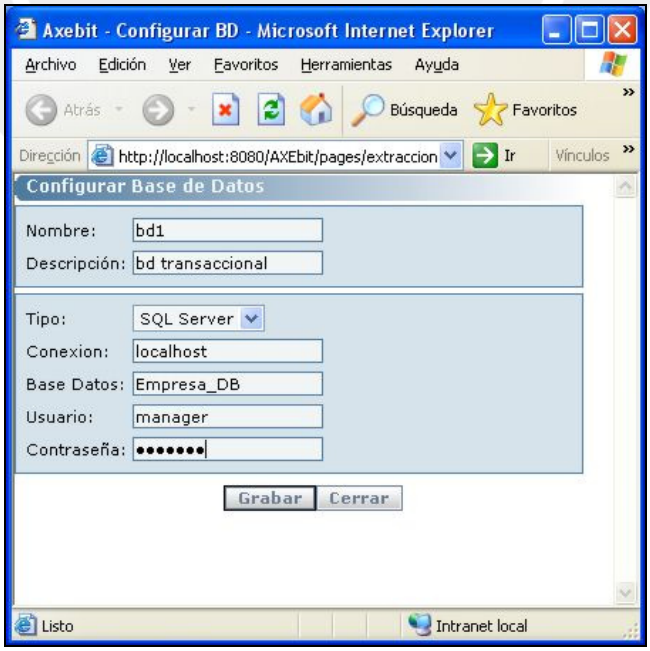


Figura 2.2: Ejemplo 1 – Configurar conexión a fuente de datos

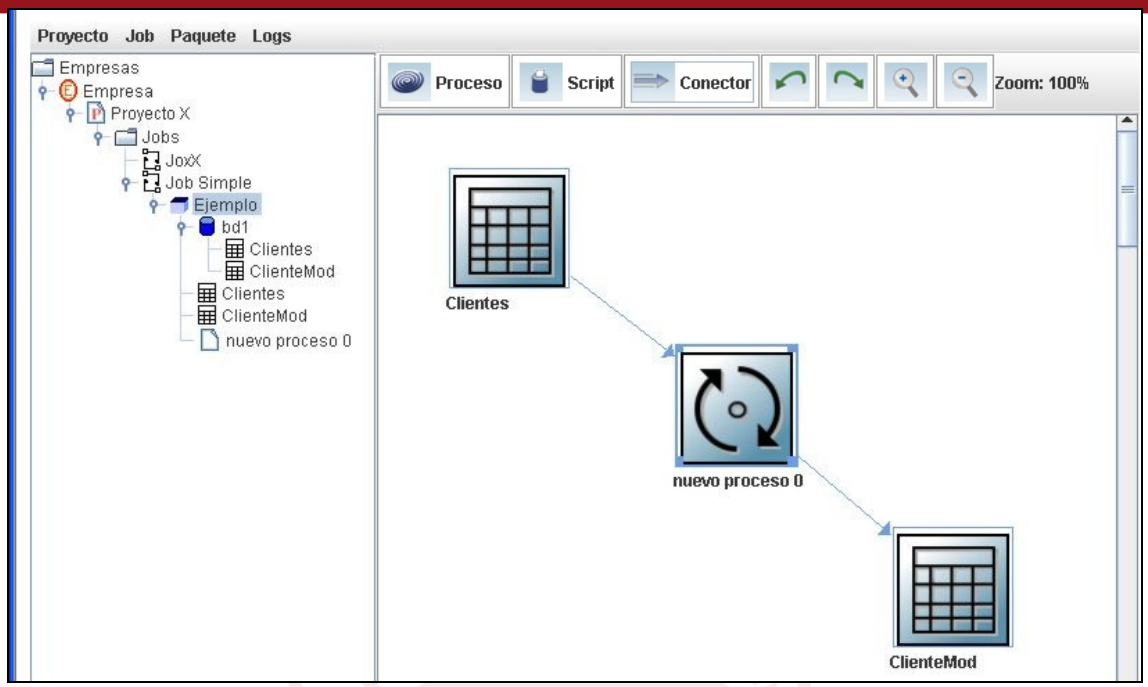


Figura 2.3: Ejemplo 1 – Formar flujo de transformación

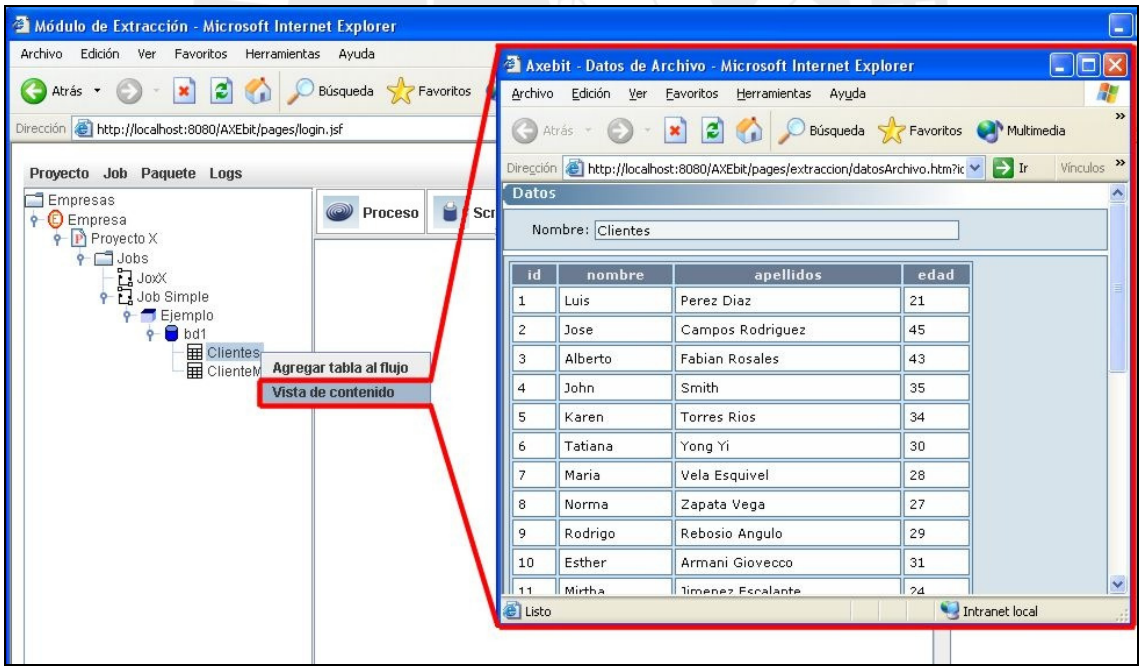


Figura 2.4: Ejemplo 1 – Vista de contenido de la tabla origen

Figura 2.5: Ejemplo 1 – Configurar filtro

Figura 2.6: Ejemplo 1 – Configurar transformación

Figura 2.7: Ejemplo 1 – Configurar estandarización

Figura 2.8: Ejemplo 1 – Vista de datos en la tabla destino

En el segundo ejemplo se crearán 2 paquetes para el *job* de carga del *data warehouse*. El primer paquete contendrá los flujos para realizar la carga de las tablas dimensión correspondiente a clientes, productos y tiendas. El segundo paquete contendrá el flujo de transformación que carga la tabla *fact* ventas con los datos sumarizados, agrupados por cliente, producto y tienda.

Se muestra a continuación las pantallas que muestran el proceso de construcción del flujo de transformación para este *job*.

En el primer paquete del *job* complejo se agregan 2 fuentes de datos: el origen, llamado 'bd1', y que contiene los datos de las tablas de la base de datos transaccional de la empresa; y el destino, llamado 'dw', que corresponde al *data warehouse* de la empresa. Se muestra en la figura 2.9 las tablas agregadas al paquete por cada base de datos.

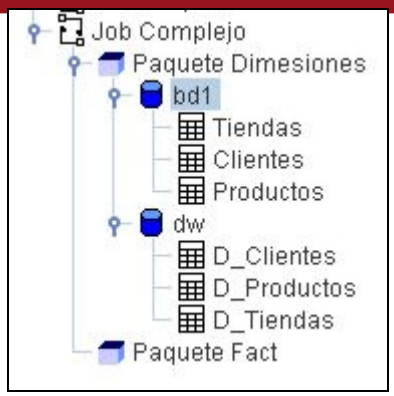


Figura 2.9: Ejemplo 2 – Bases de datos origen y destino

Luego se forman los flujos de transformación para cargar las tablas dimensión del *data warehouse*, basándose en los datos de las tablas del origen. Se crea un flujo por cada tabla a cargar, como se muestra en la figura 2.10.

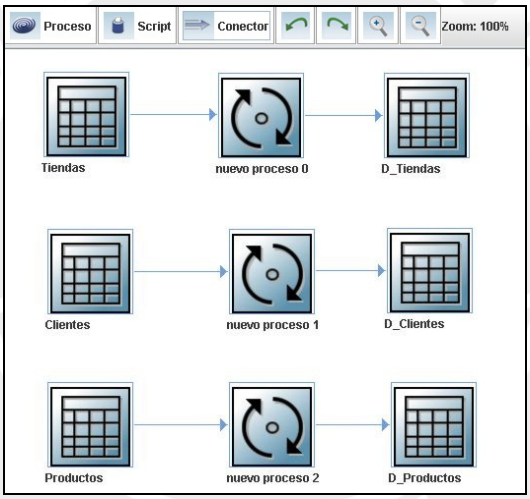


Figura 2.10: Ejemplo 2 – Flujos de carga para dimensiones

En cada proceso se configuran los subcomponentes para hacer la carga de datos desde el origen transaccional al destino en el *data warehouse*. Las transformaciones son simples en este caso, similares al primer ejemplo mostrado líneas arriba. Para cargar las dimensiones no se hará uso de filtros o de estandarización, sólo se utilizará la transformación. Se muestra en la figura 2.11 la configuración de los subcomponentes.

Figura 2.11: Ejemplo 2 – Configuración de transformaciones para dimensiones

En el segundo paquete del *job* complejo se realizará la carga de la tabla *fact*, esto se debe hacer luego de haber cargado las dimensiones por lo que se debe configurar el orden de ejecución de los paquetes en el *job*. En la figura 2.12 se muestra la pantalla de configuración de orden de paquetes.

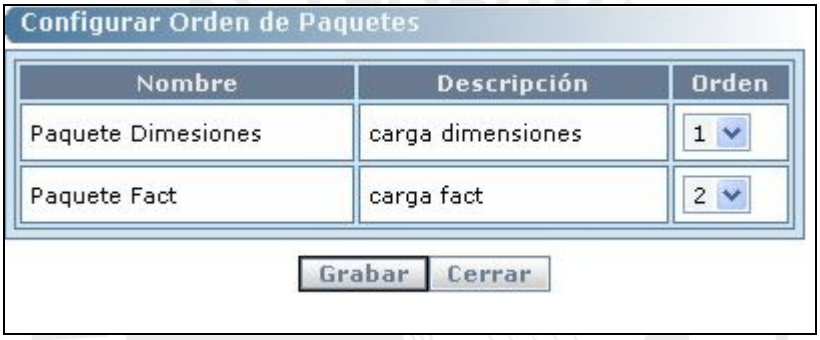


Figura 2.12: Ejemplo 2 – Orden de ejecución de paquetes

Luego se configuran las fuentes de datos para este paquete, agregando las tablas con los datos a ser cargados en la tabla *fact* como se muestra en la figura 2.13.

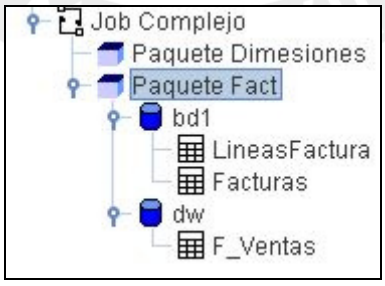


Figura 2.13: Ejemplo 2 – Tablas en el paquete fact

A continuación se forma el flujo de transformación para cargar los datos necesarios en la tabla *fact* F_Ventas. En la figura 2.14 se muestra la estructura de esta tabla.

Figura 2.14: Ejemplo 2 – Estructura de la tabla fact F_Ventas

En el flujo formado (figura 2.15) se configura la transformación para obtener de las tablas del origen los datos necesarios: idCliente, idProducto, idTienda, valorVenta.

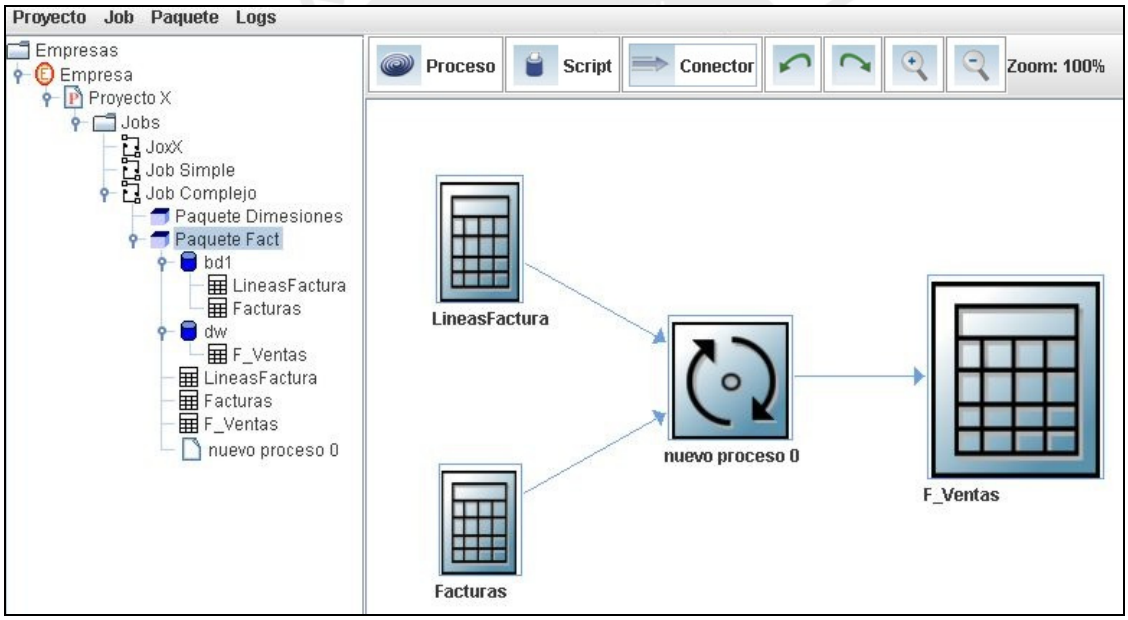


Figura 2.15: Ejemplo 2 – Flujo de transformación para cargar fact

En la configuración de la transformación (figura 2.16) se hace uso de la función sum() para acumular el valor de las ventas de acuerdo a los valores de las dimensiones. Con esto se busca saber cuál es el valor de venta de un producto específico por un cliente y por tienda.

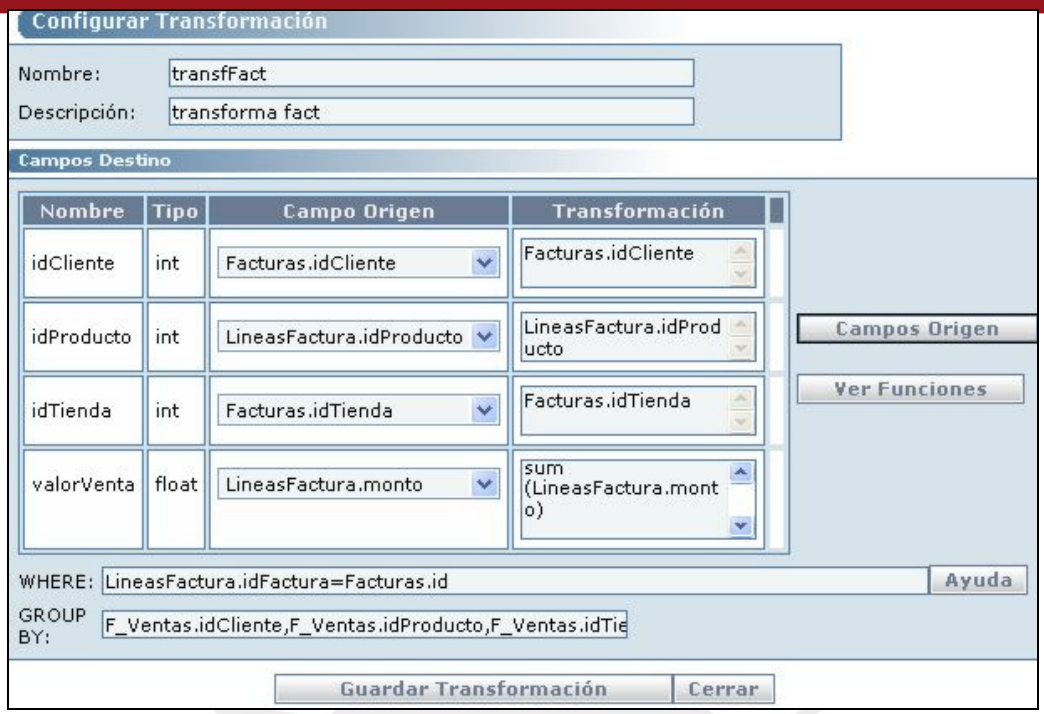


Figura 2.16: Ejemplo 2 – Configurar transformación para cargar *fact*

En la figura 2.17 se muestran el contenido de las tablas Facturas y LineasFactura, de donde se obtendrán los datos para cargar la tabla *fact*.

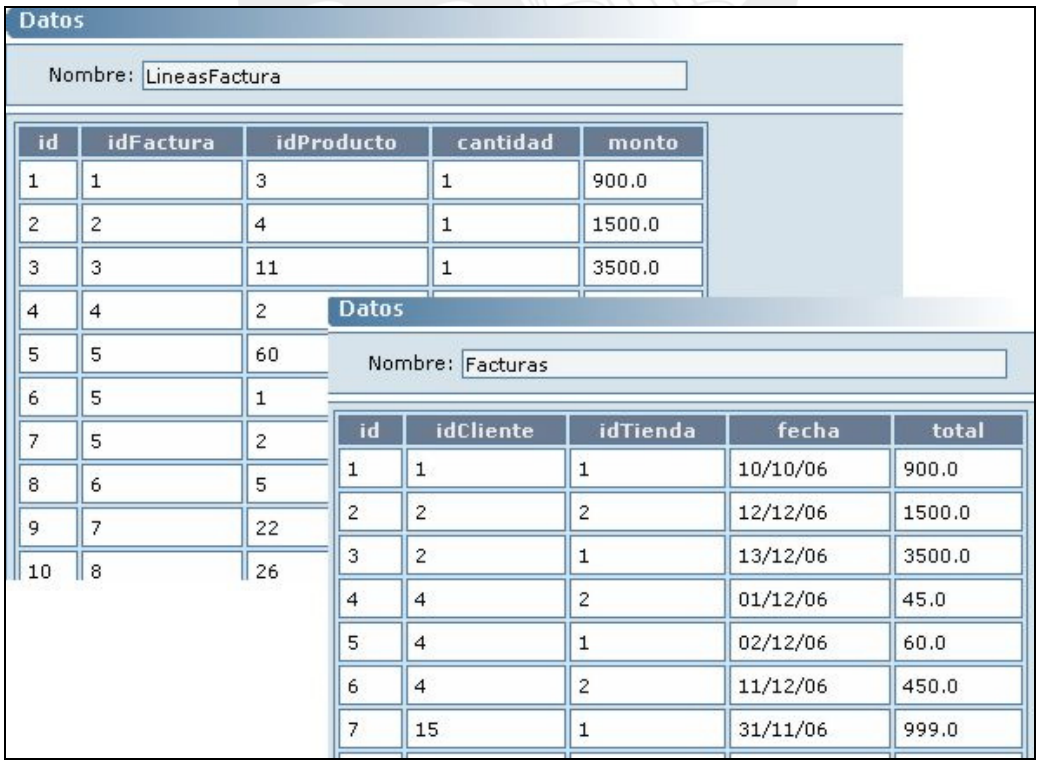


Figura 2.17: Ejemplo 2 – Contenido de tablas transaccionales

Finalmente, se muestran los datos en la tabla fact F_Ventas luego de haber ejecutado el job en la figura 2.18. Estos datos podrán ser convertidos en reportes de acuerdo a las necesidades del usuario usando el módulo de explotación de la herramienta AXEbit.

Datos

Nombre: F_Ventas

idCliente	idProducto	idTienda	valorVenta
1	3	1	900.0
2	4	2	1500.0
2	11	1	3500.0
4	2	2	45.0
4	60	1	10.0
4	1	1	20.0
4	2	1	30.0
4	5	2	450.0
15	22	1	999.0
18	26	2	500.0
10	13	1	7889.0
10	15	1	900.0
8	30	1	600.0

Figura 2.18: Ejemplo 2 – Contenido de la tabla *fact*

3. OBSERVACIONES, CONCLUSIONES Y RECOMENDACIONES

3.1. Observaciones

- En la actualidad existen diversas herramientas que ofrecen soluciones de Inteligencia de Negocios, la mayor parte de ellas orientadas a las grandes organizaciones, esta herramienta está dirigida a medianas organizaciones de forma que puedan explotar al máximo su información para su crecimiento.
- El proyecto que se presenta como trabajo de tesis ha pasado por varias etapas de evolución. En un inicio surgió como una idea de un grupo de profesores de la universidad. Con miras a materializar la idea, se convocó a un grupo de estudiantes para formar parte del equipo de desarrollo. Luego de formado el equipo, se concretó una serie de sesiones de capacitación sobre el tema de Inteligencia de Negocios para que el equipo tenga la idea de lo que se buscaba lograr con el proyecto (verano 2004). El producto fue tomando forma en una versión preliminar al ser trabajado por el equipo de desarrollo como parte del curso de Desarrollo de Programas 1 (primer semestre 2005). A inicios del 2006 se comenzó a definir los alcances de la versión definitiva del producto. En ese momento, el proyecto fue tomado como tema de tesis para los integrantes del equipo de desarrollo. Finalmente, el desarrollo del producto se ha prolongado por todo el año 2006 y se tiene planeado presentarlo como un producto comercial en el primer semestre del 2007.
- Mediante el presente trabajo de tesis no sólo se ha desarrollado una solución integral de Inteligencia de Negocios, sino que para el mismo se ha tenido en cuenta una investigación completa sobre los conceptos, metodologías y técnicas que la constituyen.

- La herramienta realizada puede alimentarse no sólo de una base datos especializadas como la de Oracle, sino que puede recibir también datos de archivos de texto, archivos de hoja de cálculo Excel y archivos XML.
- Los motores de base de datos contemplados son los siguientes: Microsoft Sql Server 7, Oracle 8, MySql 5.0.
- La herramienta ha sido desarrollada sobre la plataforma web, en lugar de un modelo cliente servidor, lo cual hace que esta tenga facilidad de acceso, no se requiere una instalación de un componente cliente sino basta con un navegador web. Ello hace que la herramienta tenga un valor agregado frente a las demás ofrecidas en el mercado.
- La solución de Inteligencia de Negocios presentada es una herramienta útil, flexible, de fácil manejo, coherente con el ambiente empresarial actual, que puede ser fácilmente utilizada en diferentes organizaciones.
- La aplicación es versátil en el sentido que no está sujeta a un estándar específico sino que da libertad, a la empresa que la utilice, de poder aplicar sus propios análisis y estructura de orígenes de datos. Esta característica permite un manejo más independiente y personalizado de las funcionalidades. El modelo de ejecución y trabajo dentro de la aplicación está apoyado en criterio de simplicidad y facilidad de manipulación.
- El presente trabajo de tesis fue realizado por cinco integrantes (3 en análisis y diseño, 2 en construcción y pruebas), en interacción constante con los otros dos módulos que constituyen el producto integral. El objetivo final es enlazar el módulo a una Solución Integral de Inteligencia de negocios, que permita alimentarse y entregar datos a los otros dos módulos que conforman el paquete de aplicaciones.

3.2. Conclusiones

- Se logró que el navegador web descargara menos de 500 *kilobytes* de datos para poder visualizar la pantalla del *applet*, lo cual se logra en menos de un minuto, cumpliendo uno de los objetivos planteados para la herramienta que especificaba la rápida carga de la aplicación en el equipo cliente. Con este fin, se utilizó gráficos del tamaño exacto a las necesidades del *applet*, así como también con un número limitado de colores para reducir la cantidad de datos a descargar para visualizar el *applet*. Del mismo modo, se buscó colocar en el *applet* solamente las funcionalidades indispensables que necesariamente requerían ser implementadas en el mismo, debido a restricciones tecnológicas en las páginas *JSF*.
- Además de reducir el tiempo de carga del *applet*, se convino colocar el código *JavaScript* en archivos separados de las páginas *JSF*. Esto permitió disminuir el

tiempo que demora en cargar una página, y tuvo la ventaja adicional de una administración del código más óptima.

- Se logró un rendimiento de procesamiento de 383 registros por segundo en promedio en un caso diseñado lo que no es nada despreciable considerando que la herramienta trabaja intensivamente con el disco duro. Cabe resaltar que las tasas de procesamiento dependen de varios factores tanto en el servidor de aplicaciones como en los orígenes y destino de datos del flujo de transformación.

- El rendimiento de la herramienta aún puede mejorarse modificando la forma de procesamiento de los datos, minimizando los accesos a disco duro y tratando de trabajar lo más posible en memoria. Se planea realizar estas optimizaciones en la versión comercial del producto de manera que pueda competir con los productos que se ofrecen en el mercado.

- Las herramientas usadas para la implementación del producto cumplieron con las expectativas del equipo de desarrollo para el proyecto, brindando facilidades que agilizaron el proceso y permitieron una integración rápida y sin contratiempos de los módulos del producto.

- La temprana definición del estándar de pantallas y navegación permitió al equipo construir un conjunto de herramientas consistente y que permite al usuario ubicarse rápidamente al pasar de un módulo al siguiente. Al realizar una prueba con un usuario nuevo, este demoró menos de 15 minutos en acostumbrarse al manejo de la herramienta y ubicación de las funcionalidades principales para las tareas básicas de la herramienta como son: crear un *job*, crear un paquete, conectar a una fuente de datos y formar el flujo de transformación.

- La herramienta construida es escalable a futuro, tanto a nivel de nuevos tipos de fuentes de datos para realizar las tareas de *ETL* como a nivel de páginas *JSF* para poder mostrar la herramienta en diferentes idiomas, de acuerdo al equipo cliente. Para lograr este objetivo se hizo uso de técnicas de diseño que se listan en los siguientes 3 puntos.

- El hacer uso de un esquema de herencia y polimorfismo en los objetos activos y pasivos, así como también en los objetos de dibujo permitieron construir fácilmente las diferentes clases de fuentes de datos derivando de un tipo común. De esta forma el mantenimiento y mejoras futuras de la aplicación no son una tarea ardua sino por el contrario sencillas.

- En el caso de las conexiones a motores de bases de datos, se utilizó un esquema que permite ampliar los tipos de motores a los cuales puede conectarse la herramienta de forma sencilla. Sólo es necesario contar con el *driver* adecuado para el motor de datos a agregar y realizar pequeños cambios en el controlador de fuentes de datos. Esto le da a la herramienta una cualidad de alto potencial de escalabilidad,

pues permitiría, con simples modificaciones, la inclusión de nuevas formas de manejo de datos que puedan surgir en el futuro.

- Se utilizó archivos *bundle* para contener el texto de las páginas JSF, tanto del título como en el cuerpo y componentes de las mismas. De esta forma se logró separar el contenido de las páginas del código *JSF* lo que permite la internacionalización de la herramienta. Es decir, mediante la inclusión de varios archivos *bundle*, uno para cada idioma, será posible mostrar la herramienta en varios idiomas sin necesidad de hacer cambios en el código *JSF*.

3.3. Recomendaciones

- Dado que la aplicación es web es necesario proteger el servidor de ataques externos, ya sea con firewalls o cualquier tecnología que exista actualmente, de manera que se asegure el funcionamiento del servidor las 24 horas del día, sin ningún tipo de interrupciones.
- Es necesario para ejecutar la aplicación la instalación del servidor Web en un Sistema Operativo que puede ser Linux o Windows, puesto que se trata del servidor Apache. Mientras que el cliente para correr la aplicación sólo necesita conectarse, ya sea por Internet o por una red privada al Servidor Web, usando un navegador (Internet Explorer 6 o Mozilla Firefox 1.5 o superiores) para correr la aplicación, por lo que el usuario puede estar situado en cualquier parte del mundo.
- La herramienta ha sido diseñada para ser de fácil uso, sin embargo, se considera necesario contar con la ayuda completa, para que se pueda usar el máximo de sus potencialidades.
- En el futuro, si se necesita ampliar la funcionalidad de la herramienta, como por ejemplo, agregar más tipos de fuentes de datos, lo más conveniente será tomar como base las clases genéricas existentes. Creando las nuevas fuentes de datos como clases que heredan de las clases genéricas se logrará aprovechar al máximo la implementación existente, sobreescribiendo sólo los métodos específicos para el funcionamiento del nuevo tipo de fuente de datos.

BIBLIOGRAFÍA

1. [ALM 1999] Maria Sueli Almeida, Missao Ishikawa, Joerg Reinshmidt, Torsten Roeber. 1999. Getting Started with Data Warehouse and Business Intelligence. IBM Technical Support Organization.

2. [KIM 2002] Ralph Kimball, Margy Ross. 2002. The Data Warehouse Toolkit. Second Edition, Wiley Computer Publishing.

3. [SAG 2006] http://www.sagent.com.ar/stecno.htm. Sagent Technology home page.

4. [MST 2006] http://www.microstrategy.com. MicroStrategy home page.

5. [BUS 2006] http://www.businessobjects.com. Business Objects home page.

6. [COG 2006] http://www.cognos.com. Cognos home page.

7. [SUN 2006] http://www.sunopsis.com/corporate/index.htm. Sunopsis home page.

8. [MSTG 2006] http://es.ascential.com/productos/datastage.html. DataStage home page.

9. [SMS 2004] http://java.sun.com/j2se/1.5.0/docs/relnotes/features.html. Sun Microsystems, Inc. 2004.Documentación del API de Java 1.5.

10. [W3C 2006] http://www.w3counter.com/globalstats/. Estadística de uso de exploradores

web en el mundo basado en una muestra de 6 millones de visitas a sitios web seleccionados. 2006. W3Counter home page.

11. [BOU 2005] http://www.rpbourret.com/xml/XMLAndDatabases.htm. Ronald Bourret. 1999-2005. XML and Databases. Ronald Bourret home page.

12. [JDO 2006] http://www.jdom.org/. JDOM Project home.

13. [CAS 2006] http://www.castor.org/ Castor Project home.

14. [RMI 2006] http://java.sun.com/javase/technologies/core/basic/rmi/index.jsp. Remote Method Invocation home.

15. [SAL 2005] http://www.codejava.org/?idxpagina=9&idxnota=32795&destacada=1 . Fabián Salvatierra. 2005. Struts y el Patrón de diseño MVC. Code Java home page.

16. [REA 2005] http://www.r-software.com/Doctos/patrones.pdf. Real Software S.A. de C.V. 2005. Diseño de aplicaciones internet usando los patrones de diseño J2EE.

17. [KRU 2000] Phillippe Krutchen. 2000. The Rational Unified Process: An Introduction. Segunda edición, Addison-Wesley Professional.

18. [IFP 1999] The International Function Point User Group (IFPUG). 1999. Function Point Counting Practices Manual Release 4.1. USA.

19. [COC 2002] http://sunset.usc.edu/research/COCOMOII/. Cocomo II. 2002.

ANEXOS

ANEXO A. PROPUESTA DEL PROYECTO

Plataforma de Inteligencia de Negocios.

Planteamiento de la necesidad

Una gran cantidad de organizaciones, sobre todo las medianas y pequeñas, toman las decisiones de manera empírica, siendo una de las principales dificultades la ausencia de información en un nivel de consolidación deseado. La baja calidad en las decisiones provoca pérdida de productividad y calidad en los procesos productivos y administrativos, haciendo menos eficiente, o más costosa, innecesariamente, los procesos de la organización.

Un ejemplo sencillo (y tomado de un caso real pero ligeramente modificado) es el de una empresa operadora de turismo que al hacer un análisis de sus servicios ofertados durante los cinco últimos años, descubre que en los meses m1 y m2 la demanda de adultos jóvenes que provienen de Europa por turismo de aventura es mayor que en el resto del año y en cambio, los que provienen de USA lo hacen en los meses m6 y m7. La empresa, a partir del conocimiento obtenido, decide cambiar mejorar su estrategia de marketing para ofrecer los paquetes en los meses previos a m1 para Europa y en los meses previos a m6 para USA. Este cambio de estrategia le significo un ahorro significativo pues focalizo mejor las campañas para cada tipo de público.

Descripción General del Proyecto

La UNESCO declaró en el 2003 la necesidad de contribuir a la constitución de la Sociedad de la Información (UNESCO, 2003). En el Perú, se publicó en el 2005 el Plan de Desarrollo de la Sociedad de la Información, en donde se le da prioridad estratégica a este tema como pilar del desarrollo (Reyes, 2004). Para todos es aceptable que hoy se habla de la sociedad de la información y el conocimiento, y que la construcción de ese nuevo escenario viene soportado por el uso de diversas tecnologías de la información y la comunicación.

Las organizaciones en el Perú y el mundo, han venido usando diversos sistemas informáticos para registro de sus transacciones, que dan soporte a sus actividades diarias. Los datos recogidos por diversos procesos se consolidan en información que es útil para la toma de decisiones. El cuadro 1, que se muestra a continuación, es un extracto del análisis desarrollado por la empresa Apoyo Opinión y Mercado, para el año 2003, donde se muestra el conocimiento, uso e implementación de diversas tecnologías de la información en el Perú clasificadas por medianas y grandes empresas (Apoyo, 2003).

Se puede apreciar que las tecnologías de Bases de Datos y Herramientas de desarrollo son las más ampliamente usadas en ambos grupos, debido a que con ellas se suele construir las soluciones de sistemas de información necesarias para las empresas. Tecnologías como el ERP (*Enterprise Resource Planning*) y *Data Warehouse* son conocidas, pero tienen un menor nivel de aplicación en general; y en particular, a nivel de medianas organización, existe una demanda grande por atender. Finalmente, para el caso de tecnologías asociadas a soluciones

de CRM (*Customer Relationship Management*) o SCM (*Supply Chain Management*) es mucho menor por lo altamente especializado y costoso que suelen ser.

Software de	Conocimiento			Uso			Implementación		
	Total 2003	Entidad		Total 2003	Entidad		Total 2003	Entidad	
		Grande	Mediana		Grande	Mediana		Grande	Mediana
Base de Datos	94	100	92	90	96	88	37	27	40
Herramientas de Desarrollo	97	98	96	83	87	82	37	31	39
ERP		72	45	37	44	34	22	25	21
Datawarehouse	44	65	38	22	41	16	28	32	27
CRM	27	47	21	13	15	13	16	25	14
SCM	15	25	12	8	17	6	14	10	16
Antivirus	100	99	100	99	99	99	31	22	33

Cuadro 1. Conocimiento, Uso e Implementación de TI en el Perú (Apoyo, 2003).

La consultora internacional Gartner Dataquest realizó un pronóstico a cinco años, basado en una estimación preliminar de tamaño del mercados y una revisión de los inhibidores e impulsores, llegando a la conclusión de que el total de mercado de herramientas de BI proyecta un crecimiento de $ 2.5 billones en 2004 a $ 2.9 billones en 2009, con una tasa de crecimiento anual de 7.4% (Medina, 2004). Según un reporte electrónico se informa que el mercado de soluciones de *Business Intelligence* en la región Asia/Pacifico (sin contar al Japón) se presume que crecerá de US$529.8 millones en el 2003 a US$1.2 mil millones en el 2008 (BI-New, 2005). En ese mismo año, en el Brasil, la inversión en servicios de consultoría de tecnologías de información relacionados a proyectos se ha incrementado; lo cual se muestra en el siguiente gráfico, donde se aprecia una mayor inversión en proyectos de *Business Intelligence* (IDC, 2004).

Gráfico 1: Servicios de consultoría de tecnologías de información

Considerando las tendencias antes presentadas a nivel nacional e internacional, se puede decir que hoy en día se hace muy necesario ir más allá de la recopilación de datos y se requiere hacer un tratamiento más elaborado de los datos, complejas transformaciones de información y

hacer un análisis más profundo para obtener un mejor conocimiento de las cosas y a partir de ellas, tomar mejores decisiones.

Aplicación de Inteligencia de Negocios

Las empresas han sido las primeras en aplicar esta tecnología para lograr un mayor conocimiento de sus operaciones y sacar ventaja de ello. La información obtenida de manera inteligente es considerada hoy una ventaja competitiva y son cada vez más organizaciones las que buscan un mejor acercamiento a sus clientes individuales u organizaciones y han hecho de la Inteligencia de Negocios (BI de las siglas en inglés de *Business Intelligence*) una prioridad de inversión. Con esta tecnología se puede obtener análisis de mercados verticales *end-to-end*, soluciones del pronóstico y optimización del mercado; entre una gama muy amplia de posibilidades.

En el cuadro 2 se aprecia que ya existe en el mercado un uso de esta tecnología en empresas que son grandes, sin embargo para medianas y pequeñas organizaciones les resulta muy costoso.

Giro / Empresa	**Análisis de Inteligencia de Negocios Aplicados**
Comercializadora Ewong	Ventas específicas por Cliente Servicio al cliente (preferencias, alejamiento de clientes) Reclamos a proveedores
Entretenimiento Cine Planet	Utilización efectiva de su capacidad instalada. Programación de pedidos a proveedores. Fidelización de clientes (hábitos de clientes)
Seguros La Positiva	Características de siniestralidad (pólizas) Estrategias definidas por productos. Renovación de pólizas
Hoteles Casa Andina	Análisis de ventas (%ocupación, hotel-temporada) Análisis de reservas (razones de cancelación, porcentaje de cancelación por temporada). Características de clientes (nacionalidad, sexo, edad, etc).
Gobierno SUNAT	Cruce de información de contribuyentes.

Cuadro 2: Aplicaciones de Inteligencia de Negocios en el Perú

Definición

La inteligencia de negocios implica establecer una forma de hacer las cosas y de ser disciplinado para obtener el máximo provecho. Citando a CherryTree & Co se presenta una definición como sigue:

“Las aplicaciones de Business Intelligence (BI) son herramientas de soporte de decisiones que permiten en tiempo real, acceso interactivo, análisis y manipulación de información crítica para

la empresa. Estas aplicaciones proporcionan a los usuarios un mayor entendimiento que les permite identificar las oportunidades y los problemas de los negocios. Los usuarios son capaces de acceder y apalancar una vasta cantidad de información y analizar sus relaciones y entender las tendencias que últimamente están apoyando las decisiones de los negocios. Estas herramientas ayudan a que las organizaciones no pierdan información valiosa (conocimiento de la organización) que se va acumulando con el transcurrir de las actividades y que no resulta de facil lectura (Medina, 2005).

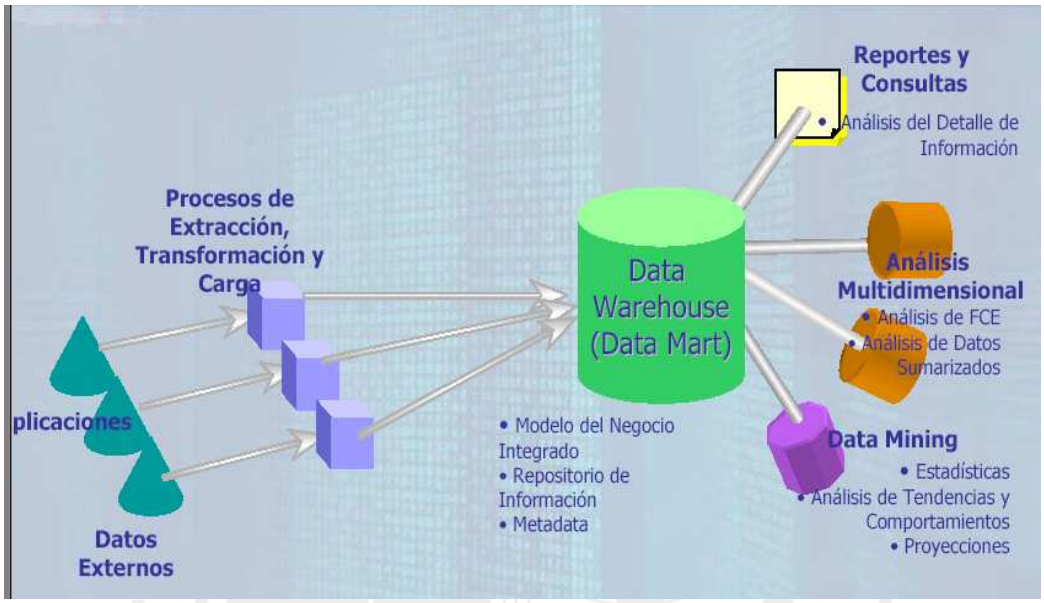


Gráfico 2: Modelo de aplicación de Inteligencia de Negocios (Medina, 2004).

Descripción Actual del Mercado

Para la implementación de un proyecto de Inteligencia de Negocios, se utilizan varios tipos de software que realizan una o varias etapas del proyecto, pero no integran todas ellas. Entre estos productos podemos destacar:

Producto	**Descipción**
Cognos:	Posee.una suite de productos que permiten realizar la explotación de datos y la extracción
Business Object:	Posee una suite de productos que permite realizar la explotación y extracción de datos
Proclarity:	Producto que permite realizar la explotación de datos
Microsoft	Posee una suite de productos que permite extraer y explotar datos

En una implementación de Inteligencia de Negocios, la interacción con el usuario final es importante ya que es él quien, en base al modelo de negocios, va a analizar la información. Por lo que el analista, en la etapa de modelado, necesita contar con un prototipo que permita

conocer el resultado final del modelo del negocios analizado. En el mercado sólo existen generadores de datos; pero no integrados a todas las etapas del proyecto.

Un proyecto de Data Warehouse necesita poder llegar a integrar todos los modelos de negocios de cada área de una empresa (Data Marts) para poder realizar la toma de decisiones en base a la información global del negocio.

Objetivo del Proyecto

1. Construir una plataforma informática para analistas de Inteligencia de Negocios, basados en un modelo de ventas de servicios de consultoría y procesamiento en Inteligencia de Negocios.
2. Implanta la plataforma en dos organizaciones de tamaño pequeño y mediano para evaluar el desempeño de la plataforma.

Componentes y Características del Producto: El producto final del proyecto es un software integrado que a nivel funcional cuente con los siguientes Componentes:

1. **Modelador Dimensional**, que permite a un consultor de Data Warehouse poder modelar un tema de negocios e interactuar directamente con un usuario final mostrándole el resultado como prototipo.
2. **Extractor de Datos**, que permite al usuario del sistema, poder esquematizar y realizar el proceso de carga de datos desde la fuente misma hacia el repositorio final en base al resultados del modelador dimensional.
3. **Explotador de Datos**, permite al usuario final poder generar sus propios reportes en base a los datos obtenidos por el Extractor de Datos.

El software presentará a nivel técnico las siguientes características:

1. **Modular**, es decir, que cubre todas las etapas para el desarrollo de un proyecto de Data Warehouse.
2. **Escalable**, es decir, que permite modelar diversos temas de negocios (Data Marts) para así integrar todo el modelo de la empresa (Data Warehouse). Práctica recomendada por Ralph Kimball
3. **Multiplataformas**, es decir, un producto desarrollado con tecnologías que permita su uso sobre las plataformas de sistemas operativas más ampliamente usados en nuestro medio como son Windows y Unix.

Requisitos Funcionales del Producto: Los requisitos de alto nivel que cubrirá el software desarrollado en el proyecto son las siguientes:

Modelado Dimensional: es una actividad donde se realiza la representación de distintas entidades (de datos) que permiten un análisis de una determinada situación de la organización

Generación del Modelo Dimensional

1. Administración de temas de análisis (Crear, modificar y eliminar)
2. Control el acceso de los usuarios por tema de análisis
3. Generación Modelo Dimensional (Estrella) de forma visual.
4. Administración de componentes del diseño dimensional (crear, modificar y eliminar): Dimensiones, Jerarquías y Atributos; Facts y Medidas.
5. Búsqueda de componentes del modelo (dimensiones, facts, atributos y medidas) para evitar generar entidades iguales
6. Reutilización de componentes en diferentes temas de análisis
7. Copiado de componentes en diferentes temas de análisis
8. Asistencia en la generación del modelo dimensional
9. Asignación el tipo de dato por cada atributo de una dimensión y las medidas de cada Fact
10. Visualización y generación de scripts de creación de tablas de diferentes manejadores de bases de datos
11. Bloque de un modelo dimensional en caso éste sea modificado por otro usuario
12. Visualización de mensajes de error en caso que el modelo dimensional no sea adecuado

Generación MetaData

1. Modificación y generacion la metadata de los modelos en formato Excel, HTML y Texto
2. Visualización del documento preliminar de la metadata

Simulación de Datos

1. Configuración de reglas de generación de datos de un modelo dimensional indicado
2. Generación de datos a partir de las reglas de un modelo dimensional indicado
3. Almacenamiento de los datos generados
4. Visualización de los datos generados en el módulo de explotación de datos

Extracción

1. Administración de los proyectos y paquetes de extracción
2. Control el acceso de los usuarios por proyectos y paquetes
3. Creación de un paquete de extracción de forma visual, drag & drop de componentes desde la paleta de componentes
4. Control del tamaño de la vista del área de trabajo (zoom)
5. Administración de varias ventanas por área de trabajo
6. Presentación gráfica del flujo de datos y el progreso del proceso de extracción de un paquete
7. Almacenamiento de la historia del estado de los procesos de extracción ejecutados
8. Conexión de varias bases de datos como fuente de datos y repositorio de destino

9. Control de los accesos de la fuente de datos y el destino del repositorio de datos
10. Generación de tablas temporales antes de enviar los datos definitivos al repositorio destino
11. Limpieza y estandarización de de datos como paso previo para la carga de datos
12. Transformación mediante operaciones definidas por el sistema o por una operación personalizada con lenguaje SQL Standar
13. Visualización previa de los datos de origen de acuerdo a la extracción definida
14. Visualización previa de los datos a cargar de acuerdo a las reglas de transformación definida
15. Personalización de un log de errores por cada componente de extracción
16. Programación de las tareas de extracción en un calendario
17. Reubicación de las tareas en un calendario y cancelaciñon de las ya programadas
18. Visualizar el estado de las tareas programadas
19. Presentación de reportes estadísticos sobre los tiempos de ejecución por tarea programada.

Explotación

1. Administración de los proyectos de explotación
2. Administración los Cubos Olap que forman parte de los proyectos
3. Administración los componentes de un cubo: Dimensiones, Jerarquías, Atributos, Medidas, Indicadores y Filtros.
4. Generación cubos a partir de la configuración de sus componentes
5. Creación reportes tabulares y/o gráficos a partir del cubo generado. Se puede utilizar Drg-Drop para la creación de los reportes
6. Ejecución de Drill-Down y Drill-Up a través de los informes generados
7. Control del acceso de los usuarios por proyectos, cubos e informes
8. Exportación de los datos a MS Excel, HTML y XML
9. Búsqueda de los componentes de los cubos configurados

Seguridad

1. Administración de Usuarios (Crear, modificar y eliminar)
2. Administración de Perfiles por opción del menú
3. Administración de contraseñas.

REFERENCIA BIBLIOGRAFICA.

- UNESCO, Annan Calls On Broadcasters to Help Create 'Open Inclusive' Information Society http://portal.unesco.org/ci/en/ev.php-URL_ID=13801&URL_DO=DO_TOPIC&URL_SECTION=201.html 2003.
- Apoyo Opinión y Mercado. Mercado Informático y Tecnología de Información, 2003.

- Dávila Abraham, Melgar Andrés, Modelado de la Metadata para el Desarrollo de Herramientas de Productividad sobre Múltiples Manejadores de Bases de Datos Relacionales. JIISIC04 (4ª Jornadas Iberoamericanas de Ingeniería del Software e Ing. del Conocimiento, España, 2004
- Reyes J., Plan de Desarrollo de la Sociedad de la Información en el Perú. LA agenda digital. Presidencia del Consejo de Ministros. CODESI: Comisión Multisectorial para el Desarrollo de la Sociedad de la Información, 2005.
- Medina Edison, Business Intelligence Un vistazo a la realidad peruana. Presentación COMMON DAY, 2004.
- Medina Jorge, Business Intelligence: Conceptos y Actualidad. 2005 http://www.gestiopolis.com/recursos5/docs/ger/buconce.htm.
- IDC, It Solution Market 2004, Agosto 2004.
- BI-New, BI booming in Asia / Pacific. 2005. http://www.bi-news.com/

ANEXO B. LISTA DE EXIGENCIAS

1. Lista de Exigencias de Proyecto AXEBit

N°	Características / Funcionalidades
No Funcional	
1.	Manejará los RBBMS: Sql Server, Oracle y MySQL.
2.	Tiempo de respuesta de búsquedas menor a 2 minutos.
3.	Repositorio de objetos será en XML.
4.	Herramienta WEB.
5.	Se desarrollará usando Eclipse.
6.	Se desarrollará usando Linux y soportar otros.
7.	Autosave por parámetro.
8.	Transacciones seguras.
9.	Confirmación de acciones en una barra de estado.
10.	Manual de usuario en web.
11.	Manual de instalación y configuración.
12.	Permite el manejo de drag&drop.
13.	Permite la configuración de apariencia gráfica de cada elemento o tipo de elemento (color de fondo, letra, tamaño, tipo de línea).
14.	Permite el zoom +/- y zap del modelo.
15.	Permite la configuración de reportes (fuentes, colores, gráficos, color de fondo, etc).
16.	Permite la generación de pistas (logs) de auditoria de los procesos.
17.	Permite la generación de pistas (logs) de errores de los procesos.
Análisis - Modelador	
1.	Entorno gráfico para el diseño del modelo estrella.
2.	Permite el manejo de elementos de manera gráfica.
3.	Permite la administración de elementos del modelo: temas de análisis, facts, dimensiones, atributos, jerarquías.
4.	Permite la administración de elementos del modelo: medidas, indicadores y relaciones.
5.	Permite el modelado físico y lógico
6.	Permite organizar el modelo según objetivos-temas-facts-dimensiones.
7.	Permite organizar el modelo según temas-facts-dimensiones.
8.	Permite organizar el modelo según facts-dimensiones.
9.	Permite organizar el modelo según temas-objetivos// facts-dimensiones.
10.	Permite organizar el modelo según temas-dimensiones.
11.	Permite definir una dimensión como histórica (subrrogate-key).
12.	Permite el manejo de periodicidad de carga de una fact (política de actualización de fact).
13.	Entorno gráfico para el diseño del modelo snow-flake.
14.	Alarma para señalar uso de malas prácticas (por identificar).
	- mensaje por no tener una dimensión tiempo
	- mensaje por una dimensión que tenga más de una llave
15.	Soporte para la personalización de identificadores dentro del modelo: pre-fijo, postfijo para diversos elementos (reglas de identificadores)
16.	Permite identificar las medidas como aditivas, semi-aditivas y no-aditivas.
17.	Permite la importación de elementos entre modelos.
18.	Permite la copia de elementos (para modificación) de otros modelos (incluso diccionario).
19.	Permite la administración de formatos de datos para cada elemento.

20.	Permite la asignación de formatos de datos para cada elemento.
21.	Maneja todos los tipos de datos de acuerdo al RDBMS seleccionado.
22.	Permite registro de descripciones y notas sobre los elementos.
23.	Permite el bloqueo de un modelo dimensional en caso esté siendo modificado por otro usuario.
24.	Permite el manejo de plantillas de temas.
25.	Permite el manejo de plantillas de modelos.
26.	Generación de scritps para la creación de las tablas del modelo.
27.	Generación directa de las tablas del modelo en la base de datos.
28.	Visualización de scritps para la creación de las tablas del modelo.
29.	Almacenamiento de scritps para la creación de las tablas del modelo.
30.	Ingeniería reversa de la definición de las tablas.
31.	Exportar el modelo a un formato gráfico (jpeg, tiif, pdf).
32.	Visualización icónica del modelo.
33.	Identificación de usuario que tiene en uso un elemento.
Análisis - Metadata	
1.	Almacenamiento de la data (modelo) de los elementos (guardar)
2.	Permite que cada objeto tenga su diccionario de datos.
3.	Permite realizar búsquedas para todos los objetos (elementos): temas de análisis, facts, dimensiones, atributos, jerarquías.
4.	Permite realizar búsquedas para todos los objetos (elementos): medidas, indicadores y relaciones.
5.	Permite realizar búsquedas para los objetos: proyecto, reglas de extracción o cubos.
6.	Permite realizar búsqueda sobre objetos con patrones de cadenas en su signif.icado o identificador
7.	Visualización gráfica de la dependencia de los objetos:
	- a partir de un objeto seleccionado.
	- a partir de un tipo de objeto.
8.	Permite registro de descripciones y notas sobre los elementos.
9.	Permite la generación de reportes a partir de la metatada (ver power desginer).
10.	Permite la definición de TODOS los objetos del sistema (incluye la importación y definición de orígenes).
Análisis - Generador	
1.	Desarrollo basado en DatProducer (ya tiene desarrollado varias cosas, se debe pasar a web).
2.	Permite la configuración de reglas de formación de la data (GUI Web).
3.	Permite la generación de datos para RBDMS: MSSqlServer, Oracle, MySQl (en realidad falta MySQL).
4.	Permite el manejo de archivos de texto como base para generar datos.
5.	Permite el uso de datos generados previamente en bases de datos.
6.	Permite el manejo de reglas de integridad (ya tiene parcialmente).
7.	Permite la vista preliminar de la data parcial.
Análisis - Otros	
1.	Permite la estimación del tamaño del DataWarehouse (en base al modelo físico de acuerdo al RBDMS).
2.	Permite el manejo de privilegios por objetos.
3.	Permite el manejo de privilegios por empresas.
4.	Permite el manejo de privilegios por proyectos.
5.	Permite la administración de múltiples empresas.

6.	Permite la administración de múltiples proyectos por empresa.
7.	Permite la administración de usuarios por empresa (nota una persona puede estar en varias empresas y solo tener un id).
8.	Permite la administración de perfiles.
9.	Permite la asignación de usuarios a proyectos.
10.	Permite la administración de privilegios del usuario por objetos.
11.	Administración de pista de auditoria (log).
12.	Visualización de pista de auditoria.
13.	Administración de pista de errores (guardar, borrar).
14.	Visualización de pista de errores.
15.	Registro de esfuerzo por modelo.
16.	Permite definir la regla de limpieza de datos por cada objeto del modelo.
Extracción - Comunicación	
1.	Permite la comunicación con fuentes: Oracle, MSSql Server, Textos, XML, hojas de cálculo, MySQL, Archivo Hash.
2.	Permite la visualización de objetos desde las fuentes de datos:
	- tablas, campos, características de los campos (longitud, escala, tipo de datos, nulos, si es PK), atributos.
3.	Utiliza un parser para la comprobación de un script (de comandos).
4.	Ingeniería Reversa de la definición de tablas del modelo relacional.
5.	Agrupación de los programas de extracción por temas (jobs).
6.	Llamada de programas de extracción desde un programa de extracción.
Extracción - Extrae	
1.	Extracción de datos considerando las fuentes y los objetos para la consolidación.
2.	Permite el manejo de objetos de Extracción, Transformación y Carga.
3.	- Objetos pasivos (fuentes y destinos de información, stored procedure, ODBC, FTP, webservices, xml, etc.).
4.	- Objetos activos:
5.	- objeto que pueda extraer, transformar y cargar (genérico), incorpora el lookup.
6.	- objeto especializado Merge
7.	- objeto especializado Sum
8.	- objeto especializado Lookup
9.	- objeto especializado Transpuesta
10.	- objeto especializado Sort
11.	- objeto especializado Ejecutar comando (interno y del sistema operativo)
12.	- objeto de operación para mandar mensajes (alarmas) a usuarios específicos
13.	- objeto de operación para sincronizar procesos.
14.	El diseño de la extracción y la configuración de los objetos será en forma gráfica
15.	Ver modelador para la parte gráfica y de seguridad
16.	Permite el manejo de objetos de limpieza de datos durante la definición de los otros objetos activos
17.	Permite el manejo de reglas de limpieza como lista de comprobación.
18.	Permite la definición de pistas (log) de procesamiento y rechazos
19.	Permite generar log de eventos y errores de apoyo a la extracción y transformación.
20.	Permite generar log de eventos y errores de apoyo a la extracción y transformación.
Extracción - Ejecución	
1.	Integración: programación (schedule) y monitoreo.
2.	Permite programar la ejecución de los programas de carga de datos.
3.	Permite conocer los tiempos de ejecución promedio de cada programa.

4.	Permite visualizar el monitoreo de cada ejecución con sus tiempos de carga y número de filas por segundo por cada objeto activo.
5.	Permite visualizar el monitoreo de recursos de hardware y enviar alarmas.
Explotación - Cubos	
1.	Permite sugerir cubos a partir de las facts, agrupados por temas.
2.	Permite diseñar cubos en base a las dimensiones y jerarquías definidas en el análisis.
3.	Permite la creación de dimensiones, jerarquías, medidas calculadas definidas en el análisis.
Explotación - Reportes	
1.	Permite la generación de reportes gráficos y tabulares pre-definidos
	- pastel, líneas, barras, pareto
2.	Permite la generación de reportes gráficos y tabulares definido por el usuario
3.	Permite la administración de accesos para la visualización de reportes.
4.	Permite la definición de filtro.
5.	Permite la definición de ordenamiento, ranking y excepciones, los primeros, los últimos y entre dos o más posiciones.
6.	Permite el cálculo de y presentación de promedios, mínimos, máximos, curva de ajuste (mínimos cuadrados)
7.	Permite presentar gráficamente el cálculo de promedios, mínimos, máximos, curva de ajuste (mínimos cuadrados)
8.	Permite crear totales y subtotales de los reportes como: sumas, contadores, promedios, mínimos, máximos.
9.	Permitir exportar reportes bajo formato XML, HTML, pdf, hoja de cálculo electrónico, cvs, imágenes (giff, jpeg, bmp).
10.	Permite la visualización mdx del reporte.
Explotación - Integrado	
1.	Permite la generación de un cubo a partir de los datos de generación de data de análisis
2.	Permite presentar gráficamente el cálculo de promedios, mínimos, máximos, curva de ajuste (mínimos cuadrados)
3.	Permite programar la generación de cubos para el análisis.
Explotación - What-if	
1.	Permite la creación de reportes gráficos de proyección.
2.	Permite el análisis de sensibilidad a futuro, cambiando valores del presente.

ANEXO C. DOCUMENTO DE ARQUITECTURA

1. Introducción

1.1. Objetivo

Este documento nos proporcionará una descripción de la arquitectura de la aplicación desarrollada por el proyecto BI-PUCP a través de una serie de diferentes vistas arquitectónicas del mismo para representar diferentes aspectos del sistema.

Con la arquitectura se podrá también presentar las relaciones existentes entre los 3 módulos de Análisis, Extracción y Explotación y la manera de verificar dichas relaciones, lo que ayudará al equipo de desarrollo a validar la consistencia de las mismas.

1.2. Alcances

Este documento describirá las vistas de Caso de Uso, Lógica, de Despliegue y de Implementación de la representación arquitectónica del software que se desarrollará. Este documento se aplica al desarrollo del prototipo de arquitectura y sienta las bases para el diseño e implementación de las iteraciones de la fase de construcción de la herramienta AXEbit.

1.3. Referencias

Para la elaboración del presente documento, se tomaron como referencias a los siguientes documentos:

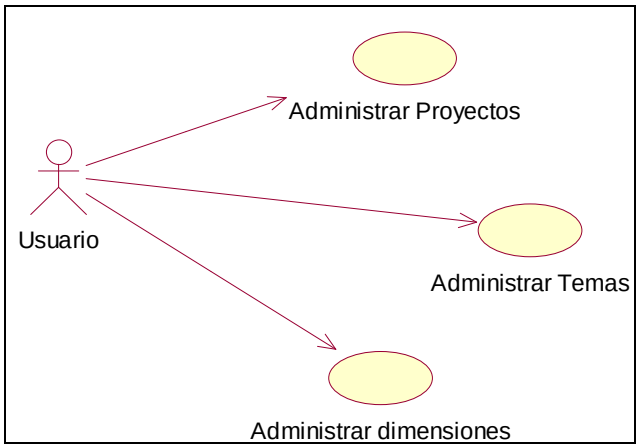
1. Especificación de requerimientos de software, v1, BI-PUCP, 2006.
2. Plan de proyecto, v1, BI-PUCP, 2006.
3. Documento de análisis, v1, BI-PUCP, 2006.

2. Vista de casos de uso

Estos son los casos de uso que se implementaron en el prototipo de arquitectura de manera que se pudiera comprobar su viabilidad.

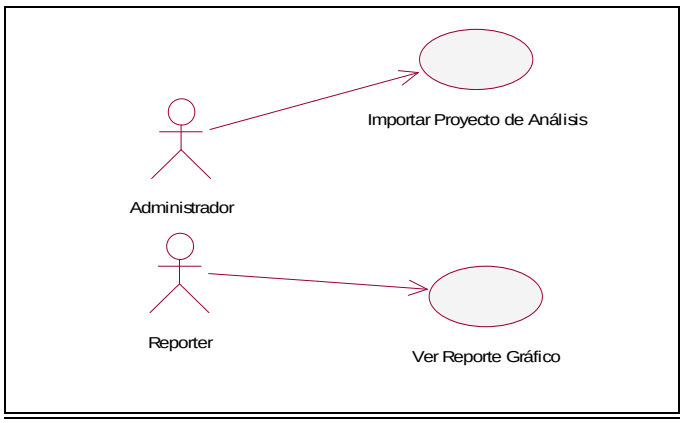
Se utilizaron los componentes mencionados en la Vista de Despliegue del presente documento en los tres módulos del prototipo y se comprobó el efectivo funcionamiento del mismo.

2.1. Módulo de Análisis



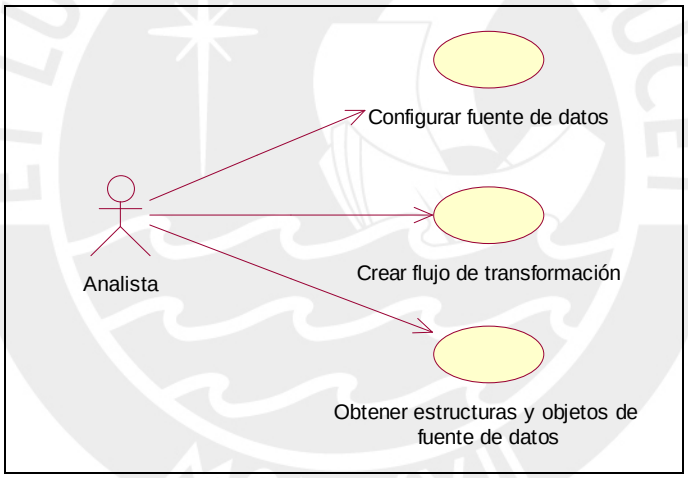
- **Administrar proyectos:** Se implementó la parte de creación de un proyecto para el sistema, el cual administrará los temas de análisis y sus componentes.
- **Administrar temas de análisis:** Consiste en la creación de un tema de análisis dentro de un proyecto, para contener tablas relacionadas entre sí, las cuales tendrán la información del tema, en las dimensiones y facts.
- **Administrar dimensiones:** En esta etapa consiste en crear tablas que contendrán la información que generará la metadata. Estas tablas tendrán aquellos valores que serán objeto de análisis.
- **Persistencia en xml:** Consiste en un modulo especializado para el manejo de la información de los proyectos que se generen al momento de utilizar la herramienta. Básicamente el Servlet llamaría a este modulo utilizando un controlador para que este guarde y recupere la información de los proyectos. Este no es propiamente un caso de uso pero es un tema importante dentro de nuestra implementación.

2.2. Módulo de Explotación



- **Importar proyecto de análisis:** Engloba las siguientes funcionalidades:
- **Importar Tema de Análisis:** Consiste en formar un cubo de explotación en base a un tema creado por el módulo de análisis. El cubo sería creado en base a la metadata del tema de análisis.
- **Crear Esquema de explotación:** Consiste en la creación de un esquema XML basado en la información obtenida del cubo creado en base al esquema de análisis. Dicho esquema sigue el formato específico de comunicación con el motor OLAP.
- **Administrar Reporte (flujo de creación) y Ver Reporte (gráfico):** Consiste en mostrar el resultado gráfico de un reporte. la implementación de la comunicación con el motor OLAP por medio de sentencias MDX.

2.3. Módulo de Extracción



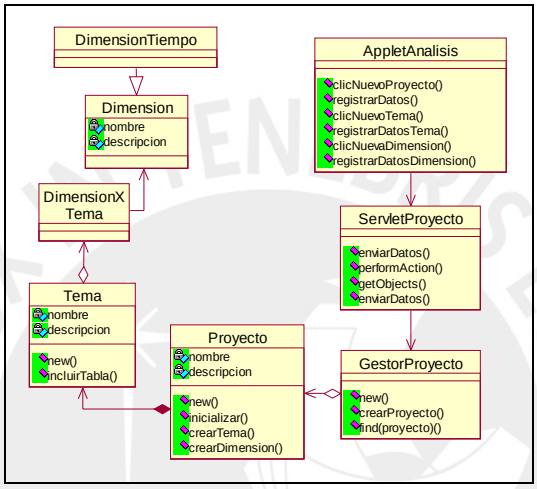
- **Configurar fuente de datos:** Consiste en ingresar los datos necesarios para realizar una conexión a la base de datos que se está configurando para su uso en la herramienta.
- **Crear flujo de transformación:** Consiste en formar un flujo de transformación en el área de dibujo de la herramienta de manera gráfica.
- **Obtener estructuras y objetos de fuente de datos:** Consiste en visualizar la estructura de tablas y campos de las fuentes de datos configuradas en la herramienta.

3. Vista Lógica

Esta sección es una descripción de la vista lógica de la arquitectura. Describe las clases más importantes y la interacción entre las mismas.

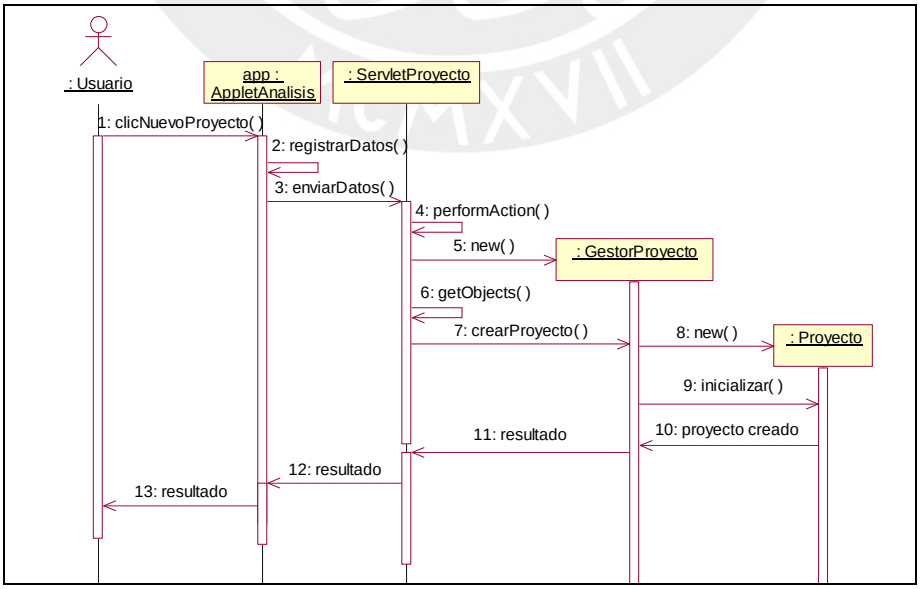
3.1. Módulo de Análisis

3.1.1. Diagrama de Clases

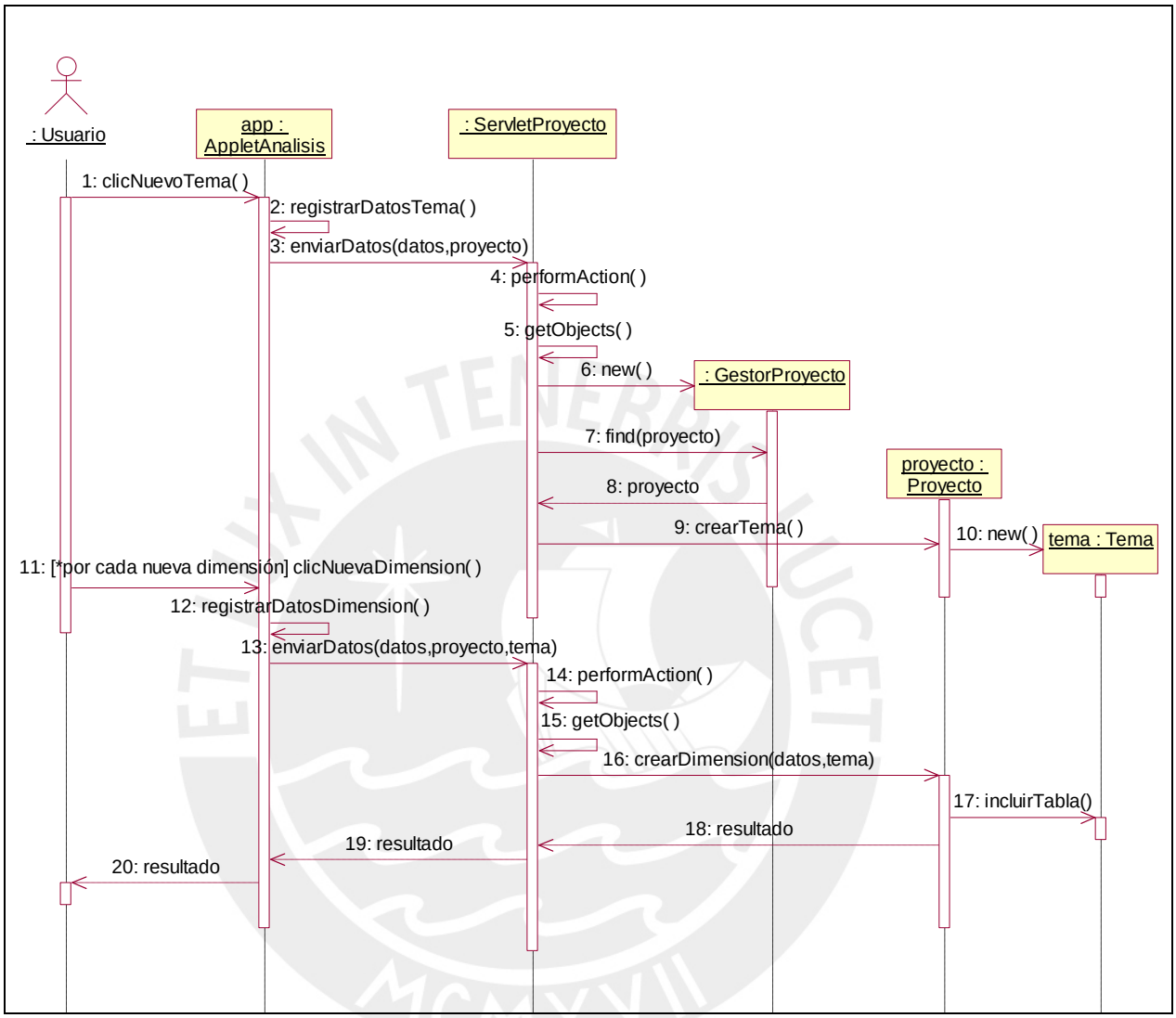


3.1.2. Diagramas de secuencias

Caso de uso “Administrar proyecto” (flujo creación)

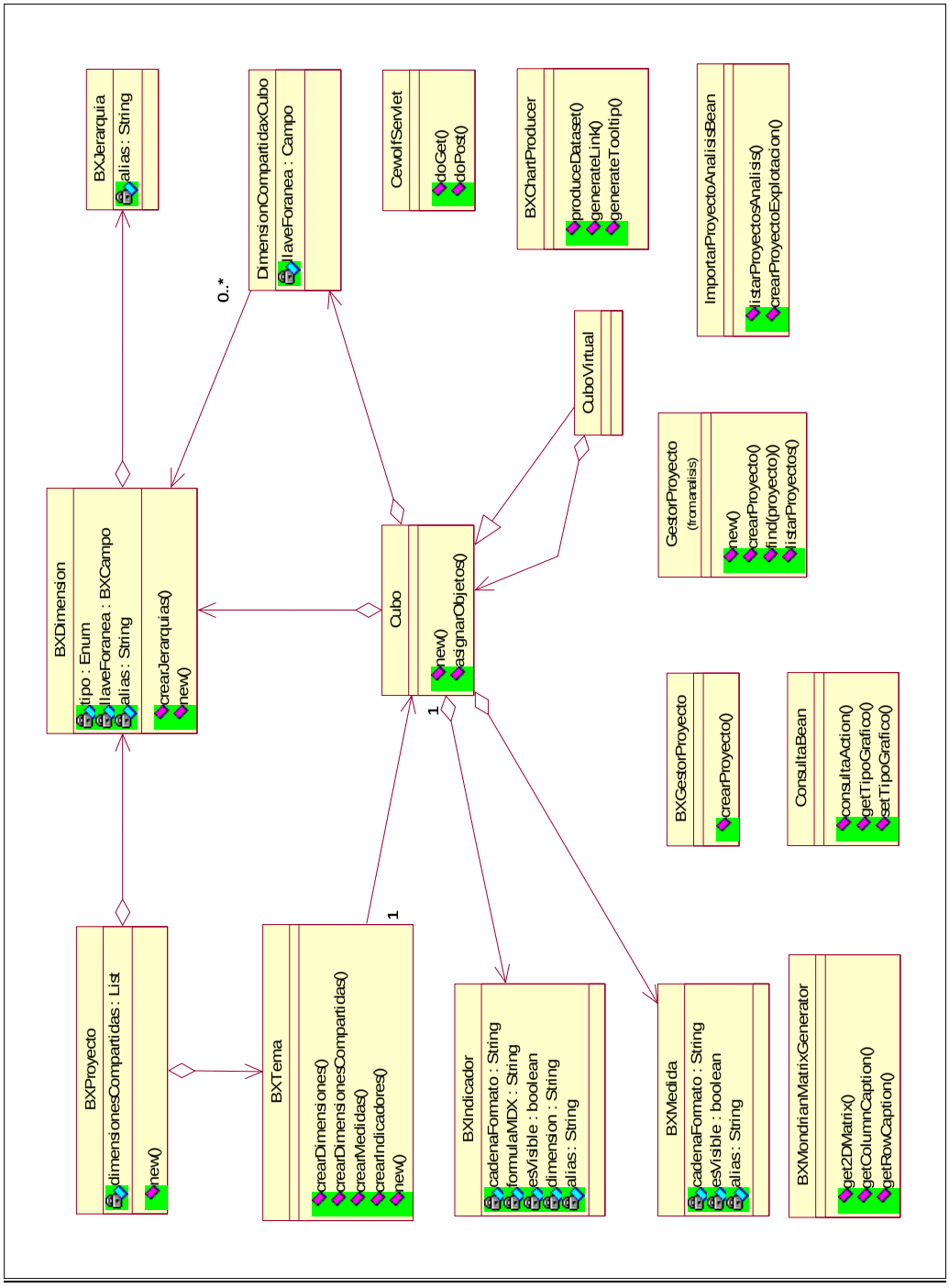


Casos de uso “Administrar dimensiones” y “Administrar temas de análisis” (flujos creación).



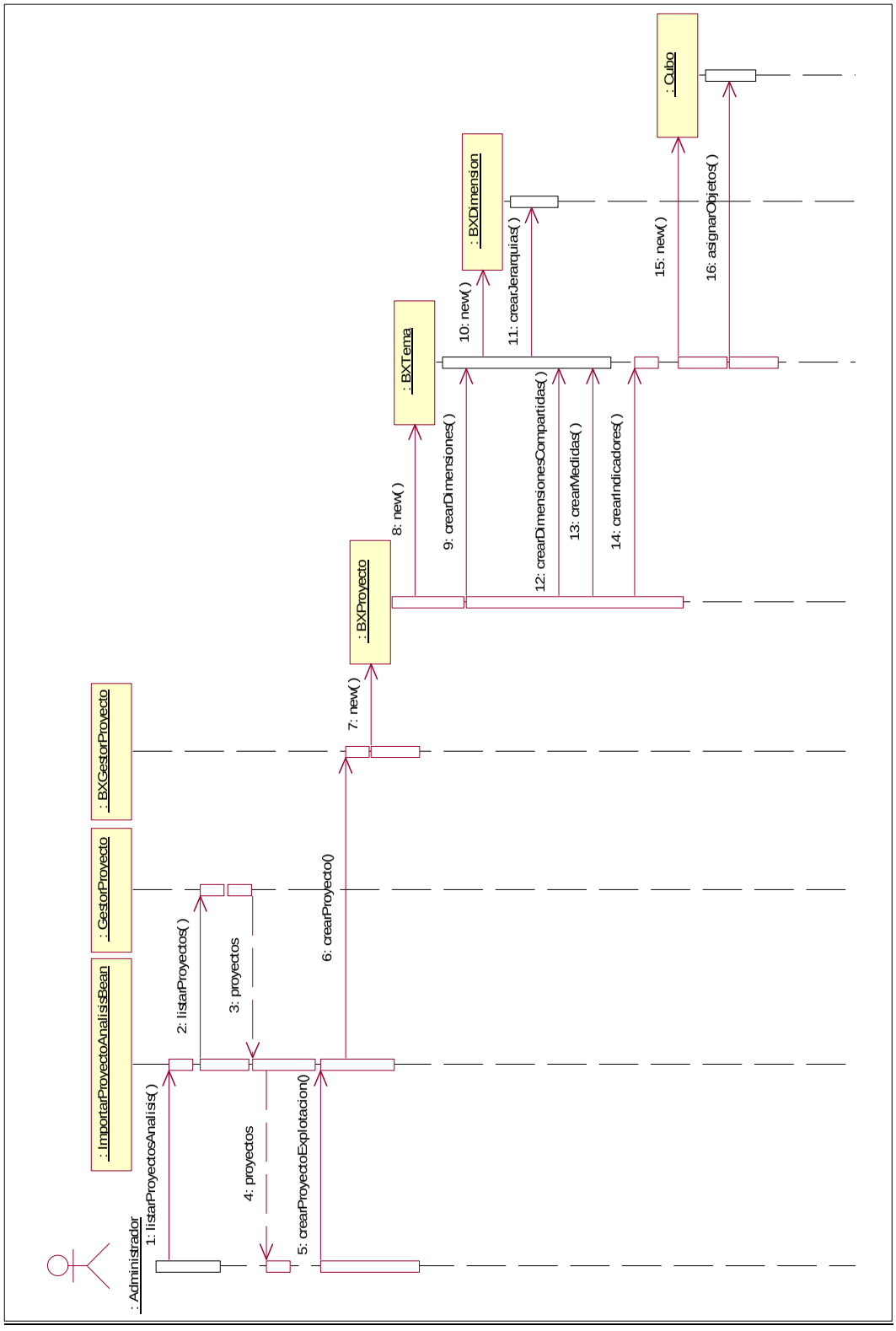
3.2. Módulo de Explotación

3.2.1. Diagrama de Clases

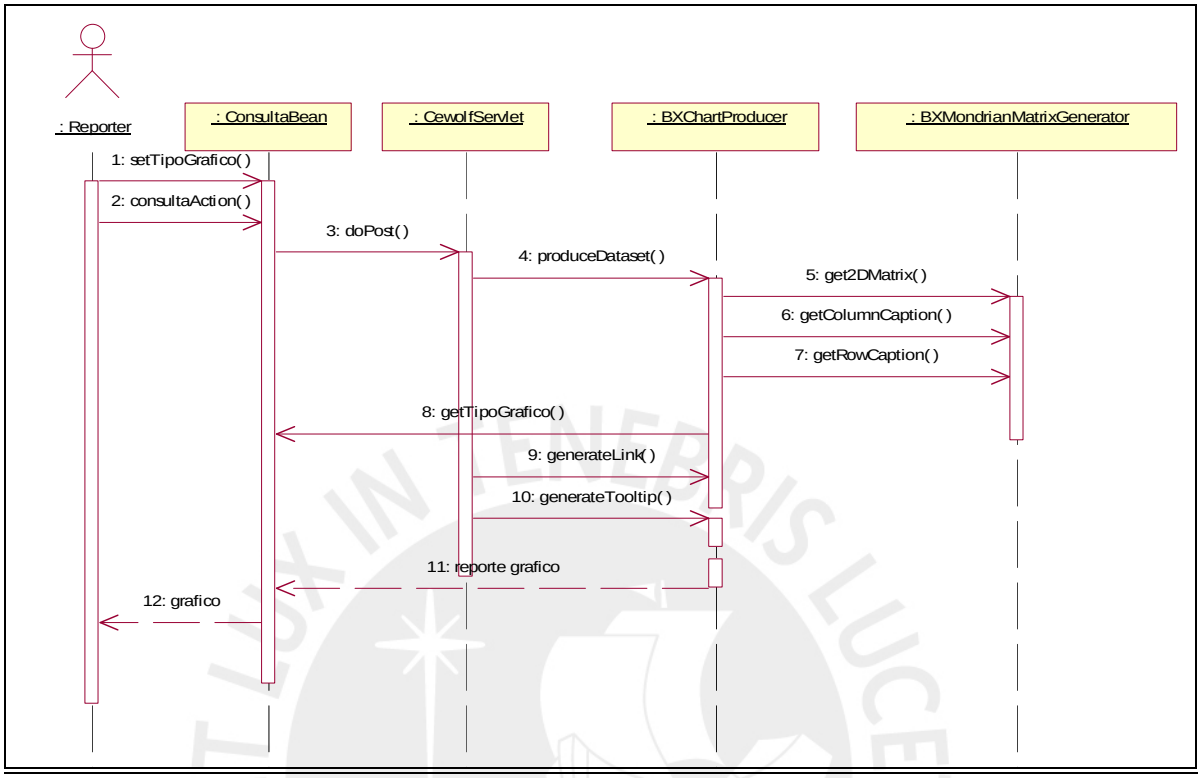


3.2.2. Diagrama de Secuencias

Caso de Uso Importar Proyecto de Análisis

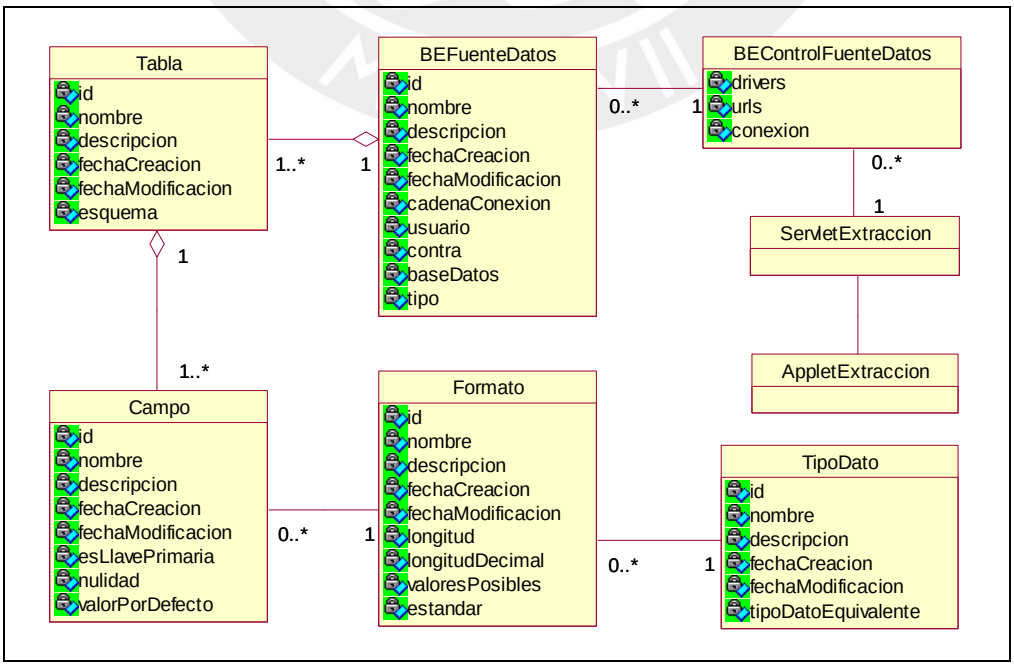


Caso de Uso Ver Reporte Gráfico



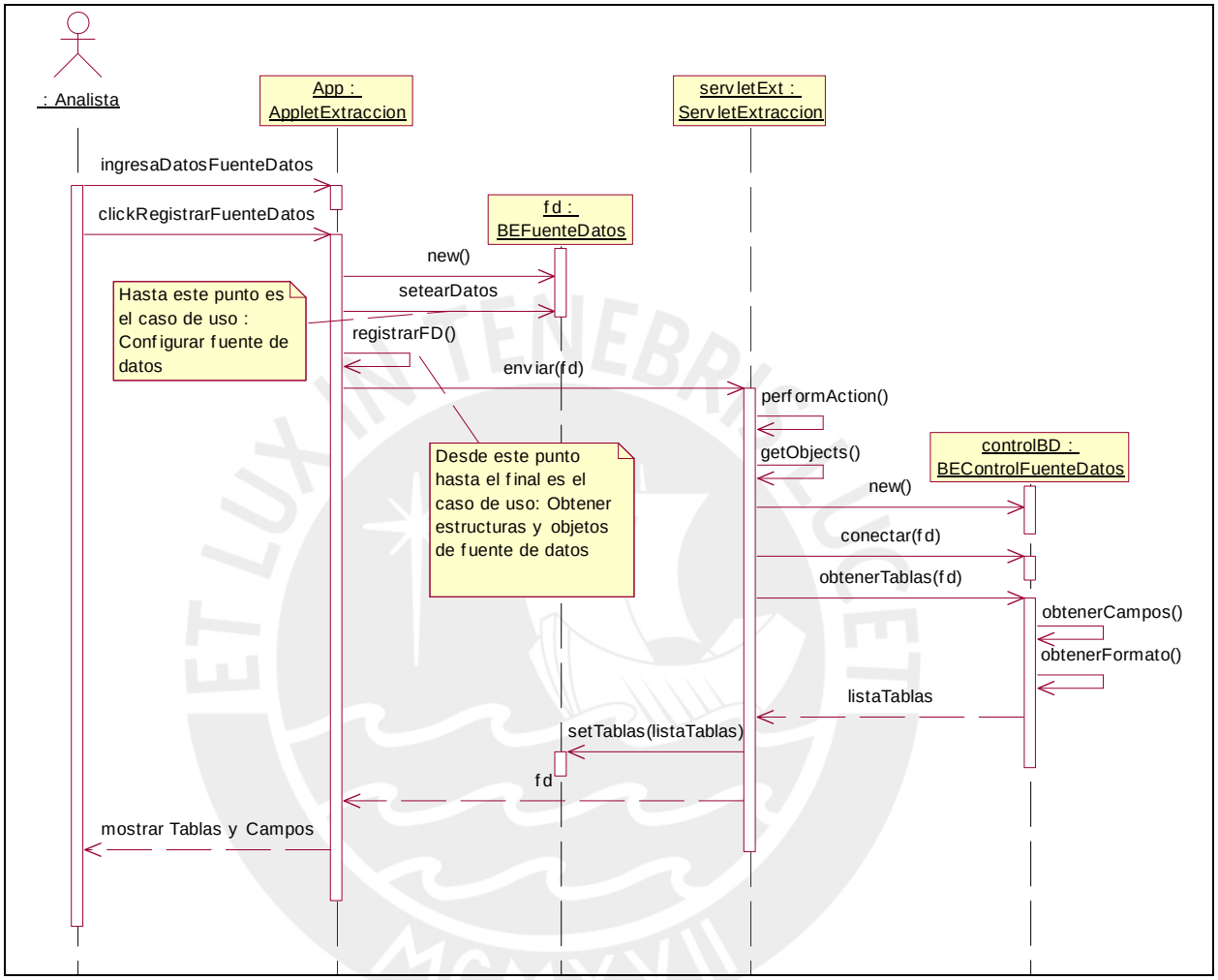
3.3. Módulo de Extracción

3.3.1. Diagrama de Clases

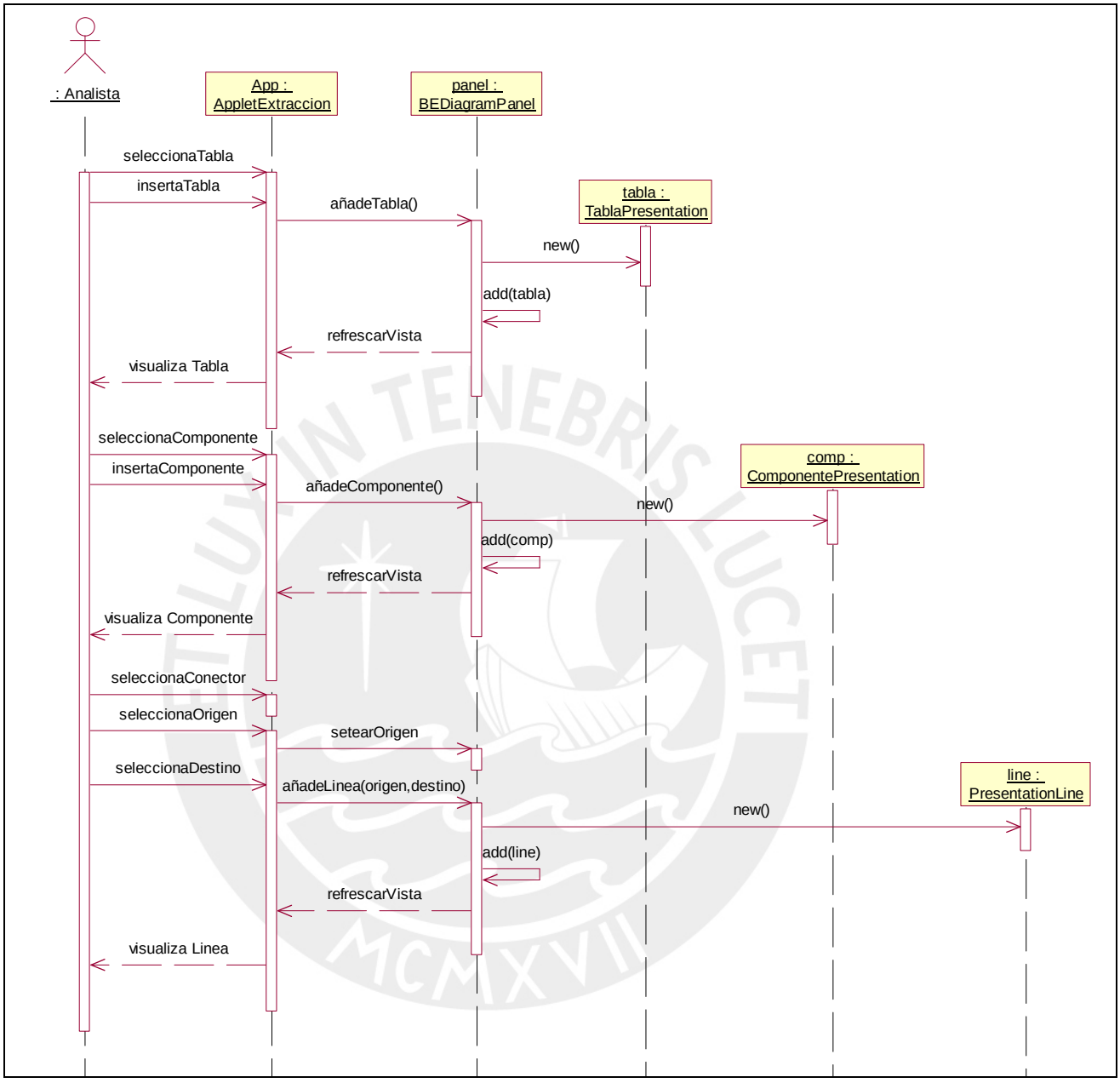


3.3.2. Diagramas de secuencias

Casos de uso “Configurar fuente de datos” y “Obtener estructuras y objetos de fuente de datos”.

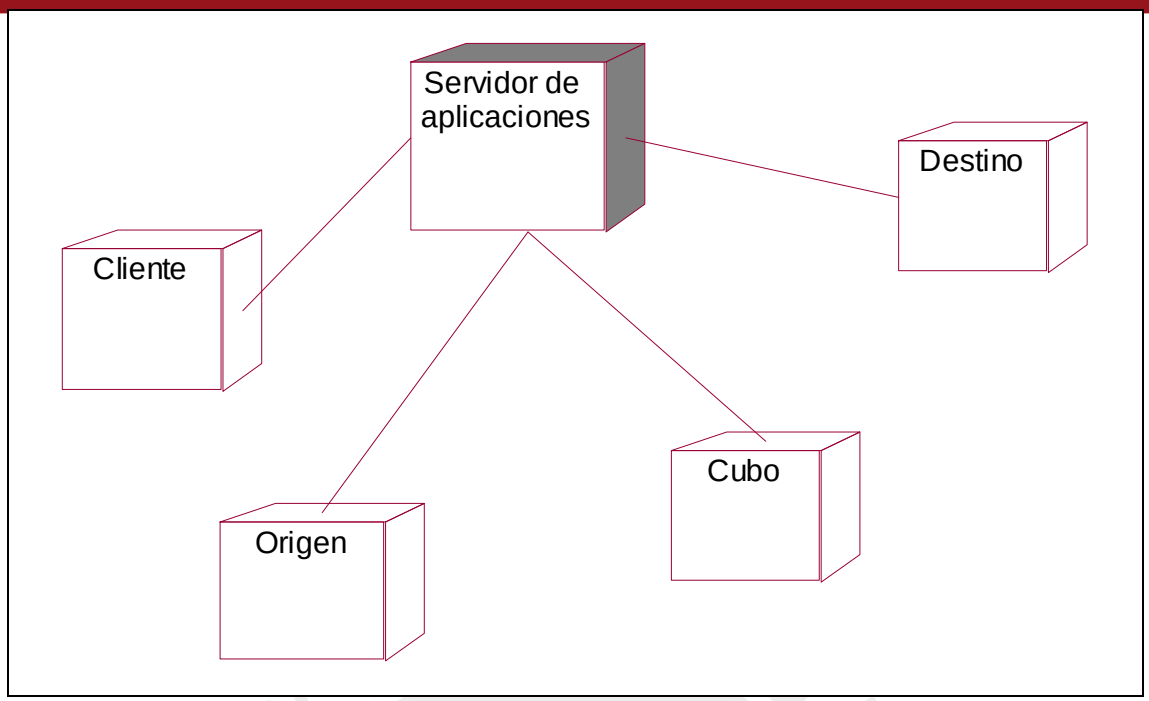


Caso de uso “Crear flujo de transformación”.



4. Vista de despliegue

En esta vista se describe la configuración de red física sobre la cual se ejecuta la herramienta AXEbit. Indica los diferentes nodos físicos en los cuales se ejecuta el software y sus interconexiones.



Paquetes	
Nombre	**Descripción**
Servidor de aplicaciones	Es el servidor que procesa la información que envía el cliente y devuelve los resultados después de las operaciones que se realizan. Es el nexo entre los demás nodos y administra el flujo de trabajo entre ellos.
Cliente	Los usuarios de la herramienta harán uso de ésta a través de Internet por medio de un navegador que es donde se descargarán los applets para trabajar con la herramienta, además se visualizarán las páginas jsp y jsf.
Origen	Es la fuente de datos de donde se obtiene la data a ser procesada. Viene a ser un servidor de base de datos con bases de datos con la data origen.
Cubo	Guardará información de la estructura de los cubos generados para la explotación, serán utilizados por el motor OLAP para obtener la data desde las fuentes definidas para dicho cubo.
Destino	Es la fuente de datos a donde se deriva la data luego de ser procesada. Viene a ser un servidor de base de datos con bases de datos que contendrán la data transformada y la que surge de la explotación de datos.

5. Vista de implementación

Esta sección muestra el conjunto de paquetes reutilizables de cada módulo de la herramienta AXEbit, así como las respectivas interfaces que deben exponer para que se interrelacionen de una manera clara.

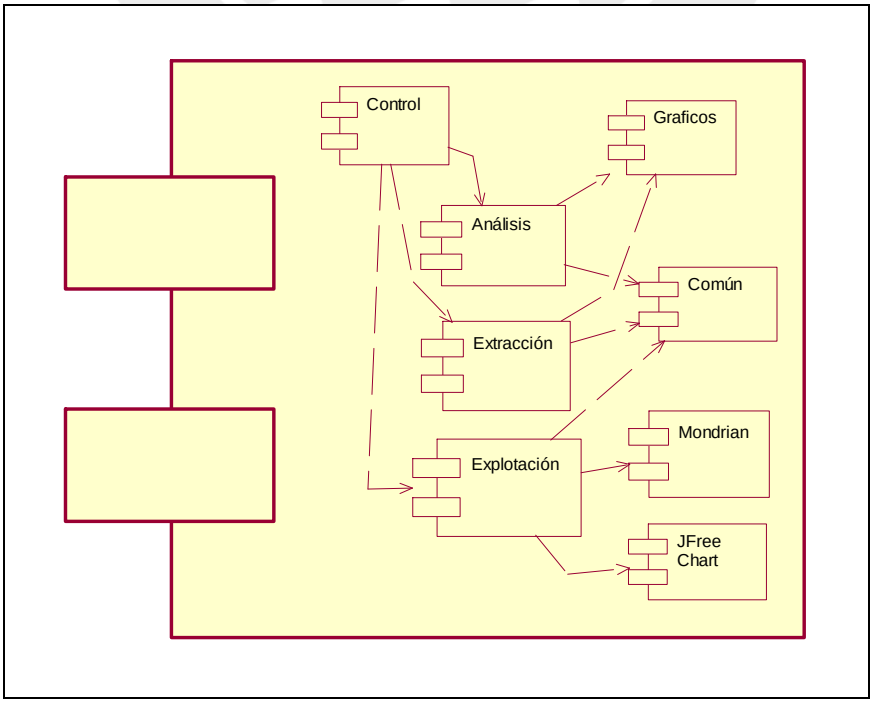
Entonces, la arquitectura del producto software esta conformado por los siguientes componentes:

- Aplicación
- Control
- Común
- Gráficos
- Análisis
- Extracción
- Explotación

Estos componentes se describen a continuación:

5.1. Componente: Aplicación

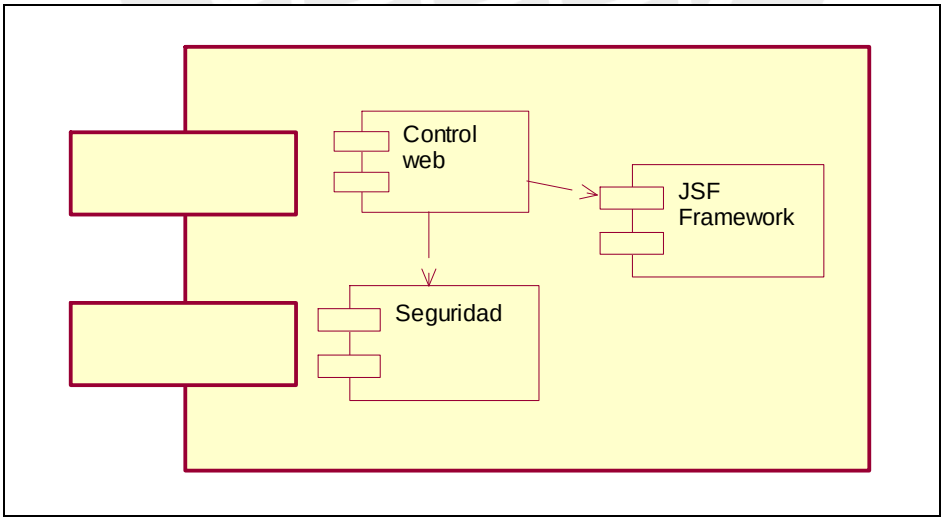
Este constituye el componente principal de la herramienta ya que contiene a todos los demás y en su interior se define el flujo de trabajo de los demás componentes. En caso de que la herramienta se instale para un nueva empresa o grupo de usuarios se deberá usar este componente para acceder a la funcionalidad de la herramienta en su conjunto.



Aplicación	
Componente	**Descripción**
Análisis	Este componente contiene todos los subcomponentes que se utilizan en el módulo de análisis de la herramienta.
Común	Este componente contiene todos los subcomponentes comunes que son usados en todos los módulos de la herramienta.
Control	Este componente contiene todos los subcomponentes encargados del flujo de trabajo entre los tres módulos principales de la herramienta, además permite el acceso a estos componentes.
Explotación	Este componente contiene todos los subcomponentes que se utilizan en el módulo de explotación de la herramienta.
Extracción	Este componente contiene todos los subcomponentes que se utilizan en el módulo de extracción de la herramienta.
Gráficos	Este componente contiene todos los subcomponentes necesarios para realizar los gráficos en los applets.
CeWolf / JFreeChart	Componente utilizado para realizar los gráficos de explotación (charts). JFreeChart realiza los gráficos basados en datasets, y CeWolf permite mostrarlos vía web.
Mondrian	Motor OLAP utilizado por explotación, permite la conexión al DataMart y obtener los resultados de la explotación.

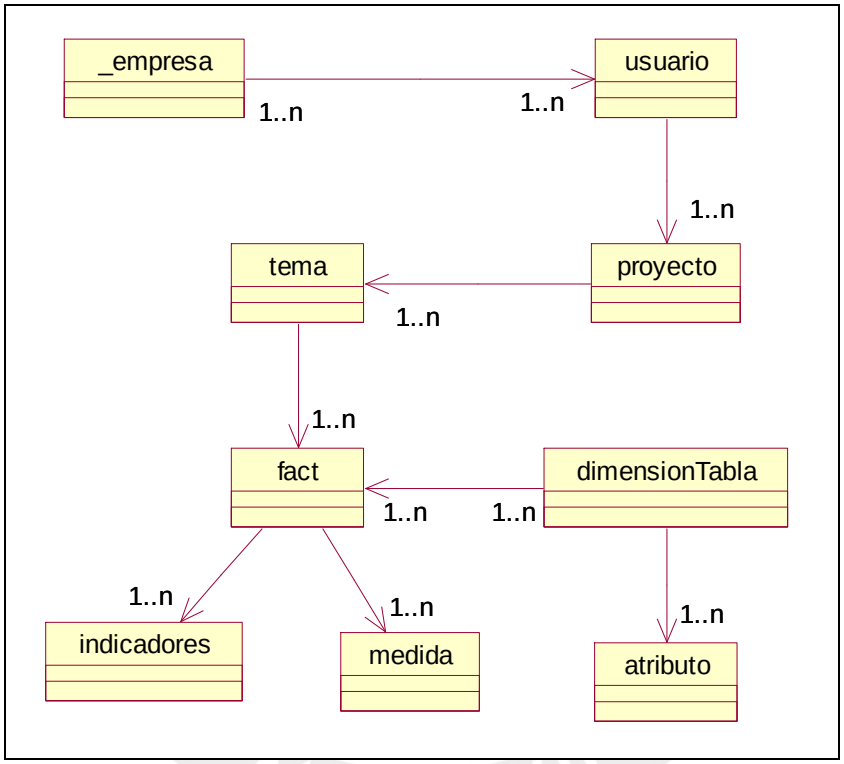
5.2. Componente: Control

Este componente contiene las clases necesarias para garantizar un acceso seguro a la herramienta, así como el framework usado en ésta, por último contiene el control web que es usado para poder interactuar con la herramienta.



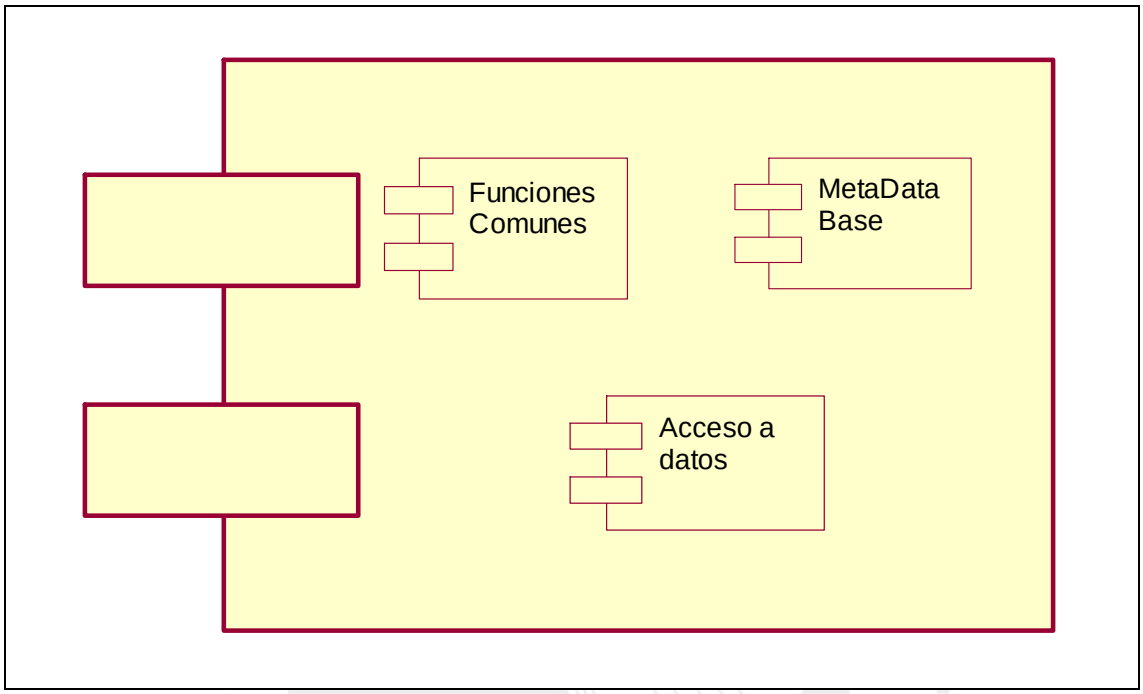
Aplicación	
Componente	**Descripción**
Control web	Componente utilizado para interactuar con la aplicación, implementa la interfaz web que permitirá el acceso a la lógica del negocio.
JSF Framework	Componente que maneja el flujo de navegación y comunicación con la aplicación. Implementa el patrón MVC.
Seguridad	Componente de seguridad y control de accesos. Implementará e manejo de LDAP.

Para entender mejor el diagrama de componentes anterior se tiene el siguiente diagrama de clases:



5.3. Componente: Común

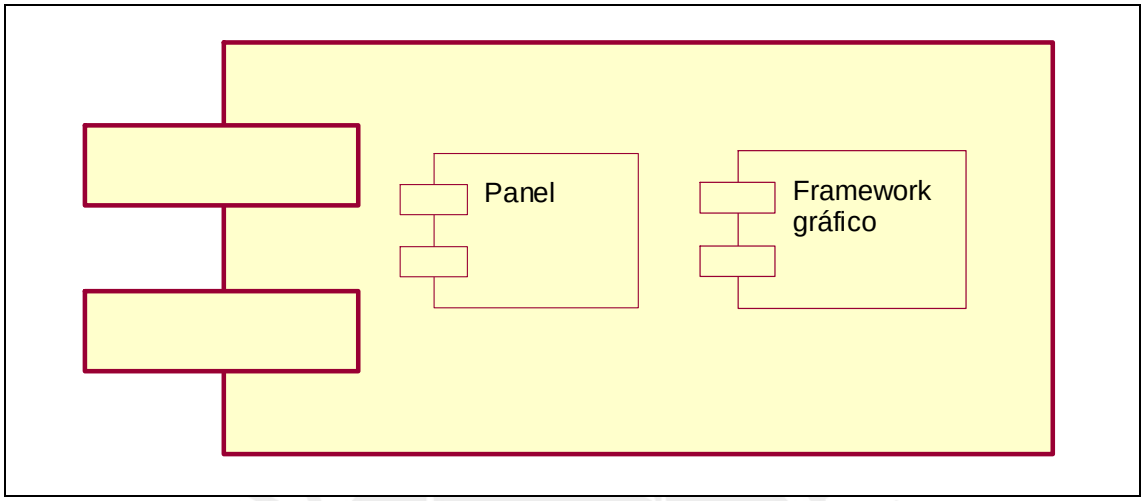
Este componente contiene todos los subcomponentes que se utilizan en todos los módulos de la herramienta.



Aplicación	
Componente	**Descripción**
Funciones Comunes	Este componente agrupa las clases que contienen funciones que son accedidas por todos los módulos. Ejemplo: El servlet de persistencia que se encarga de la grabación de datos en archivos XML utilizando el Castor.
MetaData Base	Este componente contiene los objetos de datos comunes a todos los módulos. Estas son las clases comunes como Tabla, Campo, Dimensión, etc.
Acceso a datos	Este componente permite a los módulos el acceso a bases de datos. Contiene los JARs con los drivers que permiten interactuar con manejadores de bases de datos y el control de conexiones a bases de datos.

5.4. Componente: Gráficos

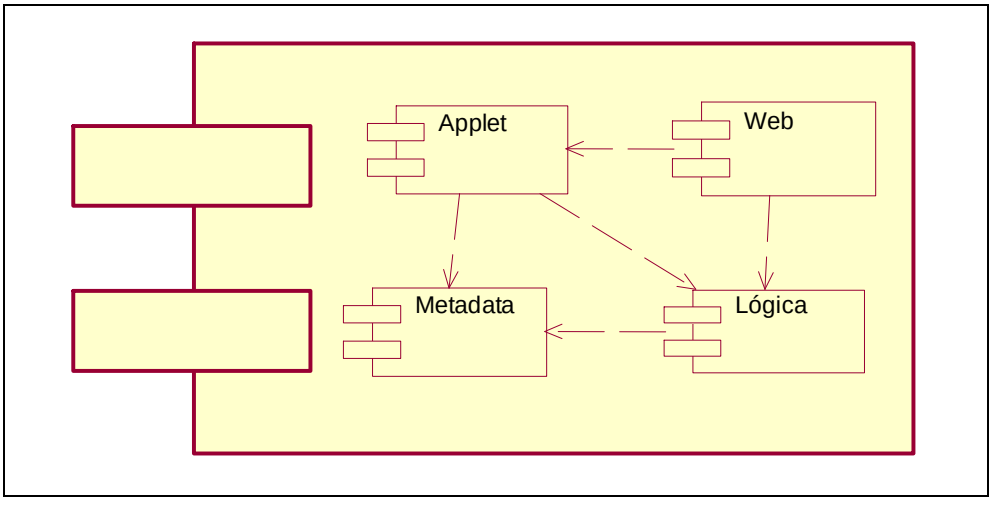
Este componente contiene todos los subcomponentes necesarios para realizar los gráficos en los applets.



Aplicación	
Componente	**Descripción**
Framework gráfico	Este componente contiene los elementos que representan los gráficos en el área de dibujo y tiene la implementación de acciones básicas sobre el área de dibujo como el drag and drop.
Panel	Este componente contiene el área donde se realizara el gráfico y la paleta de herramientas que permite seleccionar los elementos a graficar.

5.5. Componente: Análisis

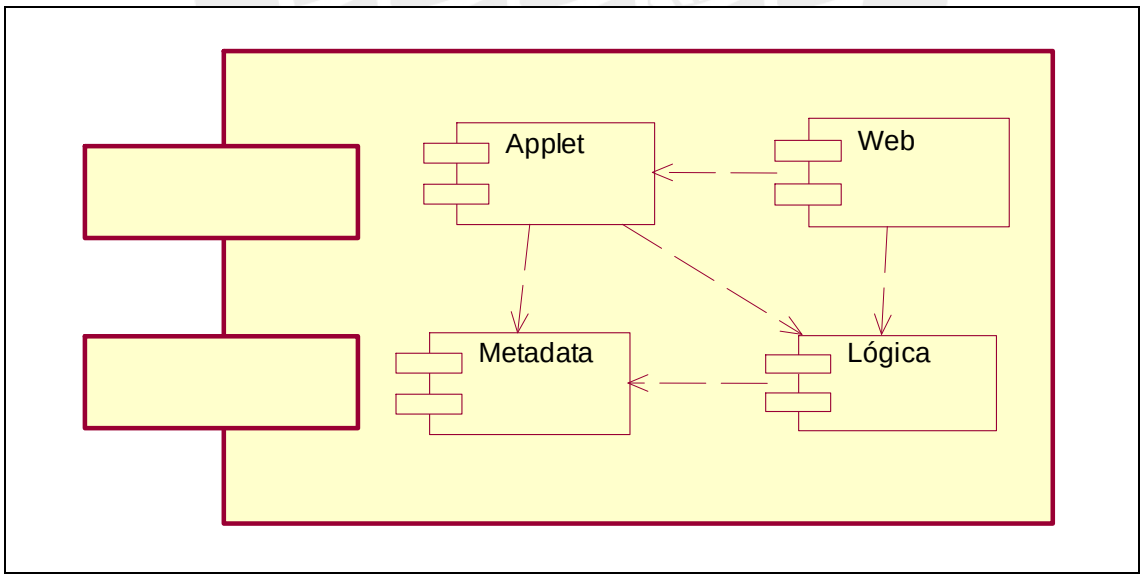
Este componente contiene todos los subcomponentes que se utilizan exclusivamente en el módulo de Análisis de la herramienta.



Aplicación	
Componente	**Descripción**
Applet	Este componente se descargará al equipo del cliente y permitirá realizar operaciones que gráficamente son más convenientes de realizar que empleando páginas JSF o web. El applet actuará con el sistema mediante servlets. Sus partes principales son el árbol de navegación de objetos y el área de dibujo. El modo de trabajo es muy similar al del módulo de Extracción.
Lógica	Este componente contiene las clases que tienen definida la lógica del negocio y que trabajan con la metadata. Ejemplo: El control de análisis.
Metadata	Este componente contiene los objetos que guardan la información de los elementos de análisis que hayan sido configurados por el usuario. Ejemplo: Reglas de Generación de Datos, Preferencias de Usuario, Definición de Tipos de Datos, etc. También contiene los beans de persistencia utilizados para enviar los datos de los objetos al servlet de persistencia el cual guarda estos datos en archivos XML.
Web	Este componente esta conformado por los servlets y paginas JSF alojadas en el servidor.

5.6. Componente: Extracción

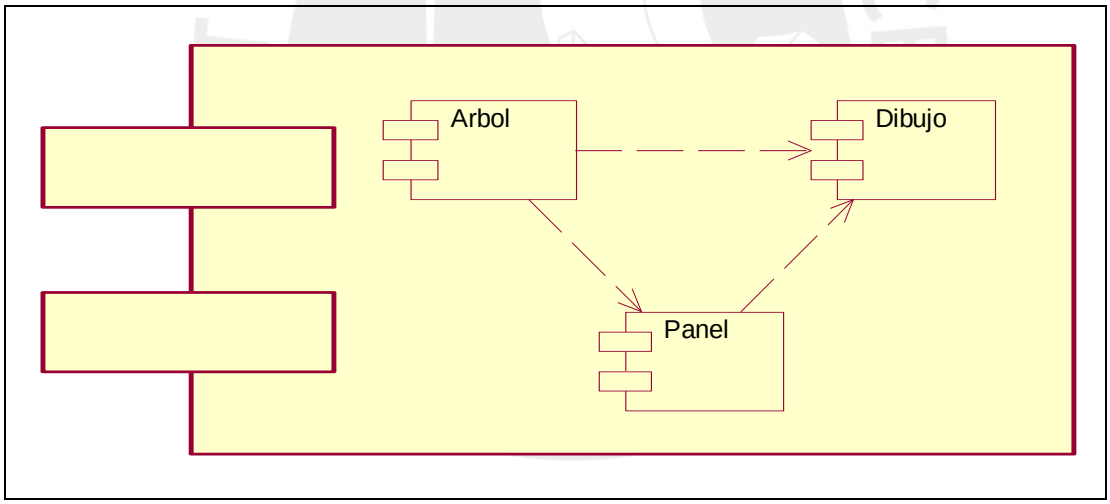
Este componente contiene todos los subcomponentes que se utilizan en el módulo de extracción de la herramienta.



Aplicación	
Componente	**Descripción**
Applet	Este componente se descargará al equipo del cliente y permitirá realizar operaciones que requieren uso de gráficos que no puedan ser realizadas por medio de formularios web o paginas JSF. Interactuará con un servlet ubicado en el lado del servidor, el cual recibirá los datos de las operaciones realizadas en el applet. Sus partes principales son el árbol de navegación de objetos y el área de dibujo.
Lógica	Este componente contiene las clases que tienen definida la lógica del negocio y que trabajan con la metadata. Ejemplo: El control de extracción y el ejecutor de jobs programados.
Metadata	Este componente contiene las clases de dominio, que se usan para la lógica del negocio, y las clases que representan el mapeo correspondiente para su persistencia y recuperación en cualquier momento.
Web	Este componente esta conformado por los servlets, las páginas JSF alojadas en el servidor, los beans que representan las páginas y los archivos de recursos.

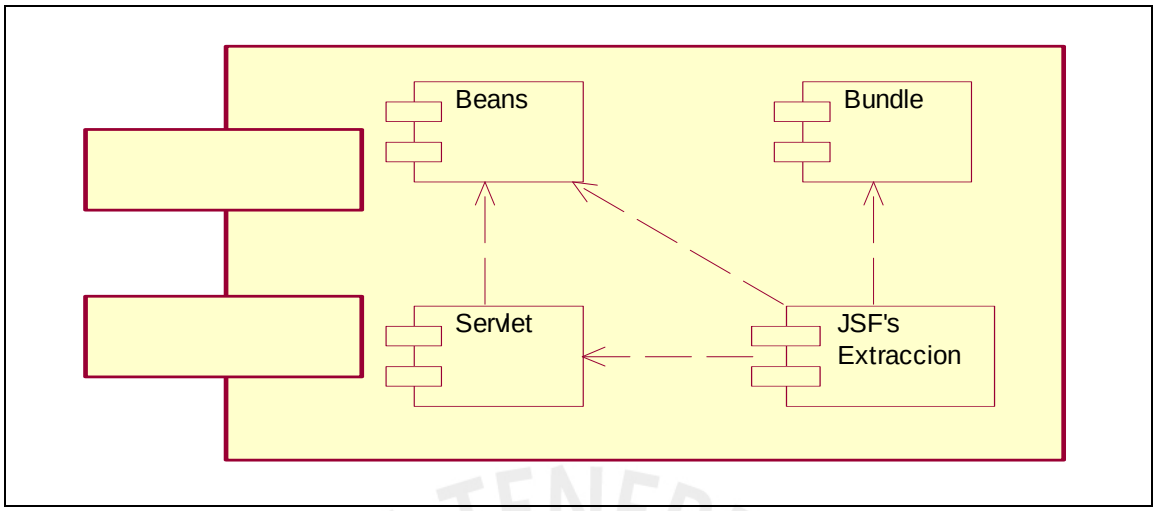
Estos subcomponentes se subdividen según lo siguiente:

5.6.1. Componente: Applet



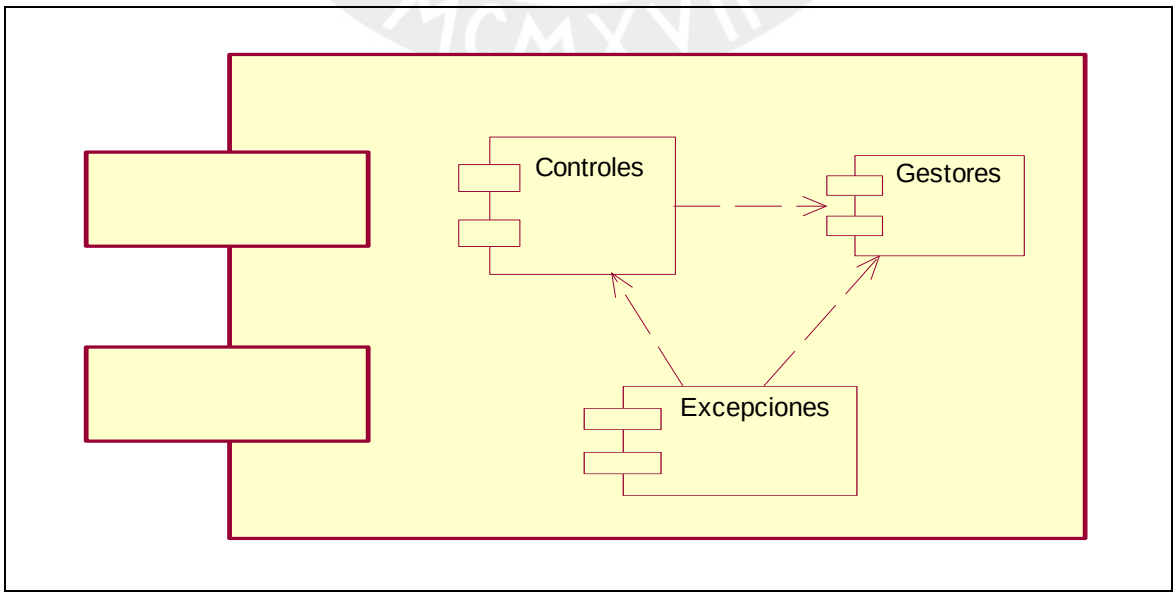
Aplicación	
Componente	**Descripción**
Arbol	Este componente contiene las clases necesarias para manejar el árbol de navegación de objetos que contendrá la estructura de los proyectos que se muestra en el applet. Se encarga de definir los menús y las operaciones que realiza el usuario en dicha área.
Dibujo	Este componente contiene las clases que definen como se dibujarán los objetos sobre el área de dibujo y las operaciones a realizar con ellos. Ejemplo: La clase del objeto activo Script : BEScriptDib
Panel	Este componente contiene los objetos que controlan la forma como se muestra el área de dibujo, por ejemplo se encarga de la funcionalidad del deshacer y rehacer con las acciones que realiza el usuario sobre el área de dibujo en el applet.

5.6.2. Componente: Web



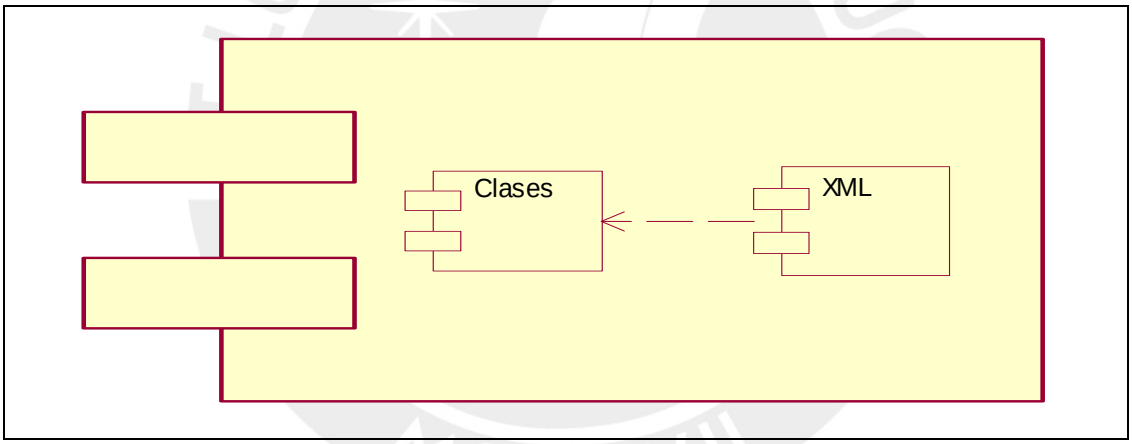
Aplicación	
Componente	**Descripción**
Beans	Este componente contiene las clases de los beans de respaldo de las páginas JSF. Estos beans definen las propiedades y las lógicas de manejo asociadas con los componentes de interfaz gráfica utilizados en la página.
Servlet	Este componente contiene las clases que definen las operaciones que debe realizar el servidor en la interacción con las páginas JSF. Ejemplo: El servlet de extracción aparte de ejecutar las programaciones, recibe solicitudes de las páginas JSF para saber el estado de ejecución de los jobs.
Bundle	Este componente contiene los archivos de recursos de las páginas JSF, por ejemplo para la internacionalización de los mensajes.
JSF's Extraccion	Este componente contiene las páginas JSF, que se mostrarán del lado del cliente en el navegador Web.

5.6.3. Componente: Lógica



Aplicación	
Componente	**Descripción**
Controles	Este componente contiene las clases que se encargarán de controlar las operaciones y la comunicación entre las distintas clases, que se realizan en la capa de la lógica del negocio en el sistema. Ejemplo: BEControlEjecución, clase que maneja la ejecución de una transformación.
Gestores	Este componente contiene las clases que manejan la lógica de los objetos del dominio. Ejemplo: BEGestorJob, clase que maneja las operaciones a realizar sobre los jobs que pertenecen a un proyecto.
Excepciones	Este componente contiene las clases de las excepciones personalizadas que se manejan en el sistema para controlar los distintos errores que podría presentar la aplicación.

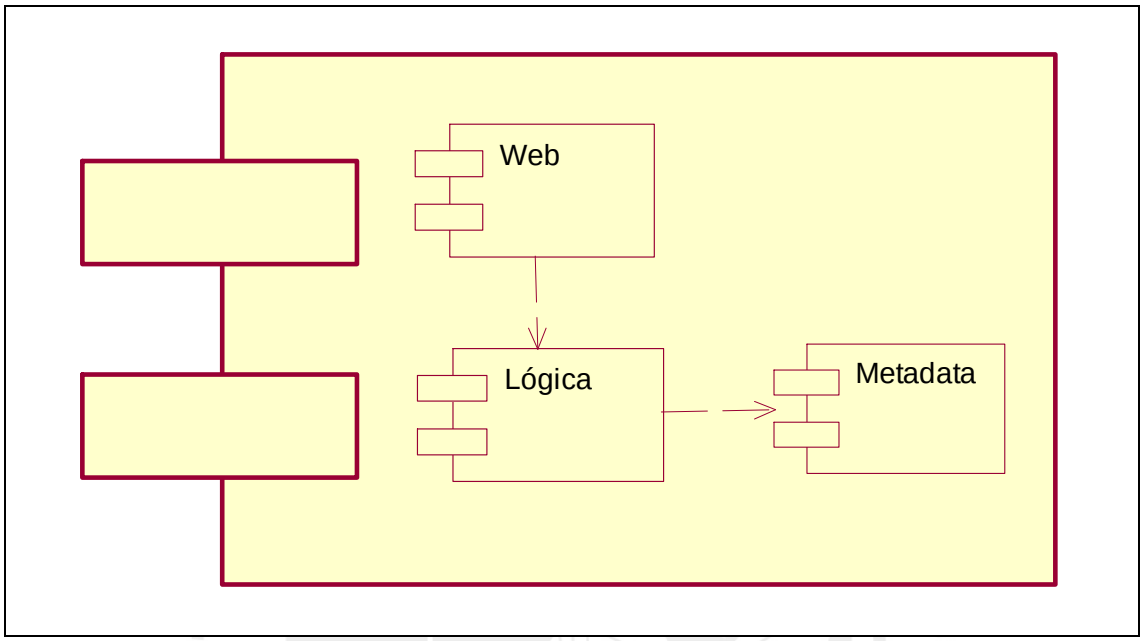
5.6.4. Componente: Metadata



Aplicación	
Componente	**Descripción**
Clases	Este componente contiene los objetos que guardan la información de los elementos de extracción que hayan sido configurados por el usuario. Ejemplo: clases de datos de extracción tales como Transformación, Fuente de Datos, etc.
XML	Este componente contiene los beans de persistencia utilizados para enviar los datos de los objetos al servlet de persistencia el cual guarda estos datos en archivos XML.

5.7. Componente: Explotación

Este componente contiene todos los subcomponentes que se utilizan en el módulo de explotación de la herramienta.



Aplicación	
Componente	**Descripción**
Lógica	Este componente contiene las clases que tienen definida la lógica de la explotación y que trabajan con la metadata. Esto incluye el manejo de reportes, gráficos (charts) y las clases encargadas de comunicarse con el motor OLAP. También contiene los beans de persistencia utilizados para enviar los datos de los objetos al servlet de persistencia el cual guarda estos datos en formato XML.
Metadata	Este componente contiene los objetos que guardan la información de los elementos de explotación configurados por el usuario. Incluye información de cubos y objetos de explotación (esquemas, reportes, filtros, etc).
Web	Este componente esta conformado por los servlets y paginas JSF alojadas en el servidor.

6. Requerimientos de desempeño

La arquitectura de software del sistema satisface los requerimientos clave de tamaño y desempeño:

- El sistema permitirá que las transacciones que se realicen sean seguras, de modo que ningún ente externo al sistema pueda acceder a la información que éstas manejen. Para tal fin se definirán varios niveles de acceso.

- El sistema permitirá el acceso concurrente de varios usuarios, de modo que puedan realizar transacciones simultáneas.

- Las computadoras personales del lado del cliente deberán contar con un procesador Pentium III, 128 MB de memoria RAM, acceso a Internet y un navegador Web.

- El módulo de extracción requerirá de un mínimo de 1Gigabyte de RAM en el servidor principal. Los requerimientos del servidor se incrementarán de acuerdo con la capacidad que se quiera dar al mismo, es decir, cuantos flujos simultáneos se necesita que pueda ejecutar.

ANEXO D. DOCUMENTO DE XML

1. Introducción

1.1 Objetivos

El presente documento forma parte del diseño final de la herramienta AXEbit, en éste se muestra el esquema XML usado para almacenar la información persistente requerida dentro de la herramienta. Se provee la descripción de la estructura de los archivos XML usados en la herramienta AXEbit, de esta forma se logrará transmitir una visión más clara sobre la forma en que se trabajo la construcción de la herramienta.

1.2 Alcance

El documento describirá el esquema XML usado dentro de la herramienta AXEbit para almacenar la información, indicando las carpetas en las cuales se distribuirá los archivos y la estructura de un archivo de ejemplo.

1.3 Visión general del documento

Este documento consta de dos secciones. Esta sección es la introducción y proporciona una visión general del Documento de XML. En la siguiente sección se detalla el esquema usado en la herramienta para el almacenamiento de información.

2. Esquema XML

En esta sección se muestra el detalle de la estructura usada para almacenar la información en archivos XML, comenzando por la estructura de carpetas y mostrando además el contenido de un archivo XML de ejemplo.

2.1 Estructura de carpetas XML

Las carpetas usadas para almacenar los archivos XML son las siguientes:

jobs
fuentedatos
paquetes
procesos
scripts
archivos
tablas
proyectos
empresas
logs
fusuario

programación

- **Jobs:** Contiene los archivos XML de los jobs configurados en la herramienta
- **FuenteDatos:** Contiene los archivos XML que describen las fuentes de datos configuradas en la herramienta.
- **Paquetes:** Contiene los archivos XML que describen los paquetes configurados en la herramienta, incluyendo referencias a los componentes internos del mismo (objetos activos y objetos pasivos)
- **Procesos:** Contiene los archivos XML que describen los componentes activos tipo proceso que pertenecen a los paquetes de la herramienta. Cada XML contiene además la información de los subcomponentes configurados para cada proceso. (filtro, transformación, estandarización y lookup)
- **Scripts:** Contiene los archivos XML que describen los scripts configurados en la herramienta.
- **Archivos:** Contiene los archivos XML que describen los archivos planos que sirven como origen y destino de datos para los paquetes configurados en la herramienta.
- **Tablas:** Contiene los archivos XML que describen las tablas de base de datos que sirven como origen y destino de datos para los paquetes configurados en la herramienta.
- **Proyectos:** Contiene los archivos XML que describen los proyectos de inteligencia de negocios configurados en la empresa.
- **Empresas:** Contiene los archivos XML que describen a las empresas con las cuales se trabaja en la herramienta.
- **Logs:** Contiene los archivos XML que contienen la información de ejecución de los jobs de extracción.
- **Fusuario:** Contiene los archivos XML que describen las funciones definidas por el usuario en la herramienta.
- **Programacion:** Contiene los archivos XML que contienen la información de programación para la ejecución de los jobs de la herramienta.

2.2 Ejemplo archivo XML

```
<?xml version="1.0" encoding="UTF-8" ?>
- <BEFuenteDatosBean tipo-bD="0" id="1154479273390113">
<base-datos>axe</base-datos>
<contra>biex</contra>
<usuario>biex</usuario>
<id-tablas xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="java:java.lang.String">1154479284078649</id-tablas>
<id-tablas xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="java:java.lang.String">1154479284562499</id-tablas>
<id-tablas xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="java:java.lang.String">1154479284593474</id-tablas>
<id-tablas xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="java:java.lang.String">1154479284609509</id-tablas>
<cadena-conexion>localhost</cadena-conexion>
<descripcion>maparam</descripcion>
<fecha-modificacion>2006-08-01T19:41:13.390-05:00</fecha-modificacion>
<fecha-creacion>2006-08-01T19:41:13.390-05:00</fecha-creacion>
<nombre>fdParam</nombre>
</BEFuenteDatosBean>
```

Todos los archivos XML tendrán como mínimo los siguientes atributos:

- id
- nombre
- descripcion
- fechaCreacion
- fechaModificacion

ANEXO E. DOCUMENTO DE ESTÁNDARES DE PROGRAMACIÓN

1. Introducción

1.1 Objetivos

El presente documento ha sido elaborado con la finalidad de presentar los estándares y buenas prácticas que se usaron en la implementación de la herramienta AXEbit.

1.2 Alcance

El presente documento de estándares de programación es aplicable no sólo al módulo de Extracción, sino a los otros dos módulos involucrados en el desarrollo de AXEbit. Los estándares aquí presentados han sido considerados los necesarios para hacer un código entendible por los miembros del equipo y por ajenos al mismo que requieran revisarlo. El estándar presentado será aplicado por todos los miembros del equipo de desarrollo para todas las funcionalidades que se presenten.

2. Estándares de programación

Los siguientes son los estándares aplicados dentro del desarrollo de la herramienta:

2.1 Sobre variables, métodos, constantes y excepciones

2.1.1 Nombres de variables

Los nombres de las variables cumplirán con las siguientes características:

- Ser descriptivos.
- No serán muy largos, pero se entenderá a simple vista la información que guardan.
- La primera palabra estará en minúsculas. Si contienen varias palabras la primera letra de la segunda palabra será en mayúsculas.
  Ejemplos: totalObjetos, objetosSelecc, numProyectos, etc.

2.1.2 Nombres de métodos

Los nombres de los métodos cumplirán con las siguientes características:

- Se respetarán las características de las variables.
  Ejemplos: actualizarFact, eliminarObjeto, etc.

2.1.3 Constantes

Los nombres de las constantes cumplirán con las siguientes características:

- Estarán exclusivamente en mayúsculas.
- En caso se tenga una constante compuesta por más de una palabra, estas estarán separadas por el carácter: "_".
  Ejemplos: NUM_FACT, TOTAL_OBJETOS, etc.

2.1.4 Excepciones

Los nombres de las excepciones cumplirán con las siguientes características:

- Los métodos que realicen algún procedimiento de resultado esperado, serán del tipo void y lanzarán excepciones específicas y personalizadas. Ejemplo: Con el método "buscarDimension(dimension)" se espera encontrar un objeto dimensión. En caso no se encuentre el objeto esperado, se lanzara una excepción personalizada (por ejemplo BuscarDAOException).
- Las excepciones personalizadas serán clases que heredan de Exception, y se personalizara el ExceptionMessage devuelto.
- Las excepciones personalizadas serán creadas a nivel de funcionalidad (excepción de procesamiento de BD, error lógico, etc.).
  Ejemplo: DAOException (excepción a nivel de BD): Buscar DAOException.
  XMLException (excepción de persistencia): CrearXMLException, BuscarXMLException, ActualizarXMLException, etc.

2.2 Generalidades

- Se usara Java 1.5, el estándar de programación base será el de java (propuesto por Sun), sobre ello se usara el formato de Eclipse (reglas de indentación, espaciado, etc). Para acceder a dicho formato basta con: hacer click derecho->Source->Format. desde el IDE.
- Se utilizarán los siguientes plugins:
  - Amateras Faces IDE, Amateras HtmlEditor, version 2.0.0.
- Se utilizarán las siguientes implementaciones:
  - Apache
  - MyFaces Tomahawk, implementación de componentes JSF, como trees, tabbedPanes y calendars.
  - Apache Tiles, antes struts-tiles, para la separación de zonas en las páginas.

- Un archivo fuente sólo contendrá una clase (además de inner classes que puedan existir). Las clases serán siempre públicas, los atributos serán protected cuando haya herencia (clase padre), private en cualquier otro caso.
- Para listas de objetos no se usará la clase vector, se usara la interface List, la cual se instanciará como ArrayList. Por lo general, las listas llevarán el prefijo "lista" (ej: listaDimensiones) o algo que lo identifique como una agrupación.
- La lógica de los servlets se hará en un método personalizado que se llamará doAction, el cual será llamado desde el doPost() y/o doGet().
- Los beans de persistencia heredarán de la clase XMLBean, la cual implementará Serializable (flag para un correcto marshalling).
- Sobre los comentarios:
    - Se utilizarán comentario JavaDoc y comentarios normales.
    - Los comentarios JavaDoc se harán siempre, tanto a nivel de clase, métodos y constantes (comentarios para la documentación de las clases implementadas). Para detalles, ver la especificación de JavaDocs, desde el IDE: click derecho ->source -> add java comment.
    - Se utilizarán comentarios dentro de la implementación de métodos (comentarios de desarrollo) cuando haya algo que especificar (no es obligatorio comentar toda la implementación).
    - Al inicio de cada clase se colocara el autor, fecha de creación y a lista de las fechas de modificación (acompañadas del responsable y de una breve descripción de lo que se ha ido modificando).
- Se usarán archivos properties (bundle). Estos archivos serán usados para todo texto inserto en las páginas JSF utilizadas (labels, títulos, subtítulos, encabezados).
- El uso de scriptlets (<% %>) sólo se hará cuando sea absolutamente necesario y no exista otra posibilidad de desarrollo.
- Los import se harán hasta la clase específica, no se usará el formato: import paquete.*;
- Sobre los títulos de las paginas, se usará el nombre del proyecto seguido de la funcionalidad, "AXEbit - Funcionalidad".
- Ejemplo: AXEbit – Crear Fuente de Datos.
- Las hojas de estilo y javascript se referenciarán externamente, usando link para los css y <script src=".."> para los javascript. No se colocarán estilos o scripts embebidos en las páginas, a menos que sea algo muy específico.
- Los tipos enumerados no llevaran prefijos.

ANEXO G. DOCUMENTO DE ESTIMACIÓN

1. Introducción

1.1. Objetivos

El presente documento tiene como objetivo mostrar el esfuerzo previsto para la realización del proyecto en su segunda etapa. Teniendo como experiencia previa la primera etapa en la cual la herramienta tenia una arquitectura cliente-servidor.

1.2. Alcance

El presente documento de estimación se aplicará al módulo de Extracción de AxeBit tomando en consideración la segunda etapa de la realización de la herramienta, la misma que es descrita a lo largo de todo el documento de tesis.

Las técnicas empleadas para la estimación son las siguientes: Puntos de función y Cocomo II.

1.3. Definiciones y acrónimos

Acrónimo	Significado
Puntos de función	
EI	Entrada externa, datos o información de control que procede de fuera de los límites de la aplicación.
EO	Salida externa, datos o información de control que sale fuera de los límites de la aplicación mediante un proceso lógico.
EQ	Consulta externa, procesos elementales que obtienen una combinación de entrada/salida como resultado de una recuperación de datos, pero no mantienen ningún ILF.
ILF	Fichero lógico interno, son grupos de datos lógicamente relacionados identificables por el usuario, o información de control contenida dentro de los límites de la aplicación.
EIF	Fichero de interfaz externo, información de control utilizada en la aplicación pero mantenida por medio de otra.
DET	Elemento de tipo dato, campo no recursivo y reconocible por el usuario.
RET	Elemento de tipo registro, subgrupo de DETs reconocibles por el usuario.
FTR	Tipo de fichero referenciado, fichero al que una transacción hace referencia.
PFSA	Puntos de función sin ajustar.
Cocomo II	
Factor Producto	

RELY	Fiabilidad del producto requerida.
DATA	Tamaño de la base de datos.
CLPX	Complejidad del producto.
RUSE	Reutilización requerida.
DOCU	Adecuación de la documentación a las necesidades del ciclo de vida.
Factor Plataforma	
TIME	Limitaciones en el tiempo de ejecución.
STOR	Limitaciones en el almacenamiento principal.
PVOL	Volatilidad de la plataforma.
Factor Personal	
ACAP	Capacidad de los analistas.
AEXP	Capacidad del programador.
PCAP	Continuidad del personal.
PEXP	Experiencia en aplicaciones.
LTEX	Experiencia en la plataforma
PCON	Experiencia con el lenguaje y las herramientas.
Factor Proyecto	
TOOL	Uso de herramientas software.
SITE	Desarrollo en varios sitios.
SCED	Calendario de desarrollo requerido.

2. Determinación de puntos de función

Con el fin de obtener el cálculo de puntos de función se determinarán los siguientes puntos: archivos internos lógicos, entradas externas, consultas externas y salidas externas.

2.1. Archivos Internos Lógicos

Se determinan los DETs y RETs de cada uno de los IFL. El resultado de este cálculo se muestra a continuación:

Archivos Internos Lógicos	
ILF	**DET**
Archivo	ruta extension tipo nombres ancho lonAncho caracterCampo caracterRegistro append valido nCampo nRegistro nombreArchivo
Filtro	SentenciasFiltro tipo baseDatos cadenaConexion usuario contra ruta idTablas esquema idBD listaCampos
FuenteDatos	tipo baseDatos cadenaConexion usuario contra ruta idTablas
Job	prioridad List paquetes listaParametros correoOK correoERR
Log	idJob fechaIni fechaFin lineas numReg tiempoTotal rendimiento
Paquete	prioridad tema componentes fuenteDatos funciones diagrama guardaLog
Script	script fuenteDatos prioridad tipoScript
Tabla	esquema idBD listaCampos

Transformación	SentenciasTransformacion tipo baseDatos cadenaConexion usuario contra ruta idTablas esquema idBD listaCampos
Parámetro	nombre etiqueta valor

En el sistema planteado no encontramos la presencia de EIF, ya que no se contará con aplicaciones externas que interactúan en la aplicación.

Usando el resultado de la tabla anterior y la tabla de cálculo de complejidad para ILF (*ver anexo 4.1*) se obtendrá a complejidad de cada uno de ellos.

Calculo de Complejidad			
ILF	**#DET**	**#RET**	**Complejidad**
Archivos	13	1	BAJA
Filtros	11	3	BAJA
Fuentes de datos	5	2	BAJA
Jobs	7	3	BAJA
Logs	7	3	BAJA
Paquetes	4	3	BAJA
Scripts	3	2	BAJA
Tablas	1	2	BAJA
Transformaciones	11	3	BAJA
Parámetro	3	1	BAJA

A continuación se realizará el cálculo de los puntos de función para los ILF (*ver anexo 4.5*).

Puntos de Función			
Tipo de Función	**Complejidad Funcional**	**Totales de Complejidad**	**Totales de Tipo de Función**
ILF	10 BAJA x 7	70	70
	0 MEDIA x 10	0	
	0 ALTA x 15	0	

2.2. Entradas Externas

Se determinarán las entradas externas para los casos de uso del sistema.

Entradas Externas	
Casos de Uso	**Entrada Externa**
Configurar Fuente de Datos	Crear Fuente de Datos Modificar Fuente de Datos Eliminar Fuente de Datos
Conectar a fuente de datos	Conectar Fuente de Datos
Obtener estructuras y objetos de fuente de datos	Obtener estructuras Obtener objetos de fuente de datos
Configurar estructura de archivo plano	Configurar archivo Configurar tipo de datos de registros
Crear Paquete	Crear Paquete Configurar Flujo Eliminar Paquete
Crear Job	Crear Job Modificar Job Eliminar Job
Configurar Parámetros	Configurar Parámetros
Crear flujo de transformación	Crear flujo de transformación
Personalizar transformación usando objetos activos	Configurar flujo de transformación con objetos activos
Validar Scripts mediante Parser	Validar Scripts mediante Parser
Configurar Orden de Ejecución de Paquetes	Configurar Orden de Ejecución de Paquetes
Crear funciones definidas por el usuario	Crear funciones definidas por el usuario
Programar Job	Programar Job

Usando la tabla anterior y la tabla de cálculo de complejidad para EI (*ver anexo 4.2*) se obtendrá las DETs y FTRs como la complejidad de cada uno de ellos.

Calculo de Complejidad					
Entrada Externa	**DET**	**FTR**	**#DET**	**#FTR**	**Comp.**
Crear Fuente de Datos	tipo baseDatos cadenaConexion usuario contra ruta idTablas	FuenteDatos	7	1	BAJA
Modificar Fuente de Datos	tipo baseDatos cadenaConexion usuario contra ruta idTablas	FuenteDatos	7	1	BAJA
Eliminar Fuente de Datos	tipo baseDatos cadenaConexion usuario contra ruta idTablas	FuenteDatos	7	1	BAJA

Conectar Fuente de Datos	tipo baseDatos cadenaConexion usuario contra ruta idTablas	FuenteDatos	7	1	BAJA
Obtener estructuras	tipo baseDatos cadenaConexion usuario contra ruta idTablas esquema idBD listaCampos	Tabla FuenteDatos	10	2	MEDIA
Obtener objetos de fuente de datos	tipo baseDatos cadenaConexion usuario contra ruta idTablas esquema idBD listaCampos	Tabla FuenteDatos	10	2	MEDIA
Configurar archivo	ruta extension tipo nombres ancho lonAncho caracterCampo caracterRegistro append valido nCampo nRegistro nombreArchivo	Archivo	13	1	BAJA
Configurar tipo de datos de registros	ruta extension tipo nombres ancho lonAncho caracterCampo caracterRegistro append valido nCampo nRegistro nombreArchivo	Archivo	13	1	BAJA
Crear Paquete	prioridad tema componentes fuenteDatos funciones diagrama guardaLog	Paquete	7	1	BAJA

Configurar Flujo	prioridad tema componentes fuenteDatos funciones diagrama guardaLog SentenciasFiltro SentenciasTransformación prioridad List paquetes listaParametros correoOK correoERR	Paquete Filtro Transformación Job	14	4	ALTA
Eliminar Paquete	prioridad tema componentes fuenteDatos funciones diagrama guardaLog	Paquete	7	1	BAJA
Crear Job	prioridad List paquetes listaParametros correoOK correoERR	Job	5	1	BAJA
Modificar Job	prioridad List paquetes listaParametros correoOK correoERR	Job	5	1	BAJA
Eliminar Job	prioridad List paquetes listaParametros correoOK correoERR	Job	5	1	BAJA
Configurar Parámetros	nombre etiqueta valor	Parámetro	3	1	BAJA
Crear flujo de transformación	Sentencias Transformación tipo baseDatos cadenaConexion usuario contra ruta idTablas esquema idBD listaCampos	Transformación FuenteDatos Tabla	11	3	ALTA

Configurar flujo de transformación con objetos activos	Sentencias Transformación tipo baseDatos cadenaConexion usuario contra ruta idTablas esquema idBD listaCampos	Transformación FuenteDatos Tabla	11	3	ALTA
Validar Scripts mediante Parser	script fuenteDatos prioridad tipoScript	Script FuenteDatos	4	2	BAJA
Configurar Orden de Ejecución de Paquetes	prioridad tema componentes fuenteDatos funciones diagrama guardaLog	Paquete	7	1	BAJA
Crear funciones definidas por el usuario	nombre descripcion aplicableA	FuncionPred	3	1	BAJA
Programar Job	prioridad List paquetes listaParametros correoOK correoERR	Job	5	1	BAJA

A continuación se realizará el cálculo de los puntos de función para las EI (*ver anexo 4.5).*

Puntos de Función			
Tipo de Función	**Complejidad Funcional**	**Totales de Complejidad**	**Totales de Tipo de Función**
EI	16 BAJA x 3	45	74
	2 MEDIA x 4	8	
	3 ALTA x 6	18	

2.3. Consultas Externas

Se determinarán las consultas externas para cada uno de los casos de uso del sistema.

Salidas Externas	
Casos de Uso	**Salida Externa**
Visualizar Log de Ejecución	Log de Ejecución

Usando la tabla anterior y la tabla de cálculo de complejidad para EQ (*ver anexo 4.4*) se obtendrá las DETs y FTRs como la complejidad de cada uno de ellos.

Calculo de Complejidad					
Consulta Externa	**DET**	**FTR**	**#DET**	**#FTR**	**Comp.**
Visualizar Log de Ejecución	idJob fechaIni fechaFin lineas numReg tiempoTotal rendimiento	Log	7	1	BAJA

A continuación se realizará el cálculo de los puntos de función para las EQ (*ver anexo 4.5*).

Puntos de Función			
Tipo de Función	**Complejidad Funcional**	**Totales de Complejidad**	**Totales de Tipo de Función**
EQ	1 BAJA x 4	4	4
	0 MEDIA x 5	0	
	0 ALTA x 7	0	

2.1. Salidas Externas

Se determinarán las salidas externas para cada uno de los casos de uso del sistema.

Salidas Externas	
Casos de Uso	**Salida Externa**
Ejecutar Job	Ejecutar Job

Usando la tabla anterior y la tabla de cálculo de complejidad para EO (*ver anexo 4.4*) se obtendrá las DETs y FTRs como la complejidad de cada uno de ellos.

Calculo de Complejidad					
Salida Externa	**DET**	**FTR**	**#DET**	**#FTR**	**Comp.**
Ejecutar Job	prioridad correoOK correoERR prioridad tema componentes fuenteDatos funciones diagrama guardaLog tipo baseDatos cadenaConexion usuario contra ruta idTablas nombre etiqueta valor	Job Paquete Parámetro FuenteDatos	20	4	ALTA

A continuación se realizará el cálculo de los puntos de función para las EO (*ver anexo 4.5*).

Puntos de Función			
Tipo de Función	**Complejidad Funcional**	**Totales de Complejidad**	**Totales de Tipo de Función**
EO	0 BAJA x 4	4	7
	0 MEDIA x 5	0	
	1 ALTA x 7	0	

Realizando la sumatoria de todos los tipos de puntos de función se tiene:

Tipo Punto de Función	Cantidad PF
ILF	74
EI	68
EQ	4
EO	7
Total (sumatoria)	**153**

Según la tabla de transformación (*ver anexo 4.6*) para el lenguaje Java que será utilizado en la realización de este proyecto, se tienen 53 líneas de código por punto de función. Haciendo los cálculos respectivos tenemos:

53 LCF/PF * 153 PF = 8109 LCF = 8.109 KLCF

3. Modelo de estimación Cocomo II

3.1. Factores del proyecto

Criterios de aplicación	
Factor escala	**Valor**
Precedencias	2.48
Flexibilidad de desarrollo	4.05
Arquitectura / Solución de riesgo	2.83
Cohesión del equipo / Interacción	3.29
Madurez del proceso	4.68
Total (sumatoria)	**17.33**

3.2. Multiplicadores de esfuerzo

Multiplicadores de esfuerzo	
Factor criterio de costo PRODUCTO	**Valor**
RELY	0.82
DATA	1.00
CLPX	1.17
RUSE	1.00
DOCU	1.00
Total (productoria)	**0.9594**

Multiplicadores de esfuerzo	
Factor criterio de costo PERSONAL	**Valor**
ACAP	0.71
AEXP	1.00
PCAP	0.88
PEXP	1.09
LTEX	1.09
PCON	0.81
Total (productoria)	**0.6013**

Multiplicadores de esfuerzo	
Factor criterio de costo PLATAFORMA	**Valor**
TIME	1.00
STOR	1.00
PVOL	1.00
Total (productoria)	**1.00**

Multiplicadores de esfuerzo	
Factor criterio de costo PROYECTO	**Valor**
TOOL	0.78
SITE	1.00
SCED	1.00
Total (productoria)	**0.78**

3.3. Determinación del esfuerzo requerido

Con los datos obtenidos se puede finalmente encontrar el Esfuerzo, a continuación se muestran los cálculos finales:

Exponente = 0.91 + 0.01 * (∑ Factores) = 0.91 + 0.01*17.33 = 1.0833

Esfuerzo = 2.94 * (KLOC) $^{\text{EXPONENTE}}$ * (∏ Multiplicadores)

= 2.94 * $(8.109)^{1.0833}$ * (0.9594*0.6013*1.00*0.78)

= 2.94 * 9.6535 * 0.44997203

= 2.94 * $(8.109)^{1.0833}$ * 0.44997203

= 2.94 * $(8.109)^{1.0833}$ * 0.44997203

= 12.77 meses hombre

Debido a que el equipo de trabajo esta constituido por 5 personas, el esfuerzo por persona sería el siguiente:

ESFUERZO = 2.554 meses por integrante

Lo cual constituye tres meses de desarrollo por integrante.

4. Tablas referenciales

4.1. Complejidad de un ILF o EIF

Complejidad de un IFL o EIF			
	1 a 19 DET	**20 a 50 DET**	**51 a más DET**
1 RET	BAJA	BAJA	MEDIA
2 a 5 RET	BAJA	MEDIA	ALTA
6 o más RET	MEDIA	ALTA	ALTA

4.2. Complejidad de un EI

Complejidad de un EI			
	1 a 4 DET	**5 a 15 DET**	**16 o más DET**
0 a 1 FTR	BAJA	BAJA	MEDIA
2 FTR	BAJA	MEDIA	ALTA
3 o más FTR	MEDIA	ALTA	ALTA

4.3. Complejidad de un EQ

Complejidad de un EQ			
	0 a 5 DET	6 a 19 DET	20 o más DET
0 a 1 FTR	BAJA	BAJA	MEDIA
2 a 3 FTR	BAJA	MEDIA	ALTA
4 o más FTR	MEDIA	ALTA	ALTA

4.4. Complejidad de un EO

Complejidad de un EO			
	1 a 5 DET	6 a 19 DET	20 o más DET
0 a 1 FTR	BAJA	BAJA	MEDIA
2 a 3 FTR	BAJA	MEDIA	ALTA
4 o más FTR	MEDIA	ALTA	ALTA

4.5. Tabla de Transformación

Tabla de Transformaciones					
	IFL	EIF	EI	EQ	EO
BAJA	7	5	3	3	4
MEDIA	10	7	4	4	5
ALTA	15	10	6	6	7

4.6. Tabla de Transformación LCF/PF

Tabla de Transformaciones	
Lenguaje	LCF/PF
Ensamblador	320
C	150
Cobol	106
Pascal	91
Basic	64
TCL	64
Java	53
C++	29

ANEXO H. DOCUMENTO DE PRUEBAS

1. Introducción

Este documento es Plan de Pruebas de Software para la herramienta AXEbit, módulo de extracción y contiene los casos de prueba que se van a realizar al sistema; así como también los resultados esperados de las mismas. Esta especificación se ha realizado de acuerdo al estándar "IEEE Recommended Practice for Software Requirements Specification IEEE Std 830-1998".

1.1 Objetivos

El propósito de este documento es presentar los casos de pruebas unitarias para los casos de uso que se implementarán, de manera que se pueda ejecutar las pruebas de forma ordenada y sabiendo exactamente que esperar de cada una para poder validar si fue satisfactoria o no.

1.2 Alcance

El presente plan abarca a todos los casos de uso por desarrollar de la herramienta AXEbit, módulo extracción.

1.3 Referencias

Las referencias aplicables son:

1. Especificación de requisitos de software – Anexo C

1.4 Visión general del plan de pruebas

Este documento consta de 3 secciones. Esta sección, la introducción, proporciona una visión general del documento.

En la sección 2 se listan los requerimientos del producto que se busca satisfacer con las pruebas planteadas.

En la sección 3, se presentan los casos de pruebas unitarias para el módulo de extracción, los cuales indican paso a paso como debe realizarse la prueba, las precondiciones que deben cumplirse antes de ejecutar la prueba y el resultado esperado para poder dar como satisfactoria la prueba.

2. Requerimientos de Pruebas

La lista que se presenta a continuación identifica los requerimientos especificados en el ERS, para los cuales se realizarán los casos de prueba.

2.1 Módulo de Extracción

- Configurar Fuente de Datos y obtener estructuras y objetos de fuentes de datos.
- Configurar estructura de archivo plano.
- Crear job.
- Configurar parámetros.
- Crear paquete.
- Configurar orden de ejecución de paquetes.
- Crear flujo de transformación.
- Personalizar transformación usando objetos activos.
- Validar Scripts mediante parser.
- Crear funciones definidas por el usuario.
- Programar Jobs.
- Ejecutar Job.
- Visualizar log de ejecución.

3. Casos de Pruebas

A continuación se presentan los casos de prueba del sistema:

3.1 Configurar fuente de datos y obtener estructuras y objetos de fuente de datos

Configurar y Conectar Fuente de Datos	
ID Prueba:	BE01
Objetivo :	Configurar una fuente de datos mediante el ingreso de datos válidos para una base de datos SQL Server.
Precondición:	El usuario ha ingresado al sistema. El usuario ha abierto un paquete.
Clases :	JSFServlet, BEConfigurarBDBean, BEGestorFuenteDatos, BEControlFuenteDatos, BEFuenteDatos, IdFactory, Empresa, Proyecto, BEPaquete.
Proceso:	1. En el menú seleccione la opción “Paquete” y “Agregar Fuente de Datos”. 2. Seleccione “Conexión a Base de datos”. 3. Ingrese en el campo “Nombre” el valor “Fuente de prueba”. 4. Ingrese en el campo “Descripción” el valor “Fuente creada para caso de prueba”.

	Elija del combobox de “Tipo” el valor “SQL Server” de la lista. 5. Ingrese en el campo “Conexión” el valor “Inti”. 6. Ingrese en el campo “Base Datos” el valor “Bipucp”. 7. Ingrese en el campo “Usuario” el valor “biextrausr”. 8. Ingrese en el campo “Contraseña” el valor “2006biusr”. 9. Seleccione la opción “Grabar”.
Resultado Esperado:	Se regresa al área de trabajo inicial y se muestra la estructura de la Fuente de Datos.

Configurar y Conectar Fuente de Datos	
ID Prueba:	BE02
Objetivo :	Configurar una fuente de datos mediante el ingreso de datos válidos para una base de datos ORACLE.
Precondición:	El usuario ha ingresado al sistema. El usuario ha abierto un paquete.
Clases :	JSFServlet, BEConfigurarBDBean, BEGestorFuenteDatos, BEControlFuenteDatos, BEFuenteDatos, IdFactory, Empresa, Proyecto, BEPaquete.
Proceso:	1. En el menú seleccione la opción “Paquete” y “Agregar Fuente de Datos”. 2. Seleccione “Conexión a Base de datos”. 3. Ingrese en el campo “Nombre” el valor “Fuente de prueba”. 4. Ingrese en el campo “Descripción” el valor “Fuente creada para caso de prueba”. 5. Elija del combobox de “Tipo” el valor “ORACLE” de la lista. 6. Ingrese en el campo “Conexión” el valor “Inti”. 7. Ingrese en el campo “Base Datos” el valor “Bipucp”. 8. Ingrese en el campo “Usuario” el valor “biextrausr”. 9. Ingrese en el campo “Contraseña” el valor “2006biusr”. 10. Seleccione la opción “Grabar”.
Resultado Esperado:	Se regresa al área de trabajo inicial y se muestra la estructura de la Fuente de Datos.

Configurar y Conectar Fuente de Datos	
ID Prueba:	BE03
Objetivo :	Configurar una fuente de datos mediante el ingreso de datos válidos para una base de datos MySQL.
Precondición:	El usuario ha ingresado al sistema. El usuario ha abierto un paquete.
Clases :	JSFServlet, BEConfigurarBDBean, BEGestorFuenteDatos, BEControlFuenteDatos, BEFuenteDatos, IdFactory, Empresa, Proyecto, BEPaquete.
Proceso:	1. En el menú seleccione la opción “Paquete” y “Agregar Fuente de Datos”. 2. Seleccione “Conexión a Base de datos”. 3. Ingrese en el campo “Nombre” el valor “Fuente de prueba”. 4. Ingrese en el campo “Descripción” el valor “Fuente creada para caso de prueba”. 5. Elija del combobox de “Tipo” el valor “MySQL 5.0” de la lista. 6. Ingrese en el campo “Conexión” el valor “Inti”. 7. Ingrese en el campo “Base Datos” el valor “Bipucp”. 8. Ingrese en el campo “Usuario” el valor “biextrausr”.

	9. Ingrese en el campo “Contraseña” el valor “2006biusr”. 10. Seleccione la opción “Grabar”.
Resultado Esperado:	Se regresa al área de trabajo inicial y se muestra la estructura de la Fuente de Datos.

Configurar y Conectar Fuente de Datos	
ID Prueba:	BE04
Objetivo :	Configurar una fuente de datos mediante el ingreso de datos válidos y parámetros.
Precondición:	El usuario ha ingresado al sistema. El usuario ha abierto un job que contiene un parámetro denominado “conexion”. El usuario ha abierto un paquete del job..
Clases :	JSFServlet, BEConfigurarBDBean, BEGestorFuenteDatos, BEControlFuenteDatos, BEFuenteDatos, IdFactory, Empresa, Proyecto, BEPaquete.
Proceso:	1. En el menú seleccione la opción “Paquete” y “Agregar Fuente de Datos”. 2. Seleccione “Conexión a Base de datos”. 3. Ingrese en el campo “Nombre” el valor “Fuente de prueba”. 4. Ingrese en el campo “Descripción” el valor “Fuente creada para caso de prueba”. 5. Elija del combobox de “Tipo” el valor “SQL Server” de la lista. 6. Ingrese en el campo “Conexión” el valor “#conexion#”. 7. Ingrese en el campo “Base Datos” el valor “Bipucp”. 8. Ingrese en el campo “Usuario” el valor “biextrausr”. 9. Ingrese en el campo “Contraseña” el valor “2006biusr”. 10. Seleccione la opción “Grabar”.
Resultado Esperado:	Se regresa al área de trabajo inicial y se muestra la estructura de la Fuente de Datos.

Configurar y Conectar Fuente de Datos	
ID Prueba:	BE05
Objetivo :	Verificar la muestra de mensajes de error si no se ingresan correctos los datos de la Base de datos campo “Conexión”.
Precondición:	El usuario ha ingresado al sistema. El usuario ha abierto un paquete.
Clases :	JSFServlet, BEConfigurarBDBean, BEGestorFuenteDatos, BEControlFuenteDatos, BEFuenteDatos, IdFactory, Empresa, Proyecto, BEPaquete.
Proceso:	1. En el menú seleccione la opción “Paquete” y “Agregar Fuente de Datos”. 2. Seleccione “Conexión a Base de datos”. 3. Ingrese en el campo “Nombre” el valor “Fuente de prueba”. 4. Ingrese en el campo “Descripción” el valor “Fuente creada para caso de prueba”. 5. Elija del combobox de “Tipo” el valor “SQl Server” de la lista. 6. Ingrese en el campo “Conexión” el valor “Servidor”. 7. Ingrese en el campo “Base Datos” el valor “Bipucp”. 8. Ingrese en el campo “Usuario” el valor “biextrausr”. 9. Ingrese en el campo “Contraseña” el valor “2006biusr”.

	10. Seleccione la opción “Grabar”.
Resultado Esperado:	Se muestra el mensaje “Error al intentar leer información de la BD”.

Configurar y Conectar Fuente de Datos	
ID Prueba:	BE06
Objetivo :	Verificar la muestra de mensajes de error si no se ingresan correctos los datos del usuario de Base de datos campo “Usuario”.
Precondición:	El usuario ha ingresado al sistema. El usuario ha abierto un paquete.
Clases :	JSFServlet, BEConfigurarBDBean, BEGestorFuenteDatos, BEControlFuenteDatos, BEFuenteDatos, IdFactory, Empresa, Proyecto, BEPaquete.
Proceso:	1. En el menú seleccione la opción “Paquete” y “Agregar Fuente de Datos”. 2. Seleccione “Conexión a Base de datos”. 3. Ingrese en el campo “Nombre” el valor “Fuente de prueba”. 4. Ingrese en el campo “Descripción” el valor “Fuente creada para caso de prueba”. 5. Elija del combobox de “Tipo” el valor “SQL Server” de la lista. 6. Ingrese en el campo “Conexión” el valor “Inti”. 7. Ingrese en el campo “Base Datos” el valor “Bipucp”. 8. Ingrese en el campo “Usuario” el valor “infdba”. 9. Ingrese en el campo “Contraseña” el valor “2006biusr”. 10. Seleccione la opción “Grabar”.
Resultado Esperado:	Se muestra el mensaje “Error al intentar leer información de la BD”.

Configurar y Conectar Fuente de Datos	
ID Prueba:	BE07
Objetivo :	Verificar la muestra de mensajes de error si no se puede realizar la conexión a la Base de datos
Precondición:	El usuario ha ingresado al sistema. El usuario ha abierto un paquete.
Clases :	JSFServlet, BEConfigurarBDBean, BEGestorFuenteDatos, BEControlFuenteDatos, BEFuenteDatos, IdFactory, Empresa, Proyecto, BEPaquete.
Proceso:	1. En el menú seleccione la opción “Paquete” y “Agregar Fuente de Datos”. 2. Seleccione “Conexión a Base de datos”. 3. Ingrese en el campo “Nombre” el valor “Fuente de prueba”. 4. Ingrese en el campo “Descripción” el valor “Fuente creada para caso de prueba”. 5. Elija del combobox de “Tipo” el valor “SQL Server” de la lista. 6. Ingrese en el campo “Conexión” el valor “Inti”. 7. Ingrese en el campo “Nombre de la BD” el valor “Bipucp”. 8. Ingrese en el campo “Usuario” el valor “biextrausr”. 9. Ingrese en el campo “Contraseña” el valor “2006biusr”. 10. Seleccione la opción “Grabar”.
Resultado Esperado:	Se muestra el mensaje “Error al intentar leer información de la BD”

3.2 Modificar Fuente de Datos

Modificar Fuente de Datos	
ID Prueba:	BE08
Objetivo :	Verificar la correcta modificación de una fuente de datos tipo Base de Datos mediante el ingreso de datos válidos
Precondición:	Existe una base de datos SQLServer con conexión “inti2”. El usuario ha ingresado al sistema. El usuario ha abierto un paquete.
Clases :	JSFServlet, BEConfigurarBDBean, BEGestorFuenteDatos, BEControlFuenteDatos, BEFuenteDatos, IdFactory, Empresa, Proyecto, BEPaquete.
Proceso:	1. En el árbol de navegación de objetos seleccione la fuente de datos “Fuente de Prueba” haciendo click derecho sobre el nombre y seleccione “Modificar Fuente” 2. Modifique el campo “Nombre” ingresando el nuevo valor “Fuente de prueba 2”. 3. Modifique el campo “Descripción” ingresando el nuevo valor “Fuente creada para caso de prueba 2”. 4. Modifique el campo “Conexion” ingresando el nuevo valor “inti2”. 5. Seleccione la opción “Grabar”.
Resultado Esperado:	Se muestra en el árbol de navegación de objetos la fuente de datos modificada y se muestra la estructura actualizada.

Modificar Fuente de Datos	
ID Prueba:	BE09
Objetivo :	Verificar la muestra de mensajes de error en la modificación de una fuente de datos tipo Base de Datos mediante el ingreso de datos inválidos
Precondición:	No existe una base de datos SQL Server con conexión “inti3”. El usuario ha ingresado al sistema. El usuario ha abierto un paquete.
Clases :	JSFServlet, BEConfigurarBDBean, BEGestorFuenteDatos, BEControlFuenteDatos, BEFuenteDatos, IdFactory, Empresa, Proyecto, BEPaquete.
Proceso:	1. En el árbol de navegación de objetos seleccione la fuente de datos “Fuente de Prueba 2” haciendo click derecho sobre el nombre y seleccione “Modificar Fuente” 2. Modifique el campo “Nombre” ingresando el nuevo valor “Fuente de prueba 3”. 3. Modifique el campo “Descripción” ingresando el nuevo valor “Fuente creada para caso de prueba 3”. 4. Modifique el campo “Conexion” ingresando el nuevo valor “inti3”. 5. Seleccione la opción “Grabar”.
Resultado Esperado:	Se muestra el mensaje “Error al intentar leer información de la BD”

3.3 Eliminar Fuente de Datos

Eliminar Fuente de Datos	
ID Prueba:	BE10
Objetivo :	Verificar la correcta eliminación de una Fuente de Datos sin objetos pasivos en flujo
Precondición:	El usuario ha ingresado al sistema. El usuario ha abierto un paquete.
Clases :	JSFServlet, BEConfigurarBDBean, BEGestorFuenteDatos, BEControlFuenteDatos, BEFuenteDatos, IdFactory, Empresa, Proyecto, BEPaquete.
Proceso:	1. En el árbol de navegación de objetos seleccione la fuente de datos “Fuente de Prueba 2” haciendo click derecho sobre el nombre y seleccione “Eliminar Fuente” 2. Sobre el mensaje de confirmación de eliminación seleccione “Sí”.
Resultado Esperado:	Se muestra el árbol de navegación de objetos actualizado sin contener la fuente de datos.

Eliminar Fuente de Datos	
ID Prueba:	BE11
Objetivo :	Verificar la correcta eliminación de una Fuente de Datos con objetos pasivos en flujo
Precondición:	El usuario ha ingresado al sistema. El usuario ha abierto un paquete. El paquete tiene una fuente de datos denominada “Fuente de Prueba 2” que contiene objetos pasivos en el flujo.
Clases :	JSFServlet, BEConfigurarBDBean, BEGestorFuenteDatos, BEControlFuenteDatos, BEFuenteDatos, IdFactory, Empresa, Proyecto, BEPaquete.
Proceso:	1. En el árbol de navegación de objetos seleccione la fuente de datos “Fuente de Prueba 2” haciendo click derecho sobre el nombre y seleccione “Eliminar Fuente” 2. Sobre el mensaje de confirmación de eliminación seleccione “Sí”.
Resultado Esperado:	Se muestra el árbol de navegación de objetos actualizado sin contener la fuente de datos, y en el área de trabajo se muestra el flujo sin contener los objetos pasivos de la fuente de datos eliminada.

3.4 Agregar archivo y configurar estructura

Agregar Archivo y Configurar Estructura	
ID Prueba:	BE12
Objetivo :	Agregar un archivo plano tipo TXT Obtener la estructura de un archivo plano tipo TXT mediante el ingreso de datos válidos.
Precondición:	El usuario ha ingresado al sistema. Existe un paquete denominado “Paquete 1” y el usuario lo ha abierto. Existe un archivo TXT en la carpeta del usuario con datos válidos.
Clases :	BEConfigurarEstructuraBean,BECampoBeanEstructura, BECampo, BEArchivo, BEControlFuenteDatos, BEFuenteDatos, BEFormato,

	BETipoDato.
Proceso:	1. En el árbol de navegación de objetos elija el paquete “Paquete 1” haciendo click derecho sobre el nombre y seleccione “Agregar Fuente de Datos Archivo”. 2. Ingrese en el campo “Nombre” el valor “TXT Prueba 1”. 3. Ingrese en el campo “Descripción” el valor “Descripcion de prueba para txt”. 4. En el campo “Tipo” seleccione TXT 5. En el campo “Carpeta” seleccione la ubicación del archivo .txt. 6. Seleccione “Grabar”. 7. En el árbol de navegación de objetos elija “TXT Prueba 1” haciendo click derecho sobre el nombre y seleccione “Configurar Estructura”. 8. Ingrese en el campo “Primera línea Nombre de Campos”, el valor “Si”. 9. Ingrese en el campo “Carácter de fin de registro” el valor “.”. 10. Ingrese en el campo “Carácter de separación” el valor “,”. 11. Ingrese en el campo “Ancho Fijo” el valor “2” 12. Seleccione “Cargar Estructura”. 13. Para cada campo ingrese los valores correctos según el archivo. 14. Seleccione “Analizar Archivo”.
Resultado Esperado:	El sistema guarda la configuración del archivo y se regresa al área de trabajo sin mostrar ningún mensaje de error.

Agregar Archivo y Configurar Estructura	
ID Prueba:	BE13
Objetivo :	Verificar la muestra de mensajes de error cuando se ingresa datos inválidos para la configuración de un archivo plano tipo TXT.
Precondición:	El usuario ha ingresado al sistema. La fuente de datos tipo TXT ha sido agregada correctamente.
Clases :	BEConfigurarEstructuraBean,BECampoBeanEstructura, BECampo, BEArchivo, BEControlFuenteDatos, BEFuenteDatos, BEFormato, BETipoDato.
Proceso:	1. En el árbol de navegación de objetos seleccione el archivo “TXT Prueba 1” y seleccione “Configurar Estructura” 2. Ingrese en el campo “Primera línea Nombre de Campos”, el valor “Si”. 3. Ingrese en el campo “Carácter de fin de registro” el valor “/”. 4. Ingrese en el campo “Carácter de separación” el valor “,”. 5. Ingrese en el campo “Ancho Fijo” el valor “2” 6. Seleccione “Cargar Estructura”. 7. Para cada campo ingrese los valores correctos según el archivo. 8. Seleccione “Analizar Archivo”.
Resultado Esperado:	Se muestra el mensaje “Error al analizar archivo”.

Agregar Archivo y Configurar Estructura	
ID Prueba:	BE14
Objetivo :	Agregar un archivo tipo XLS Obtener la estructura de un archivo tipo XLS mediante el ingreso de datos válidos.
Precondición:	El usuario ha ingresado al sistema. Existe un paquete denominado “Paquete 1” y el usuario lo ha abierto. Existe un archivo XLS en la carpeta del usuario con datos válidos.
Clases :	BEConfigurarEstructuraBean,BECampoBeanEstructura, BECampo,

	BEArchivo, BEControlFuenteDatos, BEFuenteDatos, BEFormato, BETipoDato.
Proceso:	1. En el árbol de navegación de objetos elija el paquete “Paquete 1” haciendo click derecho sobre el nombre y seleccione “Agregar Fuente de Datos Archivo”. 2. Ingrese en el campo “Nombre” el valor “XLS Prueba 2”. 3. Ingrese en el campo “Descripción” el valor “Descripcion de prueba para xls”. 4. En el campo “Tipo” seleccione XLS. 5. En el campo “Carpeta” seleccione la ubicación del archivo .xls. 6. Seleccione “Grabar”. 7. En el árbol de navegación de objetos elija “XLS Prueba 2” haciendo click derecho sobre el nombre y seleccione “Configurar Estructura”. 8. Ingrese en el campo “Primera línea Nombre de Campos”, el valor “Si”. 9. Seleccione “Cargar Estructura”. 10. Para cada campo ingrese los valores correctos según el archivo. 11. Seleccione “Analizar Archivo”.
Resultado Esperado:	El sistema guarda la configuración del archivo y se regresa al área de trabajo sin mostrar ningún mensaje de error.

Agregar Archivo y Configurar Estructura	
ID Prueba:	BE15
Objetivo :	Verificar la muestra de mensajes de error cuando se ingresa datos inválidos para la configuración de un archivo tipo XLS.
Precondición:	El usuario ha ingresado al sistema. La fuente de datos tipo XLS ha sido agregada correctamente. El archivo xls contiene datos inválidos.
Clases :	BEConfigurarEstructuraBean,BECampoBeanEstructura, BECampo, BEArchivo, BEControlFuenteDatos, BEFuenteDatos, BEFormato, BETipoDato.
Proceso:	1. En el árbol de navegación de objetos seleccione el archivo “XLS Prueba 2” y seleccione “Configurar Estructura” 2. Seleccione “Cargar Estructura”. 3. Para cada campo ingrese los valores correctos según el archivo. 4. Seleccione “Analizar Archivo”.
Resultado Esperado:	Se muestra el mensaje “Error al analizar archivo”.

Agregar Archivo y Configurar Estructura	
ID Prueba:	BE16
Objetivo :	Agregar un archivo tipo XML Obtener la estructura de un archivo tipo XML mediante el ingreso de datos válidos.
Precondición:	El usuario ha ingresado al sistema. Existe un paquete denominado “Paquete 1” y el usuario lo ha abierto. Existe un archivo XML en la carpeta del usuario con datos válidos. Existe un archivo XSD en la carpeta del usuario con un esquema válido para el xml.
Clases :	BEConfigEstrucXmlBean,BECampoBeanEstructura, BECampo, BEArchivo, BEControlFuenteDatos, BEFuenteDatos, BEFormato, BETipoDato, BEEstructuraXML

Proceso:	1. En el árbol de navegación de objetos elija el paquete “Paquete 1” haciendo click derecho sobre el nombre y seleccione “Agregar Fuente de Datos Archivo”. 2. Ingrese en el campo “Nombre” el valor “XML Prueba 3”. 3. Ingrese en el campo “Descripción” el valor “Descripcion de prueba para xml”. 4. En el campo “Tipo” seleccione XML. 5. En el campo “Carpeta” seleccione la ubicación del archivo .xml. 6. En el árbol de navegación de objetos elija “XML Prueba 3” haciendo click derecho sobre el nombre y seleccione “Configurar Estructura”. 7. Seleccione el archivo esquema .xsd correspondiente al xml. 8. Seleccione “Cargar Estructura”. 9. Seleccione “Analizar Archivo”.
Resultado Esperado:	El sistema guarda la configuración del archivo y se regresa al área de trabajo sin mostrar ningún mensaje de error.

Agregar Archivo y Configurar Estructura	
ID Prueba:	BE17
Objetivo :	Verificar la muestra de mensajes de error cuando se ingresa datos inválidos para la configuración de un archivo plano tipo XML.
Precondición:	El usuario ha ingresado al sistema. La fuente de datos tipo XML ha sido agregada correctamente. El archivo xml contiene datos inválidos. El archivo xsd contiene un esquema correcto.
Clases :	BEConfigEstrucXmlBean,BECampoBeanEstructura, BECampo, BEArchivo, BEControlFuenteDatos, BEFuenteDatos, BEFormato, BETipoDato, BEEstructuraXML
Proceso:	1. En el árbol de navegación de objetos seleccione el archivo “XML Prueba 3” y seleccione “Configurar Estructura”. 2. Seleccione un archivo esquema .xsd válido. 3. Seleccione “Cargar Estructura”.. 4. Seleccione “Analizar Archivo”.
Resultado Esperado:	Se muestra el mensaje “El archivo XML no corresponde con el esquema”.

3.5 Crear job

Crear job	
ID Prueba:	BE18
Objetivo :	Crear un job mediante el ingreso de datos válidos.
Precondición:	El usuario ha ingresado al sistema. El usuario ha abierto un proyecto.
Clases :	JSFServlet, BEJobBean, BEControlJob, BEGestorJob, BEJob, Empresa, Proyecto, IdFactory.
Proceso:	1. En el menú seleccione “Administrar Jobs” y “Crear”. 2. Ingrese en el campo “Nombre” el valor “Job Prueba”. 3. Ingrese en el campo “Descripción” el valor “Job creado para caso de prueba”. 4. En el formulario del job, seleccione la opción “Grabar”.
Resultado Esperado:	Se muestra en el formulario de Administración del Jobs el nuevo job creado.

Crear job	
ID Prueba:	BE19
Objetivo :	Verificar la muestra de mensajes de error si no se ingresan correctos los datos del Job campo “Nombre”.
Precondición:	El usuario ha ingresado al Sistema. El usuario ha abierto un proyecto.
Clases :	JSFServlet, BEJobBean, BEControlJob, BEGestorJob, BEJob, Empresa, Proyecto, IdFactory.
Proceso:	1. En el menú seleccione “Administrar Jobs” y “Crear”. 2. Deje vacío el campo “Nombre”. 3. Ingrese en el campo “Descripción” el valor “Job creado para caso de prueba”. 4. En el formulario del job, seleccione la opción “Grabar”.
Resultado Esperado:	Se muestra el mensaje “Valor incorrecto” para el campo “Nombre”.

Crear job	
ID Prueba:	BE20
Objetivo :	Verificar la muestra de mensajes de error si se ingresa el nombre de un job previamente existente.
Precondición:	El usuario ha ingresado al Sistema. El usuario ha abierto un proyecto.
Clases :	JSFServlet, BEJobBean, BEControlJob, BEGestorJob, BEJob, Empresa, Proyecto, IdFactory.
Proceso:	1. En el menú seleccione “Administrar Jobs” y “Crear”. 2. Ingrese en el campo “Nombre” el valor “Job Prueba” que es el nombre del job previamente creado. 3. Ingrese en el campo “Descripción” el valor “Job creado para segundo caso de prueba”. 4. En el formulario del job, seleccione la opción “Grabar”.
Resultado Esperado:	Se muestra el mensaje “Nombre de Job ya existe” para el campo “Nombre”.

3.6 Modificar Job

Modificar job	
ID Prueba:	BE21
Objetivo :	Modificar un Job mediante el ingreso de datos válidos
Precondición:	El usuario ha ingresado al Sistema. El usuario ha abierto un proyecto.
Clases :	JSFServlet, BEJobBean, BEControlJob, BEGestorJob, BEJob, Empresa, Proyecto, IdFactory.
Proceso:	1. En el menú seleccione “Administrar Jobs” y elija el job “Job Prueba” de la lista haciendo click sobre el nombre. 2. Modifique el campo “Nombre” ingresando el nuevo valor “Job 2” 3. Modifique el campo “Descripción” ingresando el nuevo valor “Job creado para caso de prueba 2”. 4. En el formulario del job, seleccione la opción “Grabar”.
Resultado Esperado:	Se muestra en el formulario de Administración del Jobs el job modificado.

Modificar Job	
ID Prueba:	BE22
Objetivo :	Verificar la muestra de mensajes de error si no se modifican correctamente los datos del Job campo “Nombre”.
Precondición:	El usuario ha ingresado al Sistema. El usuario ha abierto un proyecto.
Clases :	JSFServlet, BEJobBean, BEControlJob, BEGestorJob, BEJob, Empresa, Proyecto, IdFactory.
Proceso:	1. En el menú seleccione “Administrar Jobs” y elija el job “Job 2” de la lista haciendo click sobre el nombre.. 2. Modifique el campo “Nombre” dejándolo vacío. 3. En el formulario del job, seleccione la opción “Grabar”.
Resultado Esperado:	Se muestra el mensaje “Valor incorrecto” para el campo “Nombre”.

Modificar job	
ID Prueba:	BE23
Objetivo :	Verificar la muestra de mensajes de error si se modifica el nombre de un job ingresando un valor previamente existente.
Precondición:	El usuario ha ingresado al Sistema. El usuario ha abierto un proyecto.
Clases :	JSFServlet, BEJobBean, BEControlJob, BEGestorJob, BEJob, Empresa, Proyecto, IdFactory.
Proceso:	1. En el menú seleccione “Administrar Jobs”. 2. Cree un job con nombre “Job 3”. 3. En la lista de jobs, elija el job “Job 2” haciendo click sobre el nombre.. 4. Ingrese en el campo “Nombre” el valor “Job 3” que es el nombre del job creado en el paso 2. 5. En el formulario del job, seleccione la opción “Grabar”.
Resultado Esperado:	Se muestra el mensaje “Nombre de Job ya existe” para el campo “Nombre”.

3.7 Eliminar Job

Eliminar job	
ID Prueba:	BE24
Objetivo :	Verificar la correcta eliminación de un job que contiene paquetes.
Precondición:	El usuario ha ingresado al Sistema. El usuario ha abierto un proyecto.
Clases :	JSFServlet, BEJobBean, BEControlJob, BEGestorJob, BEJob, Empresa, Proyecto, IdFactory.
Proceso:	1. En el menú seleccione “Administrar Jobs”. 2. Cree un nuevo job con nombre “Job 4”. 3. Agregue paquetes al job creado en el paso 2. 4. Ingrese al formulario de Administración de Jobs y elija el job “Job 4” haciendo click sobre la casilla de la izquierda del nombre. 5. Seleccione la opción “Eliminar”.
Resultado Esperado:	En el formulario de Administración de Jobs se muestra la lista actualizada sin contener el job “Job 4”.

Eliminar job	
ID Prueba:	BE25
Objetivo :	Verificar la correcta eliminación de un job que no contiene paquetes.
Precondición:	El usuario ha ingresado al Sistema. El usuario ha abierto un proyecto.
Clases :	JSFServlet, BEJobBean, BEControlJob, BEGestorJob, BEJob, Empresa, Proyecto, IdFactory.
Proceso:	1. En el menú seleccione “Administrar Jobs”. 2. Cree un job con nombre “Job 5”. 3. En el formulario de administración de Jobs elija el job “Job 5” haciendo click sobre la casilla de la izquierda del nombre. 4. Seleccione la opción “Eliminar”.
Resultado Esperado:	En el formulario de Administración de Jobs se muestra la lista actualizada sin contener el job “Job 5”.

3.8 Configurar parámetro

Configurar parámetro	
ID Prueba:	BE26
Objetivo :	Crear un parámetro para el job mediante el ingreso de datos válidos.
Precondición:	El usuario ha ingresado al sistema. El usuario ha abierto un proyecto. El usuario se encuentra en proceso de creación de un job.
Clases :	JSFServlet, BEParametroBean, BEJob, FacesUtil.
Proceso:	1. En el formulario del Job seleccione “Agregar Parámetro”. 2. Ingrese en el campo “Nombre” el valor “Parametro Prueba”. 3. Ingrese en el campo “Etiqueta” el valor “Etiqueta Prueba”. 4. Ingrese en el campo “Valor por Defecto” el valor “Valor Defecto Prueba”. 5. Seleccione la opción “Grabar”.
Resultado Esperado:	Se muestra el nuevo parámetro creado en el formulario del Job.

Configurar parámetro	
ID Prueba:	BE27
Objetivo :	Verificar la muestra de mensajes de error si no se ingresa correctamente los datos del Parámetro campo “Nombre”.
Precondición:	El usuario ha ingresado al sistema. El usuario ha abierto un proyecto. El usuario se encuentra en proceso de creación de un job.
Clases :	JSFServlet, BEParametroBean, BEJob, FacesUtil.
Proceso:	1. En el formulario del Job seleccione “Agregar Parámetro”. 2. Deje vacío el campo “Nombre”. 3. Ingrese en el campo “Etiqueta” el valor “Etiqueta Prueba 2”. 4. Ingrese en el campo “Valor por Defecto” el valor “Valor Defecto Prueba 2”. 5. Seleccione la opción “Grabar”.
Resultado Esperado:	Se muestra el mensaje “Valor incorrecto” para el campo “Nombre”.

Configurar parámetro	
ID Prueba:	BE28
Objetivo :	Verificar la muestra de mensajes de error si se ingresa el nombre de un parámetro previamente existente.
Precondición:	El usuario ha ingresado al sistema. El usuario ha abierto un proyecto. El usuario se encuentra en proceso de creación de un job.
Clases :	JSFServlet, BEParametroBean, BEJob, FacesUtil.
Proceso	1. En el formulario del Job seleccione "Agregar Parámetro". 2. Ingrese en el campo "Nombre" el valor "Parametro Prueba" que es el nombre del parámetro previamente creado. 3. Ingrese en el campo "Etiqueta" el valor "Etiqueta Prueba 3". 4. Ingrese en el campo "Valor por Defecto" el valor "Valor Defecto Prueba 3". 5. Seleccione la opción "Grabar".
Resultado Esperado:	Se muestra el mensaje "Nombre de Parámetro ya existe" para el campo "Nombre".

3.9 Modificar Parámetro

Modificar parámetro	
ID Prueba:	BE29
Objetivo :	Modificar un parámetro mediante el ingreso de datos válidos.
Precondición:	El usuario ha ingresado al sistema. El usuario ha abierto un proyecto. El usuario se encuentra en proceso de creación de un job.
Clases :	JSFServlet, BEParametroBean, BEJob, FacesUtil.
Proceso	1. En el formulario de creación de Job seleccione el parámetro "Parametro Prueba" haciendo click sobre el nombre. 2. Modificar el campo "Nombre" ingresando el nuevo valor "Parametro 2". 3. Modificar el campo "Etiqueta" ingresando el nuevo valor "Etiqueta Prueba 2". 4. Ingrese en el campo "Valor por Defecto" el valor "Valor Defecto Prueba 2". 5. Seleccione la opción "Grabar".
Resultado Esperado:	Se muestra en el formulario de creación de job el parámetro modificado.

Modificar parámetro	
ID Prueba:	BE30
Objetivo :	Verificar la muestra de mensajes de error si no se modifica correctamente los datos del Parámetro campo “Nombre”.
Precondición:	El usuario ha ingresado al sistema. El usuario ha abierto un proyecto. El usuario se encuentra en proceso de creación de un job.
Clases :	JSFServlet, BEParametroBean, BEJob, FacesUtil.
Proceso:	1. En el formulario de creación de Job seleccione el parámetro “Parametro 2” haciendo click sobre el nombre. 2. Modifique el campo “Nombre” dejándolo vacío. 3. Seleccione la opción “Grabar”.
Resultado Esperado:	Se muestra el mensaje “Valor incorrecto” para el campo “Nombre”.

Modificar parámetro	
ID Prueba:	BE31
Objetivo :	Verificar la muestra de mensajes de error si se modifica el nombre de un parámetro ingresando un valor previamente existente.
Precondición:	El usuario ha ingresado al sistema. El usuario ha abierto un proyecto. El usuario se encuentra en proceso de creación de un job.
Clases :	JSFServlet, BEParametroBean, BEJob, FacesUtil.
Proceso:	1. Cree un nuevo parámetro con nombre “Parametro 3”. 2. En la relación de parámetros seleccione el parámetro “Parametro 2” haciendo click sobre el nombre. 3. Modifique el campo “Nombre” ingresando el nuevo valor “Parametro 3”. que es el nombre del parámetro creado en el paso 1. 4. Seleccione la opción “Grabar”.
Resultado Esperado:	Se muestra el mensaje “Nombre de Parámetro ya existe” para el campo “Nombre”.

3.10 Eliminar Parámetro

Eliminar parámetro	
ID Prueba:	BE32
Objetivo :	Verificar la correcta eliminación de un parámetro.
Precondición:	El usuario ha ingresado al sistema. El usuario ha abierto un proyecto. El usuario se encuentra en proceso de creación de un job.
Clases :	JSFServlet, BEParametroBean, BEJob, FacesUtil.
Proceso:	1. En el formulario de creación de Job elija el parámetro “Parametro 3” haciendo click sobre la casilla de la izquierda del nombre. 2. Seleccione la opción “Eliminar”.
Resultado Esperado:	Se muestra la relación de parámetros actualizada sin contener el parámetro “Parametro 3”.

3.11 Crear paquete

Crear paquete	
ID Prueba:	BE33
Objetivo :	Crear un paquete mediante el ingreso de datos válidos.
Precondición:	El usuario ha ingresado al sistema. El usuario ha abierto un job.
Clases :	JSFServlet, BEPaqueteBean, BEGestorJob, BEJob, GestorTema, BEControlPaquete, BEGestorPaquete, BEPaquete, Empresa, Proyecto, IdFactory.
Proceso:	1. En el menú seleccione la opción “Agregar nuevo paquete”. 2. Ingrese en el campo “Nombre” el valor “Paquete Prueba”. 3. Ingrese en el campo “Descripción” el valor “Paquete creado para caso de prueba”. 4. Elija del combobox “Tema” el valor “Tema12”. 5. Seleccione la opción “Grabar”.
Resultado Esperado:	Se graba exitosamente el paquete, se regresa al área de trabajo Inicial y se muestra en el árbol de navegación de objetos, el nuevo paquete creado.

Crear paquete	
ID Prueba:	BE34
Objetivo :	Verificar la muestra de mensajes de error si no se ingresan correctos los datos del Paquete campo “Nombre”.
Precondición:	El usuario ha ingresado al sistema. El usuario ha abierto un job.
Clases :	JSFServlet, BEPaqueteBean, BEGestorJob, BEJob, GestorTema, BEControlPaquete, BEGestorPaquete, BEPaquete, Empresa, Proyecto, IdFactory.
Proceso:	1. En el menú seleccione la opción “Agregar nuevo paquete”. 2. Deje vacío el campo “Nombre”. 3. Ingrese en el campo “Descripción” el valor “Paquete creado para caso de prueba”. 4. Elija del combobox “Tema” el valor “Tema12”. 5. Seleccione la opción “Grabar”.
Resultado Esperado:	Se muestra el mensaje “Valor Incorrecto” para el campo “Nombre”.

Crear paquete	
ID Prueba:	BE35
Objetivo :	Verificar la muestra de mensajes de error si se ingresa el nombre de un paquete previamente existente.
Precondición:	El usuario ha ingresado al sistema. El usuario ha abierto un job.
Clases :	JSFServlet, BEPaqueteBean, BEGestorJob, BEJob, GestorTema, BEControlPaquete, BEGestorPaquete, BEPaquete, Empresa, Proyecto, IdFactory.
Proceso:	1. En el menú seleccione la opción “Agregar nuevo paquete”. 2. Ingrese en el campo “Nombre” el valor “Paquete Prueba” que es el nombre del paquete previamente creado. 3. Ingrese en el campo “Descripción” el valor “Paquete creado para segundo caso de prueba”. 4. Elija del combobox “Tema” el valor “Tema12”. 5. Seleccione la opción “Grabar”.
Resultado Esperado:	Se muestra el mensaje “Nombre de Paquete ya existe” para el campo “Nombre”.

3.12 Modificar Paquete

Modificar paquete	
ID Prueba:	BE36
Objetivo :	Modificar un paquete mediante el ingreso de datos válidos.
Precondición:	El usuario ha ingresado al sistema. El usuario ha abierto un job.
Clases :	JSFServlet, BEPaqueteBean, BEGestorJob, BEJob, GestorTema, BEControlPaquete, BEGestorPaquete, BEPaquete, Empresa, Proyecto, IdFactory.
Proceso:	1. En el arbol de navegación de objetos haga click derecho sobre el paquete “Paquete Prueba” y seleccione “Configurar paquete”. 2. Modifique el campo “Nombre” ingresando el nuevo valor “Paquete 2”. 3. Modifique el campo “Descripción” ingresando el nuevo valor “Paquete creado para caso de prueba 2” 4. Seleccione la opción “Grabar”.
Resultado Esperado:	Se modifica exitosamente el paquete, se regresa al área de trabajo Inicial y se muestra en el árbol de navegación de objetos, el paquete modificado.

Modificar paquete	
ID Prueba:	BE37
Objetivo :	Verificar la muestra de mensajes de error si no se modifica correctamente los datos del Paquete campo “Nombre”.
Precondición:	El usuario ha ingresado al sistema. El usuario ha abierto un job.
Clases :	JSFServlet, BEPaqueteBean, BEGestorJob, BEJob, GestorTema, BEControlPaquete, BEGestorPaquete, BEPaquete, Empresa, Proyecto, IdFactory.
Proceso:	1. En el árbol de navegación de Objetos haga click derecho sobre el paquete “Paquete 2” y seleccione la opción “Configurar paquete”. 2. Modifique el campo “Nombre” dejándolo vacío. 3. Seleccione la opción “Grabar”.

Resultado Esperado:	Se muestra el mensaje “Valor Incorrecto” para el campo “Nombre”.

Modificar paquete	
ID Prueba:	BE38
Objetivo :	Verificar la muestra de mensajes de error si se modifica el nombre de un paquete ingresando un valor previamente existente.
Precondición:	El usuario ha ingresado al sistema. El usuario ha abierto un job.
Clases :	JSFServlet, BEPaqueteBean, BEGestorJob, BEJob, GestorTema, BEControlPaquete, BEGestorPaquete, BEPaquete, Empresa, Proyecto, IdFactory.
Proceso:	1. Cree un nuevo paquete con nombre “Paquete 3”. 2. En el arbol de navegación de objetos haga click derecho sobre el paquete “Paquete 2” y seleccione “Configurar paquete”. 3. Modifique el campo “Nombre” ingresando el nuevo valor “Paquete 3”. 4. Seleccione la opción “Grabar”.
Resultado Esperado:	Se muestra el mensaje “Nombre de Paquete ya existe” para el campo “Nombre”.

3.13 Eliminar Paquete

Eliminar paquete	
ID Prueba:	BE39
Objetivo :	Verificar la correcta eliminación de un paquete que contiene componentes.
Precondición:	El usuario ha ingresado al sistema. El usuario ha abierto un job.
Clases :	JSFServlet, BEPaqueteBean, BEGestorJob, BEJob, GestorTema, BEControlPaquete, BEGestorPaquete, BEPaquete, Empresa, Proyecto, IdFactory.
Proceso:	1. Cree un nuevo paquete con nombre “Paquete 4”. 2. Agregue componentes al paquete. 3. En el árbol de navegación de Objetos haga click derecho sobre el paquete “Paquete 4” y seleccione “Eliminar Paquete”. 4. Sobre el mensaje de confirmación de eliminación haga click en Sí.
Resultado Esperado:	Se elimina exitosamente el paquete, se regresa al área de trabajo Inicial y se muestra el árbol de navegación de objetos sin el paquete “Paquete 4”

Eliminar paquete	
ID Prueba:	BE40
Objetivo :	Verificar la correcta eliminación de un paquete que no contiene componentes.
Precondición:	El usuario ha ingresado al sistema. El usuario ha abierto un job.
Clases :	JSFServlet, BEPaqueteBean, BEGestorJob, BEJob, GestorTema, BEControlPaquete, BEGestorPaquete, BEPaquete, Empresa, Proyecto, IdFactory.
Proceso:	1. Cree un nuevo paquete con nombre “Paquete 5”. 2. En el árbol de navegación de Objetos haga click derecho sobre el

	paquete “Paquete 5” y seleccione “Eliminar Paquete”. 3. Sobre el mensaje de confirmación de eliminación haga click en Sí.
Resultado Esperado:	Se elimina exitosamente el paquete, se regresa al área de trabajo Inicial y se muestra el árbol de navegación de objetos sin el paquete “Paquete 5”.

3.14 Configurar orden de ejecución de paquetes

Configurar orden de ejecución de paquetes	
ID Prueba:	BE41
Objetivo :	Configurar el orden de ejecución de paquetes mediante el ingreso de datos válidos.
Precondición:	El usuario ha ingresado al Sistema. El usuario ha abierto un job. El job debe contener por lo menos un paquete.
Clases :	JSFServlet, BEConfigOrdenPaqBean, BEPaqueteJobBean, BEControlJob, BEGestorJob, BEJob, BEPaquete.
Proceso:	1. En el menú, seleccione la opción “Configurar orden de paquetes” 2. Elija el job “Carga Inicial”. 3. Elija para el Paquete “Flujo Dimensiones” el orden “1”. 4. Elija para el Paquete “Flujo Facts” el orden “2”. 5. Seleccione la opción “Grabar”.
Resultado Esperado:	Se graba exitosamente el orden de paquetes configurado, se cierra la ventana sin mostrar mensaje de error y se regresa al área de trabajo inicial.

3.15 Crear flujo de transformación

Crear flujo de transformación	
ID Prueba:	BE42
Objetivo :	Crear el flujo de transformación mediante la selección de datos válidos.
Precondición:	El usuario ha ingresado al sistema. El usuario ha abierto un job. Las fuentes de datos a utilizar han sido configuradas correctamente.
Clases :	AppletExtraccion, DiagramaDibujo, BEControlDibXML, DiagramaDibujoBean, AppletServlet, BEControlPaquete, Empresa, Proyecto, BEJob, BEPaquete,
Proceso:	1. Seleccione del árbol de navegación de objetos el paquete con nombre “Paquete Prueba”. 2. Elija de la lista de tablas de la Fuente de Datos con nombre “Fuente Prueba” la tabla “Persona”. 3. Click derecho sobre la tabla y seleccione la opción “Agregar tabla al flujo”. 4. Elija de la lista de tablas de la Fuente de Datos con nombre “Fuente Destino - DW” la tabla “Fact_Persona”. 5. Click derecho sobre la tabla y seleccione la opción “Agregar tabla al flujo”. 6. Presione el botón que corresponde al componente “Proceso” 7. Presione en el punto donde se insertará el componente. 8. Presione el botón que corresponde al componente “Conector” 9. Seleccione la tabla “Persona” para indicar el origen.

	10. Seleccione el componente "Proceso" para indicar el destino. 11. Presione el botón que corresponde al componente "Conector" 12. Seleccione el componente "Proceso" para indicar el origen. 13. Seleccione la tabla "Fact_Persona" para indicar el destino. 14. Seleccione la opción "Grabar".
Resultado Esperado:	Se graba exitosamente el flujo y se muestra el área de trabajo sin ningún mensaje de error.

3.16 Personalizar transformación usando objetos activos

Re

Personalizar transformación usando objetos activos – Transformación de datos	
ID Prueba:	BE43
Objetivo :	Configurar transformación mediante el ingreso de datos válidos.
Precondición:	El usuario ha ingresado al sistema. El usuario ha abierto un paquete. El usuario ha creado un flujo de transformación correcto.
Clases :	JSFServlet, BEControlComponente, BETransformación.
Proceso:	1. En el flujo del paquete click derecho en el componente "Proceso" y seleccione "Configurar Transformación". 2. Ingresar el nombre "Transformacion". 3. Ingresar la descripción "prueba". 4. Ingresar la sentencia en el campo idPersona: "Persona.idPersona" 5. Ingresar la sentencia en el campo Apellidos: "Persona.ApellidoPaterno+' '+Persona.ApellidoMaterno" 6. Ingresar la sentencia en el campo Nombre: "Persona.Nombre" 7. Presione el botón "Guardar Transformación".
Resultado Esperado:	Se graba exitosamente la transformación y se muestra el área de trabajo sin mostrar mensaje de error.

Personalizar transformación usando objetos activos – Transformación de datos	
ID Prueba:	BE44
Objetivo :	Verificar la muestra de mensajes de error si se ingresan sentencias incorrectas.
Precondición:	El usuario ha ingresado al sistema. El usuario ha abierto un paquete. El usuario ha creado un flujo de transformación correcto.
Clases :	JSFServlet, BEControlComponente, BETransformación.
Proceso:	1. En el flujo del paquete click derecho en el componente "Proceso" y seleccione "Configurar Transformación". 2. Ingresar el nombre "Tranformacion 2" 3. Ingresar la descripción "prueba 2". 4. Ingresar la sentencia en el campo idPersona: "Persona.idPersona" 5. Ingresar la sentencia en el campo Apellidos: "Persona.ApellidoPaterno++' '+Persona.ApellidoMaterno" 6. Ingresar la sentencia en el campo Nombre: "Persona.Nombre" 7. Presione el botón "Guardar Transformación".
Resultado Esperado:	Se muestra el mensaje "La sentencia es incorrecta para el campo "Apellidos".

Re

Personalizar transformación usando objetos activos – Filtrado de datos	
ID Prueba:	BE45
Objetivo :	Configurar el filtrado de datos mediante el ingreso de datos válidos.
Precondición:	El usuario ha ingresado al sistema. El usuario ha abierto un paquete. Debe haberse creado un flujo de transformación correcto.
Clases :	JSFServlet, BEFiltroBean, BEFiltro, BEProceso, BEControlComponente, BEControlPaquete, BEPaquete, BEGestorComponente, BEComponente.
Proceso:	1. En el flujo del paquete click derecho en el componente "Proceso" y seleccione "Configurar Filtro". 2. Ingrese el nombre "Filtro". 3. Ingrese la descripción "prueba". 4. Seleccione de la lista "Campos Origen" el valor "Código". 5. Seleccione la opción "Bajar Campos". 6. Ingrese en el campo Sentencias, al lado del valor "Código" el valor "=0001". 7. Seleccione la opción "Guardar Filtro".
Resultado Esperado:	Se graba exitosamente el filtro y se muestra el área de trabajo sin ningún mensaje de error.

Personalizar transformación usando objetos activos – Filtrado de datos	
ID Prueba:	BE46
Objetivo :	Verificar la muestra de mensajes de error si se ingresa una sentencia incorrecta.
Precondición:	El usuario ha ingresado al sistema. El usuario ha abierto un paquete. El usuario ha creado un flujo de transformación correcto.
Clases :	JSFServlet, BEFiltroBean, BEFiltro, BEProceso, BEControlComponente, BEControlPaquete, BEPaquete, BEGestorComponente, BEComponente.
Proceso:	1. En el flujo del paquete click derecho en el componente "Proceso" y "Configurar Filtro". 2. Ingrese el nombre "Filtro 2". 3. Ingrese la descripción "prueba". 4. Seleccione de la lista "Campos Origen" el valor "Edad". 5. Seleccione la opción "Bajar Campos". 6. Ingrese en el campo Sentencias, al lado del valor "Edad" el valor ">>20'". 7. Seleccione la opción "Guardar Filtro".
Resultado Esperado:	Se muestra el mensaje "Sentencia con Error de operador incorrecto".

Personalizar transformación usando objetos activos – Estandarización de datos	
ID Prueba:	BE47
Objetivo :	Configurar estandarización mediante el ingreso de datos válidos.
Precondición:	El usuario ha ingresado al sistema. El usuario ha abierto un paquete. El usuario ha creado un flujo de transformación correcto.
Clases :	JSFServlet, BEEstandarizacionBean, BEEstandarizacion, BEProceso, BEControlComponente, BEControlPaquete, BEPaquete, BEGestorComponente, BEComponente.
Proceso:	1. En el flujo del paquete, click derecho en el componente "Proceso" y

	seleccione “Configurar Estandarización”. 2. Ingrese el nombre “Estandarizacion”. 3. Ingrese la descripcion “prueba”. 4. Seleccione de la lista “Campos Destino” el valor “Nombre”. 5. Seleccione la opción “Bajar Campos”. 6. Ingrese en la sentencia la función primMayuscula() y entre los paréntesis coloque el valor “Nombre” del paso 2. 7. Seleccione la opción “Guardar Estandarización”.
Resultado Esperado:	Se guarda exitosamente la estandarización y se muestra el área de trabajo sin ningún mensaje de error.

Personalizar transformación usando objetos activos – Estandarización de datos	
ID Prueba:	BE48
Objetivo :	Verificar la muestra de mensajes de error si se ingresa una sentencia incorrecta.
Precondición:	El usuario ha ingresado al sistema. El usuario ha abierto un paquete. El usuario ha creado un flujo de transformación correcto.
Clases :	JSFServlet, BEEstandarizacionBean, BEEstandarizacion, BEProceso, BEControlComponente, BEControlPaquete, BEPaquete, BEGestorComponente, BEComponente.
Proceso:	1. En el flujo del paquete, click derecho en el componente “Proceso” y seleccione “Configurar Estandarización”. 2. Ingrese el nombre “estandarizacion 2”. 3. Ingrese la descripción “prueba”. 4. Seleccione de la lista “Campos Destino” el valor “Edad”. 5. Seleccione la opción “Bajar Campos”. 6. Ingrese en la sentencia la función primMayuscula(() y entre el segundo y tercer paréntesis coloque el valor “Edad” del paso 2. 7. Seleccione la opción “Guardar Estandarización”.
Resultado Esperado:	Se muestra el mensaje “La sentencia para el campo Edad es incorrecta”.

Personalizar transformación usando objetos activos – Lookup	
ID Prueba:	BE49
Objetivo :	Verificar la correcta configuración del componente lookup cuando tiene una tabla relacionada
Precondición:	El usuario ha ingresado al sistema. El usuario ha abierto un paquete. Existe un flujo de transformación que conecta una tabla resumen a su correspondiente tabla fact en el data mart. Un campo de la tabla fact corresponde al Id de una dimensión relacionada, en la tabla resumen se tiene el valor de esa dimensión . El subcomponente transformación tiene configurada la función directo()
Clases :	JSFSErvlet, BELookUpBean,BECombosRelacion, BELookup, BEControlPaquete, BEControlComponente, BEObjetoActivo, BEFuenteDatos, BEProceso, BERelacionLookup, IDFactory
Proceso:	1. Haga clic derecho en el componente 'Proceso' del flujo de transformación, seleccione la opción "Configurar Lookup". 2. Ingrese un nombre y descripción para el subcomponente. Para el campo de la fact que corresponde al id de la dimension

	relacionada, seleccione el nombre del campo de la tabla origen que contiene el valor a buscar. 3. Seleccione la tabla de la dimensión relacionada en el siguiente combo. 4. Seleccione luego en el tercer combo el campo de la tabla dimension que contiene el valor a buscar. 5. Seleccione en el ultimo combo el campo de la tabla dimension que contiene el id a obtener para insertar en el registro de la tabla fact. 6. Para los demás campos de la fact seleccionar el nombre del campo de la tabla origen que contiene el valor a insertar. Seleccione la opción "Grabar".
Resultado Esperado:	Se guarda exitosamente el lookup y se muestra el área de trabajo sin ningún mensaje de error.

Personalizar transformación usando objetos activos – Lookup	
ID Prueba:	BE50
Objetivo :	Verificar la correcta configuración del componente lookup cuando tiene varias tablas relacionadas.
Precondición:	El usuario ha ingresado al sistema. El usuario ha abierto un paquete. Existe un flujo de transformación que conecta una tabla resumen a su correspondiente tabla fact en el data mart. Varios campos de la tabla fact corresponden al Id de una dimensión relacionada, en la tabla resumen se tiene el valor de esas dimensiones. El subcomponente transformación tiene configurada la función directo()
Clases :	JSFSErvlet, BELookUpBean,BECombosRelacion, BELookup, BEControlPaquete, BEControlComponente, BEObjetoActivo, BEFuenteDatos, BEProceso, BERelacionLookup, IDFactory
Proceso:	1. Haga clic derecho en el componente 'Proceso' del flujo de transformación, seleccione la opción "Configurar Lookup". 2. Ingrese un nombre y descripcion para el subcomponente. 3. Para un campo de la fact que corresponde al id de la dimension relacionada, seleccione el nombre del campo de la tabla origen que contiene el valor a buscar. 4. Seleccione la tabla de la dimension relacionada en el siguiente combo. 5. Seleccione luego en el tercer combo el campo de la tabla dimension que contiene el valor a buscar. 6. Seleccione en el ultimo combo el campo de la tabla dimension que contiene el id a obtener para insertar en el registro de la tabla fact. 7. Repetir los pasos 2-6 para los demas campos de la fact relacionados a dimensiones. 8. Para los demas campos de la fact seleccionar el nombre del campo de la tabla origen que contiene el valor a insertar. 9. Seleccione la opción "Grabar".
Resultado Esperado:	Se guarda exitosamente el lookup y se muestra el área de trabajo sin ningún mensaje de error.

Personalizar transformación usando objetos activos – Lookup	
ID Prueba:	BE51
Objetivo :	Verificar la correcta configuración del componente lookup cuando tiene una tabla relacionada

Precondición:	El usuario ha ingresado al sistema. El usuario ha abierto un paquete. Existe un flujo de transformación que conecta una tabla resumen a su correspondiente tabla fact en el data mart. El subcomponente transformación tiene configurada la función directo()
Clases :	JSFSErvlet, BELookUpBean,BECombosRelacion, BELookup, BEControlPaquete, BEControlComponente, BEObjetoActivo, BEFuenteDatos, BEProceso, BERelacionLookup, IDFactory
Proceso:	1. Haga clic derecho en el componente 'Proceso' del flujo de transformación, seleccione la opción "Configurar Lookup". 2. Ingrese un nombre y descripcion para el subcomponente. 3. Para cada campo de la fact seleccionar el nombre del campo de la tabla origen que contiene el valor a insertar. 4. Seleccione la opción "Grabar".
Resultado Esperado:	Se guarda exitosamente el lookup y se muestra el área de trabajo sin ningún mensaje de error.

Personalizar transformación usando objetos activos – Configurar mediante script	
ID Prueba:	BE52
Objetivo :	Configurar script mediante el ingreso de un procedimiento de una Fuente de Datos tipo ORACLE.
Precondición:	El usuario ha ingresado al sistema. El usuario ha abierto un paquete. El usuario ha creado un flujo de transformación correcto. Existe la fuente de datos “BiOracle” de tipo Base de Datos Oracle configurada correctamente.
Clases :	JSFServlet, BEScriptBean, BEScript, BEControlPaquete, BEPaquete, Empresa, Proyecto, BEJob, BEGestorComponente.
Proceso:	1. Presione el botón que corresponde al componente “Script” 2. Presione en el punto donde se insertará el componente. 3. Click derecho en el componente “Script” y seleccione “Configurar Script”. 4. Seleccione “Utilizar Script Existente” y “Buscar Procedimiento”. 5. Elija del combobox de “Fuente de Datos” el valor “BiORACLE” de la lista. 6. Elija de la lista de procedimientos el valor “Script Prueba”. 7. Seleccione la opción “Guardar Script”.
Resultado Esperado:	Se guarda exitosamente el script, y se muestra el área de trabajo sin ningún mensaje de error.

Personalizar transformación usando objetos activos – Configurar mediante script	
ID Prueba:	BE53
Objetivo :	Configurar script mediante el ingreso de un procedimiento de una Fuente de Datos tipo SQL Server.
Precondición:	El usuario ha ingresado al sistema. El usuario ha abierto un paquete. El usuario ha creado un flujo de transformación correcto. Existe la fuente de datos “BiSqlSer” de tipo Base de Datos SQL Server configurada correctamente.

Clases :	JSFServlet, BEScriptBean, BEScript, BEControlPaquete, BEPaquete, Empresa, Proyecto, BEJob, BEGestorComponente.
Proceso:	1. Presione el botón que corresponde al componente “Script” 2. Presione en el punto donde se insertará el componente. 3. Click derecho en el componente “Script” y seleccione “Configurar Script”. 4. Seleccione “Utilizar Script Existente” y “Buscar Procedimiento”. 5. Elija del combobox de “Fuente de Datos” el valor “BiSqlSer” de la lista. 6. Elija de la lista de procedimientos el valor “Script Prueba”. 7. Seleccione la opción “Guardar Script”.
Resultado Esperado:	Se guarda exitosamente el script, y se muestra el área de trabajo sin ningún mensaje de error.

Personalizar transformación usando objetos activos – Configurar mediante script	
ID Prueba:	BE54
Objetivo :	Configurar script mediante el ingreso de un procedimiento de una Fuente de Datos tipo MySQL.
Precondición:	El usuario ha ingresado al sistema. El usuario ha abierto un paquete. El usuario ha creado un flujo de transformación correcto. Existe la fuente de datos “BiMySql” de tipo Base de Datos MySQL configurada correctamente.
Clases :	JSFServlet, BEScriptBean, BEScript, BEControlPaquete, BEPaquete, Empresa, Proyecto, BEJob, BEGestorComponente.
Proceso:	1. Presione el botón que corresponde al componente “Script” 2. Presione en el punto donde se insertará el componente. 3. Click derecho en el componente “Script” y seleccione “Configurar Script”. 4. Seleccione “Utilizar Script Existente” y “Buscar Procedimiento”. 5. Elija del combobox de “Fuente de Datos” el valor “BiMySql” de la lista. 6. Elija de la lista de procedimientos el valor “Script Prueba”. 7. Seleccione la opción “Guardar Script”.
Resultado Esperado:	Se guarda exitosamente el script, y se muestra el área de trabajo sin ningún mensaje de error.

Personalizar transformación usando objetos activos – Configurar mediante script	
ID Prueba:	BE55
Objetivo :	Configurar script mediante el ingreso de una sentencia correcta.
Precondición:	El usuario ha ingresado al sistema. El usuario ha abierto un paquete. El usuario ha creado un flujo de transformación correcto.
Clases :	.JSFServlet, BEScriptBean, BEScript, BEControlPaquete, BEPaquete, Empresa, Proyecto, BEJob, BEGestorComponente
Proceso:	1. Presione el botón que corresponde al componente “Script” 2. Presione en el punto donde se insertará el componente. 3. Click derecho en el componente “Script” y seleccione “Configurar Script”. 4. Ingrese en el campo “Sentencia” el valor “Select * from Persona”.

	5. Elija del combobox de “Fuente de Datos” el valor “Bipucp” de la lista. 6. Seleccione la opción “Grabar”.
Resultado Esperado:	Se guarda exitosamente el script y se regresa al área de trabajo sin mostrar ningún mensaje de error.

Personalizar transformación usando objetos activos – Configurar mediante script	
ID Prueba:	BE56
Objetivo :	Verificar la muestra de mensajes de error si se ingresa datos inválidos para configurar el script.
Precondición:	El usuario ha ingresado al sistema. El usuario ha abierto un paquete. Debe haberse creado un flujo de transformación correcto.
Clases :	JSFServlet, BEScriptBean, BEScript, BEControlPaquete, BEPaquete, Empresa, Proyecto, BEJob, BEGestorComponente
Proceso:	1. Presione el botón que corresponde al componente “Script” 2. Presione en el punto donde se insertará el componente. 3. Click derecho en el componente “Script” y seleccione “Configurar Script”. 4. Deje vacío el campo “Sentencia”. 5. Elija del combobox de “Fuente de Datos” el valor “Bipucp” de la lista. 6. Seleccione la opción “Grabar”.
Resultado Esperado:	Se muestra el mensaje “Valor incorrecto” para la sentencia.

3.17 Validar script mediante parser

Validar script mediante parser	
ID Prueba:	BE57
Objetivo :	Validar un script mediante el ingreso de una sentencia correcta.
Precondición:	El usuario ha ingresado al sistema. El usuario ha abierto un paquete. Debe haberse creado un flujo de transformación correcto.
Clases :	.JSFServlet, BEScriptBean, BEScript, BEControlPaquete, BEPaquete, Empresa, Proyecto, BEJob, BEGestorComponente, BEScriptParser
Proceso:	1. Ingrese en el campo “Sentencia” el valor “select * from Persona”. 2. Elija del combobox de “Fuente de Datos” el valor “Bipucp” de la lista. 3. Seleccione “Guardar Script”.
Resultado Esperado:	Se regresa al área de trabajo sin mostrar ningún mensaje de error.

Validar script mediante parser	
ID Prueba:	BE58
Objetivo :	Verificar la muestra de mensajes de error cuando se ingresa una sentencia incorrecta.
Precondición:	El usuario ha ingresado al sistema. El usuario ha abierto un paquete. Debe haberse creado un flujo de transformación correcto.
Clases :	.JSFServlet, BEScriptBean, BEScript, BEControlPaquete, BEPaquete, Empresa, Proyecto, BEJob, BEGestorComponente, BEScriptParser
Proceso:	1. Ingrese en el campo “Sentencia” el valor “select from Persona”. 2. Elija del combobox de “Fuente de Datos” el valor “Bipucp” de la lista. 3. Seleccione “Guardar Script”.
Resultado Esperado:	Se muestra un mensaje de error indicando el lugar en la sentencia donde se detectó el problema.

3.18 Crear funciones definidas por el usuario

Re

Crear funciones definidas por el usuario	
ID Prueba:	BE59
Objetivo :	Crear una función tipo Regla mediante el ingreso de datos válidos
Precondición:	El usuario ha ingresado al Sistema. El usuario ha abierto un paquete.
Clases :	JSFServlet, BEFuncionBean, BEFuncionDefUsuario, BEGestorFuncion, BEPaquete, IdFactory,
Proceso:	1. En el menú de Paquete seleccione la opción “Funciones definidas por el usuario” y “Crear”. 2. Ingrese en el campo “Nombre” el valor “Función prueba”. 3. Ingrese en el campo “Descripción” el valor “Función creada por el usuario para caso de prueba”. 4. Seleccione en el campo “Tipo” el valor “Regla”. 5. Seleccione la opción “Agregar”. 6. Ingrese en el campo “Valor Entrada” el valor “M”. 7. Ingrese en el campo “Valor Salida” el valor “1”. 8. Seleccione la opción “Agregar”. 9. Ingrese en el campo “Valor Entrada” el valor “F”. 10. Ingrese en el campo “Valor Salida” el valor “0”. 11. Seleccione la opción “Agregar”. 12. Ingrese en el campo “Valor Entrada” el valor “Masc”. 13. Ingrese en el campo “Valor Salida” el valor “1”. 14. Seleccione la opción “Grabar”.
Resultado Esperado:	Se guarda exitosamente la función y se muestra en el formulario de “Administración de Funciones” la nueva función creada.

Crear funciones definidas por el usuario	
ID Prueba:	BE60
Objetivo :	Crear una función tipo Sentencia mediante el ingreso de datos válidos
Precondición:	El usuario ha ingresado al Sistema. El usuario ha abierto un paquete.
Clases :	JSFServlet, BEFuncionBean, BEFuncionDefUsuario, BEGestorFuncion, BEPaquete, IdFactory,

Proceso:	1. En el menú de Paquete seleccione la opción “Funciones definidas por el usuario” y “Crear”. 2. Ingrese en el campo “Nombre” el valor “Función prueba”. 3. Ingrese en el campo “Descripción” el valor “Función creada por el usuario para caso de prueba”. 4. Seleccione en el campo “Tipo” el valor “Sentencia”. 5. Seleccione en el campo “Nro. de parámetros de Entrada” el valor “2” 6. Seleccione en el campo “Salida de tipo” el valor “Real”. 7. Seleccione para los tipos de los parámetros de entrada:”Entero” y “Real” para los 2 parámetros respectivamente. 8. Ingrese en el campo “Sentencia” el valor “(x1 * x1) / x2”. 9. Seleccione la opción “Grabar”.
Resultado Esperado:	Se guarda exitosamente la función y se muestra en el formulario de “Administración de Funciones” la nueva función creada.

Crear funciones definidas por el usuario	
ID Prueba:	BE61
Objetivo :	Crear una función tipo Sentencia mediante el ingreso de datos inválidos
Precondición:	El usuario ha ingresado al Sistema. El usuario ha abierto un paquete.
Clases :	JSFServlet, BEFuncionBean, BEFuncionDefUsuario, BEGestorFuncion, BEPaquete, IdFactory,
Proceso:	1. En el menú de Paquete seleccione la opción “Funciones definidas por el usuario” y “Crear”. 2. Ingrese en el campo “Nombre” el valor “Función prueba”. 3. Ingrese en el campo “Descripción” el valor “Función creada por el usuario para caso de prueba”. 4. Seleccione en el campo “Tipo” el valor “Sentencia”. 5. Seleccione en el campo “Nro. de parámetros de Entrada” el valor “2” 6. Seleccione en el campo “Salida de tipo” el valor “Real”. 7. Seleccione para los tipos de los parámetros de entrada:”Entero” y “Real” para los 2 parámetros respectivamente. 8. Ingrese en el campo “Sentencia” el valor “((x1 * x1) / x2”. 9. Seleccione la opción “Grabar”.
Resultado Esperado:	Se muestra un mensaje de error “Sentencia Incorrecta”.

3.19 Modificar Funciones Definidas por el usuario

Modificar funciones definidas por el usuario	
ID Prueba:	BE62
Objetivo :	Modificar una función tipo Sentencia mediante el ingreso de datos válidos
Precondición:	El usuario ha ingresado al Sistema. El usuario ha abierto un paquete.
Clases :	JSFServlet, BEFuncionBean, BEFuncionDefUsuario, BEGestorFuncion, BEPaquete, IdFactory,
Proceso:	1. En el menú de Paquete seleccione la opción “Funciones definidas por el usuario”. 2. Seleccione la función “Funcion prueba” haciendo click sobre el nombre. 3. Modifique el campo “Nombre” con el valor “Función 2”.

	4. Modifique el campo “Sentencia” ingresando el valor “(x1 * x1) /x2 - x1”. 5. Seleccione la opción “Grabar”.
Resultado Esperado:	Se modifica exitosamente la función y se muestra en el formulario de “Administración de Funciones” la función modificada.

Modificar funciones definidas por el usuario	
ID Prueba:	BE63
Objetivo :	Modificar una función tipo Regla mediante el ingreso de datos válidos
Precondición:	El usuario ha ingresado al Sistema. El usuario ha abierto un paquete.
Clases :	JSFServlet, BEFuncionBean, BEFuncionDefUsuario, BEGestorFuncion, BEPaquete, IdFactory,
Proceso:	1. En el menú de Paquete seleccione la opción “Funciones definidas por el usuario”. 2. Seleccione la función “Funcion prueba” haciendo click sobre el nombre. 3. Modifique el campo “Nombre” ingresando el valor “Función 2”. 4. Seleccione la opción “Agregar”. 5. Ingrese en el campo “Valor Entrada” el valor “Femen”. 6. Ingrese en el campo “Valor Salida” el valor “0”. 7. Seleccione la opción “Grabar”.
Resultado Esperado:	Se modifica exitosamente la función y se muestra en el formulario de “Administración de Funciones” la función modificada.

3.20 Eliminar Funciones Definidas por el Usuario

Eliminar funciones definidas por el usuario	
ID Prueba:	BE64
Objetivo :	Verificar la correcta eliminación de una función definida por el usuario.
Precondición:	El usuario ha ingresado al Sistema. El usuario ha abierto un paquete.
Clases :	JSFServlet, BEFuncionBean, BEFuncionDefUsuario, BEGestorFuncion, BEPaquete, IdFactory,
Proceso:	1. En el menú de Paquete seleccione la opción “Funciones definidas por el usuario”. 2. Seleccione la función “Funcion prueba” haciendo click sobre la casilla de la izquierda del nombre. 3. Seleccione la opción “Eliminar”.
Resultado Esperado:	Se elimina exitosamente la función y se muestra en el formulario de “Administración de Funciones” la lista de funciones actualizada sin contener la función “Funcion Prueba”.

3.21 Programar jobs

R

Programar Jobs	
ID Prueba:	BE65

Objetivo :	Crear una Programación mediante el ingreso de datos válidos para una programación Diaria.
Precondición:	El usuario ha ingresado al sistema. El usuario ha abierto un proyecto.
Clases :	JSFServlet, BEProgramacionBean, BEProgramacion, BEControlProgramacion, BEJob, Empresa, Proyecto, IdFactory.
Proceso:	1. En el menú, seleccione la opción “Programar Jobs” y “Crear”. 2. Ingrese en el campo “Nombre” el valor “Programacion Prueba”. 3. Ingrese en el campo “Descripción” el valor “Programacion para el caso de prueba”. 4. Elija del combobox de “Job” el valor “JobPrueba”.de la lista. 5. Ingrese en el campo “Hora de ejecución” el valor “20:00”. 6. Elija del combobox de “Periodicidad” el valor “Diaria” de la lista. 7. Seleccione los días “Lunes” y “Miércoles”. 8. Seleccione la opción “Grabar”.
Resultado Esperado:	Se guarda exitosamente la programación y se muestra en el formulario de “Administración de Programaciones” la nueva programación creada.

Programar Jobs	
ID Prueba:	BE66
Objetivo :	Crear una Programación mediante el ingreso de datos válidos para una programación Semanal.
Precondición:	El usuario ha ingresado al sistema. El usuario ha abierto un proyecto.
Clases :	JSFServlet, BEProgramacionBean, BEProgramacion, BEControlProgramacion, BEJob, Empresa, Proyecto, IdFactory.
Proceso:	1. En el menú, seleccione la opción “Programar Jobs” y “Crear”. 2. Ingrese en el campo “Nombre” el valor “Programacion Prueba”. 3. Ingrese en el campo “Descripción” el valor “Programacion para el caso de prueba”. 4. Elija del combobox de “Job” el valor “JobPrueba”.de la lista. 5. Ingrese en el campo “Hora de ejecución” el valor “20:00”. 6. Elija del combobox de “Periodicidad” el valor “Semanal” de la lista. 7. Seleccione ejecutar los Lunes cada 2 semanas. 8. Seleccione la opción “Grabar”.
Resultado Esperado:	Se guarda exitosamente la programación y se muestra en el formulario de “Administración de Programaciones” la nueva programación creada.

Programar Jobs	
ID Prueba:	BE67
Objetivo :	Crear una Programación mediante el ingreso de datos válidos para una programación Mensual.
Precondición:	El usuario ha ingresado al sistema. El usuario ha abierto un proyecto.
Clases :	JSFServlet, BEProgramacionBean, BEProgramacion, BEControlProgramacion, BEJob, Empresa, Proyecto, IdFactory.
Proceso:	1. En el menú, seleccione la opción “Programar Jobs” y “Crear”. 2. Ingrese en el campo “Nombre” el valor “Programacion Prueba”. 3. Ingrese en el campo “Descripción” el valor “Programacion para el caso de prueba”. 4. Elija del combobox de “Job” el valor “JobPrueba”.de la lista. 5. Ingrese en el campo “Hora de ejecución” el valor “20:00”.

	6. Elija del combobox de “Periodicidad” el valor “Mensual” de la lista. 7. Seleccione ejecutar el 2 de cada mes, cada 3 meses. 8. Seleccione la opción “Grabar”.
Resultado Esperado:	Se guarda exitosamente la programación y se muestra en el formulario de “Administración de Programaciones” la nueva programación creada.

Programar Jobs	
ID Prueba:	BE68
Objetivo :	Crear una Programación mediante el ingreso de datos válidos para una programación Anual.
Precondición:	El usuario ha ingresado al sistema. El usuario ha abierto un proyecto.
Clases :	JSFServlet, BEProgramacionBean, BEProgramacion, BEControlProgramacion, BEJob, Empresa, Proyecto, IdFactory.
Proceso:	1. En el menú, seleccione la opción “Programar Jobs” y “Crear”. 2. Ingrese en el campo “Nombre” el valor “Programacion Prueba”. 3. Ingrese en el campo “Descripción” el valor “Programacion para el caso de prueba”. 4. Elija del combobox de “Job” el valor “JobPrueba”.de la lista. 5. Ingrese en el campo “Hora de ejecución” el valor “20:00”. 6. Elija del combobox de “Periodicidad” el valor “Anual” de la lista. 7. Seleccione ejecutar el 3 de Abril, cada 2 años. 8. Seleccione la opción “Grabar”.
Resultado Esperado:	Se guarda exitosamente la programación y se muestra en el formulario de “Administración de Programaciones” la nueva programación creada.

Programar Jobs	
ID Prueba:	BE69
Objetivo :	Verificar la muestra de mensajes de error si no se ingresa el campo “Nombre”
Precondición:	El usuario ha ingresado al sistema. El usuario ha abierto un proyecto.
Clases :	JSFServlet, BEProgramacionBean, BEProgramacion, BEControlProgramacion, BEJob, Empresa, Proyecto, IdFactory.
Proceso:	1. En el menú, seleccione la opción “Programar Jobs” y “Crear”. 2. Deje vacío el campo “Nombre”. 3. Ingrese en el campo “Descripción” el valor “Programacion para el caso de prueba”. 4. Elija del combobox de “Job” el valor “JobPrueba”.de la lista. 5. Ingrese en el campo “Hora de ejecución” el valor “20:00”. 6. Elija del combobox de “Periodicidad” el valor “Diaria” de la lista. 7. Seleccione los días “Lunes” y “Miércoles”. 8. Seleccione la opción “Grabar”.
Resultado Esperado:	Se muestra el mensaje “Valor Incorrecto” para el campo “Nombre”.

Programar Jobs	
ID Prueba:	BE70
Objetivo :	Verificar la muestra de mensajes de error si se ingresa en el campo “Hora” un valor incorrecto.
Precondición:	El usuario ha ingresado al sistema.

	El usuario ha abierto un proyecto.
Clases :	JSFServlet, BEProgramacionBean, BEProgramacion, BEControlProgramacion, BEJob, Empresa, Proyecto, IdFactory.
Proceso:	1. En el menú, seleccione la opción “Programar Jobs” y “Crear”. 2. Ingrese en el campo “Nombre” el valor “Programacion 2”. 3. Ingrese en el campo “Descripción” el valor “Programacion para el caso de prueba”. 4. Elija del combobox de “Job” el valor “JobPrueba”.de la lista. 5. Ingrese en el campo “Hora de ejecución” el valor “24:60”. 6. Elija del combobox de “Periodicidad” el valor “Diaria” de la lista. 7. Seleccione los días “Lunes” y “Miércoles”. 8. Seleccione la opción “Grabar”.
Resultado Esperado:	Se muestra el mensaje “Esta hora no existe” para el campo “hora”.

Programar Jobs	
ID Prueba:	BE71
Objetivo :	Verificar la correcta creación de una programación con parámetros.
Precondición:	El usuario ha ingresado al sistema. El usuario ha abierto un proyecto. El job “Job Param” contiene parámetros.
Clases :	JSFServlet, BEProgramacionBean, BEProgramacion, BEControlProgramacion, BEJob, Empresa, Proyecto, IdFactory.
Proceso:	1. En el menú, seleccione la opción “Programar Jobs” y “Crear”. 2. Ingrese en el campo “Nombre” el valor “Programacion Prueba”. 3. Ingrese en el campo “Descripción” el valor “Programacion para el caso de prueba”. 4. Elija del combobox de “Job” el valor “Job Param”.de la lista. 5. Ingrese en el campo “Hora de ejecución” el valor “20:00”. 6. Elija del combobox de “Periodicidad” el valor “Diaria” de la lista. 7. Seleccione los días “Lunes” y “Miércoles”. 8. Según el job seleccionado “Job Param”, ingrese los valores de los parámetros: para el parámetro “User” ingrese el valor “biextrausr” y para el parámetro “Password” ingrese el valor “biextrapwd”. 9. Seleccione la opción “Grabar”.
Resultado Esperado:	Se guarda exitosamente la programación y se muestra en el formulario de “Administración de Programaciones” la nueva programación creada.

Programar Jobs	
ID Prueba:	BE72
Objetivo :	Verificar la correcta creación de varias programaciones para un job.
Precondición:	El usuario ha ingresado al sistema. El usuario ha abierto un proyecto. El job “Job Prueba” contiene una programación Semanal
Clases :	JSFServlet, BEProgramacionBean, BEProgramacion, BEControlProgramacion, BEJob, Empresa, Proyecto, IdFactory.
Proceso:	1. En el menú, seleccione la opción “Programar Jobs” y “Crear”. 2. Ingrese en el campo “Nombre” el valor “Programacion Prueba”. 3. Ingrese en el campo “Descripción” el valor “Programacion para el caso de prueba”. 4. Elija del combobox de “Job” el valor “JobPrueba”.de la lista, el cual ya contiene una programación semanal. 5. Ingrese en el campo “Hora de ejecución” el valor “20:00”.

	6. Elija del combobox de “Periodicidad” el valor “Diaria” de la lista. 7. Seleccione los días “Lunes” y “Miércoles”. 8. Según el job seleccionado “JobPrueba”, ingrese los valores de los parámetros: para el parámetro “User” ingrese el valor “biextrausr” y para el parámetro “Password” ingrese el valor “biextrapwd”. 9. Seleccione la opción “Grabar”.
Resultado Esperado:	Se guarda exitosamente la programación y se muestra en el formulario de “Administración de Programaciones” la nueva programación creada.

3.22 Modificar Programación

Modificar Programación	
ID Prueba:	BE73
Objetivo :	Modificar una Programación mediante el ingreso de datos válidos.
Precondición:	El usuario ha ingresado al sistema. El usuario ha abierto un proyecto.
Clases :	JSFServlet, BEProgramacionBean, BEProgramacion, BEControlProgramacion, BEJob, Empresa, Proyecto, IdFactory.
Proceso:	1. En el menú, seleccione la opción “Programar Jobs” 2. Seleccione la programacion “Programacion Prueba” haciendo click sobre el nombre. 3. Modifique el campo “Nombre” ingresando el nuevo valor “Programacion 2”. 4. Modifique el combobox de “Periodicidad” al valor “Mensual” de la lista. 5. Seleccione ejecutar el 5 de cada mes, cada 5 meses. 6. Seleccione la opción “Grabar”.
Resultado Esperado:	Se guarda exitosamente la programación y se muestra en el formulario de “Administración de Programaciones” la nueva programación creada.

Modificar Programación	
ID Prueba:	BE74
Objetivo :	Modificar una Programación mediante el ingreso de datos inválidos.
Precondición:	El usuario ha ingresado al sistema. El usuario ha abierto un proyecto.
Clases :	JSFServlet, BEProgramacionBean, BEProgramacion, BEControlProgramacion, BEJob, Empresa, Proyecto, IdFactory.
Proceso:	1. En el menú, seleccione la opción “Programar Jobs” 2. Seleccione la programacion “Programacion Prueba” haciendo click sobre el nombre. 3. Modifique el campo “Nombre” dejándolo vacío. 4. Seleccione la opción “Grabar”.
Resultado Esperado:	Se muestra un mensaje de error “Valor Incorrecto” para el campo “Nombre”

Modificar Programación	
ID Prueba:	BE75
Objetivo :	Verificar la muestra de mensajes de error si se modifica el nombre de una programación ingresando un valor previamente existente.
Precondición:	El usuario ha ingresado al sistema. El usuario ha abierto un proyecto.
Clases :	JSFServlet, BEProgramacionBean, BEProgramacion,

	BEControlProgramacion, BEJob, Empresa, Proyecto, IdFactory.
Proceso:	1. En el menú, seleccione la opción “Programar Jobs”. 2. Cree una programación con nombre “Programacion 3”. 3. Seleccione la programacion “Programacion 2” haciendo click sobre el nombre. 4. Modifique el campo “Nombre” ingresando el nuevo valor “Programacion 3”, que es el nombre de la programación creada en el paso 2. 5. Seleccione la opción “Grabar”.
Resultado Esperado:	Se muestra un mensaje de error “Nombre de programación ya existe” para el campo “Nombre”.

3.23 Eliminar Programación

Eliminar Programación	
ID Prueba:	BE76
Objetivo :	Verificar la correcta eliminación de una programación.
Precondición:	El usuario ha ingresado al sistema. El usuario ha abierto un proyecto.
Clases :	JSFServlet, BEProgramacionBean, BEProgramacion, BEControlProgramacion, BEJob, Empresa, Proyecto, IdFactory.
Proceso:	1. En el menú, seleccione la opción “Programar Jobs”. 2. Seleccione la programacion “Programacion 3” haciendo click sobre la casilla de la izquierda del nombre. 3. Seleccione la opción “Eliminar”.
Resultado Esperado:	Se muestra el listado de programaciones actualizado sin contener la programación “Programacion 3”.

3.24 Ejecutar Job

Re

Ejecutar job	
ID Prueba:	BE77
Objetivo :	Ejecutar un job manualmente mediante el ingreso de datos válidos, el job posee parámetros.
Precondición:	El usuario ha ingresado al Sistema. El usuario ha abierto un proyecto. El usuario ha creo un job correctamente. El usuario ha abierto un job. El job posee un parámetro denominado “conexion”.
Clases :	JSFServlet, BEVerEjecucionBean, BEControlEjecucion, BERendimientoEjecucion, BEEjecucionJob, BEControlJob, BEJob, BEProgramacion, ProgServlet.
Proceso:	1. En el menú, seleccione la opción “Ejecutar Job”. 2. Ingrese el valor “inti” al parámetro “conexion”. 3. Seleccione la opción “Ejecutar”.
Resultado Esperado:	Se muestra en pantalla las tareas que realiza el Job, indicando fecha de inicio, paquetes y componentes ejecutados.

Ejecutar job	
ID Prueba:	BE78

Objetivo :	Ejecutar un job manualmente mediante el ingreso de datos válidos, el job no posee parámetros
Precondición:	El usuario ha ingresado al Sistema. El usuario ha abierto un proyecto. El usuario ha creo un job correctamente. El usuario ha abierto un job.
Clases :	JSFServlet, BEVerEjecucionBean, BEControlEjecucion, BERendimientoEjecucion, BEEjecucionJob, BEControlJob, BEJob, BEProgramacion, ProgServlet.
Proceso:	1. En el menú, seleccione la opción “Ejecutar Job”. 2. En el siguiente formulario seleccione la opción “Ejecutar”.
Resultado Esperado:	Se muestra en pantalla las tareas que realiza el Job, indicando fecha de inicio, paquetes y componentes ejecutados.

Ejecutar job	
ID Prueba:	BE79
Objetivo :	Ejecutar un job manualmente que tiene como origen un archivo txt y destino un archivo txt.
Precondición:	El usuario ha ingresado al Sistema. El usuario ha abierto un proyecto. El usuario ha creo un job correctamente. El usuario ha abierto un job. Los archivos origen y destino han sido configurados correctamente.
Clases :	JSFServlet, BEVerEjecucionBean, BEControlEjecucion, BERendimientoEjecucion, BEEjecucionJob, BEControlJob, BEJob, BEProgramacion, ProgServlet.
Proceso:	1. Agregue como origen al flujo un archivo tipo TXT. 2. Agregue como destino al flujo un archivo tipo TXT. 3. Agregue un componente “Proceso” al flujo que una el origen con el destino. 4. Ingrese alguna operación a realizar en la configuración del objeto activo “Proceso”. 5. En el menú, seleccione la opción “Ejecutar Job”. 6. En el siguiente formulario seleccione la opción “Ejecutar”.
Resultado Esperado:	Se muestra en pantalla las tareas que realiza el Job, indicando fecha de inicio, paquetes y componentes ejecutados.

Ejecutar job	
ID Prueba:	BE80
Objetivo :	Ejecutar un job manualmente que tiene como origen un archivo txt y destino un archivo xml.
Precondición:	El usuario ha ingresado al Sistema. El usuario ha abierto un proyecto. El usuario ha creo un job correctamente. El usuario ha abierto un job. Los archivos origen y destino han sido configurados correctamente.
Clases :	JSFServlet, BEVerEjecucionBean, BEControlEjecucion, BERendimientoEjecucion, BEEjecucionJob, BEControlJob, BEJob,

	BEProgramacion, ProgServlet.
Proceso:	1. Agregue como origen al flujo un archivo tipo TXT. 2. Agregue como destino al flujo un archivo tipo XML. 3. Agregue un componente “Proceso” al flujo que una el origen con el destino. 4. Ingrese alguna operación a realizar en la configuración del objeto activo “Proceso”. 5. En el menú, seleccione la opción “Ejecutar Job”. 6. En el siguiente formulario seleccione la opción “Ejecutar”.
Resultado Esperado:	Se muestra en pantalla las tareas que realiza el Job, indicando fecha de inicio, paquetes y componentes ejecutados.

Ejecutar job	
ID Prueba:	BE81
Objetivo :	Ejecutar un job manualmente que tiene como origen un archivo txt y destino un archivo xls.
Precondición:	El usuario ha ingresado al Sistema. El usuario ha abierto un proyecto. El usuario ha creo un job correctamente. El usuario ha abierto un job. Los archivos origen y destino han sido configurados correctamente.
Clases :	JSFServlet, BEVerEjecucionBean, BEControlEjecucion, BERendimientoEjecucion, BEEjecucionJob, BEControlJob, BEJob, BEProgramacion, ProgServlet.
Proceso:	1. Agregue como origen al flujo un archivo tipo TXT. 2. Agregue como destino al flujo un archivo tipo XLS. 3. Agregue un componente “Proceso” al flujo que una el origen con el destino. 4. Ingrese alguna operación a realizar en la configuración del objeto activo “Proceso”. 5. En el menú, seleccione la opción “Ejecutar Job”. 6. En el siguiente formulario seleccione la opción “Ejecutar”.
Resultado Esperado:	Se muestra en pantalla las tareas que realiza el Job, indicando fecha de inicio, paquetes y componentes ejecutados.

Ejecutar job	
ID Prueba:	BE82
Objetivo :	Ejecutar un job manualmente que tiene como origen un archivo xml y destino un archivo txt.
Precondición:	El usuario ha ingresado al Sistema. El usuario ha abierto un proyecto. El usuario ha creo un job correctamente. El usuario ha abierto un job. Los archivos origen y destino han sido configurados correctamente.
Clases :	JSFServlet, BEVerEjecucionBean, BEControlEjecucion, BERendimientoEjecucion, BEEjecucionJob, BEControlJob, BEJob, BEProgramacion, ProgServlet.

Proceso:	1. Agregue como origen al flujo un archivo tipo XML. 2. Agregue como destino al flujo un archivo tipo TXT. 3. Agregue un componente “Proceso” al flujo que una el origen con el destino. 4. Ingrese alguna operación a realizar en la configuración del objeto activo “Proceso”. 5. En el menú, seleccione la opción “Ejecutar Job”. 6. En el siguiente formulario seleccione la opción “Ejecutar”.
Resultado Esperado:	Se muestra en pantalla las tareas que realiza el Job, indicando fecha de inicio, paquetes y componentes ejecutados.

Ejecutar job	
ID Prueba:	BE83
Objetivo :	Ejecutar un job manualmente que tiene como origen un archivo xls y destino un archivo txt.
Precondición:	El usuario ha ingresado al Sistema. El usuario ha abierto un proyecto. El usuario ha creo un job correctamente. El usuario ha abierto un job. Los archivos origen y destino han sido configurados correctamente.
Clases :	JSFServlet, BEVerEjecucionBean, BEControlEjecucion, BERendimientoEjecucion, BEEjecucionJob, BEControlJob, BEJob, BEProgramacion, ProgServlet.
Proceso:	1. Agregue como origen al flujo un archivo tipo XLS. 2. Agregue como destino al flujo un archivo tipo TXT. 3. Agregue un componente “Proceso” al flujo que una el origen con el destino. 4. Ingrese alguna operación a realizar en la configuración del objeto activo “Proceso”. 5. En el menú, seleccione la opción “Ejecutar Job”. 6. En el siguiente formulario seleccione la opción “Ejecutar”.
Resultado Esperado:	Se muestra en pantalla las tareas que realiza el Job, indicando fecha de inicio, paquetes y componentes ejecutados.

Ejecutar job	
ID Prueba:	BE84
Objetivo :	Ejecutar un job manualmente que tiene como origen una table de base de datos y destino un archivo xls.
Precondición:	El usuario ha ingresado al Sistema. El usuario ha abierto un proyecto. El usuario ha creo un job correctamente. El usuario ha abierto un job. Los archivos origen y destino han sido configurados correctamente.
Clases :	JSFServlet, BEVerEjecucionBean, BEControlEjecucion, BERendimientoEjecucion, BEEjecucionJob, BEControlJob, BEJob, BEProgramacion, ProgServlet.

Proceso:	1. Agregue como origen al flujo una tabla de base de datos. 2. Agregue como destino al flujo un archivo tipo XLS. 3. Agregue un componente “Proceso” al flujo que una el origen con el destino. 4. Ingrese alguna operación a realizar en la configuración del objeto activo “Proceso”. 5. En el menú, seleccione la opción “Ejecutar Job”. 6. En el siguiente formulario seleccione la opción “Ejecutar”.
Resultado Esperado:	Se muestra en pantalla las tareas que realiza el Job, indicando fecha de inicio, paquetes y componentes ejecutados.

Ejecutar job	
ID Prueba:	BE85
Objetivo :	Ejecutar un job manualmente que tiene como origen una tabla de base de datos y destino otra tabla de base de datos.
Precondición:	El usuario ha ingresado al Sistema. El usuario ha abierto un proyecto. El usuario ha creo un job correctamente. El usuario ha abierto un job. Los archivos origen y destino han sido configurados correctamente.
Clases :	JSFServlet, BEVerEjecucionBean, BEControlEjecucion, BERendimientoEjecucion, BEEjecucionJob, BEControlJob, BEJob, BEProgramacion, ProgServlet.
Proceso:	1. Agregue como origen al flujo una tabla de base de datos 2. Agregue como destino al flujo una tabla de base de datos. 3. Agregue un componente “Proceso” al flujo que una el origen con el destino. 4. Ingrese alguna operación a realizar en la configuración del objeto activo “Proceso”. 5. En el menú, seleccione la opción “Ejecutar Job”. 6. En el siguiente formulario seleccione la opción “Ejecutar”.
Resultado Esperado:	Se muestra en pantalla las tareas que realiza el Job, indicando fecha de inicio, paquetes y componentes ejecutados.

Ejecutar job	
ID Prueba:	BE86
Objetivo :	Ejecutar un job manualmente que tiene más de un paquete.
Precondición:	El usuario ha ingresado al Sistema. El usuario ha abierto un proyecto. El usuario ha creo un job correctamente. El usuario ha abierto un job “Job Prueba”.
Clases :	JSFServlet, BEVerEjecucionBean, BEControlEjecucion, BERendimientoEjecucion, BEEjecucionJob, BEControlJob, BEJob, BEProgramacion, ProgServlet.
Proceso:	1. Agregue un paquete “PaqA” al job “Job Prueba”. 2. Configure un flujo para el paquete “PaqA” 3. Agregue un segundo paquete “PaqB” al job. 4. Configure un flujo para el segundo paquete. 5. En el menú, seleccione la opción “Ejecutar Job”.

	6. En el siguiente formulario seleccione la opción “Ejecutar”.
Resultado Esperado:	Se muestra en pantalla las tareas que realiza el Job, indicando fecha de inicio, los dos paquetes configurados y los componentes ejecutados.

Ejecutar job	
ID Prueba:	BE87
Objetivo :	Ejecutar un job manualmente que tiene varios orígenes y un solo destino.
Precondición:	El usuario ha ingresado al Sistema. El usuario ha abierto un proyecto. El usuario ha creo un job correctamente. El usuario ha abierto un job.
Clases :	JSFServlet, BEVerEjecucionBean, BEControlEjecucion, BERendimientoEjecucion, BEEjecucionJob, BEControlJob, BEJob, BEProgramacion, ProgServlet.
Proceso:	1. Agregue como origen al flujo un archivo tipo TXT. 2. Agregue un segundo origen al flujo una tabla de base de datos. 3. Agregue como destino al flujo un archivo tipo TXT. 4. Agregue un componente “Proceso” al flujo que una los dos orígenes con el destino. 5. Ingrese alguna operación a realizar en la configuración del objeto activo “Proceso”. 6. En el menú, seleccione la opción “Ejecutar Job”. 7. En el siguiente formulario seleccione la opción “Ejecutar”.
Resultado Esperado:	Se muestra en pantalla las tareas que realiza el Job, indicando fecha de inicio, paquetes y componentes ejecutados.

Ejecutar job	
ID Prueba:	BE88
Objetivo :	Ejecutar un job manualmente que tiene en su proceso de flujo funciones definidas por el usuario.
Precondición:	El usuario ha ingresado al Sistema. El usuario ha abierto un proyecto. El usuario ha creo un job correctamente. El usuario ha abierto un job. El archivo destino tipo TXT ha sido configurado correctamente. El usuario ha creado una función denominada “Calcula_Impuesto()”
Clases :	JSFServlet, BEVerEjecucionBean, BEControlEjecucion, BERendimientoEjecucion, BEEjecucionJob, BEControlJob, BEJob, BEProgramacion, ProgServlet.
Proceso:	1. Agregue como origen al flujo un tabla de base de datos. 2. Agregue como destino al flujo un archivo tipo TXT. 3. Agregue un componente “Proceso” al flujo que una el origen con el destino. 4. Configure la transformación del proceso ingresando una función definida por el usuario, llamada “Calcula_Impuesto()”. 5. En el menú, seleccione la opción “Ejecutar Job”. 6. En el siguiente formulario seleccione la opción “Ejecutar”.
Resultado Esperado:	Se muestra en pantalla las tareas que realiza el Job, indicando fecha de inicio, paquetes y componentes ejecutados.

Ejecutar job	
ID Prueba:	BE89
Objetivo :	Ejecutar un job manualmente que tiene en su proceso del flujo funciones predefinidas.
Precondición:	El usuario ha ingresado al Sistema. El usuario ha abierto un proyecto. El usuario ha creo un job correctamente. El usuario ha abierto un job. El archivo origen XML ha sido configurado correctamente.
Clases :	JSFServlet, BEVerEjecucionBean, BEControlEjecucion, BERendimientoEjecucion, BEEjecucionJob, BEControlJob, BEJob, BEProgramacion, ProgServlet.
Proceso:	1. Agregue como origen al flujo un archivo tipo XML. 2. Agregue como destino una tabla de base de datos. 3. Agregue un componente “Proceso” al flujo que una el origen con el destino. 4. Configure la transformación del proceso ingresando la función predefinida “prom()”. 5. En el menú, seleccione la opción “Ejecutar Job”. 6. En el siguiente formulario seleccione la opción “Ejecutar”.
Resultado Esperado:	Se muestra en pantalla las tareas que realiza el Job, indicando fecha de inicio, paquetes y componentes ejecutados.

3.25 Visualizar log de ejecución

Visualizar Log de ejecución	
ID Prueba:	BE90
Objetivo :	Visualizar el log de una ejecución.
Precondición:	El usuario ha ingresado al Sistema. El usuario ha abierto un proyecto. Un job se ha creado y ejecutado correctamente
Clases :	JSFServlet, BEListarLogsBean, BEVerLogBean, BEControlLogs, BELog, BELineaLogE.
Proceso:	1. En el menú, seleccione la opción “Ver Logs” 2. Seleccione de la lista “Jobs programados” el job “Carga Inicial“. 3. Seleccione de la lista “Logs del job” la primera ejecución.
Resultado Esperado:	Se muestra la página con las tareas que realizó el Job, indicando fecha de inicio, paquetes y componentes ejecutados.

ANEXO I. REPORTE DE PRUEBAS

1. Introducción

1.1 Objetivos

El presente documento ha sido elaborado con la finalidad de presentar los reportes de pruebas que se generaron luego de aplicar los casos de prueba al módulo de extracción de la herramienta AXEbit.

1.2 Alcance

El presente reporte de pruebas abarca todos los casos de pruebas listados en el documento de pruebas de la herramienta AXEbit, para el módulo de extracción.

1.3 Referencias

Las referencias aplicables son:

1. Documento de pruebas – Anexo H

2. Reportes de pruebas

Los escenarios para los casos de prueba realizados, en los cuales se prueba la correcta ejecución de una funcionalidad, se detallan a continuación:

Caso de prueba	BE01- Configurar y conectar fuente de datos
Escenario de prueba	Existe una base de datos SQL Server, cuyos datos de conexión son: Conexion:inti, BaseDatos:Bipucp Usuario:biextrausr Contraseña:2006biusr
Resultado obtenido	El área de trabajo contiene la estructura de la base de datos Bipucp
Modo de verificación	Se observa la estructura en el árbol de navegación de objetos.

Caso de prueba	BE12- Agregar Archivo y configurar Estructura Tipo TXT
Escenario de prueba	Existe un archivo TXT válido en la carpeta del usuario, con los campos siguientes: id: Integer nombre:String apellido:String edad:Integer
Resultado obtenido	La configuración del archivo TXT se configuró correctamente.
Modo de verificación	Se observa en la página de configurar estructura la información guardada.

Caso de prueba	BE14- Agregar Archivo y Configurar Estructura Tipo XLS
Escenario de prueba	Existe un archivo XLS válido en la carpeta del usuario con los campos siguientes: id:Integer producto:String precio:Double
Resultado obtenido	La configuración del archivo XLS se configuró correctamente.
Modo de verificación	Se observa en la página de configurar estructura la información guardada.

Caso de prueba	BE16- Agregar Archivo y Configurar Estructura Tipo XML
Escenario de prueba	Existe un archivo XML válido, y un archivo esquema XSD válido en la carpeta del usuario que indica la estructura siguiente del xml: id: Integer proveedor:string direccion:string
Resultado obtenido	La configuración del archivo XML se configuró correctamente.
Modo de verificación	Se observa en la página de configurar estructura la información guardada.

Caso de prueba	BE18- Crear Job
Escenario de prueba	Se ingresan los siguientes datos para el nuevo job: nombre: “Job Prueba” descripcion:”Job creado para caso de prueba”.
Resultado obtenido	El job se guardó exitosamente.
Modo de verificación	Se observa en el formulario de Administración de jobs el nuevo job creado.

Caso de prueba	BE26- Configurar Parámetro
Escenario de prueba	Se ingresan los siguientes datos para el nuevo parámetro: nombre:”Parametro Prueba” Etiqueta:”Etiqueta prueba” Valor por Defecto:”Valor Defecto Prueba”
Resultado obtenido	El parámetro se agregó exitosamente al job.
Modo de verificación	En el formulario de creación de job aparece el nuevo parámetro.

Caso de prueba	BE33- Crear Paquete
Escenario de prueba	Se ingresan los siguientes datos para el nuevo paquete: nombre:”Paquete Prueba” descripcion:”Paquete creado para caso de prueba” Tema:”Tema12”
Resultado obtenido	El paquete se guardó exitosamente.
Modo de verificación	Se observa en el árbol de navegación de objetos el nuevo paquete creado.

Caso de prueba	BE41- Configurar orden de ejecución de paquetes
Escenario de prueba	Se ingresa el siguiente orden para los paquetes: “Flujo Dimensiones”: 1 “Flujo Facts”: 2
Resultado obtenido	El orden de ejecución se guardó exitosamente.
Modo de verificación	Se observa en la página de Configurar orden la información guardada.

Caso de prueba	BE42- Crear flujo de transformación
Escenario de prueba	Se agrega las siguientes tablas al flujo: “Persona”, “Fact_Persona” Se agrega el componente “Proceso” al flujo. Se agregan los respectivos conectores.
Resultado obtenido	El flujo se guardó exitosamente.
Modo de verificación	Se puede cerrar el paquete y volverlo a abrir para observar el flujo guardado.

Caso de prueba	BE43- Personalizar transformación usando objetos activos – Transformación de Datos.
Escenario de prueba	Se usa las siguientes sentencias de transformación: idPersona= “Persona.idPersona” Apellidos= “Persona.apellidoPaterno”+ ' '+”Persona.apellidoMaterno”
Resultado obtenido	La transformación se guardó exitosamente.
Modo de verificación	Se puede abrir el formulario de Transformación y observar la información guardada.

Caso de prueba	BE01- Configurar y conectar fuente de datos
Escenario de prueba	Existe una base de datos SQL Server, cuyos datos de conexión son: Conexion:inti, BaseDatos:Bipucp Usuario:biextrausr Contraseña:2006biusr
Resultado obtenido	El área de trabajo contiene la estructura de la base de datos Bipucp
Modo de verificación	Se observa la estructura en el árbol de navegación de objetos.

Caso de prueba	BE45- Personalizar transformación usando objetos activos – Filtro de Datos.
Escenario de prueba	Se usa la siguiente sentencia para filtrar: Codigo='0001'
Resultado obtenido	El filtro se guardó exitosamente
Modo de verificación	Se puede abrir el formulario de Filtro y observar la información guardada.

Caso de prueba	BE47- Personalizar transformación usando objetos activos – Estandarización de Datos.
Escenario de prueba	Se usa la siguiente sentencia para estandarizar: “primMayuscula(Nombre)”
Resultado obtenido	La estandarización se guardó exitosamente
Modo de verificación	Se puede abrir el formulario de Estandarización y observar la información guardada.

Caso de prueba	BE49- Personalizar transformación usando objetos activos - Lookup
Escenario de prueba	Se agregan al flujo las tablas “ResumenRelacional” y “Fact_Ventas”. Se agrega el componente proceso y se agregan los conectores. En la configuración del lookup, seleccionar para cada campo del destino, el campo origen apropiado, la tabla relacionada y los campos de referencia y de valor.
Resultado obtenido	El lookup se guardo exitosamente.
Modo de verificación	Se puede abrir el formulario de lookup y observar la información guardada.

Caso de prueba	BE52- Personalizar transformación usando objetos activos - Script
Escenario de prueba	Existe una base de datos Oracle que tiene un procedimiento definido llamado “BiOracle”
Resultado obtenido	El script se guardó exitosamente.
Modo de verificación	Se puede abrir el formulario de Script y observar la información guardada.

Caso de prueba	BE57- Validar Script mediante parser
Escenario de prueba	Se usa la siguiente sentencia para validar: “select * from Persona”
Resultado obtenido	Se validó correctamente la sentencia.
Modo de verificación	Se observa que al guardar el script no aparece ningún mensaje de error.

Caso de prueba	BE59- Crear funciones definidas por el usuario tipo:Regla
Escenario de prueba	Se agregan las siguientes reglas: Entrada: “M”,Salida:”1” Entrada: “F”, Salida:”0” Entrada:”Masc”, Salida:”1”
Resultado obtenido	Se guardó correctamente la función
Modo de verificación	Se puede abrir la función nuevamente y observar la información guardada.

Caso de prueba	BE60- Crear funciones definidas por el usuario tipo: Sentencia
Escenario de prueba	Se ingresan los siguientes datos para la función: Nro de parámetros: 2 Salida: Real Tipos de Entrada: Entero, Real Sentencia: (x1* x1)/x2
Resultado obtenido	Se guardó correctamente la función
Modo de verificación	Se puede abrir la función nuevamente y observar la información guardada.

Caso de prueba	BE65- Programar Jobs
Escenario de prueba	Se ingresan los siguientes datos para la programación: Job:"Job Prueba" Hora:"20:00" Periodicidad:"Diaria" Seleccionar Lunes y Miercoles
Resultado obtenido	Se guardó correctamente la programación
Modo de verificación	Se observa en el formulario de Administración de programaciones la nueva programación.

Caso de prueba	BE77- Ejecutar Job
Escenario de prueba	Se ingresa como parámetro de conexión el valor "inti"
Resultado obtenido	Se ejecutó el job correctamente.
Modo de verificación	Se observa en pantalla las tareas que realiza el job.

Caso de prueba	BE79- Ejecutar Job Origen:TXT Destino: TXT
Escenario de prueba	El archivo TXT origen contiene los siguientes campos: id:Integer, nombre: String apellidoPat:String apellidoMat:String edad:Integer telefono:String El archivo TXT destino contiene los siguientes campos: id: Integer, nombres: String, telefono: String
Resultado obtenido	El proceso de ejecución se realizó correctamente.
Modo de verificación	Se observa en pantalla las tareas que realiza el job. El archivo destino TXT contiene la data correcta.

Caso de prueba	BE80- Ejecutar Job Origen: TXT Destino:XML
Escenario de prueba	El archivo TXT origen contiene los siguientes campos: id:Integer, nombre: String apellidoPat:String apellidoMat:String edad:Integer telefono:String El archivo XML destino posee la siguiente estructura: id: Integer, nombres: String, telefono: String
Resultado obtenido	El proceso de ejecución se realizó correctamente.
Modo de verificación	Se observa en pantalla las tareas que realiza el job. El archivo destino XML contiene la data correcta.

Caso de prueba	BE81- Ejecutar Job Origen: TXT Destino: XLS
Escenario de prueba	El archivo TXT origen contiene los siguientes campos: id:Integer, nombre: String apellidoPat:String apellidoMat:String edad:Integer telefono:String El archivo XLS destino contiene los siguientes campos: id: Integer, nombres: String, telefono: String
Resultado obtenido	El proceso de ejecución se realizó correctamente.
Modo de verificación	Se observa en pantalla las tareas que realiza el job. El archivo destino XLS contiene la data correcta.

Caso de prueba	BE82- Ejecutar Job Origen: XML Destino: TXT
Escenario de prueba	El archivo XML origen posee la siguiente estructura: id: Integer nombre: String apellidoPat:String apellidoMat:String edad:Integer telefono:String El archivo TXT destino contiene los siguientes campos: id: Integer, nombres: String, telefono: String
Resultado obtenido	El proceso de ejecución se realizó correctamente.
Modo de verificación	Se observa en pantalla las tareas que realiza el job. El archivo destino TXT contiene la data correcta.

Caso de prueba	BE83- Ejecutar Job Origen: XLS Destino: TXT
Escenario de prueba	El archivo XLS origen contiene los siguientes campos: id:Integer, nombre: String apellidoPat:String apellidoMat:String edad:Integer telefono:String El archivo TXT destino contiene los siguientes campos: id: Integer, nombres: String, telefono: String
Resultado obtenido	El proceso de ejecución se realizó correctamente.
Modo de verificación	Se observa en pantalla las tareas que realiza el job. El archivo destino TXT contiene la data correcta.

Caso de prueba	BE84- Ejecutar Job Origen: Tabla Destino: XLS
Escenario de prueba	La tabla origen contiene los siguientes campos: id:Integer, nombre: String apellidoPat:String apellidoMat:String edad:Integer telefono:String El archivo XLS destino contiene los siguientes campos: id: Integer, nombres: String, telefono: String
Resultado obtenido	El proceso de ejecución se realizó correctamente.
Modo de verificación	Se observa en pantalla las tareas que realiza el job. El archivo destino XLS contiene la data correcta.

Caso de prueba	BE84- Ejecutar Job Origen: Tabla Destino:Tabla
Escenario de prueba	La tabla origen contiene los siguientes campos: id:Integer, nombre: String apellidoPat:String apellidoMat:String edad:Integer telefono:String La tabla destino contiene los siguientes campos: id: Integer, nombres: String, telefono: String
Resultado obtenido	El proceso de ejecución se realizó correctamente.
Modo de verificación	Se observa en pantalla las tareas que realiza el job. La tabla destino contiene la data correcta.

Caso de prueba	BE87- Ejecutar Job Origen: Tabla Origen: TXT Destino: TXT
Escenario de prueba	La tabla origen contiene los siguientes campos: id:Integer, nombre: String apellidoPat:String apellidoMat:String edad:Integer telefono:String El archivo TXT origen contiene los siguientes campos: id:Integer compras:double ventas:double El archivo TXT destino contiene los siguientes campos: id: Integer, nombres: String, telefono: String compras: Double
Resultado obtenido	El proceso de ejecución se realizó correctamente
Modo de verificación	Se observa en pantalla las tareas que realiza el job. El archivo destino TXT contiene la data correcta.

Caso de prueba	BE88- Ejecutar Job Origen: Tabla Destino: TXT Uso de función
Escenario de prueba	La tabla origen contiene los siguientes campos: id:Integer, nombre: String apellidoPat:String apellidoMat:String edad:Integer telefono:String El archivo TXT destino contiene los siguientes campos: id: Integer, nombres: String, telefono: String. Se usa la función correctamente definida por el usuario: “Calcula_Impuesto()”
Resultado obtenido	El proceso de ejecución se realizó correctamente.
Modo de verificación	Se observa en pantalla las tareas que realiza el job. El archivo destino TXT contiene la data correcta.

ANEXO J. ESTRUCTURA DE PAQUETES

1. Introducción

El presente es el documento que muestra en detalle la estructura de paquetes que se manejó para el uso organizado de las clases. Se agrupó de acuerdo a la funcionalidad que representan para el Sistema.

1.1 Propósito

El propósito principal de este documento es el de proporcionar un mayor entendimiento sobre la estructura que se utilizó para organizar las clases en paquetes.

1.2 Alcance

Este documento consta de un listado detallado de todos los paquetes que se utilizaron además de una pequeña descripción del contenido de cada paquete.

1.3 Visión general del Documento

Este documento consta de dos secciones. Esta sección es la introducción y proporciona una visión general del contenido del documento. En la sección dos se muestra el listado detallado de los paquetes que se usaron.

2. Descripción de Paquetes

Nombre de Paquete	Descripción
comun	Contiene las clases en común para los 3 módulos del proyecto: Análisis, Extracción y Explotación. La estructura de este paquete es la misma usada para cada módulo, la cual se detalla luego.
extraccion	Es el paquete global del módulo de Extracción. En el siguiente nivel de este directorio se encuentra los paquetes principales del módulo.
applet	Contiene las clases que administran el funcionamiento del applet. Se encuentra subdividido en 3 paquetes: arbol, dib, panel.
arbol	Contiene las clases que administran el árbol de navegación de objetos, además de las clases que permiten el drag & drop del árbol al área de trabajo.
dib	Contiene las clases que administran la lógica de los componentes que se dibujan en el área de trabajo.
panel	Contiene las clases que administran el panel principal del applet, controlan diversas funciones como el zoom en el área de trabajo por ejemplo.
beans	Contiene las clases que representan los beans de respaldo de las páginas JSF que se muestran al usuario.
bundle	Contiene los archivos de recursos que administran los mensajes que se muestran al usuario en las páginas JSF.
clases	Contiene las clases que representan el dominio del Sistema, guarda la información de los elementos de extracción configurados por el usuario.
controles	Contiene las clases que administran las diversas funciones que se realizan con las clases de Dominio y la comunicación entre éstas.
excepciones	Contiene las clases que administran las excepciones que se presentan en la ejecución del sistema.
gestores	Contiene las clases que manejan la lógica de los objetos del dominio.
servlet	Contiene el programa que se ejecuta en el servidor para administrar las programaciones que se han hecho en el Sistema.
xml	Agrupa las clases que controlan la persistencia de la metadata. Contiene sólo un paquete: beans.
beans	Contiene las clases que se leen y escriben a disco, para una posterior recuperación.

ANEXO K. ESTRUCTURA DE DESAGREGACIÓN DEL TRABAJO

1. Propósito

1.1. Objetivos

El presente documento tiene como objetivo brindar un panorama general de todo el proyecto, mostrando la manera como se han descompuesto y distribuido las tareas necesarias para completarlo.

Mediante el uso de este documento se espera tener mayor facilidad para el planeamiento, organización y control del proyecto pues permite visualizar directamente que tareas deben realizarse para poder completar cada parte del desarrollo.

1.2. Alcance

El presente documento de estructura de desagregación del trabajo se aplica al módulo de Extracción de AxeBit tomando en consideración todas las etapas de la realización de la herramienta, la misma que es descrita a lo largo de todo el documento de tesis.

2. Estructura de Desagregación del Trabajo

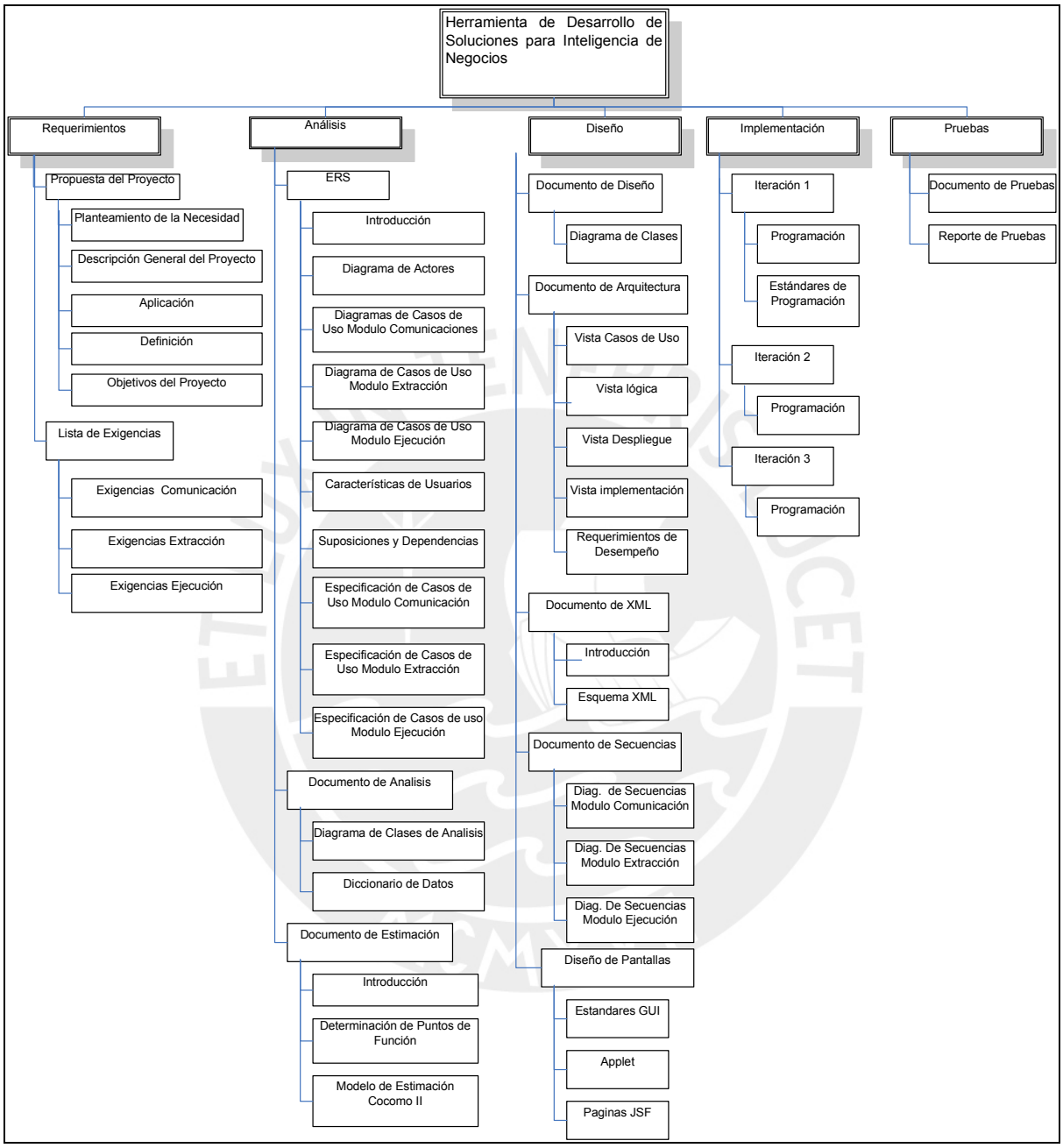


Ilustración 2: Diagrama de Estructura de Desagregación del Trabajo

2.1 Lista de Tareas

1. Requisitos
   1.1. Propuesta del Proyecto
      1.1.1 Planteamiento de la Necesidad
      1.1.2 Descripción General del Proyecto
      1.1.3 Aplicación
      1.1.4 Definición

ANEXO F. DISEÑO DE PANTALLAS

1. Introducción

1.1 Objetivos

El presente documento ha sido elaborado con la finalidad de presentar los diseños de pantalla para la herramienta AXEbit, módulo de extracción.

1.2 Alcance

El presente documento de diseño de pantallas es aplicable al módulo de extracción de la herramienta AXEbit y comprende tanto las pantallas del *applet* como las páginas JSF para la configuración de los diversos componentes de la herramienta.

2. Estándares de programación

A continuación se muestran las pantallas de la herramienta:

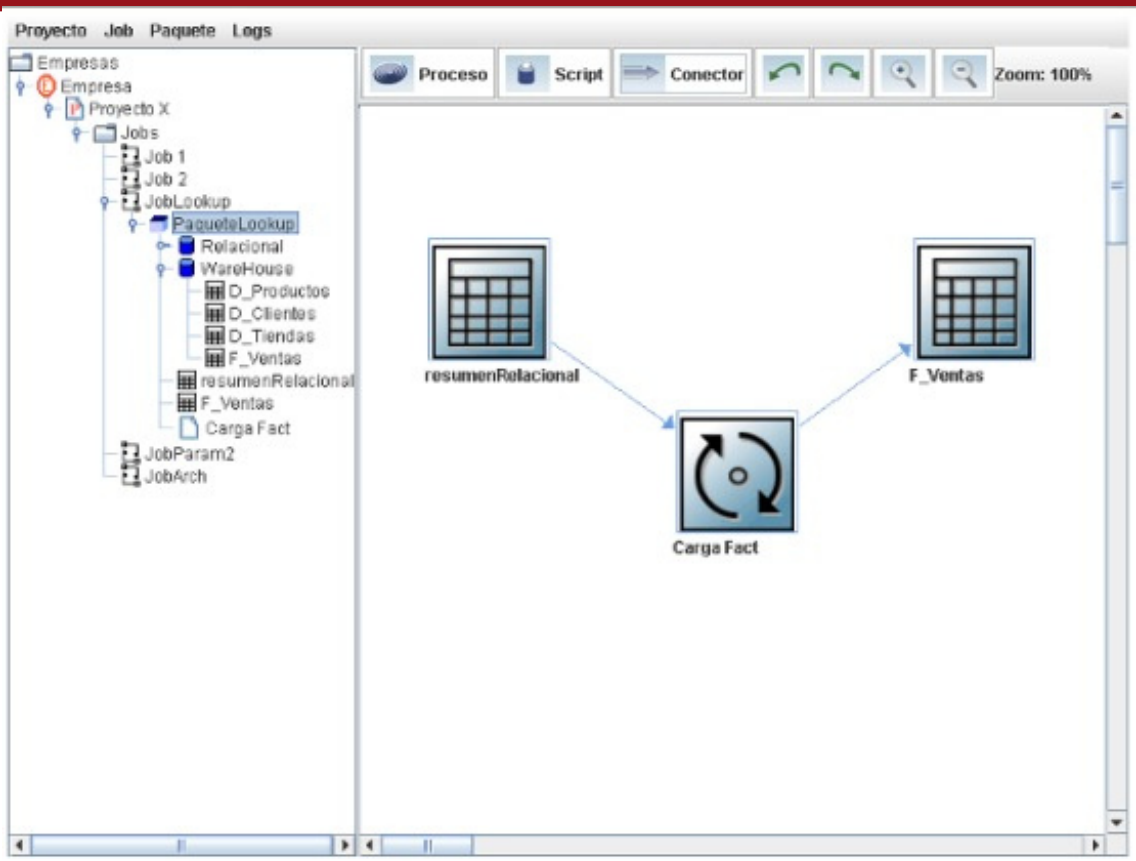
2.1 Applet

El *applet* de la herramienta contiene la interfaz principal con la que interactúa el usuario para el módulo de extracción de la herramienta. Las partes que componen el *applet* son:

a) Árbol de proyecto: Ubicado al lado izquierdo del *applet*, permite navegar entre los objetos del proyecto de forma jerárquica, primero la raíz donde se ubican las empresas con las que trabaja el usuario, luego los proyectos que pertenecen a la empresa, en el siguiente nivel los *jobs* y luego los paquetes. Cada elemento tiene un menú contextual que permite configurarlo.

b) Área de dibujo: Corresponde al área más grande del *applet* y es donde el usuario puede formar el flujo de transformación para un paquete. Cuenta con una barra de herramientas en la cual se pueden seleccionar los componentes a agregar al flujo.

c) Menú principal: Ubicado en la parte superior del *applet* permite acceder rápidamente a todas las funcionalidades de la herramienta que estén activas para el elemento actual.



Applet del módulo de extracción

2.2 Páginas JSF

Para configurar un componente del flujo de transformación se llama a una ventana popup con una página *JSF* apropiada. Del mismo modo se hace para cada funcionalidad del menú, administración de *jobs*, programaciones, muestra de *logs* de ejecución, etc.

Las características principales de las páginas *JSF* que se han implementado para la herramienta son las siguientes:

a) Se utiliza las hojas de estilo o *cascading style sheets* (CSS) para unificar la definición de estilos de todas las páginas y facilitar de esta forma el mantenimiento de las páginas. Un cambio en la hoja de estilos se verá reflejado en todas las páginas inmediatamente.

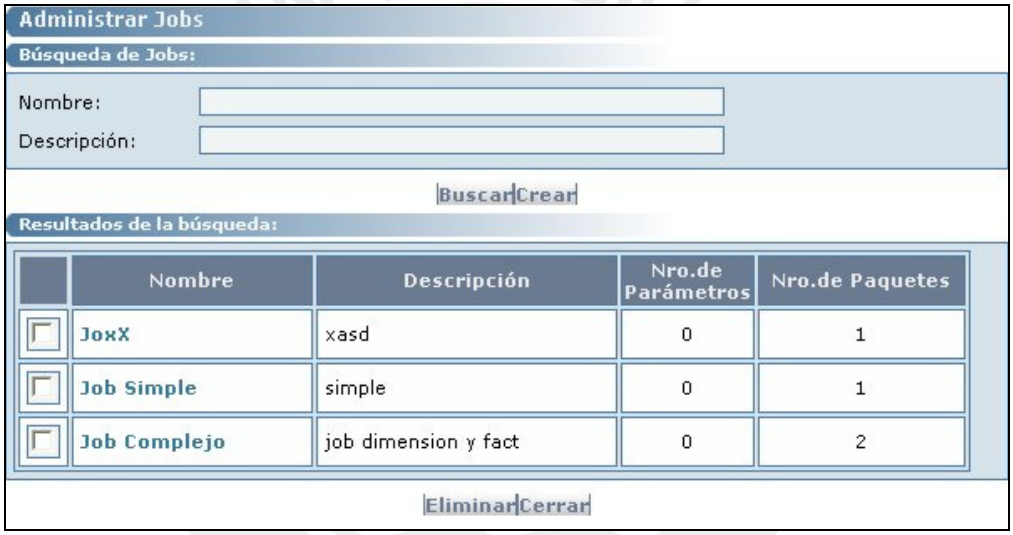
b) Se ha seleccionado un color tenue para evitar el agotamiento visual luego de usar la herramienta por tiempo prolongado.

Se hace uso de archivos properties o bundle para obtener el texto que se muestra en las páginas, de esta forma se puede establecer un archivo properties por idioma y permitir que se seleccione dinámicamente el archivo de idioma apropiado según la configuración del explorador web que accede a la herramienta. Esto permite que la herramienta soporte múltiples idiomas con un mínimo esfuerzo, pues no hay que modificar las páginas JSF, sino solamente definir nuevos archivos properties para los idiomas adicionales.

A continuación se muestra la lista de páginas que forman la aplicación organizadas por funcionalidad y acompañadas de una breve descripción.

2.2.1 Administrar jobs

Esta pantalla permite administrar los jobs del proyecto actual. Se pueden crear, modificar o eliminar los jobs.



Página de administración de jobs

2.2.2 Nuevo job

Esta pantalla permite crear y modificar los datos de un job.

Página de creación y modificación de jobs

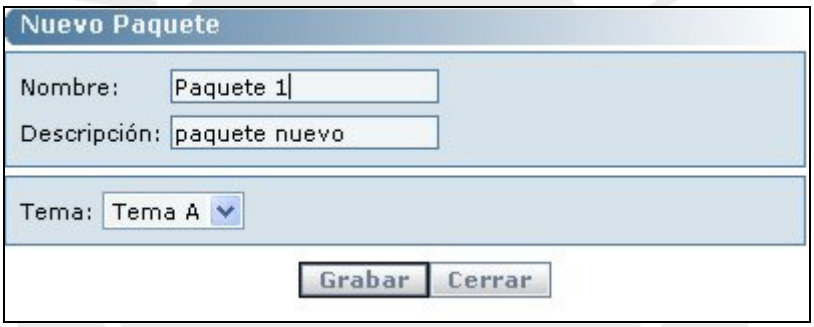
2.2.3 Nuevo parámetro

Esta pantalla permite crear o modificar los datos de un parámetro para un job.

Página de creación y modificación de parámetros

2.2.4 Nuevo paquete

Esta pantalla permite crear un nuevo paquete para un job.



Página de creación de paquete

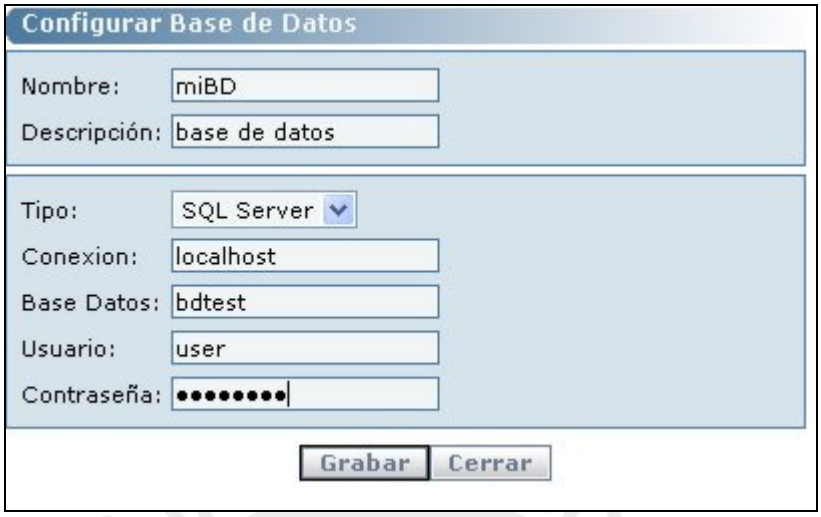
2.2.5 Configurar orden de paquetes

Esta pantalla permite configurar el orden en que se ejecutarán los paquetes que pertenecen a un job.

Página de configuración del orden de paquetes

2.2.6 Configurar fuente de datos base de datos

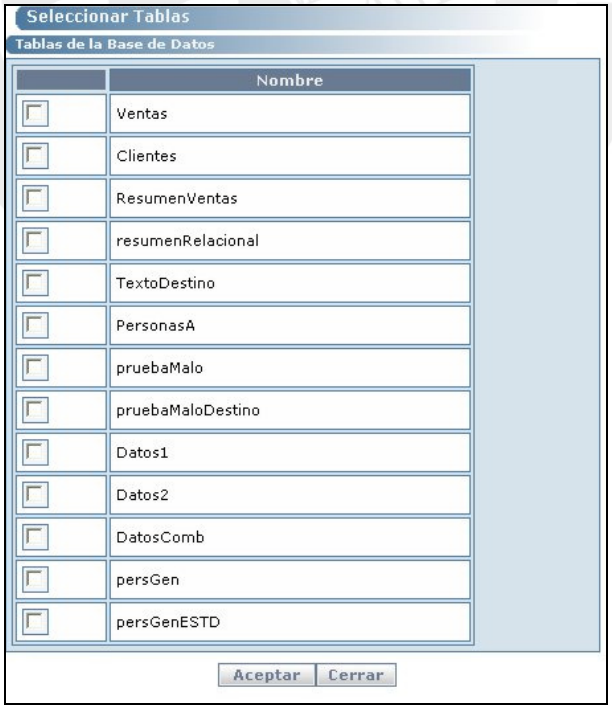
Esta pantalla permite configurar los datos de conexión a una base de datos para ser utilizada como fuente de datos del paquete.



Página de configuración de base de datos

2.2.7 Seleccionar tablas de base de datos

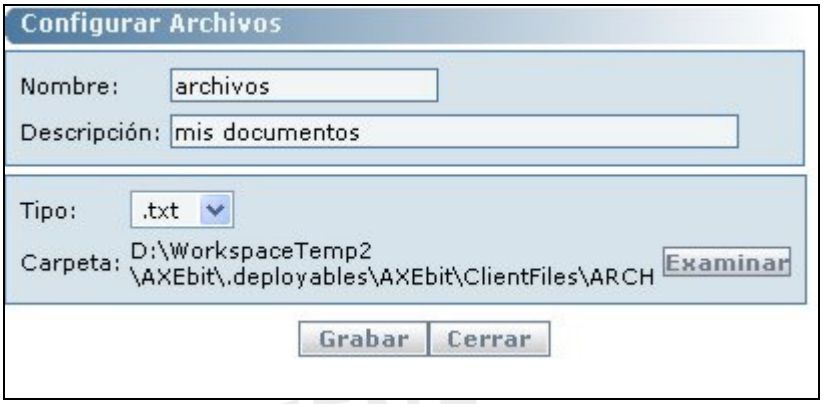
Esta pantalla permite seleccionar las tablas de la base de datos que serán agregadas al paquete para poder formar parte del flujo de transformación.



Página de selección de tablas de la base de datos

2.2.8 Configurar fuente de datos archivo

Esta pantalla permite seleccionar la carpeta desde la cual se leerán los archivos planos para ser agregados al paquete.

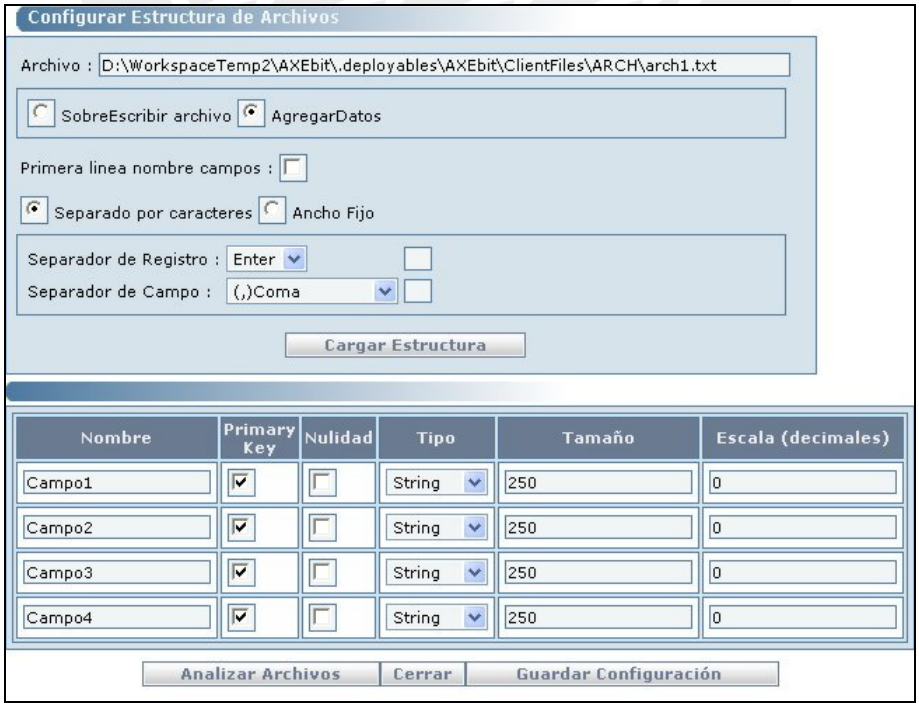


Página de configuración de ruta de archivos planos

2.2.9 Configurar estructura de archivo plano

Esta pantalla permite indicar la estructura de un archivo plano agregado al paquete. Se debe especificar la separación entre campos y registros del archivo, o si es que tiene ancho fijo para sus campos. También se debe indicar si al momento de escribir en el archivo se reemplazarán todos los datos o se agregará al final.

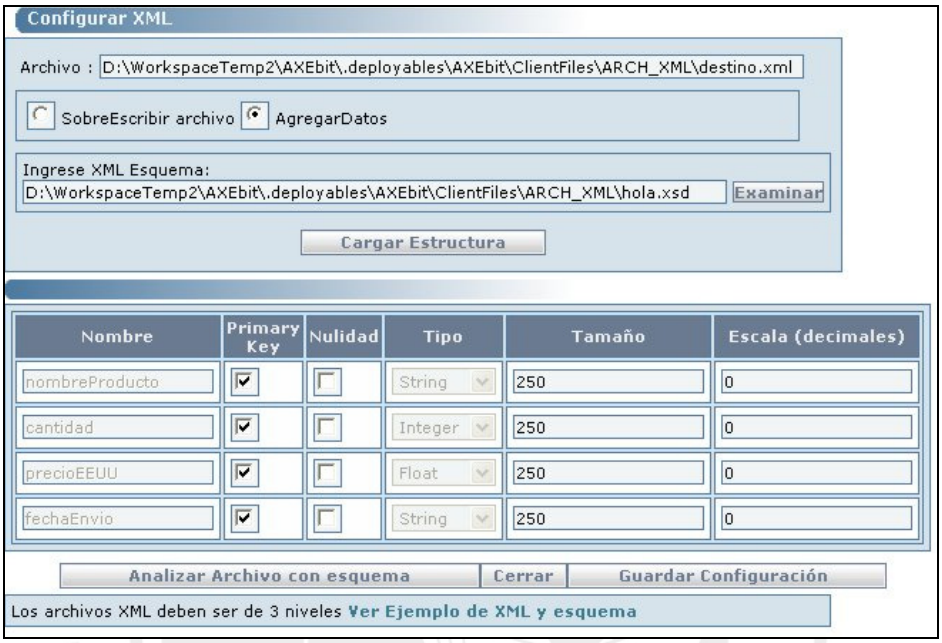
Finalmente, cuando se tienen los campos, se debe indicar el nombre y tipo de dato de cada uno.



Página de configuración de estructura de archivo plano

2.2.10 Configurar estructura de archivo xml

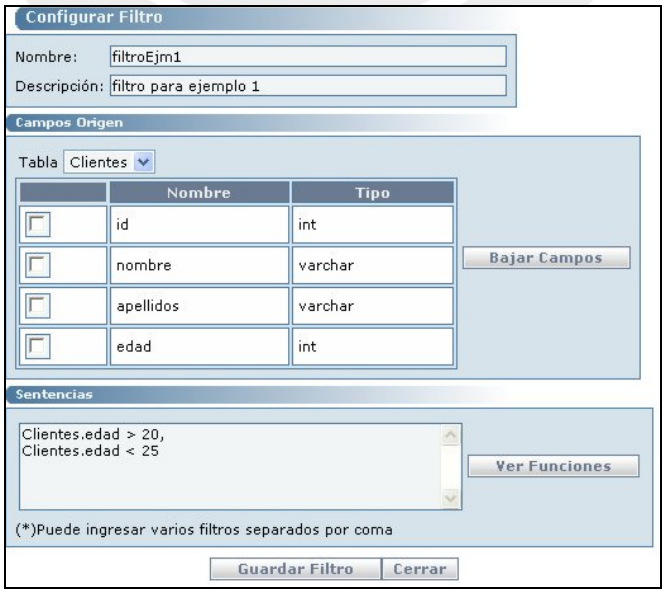
Esta pantalla permite especificar la ruta desde la cual se cargará el archivo XSD (esquema XML) para obtener la información de los campos que conforman un archivo XML agregado al paquete.



Página de configuración de estructura de archivo XML

2.2.11 Configurar filtro

Esta pantalla permite configurar el subcomponente 'Filtro' del flujo de transformación.



Página de configuración de filtro

2.2.12 Configurar transformación

Esta pantalla permite configurar el subcomponente 'Transformación' del flujo de transformación.

Página de configuración de transformación

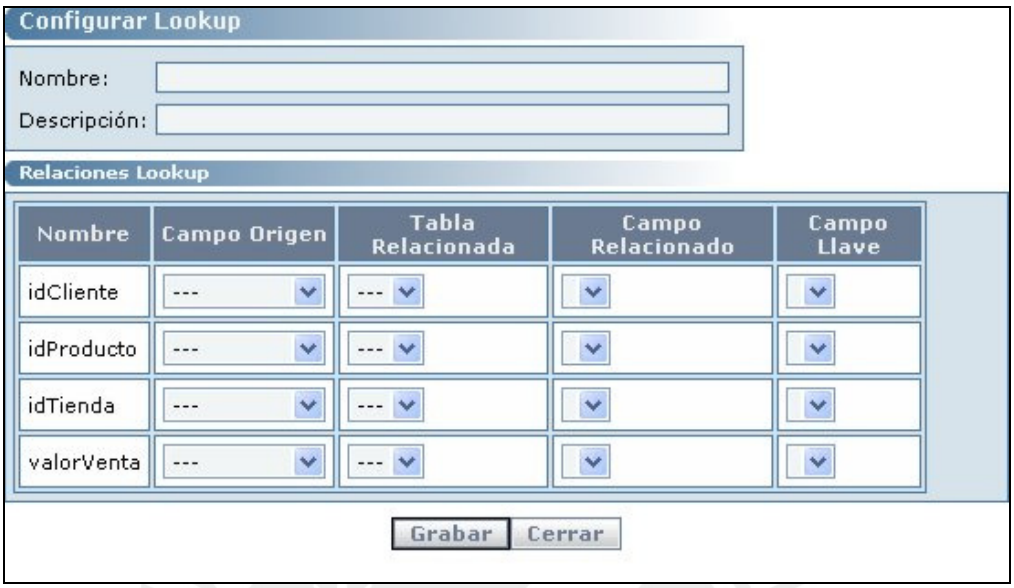
2.2.13 Configurar estandarización

Esta pantalla permite configurar el subcomponente 'Estandarización' del flujo de transformación.

Página de creación y modificación de parámetros

2.2.14 Configurar lookup

Esta pantalla permite configurar el subcomponente 'Lookup' del flujo de transformación.



Página de configuración de lookup

2.2.15 Ver funciones

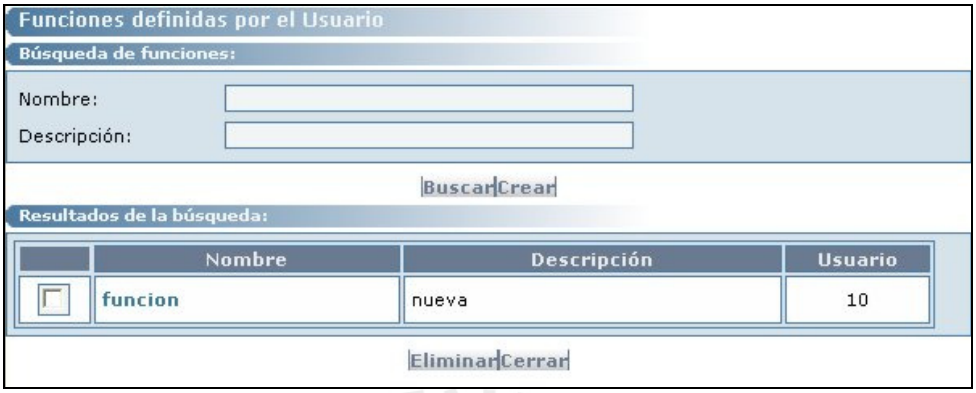
Esta pantalla permite visualizar las funciones que aplican al subcomponente que se está configurando.

Nombre	Descripción	Aplicable a
+	Operador de Concatenación de Cadenas	Tipos Cadena
+	Operador de suma de valores	Tipos Numéricos
-	Operador de resta de valores	Tipos Numéricos
*	Operador de multiplicación de valores	Tipos Numéricos
/	Operador división de valores	Tipos Numéricos
^	Operador de potencia	Tipos Numéricos
sum(x)	Función de suma acumulativa de un campo	Tipos Numéricos
prom(x)	Función de promedio de valores de un campo	Tipos Numéricos
directo(x)	Función de paso directo de datos. Bloquea la estandarización.	Cualquier tipo
substring (x,y,z)	Función de recorte de cadenas	Tipos Cadena

Página de listado de funciones

2.2.16 Administrar funciones definidas por el usuario

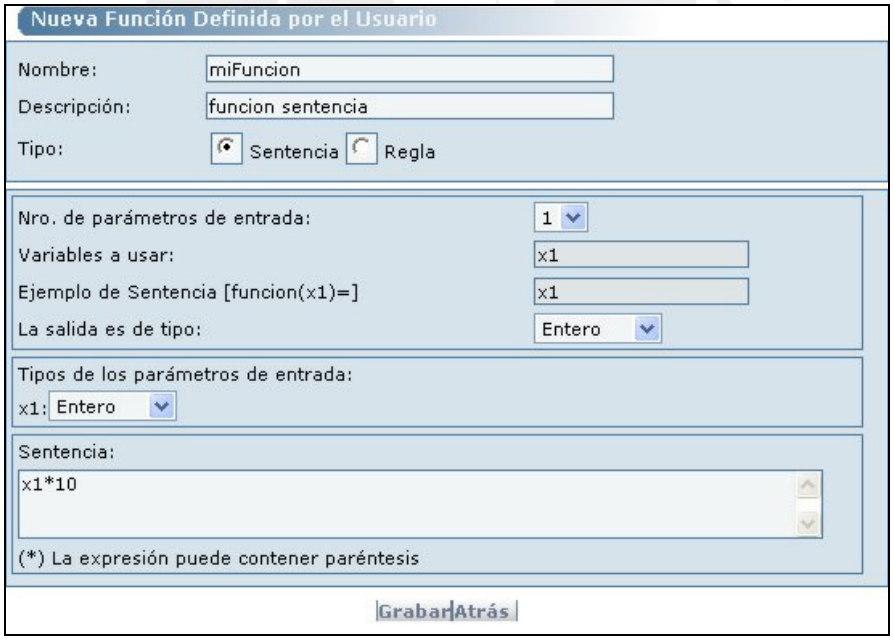
Esta pantalla permite administrar las funciones definidas por el usuario. Permite crear, modificar y eliminar funciones.



Página de administración funciones definidas por el usuario

2.2.17 Nueva función definida por el usuario tipo sentencia

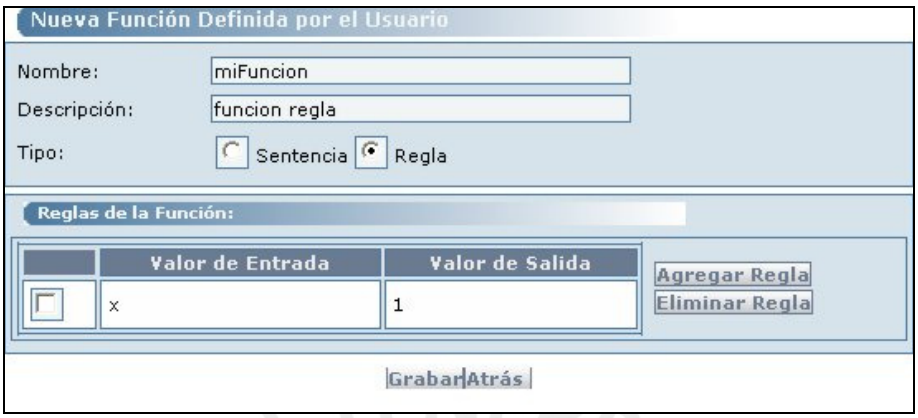
Esta pantalla permite crear nuevas funciones definidas por el usuario de tipo sentencia.



Página de creación de función definida por el usuario tipo sentencia

2.2.18 Nueva funcion definida por el usuario tipo reglas

Esta pantalla permite crear nuevas funciones definidas por el usuario de tipo reglas.



Página de creación de función definida por el usuario tipo reglas

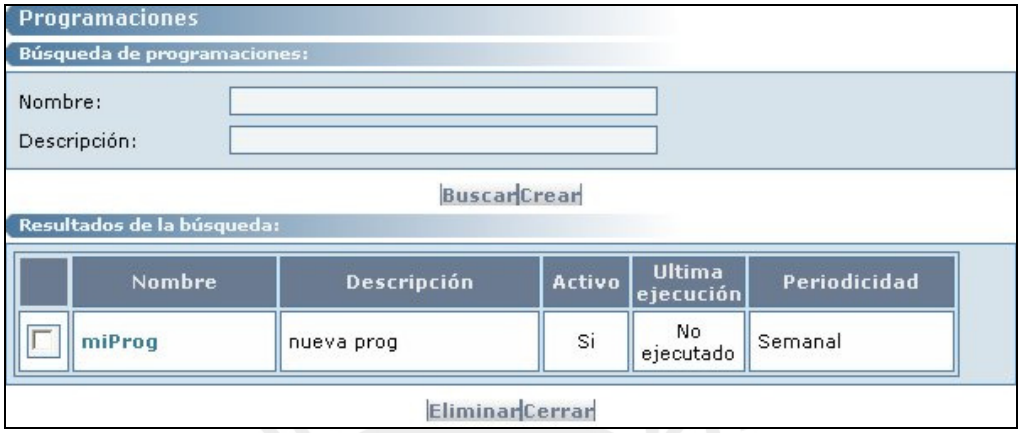
2.2.19 Formar sentencia where

Esta pantalla permite formar fácilmente la sentencia where para una transformación que involucre más de una tabla origen.

Página de formación de sentencia where

2.2.20 Administrar programaciones

Esta pantalla permite administrar las programaciones del proyecto actual. Permite agregar, modificar y eliminar programaciones.



Página de administración de programaciones

2.2.21 Programar job

Esta pantalla permite configurar una nueva programación para ejecutar automáticamente un job.

Página de programación de job

2.2.22 Ejecutar job

Esta pantalla permite ejecutar manualmente un job.

Página de ejecución manual de job

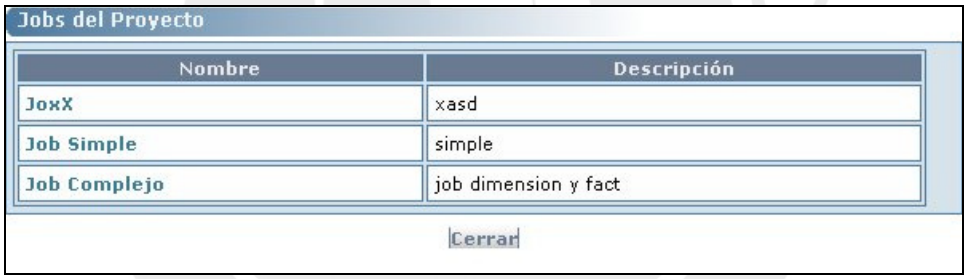
2.2.23 Ver ejecución de job

Esta pantalla permite visualizar la ejecución del job en tiempo real.

Página de vista de ejecución de job

2.2.24 Listar jobs

Esta pantalla muestra la lista de jobs del proyecto, desde aqué se selecciona el job para el cual se desea ver los logs de ejecución.



Página de listado de jobs para un proyecto

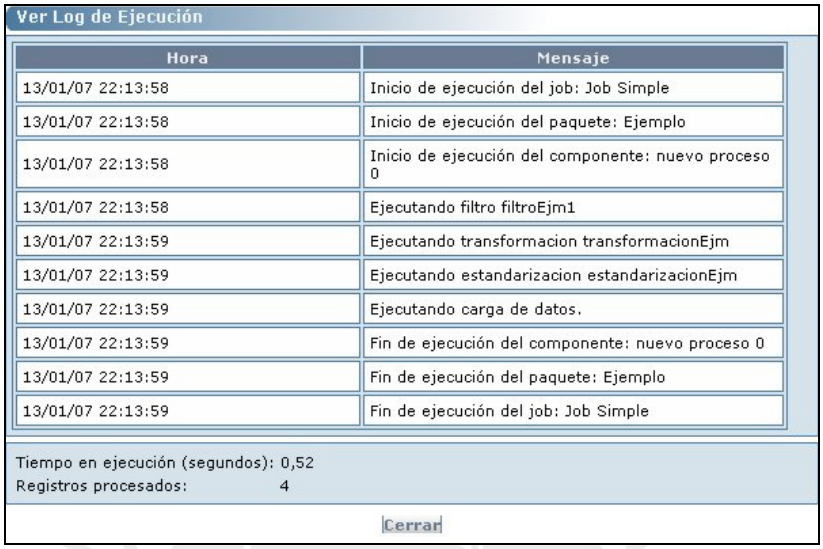
2.2.25 Listar logs por job

Esta pantalla muestra la lista de logs de ejecución para un job específico.

Página de listado de logs para un job

2.2.26 Ver log

Esta pantalla muestra el contenido de un log de ejecución.



Página de vista de log de ejecución