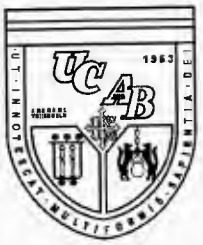

UNIVERSIDAD CATÓLICA ANDRÉS BELLO

FACULTAD DE INGENIERÍA

ESCUELA DE INGENIERÍA INFORMÁTICA

Transporte de Configuración de Aplicaciones Desarrolladas con el Framework TRIAD

TRABAJO INSTRUMENTAL DE GRADO

Presentado ante la

UNIVERSIDAD CATÓLICA ANDRÉS BELLO

Como parte de los requisitos para optar al título de

INGENIERO EN INFORMÁTICA

REALIZADO POR **Giancarlo Sánchez**

TUTOR EMPRESARIAL **Carlos Zambrano**

TUTOR ACADÉMICO **Omar Hernández**

FECHA **Septiembre, 2011**

UNIVERSIDAD CATÓLICA ANDRÉS BELLO

FACULTAD DE INGENIERÍA

ESCUELA DE INGENIERÍA INFORMÁTICA

Transporte de Configuración de Aplicaciones Desarrolladas con el Framework TRIAD

Este Jurado; una vez realizado el examen del presente trabajo ha evaluado su contenido con el resultado: DIECINUEVE (19)

JURADO EXAMINADOR

Firma:		Firma:		Firma:	
Nombre:	CARLOS BARDOLETA	Nombre:	OMAR HERNANDEZ	Nombre:	CARLOS ALFONZO

REALIZADO POR	**Giancarlo Sánchez**
TUTOR EMPRESARIAL	**Carlos Zambrano**
TUTOR ACADÉMICO	**Omar Hernández**
FECHA	**Septiembre, 2011**

DEDICATORIA

Dedico este Trabajo Instrumental de Grado a mis padres. Su apoyo incondicional y siempre presente han hecho posible alcanzar esta meta personal tan importante. Espero siempre estar a la altura de sus expectativas y enorgullecerlos con mis logros.

A mi Nonna, quien ha sido mi segunda madre, por su fuerza, valor, perseverancia y espíritu de superación. Ha sido un ejemplo de vida que me ha motivado a siempre seguir adelante con optimismo.

A mi hermana, quien me demostró reiteradamente que el éxito y la satisfacción personal son directamente proporcionales al esfuerzo y el trabajo y que todo es posible en la vida.

A toda mi familia, quienes me ayudan a cosechar mis logros con su presencia y sus palabras.

AGRADECIMIENTOS

Gracias a toda mi familia por siempre estar presentes. A mis padres por su apoyo infalible, a mi Nonna por enseñarme el valor de la perseverancia, a mi hermana por haber estado a mi lado en todo momento, a mi tío Tony y mi tía Velma por haber hecho de su hogar el mío, por sus consejos siempre acertados y por ser un ejemplo de vida.

A mis profesores, tutores, preparadores y demás instructores que no solamente me transmitieron los conocimientos, sino también forjaron y estructuraron mi mente y mi forma de pensar y resolver las situaciones.

Gracias a todos mis amigos y hermanos de la Universidad, en especial a aquellos de Módulo 1, ya saben quiénes son. Cada día fue una experiencia inolvidable, gracias por su amistad, su presencia y su apoyo incondicional. Por estar siempre pendiente de mí progreso y estar dispuestos a ayudar.

Alejandro Blanco, Daniel Terraza, Joel Ojeda y el grupo de trabajo de Cinet Solutions, gracias por el apoyo y la experiencia que me transmitieron. Más allá de un grupo de trabajo, buenos amigos los describiría mejor. Lograron hacer el trabajo una experiencia inolvidable, por la cual siempre estaré en deuda con ustedes.

Gracias a Padme, mi fiel e incondicional amiga. Siempre aportas el toque de alegría que hace de cada día, uno especial.

Gracias a todos los que me ofrecieron apoyo incondicional y compartieron mis experiencias.

TABLA DE CONTENIDO

INDICE DE FIGURAS

INDICE DE TABLAS

SINOPSIS

El presente proyecto se clasifica dentro del marco de transporte de aplicaciones y nuevas implementaciones y pretende, agregar nuevas funcionalidades al framework TRIAD, pilar fundamental del desarrollo de las aplicaciones de la empresa. Estas funcionalidades otorgan al framework TRIAD la capacidad para transportar nuevas implementaciones de aplicaciones. Entre las ventajas operativas que estas nuevas funcionalidades aportan podemos mencionar:

- Capacidad para transportar nuevas implementaciones a los diferentes clientes, con un mínimo impacto sobre la data y configuraciones que estos posean.
- Facilitar la implementación de las nuevas funcionalidades desarrolladas sobre un ambiente de prueba o calidad. Optimizando así el tiempo utilizado durante la fase de prueba
- Identificación gráfica de la configuración de las vistas de la aplicación y todos sus componentes necesarios para su operatividad
- Versionamiento de pantallas y componentes de aplicaciones.

La realización de este proyecto resultó en una aplicación de transporte, complemento del framework TRIAD que permite llevar un control de versiones sobre las nuevas implementaciones y permite transportarlas de manera fácil y rápida, manteniendo la integridad de la data entre los ambientes y proporcionando mecanismos de monitoreo y respaldos en el proceso.

INTRODUCCIÓN

Cinet Solutions C.A. es una empresa dedicada al desarrollo de Sistemas de Información. Todas las aplicaciones desarrolladas se basan en el uso de un framework llamado TRIAD, que permite ensamblar las áreas principales en un sistema empresarial, que son la presentación, los servicios y el negocio.

Actualmente, implementar nuevas funcionalidades de estas aplicaciones en clientes implica una tarea meticulosa y propensa a fallos. Varios técnicos de diferentes áreas deben estar presentes a la hora de generar manualmente los archivos para el transporte de esas nuevas funcionalidades y estar presentes durante la instalación en los clientes, generar un respaldo manualmente y esperar que todo salga de acuerdo a lo planeado y no haya faltado ningún componente por transportar, ni la data del destino haya sido alterada.

Adicionalmente hay que tomar en cuenta la configuración y data particular de cada cliente al cual deben implementarse esas nuevas funcionalidades, para evitar errores que pueden ir desde colores equivocados en las pantallas, hasta logos o precios de otro cliente diferente.

Debido a lo anterior, surge la necesidad de automatizar este proceso de transporte, la identificación de los elementos que deben ser transportados para el correcto funcionamiento de la aplicación, los ambientes operativos y de clientes y la generación automática de respaldos.

La meta de este Trabajo Instrumental de Grado es proporcionar mecanismos amigables para los desarrolladores que faciliten el transporte de configuraciones de aplicación entre ambientes operativos y clientes.

Con la realización de este Trabajo Instrumental de Grado (TIG), se tendrá disponible una aplicación de transporte, parte del framework TRIAD, a través de la cual podrán versionarse nuevas funcionalidades e implementaciones que hayan sido generadas como parte del desarrollo de una aplicación. Se estará en capacidad para generar archivos transportables con esa data y posteriormente podrá ser actualizada, transportando así, esas nuevas funcionalidades a un ambiente destino.

En el presente escrito, se describe con detalle el proceso de elaboración del Trabajo Instrumental de Grado (TIG), desglosado en los siguientes capítulos:

- **Capítulo I,** describe la necesidad de la empresa para la cual el TIG ofrece solución. Adicionalmente presenta el objetivo general, objetivos específicos, alcance y justificación para el desarrollo del proyecto.
- **Capítulo II,** se puntualizan los conceptos básicos para familiarizar al lector con los términos relacionados con el proyecto.
- **Capítulo III,** se explica la metodología utilizada en el desarrollo del TIG.
- **Capítulo IV,** describe individualmente las actividades realizadas, definidas por el esquema metodológico utilizado.
- **Capítulo V,** explica los resultados obtenidos.
- **Capítulo VI,** contiene las conclusiones alcanzadas y las recomendaciones para futuros desarrollos.

CAPÍTULO I. Problema a desarrollar

I.1 Necesidades de la empresa

Cinet Solutions C.A. es una empresa venezolana dedicada al desarrollo de soluciones tecnológicas en el área de Sistemas de Información orientadas a la gestión pública, como por ejemplo, el sector salud.

Las aplicaciones desarrolladas por la empresa están basadas en la utilización de un framework llamado TRIAD, que permite ensamblar las tres áreas principales en un sistema empresarial, que son: la presentación (Vistas), la lógica de Negocios (Servicios) y la Información (Datos). Es sobre este framework que las aplicaciones son desarrolladas.

Adicionalmente, estas aplicaciones, son utilizadas por diferentes clientes, cada uno de ellos, en capacidad de definir ciertas configuraciones y personalizaciones adaptadas a sus necesidades particulares.

Por ejemplo, para la aplicación "GCP" (Gestión Clínica de Pacientes) desarrollada por la empresa, diferentes centros de salud como lo son Clínica Ávila y el Centro Médico Docente la Trinidad la han personalizado a su medida ajustando, entre otras cosas, las características visuales de las pantallas, logos, roles del personal de la institución y llamadas a servicios. Si bien las funcionalidades de la aplicación son las mismas en ambos casos, cada cliente define y personaliza su versión de la aplicación a través de los elementos de configuración parametrizables.

Estos elementos de configuración parametrizables o simplemente, Configuraciones, son datos, propiedades y características personalizables de la aplicación que permite a múltiples clientes adaptarla a su medida.

La metodología de desarrollo de la empresa trabaja sobre 3 ambientes durante el ciclo de vida de las aplicaciones: ambiente de desarrollo, de prueba o calidad y de producción.

Si un cliente solicita una modificación con respecto a algún elemento de la configuración de su aplicación, la empresa debe ajustar los cambios solicitados en ambiente de desarrollo, transportarlos (Copiar las aplicaciones y sus configuraciones para el cliente en cuestión) a un ambiente de calidad para pruebas y luego transportar esos cambios al ambiente final de producción para el cliente que los solicitó. Esto también aplica para la incorporación de nuevas funcionalidades.

Sin embargo las técnicas de transporte actual de las configuraciones resultan ineficientes porque no existe una forma automatizada de manejar el transporte de esos elementos entre ambientes y clientes. Tampoco existe un mecanismo de control de versiones que permita a la empresa llevar un registro de las versiones de configuración individuales por cada cliente, en cada uno de los ambientes.

Es por esta razón que la empresa se ha visto en la necesidad de implementar una solución que facilite el transporte de configuraciones de una aplicación entre diferentes ambientes operativos y de clientes.

Para poder lograr este objetivo, la empresa ha identificado cuatro grandes pilares que la solución a implementar debe considerar incluir al momento de realizar el transporte para garantizar la integridad funcional de la aplicación:

1) Elementos Descriptores, Clasificaciones y Parámetros.
2) Elementos visuales o Vistas.
3) Elementos de procesos -. **FTWS**.
4) Elementos de Datos o Entidades.

Estos elementos indicados anteriormente y sus dependencias conforman la configuración de la aplicación, que cada cliente puede personalizar.

La solución de transporte debe tomar en cuenta, adicionalmente, ciertas características y regulaciones:

- ✓ Migración ligera de los entornos. En donde la data a transferir sea lo menos pesada y más simple posible. (Buscando reusabilidad de los métodos y componentes ya existentes).
- ✓ Migración sobre diferentes ambientes de trabajo. La solución debe permitir la portabilidad entre múltiples manejadores de base de datos.
- ✓ Validación de las configuraciones transportadas de un ambiente a otro. Es necesario que el procedimiento de migración se apegue a la lógica de negocio de TRIAD a fin de mantener y validar la coherencia de la configuración.

La implementación de la solución permitirá a la empresa estar en capacidad para:

- ✓ Proveer un respaldo de los parámetros de configuración de las aplicaciones a los clientes.
- ✓ Migrar entornos de configuración rápidamente desde un ambiente de desarrollo al ambiente de calidad y luego de producción respetando los datos y configuraciones creadas en cada ambiente e instalación de cliente. Estos 4 entornos de configuración fueron definidos como módulo de Elementos Descriptores, Clasificaciones y Parámetros; Elementos visuales o vistas; Elementos de procesos - **FTWS** y Elementos de Datos o entidades.
- ✓ Reducir la probabilidad de errores durante configuración en los ambientes de Calidad o prueba y de Producción.
- ✓ Agilizar los procesos de implantación y respaldo de las aplicaciones.

I.2 Solución propuesta

La solución propuesta consiste en el desarrollo de una aplicación de transporte para el framework TRIAD que implemente todas las funcionalidades necesarias para la creación, configuración, monitoreo y control de transporte de configuraciones entre diferentes ambientes operativos y de clientes.

Esta aplicación permitirá a la empresa:

Creación de transporte: la creación del transporte de configuraciones permite identificar y seleccionar qué elementos de configuración van a transportarse. De igual manera identifica todas las dependencias necesarias a transportar por cada elemento y los incluye en el transporte. Dentro de estos elementos encontramos:

- ✓ Data de configuración, clasificación, descriptores y parámetros de la aplicación.
- ✓ Vistas
- ✓ Servicios
- ✓ Engines
- ✓ Archivos adicionales (logos, etc.)

Todos estos elementos deben versionarse individualmente para identificar las modificaciones realizadas y al cliente al que pertenecen. La naturaleza modular del desarrollo obliga a llevar un control de versiones individual sobre componentes y dependencias a fin de validar la coherencia y compatibilidad.

El control de versiones debe, adicionalmente, proveer mecanismos para identificar sobre qué ambiente se está trabajando (desarrollo, calidad o prueba, producción) y coordinar la compatibilidad entre las distintas versiones de elementos y sus dependencias, por cada cliente.

Configuración de transporte: al momento de crear, actualizar o eliminar una estructura de datos, es necesario seguir una lógica secuencial que está basada en la navegabilidad y dependencias que pueden existir entre los elementos de esa estructura.

Una vez conocidos todos los elementos a transportar, la configuración de transporte define la lógica secuencial con la que los elementos, componentes y dependencias deben ser transportados y restaurados, respetando la integridad lógica de los procesos de creación, actualización y borrado de esas estructuras.

Monitoreo de transporte: el proceso de transporte se realiza en dos etapas:

- Proceso de transporte: en un ambiente origen, se crea la estructura de archivos binarios transportable que incluye las estructuras de datos con los elementos seleccionados y sus dependencias, las versiones de cada componente y las instrucciones para la restauración. Adicionalmente se provee información al usuario sobre las características que debe poseer el ambiente destino correspondiente para que la restauración pueda llevarse a cabo.

- Proceso de restauración: en un ambiente destino, se cargan los archivos binarios transportables y siguiendo las instrucciones de restauración y validando las versiones de cada componente y sus dependencias se restauran siguiendo la lógica secuencial definida en el proceso de configuración de transporte.

  Éste y los demás procesos involucrados en el transporte son monitoreados en tiempo real y almacenados en un log. De esta forma puede llevarse un control sobre los procesos y sus estados e identificar fallas rápidamente.

Control de transporte: los procesos deben ser controlados. Es necesario un ente que coordine las acciones durante los procesos de creación, configuración, transporte y restauración, al igual que la integridad del transporte en general.

Adicionalmente, durante el proceso de restauración, pueden surgir eventualidades que obliguen al usuario a revertir los cambios realizados (restauración), con lo cual, el sistema debería estar en capacidad de eliminar los cambios y restaurar los elementos que tenía originalmente (a través de un respaldo).

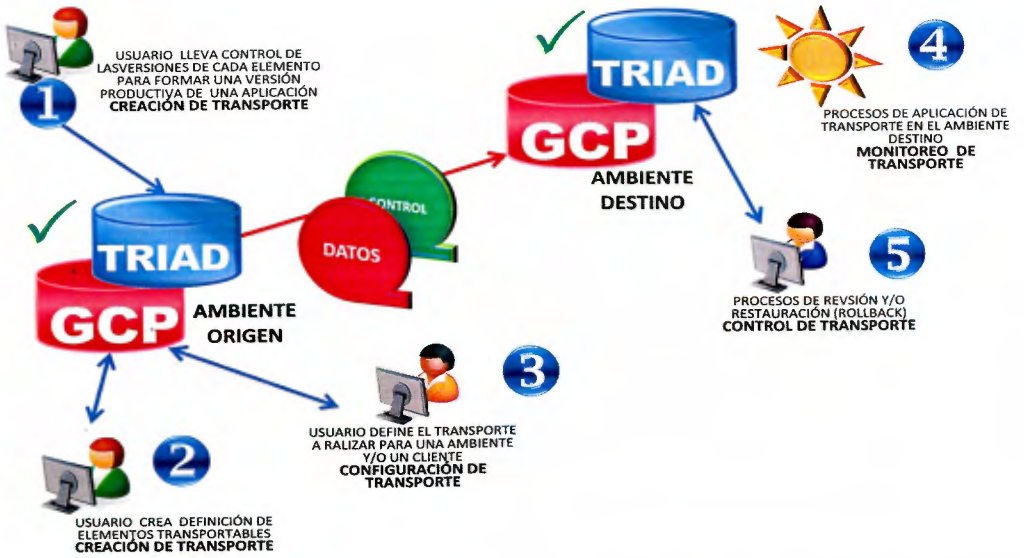

Figura 1. Diagrama general de la solución de transporte planteada

Fuente: Elaboración propia

Con base en los puntos anteriormente explicados, la solución propuesta divide las funcionalidades en varios módulos:

1. Módulo de control de versiones: Administra la versión de los componentes y elementos de configuración a transportar y sus dependencias, validando así, que lo que se esté transportando, no solo esté acorde con la última versión del componente o elemento, sino que también pertenezca al cliente en particular.

   Cada cliente maneja una versión individual de cada elemento o componente de configuración y sus respectivas dependencias, definidas por actualizaciones y/o mejoras solicitadas o necesitadas por el cliente particular, que solo se plasman en su configuración personalizada.

El control de versiones también estará en capacidad para identificar y facilitar el transporte entre los diferentes ambientes (desarrollo, calidad o prueba, producción) identificando cada cliente y sus respectivas necesidades.

Este módulo proveerá al usuario de una vista a través de la cual podrá identificar el estatus y versión de las aplicaciones por cada cliente y la versión y revisión correspondiente por cada elemento de configuración. Adicionalmente, esta vista implementará herramientas gráficas que permitirán al usuario identificar, por cada cliente, el historial de transportes y versiones de configuración en cada ambiente.

2. Módulo de creación de transporte: es necesario en primera instancia, identificar por cada elemento, que data debe ser incluida dentro del proceso de transporte. Este módulo implementa las funcionalidades de identificación y selección de los elementos válidos de transporte. Por medio de una vista se permitirá al desarrollador seleccionar de un listado de elementos y sus componentes aquellos que serán transportados al ambiente destino.

   El módulo estará en capacidad de identificar todas las dependencias necesarias que deben transportarse adicionalmente a los elementos seleccionados directamente por el usuario.

3. Módulo de configuración de transporte: si bien las aplicaciones están destinadas a crecer tanto en tamaño como en complejidad, es necesario

implementar una solución que no sólo se limite al estado y estructura actual de las aplicaciones sino que también permita adaptarla a su crecimiento. Este módulo provee mecanismos flexibles para definir una estructura de datos a migrar y la lógica secuencial para su creación y borrado. De esta manera se asegura que aún modificando estructuras actuales o implementando nuevas a futuro se puedan integrar al proceso de transporte de configuraciones aquí planteado. El usuario, a través de una vista, podrá configurar esta lógica de transporte.

Se apoya en el módulo de control de versiones para identificar el cliente y es la versión más reciente de cada elemento o componente a transportar y sus dependencias.

4. Módulo de monitoreo de transporte: es el encargado de transportar y restaurar las configuraciones y proveer mecanismos de monitoreo para los procesos.

   Para transportar, el módulo crea las estructuras de datos (archivos binarios) de los elementos de configuración que ya han sido definidos y configurados previamente. Estas estructuras de datos poseen, la data, la lógica de restauración y los detalles de versión.

   Para la restauración, el módulo lee los archivos generados a partir de creación de un transporte, utiliza al módulo de control de transporte para estudiar y comparar la versión de cada elemento y dependencias a

restaurar contra los mismos elementos en la base de datos local y valida la creación o actualización de los mismos.

Es necesario conocer el estado de cada proceso a fin de verificar si se ha llevado a cabo con éxito. Este módulo permite, en tiempo real, estar al tanto de las actividades que rigen la creación, configuración, transporte, actualización y restauración de configuraciones.

5. Módulo de control de transporte: controla el proceso de transporte y actualización de configuraciones y permite al usuario estar en capacidad de revertir los cambios realizados durante el transporte (restauración).

   A través del módulo de control de versiones, identifica que estructuras de datos, en el ambiente destino, deben ser reemplazadas por aquellas generadas en el proceso de transporte.

I.3 Objetivos del proyecto

I.3.1 Objetivo general

Desarrollar una aplicación de transporte para el framework TRIAD que permita la creación, configuración, monitoreo y control de transporte de configuraciones entre diferentes ambientes operativos y de clientes, para las aplicaciones desarrolladas en la empresa, haciendo uso de elementos visuales que faciliten las tareas al desarrollador.

I.3.2 Objetivos específicos

- ✓ Desarrollar un módulo de control de versiones orientado al transporte de configuraciones. Este módulo debe estar en capacidad para:
  - Identificar los diferentes ambientes operativos por cada cliente y sus respectivas aplicaciones.
  - Proveer mecanismos para versionar la configuración de aplicaciones, elementos y dependencias.
  - Registrar los transportes realizados entre los diferentes ambientes operativos y de clientes a manera de historial.
  - Proveer al usuario de herramientas visuales a través de una vista que permita identificar ambientes operativos y de clientes, versión de las aplicaciones y configuraciones e historial de transportes.
- ✓ Desarrollar un módulo de creación de transporte que permita:

- Identificar y seleccionar los elementos de configuración a transportar.
- Identificar e incorporar las dependencias de los elementos seleccionados al transporte.
- Validar la compatibilidad entre la versión de los elementos y las dependencias que conforman el transporte.
- Proveer al usuario de herramientas visuales a través de una vista que permita la selección de los elementos de configuración transportables, tomando en cuenta los ambientes operativos y de cliente y la versión de los componentes transportables.

✓ Desarrollar un módulo de configuración de transporte que este en capacidad para:

- Definir la lógica del proceso de transporte para un conjunto de elementos y dependencias de configuración, tomando en cuenta el proceso de creación de esas estructuras y las restricciones y validaciones necesarias entre sus dependencias.
- Definir la lógica del proceso de restauración para un conjunto de elementos y dependencias de configuración, tomando en cuenta los procesos de creación, actualización y borrado de esas estructuras y las restricciones y validaciones necesarias entre sus dependencias.

- Proveer al usuario de herramientas visuales a través de una vista que permita la configuración de la lógica de transporte y restauración para un transporte de configuración determinado.

✓ Desarrollar un módulo de monitoreo de transporte que permita:

- Realizar el transporte, a partir de la creación de un conjunto de archivos secuencializados de transporte (AST), los cuales serán los elementos transportados. Estos archivos contendrán toda la data referente a las estructuras a transportar, la lógica que rige la restauración y versión de los elementos y las dependencias.

- Realizar la restauración, a partir de la carga de los archivos secuencializados de transporte (AST) y la restauración de la data en el ambiente destino.

- Monitorear en tiempo real y vía log los procesos de creación, configuración, transporte, actualización y restauración de configuraciones.

- Proveer al usuario de herramientas visuales a través de una vista que permita el monitoreo en tiempo real y vía log de los procesos de creación, configuración, transporte, actualización y restauración de configuraciones.

✓ Desarrollar un módulo de control de transporte que facilite:

- Controlar el proceso de transporte y restauración de configuraciones validando los elementos y dependencias de configuración entre los ambientes de origen y destino.

- Respaldar la data de configuración existente, previa restauración de transporte, por medio de la creación de archivos secuencializados de restauración (ASR) que incluya los elementos y las dependencias que van a ser modificados y sus respectivas versiones, al igual que la lógica de restauración de esas estructuras.

- Revertir los cambios realizados durante el transporte a partir de data de respaldo que permita regresar la aplicación al estado de configuración original.

I.4 Alcance y limitaciones

I.4.1 Alcance

A través de la propuesta planteada se espera obtener una solución integral al transporte de configuración de aplicaciones de la empresa entre los diferentes ambientes operativos y de clientes. El alcance definido para efectos de este proyecto contempla:

Una aplicación de transporte de configuraciones desarrollada sobre el framework TRIAD que implemente todas las funcionalidades necesarias para la creación, configuración, monitoreo y control de transporte de configuraciones.

La aplicación dividirá sus funcionalidades en los siguientes módulos:

Módulo de control de versiones: este módulo administra la compatibilidad entre los diferentes ambientes operativos y de clientes, las aplicaciones, sus

elementos de configuración y dependencias asociadas. El desarrollo de este módulo implica:

Un sub-módulo que permita identificar las aplicaciones de cada cliente sobre cada uno de los ambientes operativos e identificar así, en que etapa del ciclo de vida se encuentra una aplicación. Adicionalmente permite versionar la configuración de las aplicaciones, sus elementos y dependencias, para mantener la integridad y compatibilidad de cada uno de esos elementos y sus posibles interacciones, dependencias, etc.

Un sub-módulo de historial de transporte de configuraciones que permita llevar un registro de todos los transportes realizados entre los diferentes ambientes operativos y de clientes.

Una vista que permita identificar ambientes operativos y de clientes, versión de las aplicaciones y configuraciones e historial de transportes.

Módulo de creación de transporte: este módulo implementa las funcionalidades de identificación y selección de los elementos válidos de transporte y sus dependencias. El desarrollo de este módulo implica:

Un sub-módulo para identificar y seleccionar los elementos de configuración a transportar, que sea capaz de incorporar automáticamente las dependencias necesarias de esos elementos seleccionados al transporte.

Proveer al usuario de herramientas visuales a través de una vista que permita la selección de los elementos de configuración transportables, tomando en cuenta los ambientes operativos y de cliente y la versión de los componentes transportables.

Módulo de configuración de transporte: este módulo provee mecanismos flexibles para definir la lógica secuencial con la que se deben crear, transportar y restaurar los elementos válidos de transportes definidos durante el proceso de creación. El desarrollo de este módulo implica:

Un sub-módulo que defina la lógica del proceso de transporte para un conjunto de elementos y dependencias de configuración, tomando en cuenta el proceso de creación de esas estructuras y las restricciones y validaciones necesarias entre sus dependencias.

Un sub-módulo que defina la lógica del proceso de restauración para un conjunto de elementos y dependencias de configuración, tomando en cuenta los procesos de creación, actualización y borrado de esas estructuras y las restricciones y validaciones necesarias entre sus dependencias.

Una vista que permita la configuración de la lógica de transporte y restauración para un transporte de configuración determinado.

Módulo de monitoreo de transporte: es el encargado de transportar y restaurar las configuraciones y proveer mecanismos de monitoreo del transporte. El desarrollo de este módulo implica:

Un sub-módulo para la creación de los archivos secuencializados de transporte (AST).

Un sub-módulo para la restauración de archivos secuencializados de transporte (AST) en el ambiente destino.

Un sub-módulo de monitoreo de los procesos de creación, configuración, transporte, actualización y restauración de data de configuración.

Una vista que facilite el monitoreo de cada uno de los procesos involucrados en el transporte, en tiempo real y vía log.

Módulo de control de transporte: controla el proceso de transporte y restauración de configuraciones y permite al usuario estar en capacidad de revertir los cambios realizados durante el transporte. El desarrollo de este módulo implica:

Un sub-módulo de control que valide la integridad de las configuraciones durante el proceso de transporte y restauración tomando en cuenta los

elementos y dependencias de configuración entre los distintos ambientes operativos.

Un sub-módulo de respaldo que facilite salvaguardar la data de configuración existente, previa restauración de transporte. Adicionalmente debe estar en capacidad de revertir los cambios realizados y permitir regresar la aplicación al estado de configuración original

I.4.2 Limitaciones

La solución será desarrollada utilizando Adobe Flex y Java EE siguiendo los estándares de diseño según el lenguaje de modelado UML 1.4.

Se reconocen como "elementos de configuración" estructuras y dependencias definidas en los módulos: Elementos Descriptores, Clasificaciones y Parámetros, Elementos de visuales o vistas, Elementos de procesos - FTWS y Elementos de Datos o entidades.

El proceso de Transporte generará los archivos secuencializados de transporte (AST) localmente, necesarios para el proceso. Posteriormente desde el ambiente destino, el proceso de Restauración realizará la carga de los datos y restaurará los elementos transportados. Queda fuera del alcance de este trabajo definir la modalidad para llevar el archivo desde origen hasta destino.

I.5 Justificación

TRIAD es la plataforma tecnológica desarrollada para la construcción personalizada, rápida y eficiente, de soluciones empresariales orientadas a servicios, que permite reducir el complejo proceso de desarrollo de programas

a la elaboración de modelos gráficos y diseños de vistas, generando así automáticamente el código necesario para las aplicaciones.

Parte de la construcción de esas aplicaciones, y por ende, del ciclo de vida de los proyectos empresariales, corresponde a la implementación de las nuevas funcionalidades en los diferentes ambientes operativos (fase de prueba) y los diferentes ambientes de cliente (fase de implementación).

A través de una herramienta automatizada para el transporte de aplicaciones podrían cosecharse gran cantidad de beneficios.

Desde el punto de vista práctico, se minimizaría el tiempo necesario para adaptar un ambiente de prueba para nuevas funcionalidades, se minimizaría el tiempo necesario para actualizar las aplicaciones en los clientes, se optimizarían los procesos de soporte en los clientes.

Desde el punto de vista teórico, los procesos de transporte de nuevas funcionalidades y soporte podrían ser llevados a cabo por un número reducido de técnicos que no van a necesitar conocimientos específicos relacionados con las funcionalidades que están siendo transportadas, ya que el procedimiento de transporte sería estándar y transparente para quien lo utiliza.

Desde el punto de vista metodológico, se tendrían las herramientas y bases tecnológicas para estandarizar el versionamiento de las aplicaciones y todos sus componentes, organizando el procedimiento.

CAPÍTULO II. Marco teórico

La creciente necesidad de optimizar el soporte a los clientes de las aplicaciones desarrolladas, conlleva a la búsqueda de soluciones automatizadas que intercedan en los procesos inherentes.

A fin de automatizar tales procesos es necesario adaptarlos a la arquitectura empresarial de desarrollo, basada en los estándares J2EE por los que TRIAD y las aplicaciones se rigen.

Es a través del estudio de proceso de despliegue de las aplicaciones y del conocimiento de la arquitectura de la aplicación que se puede determinar concretamente que se debe transportar y como ha de realizarse el procedimiento.

II.1 Aplicaciones J2EE

"Java Platform, Enterprise Edition o Java EE (anteriormente conocido como Java 2 Platform, Enterprise Edition o J2EE hasta la versión 1.4), es una plataforma de programación, parte de la plataforma Java, para desarrollar y ejecutar software de aplicaciones en lenguaje de programación Java con arquitectura de N capas distribuidas y que se apoya ampliamente en componentes de software modulares ejecutándose sobre un servidor de aplicaciones. La plataforma Java EE está definida por una especificación. Similar a otras especificaciones del Java Community Process, Java EE es también considerada informalmente como un estándar debido a que los

proveedores deben cumplir ciertos requisitos de conformidad para declarar que sus productos son conformes a Java EE; estandarizado por The Java Community Process / JCP." (1)

"Uno de los beneficios de Java EE como plataforma es que es posible empezar con poco o ningún coste. La implementación Java EE de Sun Microsystems puede ser descargada gratuitamente, y hay muchas herramientas de código abierto disponibles para extender la plataforma o para simplificar el desarrollo." (1)

Cinet Solutions C.A. hace uso de la plataforma Eclipse, un IDE basado en Java, que apoyado sobre el framework AndroMDA (permite, a partir de un modelo, crear una aplicación desplegable) y la herramienta Maven (administradora de proyectos de software) permite manejar los proyectos.

Aplicaciones J2EE pueden ser desplegados sobre una amplia gama de servidores de aplicación, tanto libres como propietarios.

La empresa hace uso de JBOSS como servidor de aplicación empresarial.

II.1.1 Empaquetado de las aplicaciones Java EE

"Una aplicación Java EE se compone de diferentes módulos o componentes. Las aplicaciones Java EE y sus módulos necesitan contener información sobre cómo se relacionan entre ellos. Esta información se incluye en documentos XML denominados **DD** (Deployment Descriptors, Descriptores de Despliegue).

La siguiente figura muestra la distribución de una aplicación Java EE con todos sus módulos." (9)

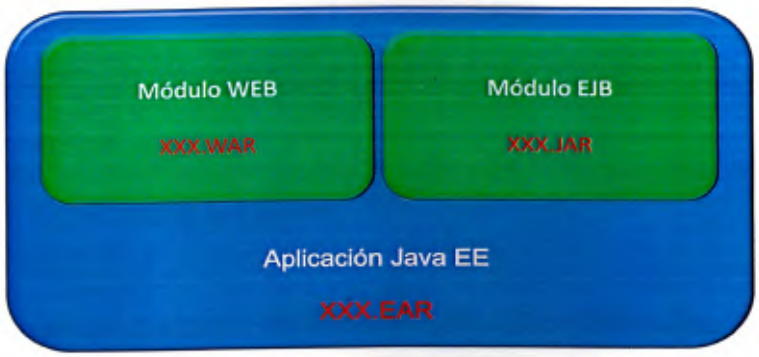

Figura 2. Arquitectura modular de una aplicación J2EE

Fuente: The J2EE Architect's Handbook (2004).

"La aplicación Java EE está contenida en una carpeta con extensión .EAR. Dentro de ella se encuentra el módulo web y los módulos EJB necesarios para su funcionamiento.

Aplicación Java EE: se empaqueta en un **fichero EAR** (Enterprise Archive) con extensión **.ear** y contiene:

- Archivos con extensión **.war** y **.jar.**
- Un DD llamado **application.xml.**

Módulo Web: contiene todos los elementos de tipo web necesarios para el funcionamiento de la aplicación. Se empaqueta en un **fichero WAR** (Web Archive) con extensión **.war** y contiene:

- JSPs, Servlets y JavaBeans.
- Otros ficheros que no son considerados como componentes Web por la especificación Java EE:
    - Páginas HTML, Applets y Clases relacionadas con la parte servidora.
    - Un DD llamado **web.xml.**

Módulo EJB: se empaqueta en un **fichero EJB-JAR** (Enterprise JavaBean-Java Archive) con extensión **.jar** y contiene:

- Clases EJBs (.class).
- Clases de ayuda relacionadas (.class).
- Un DD llamado **ejb-jar.xml**." (9)

II.1.3 Despliegue de aplicaciones JAVA EE en el servidor de aplicaciones JBOSS

Para poder desplegar aplicaciones J2EE es necesario conocer el proceso de empaquetado de sus aplicaciones J2EE para JBoss.

Este extracto es un compilado del conocimiento recopilado de diferentes websites y libros, junto con la experiencia que se ha adquirido a lo largo del desarrollo de este proyecto por parte del equipo de trabajo.

II.1.3.1 Introducción a JARs, WARs, SARs, RARs y EARs

"Parte de las características de los archivos JAR es que pueden almacenar meta información valiosa a cerca de su contenido. El fichero especial *META-INF/MANIFEST.MF* es donde el jar guarda la meta información.

Los EJBs también se empaquetan como ficheros jar, aunque contienen uno o más ficheros especiales bajo *META-INF*, llamados descriptores de despliegue (deployment descriptors)." (6)

"Un fichero WAR empaqueta una aplicación web. Las aplicaciones web están compuestas por

- Páginas HTML/JSP así como otros recursos tales como imágenes/archivos javascript/hojas de estilo que van a ser accesibles mediante URLs.
- Ficheros adicionales tales como servlets java, clases de utilidad y jars usados en servlets/JSPs así como descriptores de despliegue que no deberían ser accesibles mediante URLs.

Los últimos son almacenados bajo una carpeta especial llamada *WEB-INF*. Ambos se archivan en un JAR cuya extensión es modificada a WAR para distinguirla como aplicación web java."

"De modo similar, los ficheros RAR son ficheros JAR usados para empaquetar módulos adaptadores de recursos tal y como se define en la J2EE Connector

Architecture (JCA) y los ficheros SAR se utilizan para empaquetar servicios con capacidades JMX en JBoss.

Del mismo modo que el destino de todos los ríos es juntarse en un mar, todos los archivos relacionados con una aplicación empresarial se pueden empaquetar juntos bajo un EAR."(6)

II.2 Configuración de aplicaciones

"Las aplicaciones desarrolladas por la empresa hacen uso del framework TRIAD como pilar fundamental de desarrollo y operación. El framework TRIAD ofrece todos los recursos y herramientas para diseñar, configurar y validar las pantallas y sus componentes, manejar y configurar los roles de usuario, los catálogos, parámetros de la aplicación y transacciones." (Fuente: Elaboración propia)

II.2.1 Arquitectura TRIAD

"TRIAD provee una experiencia integrada y optimizada para desarrolladores y expertos en procesos de negocio para aumentar su productividad dramáticamente.

Este ambiente ofrece los siguientes beneficios al negocio:

- Permite que la organización cree aplicaciones configurables rápidamente y en un significante bajo costo.
- Acelera la adopción al Enterprise SOA y composición de procesos de negocios.

- Incremento en la productividad en el desarrollo a través de herramientas y ambientes integrados.
- Reduce el costo de desarrollo, entrenamiento y soporte.

"La clave motivadora para adoptar TRIAD como una plataforma elegida es el desarrollo de aplicaciones personalizadas. Es ahora bien aceptado que las arquitecturas orientadas a servicios tengan bastante sentido, la pregunta es cómo llegar ahí. " (Fuente: Elaboración propia)

"TRIAD ayuda a acelerar el camino para llegar a una arquitectura orientada a servicios, simplifica las tareas existentes en el ambiente IT exponiendo las funcionalidades del Core actual de la aplicación como servicios y permite combinar esos servicios con otros para darle poder a los componentes. " (Fuente: Elaboración propia)

"Triad hace uso de un conjunto de módulos con herramientas visuales que permiten llevar a cabo la construcción de las aplicaciones:

- GenWap: es la aplicación de diseño de pantallas. Permite a los desarrolladores crear las pantallas de la aplicación, agregarle los componentes (campos de texto, comboboxes, checkboxes, listas, tablas, imágenes, etc), configurarlos (cada componente puede requerir alguna configuración específica como por ejemplo un campo de texto obligatorio), validarlas (configurar la lógica de ejecución de la pantalla, por ejemplo que un campo se habilite cuando un componente tiene cierto valor seleccionado) y funcionalizarla (asigna a componentes

llamadas a funciones que manejan todas las transacciones y operaciones con la base de datos)

- CatalogManager: es la aplicación que permite crear y editar los catálogos de las aplicaciones. Un catálogo es una estructura de datos, opcionalmente jerárquica, que se utiliza entre muchas cosas, como listado de actividades y contenido de comboboxes.
- FDTS: es la aplicación que permite construir las operaciones transaccionales. Desde la pantalla se llama directamente a una función, que define documentos que contienen transacciones que llaman servicios. Toda la lógica de creación de las estructuras de funciones, documentos, transacciones y asignaciones de servicios a esas transacciones, se realizan a través de esta aplicación.
- Paramet: es la aplicación que permite crear y editar los parámetros de las aplicaciones.
- Reportes: es la aplicación que permite configurar la generación de reportes." (Fuente: Elaboración propia)

II.3 Base de datos orientada a objetos (DB4O)

"DB4O es un novedoso motor de base de datos orientada a objetos. Sus siglas se corresponden con la expresión "DataBase 4 (for) Objects", que a su vez es el nombre de la compañía que lo desarrolla: db4objects, Inc.

Las claves innovadoras de este producto es su alto rendimiento (sobre todo en modo embebido) y el modelo de desarrollo que proporciona a las aplicaciones

para su capa de acceso a datos, el cual propugna un abandono completo del paradigma relacional de las bases de datos tradicionales." (7)

"De este modo, tenemos las siguientes consecuencias directas resultantes de este nuevo paradigma:

- Deja de existir un lenguaje SQL de consultas/modificaciones para pasar a crearse sistemas de consulta por métodos delegados y actualización/creación/borrado automático de entidades mediante código compilable.
- Se elimina la necesidad de representar el modelo de datos de la aplicación en dos tipos de esquemas: modelo de objetos y modelo relacional. Ahora el esquema de datos del dominio viene representado por la implementación que se realice del diagrama de clases.

Se consigue evitar el problema del Object-Relational Impedance Mismatch sin sacrificar el rendimiento que los mapeadores objeto-relacionales sufren actualmente para llevar a cabo el mismo objetivo." (7)

"La mayor clave del éxito que está teniendo este motor de base de datos frente a otros competidores que han desarrollado tecnologías similares, es que se ha optado por un modelo de licenciamiento idéntico al utilizado por empresas como MySQL: licencia dual GPL/comercial. Actualmente este producto funciona como una biblioteca para dos tipos de plataformas de desarrollo: Java y.NET (tanto la implementación de Microsoft como la de Mono)." (7)

CAPÍTULO III. Marco metodológico

III.1 Metodología aplicada

Para el desarrollo de la solución planteada en este proyecto se hará uso de la metodología de *Método Dinámico de Desarrollo de Sistemas (DSDM o "Dynamic System Development Method").* La razón principal para la utilización de esta metodología en el desarrollo de este proyecto radica en que permite, de manera incremental, ir agregando funcionalidades paulatinamente a versiones ya existentes del desarrollo. "Es posible desarrollar una versión con las funcionalidades básicas imperativas del proyecto en un corto tiempo y posteriormente, refinarlo a discreción hasta alcanzar el producto deseado a través de entregables continuos. Permite, además, flexibilidad al cambio e implementación de funcionalidades nuevas." (DSDM in a bird´s eye review, 2008)

III.2 Definición de Método Dinámico de Desarrollo de Sistemas (DSDM)

"Esta metodología se basa en programación rápida de aplicaciones (RAD), por ejemplo, puedes desarrollar aplicaciones usando cualquier entorno de desarrollo de aplicaciones como NetBeans, Eclipse,Sun Java Studio Creator,etc. Se considera la primera metodología ágil. La metodología a seguir será implementada utilizando la técnica de "TimeBoxing", la cual es utilizada para, dentro de un determinado tiempo y presupuesto, mantener los objetivos de la metodología en curso. La idea principal detrás de la técnica de

"TimeBoxing" es el de separar el proyecto en porciones, cada una con un presupuesto y una fecha de entrega determinada. Por cada porción se asigna a la iteración un conjunto de funcionalidades definida en los requerimientos basados en una prioridad definida por el principio MoSCoW." (8)

"El principio MoSCoW representa una forma de priorizar ítems o, aplicados a la metodología, requerimientos. El acrónimo se desglosa en:

M (Must): son funcionalidades fundamentales de la aplicación en desarrollo.

S (Should): son funcionalidades que, si bien se buscar implementar, no definen el éxito del proyecto.

C (Could): todas aquellas funcionalidades que no afectan las necesidades de negocio del proyecto.

W (Would): todas las funcionalidades que pueden ser entregadas una vez el proyecto esté "terminado" y postergados para versiones futuras. " (8)

En la siguiente figura se muestra la metodología a implementar:

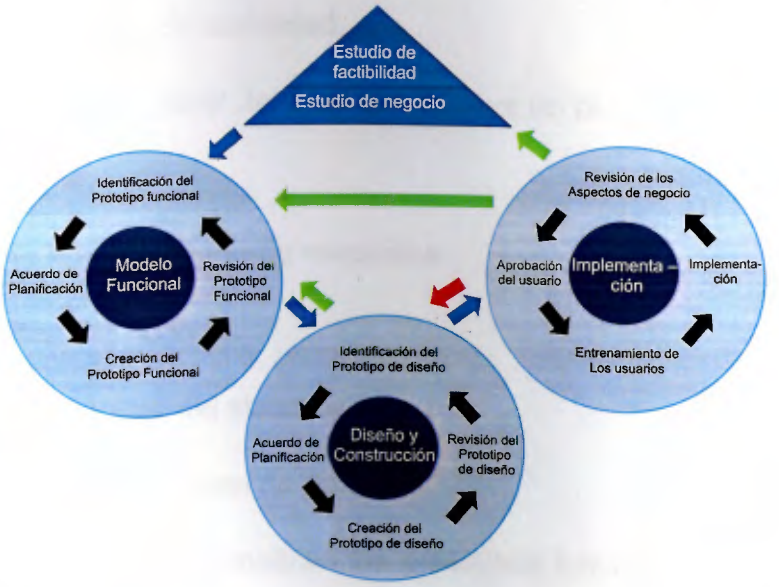

Figura 3. Método Dinámico de Desarrollo de Sistemas adaptado a la solución

Fuente: DSDM in a bird´s eye review (2008)

Con esta metodología se busca:

- ✓ Enfocar el proyecto a entregas frecuentes de productos.
- ✓ Integrar las pruebas al ciclo de vida del proyecto.
- ✓ Desarrollo de una versión funcional con las características mandatorias en corto tiempo y posteriormente, adaptación y refinamiento de funcionalidades.

La metodología cuenta con cinco (5) fases:

"Fase pre-proyecto: el proyecto es conceptualizado y se toma la decisión de iniciarse. En esta fase se realiza un estudio de viabilidad y un estudio de negocio a través de los cuales se define si el proyecto es viable dentro del tiempo y presupuesto definido y se estudian los aspectos técnicos y operativos relacionados con el desarrollo y despliegue. " (8)

Entregables:

- Se obtiene el informe de viabilidad.

- Se obtiene un prototipo de viabilidad.
- Se obtiene un esbozo global del plan que incluye un plan de desarrollo y un registro de riesgos.
- Se obtiene una lista priorizada de requisitos.
- Se define el área del negocio.
- Se define la arquitectura del sistema.
- Se obtiene un esbozo del plan de prototipo. (8)

"Fase de modelo funcional: se realizan los prototipos funcionales del sistema. Estos prototipos modelan las funciones que el sistema debería realizar y como se realizarían las mismas. " (8)

Entregables:

- Se obtiene un modelo funcional y un prototipo funcional.
- Se actualiza la lista de requisitos, borrando los que ya se han realizado y replanteando la prioridad del resto de requisitos.
- El registro de riesgos se actualiza. (8)

"Fase de diseño y construcción: el producto es diseñado y desarrollado en iteraciones. En cada iteración un área especificada del proyecto es diseñada, desarrollada y revisada. " (8)

Entregables:

- Se obtiene un prototipo de diseño.
- El sistema estará construido en su mayoría.
- Se obtiene la documentación de usuario. (8)

"Fase de Implementación: en esta fase el producto es ensamblado y documentado. Posteriormente se comparan las características del producto obtenido con los requerimientos del proyecto. " (8)

Entregables:

- El sistema entregado estará implantado en la empresa, listo para ser usado por los usuarios finales.
- Los usuarios finales conocerán el funcionamiento del sistema.
- Se obtiene el documento de revisión del proyecto. (8)

"Fase post-proyecto: una vez el producto es creado se genera un plan de mantenimiento. " (8)

"DSDM define un conjunto de **roles** necesarios que deben ser cubiertos por los miembros del equipo de proyecto:

- ✓ Embajador (Giancarlo Sánchez): es quien interactúa entre los usuarios finales y los desarrolladores del sistema. Para efectos del proyecto, el sistema está orientado a los desarrolladores.
- ✓ Visionario (Carlos Inguanzo): es quién mantiene al proyecto en rumbo a los objetivos del negocio.
- ✓ Consejero (Carlos Zambrano): es quién posee los conocimientos técnicos y prácticos del negocio sobre el cual se desarrolla la solución.
- ✓ Coordinador técnico (Carlos Zambrano): coordina los aspectos técnicos del sistema y supervisa que interactúen correctamente.
- ✓ Patrocinador ejecutivo (Carlos Inguanzo): contribuye con los fondos y recursos necesarios para el desarrollo del proyecto.

- ✓ Desarrollador (Giancarlo Sánchez): posee el conocimiento y la experiencia para llevar la planificación y requerimientos a código entregable. ” (8)

CAPÍTULO IV. Desarrollo

El desarrollo del proyecto adaptado a la metodología DSDM está definido por tres fases importantes, cada una de ellas ejecutada iterativamente hasta alcanzar el refinamiento del producto de la misma en cuestión:

La fase de pre-proyecto, estudia la factibilidad de la solución a implementar y el negocio alrededor del cual el proyecto se ejecuta. Se analizan los requerimientos, se estudia el entorno técnico alrededor del cual el proyecto ha de ser desarrollado, se establece el grupo y las mecánicas de trabajo.

La fase de proyecto, que comprende tres etapas iterativas en las cuales se refina el producto de cada una de ellas.

La primera etapa es el modelo funcional, el cual especifica las funcionalidades deseadas con base en los requerimientos especificados, conceptualización del diseño sobre las mismas bases y priorización de las funcionalidades a implementar. Para definir las funcionalidades y priorizarlas, múltiples reuniones o workshops fueron llevados a cabo en donde se discutieron las alternativas a implementar para abarcar los requerimientos y la prioridad dentro del desarrollo que estas funcionalidades llevarían. Tras refinar el listado de funcionalidades y el diseño conceptual de la aplicación de transporte en un total de tres iteraciones (en donde cada iteración comprendió una reunión para definir las funcionalidades, un modelo funcional de la solución y un diseño conceptual de la aplicación) se procedió a la fase de diseño y construcción.

La segunda etapa es el diseño y construcción. Esta etapa, también evolutiva, contó con el refinamiento periódico con un total de tres iteraciones sobre las cuales se pasaba de un diseño de la aplicación a la construcción de la misma, de manera modular. En cada iteración se diseñaba y construían las funcionalidades de la aplicación que, según su prioridad establecida en la etapa anterior, se determinaron.

La tercera etapa corresponde a la implementación. Si bien la metodología plantea también un esquema iterativo para esta etapa, para efectos del proyecto se realizó en una sola iteración. Una vez completada la construcción del sistema se realizó el despliegue en ambiente de desarrollo en donde principalmente el sistema ha de ser usado. Esta tercera etapa de la fase de proyecto, marca la conclusión de la misma.

Finalmente la tercera fase, la fase de post-proyecto desarrolla un plan de mantenimiento y evalúa el rendimiento del equipo del trabajo y el proyecto.

A continuación de manera, más detallada, se describen las fases y sus correspondientes etapas.

IV.1 Fase pre-proyecto

Esta es la fase inicial del ciclo de vida del software según la metodología DSDM. Se realizaron los estudios y recopilación de información adicional a los requerimientos ya levantados para la ejecución del proyecto.

Dos grandes actividades fueron realizadas durante esta fase:

IV.1.1 Estudio de factibilidad

La primera acción durante esta etapa fue definir la metodología. Una reunión con el equipo técnico de proyecto conformado por el tutor empresarial, el desarrollador (tesista) y el presidente de la empresa llegó a la conclusión que DSDM sería la metodología ideal para el desarrollo de la solución, gracias a que la misma incorpora características de desarrollo ágil con entregables frecuentes, permite una gestión de cambios rápida y favorece rápidos desarrollos.

Dentro de la documentación generada se obtuvo un reporte de factibilidad, un plan de trabajo global y un log de riesgos.

IV.1.2 Estudio del negocio

En esta etapa se analizaron las características técnicas y operativas del negocio. Se determinó que los workshops serían semanales y se discutieron las facetas más importantes del sistema a desarrollar. En esta fase se crea una lista de requerimientos funcionales y no funcionales, la cual es priorizada. Adicionalmente se define la arquitectura del sistema y un borrador de plan de prototipado es creado.

Entregables Obtenidos:

1. Documento de visión con el estudio de viabilidad y plan de trabajo.
2. Documento de diseño conceptual de la solución.

IV.2 Fase de proyecto

Esta fase se inicia analizando los requerimientos, e identificando las propiedades funcionales y no funcionales requeridas y deseadas de la aplicación a desarrollar. Una vez listadas esas propiedades, son priorizadas y un plan de trabajo basado en un esquema iterativo es generado, para atacar en orden de prioridad los requerimientos planteados.

El desarrollo es llevado a cabo en 3 iteraciones, cada una de ellas, incluyendo funcionalidades catalogadas por prioridad y todas bajo el mismo esquema de Diseño, Construcción e Implementación.

IV.2.1 Modelo Funcional

Analizando los requerimientos se desglosan las características funcionales y no funcionales esperadas por la aplicación a desarrollar. A través del método de MoSCoW se agrupan en 3 categorías el conjunto de requerimientos y se asignan a una iteración que corresponda con su prioridad.

Descripción	Módulo	Tipo
Mecanismos de versión para elementos y dependencias en las aplicaciones	Versionamiento	Servicio, Vistas
Identificación de las estructuras, componentes, tablas y atributos de configuración.	Configuración	Servicio
Selección de las estructuras, componentes, tablas y atributos de configuración a transportar.	Control	Servicio
Lógica de transporte para las estructuras y sus componentes. Incluye lógica de creación y de restauración.	Control	Servicio, Vistas
Creación de los archivos AST	Monitoreo	Servicio, Vistas

Restauración de los archivos AST	Monitoreo	Servicio, Vistas

Tabla 1. Funcionalidades mandatorias según MoSCoW

Descripción	Módulo	Tipo
Monitoreo del proceso de selección de elementos configurables	Monitoreo	Servicio
Monitoreo del proceso de configuración de los elementos	Monitoreo	Servicio, Vistas
Monitoreo del proceso de creación de los archivos AST	Monitoreo	Servicio, Vistas
Monitoreo del proceso de restauración	Monitoreo	Servicio, Vistas
Identificación y control de ambientes operativos y de clientes	Versionamiento, Control	Servicio
Registro de transportes (Historial)	Versionamiento	Servicio, Vistas

Tabla 2. Funcionalidades importantes según MoSCoW

Descripción	Módulo	Tipo
Mecanismos de proceso de restauración	Control	Servicio, Vistas
Control de restauración	Control	Servicio
Monitoreo de restauración	Control	Servicio, Vistas
Mecanismos de creación de respaldo	Control	Servicio, Vistas
Monitoreo creación de respaldos	Control	Servicio, Vistas
Control de creación de respaldos	Control	Servicio

Tabla 3. Funcionalidades adicionales según MoSCoW

Adicionalmente, existen un conjunto de requerimientos no funcionales, orientados a la presentación del sistema, el tiempo de respuesta y facilidad de uso que se especifican a continuación, fuera del marco de evaluación de MoSCoW y presentes durante todo el desarrollo del proyecto.

Descripción	Módulo	Tipo
Aspecto de la interfaz de Usuario. La interfaz debe ser atractiva y amigable. Fácil de usar	General	Vistas
El proceso de transporte se adapta a múltiples ambientes de trabajo bajo diferentes manejadores de base de datos.	General	Servicio
Tiempo de respuesta dentro de lo esperado	General	Servicio
Tolerancia a fallos. El sistema debe poder recuperarse ante fallos.	General	Servicio, Vistas
Debe permitir mantenibilidad para subsecuentes desarrollos	General	Servicios

Tabla 4. Requerimientos no funcionales y de calidad

Una vez listados los requerimientos el equipo de trabajo se reúne para proponer un plan de ejecución.

La creación del prototipo funcional evolucionó sobre tres iteraciones en las cuales se evaluaron los requerimientos, se priorizaron y asignaron a una iteración para la fase de construcción.

Durante la primera iteración se crea un modelo funcional global de la solución, que asigne todos los requerimientos funcionales y no funcionales a una fase del desarrollo. Se crea un documento o "prototipo funcional" explicando el modelo a través de especificación de los detalles técnicos, validaciones, casos de uso y diagramas de flujo.

Las consecuentes iteraciones tienen como objetivo el de refinar las asignaciones, incorporar aquellas nuevas funcionalidades que surgieron como consecuencia de la gestión de cambios, etc.

Una vez refinado el prototipo funcional, el equipo de trabajo se reúne nuevamente para analizar el prototipo y generar las recomendaciones que van a ser acatadas para las siguientes etapas de desarrollo.

Entregables Obtenidos:

1. Diseño Funcional refinado en tres iteraciones.
2. Prototipo Funcional.

IV.2.2 Diseño y Construcción

IV.2.2.1 Diseño

El diseño de la aplicación está dividido en dos grandes partes

Diseño de servicios (Java): comprende todo lo que es creación de servicios CRUD sobre base de datos relacional de TRIAD y sobre la ODB de Transporte y la creación de las estructuras sobre las cuales la data será transportada y desde y hasta donde será transportada.

Diseño de pantallas (Flex): abarca todo el manejo de las Vistas de la aplicación y las llamadas a los servicios.

La aplicación está en capacidad para transportar todo lo que una nueva implementación necesita para poder funcionar correctamente sobre el destino

en el que es actualizado. Para poder entender mejor qué se transporta y como se transporta se explica la granularidad del transporte.

IV.2.2.1.1 Granularidad del transporte

La granularidad del transporte determina que elementos pueden ser transportados y como esos elementos deben ser ensamblados para asegurar un correcto transporte.

En la siguiente figura se define la granularidad del transporte que está atado directamente a los elementos versionados.

Figura 4. Granularidad del transporte

Fuente: Elaboración propia

La granularidad está definida por la lógica de negocio. A través de esa lógica se concluye que al transportar una vista, deben transportarse todos los elementos que la componen y aseguran su funcionamiento. En la siguiente figura se muestran los elementos particulares que una vista necesita para su funcionamiento. Todos los elementos de la vista son incluidos en el Versionamiento y por consecuencia en el transporte.

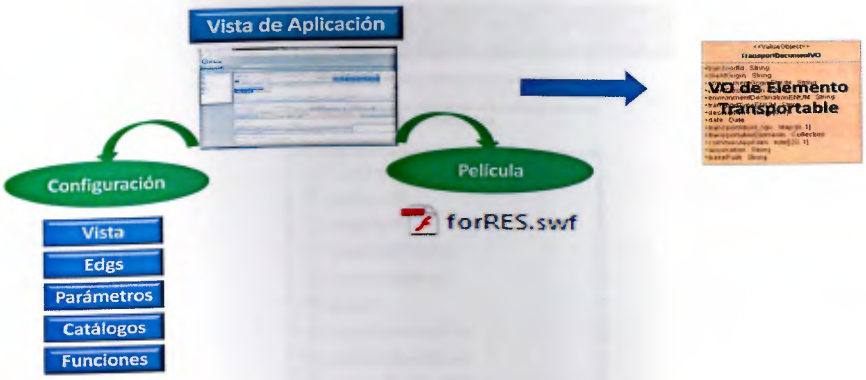

Figura 5. Arquitectura de una vista de la aplicación

Fuente: Elaboración propia

Adicionalmente existen elementos comunes de la aplicación que deben ser incluidos en cada transporte y que son comunes para todas las vistas de una aplicación. Cada módulo desplegado en el .war de la aplicación en el JBOSS posee la siguiente estructura.

Una carpeta assets, en donde las imágenes, logos, hojas de estilo y configuraciones web son colocadas. Estos archivos son comunes para toda la aplicación.

Un conjunto de archivos de tipo librería. Comunes también para la aplicación.

Otras carpetas que contienen las películas swf de cada vista de la aplicación. La figura a continuación muestra la estructura de los archivos externos comunes.

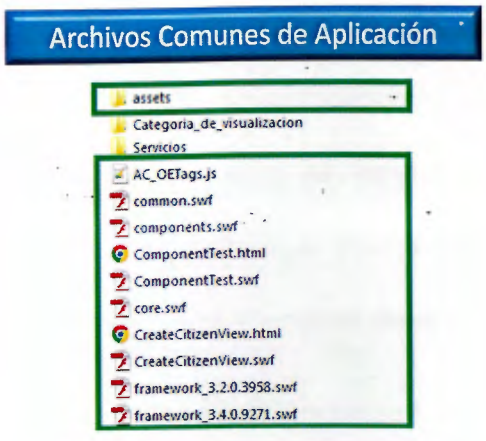

Figura 6. Archivos externos comunes de la aplicación

Fuente: Elaboración propia

Para efectos de la aplicación un elemento es transportable si está versionado. Un elemento versionado por definición puede ser una vista o un catálogo. Por convención y funcionalidad, parámetros, egds (estructuras de datos para fácil identificación de componentes entre los servicios Java y los servicios Flex) y funciones son transportados como dependencias de una vista que los utilice ya que por si solos son data dependiente de componentes adicionales. Sin embargo una vista, es un elemento principal que puede ser incluida en un transporte. Si esta vista posee vistas adicionales, edgs, parámetros, catálogos o funciones, serán versionadas como dependencia de esa vista.

Catálogos son otro elemento principal, independiente de otros componentes, al contrario, son otros componentes los que utilizan catálogos. Por esa razón y por la ventaja que representa el independizar su transporte, los catálogos son versionados independientemente de las vistas.

Es así como se muestra que durante el proceso de Versionamiento, a partir de vistas y catálogos es que se realizarán los versionamientos. Si una vista hace uso de otras subvistas o catálogos, estos se versionarán primero y serán agregados en forma de VO o ValueObject (Tipo de datos utilizado por las vistas de la aplicacion) de elemento de transporte como dependencia de la vista que los requiere

En la figura siguiente se muestra un ejemplo de una estructura compleja de Versionamiento. Dada una supervista llamada "Formulario" que posee parámetros, edgs, funciones y los catálogos de Género y País. Adicionalmente esa vista posee una subvista llamada Vista A que adicionalmente a sus parámetros, edgs y funciones utiliza el catálogo de ciudad. Como se indica en la figura múltiple Versionamiento de elementos se lleva a cabo al decidir versionar "Formulario".

En primera instancia cada catálogo se versiona por separado al igual que cada vista como fue definido según la granularidad. En el caso de la Vista A el catálogo de ciudad es incluido dentro de la versión como un objeto versionado independiente. De igual manera para el Formulario, tanto sus subvistas como catálogos son incluidos dentro de su estructura versionada pero cada uno de esos elementos es versionado independientemente.

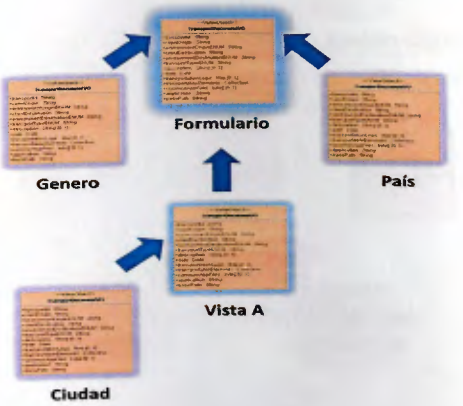

Figura 7. Dependencia entre versiones de vistas y catálogos

Fuente: Elaboración propia

Siguiendo esta misma definición, entonces es posible que al decidir versionar la pantalla Formulario, durante un transporte se pueda incluir cualquiera de sus dependencias versionadas como elemento independiente (teniendo en cuenta que al seleccionar un elemento transportable, se incluyen todas sus dependencias).

El diseño de manejo de datos se presenta en la siguiente figura. Las flechas verdes y azules indican el flujo de datos durante la creación de un transporte mientras que las rojas describen el flujo de datos durante el proceso de restauración de un transporte en el ambiente destino. Las estructuras de datos con bordes azules representan la data que es manejada por la base de datos relacional mientras que las estructuras con bordes verdes representan la data manejada por la base de datos orientada a objetos del transporte.

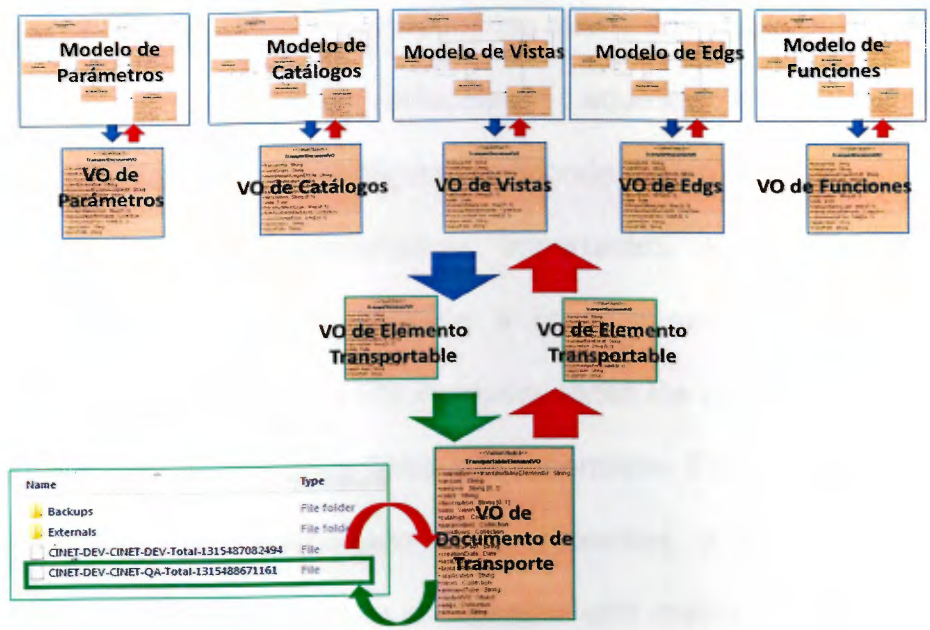

Figura 8. Flujo de datos y estructuras en el transporte

Fuente: Elaboración propia

La función central de la aplicación de transporte es la de poder transportar data entre ambientes. Estos ambientes, definidos por la arquitectura tecnológica de la empresa están manejados actualmente por bases de datos relacionales. En consecuencia el objetivo es poder transportar data entre dos bases de datos relacionales.

Utilizando la misma arquitectura MVC de TRIAD los esquemas de datos relacionales de cada modelo a incorporar en el transporte son convertidos en Vos de su misma estructura. Al crear una nueva versión de un elemento con la aplicación de transporte se genera un VO de elemento transportable que incorpora toda la data de los Vos necesaria proveniente de los módulos anteriormente mencionados. Estos VO de elementos transportables son almacenados en forma de objetos en el repositorio de transporte.

Al crear un transporte se consultan todos los elementos versionados dentro del repositorio de transporte, se seleccionan aquellos a incorporar a un determinado transporte y se configuran indicando origen, destino, aplicación a la que pertenecen y características importantes a conocer. Una vez configurado el transporte se procede a crear y esto genera un VO de documento de transporte. Este VO incorpora todas los elementos versionados seleccionados y crea un archivo físico que lo contiene. Este archivo es el objeto físico que puede ser transportado entre ambientes, a través de diferentes medios físicos, correo electrónico o cualquier otro medio (futuros desarrollos contemplan un módulo de transporte remoto para esta funcionalidad).

Una vez este archivo de transporte es cargado en el destino (utilizando la aplicación de transporte) su contenido es identificado y el VO de documento de transporte leído. Se identifican los elementos transportables que contiene y el usuario al decidir actualizar el destino llama a un conjunto de rutinas Java que se encargan copiar los archivos externos en su respectivo lugar y actualizar la data sobre la base de datos relacional. La flecha roja de la figura anterior indica el flujo de datos durante la actualización de data en el destino.

Como se especificó anteriormente, los archivos externos se dividen en archivos externos de vista y archivos externos de aplicación.

Cada vista posee una película SWF que se carga una vez que la pantalla es solicitada a través del browser.

Por otro lado cuando se inicializa la aplicación como tal se cargan los archivos comunes de la aplicación que estaban conformado por los assets y librerías.

Para efectos del transporte se toman en consideración ambos grupos de archivos externos. Durante el Versionamiento de vistas, se guarda además de la configuración de la misma, su película correspondiente. Estas estructuras son almacenadas en un VO de Elemento Transportable. Por cada vista que es versionada se crea un VO de elemento transportable que individualmente carga las estructuras correspondientes.

Por otro lado los archivos comunes de la aplicación son cargados durante la creación del transporte. Al seleccionar versiones de elementos a incluir en el transporte, la aplicación identifica a que aplicación pertenecen esas vistas y carga los archivos externos de esa aplicación automáticamente, incorporándolas al VO de Documento Transportable que finalmente es llevado a un archivo de transporte, listo para ser movilizado.

IV.2.2.1.2 Estructuras de transporte

La aplicación de transporte se aparta del esquema MVC y utiliza únicamente los ValueObjects (VO) como estructuras de almacenamiento de data. Estos VO son directamente guardados como objetos en el repositorio de transporte y los archivos de transporte y respaldo.

TransportableElementVO: es la estructura sobre la cual se manejan las versiones de los catálogos y las vistas (incluyendo sus dependencias, tanto lógicas o configuraciones como físicas o archivos externos).

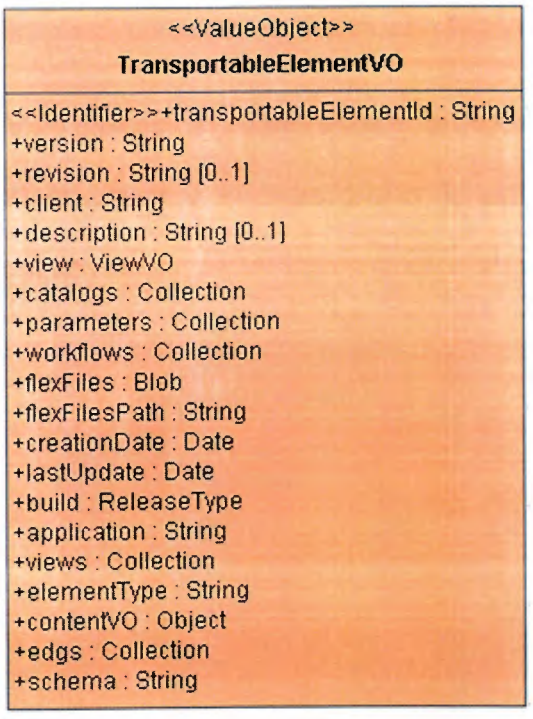

Figura 9. Estructura de un elemento versionable

Fuente: Elaboración Propia

TransportableEdgVO: estructura sobre la que se guardan los edg. Esta estructura engloba el modelo completo de edgs y puede ser encontrado por cada edg de una vista, dentro de la estructura TransportableElementVO de la vista que lo posee.

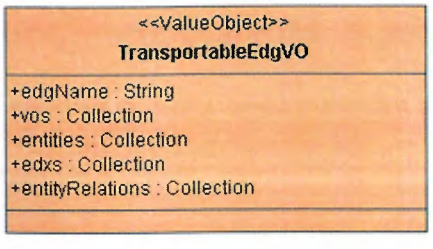

Figura 10. VO de transporte para encapsular Edgs y sus valores

Fuente: Elaboración Propia

TransportableParameterVO: similar a la estructura anterior, engloba toda la data proveniente del modelo paramet. Incluye grupo de parámetro, definición, valores e instancia de sistema.

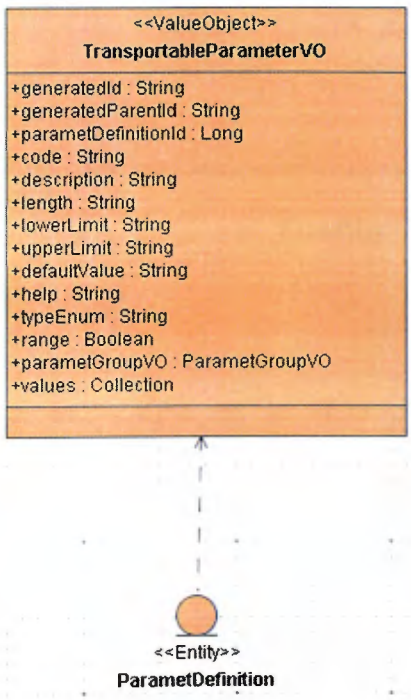

Figura 11. VOs de transporte para encapsular parámetros

Fuente: Elaboración Propia

TransportabelParameterValueVO: esta estructura engloba la data del modelo de valores de parámetros y es una dependencia directa dentro de la estructura de TransportableParameterVO.

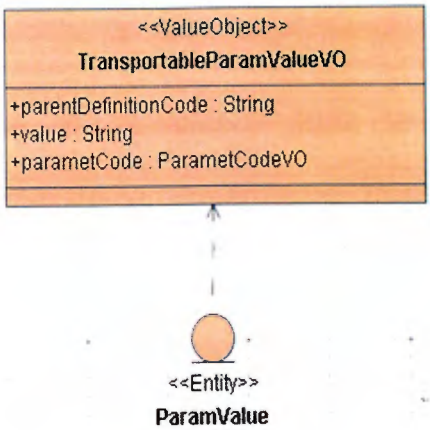

Figura 12. VOs de transporte para encapsular parámetros y sus valores

Fuente: Elaboración Propia

TransportDocumentVO: esta estructura es la que define los archivos de transporte y respaldo (cada una ensamblada de manera particular). Dentro de cada archivo de transporte o respaldo se encuentra una estructura de este tipo con todo el contenido a transportar o restaurar.

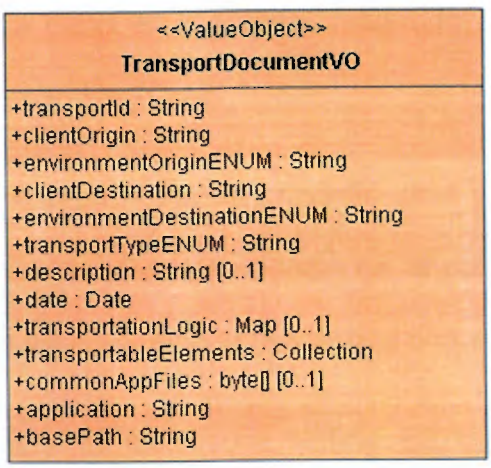

Figura 13. Estructura de un documento de transporte o respaldo

Fuente: Elaboración Propia

HistoricalTransportVO: este VO almacena toda la información histórica relacionada con los procesos de versionamiento, transporte, actualización y respaldo.

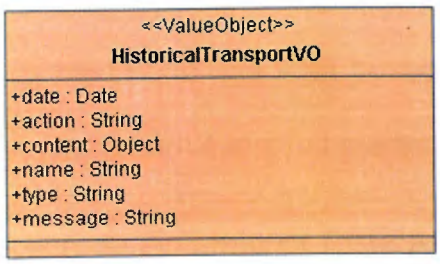

Figura 14. Estructura de un evento histórico de la aplicación de transporte

Fuente: Elaboración Propia

Entregables Obtenidos:

1. Diseño de la solución (incluye modelo de datos y pantallas).

IV.2.2.2 Construcción

La construcción se llevó a cabo en 3 etapas, refinando el prototipo según lo acordado utilizando el método MoSCoW.

Dos nuevos proyectos fueron creados. Un proyecto Java y un proyecto Flex. El proyecto Java maneja toda la lógica de negocio de la aplicación, validaciones globales y persistencia en ambos engines (relacional y orientado a objeto).

El proyecto en flex analiza la estructura, valida los campos y genera las pantallas.

IV.2.2.2.1 Módulo de versionamiento

Incorpora los servicios de consulta de la data relacional, construcción de las estructuras de Versionamiento (TransportableElementVO) y persistencia de versiones sobre la base de datos orientada a objetos.

El primer gran reto es como identificar de manera automática que es lo que necesita una vista. Si bien es cierto que una vista posee funciones, catálogos, edgs, parámetros y subvistas, saber con exactitud cuales, resulta complicado.

Haciendo uso el API Reflection de ActionScript (este API permite recorrer un objeto sin necesidad de conocer sus atributos), fue posible cargar la configuración de una vista, recorrer campo por campo analizando las propiedades del VO, que listan todos los componentes y subestructuras de esa vista en busca de atributos "inspeccionables" y filtrando aquellos de interés poder identificar aquellos de interés.

De esta manera a través de la configuración de la vista fue posible conocer que catálogos, parámetros, edgs y vistas posee cada vista.

En el caso de las funciones, el modelo de datos de funciones (desde donde se extrae la información del workflow) es posible consultar que vistas hacen uso de que funciones.

Es así como identificamos las dependencias de los objetos a versionar. Simplemente queda por consultarlos y ensamblar las estructuras necesarias.

Una vez se conocen las dependencias del objeto a versionar se hace una búsqueda sobre cuáles de esas dependencias se encuentran versionadas dándole la posibilidad al usuario de elegir si desea versionar todos los elementos o si quiere incluir alguno que ya está previamente versionado como dependencia de alguien más.

IV.2.2.2.2 Módulo de creación de transporte

La creación del transporte se origina en la selección de elementos versionados (vistas o catálogos, y sus correspondientes dependencias). Una vez los elementos versionados a incluir en un transporte son seleccionados, se configura el archivo a generar y se crea el transporte.

IV.2.2.2.3 Módulo de configuración de transporte

Definir el tipo de transporte, total o incremental, identificar el transporte, etc. Es el encargado principalmente de identificar los datos a manipular y según la lógica de negocio establecer los encabezados de las estructuras de datos de transporte y respaldo.

IV.2.2.2.4 Módulo de monitoreo de transporte

Módulo de monitoreo de Adobe Flex (pantalla) desplegado durante ejecución de pantallas. Proporciona información sobre procesos de importancia para el conocimiento del usuario y relacionado con funciones y métodos invocados Actionscript.

Módulo de monitoreo de java (basado en LOG4J) desplegado en JBOSS, proporciona información sobre procesos de consulta y persistencia de data sobre la base de datos relacional y sobre la base de datos orientada a objetos.

IV.2.2.2.5 Módulo de control de transporte

El módulo de control de transporte corresponde a un conjunto de servicios Java y ActionScript que permiten hacer las validaciones necesarias para consultar la data, ensamblar las estructuras de datos, guardarlas sobre base de datos orientada a objetos de transporte y posteriormente sobre la relacional.

Este módulo de control de transporte adicionalmente proporciona las funcionalidades de lectura y escritura de los archivos de transporte y respaldo, analiza las estructuras de datos en busca de dependencias, valida la lógica de restauración y actualización.

Entregables Obtenidos:

1. Aplicación de transporte (incluye los módulos de versionamiento, transporte, monitoreo, control y configuración).
2. Documentación de usuario (manual).

IV.2.3 Implementación

La implementación de la aplicación se divide en dos etapas. Instalación y configuración.

IV.2.3.1 Instalación

El proceso de instalación implica instalar todas las herramientas necesarias previamente (servidor de aplicaciones JBOSS, el módulo de BlazeDS que

permite la conexión remota a través de web con los servicios de java, framework TRIAD como proyecto J2EE).

La instalación de la aplicación de transporte está embebida dentro de la instalación del framework TRIAD dentro de la carpeta deploy del servidor de aplicaciones JBOSS. Hace falta actualizar esta carpeta (aunque con la actualización de las librerías common y core internas de TRIAD debería ser suficiente).

IV.2.2.2 Configuración

El proceso de configuración se lleva a cabo en diferentes puntos del sistema. Configuración de JBOSS, configuración de TRIAD y configuración de la aplicación de transporte.

Configuración del JBOSS. La ejecución de JBoss es relativamente sencilla, dentro del directorio bin de la instalación de JBoss se encuentran los archivos de arranque en forma de "scripts" para Shell. Los parámetros de ejecución están ubicados en el directorioserver/default/conf, el archivo run.bat ejecuta la aplicación para ambiente Windows.

Dentro de los parámetros se incluyen: bases de datos para trabajar con JBoss, ubicación de registros, parámetros JNDI, EJB's disponibles y cargados, entre otra información.

Configuración TRIAD. En la carpeta deploy del JBOSS se encuentra un archivo llamado triad-ds.xml. Este archivo especifica el recurso de persistencia que utilizará la aplicación triad. Especifica la ruta de la base de

datos que para efectos de la implementación será el servidor cinetap01 en el puerto 1521, el nombre de usuario y contraseña para la conexión con esa base de datos.

Adicionalmente debemos configurar la aplicación de transporte. Dentro de la carpeta del proyecto triad en el JBOSS se encuentran los módulos web como se especificó en la sección de marco teórico en donde un directorio WAR contiene los elementos web de todos los módulos de la aplicación. Dentro de la carpeta assets del módulo de transporte se encuentra un directorio llamado configurations. Este directorio posee los archivos de configuración de la aplicación. Estos archivos de configuración definen una ruta URL de despliegue que a través de la cual se solicitan las vistas del módulo correspondiente. Para efectos de implementación la url es orientada hacia el servidor cinetap01 en el puerto 8280.

Finalmente la ruta base obtenida será del tipo **http://cinetap01:8280/blazedsGCPTriad/**

Entregables Obtenidos:

1. Aplicación de transporte desplegada.
2. Manual de desarrollador.
3. Documento de revisión de proyecto, se evalúa el proyecto y el producto según métricas de calidad definidas por la empresa como son: funcionalidad, tiempo de respuesta, etc.
4. Curso instructivo a los desarrolladores sobre uso de la herramienta.

IV.3 Fase post-proyecto

Existen una gama de funcionalidades y opciones de transporte que planean incorporarse como extensión del alcance del proyecto aquí planteado. Esas nuevas funcionalidades implican desarrollos sobre la estructura de programación ya existente.

Esas consideraciones son tomadas en esta fase generando recomendaciones y documentos que agilizan futuros desarrollos.

Adicionalmente esta fase se encarga de evaluar el desempeño del equipo de trabajo y la evolución del proyecto.

Si bien existieron muchos cambios sobre la marcha del proyecto, la gestión de cambio ágil proporcionada por la metodología permitió una rápida respuesta con un mínimo de impacto sobre los elementos ya desarrollados.

Las reuniones constituyeron un factor importante en el desarrollo del proyecto debido a que permitieron a todo el equipo de proyecto moverse como una unidad, estar al tanto del estatus del desarrollo y corregir fallas y proporcionar una lluvia de ideas que no solo agilizan el desarrollo, sino que además, acompañado por la amplia experiencia del equipo de trabajo en sus respectivas áreas, optimizar el desarrollo en cuanto a código, mejorar las vistas, etc.

Un plan de mantenimiento ya está en implementación. Que permite a la aplicación de transporte incorporar elementos adicionales al proceso, ajustar el manejo de dependencias, optimizar los tiempos de respuesta, etc.

CAPÍTULO V. Resultados

TRIAD cuenta ahora con una aplicación de transporte con interfaz gráfica amigable, que le permite crear, configurar, monitorear y controlar el transporte de configuraciones entre los diferentes ambientes operativos y los diferentes clientes.

Esta aplicación de transporte está compuesta por:

Una pantalla de versionamiento, que apoyado en servicios desarrollados en Java y Actionscript provee los mecanismos para versionar catálogos y vistas (incluyendo todas sus dependencias) tomando en cuenta los diferentes ambientes operativos y de clientes. Estas versiones provienen de data obtenida a partir de la base de datos relacional de TRIAD y es guardada en la base de datos orientada a objetos del transporte (repositorio de transporte).

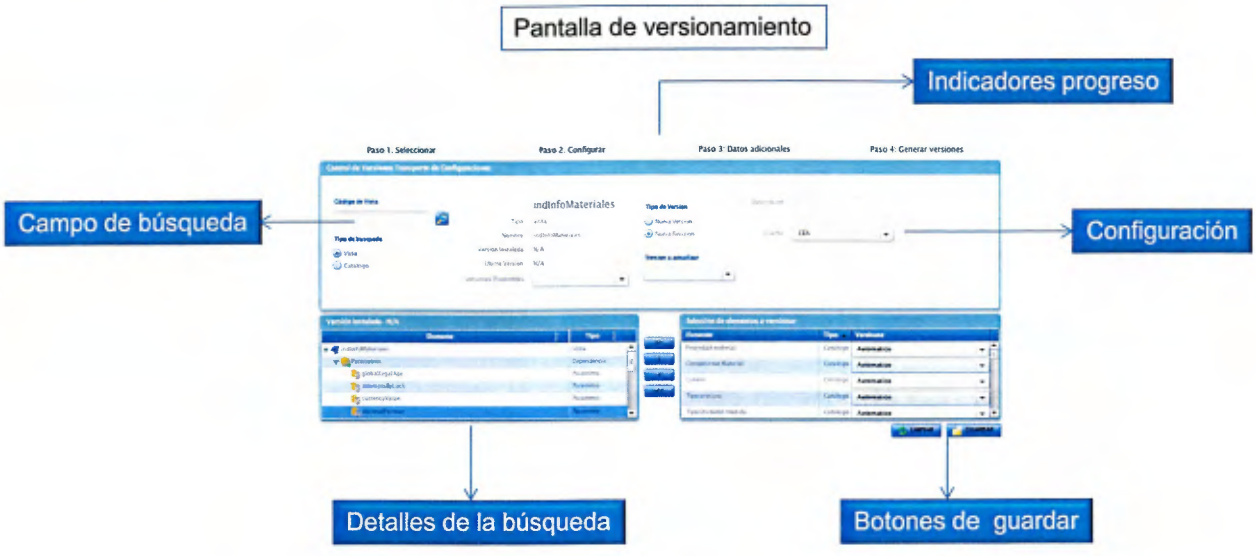

Figura 15. Pantalla de versionamiento

Fuente: Elaboración propia

Una pantalla de historial de transporte que permite visualizar el registro histórico de todas las operaciones locales de transporte realizadas (versionamientos, transportes, actualizaciones, restauraciones y creación de respaldos).

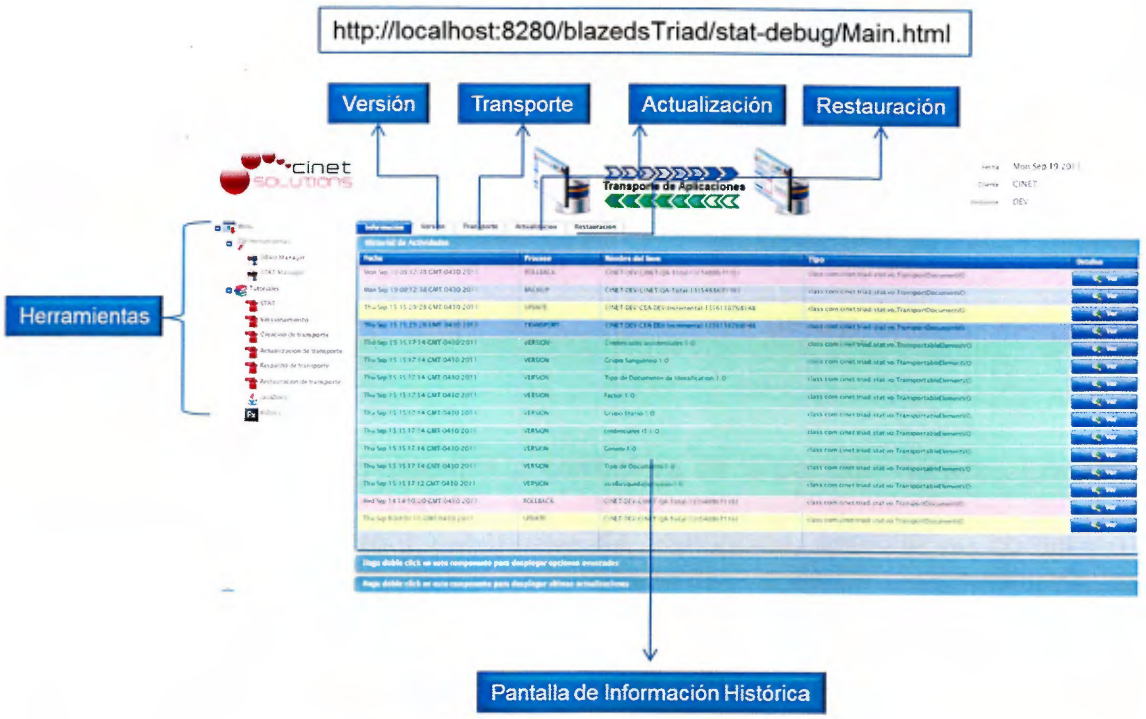

Figura 16. Pantalla principal de la aplicación (Información)

Fuente: Elaboración propia

La imagen a continuación muestra la pantalla de configuración y creación de transporte. A través de esta pantalla se configura en primera instancia la lógica de transporte/actualización a través de la selección de los elementos ya versionados a incluir, se identifica el documento de transporte con data relativa al origen, destino, aplicación y descripción. Finalmente se genera un documento de transporte sobre la ruta **/bin/Transports/** del directorio del JBOSS

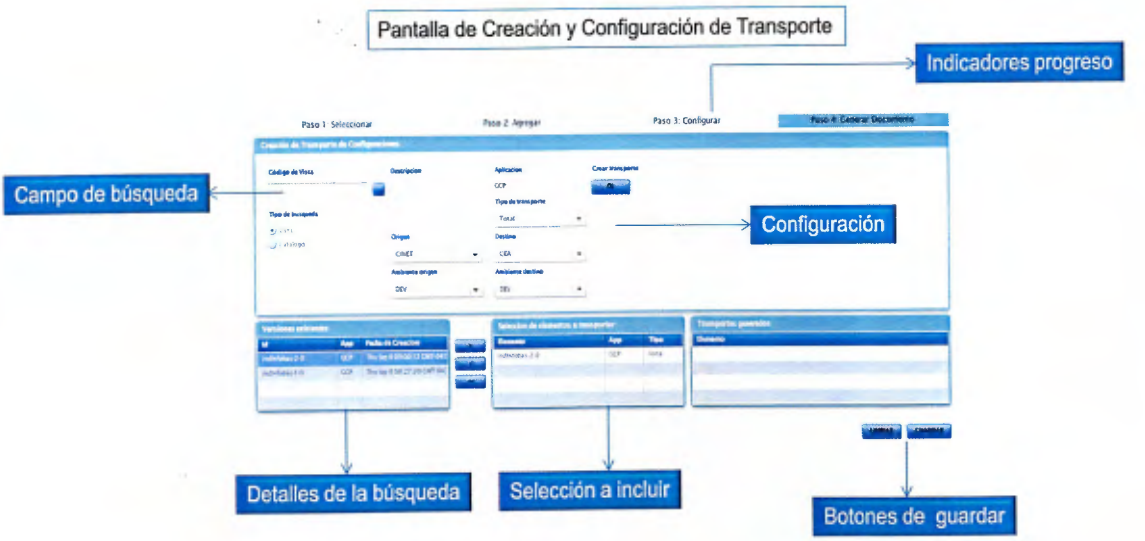

Figura 17. Pantalla de configuración de transporte

Fuente: Elaboración propia

La imagen a continuación muestra la estructura de la pantalla de actualización. Esta pantalla permite cargar un archivo de transporte, analizar su contenido, mostrar una comparación del contenido del transporte con lo que existe actualmente a nivel local y copiar el contenido del documento sobre la base de datos local.

Al momento de realizar una actualización la pantalla permite generar un archivo de transporte de respaldo según la data que va a ser alterada de acuerdo al archivo de transporte.

Estos archivos de respaldo en esencia son archivos de transporte autogenerados sin estructura jerárquica (no se guarda una dependencia relativa entre componentes, sino que todos los componentes necesarios son copiados independientemente) y son almacenados en la carpeta **/bin/Transports/Backups** con el mismo ID del documento de transporte.

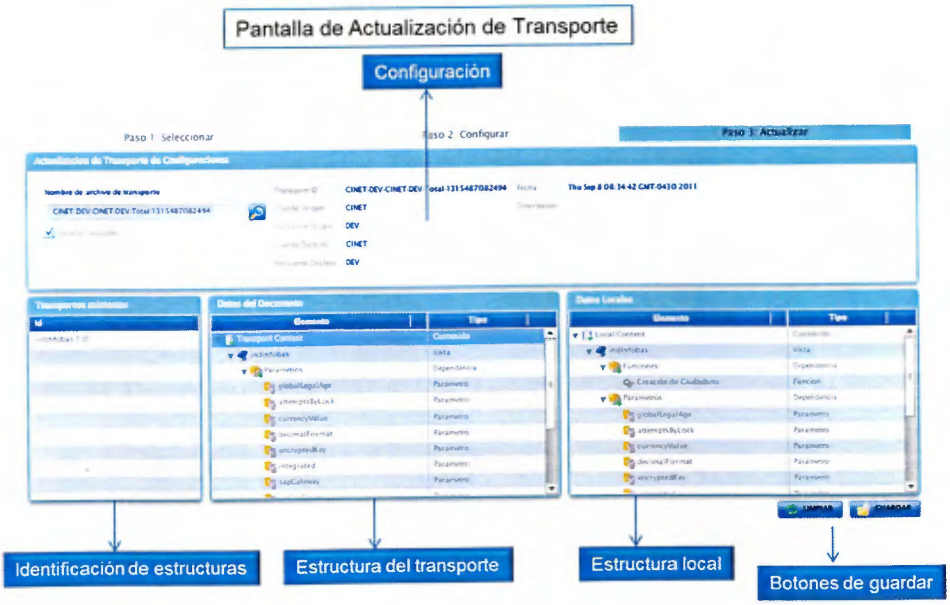

Figura 18. Pantalla de actualización de transporte

Fuente: Elaboración propia

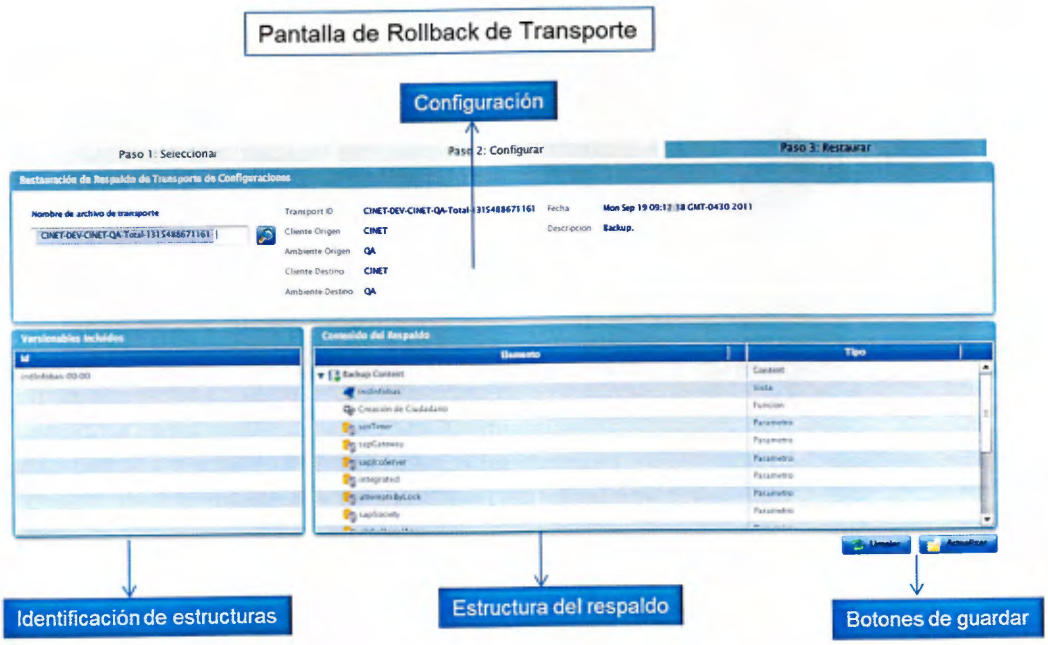

Figura 19. Pantalla de restauración de transporte

Fuente: Elaboración propia

El módulo de monitoreo como se explicó en puntos anteriores trabaja desde 2 locaciones. La primera, la pantalla de ejecución del JBOSS muestra los logs de los procesos de las aplicaciones JAVA. Aquí, todos los servicios y procesos de consulta, validación y almacenamiento sobre la base de datos relacional y la orientada a objetos son monitoreados en tiempo real con mensajes que especifican el Bean que contiene los servicios, el timestamp y detalles de la ejecución.

El segundo lugar donde se monitorea la ejecución de la aplicación de transporte es a través de un módulo visual llamado por las vistas de la aplicación. Cada vez que se realiza un proceso, el módulo de monitoreo de las vistas despliegan el proceso siendo ejecutado en tiempo real junto con una barra de progreso de tipo indeterminado.

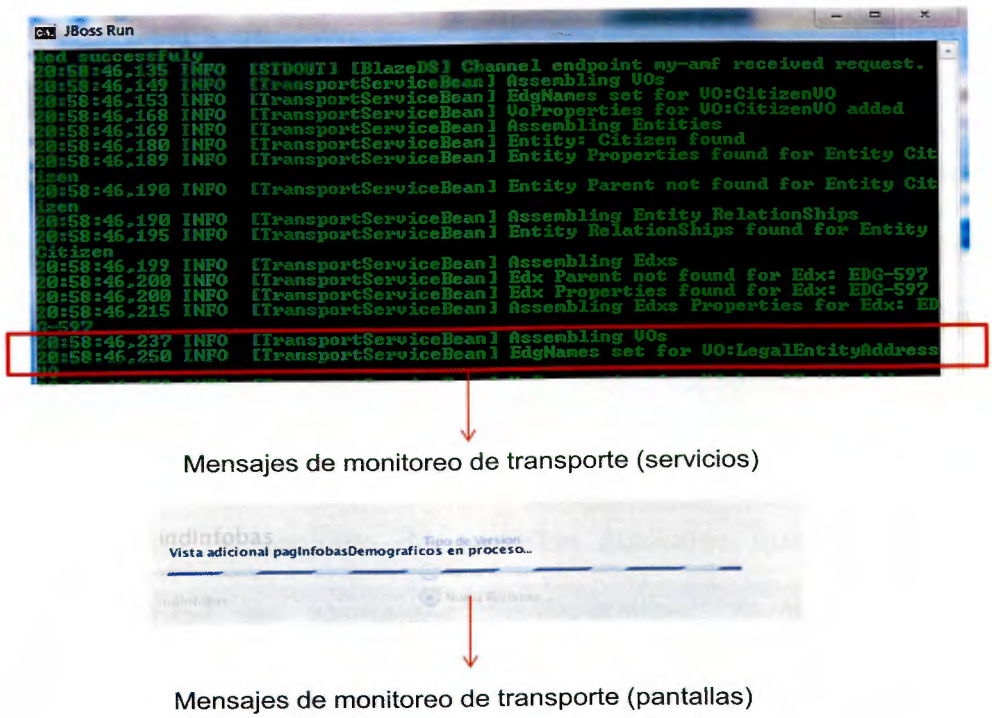

Figura 20. Pantallas de monitoreo de transporte

Fuente: Elaboración propia

Para todas estas pantallas, existen validaciones y lógica de negocio que no han de pasar desapercibidas. Varios Beans de Java proveen los servicios de control de transporte que regulan y validan los procedimientos de versionado, configuración y creación de transporte, actualización, generación de respaldo, restauración, historiales, etc.

Entre ellos se encuentra el TransportServiceBean que maneja la lógica de todos los procedimientos de consulta y actualización de data sobre las bases de datos relacionales y la orientada a objetos del transporte.

El TransportConfigurationService facilita la manipulación de los archivos externos y el manejo de historiales de transporte.

El TransportControlService provee todos los servicios que validan las operaciones, estructura de los objetos, etc.

Todos estos Beans de Java se apoyan en funciones y procedimientos de Actionscript ejecutados desde las pantallas para agilizar sus tareas.

Adicionalmente a la documentación generada al utilizar la metodología DSDM (prototipos funcionales, de diseño, prototipos de implementación y documentación del proceso), se generó documentación de tipo JavaDoc para los servicios de la aplicación en Java y ASDocs como documentación de la aplicación visual.

Existen, además, un conjunto de pantallas auxiliares que permiten ver en detalle estructuras de elementos transportables, parámetros, comparar versiones etc. A continuación se muestran algunas de ellas.

Pantalla de detalles de elementos transportables

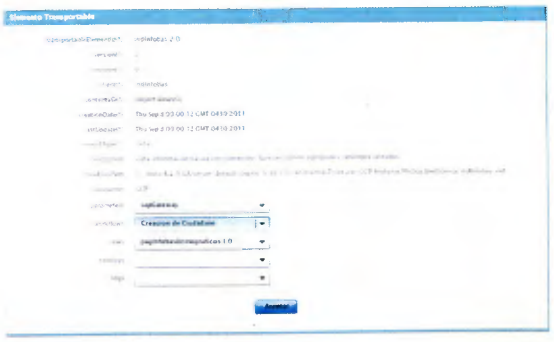

Figura 21. Pantalla de detalles de elementos transportables

Fuente: Elaboración propia

Pantalla de detalles de versiones

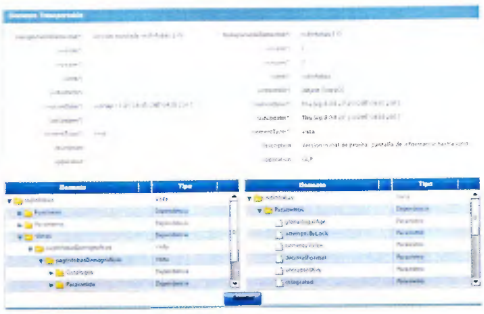

Figura 22. Pantalla de detalles de versiones

Fuente: Elaboración propia

El sistema de transporte de aplicaciones cuenta adicionalmente con video tutoriales que pueden ser llamados desde la misma aplicación, agregándole así, mayor facilidad a los usuarios a la hora de su utilización.

El peso de los archivos de transporte es directamente proporcional a su contenido. Pueden ir desde unos pocos kilobytes (transporte de catálogo individual) hasta varios megas (15 Megabytes ocupa en promedio el transporte de una vista simple).

El tiempo de procesamiento durante la actualización de datos en el destino puede tomar entre 30 segundos y 5 minutos dependiendo de la complejidad del contenido a restaurar, ya que existen muchas validaciones por cada proceso y subcomponente siendo restaurado.

CAPÍTULO VI. Conclusiones y recomendaciones

VI.1 Conclusiones

Los siguientes puntos recogen las conclusiones a las que se llegaron directamente tras cumplir con los objetivos iniciales.

La integración de la aplicación de transporte a la estructura del Framework TRIAD permite a esta herramienta funcionalidades que van más allá del simple hecho de crear archivos de transporte. Muchas otras utilidades pueden ser mencionadas como el Versionamiento de pantallas, que facilita la organización de las actividades durante el ciclo de vida de las aplicaciones. Creación de respaldos y transportes personalizados. Permite visualmente identificar las propiedades de las vistas actuales en el sistema, compararlas con contenido de un archivo de transporte, etc.

Durante la implementación se realizaron pruebas de transporte con contenido de actualización y se observó que comparado al anterior procedimiento de implementación de nuevas funcionalidades en el destino, es mucho menos problemático (al transportar solo lo que hace falta y no tener que modificar contenido adicional), es independiente al operador (anteriormente solo los desarrolladores de las pantallas conocían su contenido y debían listar las dependencias a la hora de llevarlas a un cliente) ahora es automático.

Agiliza la gestión de cambios de la empresa al poder optimizar el soporte que se la da a los clientes para el desarrollo de nuevas funcionalidades y ajustes de las ya existentes.

La arquitectura de la aplicación de transporte abre nuevas posibilidades tecnológicas a los desarrollos futuros de la empresa. Por un lado, el uso de la base de datos orientada a objetos otorga ventajas de procesamiento (muy ligero) y portabilidad, lo que la hace rentable a la hora de desarrollar módulos y tecnologías sobre las aplicaciones que requieran movilidad o rápido despliegue. Durante el desarrollo se vio la posibilidad de migrar todos los modelos de configuraciones que maneja el framework TRIAD a un sistema de base de datos orientada a objetos.

Para poder utilizar la aplicación de transporte de TRIAD es necesario implementar un plan de gestión que facilite y administre el Versionamiento de componentes de la misma. Definiendo los lineamientos a través del cual se crea una nueva versión o revisión de un componente, documentar esos cambios y llevar un control de las diferencias entre las versiones a fin de optimizar los versionamientos manteniendo la integridad de compatibilidad entre sus componentes.

Actualmente existe un manejador embebido en eclipse llamado OME (Object Manager Enterprise) pero sus capacidades son bien limitadas. Esto entorpece el desarrollo ya que se utilizan servicios y utilidades internas adaptadas al desarrollo del proyecto, para poder consultar la data de la base de datos,

analizar los objetos que almacena, ver su contenido, agregar, eliminar o editar esos objetos, etc. Al implementar una herramienta que facilite esas actividades, se agilizarán los procesos de desarrollo relacionados con la aplicación de transporte y se contará con un punto a favor sobre la robustez del sistema en general.

VI.2 Recomendaciones

En las siguientes líneas se presentan las recomendaciones que se tienen luego de culminar este proyecto.

Futuros desarrollos podrían implicar el transporte de otros tipos de configuraciones como lo son los roles de usuario, nuevas parametrizaciones no previstas en los modelos actuales, etc. Para ello debe crearse un plan de desarrollo que se adapte al plan de mantenimiento (que forma parte de la documentación de este proyecto). Tomando en cuenta la arquitectura de la aplicación, la lógica de desarrollo, etc. a fin de estandarizar el código de la aplicación, mantenerlo limpio y ordenado.

Por otro lado, es importante contar con herramientas externas o desarrolladas por la misma empresa, que faciliten los procesos de debugging sobre la data de la base de datos orientada a objetos.

Otro elemento importante a analizar es la necesidad en un futuro cercano de no solo incluir los elementos funcionales de la aplicación en el transporte (vistas, catálogos, etc) sino también el código fuente de las mismas y de los servicios que utilizan las funciones que posee. Para esto es necesario de alguna manera tener acceso a versiones de repositorios SVN de los proyectos J2EE (Java) y de Flex. Si estos códigos fuentes han de ser llevados a un destino cuyo repositorio es diferente, el sistema deberá implementar mecanismos que le permitan identificar las diferencias entre el código fuente

transportado y las últimas versiones de sus respectivos proyectos en el destino a fin de poder unificar la data y poderla actualizar en los repositorios destino.

TRIAD, podría, en un futuro crear un portal único a través del cual puedan accederse a todas su funcionalidades y se administren elementos como lo son la aplicación FDTS (administración de transacciones), Catalog Manager (administrador de catálogos), STAT (transporte de aplicaciones) y permita interactuar entre ellas (por ejemplo, poder crear un catálogo y versionarlo directamente desde una misma pantalla).

REFERENCIAS BIBLIOGRÁFICAS

1. **Sun's Official Java EE Tutorial** [Online]. **Recuperado el 27 de Mayo de 2011 de** http://download.oracle.com/javaee/5/tutorial/doc/

2. **JBoss EJB 3.0 Reference Documentation** [Online]. **Recuperado el 1 de Junio de 2011 de** http://download.oracle.com/javaee/5/tutorial/doc/

3. **Adobe Flex Documentation** [Online]. **Recuperado el 18 de Junio de 2011 de** http://www.adobe.com/support/documentation/en/flex/documentation.html

4. **Building and Deploying Flex Applications** [Online]. **Recuperado el 18 de Junio de 2011 de**

http://download.macromedia.com/pub/documentation/en/flex/2/flex2_buildanddeploy.pdf

5. **Programming Actionscript 3.0** [Online]. **Recuperado el 18 de Junio de 2011 de** http://livedocs.adobe.com/flex/3/progAS_flex3.pdf

6. **J2EE applications on the JBoss 3.2.x Server (2004)** [Online]. **Recuperado el 27 de Mayo de 2011 de** http://docs.jboss.org/jbossas/getting_started/3.2/jbossj2ee.pdf

7. **DB4O Documentation** [Online]. **Recuperado el 13 de Junio de 2011 de** http://developer.db4o.com/Documentation.aspx

8. **DSDM in a bird´s eye review (2008)** [Online]. **Recuperado el 27 de Mayo de 2011 de** http://www.dsdm.org/academic/student-resources

9. **The J2EE Architect's Handbook (2004)** [Online]. **Recuperado el 15 de Julio de 2011 de** http://www.dvtpress.com/javaarch/J2EEArchitectsHandbookV0.8.5.pdf