PROYECTO DE GRADO
FEDERADOR LINK-ALL

Informe de Proyecto de Grado

Mayo 2005

Instituto de Computación - Facultad de Ingeniería
Universidad de la República

Estudiantes: Fabricio Alvarez
Rodolfo Amador

Tutores: Federico Piedrabuena
Raúl Ruggia

Resumen

Este proyecto brinda los mecanismos de federación necesarios a un proyecto de mayor porte denominado Link-ALL, el cual consiste de Sistemas de Información Federados a través del Web. Uno de los principales objetivos es asistir al usuario con herramientas que le otorguen transparencia en el acceso a datos y servicios distribuidos, independientemente de la ubicación física de los mismos.

La realidad del proyecto Link-ALL involucra a diferentes comunidades, de distintas zonas geográficas y variadas realidades tecnológicas. Se busca conformar un marco de cooperación mutua entre los actores. Es necesario entonces el desarrollo de software que permita la implementación de dicho marco, donde los usuarios puedan compartir datos, brindar, utilizar servicios e interoperar en un entorno de seguridad apropiado.

La solución propuesta comprende una aplicación integrada a la plataforma Link-ALL, que mediante la comunicación entre servidores utilizando Web Services, posibilita la gestión y monitoreo de la federación de nodos, servicios y datos. Entre sus funcionalidades se destaca la capacidad de administrar la configuración de cada servidor de la red Link-ALL, rutear el acceso a datos y servicios en base a una identificación lógica y visualizar el estado actual de la federación tanto de forma local como remota. Esta solución se desarrolla sobre la plataforma J2EE y se basa en estándares abiertos de la industria. Fue construida basándose en patrones de diseño y utilizando herramientas de código abierto para así lograr una aplicación flexible, fácil de mantener y extender.

Palabras Clave:
Federación, Link-ALL, Web Services, J2EE.

Tabla de contenido

1. Introducción

1.1. Contexto

Este proyecto se encuentra en un contexto de Sistemas de Información Federados a través del Web llamado Link-ALL, Local-communities Insertion NetworK para América Latina [1]. Dicho proyecto está dirigido a asistir a tres sectores objetivo Artesanía, Eco-Agro Turismo y Cultura en la adquisición de prácticas de tecnologías de la información (IT) innovadoras, además pretende apoyar la colaboración, promover el intercambio de experiencias y la transferencia de conocimiento entre los sectores citados anteriormente. Link-ALL está financiado por el Programa @lis y apoyado por la Oficina de Cooperación Europe-Aid de la Comisión Europea. Entre sus socios y coordinadores se encuentran diferentes instituciones y empresas de Latinoamérica y Europa, uno de los cuales es el Instituto de Computación de la Facultad de Ingeniería.

La plataforma Link-ALL incluye herramientas IT diseñadas e integradas para proveer de una serie de facilidades claves de inclusión electrónica, que fortalezcan la integración de actividades de desarrollo local. La misma incorpora facilidades de comunicación, entre las que se destacan la cobertura satelital y soporte de aplicaciones inalámbricas. Entre otros aspectos incluye actividades de capacitación, soporte y la elaboración de un marco de negocios adecuado.

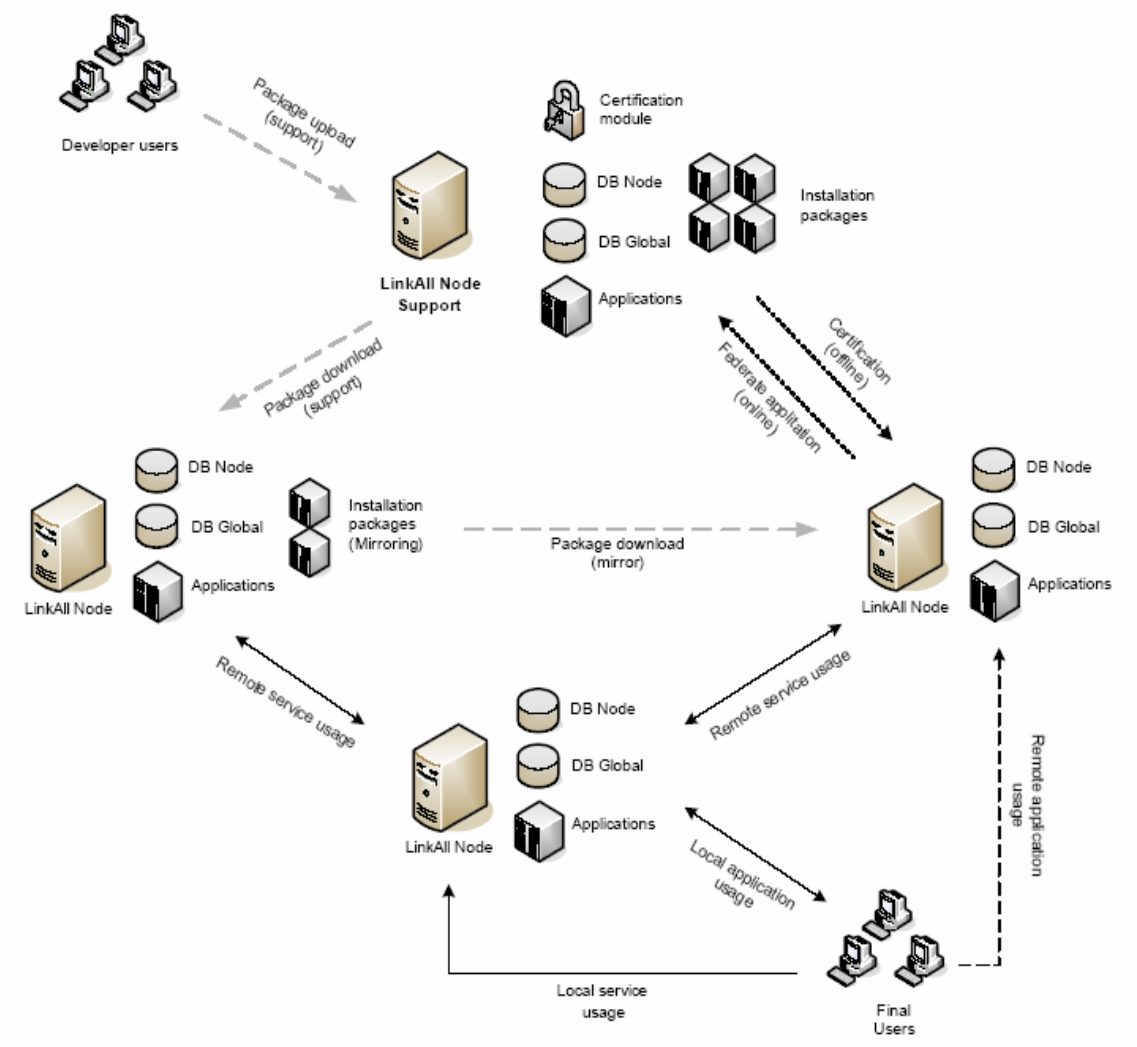


Fig. 1.1 - Red Link-ALL.

Los actores que forman parte de los sectores objetivo poseen realidades tecnológicas heterogéneas, algunos de ellos pertenecen a comunidades a las que se les hace difícil incorporar tecnologías de la información, por razones tales como aislamiento geográfico, recursos económicos escasos o infraestructura inadecuada.

Se pretende establecer un ámbito de colaboración mutua entre actores para subsanar las diferencias existentes. Para conformar este entorno es necesario el desarrollo de software que permita a los usuarios compartir y acceder a datos ubicados en otros servidores, invocar u ofrecer servicios remotos operando de forma transparente, en un marco de seguridad adecuado. Para hacer esto posible es necesario construir una Federación de Servidores Link-ALL.

La Federación Link-ALL está formada por un conjunto de servidores con autonomía propia de ejecución, gestionados de forma independiente, que cooperan entre sí a través del Web compartiendo aplicaciones y datos. El Federador es el software que provee una manipulación coordinada y controlada de los componentes de la red Link-ALL. Para la implementación del mismo se pueden distinguir dos componentes principales que permiten realizar por un lado la federación de datos, y por otro la federación de aplicaciones. Además se debe proveer de herramientas que permitan monitorear el estado de la federación.

El Federador Link-ALL debe poseer autonomía de ejecución, es decir tener la habilidad de ejecutar operaciones locales (operaciones solicitadas por un usuario local del servidor) sin interferencia de operaciones externas (operaciones enviadas por otros servidores de la federación). Las operaciones externas deben ser tratadas de la misma forma que las operaciones locales. Asimismo debe tener autonomía de asociación es decir la capacidad de un servidor de asociarse y desasociarse de la federación, cuando el administrador del nodo lo crea conveniente.

En lo que se refiere a la federación de datos, todos los servidores de la red Link-ALL poseen el mismo esquema y manejan el mismo tipo de datos, no existiendo heterogeneidad semántica. Además se utiliza igual Sistema de Gestión de Base de Datos (DBMS) e igual lenguaje de consulta. Existe la posibilidad de definir políticas de seguridad sobre los datos, donde un usuario puede administrar permisos sobre sus datos locales y a su vez solicitar permisos sobre datos de otras comunidades. El Federador de Datos debe proporcionar una Interfaz de Programación (API), que es utilizada por las aplicaciones instaladas en el servidor para acceder a los datos globales de Link-ALL.

El Federador de Aplicaciones es el responsable de brindar acceso transparente a las aplicaciones de la red Link-ALL, por lo tanto el usuario podrá invocarlas desconociendo el servidor donde se encuentran instaladas. Una importante característica que debe brindar este componente es la capacidad de exponer o no a la federación las aplicaciones instaladas en el nodo. Al momento de instalar y desinstalar una aplicación en un servidor el subsistema encargado de esta operación (Deployer) debe utilizar la API que brinda el Federador para dicho efecto. Igual que en el caso anterior el usuario debe ser capaz de definir políticas de seguridad sobre las aplicaciones, para restringir u otorgar el acceso a las mismas.

Para facilitar la administración de la federación es necesario suministrar herramientas que permitan realizar el monitoreo de la federación. Estas herramientas proporcionarán un mapa visual de la federación, mostrando servidores, comunidades y aplicaciones federadas.

1.2. Objetivos

Objetivos generales del proyecto Link-ALL:

- Proveer a comunidades locales la oportunidad de incorporarse a la sociedad de la información a través de un conjunto de facilidades, servicios y funcionalidades especialmente diseñados para PYMEs, organizaciones y actores locales claves en los 3 sectores seleccionados.

- Posibilitar y facilitar la inserción exitosa de comunidades remotas en el mercado global, a través del desarrollo de una red transnacional europeo-latinoamericana de cooperación entre actores artesanales, culturales y de eco-agro turismo.

- Ofrecer a actores de los sectores objetivo acceso a un amplio espectro de buenas prácticas en Tecnologías de la Información en Europa y América Latina y mejorar sus capacidades de colaboración conjunta.

Objetivos de este proyecto:

El principal objetivo del proyecto Federador Link-ALL es desarrollar funcionalidades que posibiliten a varios servidores compartir datos y servicios con transparencia de la ubicación física.

A continuación se detallan los objetivos más específicos del proyecto:

- Desarrollar utilitarios y componentes que se integren a la plataforma Link-ALL y permitan federar servidores, aplicaciones y datos.

- Definir, generar y administrar la información de configuración necesaria para implementar las conexiones entre un servidor y los otros.

- Rutear el acceso a datos o aplicaciones que se realicen en base a una identificación lógica, hacia los sitios físicos en donde se encuentran.

- Ofrecer herramientas para la visualización del estado de la federación.

- Brindar mecanismos que permitan establecer políticas de seguridad sobre los recursos de un servidor.

Resultados esperados

El resultado esperado es una herramienta de software que implemente los objetivos planteados. Además el componente desarrollado debe ser confiable, amigable, fácil de extender y mantener.

1.3. Conclusiones

En este proyecto se presenta la solución alcanzada al problema de la federación de aplicaciones y datos en la red Link-ALL lográndose desarrollar un producto de calidad que cumple con los objetivos planteados.

Fue definida una arquitectura en capas en donde la lógica se encuentra dividida en componentes de acuerdo a su función. En ésta definición se emplearon patrones de diseño que contribuyeron a brindarle al producto características como flexibilidad, extensibilidad y mantenibilidad.

La solución implementada consiste en un Federador de Aplicaciones y Datos integrado a la plataforma Link-ALL. Éste brinda los servicios necesarios para que el acceso a los recursos de la red sea transparente al usuario independientemente de la ubicación física de los mismos.

1.4. Organización del Documento

A continuación se describe la estructura de este documento. Dentro de cada capítulo se mencionan las secciones que son relevantes para su comprensión.

- **Capítulo 2 – Estado del Arte**

En este capítulo se presenta la investigación realizada sobre el tema federación y el estudio de las tecnologías utilizadas durante el desarrollo del proyecto. Se destaca la sección Federación.

- **Capítulo 3 – Presentación de la solución**

Se presenta la solución al problema planteado en la introducción. Se recomienda leer las secciones Consideraciones Generales y Federador Link-ALL. De la sección de funcionalidades dentro de Federador de Aplicaciones se recomienda leer Rutear Aplicación y de Federador de Datos Procesar Consulta.

- **Capítulo 4 – Implementación de la solución**

Se describe el diseño y la implementación de la solución. Dentro de la sección Diseño de Funcionalidades, Rutear Aplicación da una visión del funcionamiento del Federador de Aplicaciones y Procesar Consulta del Federador de Datos.

- **Capitulo 5 – Verificación del Sistema**

Se describen las pruebas realizadas y los resultados obtenidos.

- **Capítulo 6 – Conclusiones y Trabajo Futuro**

Se presentan las conclusiones y el trabajo futuro.

2. Estado del Arte

2.1. Federación

2.1.1. Federación de Datos

Un sistema de bases de datos federadas es una colección de sistemas de bases de datos que colaboran entre sí de una manera autónoma [3].

Algunas de las características que presentan las bases de datos federadas son la distribución de los datos, la heterogeneidad y la autonomía de las bases de datos componentes.

En general en los sistemas de bases de datos federadas la distribución de los datos se da porque la construcción de la federación es a partir de base de datos independientes que necesitaron compartir sus datos por alguna razón.

Los datos pueden estar distribuidos en múltiples bases de datos, las cuales pueden estar en el mismo sistema, en el mismo local o geográficamente separadas, pero interconectadas entre si por sistemas de comunicación. La distribución de los datos puede ser de diferentes maneras, por ejemplo partición horizontal y vertical de la base de datos. Algunos beneficios de la distribución de los datos son la disponibilidad, confiabilidad y la mejora en los tiempos de acceso. La distribución puede ser inducida, es decir, los datos pueden ser distribuidos deliberadamente para sacar ventaja de estos beneficios.

Muchos tipos de heterogeneidades son debidas a diferencias tecnológicas, por ejemplo, diferencias en hardware, software como ser sistemas operativos, y sistemas de comunicación. Los tipos de heterogeneidades en los sistemas de base de datos pueden ser divididos en aquellos que tienen diferencias respecto a la DBMS y aquellos que tienen diferencias en la semántica de los datos. Las heterogeneidades debido a diferencias en DBMSs pueden ser debido a diferencias en estructura, diferencias en restricciones o diferencias respecto a los lenguajes de consultas.

La heterogeneidad semántica ocurre cuando hay un desacuerdo respecto al significado, interpretación, o uso diferente de los mismos datos o datos relacionados.

Existen diferentes tipos de autonomía que pueden tener las bases de datos pertenecientes a una federación.

Autonomía de diseño refiere a la habilidad de un DBMS componente a elegir su propio diseño con respecto a cualquier materia incluyendo los datos a ser manejados, la representación (modelo de datos), la conceptualización o interpretación semántica de los datos, las restricciones, la funcionalidad del sistema.

Autonomía de comunicación refiere a la habilidad de un DBMS componente a decidir si se comunica con otro DBMS componente. Un componente con autonomía de comunicación puede decidir cuando y como responde a un pedido desde otro DBMS componente.

Autonomía de ejecución refiere a la habilidad de un DBMS componente para ejecutar operaciones locales (comandos o transacciones ingresadas directamente por un usuario local) sin interferencia de las operaciones externas (operaciones invocadas por otros DBMSs componentes) y decidir el orden en cual se ejecutarán las aplicaciones externas. Así, un sistema externo no puede forzar una orden o ejecución de los comandos en un DBMS componente con autonomía de ejecución.

Autonomía de asociación implica que un DBMS componente tiene la habilidad para decidir si comparte y cuánto comparte sus funcionalidades y sus recursos, esto incluye la habilidad para asociarse o desasociarse por sí mismo de la federación y la habilidad de un DBMS componente de participar en una o más federaciones.

2.1.2. Federación de aplicaciones

Durante el estudio del estado del arte, no se encontró material referente a la federación de aplicaciones, pero pueden ser aplicados aquí algunos de los conceptos descriptos anteriormente para la federación de datos. Por ejemplo se puede hacer una analogía entre servidores de aplicaciones y manejadores de bases de datos, entonces se pueden aplicar fácilmente los conceptos de autonomía de comunicación, autonomía de ejecución, autonomía de asociación, heterogeneidad en el hardware y heterogeneidad en el software.

2.2. Tecnologías

En esta sección se brinda un resumen del estudio de las tecnologías utilizadas, por más detalle referirse al anexo de Estado del Arte [18].

2.2.1. J2EE

La plataforma J2EE es una especificación que permite desarrollar sistemas y servicios con una arquitectura en capas a un bajo costo y de manera poco compleja, asegurando a las aplicaciones que cumplan dicha especificación algunas características como ser alta disponibilidad, seguridad, confiabilidad y escalabilidad. Brinda también flexibilidad a los sistemas implementados para adaptarse a los cambios requeridos de manera sencilla al permitir que las aplicaciones J2EE sean fácilmente instaladas y mejoradas [4].

J2EE alcanza estos beneficios al defender una arquitectura estándar con los siguientes elementos:

- Plataforma J2EE – Una plataforma estándar para mantener las aplicaciones J2EE.
- Test suite de Compatibilidad con J2EE – Una test suite para verificar que un producto J2EE cumple con el estándar de una plataforma J2EE.
- Implementación de Referencia J2EE – Una implementación de referencia para prototipar aplicaciones J2EE y para proveer una definición operacional de la plataforma J2EE.
- Diseños J2EE – Un conjunto de buenas prácticas para desarrollar servicios de múltiples capas y cliente finos.

La plataforma J2EE usa un modelo de aplicaciones distribuido en capas. La lógica de las aplicaciones se divide en componentes de acuerdo a su funcionalidad y estos componentes que forman una aplicación J2EE pueden estar instalados en diferentes máquinas dependiendo de la capa a la que pertenece.

2.2.2. Web Services

Un Web Service es un sistema de software identificado por una URI, del cual las conexiones y las interfaces públicas están definidas y descriptas usando XML. Su definición puede ser descubierta por otro sistema de software. Estos sistemas pueden interactuar con el Web Service de la manera indicada en su definición, usando mensajes basados en XML transportados por protocolos de Internet [5].

Los Web Services exponen funcionalidades útiles a los usuarios Web, a través de un protocolo Web estándar, que en la mayoría de los casos es SOAP.

Los Web Services proveen una manera de describir sus interfaces con el detalle suficiente para permitir a un usuario construir una aplicación cliente que se comunique con ellos. Esta descripción es usualmente provista por un documento XML llamado documento de Lenguaje de Descripción de Web Services (Web Services Description Language - WSDL).

Los Web Services son registrados en directorios para que potenciales usuarios puedan encontrarlos fácilmente. Esto está realizado con UDDI, (Universal Discovery, Description, and Integration). El UDDI puede ser visto como las páginas amarillas de los Web Services, provee de una base de datos donde se pueden realizar búsquedas por tipo de Web Services.

SOAP es un protocolo liviano para intercambio de información en un ambiente distribuido y descentralizado. Es un protocolo basado en XML que consta de tres partes, un sobre que define el marco para describir qué es un mensaje y como procesarlo, un conjunto de reglas para expresar instancia de tipos de datos definidos en una aplicación y una convención para representar llamadas y respuestas a procedimientos remotos.

2.2.2.1. AXIS

Axis es esencialmente un motor SOAP, un marco para construir procesadores SOAP como clientes, servidores y gateways [5]. La versión actual de Axis está escrita en Java y provee:

- Un servidor que funciona por si mismo.
- Un servidor el cual puede ser instalado en motores de Servlets como Tomcat.
- Soporte extensivo para el lenguaje de descripción del Web Service (WSDL).
- Herramientas que generan clases Java desde el WSDL.

2.2.3. Application Servers

2.2.3.1. JBoss

JBoss es un servidor de aplicaciones compatible con la plataforma J2EE, ampliamente utilizado en el mercado y Open Source [6]. El servidor y contenedor JBoss están implementados completamente por plugins basados en componentes. Este alto grado de modularidad permite agregar o quitar funcionalidades al servidor según las necesidades del usuario.

JBoss agrupa, integra y desarrolla los servicios que la plataforma J2EE involucra para una implementación completamente basada en J2EE [7]. Esto incluye:

- Un contenedor de EJBs (Enterprise Java Beans)
- JMS (Java Message Service)
- JTS/JTA (Java Transaction Service / Java Transaction API)
- Servlets y JSP (Java Server Pages)
- JNDI (Java Naming and Directory Interface)

Además provee otras características que incluyen la integración con Web Services.

3. Presentación de la solución.

La solución se compone de dos partes: el Federador Link-ALL y un utilitario de navegación de la federación. En las secciones siguientes se describe las características de ambos por separado.

3.1. Consideraciones generales

Un servidor Link-ALL está dividido en dos grandes regiones: una región donde los usuarios de las comunidades instalan las aplicaciones (nodo aplicativo, NAP) y otra región donde se ubican los componentes administrativos del servidor (nodo administrativo, NAD) [2]. El Federador Link-ALL forma parte del nodo administrativo, conjuntamente con otros componentes como ser los encargados del manejo de la seguridad, sesiones y base de datos del NAD del servidor Link-ALL.

Los usuarios de las comunidades del servidor podrán utilizar aplicaciones instaladas localmente en el nodo aplicativo o tal vez instaladas por otras comunidades en servidores remotos. De la misma forma debe ser posible consultar datos de varias comunidades accediendo de una forma unificada y transparente. El acceso a datos y aplicaciones debe ser controlado en un marco de seguridad acorde a la realidad actual teniendo en cuenta que no sólo se está tratando con información y servicios públicos sino también privados pertenecientes a los actores que forman parte de la comunidad virtual Link-ALL.

En el contexto descrito anteriormente se encuentran esbozadas las principales responsabilidades del Federador Link-ALL: la gestión de datos y aplicaciones de la federación de forma transparente a los usuarios.

En el Federador se pueden apreciar dos grandes componentes: el Federador de Aplicaciones y el Federador de Datos que se detallarán posteriormente. Ambos poseen dos modos de operación: local y remoto. En el modo local, el Federador recibe un contexto local, de donde se puede obtener los datos necesarios para certificar el emisor de la llamada (contiene un identificador de sesión válido). En el caso de un pedido remoto, el Federador recibe un contexto remoto, con el identificador del que realiza el pedido, la comunidad a la que pertenece y el certificado del nodo emisor. Para certificar al emisor del pedido, el Federador utiliza los servicios del nodo administrativo.

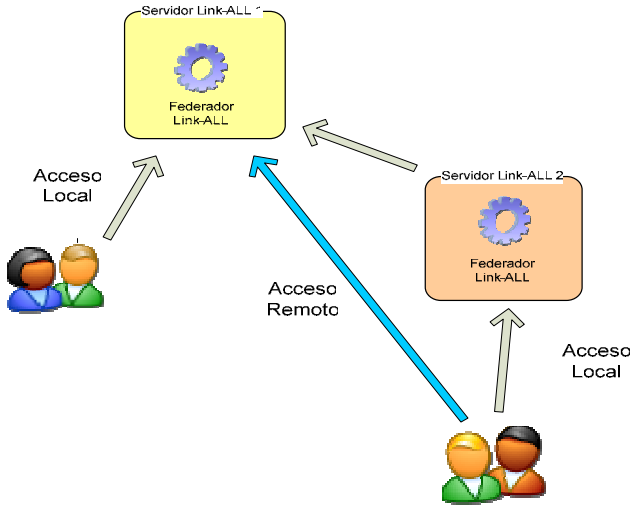


Fig. 3.1 - Modos de operación del Federador Link-ALL.

3.2. Federador Link-ALL

El Federador Link-ALL es una aplicación que permite la manipulación coordinada y controlada de los recursos de la red Link-ALL. Este componente es responsable de la federación de servidores, aplicaciones y datos. Los servidores federados podrán así compartir sus recursos (aplicaciones y datos) al resto de las comunidades de la federación, teniendo la facultad de definir políticas de seguridad sobre el acceso a los mismos.

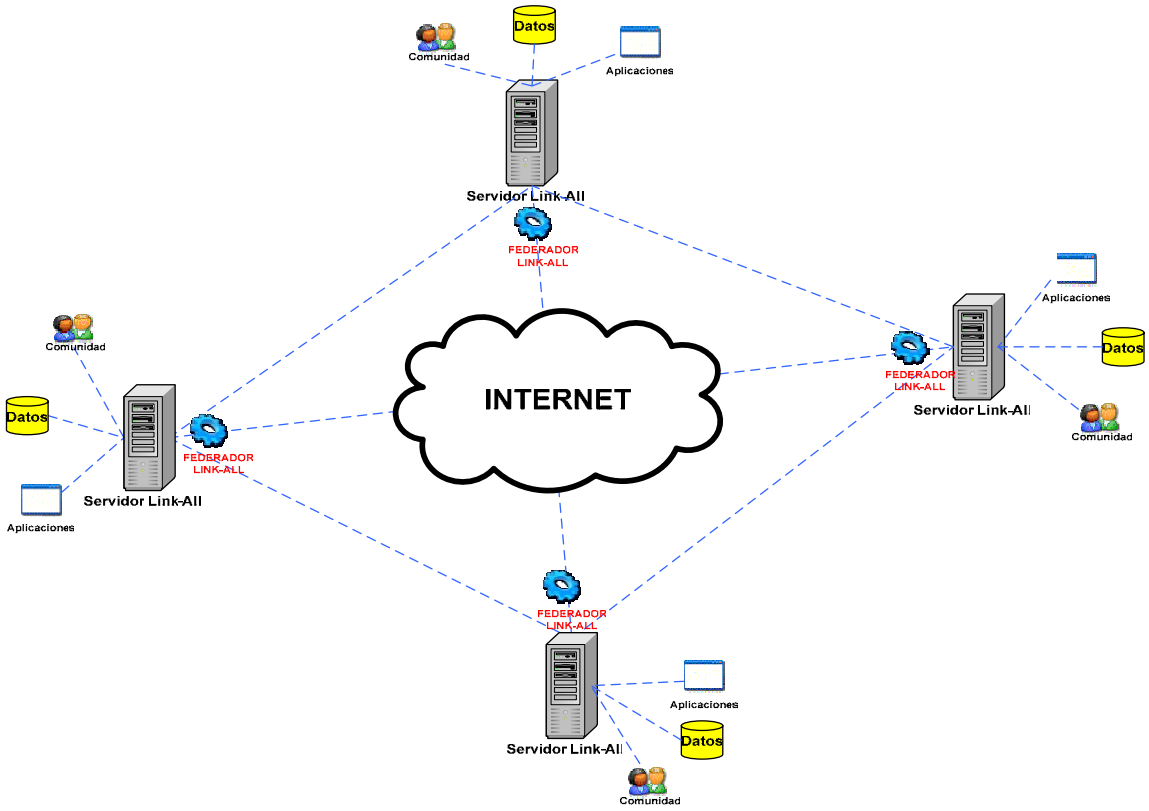


Fig. 3.2 - Federación Link-ALL, mostrando el rol que cumple el Federador en la misma.

3.2.1. Federador de Aplicaciones

El objetivo principal del Federador de Aplicaciones es proveer la ubicación de una cierta aplicación dado su identificador único. Todas las aplicaciones instaladas en el nodo aplicativo NAP, tendrán un identificador único en la red Link-ALL. Los usuarios del servidor Link-ALL podrán indicarle al Federador de Aplicaciones el código de la aplicación que desean y es éste componente quien ocultando la complejidad de la localización, devuelve la URL de la aplicación encontrada, siempre y cuando el usuario tenga permisos.

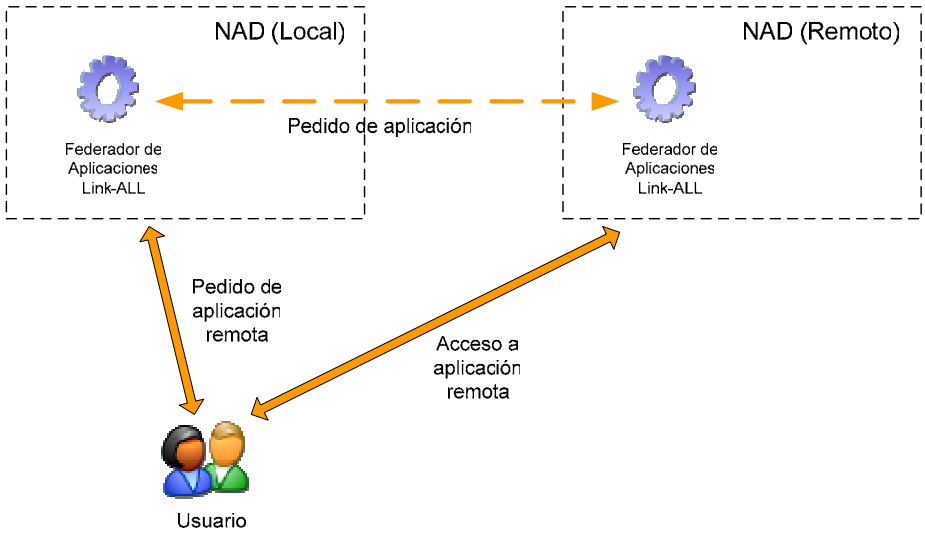


Fig. 3.3 - Acceso a aplicación remota.

3.2.1.1. Seguridad del Federador de Aplicaciones

El esquema de seguridad de Link-ALL está basado en el concepto de Usuarios, Grupos y Funcionalidades. Un usuario puede pertenecer a más de un grupo durante su tiempo de vida en el Sistema y a su vez un grupo puede estar relacionado a muchas funcionalidades. La pertenencia a un determinado grupo es lo que habilita a un usuario a ejecutar funcionalidades para las que el grupo tiene permisos.

En este escenario, el Federador de Aplicaciones implementa el control de acceso por grupos de seguridad definidos en la base de datos administrativa. Cada comunidad de la red Link-ALL podrá solicitar, otorgar o denegar permisos de ejecución de aplicaciones a grupos de comunidades remotas.

3.2.1.2. Funcionalidades

En los párrafos siguientes se brinda un detalle de las principales funcionalidades que brinda el Federador Link-ALL.

3.2.1.2.1. Rutear aplicación

Un usuario invoca la ejecución de una aplicación, el Federador rutea el pedido independientemente que la aplicación esté instalada en el servidor donde se hizo el pedido o no y devuelve la URL de la misma. Las aplicaciones pueden ser públicas o privadas, para que un usuario pueda acceder a una aplicación privada, éste deberá tener permisos para su ejecución.

Una vez que llega el pedido de ruteo de una aplicación al Federador Link-ALL, éste busca la aplicación entre las aplicaciones instaladas, si la encuentra entonces procede a verificar los permisos de ejecución que posee el usuario. Si la aplicación es pública o el usuario tiene permisos, el Federador devuelve la dirección de la misma.

En el caso de que el usuario no tenga permisos sobre la aplicación o ésta no se encuentre instalada en el nodo donde se realizó el pedido (aplicación remota), se buscan el o los servidores en los que el usuario puede acceder a la aplicación de acuerdo a los permisos remotos. Una vez localizado un servidor donde está instalada la aplicación, el Federador le solicita la creación de una sesión remota para el usuario, y retorna la dirección de la aplicación incluyendo la URL del servidor.

La figura 3.1 muestra un diagrama de secuencia que representa el funcionamiento de rutear aplicación.

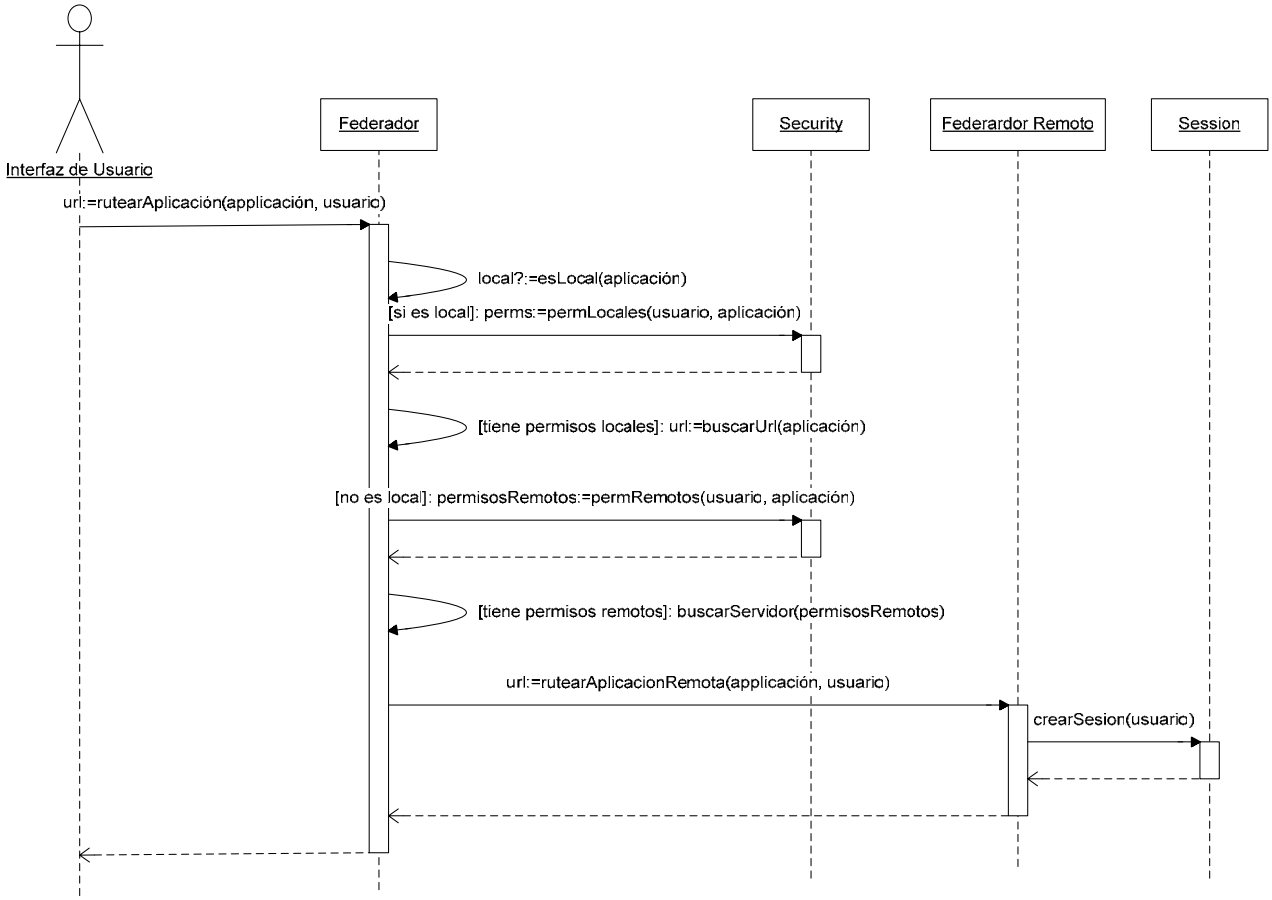


Figura 3.4 - Diagrama de secuencia representando el funcionamiento de Rutear aplicación.

3.2.1.2.2. Obtener aplicaciones del usuario

Un usuario ingresa al sistema y en ese momento se devuelve el perfil del mismo incluyendo las aplicaciones a las que tiene acceso. El Federador es el encargado de devolver las aplicaciones locales y remotas que el usuario puede ejecutar. Para encontrar las aplicaciones del usuario, el Federador busca los grupos a los que éste pertenece y a partir de ahí obtiene las funcionalidades que pertenecen a las aplicaciones locales que el usuario puede acceder. Luego busca si algún grupo del usuario tiene permisos sobre aplicaciones remotas, si es así a las aplicaciones locales se agregan las aplicaciones remotas que no estén entre las anteriores (la misma aplicación puede estar instalada en más de un servidor).

3.2.1.2.3. Federar y Desfederar Aplicación

Federar: Consiste en exportar o publicar una aplicación en la federación. Un usuario administrador de la configuración del Federador puede indicar si la federación de la aplicación es inmediata a la instalación de la misma por el componente Link-ALL encargado de tales funciones (Deployer). Si la federación no es inmediata, la aplicación quedará instalada en el Servidor y podrá ser federada manualmente por el administrador del nodo. La federación de la aplicación consiste básicamente en comunicar al servidor de soporte que la aplicación está federada y puede ser utilizada desde otros servidores, es decir se da de alta la misma en la topología de la red Link-ALL.

Desfederar: La operación consiste en dar de baja una determinada aplicación de la federación. Una vez dada de baja la aplicación no será “visible” para otros miembros de la federación. Dicha operación puede ser invocada por el Deployer al desinstalar una aplicación, o por el administrador del Servidor Link-ALL si no se desea seguir compartiéndola a la federación.

3.2.1.2.4. Solicitar, Otorgar y Denegar Permisos sobre aplicaciones

Solicitar: El administrador de una comunidad solicita permisos sobre aplicaciones federadas de otras comunidades. Una vez invocada ésta operación, el nodo solicitante realiza un pedido de permiso al nodo que tiene federada dicha aplicación. Si tiene éxito, la operación da como resultado una solicitud de permiso creada en ambos nodos en estado Pendiente. La solicitud conservará éste estado hasta que un administrador del nodo que posee la aplicación otorgue el permiso (como se detalla a continuación).

Otorgar: El administrador de una comunidad puede otorgar permisos sobre aplicaciones locales a comunidades remotas (comunidades del mismo servidor o comunidades de otros servidores de la federación).

El Federador permite a los administradores de comunidades dar permisos que pueden haber tenido una solicitud previa o no. En caso de no haber tenido una solicitud previa, es el administrador quien voluntariamente decide brindar el permiso, si existía una solicitud se podrá seleccionar dentro de los pendientes para decidir cual otorgar. Una vez realizada la adjudicación del permiso la comunidad receptora es informada del hecho quedando en ambos nodos el nuevo permiso creado en estado Aprobado.

Denegar: El administrador decide revocar los permisos otorgados a cierta comunidad para una determinada aplicación.

El Federador Link-ALL permite a los administradores de comunidades negar permisos o solicitudes de permisos que existan en el servidor. En todos los casos la comunidad a la que se le ha negado el permiso o la solicitud es informada del hecho, actualizándose los datos de su esquema de seguridad.

3.2.2. Federador de Datos

El objetivo principal del Federador de Datos es exponer una visión uniforme de la base de datos Global de Link-ALL. Esta base de datos se encuentra en cada uno de los nodos y mantiene información general de las diferentes comunidades del servidor. Los diferentes usuarios del sistema tienen acceso a la mencionada base de datos de acuerdo a los permisos que poseen.

El acceso a la base de datos Global es siempre realizado a través del Federador de Datos. Esto se debe a que la base de datos no tiene porque estar sólo instalada en un servidor de base de datos. El único que provee un esquema unificado de la base de datos Global es el Federador de Datos, el cual puede ser accedido por las aplicaciones instaladas en el NAP.

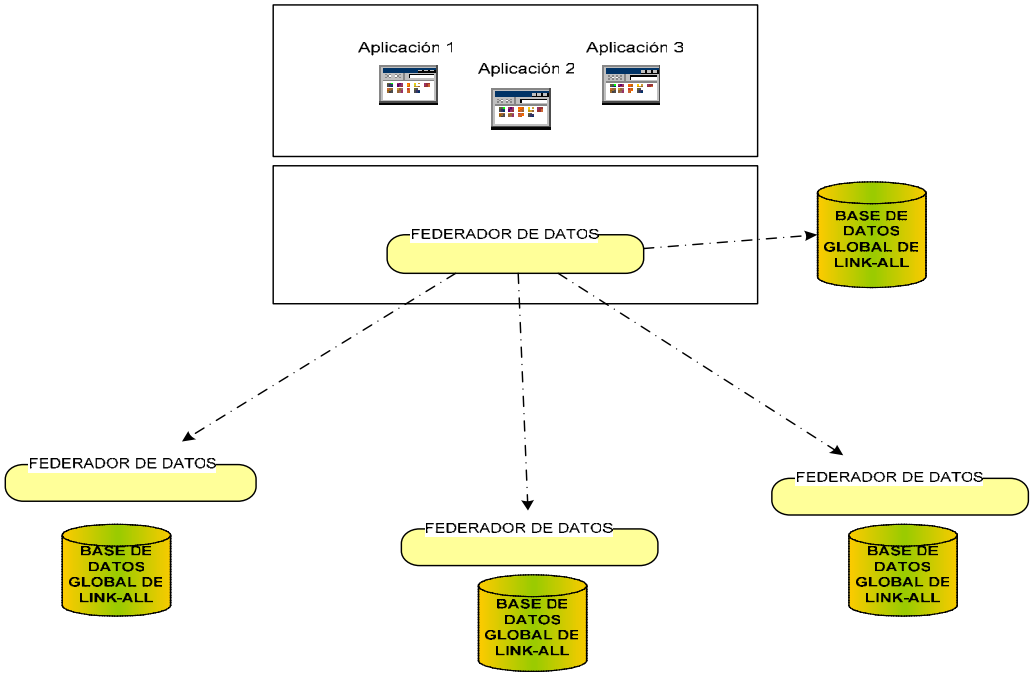


Fig. 3.5 - Federador de Datos.

Al igual que para el Federador de Aplicaciones el Federador de Datos posee los modos de operación local y remoto. A continuación se detalla el esquema de seguridad empleado para restringir el acceso a las tablas en base a los permisos de los usuarios.

3.2.2.1. Seguridad del Federador de Datos

Las políticas de control de acceso a los datos serán forzadas por el Federador de Datos mediante la utilización de un esquema de seguridad de dos niveles. En el primer nivel el Federador forzará los controles de seguridad a nivel de software, sobrescribiendo o mejorando los comandos enviados a la base de datos. Una vez sobrepasado este nivel se utilizarán las estructuras de la base de datos global para asegurar el control de seguridad al nivel más bajo posible.

El control de acceso será implementado por grupos de seguridad definidos en la base de datos administrativa. Cada usuario puede pertenecer a uno o más grupos, pero los permisos para cada grupo serán establecidos para cada registro o tabla de la base de datos. De esta forma los controles de seguridad son compartidos entre el federador de datos y las estructuras de la base de datos.

Al igual que con el Federador de Aplicaciones, cada comunidad de la red Link-ALL podrá solicitar, otorgar o denegar permisos de acceso a tablas a grupos de comunidades remotas.

Toda vez que una consulta sea recibida por el Federador de Datos éste deberá realizar tres tareas:

1. Recuperar los grupos a los cuales pertenece el que envía la consulta.
2. Realizar controles de seguridad básicos, como:
   2.1. El grupo puede enviar este tipo de consultas?
   2.2. El grupo puede usar el Federador de Datos?
   2.3. El grupo puede acceder a esta base de datos?
3. Modificar la consulta recibida para tener en cuenta las estructuras de seguridad a nivel de base de datos.

Seguridad de la Base de Datos Global

Una vez que el Federador de Datos modifica y propaga la consulta al nivel de base de datos, los permisos para cada grupo serán considerados.

Para implementar la seguridad a nivel de base de datos, la misma es segmentada por comunidad. Para esto cada tabla de la base global contiene un campo indicando a que comunidad pertenece y de ésta forma se pueden obtener vistas dependientes de cada comunidad.

Se incorpora además una tabla de mapeos que dado un grupo de una determinada comunidad, indica cual columna de mapeo le corresponde para cierta tabla. Otra tabla especial incorporada a la base de datos global indica cuáles tablas tienen restringido su acceso.

3.2.2.1.1. Procesar Consulta

Esta operación permite al Federador de Datos ejecutar la consulta solicitada y retornar el resultado. La misma podría ser una consulta SQL o una consulta en formato de Hibernate (HQL) [15]. El federador será capaz de procesar tanto consultas de actualización como de selección de datos. Las consultas de selección serán ejecutadas en múltiples fuentes de datos (en la federación) y las consultas de inserción y actualización serán ejecutadas en una única fuente de datos (base de datos local). Para determinar donde se ubican las fuentes de datos el Federador se vale de la información de la topología de la red Link-ALL. Las consultas podrán ser ejecutadas o retornarán resultados siempre y cuando respeten las restricciones de seguridad correspondientes.

Para poder ejecutar una consulta el Federador de datos recibe información del contexto, el cual determina el modo de operación, la consulta en sí misma e información de control que especifica como la consulta deberá ser procesada, indicando si el resultado debe ser paginado y la cantidad máxima de resultados a recuperar.

Los clientes puede que no manejen grandes cantidades de resultados, y se espera que el Federador de Datos maneje los resultados obtenidos. Muchas veces los clientes no usan todos los datos obtenidos en las búsquedas y los descartan luego de inspeccionar o utilizar una porción de ellos. Por lo tanto no es necesario retornarle al cliente el conjunto entero de resultados. Si el cliente despliega unos pocos y luego abandona la consulta no se desperdicia ancho de banda, porque los datos se pueden cachear del lado del servidor y enviarlos al cliente a medida que este los necesite.

El esquema de seguridad del Federador descrito anteriormente así como también la necesidad de conocer en que servidores se encuentran los datos involucrados hacen necesario el análisis de la consulta recibida y posterior procesamiento para incorporarle las restricciones de seguridad.

Es importante comprender la problemática abordada: cuando una consulta llega al Federador de Datos, hay dos características principales que determinan el accionar del componente, una es el tipo de la consulta en cuestión (consulta de selección o actualización) y otra es el contexto en que se realiza (esto determinará el modo de operación local o remoto). Las consultas de actualización sólo se ejecutan en la fuente de datos local, por lo que no pueden existir consultas de actualización con contextos remotos. En todos los tipos y contextos de ejecución de consultas los controles de seguridad se mantienen.

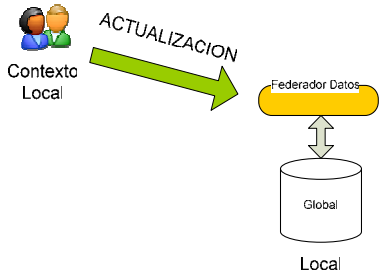


Fig. 3.6 – Consulta de actualización, contexto local.

El caso más complejo es cuando llega al Federador de Datos una consulta de selección. Si el contexto de la llamada es local (o sea un usuario que ha iniciado una sesión en el servidor) el Federador de Datos deberá realizar cuatro tareas:

1. Realizar la consulta a la base de datos local, anexando la información de seguridad correspondiente a los permisos que el usuario tenga sobre las tablas.
2. Determinar a qué servidores remotos puede enviar la consulta (en este caso puede valerse de la información de seguridad para conocer en que fuentes de datos el usuario tiene acceso).
3. Realizar la consulta a los nodos remotos. Si un usuario tiene acceso a N servidores remotos existirán entonces N potenciales proveedores de datos. La consulta recibida se envía tal cual fue recibida, ya que es responsabilidad del nodo remoto agregarle las restricciones de seguridad.
4. Realizar la integración de los resultados obtenidos no sólo del nodo local sino también de los nodos remotos y una posterior división en páginas que luego serán servidas a los clientes.

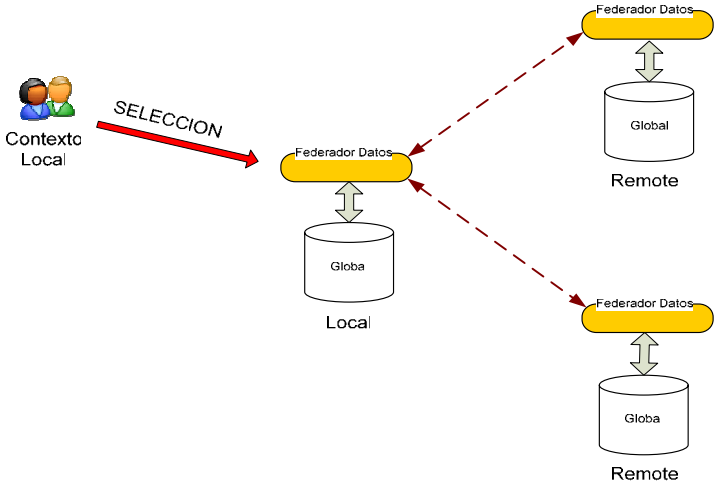


Fig. 3.7 - Consulta de selección, contexto local.

En el caso de que el contexto de la llamada sea remoto el Federador de Datos deberá:

1. Realizar la consulta a la base de datos local, anexando la información de seguridad correspondiente a los permisos que el usuario tenga sobre las tablas.
2. Realizar la división en páginas que luego serán servidas a los clientes remotos.

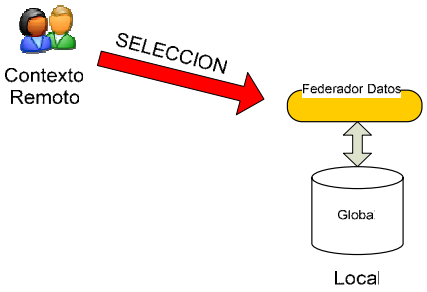


Fig. 3.8 - Consulta de selección, contexto remoto.

Una vez que todos los resultados posibles han sido obtenidos es responsabilidad del administrador de resultados la concatenación de las filas obtenidas de las distintas fuentes. Posteriormente se procede a la división en páginas, teniendo en cuenta el tamaño de página solicitado por el usuario y luego se almacenan los resultados en el Caché Local. Este caché permite almacenar las consultas realizadas por los clientes del nodo, ya sean usuarios o federadores remotos (o sea las consultas que el Federador recibe). A su vez la información de las consultas solicitadas a otros federadores se almacenan en el Caché Remoto (o sea las consultas que el Federador realiza).

3.2.2.1.2. Obtener otra página de una consulta realizada

Esta operación permite a los clientes del Federador de Datos solicitar páginas de una consulta que han realizado previamente. Así los clientes pueden “navegar” los datos iterando sobre los resultados cacheados y paginados que posee el servidor.

En el momento de la invocación el Federador recibe en el contexto de la llamada el identificador de sesión de trabajo que es utilizado para validar al usuario y verificar que tenga permiso para tal funcionalidad. También recibe información de control sobre la consulta incluyendo el identificador de la consulta realizada y el número de página solicitado. Con estos datos el Federador es capaz de recuperar información sobre la consulta así como también la página requerida por el usuario para así retornarla.

Una vez que el Federador de Datos ejecuta una consulta de selección mantiene en memoria los resultados obtenidos. Dichos resultados podrían ser locales y remotos al nodo como se detalló anteriormente. Podría suceder que por ejemplo un usuario realice una consulta, mire los primeros resultados y deje de navegar las páginas por cierto tiempo. Si la sesión de trabajo de ese usuario no expira, o tarda en expirar, podría suceder que el Federador mantenga datos des-actualizados en memoria. Es decir que el Federador mantiene páginas cacheadas obtenidas tal vez de nodos remotos y puede haber sucedido que los datos hayan sido cambiados o ni siquiera existan. El Federador de Datos intenta abordar este problema de “coherencia de caché” brindando la posibilidad de definir un time-out para las páginas de resultados almacenadas en memoria. Es decir que si se establece un tiempo de vida límite y el usuario lo sobrepasa, el Federador de Datos recuperará información de la consulta que había sido ejecutada, la ejecutará nuevamente y re-paginará los resultados obtenidos.

3.2.3. Herramientas de Configuración del Federador.

3.2.3.1. Federar Servidor

Esta operación permite a los servidores integrarse a la Federación Link-ALL. Un determinado servidor puede federarse como servidor de soporte o servidor común. La federación de un nodo de soporte implica que el federador ingrese en la topología del servidor a este nodo y sus comunidades. En cambio un nodo común solicita federarse al servidor de soporte, para esto el nodo debe tener un servidor de soporte configurado con anterioridad. Seguidamente se ingresa el nodo y sus comunidades en la topología del servidor de soporte y una vez que el nodo ha sido federado recibe la información de la topología Link-ALL desde el servidor de soporte. Esta funcionalidad es invocada por el administrador del Servidor

3.2.3.2. Desfederar Servidor

Un servidor se da de baja de la red Link-ALL, el administrador decide desfederar al servidor. El Federador del nodo informa al servidor de soporte que el nodo ya no está federado, por lo que en el servidor de soporte se elimina de la topología el nodo, las aplicaciones que este haya federado y también se eliminan los permisos sobre los recursos de ese servidor.

3.2.3.3. Crear, Actualizar y Eliminar Comunidades

El Federador brinda las funcionalidades de gestión de comunidades de un nodo, es decir permite crear, actualizar y eliminar comunidades de un nodo. Si el nodo está federado la creación, actualización y eliminación de las comunidades también son comunicadas a la federación a través del nodo de soporte.

3.2.3.4. Actualizar mapa de la federación

La actualización del mapa de la federación se hace desde el servidor de soporte hacia un nodo común, ya que todas las operaciones que un nodo efectúe en la federación son informadas o

realizadas contra el servidor de soporte por lo que éste mantendrá actualizada la topología de la federación.

Pueden existir varios mecanismos para actualizar el mapa de federación en un nodo, según las características del nodo se pueden emplear uno u otro. Éste mecanismo de actualización puede ser configurable por el usuario administrador del servidor.

Una posibilidad es que el administrador del nodo realice manualmente la actualización, o puede configurar para que la actualización se realice de forma automática.

La actualización puede hacerse de dos maneras, una forma es reemplazando completamente la topología local, por la que se recibe del servidor de soporte. Otra forma más inteligente es simplemente actualizar las modificaciones realizadas desde la última sincronización. Ambos mecanismos de actualización se pueden configurar para que se realicen con cierta frecuencia o se realicen manualmente por el administrador del nodo.

Otra posibilidad para la actualización, es que los servidores comunes se suscriban a un servicio de actualización brindado por el servidor de soporte. Mediante este servicio se enviarán las modificaciones a los servidores suscriptos a medida que estas vayan ocurriendo, o cada cierto período de tiempo se enviará un grupo de actualizaciones realizadas.

3.3. FedBrowser

El navegador de la federación (FedBrowser) permite visualizar el estado actual de la federación. Se trata de una aplicación J2EE, integrada a la plataforma Link-ALL que permite al usuario tener una idea visual del conjunto de servidores, servicios y datos federados, brindando información acerca de los mismos.

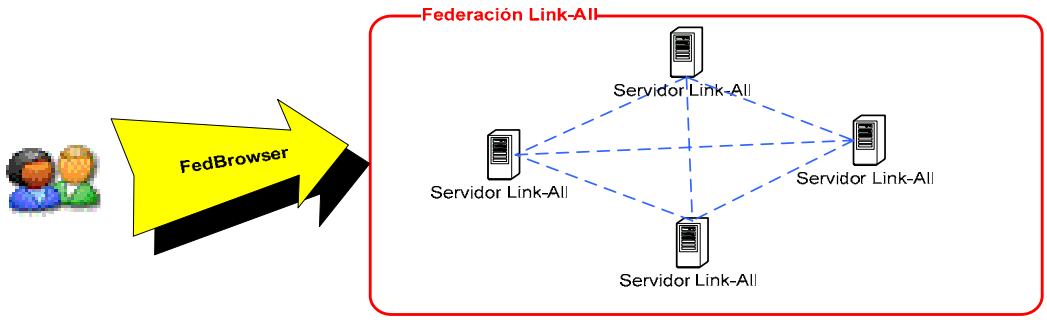


Fig. 3.9 - FedBrowser, el navegador de la federación.

Esta herramienta permite visualizar el mapa de la federación de dos formas, a través de un cliente fino (Applet) o un cliente grueso (Aplicación Swing). La representación de la información de la federación se logra a través de un proceso de tres fases: obtención de la información, procesamiento y despliegue que se detallan a continuación.

1. **Obtención de la información de la federación.**

El navegador de la federación FedBrowser cuenta con un módulo cliente que le permite consumir el Web Service brindado por el Federador Link-ALL para la obtención del Mapa de la Federación. Dicho mapa es una representación del contenido de la federación (servidores, comunidades, aplicaciones, topología) en formato XML.

2. **Procesamiento.**

En esta etapa se produce el procesamiento de la información. Es decir que el contenido de la federación obtenido del Web Service en formato XML es procesado y almacenado en memoria.

3. Despliegue.

La etapa final es la del despliegue donde la interfaz de usuario consulta los datos almacenados en memoria y los representa visualmente. Una característica a destacar es que la interfaz de usuario puede solicitar a la lógica datos específicos sobre una determinada entidad desplegada (un servidor por ejemplo). Esta operación se invoca siempre y cuando el usuario desee saber más acerca de dicha entidad, para lo cual la UI realiza el pedido y la lógica como contraparte serializa y retorna la información deseada.

4. Implementación de la solución.

4.1. Federador Link-ALL

El Federador Link-ALL está formado por dos componentes principales: el Federador de Aplicaciones y el Federador de Datos.

En la definición de la arquitectura tanto del Federador Link-ALL como de los utilitarios se tomó como referencia el modelo de Java 2 Enterprise Edition (J2EE) para sistemas de información [4]. En el diseño de los componentes se destaca el uso de patrones, con el objetivo de incorporar buenas prácticas probadas en sistemas reales, contribuyendo así a lograr una solución flexible y fácil de extender.

A continuación se describe la arquitectura global del sistema construido utilizando para ello el conjunto de vistas definido por el proceso RUP. Por más información referirse al anexo Documentación Técnica [20].

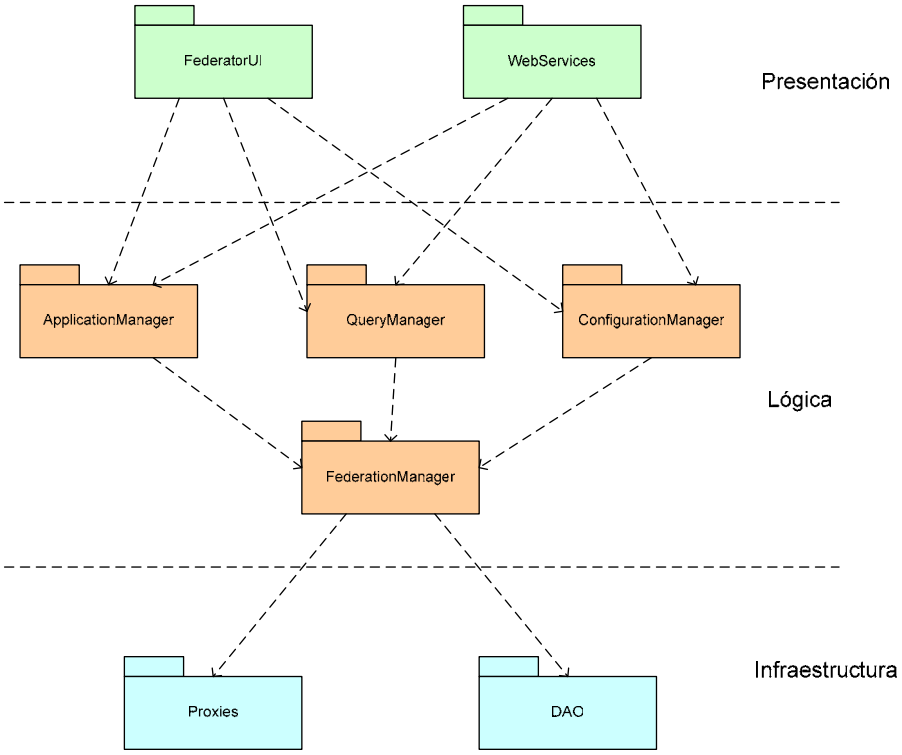


Fig. 4.1 - Vista lógica de la arquitectura del Federador Link-ALL.

El Federador Link-ALL presenta una arquitectura de tres capas en donde la lógica se encuentra dividida en componentes de acuerdo a su función.

En la capa del cliente (no representada en la figura) se encuentran los componentes que se ejecutan en la máquina del cliente. Estos componentes son los clientes Web (clientes finos) y las aplicaciones que consumen los Web Services ofrecidos por el Federador Link-ALL. Los clientes Web son clientes HTML híbridos ya que poseen tecnología JavaScript. Esta tecnología provee cierta inteligencia a las páginas posibilitando realizar validaciones básicas (por ejemplo chequeo del formato de datos ingresados y campos obligatorios) antes que la información sea enviada al Federador. Esto tiene como ventaja la disminución de la carga del servidor Web donde se encuentra y el tráfico en la red (debido a que se necesita enviar menos información). Los clientes que consumen los Web Services ofrecidos por el Federador se encuentran actualmente representados por federadores remotos y

herramientas de navegación como es FedBrowser, pero eventualmente podría ser cualquier aplicación capaz de consumir dichos Web Services.

En la capa de presentación se encuentran los Web Services de apoyo a la Federación así como también Servlets y páginas JSP que generan las páginas y contenido dinámico de las mismas.

En la capa lógica se encuentran los componentes de negocio que permiten implementar la lógica del Federador Link-ALL, es decir sus funcionalidades. Los componentes de esta capa son Enterprise Java Beans (EJBs) y Plain Java Objects (POJOs). Los EJBs utilizados son Session Beans sin estado que actúan a modo de fachada, manejando las conversaciones con los clientes.

En la capa de infraestructura se encuentran los componentes de acceso a recursos como la base de datos Global de Link-ALL y otros componentes externos al Federador como los servicios de seguridad, sesión y acceso a la base de datos Administrativa. En esta capa se encuentran también los proxies que enmascaran la comunicación con federadores remotos.

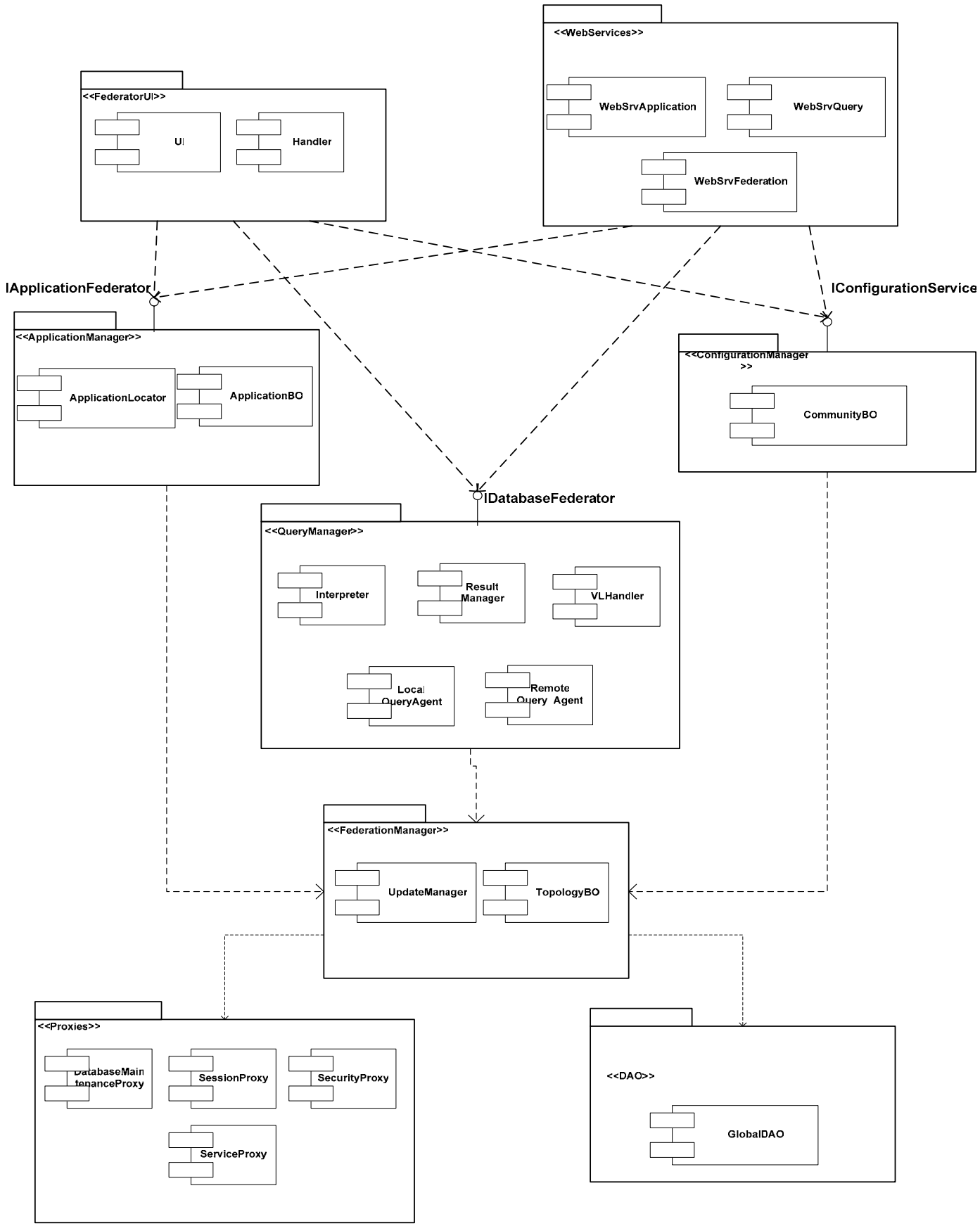


Fig. 4.2 - Vista de Implementación del Federador Link-ALL.

4.1.1. Federador de Aplicaciones

Está compuesto por los subsistemas ApplicationManager y FederationManager, donde éste último brinda también los servicios de federación al Federador de Datos.

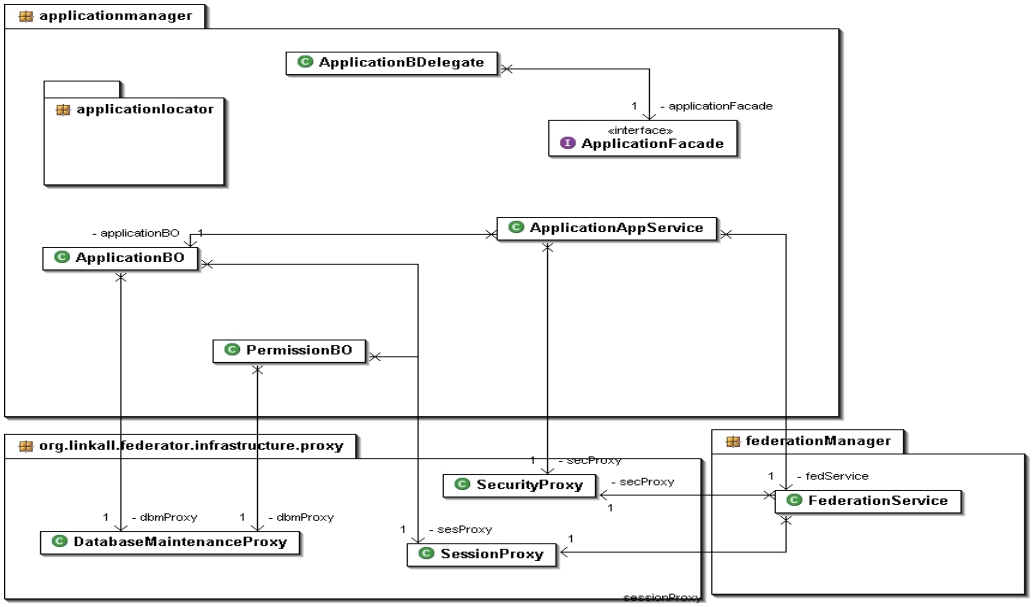


Fig. 4.3 - Subsistema ApplicationManager.

El subsistema ApplicationManager permite realizar la gestión de aplicaciones. Entre sus principales funciones se encuentran la de permitir la federación y desfederación de aplicaciones instaladas en el servidor Link-ALL, obtener las aplicaciones que tiene acceso un determinado usuario y rutear pedidos de aplicaciones.

Se describe a modo de ejemplo la implementación de rutear pedido de aplicación para mostrar los mecanismos de funcionamiento del Federador de Aplicaciones. Por una descripción más detallada de las funcionalidades referirse al capítulo anterior; para ver un diseño más detallado de las demás funcionalidades ver anexo de Diseño de Casos de Uso [19].

4.1.1.1. Rutear el pedido de una aplicación

El usuario selecciona la aplicación que desea ejecutar de la lista de aplicaciones disponibles. Como cualquier evento en la interfaz de usuario provoca una llamada al Controller y éste a su vez invoca al manejador que atenderá el pedido, que en el caso de rutear aplicación es el LocateHandler. El handler se encarga de obtener los parámetros ingresados por el usuario en la página JSP a través del request. Para ésta funcionalidad los parámetros obtenidos son el código de la aplicación y el identificador de la sesión del usuario. A partir del código de la aplicación se crea un Value Object de aplicación y a partir del identificador de la sesión se crea un Value Object de contexto. Con estos parámetros el handler invoca al método getApplicationLocation del ApplicationBDelegate.

El BusinessDelegate utiliza el ServiceLocator para obtener una referencia a la Fachada de los servicios de negocio referidos a aplicaciones (ApplicationFacade). Una vez obtenida la referencia se realiza una invocación vía RMI a dicha fachada, que es un Session Bean sin estado. A partir de aquí el Session Bean se encarga de llevar a cabo la solicitud, invocando a los servicios correspondientes; que en este caso es el método getApplicationLocation del servicio ApplicationAppService.

En la lógica de negocio primeramente se valida el contexto de la llamada y después se chequean los permisos del usuario sobre la aplicación. Estas validaciones se hacen con llamadas a los servicios de la plataforma Link–ALL por intermedio de los proxies correspondientes. Aquí se utiliza el patrón Strategy [8]: el componente AppLocator se encarga de localizar la aplicación utilizando la estrategia SeachAppStrategy. Esta estrategia básicamente consiste en buscar las aplicaciones entre las locales;

si la encuentra retorna la URL de la aplicación, en caso de no encontrarla, busca entre las aplicaciones remotas que estén federadas y que el usuario tenga permisos.

Para localizar una aplicación remota, se busca dentro de los permisos remotos del usuario. En los permisos se indica el nodo en el que está instalada la aplicación, la comunidad a la que pertenece y el grupo y comunidad de usuarios que pueden acceder. Luego de esto se verifica que ese nodo y la aplicación también estén federados realizándose una llamada al Web Service encargado del manejo de aplicaciones en el nodo remoto. Si el resultado es correcto, devuelve al igual que en el caso anterior la URL de la aplicación, que además tendrá la dirección del nodo y el identificador de la sesión remota que ha sido creada.

En el servidor donde está instalada la aplicación la llamada es atendida por el Web Service. El mismo se encarga de invocar al BusinessDelegate para rutear aplicación remota. Cuando la invocación llega al ApplicationAppService, primero se verifican los permisos del grupo del nodo remoto, que hace la solicitud de rutear la aplicación. Al igual que con las aplicaciones locales se crea una estrategia para obtener la URL de la aplicación. Al recibir un contexto remoto, la estrategia solo busca la aplicación entre las aplicaciones locales y crea una sesión remota para devolver junto con la URL, a la que incluye la dirección del nodo local.

4.1.2. Federador de Datos

El Federador de Datos es el otro componente fundamental del Federador Link-ALL. Sus principales subsistemas son QueryManager y FederationManager.

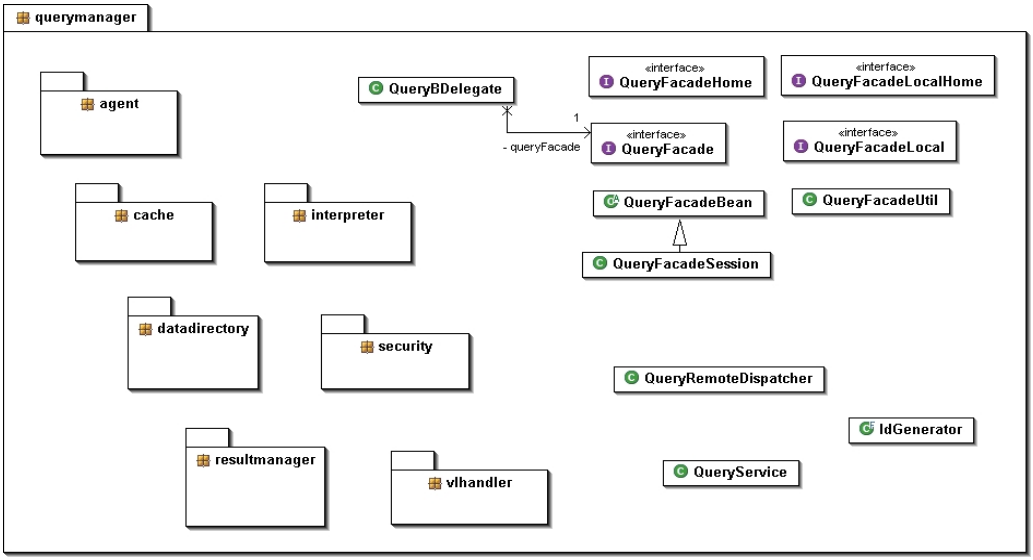


Fig. 4.4 - Subsistema QueryManager.

La función más importante de QueryManager es la de permitir la federación de datos existentes en el servidor Link-ALL. Esto incluye la actualización de datos locales y la posibilidad de consultar datos distribuidos a lo largo de toda la federación de manera transparente al usuario. Éste componente se integra al subsistema FederationManager quien le provee servicios de acceso a la federación.

Se detalla a continuación la implementación de procesar consulta con el objetivo de mostrar los mecanismos de funcionamiento del Federador de Datos. Por una descripción más detallada de las funcionalidades referirse al capítulo anterior; para ver un diseño más detallado de las demás funcionalidades ver anexo de Diseño de Casos de Uso [19].

4.1.2.1. Procesar Consulta

Se solicita al Federador de Datos la ejecución de una consulta SQL. Para ello se invoca al QueryBDelegate, quien a través del Service Locator realiza la llamada vía RMI al Session Bean sin

estado encargado de las consultas (QueryFacade) y que actúa como fachada de los servicios de negocio.

Una vez que el servicio ha recibido una consulta, las acciones que se toman dependen del tipo de consulta recibida (selección o actualización) y el contexto (local o remoto). Si la consulta recibida es de actualización entonces se instancia un adaptador JDBC (DefaultJDBCAdapter) quien es el encargado de realizar la inserción, actualización o eliminación necesaria. Anteriormente dicha consulta es modificada donde se le anexa la información de seguridad si corresponde.

Si la consulta es de selección realizará las acciones necesarias el administrador de listas de resultados (Vlhandler) utilizando para ello un agente de consulta local o remoto, dependiendo de las características del contexto de la llamada. Al administrador de listas de valores se le puede establecer un tiempo máximo de vida al caché en que almacena las páginas obtenidas.

Si el contexto es local, la consulta es manejada por el agente encargado de las consultas locales (LocalQueryAgent). Este agente realiza la consulta en la base de datos Global local después de haber sido examinada por el componente encargado de la seguridad (DataSecurityManager). Posteriormente la consulta original es enviada a los nodos remotos a los que el usuario tenga permisos de acceso. El conjunto de estos nodos se obtiene a través del directorio de datos (quien examina los permisos del grupo que realiza la consulta). La consulta a la base de datos local es realizada utilizando el adaptador JDBC DefaultJDBCAdapter. Una vez que se han conseguido tanto los resultados locales como remotos, el administrador de resultados (ResultManager) se encarga de unirlos y dividirlos en fragmentos según lo solicitó el cliente, para luego paginarlos y almacenarlos en memoria.

Si el contexto es remoto, la consulta es manejada por el agente encargado de las consultas remotas (RemoteQueryAgent). La misma es analizada por DataSecurityManager y luego se ejecuta en la base de datos Global local. Por último los resultados son administrados por ResultManager quien pagina y almacena en memoria los resultados.

4.2. FedBrowser

El navegador de la federación presenta una arquitectura de tres capas: Presentación, Lógica y Recursos como se muestra a continuación:

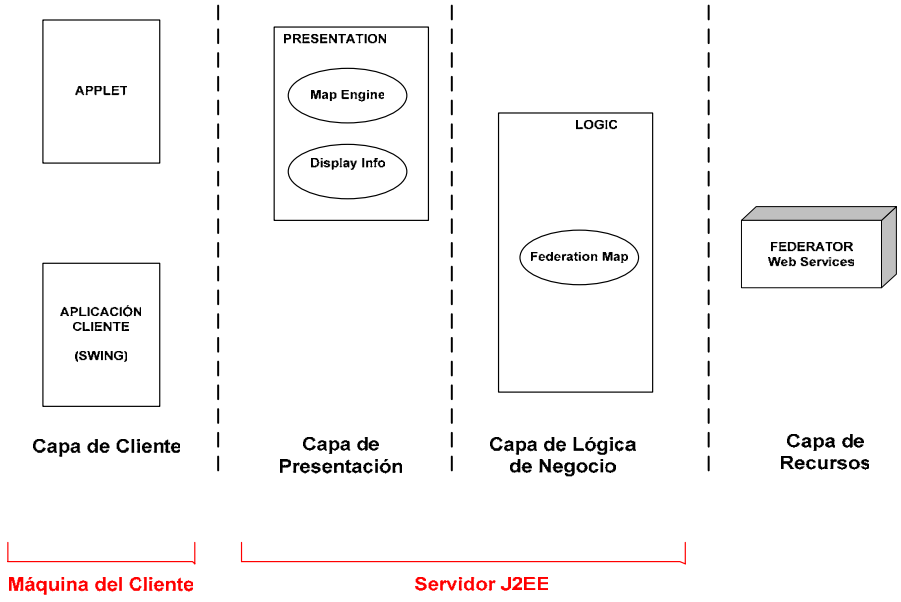


Fig. 4.5 - Arquitectura de FedBrowser.

En la capa del cliente se encuentran los componentes que se ejecutan en la máquina del cliente. Entre los mismos se encuentran un Applet y una aplicación de escritorio Swing que permiten visualizar la información de la federación.

Como se mencionó en el capítulo anterior la representación visual del estado de la federación se logra a través de un proceso de tres fases: obtención de la información, procesamiento y despliegue. A continuación se detalla dicho proceso brindando detalles técnicos de los pasos del mismo.

1. Obtención de la información de la federación

Para obtener la información de la federación FedBrowser utiliza un Proxy remoto de acceso (FederatorProxy) a los servicios del Federador. Dicho Proxy es un cliente del Web Service ofrecido por el Federador Link-ALL que permite obtener el mapa de la federación actual en formato XML.

La configuración de la comunicación con el Federador Link-ALL se encuentra en un archivo de configuración denominado fedbrowserconfig.properties.

2. Procesamiento

Una vez obtenido el XML del mapa de la federación el mismo es deserializado y almacenado en memoria en un objeto que lo representa denominado FederationMap. Este objeto posee métodos para cargar el mapa a partir de un paquete de actualización obtenido del Federador, así como también para realizar consultas sobre sus datos almacenados.

3. Despliegue

El rol más importante en la etapa de despliegue lo tiene la herramienta TouchGraph [10] quien es la encargada de dibujar cada una de las entidades del mapa de la federación en la pantalla. Esta herramienta necesita que le provean la información en un determinado formato para poder desplegarla. El método formatToFedBrowser del objeto FederationMap es un método especial que permite “formatear” la información de la Federación contenida en el mapa a un formato XML que la aplicación entienda.

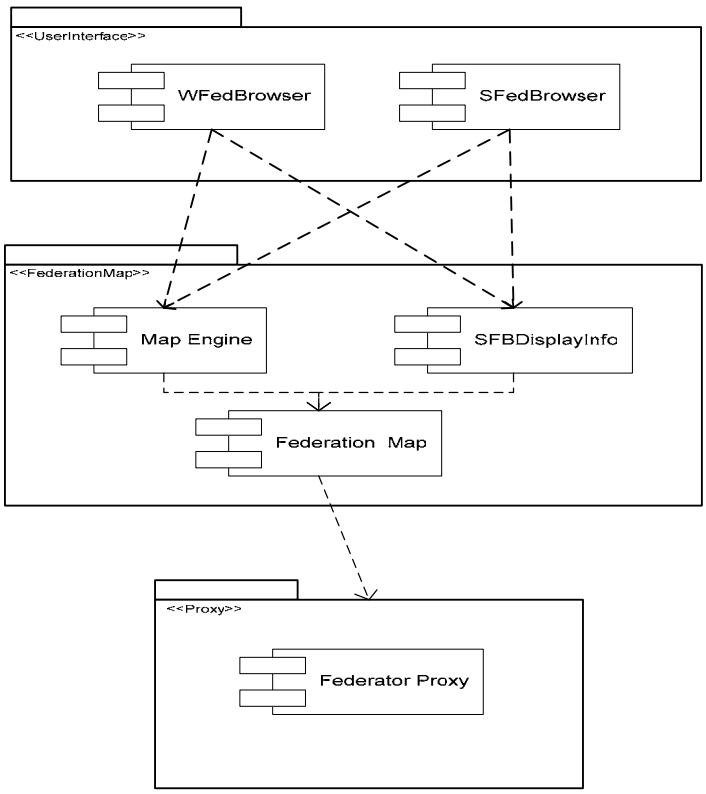


Fig. 4.6 - Vista de Implementación de FedBrowser.

4.3. Decisiones de Implementación.

En ésta sección se describen decisiones de implementación tomadas en el proyecto. Algunas de ellas fueron dictadas por los requerimientos no funcionales, como por ejemplo la utilización de la plataforma J2EE, el servidor de aplicaciones JBoss y la utilización de Web Services como forma de comunicación entre federadores remotos.

Además de lo mencionado anteriormente, se tomaron decisiones en cuanto a la utilización de frameworks que brindaran apoyo en la implementación de las funcionalidades que se detallan a continuación:

- Planificador de trabajos (QUARTZ)
- Navegador de la Federación (TouchGraph)
- Parser SQL (Jcsql)

Estos frameworks permiten promover la reutilización de código además de proporcionar soluciones implementadas y probadas a problemas puntuales, lo que permitió así concentrarse en la resolución de los requerimientos funcionales del proyecto.

Para cada una de las áreas mencionadas anteriormente se investigó y evaluó posibles frameworks a utilizar teniendo en cuenta los siguientes criterios: que cumpliera con nuestras necesidades, fuera de código abierto y un proyecto activo actualmente.

Planificador de trabajos

El framework que brinda los servicios de planificación de trabajos es Quartz [9]. Dicho framework es un sistema de planificación de trabajos de código abierto que puede ser integrado a cualquier aplicación J2EE y permite definir planificaciones complejas. Además es fácil de utilizar, liviano, eficiente y tolerante a fallas.

En el Federador Link-ALL, el acceso a dicho framework es realizado a través de un Proxy, con el objetivo de desacoplar al Federador del uso del mismo. Esto permitiría cambiar eventualmente de sistema de planificación de trabajos sin mayores consecuencias para el sistema.

Navegador de la Federación

La representación visual del estado de la federación se logra utilizando el framework TouchGraph [10]. Dicho framework fue utilizado debido a que el grafo que logra representar visualmente tiene similitudes con la representación esperada de la federación. Tiene la ventaja también de que posee buenas posibilidades de manejo de las entidades visualizadas: es posible restringir la cantidad de elementos desplegados, posee zoom y soporta eventos, entre otras. Además es un framework de código abierto, con ejemplos de uso y documentación.

Parser SQL

En el Federador de Datos se utilizó JSqlParser [11]. Este framework permite parsear una sentencia SQL y la traduce a una jerarquía de clases Java. Dicha jerarquía puede ser navegada y modificada si es preciso. Soporta además la operación inversa, es decir construir una jerarquía de clases que represente una consulta SQL y a partir de ella obtener la cadena de caracteres correspondiente. El framework fue utilizado dado que es de código abierto y proporciona la importante característica de parsear cualquier tipo de consulta SQL.

Por más detalles referirse al anexo Estado del Arte [18].

4.4. Características del producto desarrollado

En esta sección se describen las principales características del producto construido. Entre ellas se destacan sus funcionalidades más importantes así como también aspectos del diseño, calidad y verificación.

4.4.1. Generales

- Integrado a la última liberación de la plataforma Link-ALL.

4.4.2. Diseño

- Utilización de patrones de diseño. Esto tiene como ventajas la aplicación de soluciones probadas en sistemas reales e incorpora buenas prácticas al diseño del producto.
- Utilización de interfaces con el fin de reducir el acoplamiento entre los componentes y aumentar la facilidad en el momento de la verificación.

4.4.3. Calidad del código

4.4.3.1. Estándares de codificación

Para la implementación del producto se siguieron un conjunto de buenas prácticas de codificación de forma que el código producido fuera fácil de leer y mantener además de estar bien documentado. Para lograr tales objetivos se realizó lo siguiente:

- Se estableció un estándar de codificación basado en el propuesto por SUN. Como ayuda fueron configuradas las herramientas de formateo de código que Eclipse [14] posee para que la codificación cumpliera tal estándar.
- Se estableció un estándar de documentación interna del código el cual siguió las convenciones de JAVADOC.
- Se definió que los métodos estuvieran ordenados alfabéticamente dentro de las clases para una fácil ubicación de los mismos en el momento de la implementación.
- Se diseño una estructura de paquetes a partir de las sugerencias que brinda SUN para las aplicaciones J2EE.

4.4.4. Funcionalidades ofrecidas

4.4.4.1. Actualización de la información de la Federación

4.4.4.1.1. Modo de actualización

La actualización de la información de la Federación se puede realizar tanto de forma manual como automática.

- Manual:

El usuario podrá actualizar la información de la federación que posee el nodo en cualquier momento, descargando la última versión desde el servidor de soporte.

- Automática:

La actualización de la información de la federación podrá ser configurada para que el sistema la realice automáticamente. Dicha configuración es flexible permitiendo al usuario establecer que la actualización se efectúe determinado día, cada cierto intervalo de tiempo, etc.

4.4.4.1.2. Mecanismo de actualización robusto

El mecanismo de actualización de la información de la Federación es robusto siendo capaz de recuperarse a caídas del servidor. Es decir que el servidor es capaz de "recordar" el modo de actualización que tenía configurado y en caso de ser automático ejecutar las actualizaciones agendadas.

4.4.4.1.3. Estrategia de actualización parametrizable

Estrategia de actualización parametrizable (por defecto fuerza bruta) siendo fácil de integrar nuevas estrategias de actualización (ver puntos de extensión).

4.4.4.2. Transaccionalidad en el acceso a datos

Transaccionalidad administrada por el servidor de aplicaciones en el acceso a la Base de Datos Administrativa de Link-ALL (NAD) y a la Base de Datos Global. Esto provee al producto de robustez ante fallas en las actualizaciones de datos ya que el contenedor deshará los cambios realizados. Esta característica cobra vital importancia en la actualización de la topología ya que es primordial que una eventual falla en la misma no deje la topología del nodo inconsistente, lo que podría imposibilitar al Federador en sus funciones.

4.4.4.3. Puerto de acceso al Federador configurable

El puerto de acceso a los servicios ofrecidos por el Federador podrá ser configurable por el administrador. De esta forma los servidores podrán iniciar sus servicios en diferentes puertos sin perjuicio de la comunicación con los demás nodos de la federación ya que la información del puerto se transfiere en las actualizaciones de la topología.

4.4.5. Verificación

Se automatizaron casos de prueba con JUnit [13] y además el producto fue testeado en diferentes plataformas lo que permite asegurar su correcto funcionamiento en:

- Sistemas Operativos:
  Windows 2000 Professional, SP4
  Windows XP, SP1
  Suse Linux 9.0

- Servidor de Aplicaciones:
  JBoss – Versión 3.2.3
  JBoss – Versión 4.0.0

- Navegadores Web:
  Mozilla Firefox 1.0
  Internet Explorer 6.0
  Konqueror

4.5. Características del proceso de desarrollo

4.5.1. Metodología de desarrollo

Para el desarrollo del proyecto se empleó una metodología iterativa incremental en la que se realizaron cuatro iteraciones. En cada iteración se efectuaron las etapas de análisis, diseño, implementación y verificación.

4.5.1.1. Proceso de desarrollo

En la primera iteración se realizó la mayor parte del relevamiento de los requerimientos para el Federador de Aplicaciones. En base a estos requerimientos, se analizaron y crearon los casos de uso correspondientes. También se realizó el diseño del sistema y los subsistemas, el cuál debía ser lo suficientemente flexible para lograr la integración con la plataforma Link-ALL y poder incorporar las funcionalidades del Federador de Datos. En esta etapa se realizó la prueba de concepto y un primer prototipo donde se implementaron las funcionalidades obtener aplicaciones de usuario, rutear pedido de ejecución, federar y desfederar aplicación. Esta primera versión del federador no estaba integrada a la plataforma Link-ALL, pero ya contaba con Web Services para la comunicación entre servidores. Es importante destacar que este prototipo logró cumplir el objetivo de validar el diseño realizado. La verificación en esta iteración consistió en diseñar y ejecutar pruebas para los componentes y casos de uso implementados, algunos de los cuales se realizaron de forma automática utilizando JUnit [13].

Para la segunda iteración se ajustaron levemente algunos requerimientos que no tuvieron gran impacto, ni en el diseño ni en la implementación. En la etapa de implementación se logró la integración con la versión 1.6 de la plataforma Link-ALL, se modificaron y corrigieron errores de las funcionalidades implementadas en la iteración anterior y se agregaron las funcionalidades federar, desfederar nodo y actualizar el mapa de la federación. Se realizaron pruebas para las nuevas funcionalidades y se modificaron las pruebas de los casos de uso implementados anteriormente.

En la tercera iteración se integró el Federador con la versión 1.8 de la plataforma Link-ALL. La nueva versión de la plataforma trajo consigo un cambio en el manejador de base de datos (DBMS); mientras en las etapas anteriores se había utilizado MySQL [16], a partir de esta iteración se pasó a utilizar PostgreSQL [17]. Este cambio trajo consigo modificaciones en el modelo de datos del nodo administrativo (NAD) por lo que hubo que hacer ajustes en las funcionalidades implementadas. En algunos casos provocó un cambio en la lógica de las mismas; también en esta iteración se agregaron las restantes funcionalidades del Federador de Aplicaciones donde se incluyeron los mecanismos de seguridad y la visualización del mapa de la federación. Se realizaron pruebas de regresión para verificar que el cambio de la implementación de las funcionalidades no introdujo nuevos errores y se diseñaron nuevos casos de pruebas para todas las funcionalidades.

En la cuarta iteración se terminaron de definir los requerimientos correspondientes al Federador de Datos, esto implicó realizar el diseño de nuevos componentes y funcionalidades. En la etapa de implementación se realizó la integración con la versión 1.8.3_02 de la plataforma Link-ALL. Fueron implementadas las funcionalidades correspondientes al Federador de Datos y se corrigieron errores detectados en el Federador de Aplicaciones.

Cabe destacar que el diseño de la arquitectura hizo posible una fácil integración de las nuevas funcionalidades a medida que se iban agregando iteración tras iteración.

4.6. Restricciones establecidas.

A raíz de que el Federador estaba enmarcado en el contexto del proyecto Link-ALL, existieron restricciones respecto a los recursos informáticos utilizados y en las interfaces brindadas por el sistema. A continuación se mencionan dichas restricciones:

- Utilización de herramientas de software libre, esto incluye DBMS, IDE y utilitarios.
- La aplicación debía ser basada en Web, estar enmarcada en una plataforma J2EE, teniendo como servidor de aplicaciones JBoss.
- El lenguaje de programación debía ser Java, utilizando HTML, JavaScript y/o Flash para el diseño de las páginas Web de la interfaz de usuario.
- El sistema manejador de bases de datos utilizado en un primer momento fue MySQL, pero luego se migró la plataforma para utilizar PostgreSQL.
- El modelo de datos utilizado es el mismo en todos los componentes de la Plataforma Link-ALL.
- La comunicación entre Servidores Link-ALL debía ser utilizando Web Services, y el framework a utilizar para brindar tal funcionalidad debía ser Axis.

5. Verificación del sistema

5.1. Objetivos

El objetivo de la verificación del sistema es comprobar que el producto realizado cumple adecuadamente con las funcionalidades requeridas y que lo hace satisfaciendo cierto nivel de calidad. Debido a su relevancia dentro del federador ciertas funcionalidades requerían nivel de calidad cinco y otras de nivel de calidad tres [21].

En el plan de verificación del sistema se establecieron los casos de uso, requerimientos funcionales y no funcionales que serían verificados [22]. A continuación se detalla la lista de estos requerimientos:

- Retornar las aplicaciones de un usuario
- Rutear un pedido de ejecución de una aplicación
- Federar una aplicación
- Desfederar una aplicación
- Solicitar permiso de acceso a una aplicación
- Otorgar permiso de acceso a una aplicación
- Denegar permiso de acceso a una aplicación
- Alta de un servidor en la Federación
- Baja de un servidor en la Federación
- Visualizar y consultar el Mapa de la Federación
- Actualizar el Mapa de la Federación
- Alta de Comunidades
- Baja de Comunidades
- Actualizar datos de Comunidades
- Consultas de selección de datos
- Consultas de actualización de datos
- Solicitar permiso de acceso a tablas
- Otorgar permiso de acceso a tablas
- Denegar permiso de acceso a tablas

Se estableció que las siguientes funcionalidades requerían nivel de calidad cinco:

- Retornar las aplicaciones de un usuario
- Rutear un pedido de ejecución de una aplicación
- Federar una aplicación
- Desfederar una aplicación
- Alta de un servidor en la Federación
- Baja de un servidor en la Federación
- Actualizar el Mapa de la Federación

Dentro del plan se estableció que el alcance de las pruebas de verificación y validación abarcaba las pruebas unitarias, de integración, de sistema y funcionales correspondientes al Federador. Las pruebas de performance, carga, volumen, esfuerzo (estrés), seguridad y control de acceso, fallas, recuperación y configuración fueron comprendidas dentro del alcance pero su realización quedaba sometida a los tiempos que se manejaran durante el transcurso del proyecto.

La intención era reducir el esfuerzo sin desmejorar la calidad del producto final. Debido a esto se determinaron qué funcionalidades de la aplicación serían más intensamente testeadas que otras. Para ello se empleó una técnica combinada basada en User Profiles (perfiles de usuario) y Análisis de Riesgo a lo que se agregó el nivel de calidad requerido de las funcionalidades [21].

5.2. Pruebas Realizadas

Dado que un testing total de todos los componentes usualmente no es posible fue necesario utilizar técnicas de testeo que permitieran minimizar los escenarios y casos de verificación en una forma efectiva, posibilitando lograr además el cubrimiento más amplio posible con el menor esfuerzo. Se combinaron varias técnicas de verificación dado que ha sido probado que es más efectivo que utilizar una sola. Esta combinación incluyó análisis funcional, partición en clases de equivalencia y valores límites [21].

Cumpliendo con el alcance propuesto fueron realizadas las pruebas de componentes, de integración de componentes, de casos de uso y del sistema. Las pruebas de performance, carga, volumen, esfuerzo, configuración y demás no pudieron ser realizadas debido a los tiempos manejados para la implementación del proyecto.

5.2.1. Pruebas de componentes

Las pruebas de componentes incluyeron pruebas automáticas sobre los componentes que se consideraron críticos como ser ApplicationManager, FederationManager y QueryManager. Las pruebas automáticas sobre componentes se realizaron con JUnit, que es una herramienta de fácil uso y se puede integrar sencillamente con el IDE Eclipse. A partir de ellas se pudo realizar una métrica del cubrimiento de los tests de componentes. Esta medida se realizó con la herramienta automática DJUnit [12].

5.2.2. Pruebas funcionales

Los tests de casos de uso incluyeron casos de prueba por cada escenario de los mismos. En las funcionalidades de nivel de calidad cinco se diseñaron casos de prueba utilizando la técnica de partición en clases de equivalencia. Algunas de estas pruebas fueron realizadas automáticamente, pero en ellas no se incluyó la interfaz gráfica. Las pruebas de la interfaz gráfica, así como también las que incluían comunicación entre servidores, fueron realizadas manualmente.

A continuación se describen los casos de prueba de los principales casos de uso:

- **Retornar las aplicaciones de un usuario**

Los casos de pruebas consistieron básicamente en asignar permisos a un usuario sobre determinadas aplicaciones y comprobar que en el resultado estén todas las aplicaciones asignadas y solamente estas aplicaciones. En las pruebas se incluyeron aplicaciones locales y remotas, usuarios que pertenecían a uno o más grupos, servidores federados y no federados y las combinaciones entre estas entradas.

- **Rutear un pedido de ejecución de una aplicación**

Los tests tenían como entrada un código de aplicación y un contexto válido (a partir de este se obtiene el usuario). Estas entradas se dividieron en clases de equivalencia para cada escenario posible del caso de uso, como por ejemplo las aplicaciones pueden ser locales o remotas y el usuario puede tener o no permisos sobre la aplicación. Respecto a los resultados esperados, en caso de ser exitoso en las aplicaciones locales se esperaba que se retornara la dirección relativa de la aplicación y en las aplicaciones remotas la dirección de la aplicación incluyendo la URL del nodo donde está instalada. Además debía retornar también la identificación de la sesión remota creada en este servidor y verificar que efectivamente se haya creado esta sesión.

Estos tests probaron que el federador puede brindar transparencia al usuario sobre la ubicación física de las aplicaciones y además dar ciertos mecanismos de seguridad para la ejecución de las mismas.

- **Federar aplicación y desfederar aplicación**

Las entradas para los tests de federar/desfederar aplicación consistían en un código de aplicación y un contexto válido. Estas fueron divididas en diferentes clases de equivalencia: por ejemplo las aplicaciones pueden estar instaladas o no en el servidor, si están instaladas pueden o no estar federadas; el usuario puede tener permisos o no para federar/desfederar aplicaciones. También se tomaron en cuenta las condiciones de ejecución de la operación, por ejemplo el servidor puede estar federado o no, ser un servidor de soporte o un nodo común. Los casos de prueba se realizaron basándose en combinaciones posibles de las condiciones de ejecución y las clases de equivalencia en las que se dividieron las entradas. Los resultados de las pruebas debían ser verificados en ambos servidores si el servidor donde se federa/desfedera la aplicación no es un servidor de soporte.

- **Alta de un servidor en la Federación**

Las entradas para federar un servidor son los datos del servidor y un contexto válido. Para generar los casos de prueba se consideraron las diferentes clases de equivalencia para las entradas y las condiciones de ejecución. Por ejemplo el usuario puede tener o no permisos para federar el servidor, el nodo puede federarse como nodo común o como servidor de soporte. Si se federa como nodo común puede o no tener configurado el servidor de soporte contra el que se federará. Los resultados esperados varían de acuerdo a las entradas y condiciones de ejecución. Si en el test el servidor se federa como nodo común el resultado se debe verificar tanto en el nodo como en el servidor de soporte.

- **Baja de un servidor en la Federación**

La baja de un servidor en la federación solo tiene como entrada un contexto válido ya que solo se da de baja el servidor local, por lo tanto las posibilidades son que el usuario tenga o no tenga permisos para dar de baja el servidor. También se realizaron casos de prueba teniendo en cuenta las mismas condiciones de ejecución que el alta del servidor. Los resultados esperados dependen de las condiciones de entrada, si en el test el nodo no es servidor de soporte se debe chequear en ambos nodos.

- **Actualizar el Mapa de la Federación**

La entrada para el caso de uso actualizar mapa de la federación es simplemente un contexto válido, por lo que las clases de equivalencia se limitan a que el usuario tenga o no permisos para la funcionalidad. Las condiciones de ejecución son las mismas que para el caso anterior. Los casos de prueba se realizaron en base a combinaciones de la entrada y las condiciones de ejecución. Para que el resultado de la operación sea exitoso el nodo debe estar federado y no ser servidor de soporte, además el usuario debe tener permisos para poder actualizar el mapa. En este caso para que el test no falle el mapa de la federación obtenido debe ser equivalente al mapa del servidor de soporte.

5.2.3. Pruebas de regresión

Como se mencionó en el capítulo anterior, el proyecto fue desarrollado en cuatro iteraciones. En cada iteración se implementaron diferentes funcionalidades y se ejecutaron tests para las funcionalidades implementadas en cada iteración. Estas pruebas fueron reutilizadas y sirvieron como tests de regresión para las siguientes iteraciones.

5.2.4. Pruebas del sistema

Las pruebas del sistema incluyeron fundamentalmente las pruebas funcionales, realizándose mayor énfasis en los casos de uso prioritarios. En ellas se testeó el federador como un todo, realizándose ciclos de pruebas que incluyeron varias funcionalidades. Para realizar estas pruebas se utilizaron las plataformas Windows 2000, Windows XP, Suse Linux 9.0 y los navegadores Mozilla/5.0 (Firefox 1.0), Internet Explorer 6.0, y Konqueror.

5.3. Resultados Obtenidos

5.3.1. Resultados de los tests

Los resultados obtenidos en los tests muestran que el porcentaje de fallas sobre casos de prueba fue disminuyendo a medida que se avanzaba en las iteraciones. Para el caso de las pruebas de componentes realizadas sobre ApplicationManager y FederationManager en la última iteración se llegó a que la cantidad de fallas sobre la cantidad de casos de pruebas fue en promedio de 4% aproximadamente, menor al 5% requerido.

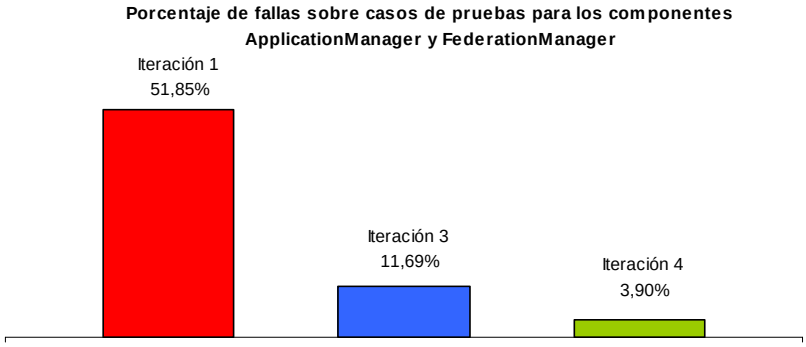


Fig. 5.1 - Muestra la cantidad de fallas sobre casos de pruebas para los componentes ApplicationManager y FederationManager en las iteraciones 1, 3 y 4.

Para los test funcionales se requería un porcentaje menor a un 10 % en la cantidad de fallas sobre casos de pruebas para las funcionalidades que requerían nivel de calidad cinco. Los resultados de las pruebas dieron que en promedio se llegó a un porcentaje aproximado al 10 % y para cada uno de estos casos de uso se llegó a un porcentaje menor o igual al 10 %, cumpliendo con el objetivo planteado.

En la siguiente tabla se detallan los resultados de las pruebas de los principales casos de uso ejecutados al finalizar la cuarta iteración:

Caso de uso	Casos de prueba	Fallas	Fallas /Casos
Rutear aplicación	20	2	10 %
Obtener aplicaciones del usuario	6	0	0 %
Federar Aplicación	16	0	0 %
Desfederar Aplicación	16	1	6.25%
Federar Nodo	11	1	9 %
Desfederar Nodo	8	0	0 %
Actualizar Topología	6	0	0 %
Total de estos casos	83	4	4.08 %

Para las demás funcionalidades del federador de aplicaciones se obtuvieron los siguientes resultados:

Iteración	Funcionalidades	Casos de prueba	Fallas	Fallas /Casos
Fin de tercera iteración	7	23	4	17.39 %
Ultimo test	7	23	0	0 %

Cabe aclarar que por razones de tiempo y prioridad, en estos casos de uso no se realizó un testeo exhaustivo de los mismos.

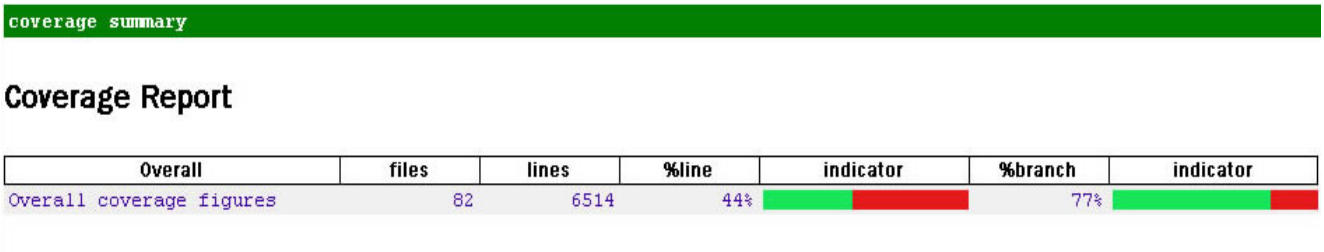
5.3.2. Cubrimiento:

El cubrimiento que se estableció aceptable en cuanto a la Especificación de Requerimientos es tener al menos un caso de prueba por cada requerimiento especificado en el alcance del producto. Por lo visto en las secciones anteriores se puede decir que los requerimientos del sistema fueron cubiertos aceptablemente.

El criterio que se consideró aceptable para el cubrimiento de Casos de Uso es tener casos de prueba para todos los escenarios posibles de cada caso de uso. Para los casos de uso considerados como críticos además se debían generar casos de prueba utilizando la técnica de partición en clases de equivalencia. Como se muestra en la sección 5.2.2, donde se describen los casos de pruebas de las funcionalidades críticas, se cumplió aceptablemente con el criterio de cubrimiento establecido.

El criterio que se estableció en cuanto a componentes fue cubrir todas las funcionalidades de aquellos componentes considerados críticos. Se resolvió establecer además un cubrimiento de un 75% en las ramas de decisión.

Los resultados obtenidos en los tests automáticos dieron que para el componente ApplicationManager el cubrimiento fue de un 35 % de líneas de código y un 79 % de las ramas de decisión. Para el caso del FederationManager el cubrimiento de líneas de código fue un 30% y las ramas de decisión un 63%. En el caso del componente QueryManager el porcentaje del cubrimiento de las líneas de código fue de un 22% y las decisiones fueron cubiertas en un 84%. En promedio los tests automáticos de componentes cubrieron un 44 % de líneas de código y un 77% de decisión como lo muestra la figura 5.2. Con estos datos se puede ver que los resultados se aproximaron al criterio de aceptación establecido para el cubrimiento de componentes.


coverage summary

Coverage Report

Overall	files	lines	%line	indicator	%branch	indicator
Overall coverage figures	82	6514	44%		77%	

Fig. 5.2 - Resultados del cubrimiento promedio de pruebas automáticas para componentes.

En base a los resultados obtenidos se puede concluir que el producto cumple con los requerimientos planteados y posee las características de calidad exigidas.

6. Conclusiones y Trabajo Futuro

6.1. Conclusiones

En este proyecto se presenta la solución alcanzada al problema de la federación de aplicaciones y datos en la red Link-ALL lográndose desarrollar un producto de calidad que cumple con los objetivos planteados.

Debido a que el Federador estaba enmarcado en el contexto de un proyecto de mayor porte existieron restricciones respecto a los recursos informáticos utilizados y en las interfaces brindadas por el sistema. Estas limitaciones implicaron el uso de herramientas open source y el desarrollo basado en la tecnología J2EE.

Fue definida una arquitectura en capas en donde la lógica se encuentra dividida en componentes de acuerdo a su función. En ésta definición se emplearon patrones de diseño que contribuyeron a brindarle al producto características como flexibilidad, extensibilidad y mantenibilidad.

La solución implementada consiste en un Federador de Aplicaciones y Datos integrado a la plataforma Link-ALL. Éste brinda los servicios necesarios para que el acceso a los recursos de la red sea transparente al usuario independientemente de la ubicación física de los mismos.

El Federador ofrece mecanismos que hacen posible la autonomía de asociación de un servidor en la federación. Esto implica que el nodo pueda compartir o no sus recursos así como también asociarse y desasociarse del resto de la red Link-ALL.

Se diseñaron y ejecutaron casos de pruebas manuales y automáticos que permitieron probar las funcionalidades del Federador, principalmente aquellas que requerían un mayor nivel de calidad.

Dentro de las principales dificultades encontradas estuvieron las modificaciones en la plataforma Link-ALL, lo que motivó cambios en el producto para mantener la integración con la misma. Estas modificaciones se debieron principalmente al entorno multi-proyecto en el que se trabajó. A esto se suma una alta curva de aprendizaje de las tecnologías necesarias para el desarrollo del producto que en su mayoría no habían sido utilizadas anteriormente.

El principal aporte del Federador fue hacer tangibles los mecanismos de federación planteados. La integración con la plataforma Link-ALL contribuyó a validar las interfaces con la misma. Asimismo, la interacción con el grupo de desarrollo Link-ALL sirvió de retroalimentación para mejorar la definición y aspectos técnicos del producto.

6.2. Trabajo Futuro

1. **Incorporar nuevas estrategias de actualización de la información de la Federación**

Es posible incorporar nuevas estrategias de actualización de la información de la Federación. Actualmente se provee una estrategia denominada “Fuerza Bruta”. Como su nombre lo indica es una estrategia simple en donde el Servidor de Soporte envía al nodo solicitante toda la información que posee sobre la topología (información de la federación). Una posibilidad de extender y a su vez mejorar el mecanismo de actualización sería incorporar metodologías que permitan enviar menos información al nodo para reducir por ejemplo tráfico en la red.

2. **Posibilidad de reintentar pedidos fallidos a otros federadores**

Una característica interesante sería hacer posible los reintentos de pedidos fallidos a servidores remotos. Esto incrementaría la robustez de los mecanismos de federación al disminuir la probabilidad de fallas en los accesos remotos.

3. **Federación con más de un servidor de soporte**

Se podrían tener varios servidores de soporte, ubicados en distintas áreas geográficas, cada uno con su propio conjunto de nodos federados. Los servidores de soporte podrían sincronizar su información utilizando las operaciones existentes. Sería necesario implementar una nueva estrategia de actualización de la topología para que no se reemplace la información local en los servidores de soporte.

4. **Incorporar nuevas estrategias de ruteo de aplicaciones.**

Actualmente el Federador cuenta solo con una estrategia de ruteo de pedidos de aplicaciones. Se podrían implementar nuevas estrategias que permitan retornar la aplicación “óptima” al solicitante de acuerdo a un determinado criterio, por ejemplo tomando en cuenta prioridades, estadísticas de acceso y características de la aplicación a rutear.

5. **Actualización en línea de modificaciones en el mapa de la federación.**

El federador podría ser capaz de proveer servicios de publicación-suscripción para informar modificaciones ocurridas en el mapa de la federación. Las entidades suscriptas podrían ser eventualmente federadores remotos o herramientas de navegación que recibirían en línea información de las modificaciones ocurridas.

6. **Extender el conjunto de sentencias SQL**

Actualmente el Federador de Datos acepta un conjunto reducido de sentencias SQL. Sería interesante extender este conjunto a cualquier sentencia SQL válida para el manejador de base de datos.

7. **Permitir realizar consultas de actualización y eliminación remotas**

Una mejora de la federación de datos podría incluir la actualización y eliminación de los datos en servidores remotos dado que actualmente estas operaciones solo se realizan en forma local.

8. **Realizar consultas utilizando el framework Hibernate**

Debido al diseño de la arquitectura es posible extender fácilmente el Federador de Datos para permitir manejar consultas HQL. Sería necesario implementar los mapeos que necesita Hibernate además de escribir un adapter que entienda el lenguaje y ejecute las consultas [15].

9. **Incorporar un planificador de pedidos**

Sería posible incorporar al Federador Link-ALL un planificador de pedidos que permita ordenarlos según un determinado criterio como por ejemplo dar preferencia a operaciones locales sobre las remotas.

Para más detalles de las extensiones descriptas referirse al Manual Técnico [23].

7. Referencias

[1] Link-ALL web site http://www.Link-ALL.org, 15/03/2005

[2] Pablo Garbuzi, Federico Piedrabuena, Laura González, Link-ALL System Architecture Document, Versión 1.8, Instituto de Computación, Facultad de Ingeniería, Universidad de la República. Febrero 2005

[3] Amit P. Sheth, James A. Larson, Federated Database Systems for Managing, Distributed, Heterogeneous, and Autonomous Databases, AMC Computing Surveys, Vol.22, No. 3. Setiembre 1990

[4] Java™ 2 Platform Enterprise Edition Specification v1.4 http://java.sun.com/j2ee/, 06/04/2005

[5] Nadir Gulzar y Kartik Ganeshan, Practical J2EE Application Architecture, McGraw-Hill/Osborne, 2003

[6] JBoss web site http://www.jboss.org, 26/04/2005

[7] Getting Starting with Jboss, Luke Taylor and The JBoss Group, 2004

[8] Deepak Alur, John Crupi, Dan Malks, Core J2EE™ Patterns: Best Practices and Design Strategies Second Edition, Prentice Hall, 2003

[9] OPENSYMPHONY web site http://www.opensymphony.com/quartz, 22/01/2005

[10] TouchGraph WikiBrowser V1.02 http://www.touchgraph.com, 05/03/2005

[11] JSQL Parser http://www.sourceforge.net/projects/jsqlparser, 04/04/2005

[12] DJUNIT http://works.dgic.co.jp/djwiki/Viewpage.do?pid=@646A556E6974, 18/11/2004

[13] Vincent Massol, Ted Husted, JUnit in Action, Manning Publications Co, 2004

[14] ECLIPSE http://www.eclipse.org/, 30/03/2005

[15] James Elliott, Hibernate: A Developer's Notebook, O'Reilly 2004

[16] MySQL web site http://www.mysql.com, 15/09/2004

[17] PostgreSQL web site http://www.postgresql.org, 10/02/2005

[18] Documento de Estado del Arte, Fabricio Alvarez, Rodolfo Amador, Proyecto Federador Link-ALL 2004

[19] Documento de Diseño de Casos de Uso, Fabricio Alvarez, Rodolfo Amador, Proyecto Federador Link-ALL 2004

[20] Documentación Técnica, Fabricio Alvarez, Rodolfo Amador, Proyecto Federador Link-ALL 2004

[21] Documento de Verificación, Fabricio Alvarez, Rodolfo Amador, Proyecto Federador Link-ALL 2004

[22] Documento de Requerimientos y Casos de Uso, Fabricio Alvarez, Rodolfo Amador, Proyecto Federador Link-ALL 2004

[23] Manual Técnico, Fabricio Alvarez, Rodolfo Amador, Proyecto Federador Link-ALL 2004

8. Anexos

Anexo 1: Documento de Requerimientos y Casos de Uso

Anexo 2: Documento de Estado del Arte

Anexo 3: Documentación Técnica

Anexo 4: Documento de Diseño de Casos de Uso

Anexo 5: Documento de Verificación

Anexo 6: Manual de Usuario e Instalación

Anexo 7: Manual Técnico

PROYECTO DE GRADO
FEDERADOR LINK-ALL

Requerimientos y Casos de Uso

Mayo 2005

Instituto de Computación - Facultad de Ingeniería
Universidad de la República

Estudiantes: Fabricio Alvarez
Rodolfo Amador

Tutores: Federico Piedrabuena
Raúl Ruggia

Tabla de contenido

1. Actores

Se presentan los actores para el Federador Link-ALL.

1.1. Administrador del Servidor

Es el encargado de dar de alta, baja, monitorear y actualizar un servidor en la federación. Además tiene el control de las aplicaciones, datos y servicios instalados en el servidor.

1.2. Usuario Link All

Es el Usuario de una comunidad Link All, que solicita aplicaciones, servicios y datos al federador a partir de la API o de la interfaz de usuario. Podrá almacenar y acceder a los datos, aplicaciones y servicios públicos de la red y aquéllos privados y/o compartidos sobre los que tenga permisos.

1.3. Usuario Público

Es el Usuario que se conecta a la red Link All a través de un browser de Internet, sin estar registrado en una comunidad pudiendo acceder a los datos y servicios públicos de la red.

1.4. Deployer

Es el Sistema encargado de la instalación y desinstalación de aplicaciones y servicios en los distintos servidores Link All.

2. Diagramas de Casos de Uso

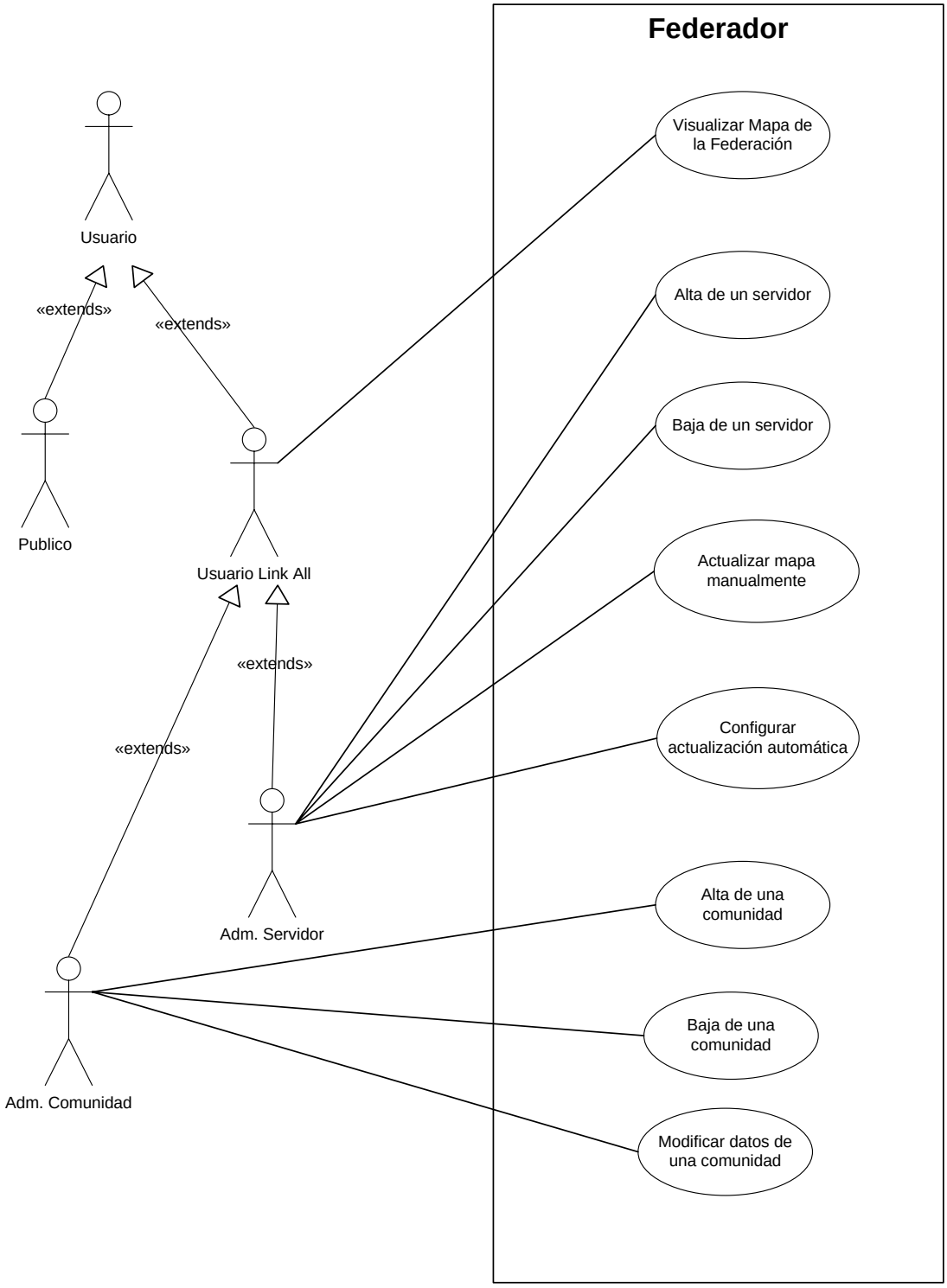


Fig. 2.1 - Casos de uso para la configuración de la federación

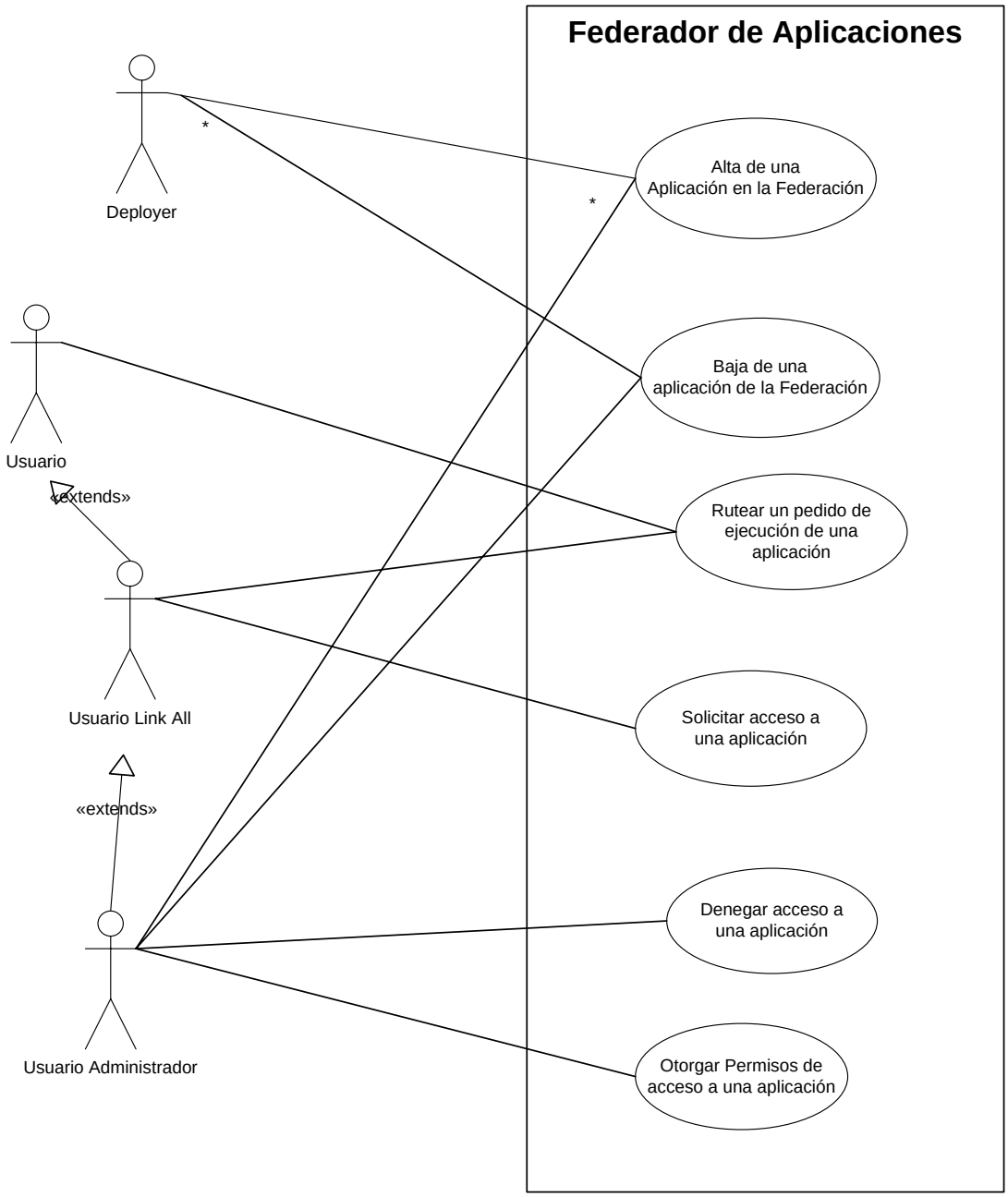


Fig. 2.2 - Casos de uso para la federación de aplicaciones

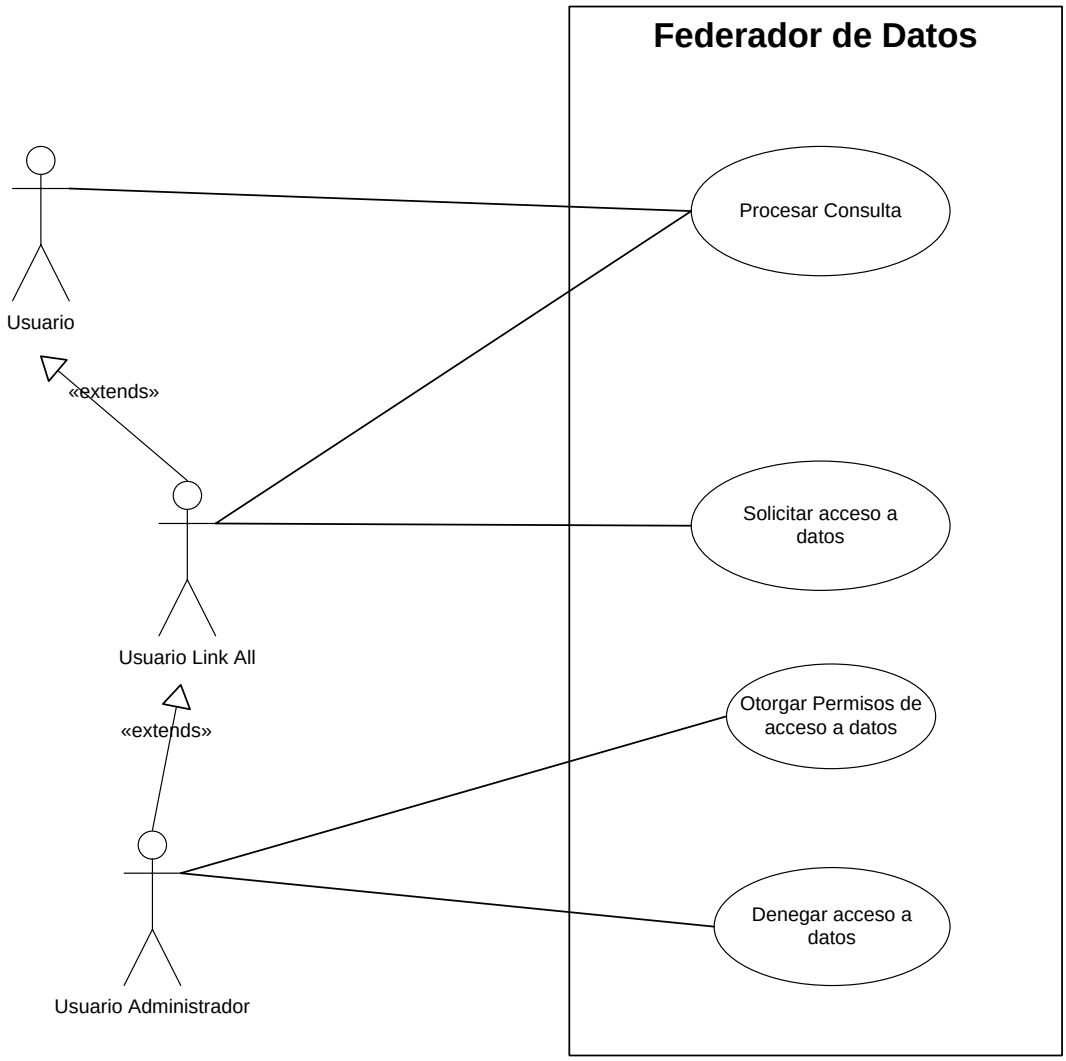


Fig. 2.3 - Casos de uso para la federación de datos

3. Descripción de los casos de uso

3.1. Administración de la Federación

3.1.1. Alta de un servidor en la Federación

Esta operación permite a los servidores integrarse a la Federación Link-ALL. Un determinado servidor puede federarse como servidor de soporte o servidor común. La federación de un nodo de soporte implica que el federador ingrese en la topología del servidor a este nodo y sus comunidades, en cambio un nodo común solicita federarse al servidor de soporte, para esto el nodo debe tener un servidor de soporte configurado con anterioridad, del mismo modo que en el caso anterior se ingresa el nodo y sus comunidades en la topología del servidor de soporte, finalmente cuando el nodo ha sido federado recibe la información de la topología Link-ALL desde el nodo de soporte. Esta funcionalidad es invocada por el administrador del Servidor

3.1.1.1. Actores

Administrador del Servidor.

3.1.1.2. Pre-Condiciones

Plataforma Link All Instalada en el servidor, usuario administrador “logeado” en el sistema.

3.1.1.3. Flujo de Eventos

Acción del Actor	Respuesta del sistema
1. El administrador accede por el browser a la herramienta de configuración del servidor.	
	2. Despliega la página de la herramienta de configuración.
3. Ingresa la opción de dar de alta el servidor a la federación.	
	4. Solicita el ingreso de datos para dar de alta el servidor.
5. Ingresa los siguientes datos: • Código • Nombre • Descripción • Dirección Física: URL del servidor. • Tipo de comunicación. • Tipo de enlace. • Ubicación Geográfica. • Nodo Común	
	6. Valida los datos ingresados. Controla los permisos del usuario para federar servidor.
	7. Envía los datos ingresados y la información de las comunidades al servidor de soporte.
	8. El Servidor de Soporte valida los datos enviados y verifica la validez del certificado.
	9. El Servidor de Soporte ingresa al nodo en la federación e informa al nodo que el alta se realizó correctamente.
	10. El nodo solicita la actualización de la

	topología.
	11. El servidor de soporte envía la información de la topología.
	12. El nodo actualiza la topología local.
	13. El sistema informa al usuario que el alta en la federación se ha realizado con éxito.

3.1.1.4. Flujo de Eventos Alternativo

5A Se federa el servidor de soporte

Acción del Actor	Respuesta del sistema
5A 1. Ingresa los siguientes datos: • Código • Nombre • Descripción • Dirección Física: URL del servidor. • Tipo de comunicación. • Tipo de enlace. • Ubicación Geográfica. • Nodo Común	
	5A 2. Valida los datos ingresados. Controla los permisos del usuario para federar servidor.
	5A 3. El nodo actualiza la topología local, ingresando al nodo y a las comunidades.
	5A 4. El sistema informa al usuario que el alta en la federación se ha realizado con éxito.

6A Los datos ingresados son erróneos

Acción del Actor	Respuesta del sistema
	6A 1. Informa el error y regresa al punto 4.

7A Fallo en el envío de datos

7A 1 No se pudo establecer la conexión con el servidor de soporte

Acción del Actor	Respuesta del sistema
	7A 1. Informa el error, solicitando verificar la dirección del servidor de soporte y regresa al punto 4.

8A Error en la validez del certificado de sitio

Acción del Actor	Respuesta del sistema
	8A 1. Informa el error.
	8A 2. Fin.

8B El servidor ya esta dado de alta en la federación

Acción del Actor	Respuesta del sistema
	8B 1. Informa que existe el servidor ya está federado
	8B 2. Fin.

3.1.2. Baja de un servidor de la Federación

3.1.2.1. Descripción

Un servidor se da de baja de la red Link-ALL, el administrador decide desfederar al servidor. El Federador del nodo informa al servidor de soporte que el nodo ya no está federado, por lo que en el servidor de soporte se elimina de la topología el nodo, las aplicaciones que este haya federado y también se eliminan los permisos sobre los recursos de ese servidor. La baja a realizar será lógica, es decir que se mantendrán los registros para conformar un histórico de los servidores dados de alta. Se almacenará la fecha y hora de baja.

3.1.2.2. Actores

Administrador del Servidor.

3.1.2.3. Pre-Condiciones

Usuario administrador del servidor “logeado” en el sistema.

3.1.2.4. Flujo de Eventos:

Acción del Actor	Respuesta del sistema
1. El administrador accede por el browser a la herramienta de configuración del servidor.	
	2. Despliega la página de la herramienta de configuración.
3. Ingresa la opción de dar de baja el servidor de la federación	
	4. El sistema solicita confirmación de la baja.
5. El administrador confirma la solicitud de baja.	
	6. El servidor notifica al servidor de soporte la baja.
	7. El servidor de soporte registra la baja del nodo ingresando la fecha y hora de baja y el usuario que la realizó. Informa a los servidores suscriptos la baja del servidor.
	8. El servidor de soporte confirma al federador la baja en la federación.
	9. Informa al usuario que el servidor se ha dado de baja.

3.1.2.5. Flujo de Eventos Alternativo

5A El administrador cancela la baja

Acción del Actor	Respuesta del sistema
	5A 1. Retorna al punto 2

6A Fallo en la conexión con el nodo de soporte

Acción del Actor	Respuesta del sistema
	6A 1. Notifica que la baja no pudo ser realizada
	6A 2. Retorna al punto 2.

3.1.3. Visualización del mapa de la Federación

3.1.3.1. Descripción

Despliega la información del estado de la Federación: los servidores disponibles en la federación, qué características tienen, cuáles están conectados, que tipo de datos, servicios y aplicaciones tienen disponibles cada uno.

3.1.3.2. Actores

Administrador del Servidor.

3.1.3.3. Pre-Condiciones

Plataforma Link All Instalada en el servidor, usuario administrador “logeado” en el sistema.

3.1.3.4. Flujo de Eventos

Acción del Actor	Respuesta del sistema
1. El administrador accede por el browser a la herramienta de monitoreo de la federación.	
	2. Despliega la página de la herramienta de monitoreo de la federación.
3. Ingresa la opción de ver el mapa de la federación	
	4. Despliega los servidores disponibles en la federación, las comunidades que tiene cada nodo y las aplicaciones federadas en cada comunidad.

3.1.4. Actualizar mapa de la Federación manualmente

3.1.4.1. Descripción

El administrador del servidor selecciona actualizar el mapa de la federación. El nodo de soporte envía la información y el nodo reemplaza la topología local con la información que le llega del nodo de soporte. La actualización del mapa de la federación se hace desde el servidor de soporte hacia un nodo común, ya que todas las operaciones que un nodo efectúe en la federación son informadas o realizadas contra el servidor de soporte por lo que éste mantendrá actualizada la topología de la federación.

3.1.4.2. Actores

Administrador del Servidor.

3.1.4.3. Pre-Condiciones

Usuario Administrador de la Federación “logeado” en el sistema.

3.1.4.4. Flujo de Eventos

Acción del Actor	Respuesta del sistema
1. El Administrador accede por el browser a la herramienta administrativa.	
	2. Despliega la página de la herramienta administrativa
3. Ingresa la opción actualizar mapa de la federación.	
	4. El Sistema presenta la opciones: • reemplazar mapa local • actualizar mapa local • configurar actualización automática
5. Selecciona la opción reemplazar mapa local	
	6. Solicita al servidor de soporte el mapa de la federación, enviando el certificado de sitio.
	7. Reemplaza el mapa local con el mapa obtenido del servidor de soporte (exceptuando a los datos del nodo local). Se registra la fecha de actualización del mapa y el número de versión del paquete de actualización.
	8. Informa al usuario que el mapa de la federación ha sido reemplazado con éxito.

3.1.4.5. Flujo de Eventos Alternativo

5A Actualizar mapa local

Acción del Actor	Respuesta del sistema
5A.1. Selecciona la opción actualizar mapa local	
	5A.2. Solicita al servidor de soporte actualizar el mapa de la federación enviando el número de versión del último paquete de actualización recibido y el certificado de sitio.
	5A.3. El servidor de soporte determina los paquetes de actualización necesarios para el nodo y los envía.
	5A.4. Procesa los paquetes de actualización recibidos, actualizando el mapa local. Se registra la fecha de actualización del mapa y el número de versión del paquete de actualización.
	5A.5. Informa al usuario que el mapa de la federación ha sido actualizado con éxito.

6A No se pudo establecer la conexión con el servidor de soporte

Acción del Actor	Respuesta del sistema

	6A 1. Informa el error y regresa al punto 2.

6B Error en la validez del certificado de sitio

Acción del Actor	Respuesta del sistema
	6B 1. Informa el error.

7A Nodo no federado

Acción del Actor	Respuesta del sistema
	7A 1. Informa el error.

5B No existen paquetes de actualización disponibles

Acción del Actor	Respuesta del sistema
	1. Informa que el servidor está actualizado

5C Error al procesar los paquetes de actualización de la federación.

Acción del Actor	Respuesta del sistema
	1. Informa el error.

3.1.5. Configurar la actualización automática del Mapa de la Federación.

3.1.5.1. Descripción

Se actualiza automáticamente el mapa de la federación local, desde un servidor de soporte o desde un servidor que brinde este servicio.

3.1.5.2. Actores

Administrador del Servidor.

3.1.5.3. Pre-Condiciones

Usuario Administrador de la Federación “logeado” en el sistema.

3.1.5.4. Flujo de Eventos

Acción del Actor	Respuesta del sistema
1. El Administrador accede por el browser a la herramienta administrativa.	
	2. Despliega la página de la herramienta administrativa
3. Ingresa la opción actualizar mapa de la federación.	
	4. Presenta la opciones: • reemplazar mapa local • actualizar mapa local • configurar actualización automática
5. El Administrador selecciona la opción configurar actualización automática del mapa de la federación.	
	6. Presenta la opciones:

	• reemplazar mapa local • actualizar mapa local • actualización desde el nodo de soporte
7. Selecciona reemplazar o actualizar mapa local	
	8. Solicita el intervalo de tiempo (dias, horas) para realizar la operación solicitada.
9. Ingresa los datos solicitados.	
	10. Valida los datos ingresados y agenda la actualización a realizar.

3.1.5.5. Flujo de Eventos Alternativo

7A Actualizar mapa local desde el servidor de soporte

Acción del Actor	Respuesta del sistema
7A.1. Selecciona la opción actualizar mapa local desde el nodo de soporte.	
	7A.2. Informa la suscripción al servidor de soporte para la actualización on line del mapa de la federación enviando el certificado de sitio.
	7A.3. El servidor de soporte valida el certificado de sitio, registra al nodo en la lista de servidores suscriptos para las actualizaciones on line e informa al nodo.
	7A.4. Informa al usuario que la suscripción se ha realizado con éxito.

10A Los datos ingresados son erróneos

Acción del Actor	Respuesta del sistema
	10A 1. Informa el error y regresa al punto 8.

7B No se pudo establecer la conexión con el servidor de soporte

Acción del Actor	Respuesta del sistema
	1. Informa el error y regresa al punto 6.

7C Error en la validez del certificado de sitio

Acción del Actor	Respuesta del sistema
	1. Informa el error.

7D Nodo no federado

Acción del Actor	Respuesta del sistema
	1. Informa el error.

3.2. Federador de Aplicaciones

3.2.1. Rutear un pedido de ejecución de una aplicación

3.2.1.1. Descripción

Un usuario invoca la ejecución de una aplicación, el Federador rutea el pedido independientemente que la aplicación esté instalada en el servidor donde se hizo el pedido o no y devuelve la URL de la misma. Las aplicaciones pueden ser públicas o privadas, para que un usuario pueda acceder a una aplicación privada, éste deberá tener permisos para su ejecución.

3.2.1.2. Actores

Usuario

3.2.1.3. Pre-Condiciones

El usuario ha ingresado a la página de una comunidad Link All.

3.2.1.4. Flujo de Eventos

Acción del Actor	Respuesta del sistema
1. Selecciona ejecutar una aplicación.	
	2. Verifica los permisos de ejecución del usuario para la aplicación solicitada.
	3. Busca la aplicación en las aplicaciones locales.
	4. Retorna la URL de la aplicación local.

3.2.1.5. Flujo de Eventos Alternativo

1) 3A La aplicación no es local (Aplicación remota)

Acción del Actor	Respuesta del sistema
	3A 1. Busca si el usuario tiene permisos para ejecutar la aplicación en algún servidor de la red Link All.
	3A 2. Envía los certificados de sitio y el rol del usuario al servidor que tiene la aplicación.
	3A 3. El servidor remoto verifica los permisos de ejecución, crea una sesión remota y responde al servidor local.
	3A 4. Retorna la URL de la aplicación remota.

2) 2A Usuario no tiene permiso de ejecución de la aplicación

Acción del Actor	Respuesta del sistema
	2A 1. Informa error indicando que el usuario no tiene permisos para ejecutar la aplicación.

4) 2C Aplicación remota – No se pudo conectar con el servidor

Acción del Actor	Respuesta del sistema

	2C 1. Localiza otro servidor que tenga la aplicación.
	2C 2. Envía los certificados de sitio y de usuario al servidor.
	2D 3. Vuelve al punto 2A 3.

5) 2E No existe identificador de aplicación.

Acción del Actor	Respuesta del sistema
	2E 1. Informa el error al Usuario.

3.2.2. Devolver las aplicaciones de un usuario.

3.2.2.1. Descripción

Un usuario ingresa al sistema y en ese momento se devuelve el perfil del mismo incluyendo las aplicaciones a las que tiene acceso. El Federador es el encargado de devolver las aplicaciones locales y remotas que el usuario puede ejecutar.

3.2.2.2. Actores

Interfaz de usuario.

3.2.2.3. Pre-Condiciones

El usuario ha ingresado a la página de una comunidad Link All.

3.2.2.4. Flujo de Eventos

Acción del Actor	Respuesta del sistema
1. Solicita las aplicaciones de un usuario.	
	2. Busca los grupos del usuario
	3. Devuelve la lista de aplicaciones de las que tiene permiso de ejecución.

3.2.3. Alta de una aplicación en la Federación.

3.2.3.1. Descripción

Una aplicación es exportada (publicada) en la federación. Una vez instalada por el Deployer y dada de alta localmente, la aplicación puede ser compartida a la federación dependiendo de la configuración del federador. Esto consiste básicamente en dar de alta la aplicación en la topología del Servidor de Soporte. El federador debe estar configurado para comportarse de una u otra manera.

3.2.3.2. Actores

Usuario Administrador, Deployer.

3.2.3.3. Pre-Condiciones

Plataforma Link All Instalada en el servidor, usuario administrador “logeado” en el sistema, aplicación instalada en el servidor. El servidor debe tener conexión permanente.

3.2.3.4. Flujo de Eventos

Acción del Actor	Respuesta del sistema
1. Solicita federar aplicación ingresando su código.	
	2. Verifica los permisos del usuario que realiza la operación. Recupera los datos de la aplicación.
	3. Envía los datos de la aplicación al servidor de soporte.
	4. El Servidor de Soporte valida los datos enviados.
	5. El Servidor de Soporte agrega la aplicación a la lista de aplicaciones de ese nodo en la federación, actualiza su mapa de la federación e informa al nodo que el alta de la aplicación se realizó correctamente.
	6. El sistema informa al usuario que la aplicación se federó exitosamente.

3.2.3.5. Flujo de Eventos Alternativo

2A No se encuentra la aplicación o el identificador no es válido.

Acción del Actor	Respuesta del sistema
	2A 1. El sistema informa el error ocurrido.

3A No se pudo establecer la conexión con el servidor de soporte

Acción del Actor	Respuesta del sistema
	3A 1. El sistema informa el error ocurrido.

4A La Aplicación ya esta federada

Acción del Actor	Respuesta del sistema
	4A 1. El sistema informa el error ocurrido.

3.2.4. Baja de una aplicación de la Federación

3.2.4.1. Descripción

Se da de baja una aplicación de la federación. Se eliminará la aplicación de la lista de aplicaciones del servidor.

3.2.4.2. Actores

Usuario Administrador, Deployer.

3.2.4.3. Pre-Condiciones

Usuario administrador del servidor “logeado” en el sistema.

3.2.4.4. Flujo de Eventos

Acción del Actor	Respuesta del sistema
1. Informa el nombre de la aplicación a dar de baja.	
	2. Chequea los permisos del usuario y verifica que la aplicación esté instalada y federada.
	3. El nodo notifica al servidor de soporte la baja de la aplicación.
	4. El servidor de soporte realiza la baja de la aplicación en su topología.
	5. El servidor de soporte confirma al federador la baja en la federación.
	6. Informa que la aplicación se ha dado de baja en la federación.

3.2.4.5. Flujo de Eventos Alternativo

8A Identificador de aplicación no válido

Acción del Actor	Respuesta del sistema
	2A 1 El sistema informa el error ocurrido.

8B Aplicación no federada

Acción del Actor	Respuesta del sistema
	2B 1. El sistema informa el error ocurrido.

9A Fallo en la conexión con el nodo de soporte

Acción del Actor	Respuesta del sistema
	3A 1. El sistema informa el error ocurrido.
	3A 2. Retorna al punto 2.

3.2.5. Solicitar permisos de acceso a una aplicación

3.2.5.1. Descripción

Un usuario solicita acceso para ejecutar una aplicación privada disponible en otro servidor.

3.2.5.2. Actores

Administrador, Usuario Link All.

3.2.5.3. Pre-Condiciones

Usuario “logeado” en el sistema.

3.2.5.4. Flujo de Eventos para el Administrador

Acción del Administrador	Respuesta del sistema
1. El administrador accede por el browser a	

la herramienta de configuración del servidor.	
	2. Despliega la página de la herramienta de configuración.
3. Ingresa la opción de solicitar acceso a una aplicación remota.	
	4. Solicita el nombre de la aplicación y el rol del usuario al que se desea otorgar permiso.
5. Informa el nombre de la aplicación y el rol del usuario.	
	6. Valida los datos ingresados. Se envía el certificado de sitio y el rol del usuario al servidor donde se encuentra la aplicación.

3.2.5.5. Flujo de Eventos Alternativo

6 A. Los datos ingresados no son correctos.

Acción del Administrador	Respuesta del sistema
	6A 1. Informa que el rol o el nombre de la aplicación ingresada no son correctos.
	6A 2. Regresa al punto 4.

3.2.5.6. Flujo de Eventos Alternativo

6 A. Los datos ingresados no son correctos.

Acción del Administrador	Respuesta del sistema
	6A 1. Informa que el rol o el nombre de la aplicación ingresada no son correctos.
	6 A 2. Regresa al punto 4.

6 B. Error en la conexión con el servidor.

Acción del Administrador	Respuesta del sistema
	6B 1. Informa el error en la conexión con el servidor.
	6B 2. Regresa al punto 2

6 C. Nodo no federado.

Acción del Administrador	Respuesta del sistema
	6C 1. Informa el error y regresa a punto 2.

3.2.5.7. Flujo de Eventos para el Usuario Link All

Acción del Usuario Link All	Respuesta del sistema
1. Solicita acceso a una aplicación remota.	
	2. Solicita el nombre de la aplicación al que desea acceder.
5. Informa el nombre de la aplicación.	
	6. Valida los datos ingresados. Se envía el certificado de sitio y el rol del usuario al

	servidor donde se encuentra la aplicación.

3.2.6. Otorgar permisos de acceso a una aplicación.

3.2.6.1. Descripción

El Administrador otorga permisos de acceso a una aplicación privada.

3.2.6.2. Actores

Usuario Developer.

3.2.6.3. Pre-Condiciones

Administrador “logeado” en el sistema.

3.2.6.4. Flujo de Eventos para el Administrador

Acción del Administrador	Respuesta del sistema
1. El administrador decide otorgar permisos de acceso a una determinada aplicación.	
	2. Solicita rol del usuario, servidor del usuario y nombre de la aplicación local.
3. Ingresa los datos solicitados.	
	4. Valida los datos ingresados.
	5. Se registran los permisos de acceso correspondientes en la ACL local. En caso de que el usuario sea de otro servidor se envía una notificación indicando los permisos otorgados.

3.2.6.5. Flujo de Eventos Alternativo

4 A. Los datos ingresados no son correctos.

Acción del Administrador	Respuesta del sistema
	4A 1. Informa el error.
	4A 2. Regresa al punto 2.

5 A. No se puede establecer la conexión.

Acción del Administrador	Respuesta del sistema
	5A 1. Reintenta y en caso contrario informa el error.

3.3. Federador de Datos

3.3.1. Procesar consulta

3.3.1.1. Descripción

Se realizan consultas de modificación locales y selección de datos en la Federación controlando en ésta última la forma en que se retornan los resultados.

3.3.1.2. Actores

Usuario, Servidor Link All.

3.3.1.3. Pre-Condiciones

El usuario tiene permisos para acceder a la funcionalidad.

3.3.1.4. Flujo de Eventos

Acción del Actor	Respuesta del sistema
1. Envía una consulta SQL al federador para procesarla	
	2. Interpreta la consulta
	3. Envía la consulta a la base de datos. (insert, update, delete)
	4. Informa el resultado de la operación.

3.3.1.5. Flujo de Eventos Alternativo

2A Falla interpretación de la consulta.

Acción del Actor	Respuesta del sistema
	3A 1. Despliega el error ocurrido.

2B Usuario no tiene permisos de actualización.

Acción del Actor	Respuesta del sistema
	3A 1. Informa el error ocurrido.

3A Falla la conexión con la base de datos.

Acción del Actor	Respuesta del sistema
	3A 1. Informa el error ocurrido.

3B Consulta de selección (select)

Acción del Actor	Respuesta del sistema
3B 1. Envía una consulta SQL al federador para procesarla	
	3B 2. Determina que la consulta fue realizada por un actor local.
	3B 3. Interpreta la consulta y determina en qué servidores se encuentran los datos

	solicitados.
	3B 4. Realiza subconsultas a los nodos determinados en el punto anterior.
	3B 5. Unifica los resultados parciales para conformar un resultado global.
	3B 6. Retorna el resultado obtenido.

3C La consulta fue realizada por un actor remoto.

Acción del Actor	Respuesta del sistema
	3C 1. Envía la consulta a la base de datos.
	3C 2. Informa el resultado de la operación.

3D Nodo no federado

Acción del Actor	Respuesta del sistema
	3D 1. Informa el error ocurrido.

3E Certificado no válido

Acción del Actor	Respuesta del sistema
	3D 1. Informa el error ocurrido.

4A Falla la conexión con alguno de los servidores.

Acción del Actor	Respuesta del sistema
	4A 1. Retorna el resultado.
	4A 2. Informa que el resultado de la consulta puede no estar completo.

PROYECTO DE GRADO
FEDERADOR LINK-ALL

Estado del Arte

Mayo 2005

Instituto de Computación - Facultad de Ingeniería
Universidad de la República

Estudiantes: Fabricio Alvarez
Rodolfo Amador

Tutores: Federico Piedrabuena
Raúl Ruggia

Tabla de contenido

1. Introducción

En este documento se describen los conceptos básicos de una federación, se presentan las tecnologías utilizadas durante la realización del proyecto y los patrones de diseño que se emplearon en el momento de definir la arquitectura.

2. Federación

2.1. Federación de Datos

Un sistema de bases de datos federadas es una colección de sistemas de bases de datos que colaboran entre sí de una manera autónoma [1].

Algunas de las características que presentan las bases de datos federadas son la distribución de los datos, la heterogeneidad y la autonomía de las bases de datos componentes.

En general en los sistemas de bases de datos federadas la distribución de los datos se da porque la construcción de la federación es a partir de base de datos independientes que necesitaron compartir sus datos por alguna razón.

Los datos pueden estar distribuidos en múltiples bases de datos, las cuales pueden estar en el mismo sistema, en el mismo local o geográficamente separadas, pero interconectadas entre si por sistemas de comunicación. La distribución de los datos puede ser de diferentes maneras, por ejemplo partición horizontal y vertical de la base de datos. Algunos beneficios de la distribución de los datos son la disponibilidad, confiabilidad y la mejora en los tiempos de acceso. La distribución puede ser inducida, es decir, los datos pueden ser distribuidos deliberadamente para sacar ventaja de estos beneficios.

Muchos tipos de heterogeneidades son debidas a diferencias tecnológicas, por ejemplo, diferencias en hardware, software como ser sistemas operativos, y sistemas de comunicación. Los tipos de heterogeneidades en los sistemas de base de datos pueden ser divididos en aquellos que tienen diferencias respecto a la DBMS y aquellos que tiene diferencias en la semántica de los datos. Las heterogeneidades debido a diferencias en DBMSs pueden ser debido a diferencias en estructura, diferencias en restricciones o diferencias respecto a los lenguajes de consultas.

La heterogeneidad semántica ocurre cuando hay un desacuerdo respecto al significado, interpretación, o uso intentado de los mismos datos o datos relacionados.

Existen diferentes tipos de autonomía que pueden tener las bases de datos pertenecientes a una federación.

Autonomía de diseño refiere a la habilidad de un DBMS componente a elegir su propio diseño con respecto a cualquier materia incluyendo los datos a ser manejados, la representación (modelo de datos), la conceptualización o interpretación semántica de los datos, las restricciones, la funcionalidad del sistema.

Autonomía de comunicación refiere a la habilidad de un DBMS componente a decidir si se comunica con otro DBMS componente. Un componente con autonomía de comunicación puede decidir cuando y como responde a un pedido desde otro DBMS componente.

Autonomía de ejecución refiere a la habilidad de un DBMS componente para ejecutar operaciones locales (comandos o transacciones ingresadas directamente por un usuario local) sin interferencia de las operaciones externas (operaciones invocadas por otros DBMSs componentes) y decidir el orden en cual se ejecutarán las aplicaciones externas. Así, un sistema externo no puede forzar una orden o ejecución de los comandos en un DBMS componente con autonomía de ejecución.

Autonomía de asociación implica que un DBMS componente tiene la habilidad para decidir si comparte y cuánto comparte sus funcionalidades y sus recursos, esto incluye la habilidad para asociarse o desasociarse por sí mismo de la federación y la habilidad de un DBMS componente de participar en una o más federaciones.

2.2. Federación de aplicaciones

Durante el estudio del estado del arte, no se encontró material referente a la federación de aplicaciones, pero pueden ser aplicados aquí algunos de los conceptos descriptos anteriormente para la federación de datos. Por ejemplo se puede hacer una analogía entre servidores de aplicaciones y manejadores de bases de datos, entonces se pueden aplicar fácilmente los conceptos de autonomía de comunicación, autonomía de ejecución, autonomía de asociación, heterogeneidad en el hardware y heterogeneidad en el software.

3. Tecnologías

3.1. J2EE

La plataforma J2EE es una especificación que permite desarrollar sistemas y servicios con una arquitectura en varias capas a un bajo costo y de manera poco compleja, asegurando a las aplicaciones que cumplan dicha especificación algunas características como ser alta disponibilidad, seguridad, confiabilidad y escalabilidad. Brinda también flexibilidad a los sistemas implementados para adaptarse a los cambios requeridos de manera sencilla al permitir que las aplicaciones J2EE sean fácilmente instaladas y mejoradas [2].

J2EE alcanza estos beneficios al defender una arquitectura estándar con los siguientes elementos:

- Plataforma J2EE – Una plataforma estándar para mantener las aplicaciones J2EE.
- Test suite de Compatibilidad con J2EE – Una test suite para verificar que un producto J2EE cumple con el estándar de una plataforma J2EE.
- Implementación de Referencia J2EE – Una implementación de referencia para prototipar aplicaciones J2EE y para proveer una definición operacional de la plataforma J2EE.
- Diseños J2EE – Un conjunto de buenas prácticas para desarrollar servicios de múltiples capas y cliente finos.

La plataforma J2EE usa un modelo de aplicaciones distribuido con una arquitectura en capas. La lógica de las aplicaciones se divide en componentes de acuerdo a su funcionalidad y estos componentes que forman una aplicación J2EE pueden estar instalados en diferentes máquinas dependiendo de la capa a la que pertenece.

3.1.1. Aplicaciones J2EE

Las aplicaciones J2EE están formadas por componentes J2EE, un componente es una unidad de software funcional que es ensamblada en una aplicación. El ambiente de ejecución define cuatro tipos de componentes de aplicación de acuerdo al grado de dependencia sobre el servidor J2EE [3]:

- Las aplicaciones cliente (cliente grueso) son programas Java de interfaz grafica de usuario (GUI), que se ejecutan en una PC de escritorio y tienen acceso a todas las facilidades de la capa media de la plataforma. La instalación y gestión de estos componentes no están definidas en la especificación.
- Interfaz de usuario web. Los Applets son componentes GUI que se ejecutan en un navegador web, pero puede ejecutarse en una variedad de artefactos. También pueden utilizarse paginas HTML simples. Estos componentes también se llaman cliente fino.
- Componentes Web. Servlets, paginas creadas con la tecnología Java Server Pages (JSP), filtros web y web events listeners, se ejecutan en un contenedor web y pueden responder a pedidos HTTP desde clientes Web. Estos pueden ser utilizados para generar páginas HTML que son la interfaz de usuario de las aplicaciones. Pueden ser utilizados también para generar XML u otro formato de datos que es consumido por otras aplicaciones componente.
- Enterprise Java Beans (EJB): son componentes que se ejecutan en un ambiente controlado que soporta transacciones. Los EJB generalmente tienen la lógica del negocio para una aplicación J2EE.

Los componentes web y los EJB son instalados, manejados y ejecutados por el servidor J2EE.

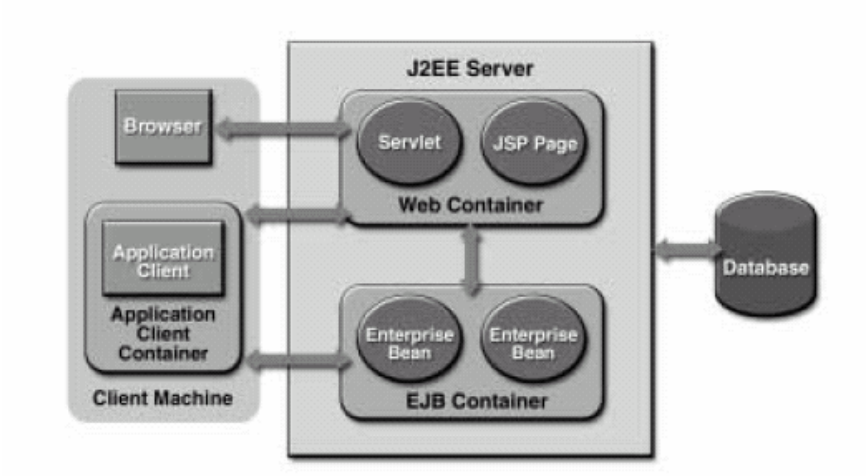
3.1.2. Contenedores:

Los contenedores proveen el soporte en tiempo de ejecución para componentes de aplicaciones, la especificación requiere que este ambiente de ejecución sea compatible con la plataforma Java 2 v 1.4 (J2SE). Los contenedores proveen una vista federada de las Interfaces de Programación de Aplicaciones (API) J2EE a los componentes de aplicación. Interponer un contenedor entre los componentes y los servicios J2EE permiten inyectar transparentemente los servicios definidos por los descriptores de deployment de los componentes, como manejo transaccional declarativo, chequeos de seguridad, poolling de recursos y manejo de estados.

Para que una aplicación cumpla con la especificación J2EE debe soportar un conjunto de servicios estándar, y el contenedor brinda la API necesaria para que los componentes de aplicación utilicen estos servicios.

3.1.3. Servidor J2EE

Los servidores J2EE brindan soporte de instalación, gestión y el ambiente de ejecución para aplicaciones J2EE. Éste servidor provee un contenedor de EJB y un contenedor Web [2].



El contenedor de Enterprise Java Beans (EJBs) maneja la ejecución de EJBs para aplicaciones J2EE. El contenedor Web maneja la ejecución de páginas JSP y servlets. Tanto los enterprise beans como los componentes web, el contenedor de EJBs y el contenedor web se ejecutan en el servidor J2EE.

El contenedor de aplicaciones cliente maneja la ejecución de componentes de aplicaciones cliente, dicho contenedor y la aplicaciones corren en la máquina cliente. El contenedor de Applets consiste en un navegador web y un plug-in java, corriendo en el cliente.

3.1.4. Enterprise Java Beans (EJB)

Los enterprise beans son componentes que encapsulan la lógica del negocio de una aplicación. Son en particular clases java que se ejecutan en el servidor de aplicaciones J2EE. Existen tres tipos de enterprise beans, session beans, entity beans y message-driven beans.

Los session beans (beans de sesión), realizan una tarea para un cliente en el servidor. Para acceder a una aplicación deployada en el servidor, el cliente invoca los métodos del session bean, éste oculta al cliente la complejidad de la ejecución de otras tareas en el servidor.

Existen dos tipos de session beans con estado (statefull) y sin estado (stateless). El estado de un statfull session bean consiste los valores de las variables instanciadas por el bean. El estado es mantenido mientras dura la sesión con el cliente. Un statefull session bean no mantiene un estado por un cliente particular.

Los entity beans (beans entidad) representan objetos de negocio que también poseen un mecanismo de persistencia. Este mecanismo generalmente se realiza por medio de base de datos relacionales. Cada entity bean se corresponden directamente con una tabla en la base de datos y cada instancia del objeto equivale a una fila de esa tabla.

Un message-driven bean es un EJB que permite a las aplicaciones J2EE procesar mensajes de forma asíncrona. Este actúa en un JMS (Java Message Service) listener el cual actúa de manera similar a un listener de eventos, excepto que recibe mensajes en vez de eventos [4]. Los mensajes pueden ser enviados por cualquier componente J2EE, aplicación cliente, otros EJB, o un componente Web.

3.2. Servidores de Aplicación

3.2.1. JBoss

JBoss es un servidor de aplicaciones compatible con la plataforma J2EE, ampliamente utilizado en el mercado y Open Source. El servidor y contenedor JBoss están implementados completamente por plug-ins basados en componentes. Este alto grado de modularidad permite agregar o quitar funcionalidades al servidor según las necesidades del usuario [5].

JBoss agrupa, integra y desarrolla los servicios que la plataforma J2EE involucra para una implementación completamente basada en J2EE [6]. Esto incluye:

- Un contenedor de EJBs
- JMS (Java Messar Service)
- JTS/JTA (Java Transaction Service / Java Transaction API)
- Servlets y JSP
- JNDI (Java Naming and Directory Interface)

Además provee otras características que incluyen la integración con Web Services.

3.3. Web Services

Un Web Service es un sistema de software identificado por una URI, del cual las conexiones y las interfaces públicas están definidas y descriptas usando XML. Su definición puede ser descubierta por otro sistema de software. Estos sistemas pueden interactuar con el Web service de la manera indicada en su definición, usando mensajes basados en XML transportados por protocolos de Internet [7].

Los Web Services exponen funcionalidades útiles a los usuarios Web, a través de un protocolo Web estándar, que en la mayoría de los caso este protocolo es SOAP.

Los Web Services proveen una manera de describir sus interfaces con el detalle suficiente para permitir a un usuario construir una aplicación cliente que se comunique con ellos. Esta descripción es usualmente provista por un documento XML llamado documento de Lenguaje de Descripción de Web Services (Web Services Description Language - WSDL).

Los Web services son registrados en directorios para que potenciales usuarios puedan encontrarlos fácilmente. Esto está realizado con UDDI, (Universal Discovery, Description, and Integración). La UDDI puede ser vista como las páginas amarillas de los Web Services, provee de una base de datos de Web Services que se pueden buscar por tipo de Web Services.

SOAP es un protocolo liviano para intercambio de información en un ambiente distribuido y descentralizado. Es un protocolo basado en XML que consta de tres partes, un sobre que define el marco para describir qué es un mensaje y como procesarlo, un conjunto de reglas para expresar instancia de tipos de datos definidos en una aplicación y una convención para representar llamadas y respuestas a procedimientos remotos.

3.3.1. AXIS

Axis es esencialmente un motor SOAP, un marco para construir procesadores SOAP como clientes, servidores y gateways [7]. La versión actual de Axis está escrita en Java y provee:

- Un servidor que funciona por si mismo.
- Un servidor el cual puede ser instalado en motores de Servlets como Tomcat.
- Soporte extensivo para el lenguaje de descripción del Web service (WSDL).
- Herramientas que generan clases Java desde el WSDL.

3.4. Quartz

Quartz es un sistema de planificación de trabajos de código abierto que puede ser integrado con, o usado junto virtualmente cualquier aplicación J2EE o J2SE. Quartz puede ser utilizado para crear planificaciones simples o complejas y ejecutar decenas, centenas o aún decenas de miles de tareas. Dichos trabajos o tareas pueden ser definidos como componentes Java estándar o EJBs [8].

Un planificador de trabajos básicamente es un sistema que es responsable de ejecutar (o notificar) otros componentes de software cuando se alcanza un tiempo predeterminado (planificado).

Algunas de las características de QUARTZ:

- Es flexible y contiene múltiples paradigmas de uso que pueden ser utilizados juntos o separados, para lograr el comportamiento deseado.
- Es liviano y requiere muy poca instalación y configuración.
- Es tolerante a fallas y puede persistir (recordar) los trabajos planificados a pesar de que el sistema se reinicie.

Porque utilizar QUARTZ y no por ejemplo la clase Timer de Java?

Enumeramos aquí algunas de las razones:

- Timers no tienen mecanismos de persistencia.
- Timers tienen un mecanismos de planificación inflexible (sólo se puede establecer fecha de comienzo y tiempo de repetición)
- Timers no utilizan un Pool de Threads (un thread por timer).
- Timers no tienen un esquema de administración real (es necesario escribir dicho mecanismo para recordar, organizar, recuperar las tareas planificadas).

3.4.1. Resumen:

- Costo: Es un framework open source.

- Documentación: Existe en el Web buena documentación incluyendo ejemplos, javadocs y tutoriales. Existen además foros de usuarios, listas de mails y preguntas frecuentes.
- Código fuente: Está disponible para la descarga.
- No incluye casos de testeo.

3.5. TouchGraph

TouchGraph es un framework que provee una manera de visualizar redes de información interrelacionada. Las redes son dibujadas como grafos interactivos que permiten una variedad de transformaciones. El usuario puede navegar a través de grandes grafos y explorar diferentes maneras de organizar los componentes del grafo [10].

3.6. JSQL Parser

Es una herramienta que permite parsear una sentencia SQL y la traduce en una jerarquía de clases Java [11].

3.7. JUnit

JUnit es un software de código abierto que brinda un framework para desarrollar tests unitarios para Java [9].

Algunas de las características que ofrece JUnit son las siguientes:

- Herramientas que permiten ejecutar los test desplegando los resultados obtenidos (test runners). La distribución de JUnit incluye test runners que se ejecutan desde la línea de comando, o tienen una interfaz AWT o Swing.
- Independencia en la ejecución de los tests.
- Métodos estándar de inicialización y liberación de recursos (setUp y tearDown).
- Variedad de métodos de validación (assert) para diferentes tipos de datos en los resultados esperados.
- Integración con otras herramientas como Ant y Maven, y con diferentes IDE como Eclipse, IntelliJ y JBuilder.

3.8. DJUnit

DJUnit es un plug-in para Eclipse que funciona de manera similar a un TestRunner de JUnit. El reporte del cubrimiento de los tests realizados se realiza utilizando JCoverage, y DJUnit presenta esta información en Eclipse [12].

4. Patrones de Diseño

¿Qué es un patrón de diseño?

Un patrón permite documentar un problema recurrente y su solución en un contexto particular, posibilitando comunicar diseño y buenas prácticas a otros. Los patrones son observados a través de la experiencia, identificados y documentados en una forma que sea fácil de compartir, discutir y entender.

A continuación se detallan algunos patrones comunes conjuntamente con algunos beneficios y posibles escenarios de aplicación.

4.1.1. Business Delegate

Este patrón pretende desacoplar a los componentes de negocio del código que los utiliza. El patrón administra la complejidad de la búsqueda distribuida y manejo de excepciones, pudiendo adaptar la interfaz del componente de negocio a una interfaz simple para utilizar.

4.1.2. Service Locator

El patrón centraliza las búsquedas de objetos de servicio distribuidos, provee un punto centralizado de control y puede actuar como caché que elimina búsquedas redundantes. Puede también encapsular procesos de búsqueda específicos para ciertos vendedores.

4.1.3. Facade

Este patrón pretende proveer una interfaz unificada y simplificada a un conjunto de interfaces en un subsistema. Describe una interfaz de alto nivel que hace al subsistema fácil de usar.

Beneficios de este patrón:

- Provee una interfaz simple a un subsistema complejo sin reducir las opciones provistas por el subsistema.
- Oculta al cliente la complejidad de los subsistemas componentes.
- Promueve el bajo acoplamiento entre el subsistema y los clientes.

Escenarios aplicables:

- Se necesita proveer una interfaz simple a un subsistema complejo.
- Si existen varias dependencias entre clientes y las clases de implementación de una abstracción.

4.1.4. DAO

Este patrón permite abstraer y encapsular todos los accesos a la fuente de datos. El DAO administra la conexión con la fuente de datos, además de obtener y almacenar datos.

4.1.5. Value List Handler

Este patrón permite asegurar que una determinada clase tiene solo una instancia y provee un punto de acceso global de acceso a la misma. Asegura que todos los objetos que utilizan una instancia de esta clase están usando la misma instancia.

4.1.6. Singleton

Este patrón permite asegurar que una determinada clase tiene solo una instancia y provee un punto de acceso global de acceso a la misma. Asegura que todos los objetos que utilizan una instancia de esta clase están usando la misma instancia.

Beneficios de este patrón:
- Controla el acceso a una sola instancia de la clase.
- Reduce el uso del espacio de nombres.
- Puede también permitir un número variable de instancias.

Escenarios aplicables:
- El escenario de uso más apropiado para el patrón Singleton es cuando una sola instancia de una clase debe ser accesible a todos los clientes desde un punto de acceso conocido.

4.1.7. Factory Method

Este patrón pretende definir una interfase para crear un objeto pero dejando que a la subclase decidir que método instanciar. El cliente del Factory Method nunca necesita conocer la clase concreta que ha sido instanciada y retornada, el mismo sólo conoce acerca de la interfaz abstracta publicada.

Beneficios de este patrón:
- Remueve la necesidad de ligar clases específicas en el código de la aplicación.código sólo interactúa con la interfase, por lo tanto lo realizará con cualquier clase que implemente dicha interfaz.
- Dado que crear objetos dentro de una clase es más flexible que hacerlo directamente, el patrón permite a la subclase proveer una versión extendida de un objeto.

Escenarios aplicables:
- Una clase no es capaz de anticipar la clase de objetos que necesita crear.
- Una clase quiere que sus subclases especifiquen los objetos que instancia.

4.1.8. Strategy

Este patrón busca definir un conjunto de funcionalidades, encapsularlas, y hacerlas intercambiables. El patrón permite a la funcionalidad variar independientemente de los clientes que la utilizan.

Beneficios de este patrón:
- Es un opción en lugar de usar subclases.
- Define cada comportamiento dentro de su propia clase, eliminando la necesidad de sentencias condicionales.
- Hace más fácil extender e incorporar nuevo comportamiento sin cambiar la aplicación.

Escenarios aplicables:
- Múltiples clases difieren sólo en sus comportamientos.
- Se necesitan diferentes variaciones de un algoritmo.
- Un algoritmo que usa datos desconocidos al cliente.

4.1.9. Proxy

Este patrón pretende crear un subsistuto para otro objeto para controlar el acceso a él. Las implementaciones más comunes son el proxy remoto y virtual.

Beneficios de este patrón:
- El Proxy remoto puede ocultar el hecho de que la implementación reside en otro espacio de direcciones.
- El Proxy virtual pude ejecutar optimizaciones, por ejemplo crear objetos bajo demanda.

Escenarios aplicables:

- El patrón es apropiado cuando una mas versátil o sofisticada referencia un objeto se necesita, en lugar de un simple puntero.

4.1.10. Decorator

Una alternativa a crear subclases es extender la funcionalidad, el patrón Decorator o Wrapper permite agregar responsabilidades adicionales a un objeto dinámicamente.

Beneficios de este patrón:

- Provee más flexibilidad que la herencia estática
- Simplifica la codificación permitiendo desarrollar una serie de clases específicas para ciertas funcionalidades, en lugar de codificar todo el comportamiento dentro del objeto.
- Mejora la extensibilidad del objeto, dado que los cambios son realizados codificando nuevas clases.

Escenarios aplicables:

- Si se desea transparentemente y dinámicamente asignar responsabilidades a objetos sin afectar otros objetos.
- Si se desea asignar responsabilidades a un objeto que puede cambiar en el futuro.

5. Referencias

[1] Amit P. Sheth, James A. Larson, Federated Database Systems for Managing, Distributed, Heterogeneous, and Autonomous Databases, AMC Computing Surveys, Vol.22, No. 3. Setiembre 1990

[2] Java™ 2 Platform Enterprise Edition Specification v1.4, http://java.sun.com/j2ee/, 06/04/2005

[3] Bodoff, Green, Haase, Jendrock, Pawlan, Stearns, The J2EE Tutorial, Addinson Wesley, 2002

[4] The Java Message Service Tutorial, http://java.sun.com/products/jms/tutorial/index.htm, 11/03/2005

[5] JBoss web site http://www.jboss.org, 20/10/2004

[6] Scott Stark and The JBoss Group, JBoss Administration and Development, Jboss Group, 2003

[7] Nadir Gulzar and Kartik Ganeshan, Practical J2EE Application Architecture, McGraw-Hill/Osborne, 2003

[8] OPENSYMPHONY web site http://www.opensymphony.com/quartz, 22/01/2005

[9] Vincent Massol, Ted Husted, JUnit in Action, Manning Publications Co, 2004

[10] TouchGraph WikiBrowser V1.02 http://www.touchgraph.com, 05/03/2005

[11] JSQL Parser http://www.sourceforge.net/projects/jsqlparser, 04/04/2005

[12] DJUNIT http://works.dgic.co.jp/djwiki/Viewpage.do?pid=@646A556E6974, 18/11/2004

[13] Deepak Alur, John Crupi, Dan Malks, Core J2EE™ Patterns: Best Practices and Design Strategies Second Edition, Prentice Hall, 2003

PROYECTO DE GRADO
FEDERADOR LINK-ALL

Documentación Técnica

Mayo 2005

Instituto de Computación - Facultad de Ingeniería
Universidad de la República

Estudiantes: Fabricio Alvarez
Rodolfo Amador

Tutores: Federico Piedrabuena
Raúl Ruggia

Tabla de contenido

1. Federador Link-ALL

El Federador Link-ALL es una aplicación que permite la manipulación coordinada y controlada de los recursos de la red Link-ALL. Este componente es responsable de la federación de servidores, aplicaciones y datos. Los servidores federados podrán así compartir sus recursos (aplicaciones y datos) al resto de las comunidades de la federación, teniendo la facultad de definir políticas de seguridad sobre el acceso a los mismos.

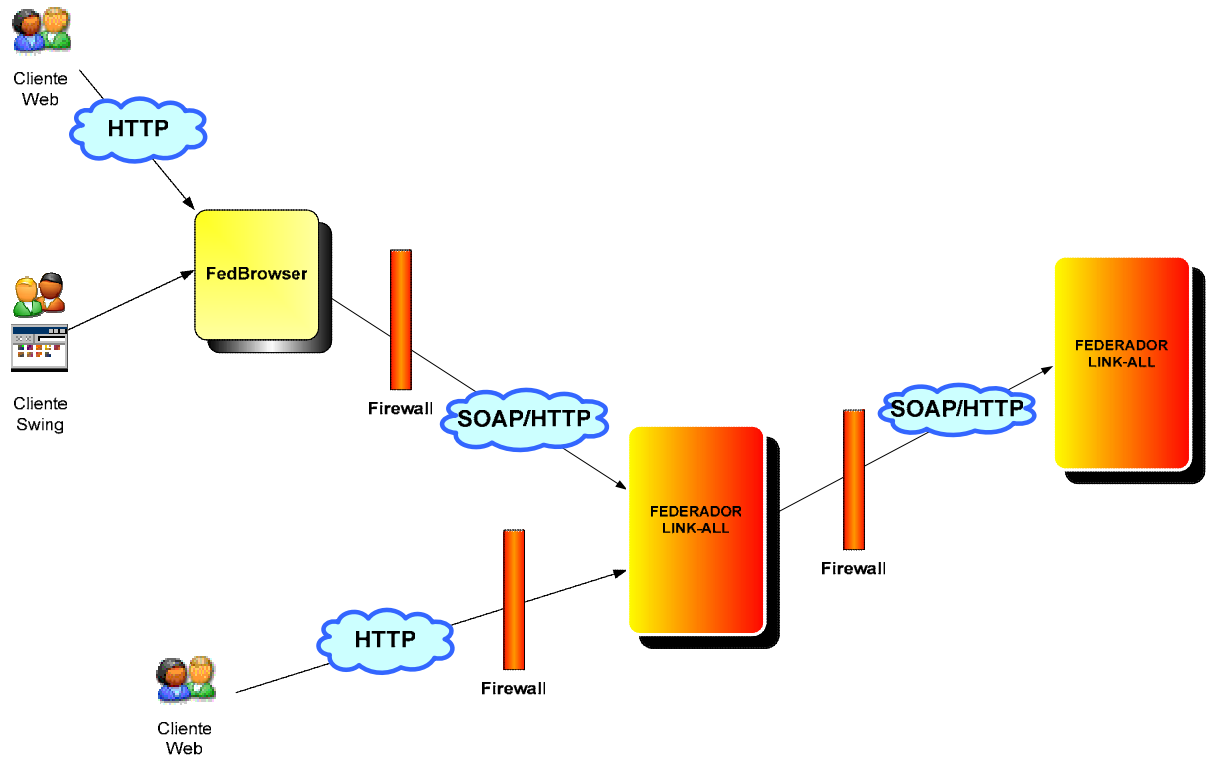


Fig. 1.1 - Vista de instalación, Federador Link-ALL.

El Federador Link-ALL está formado por dos componentes principales: el Federador de Aplicaciones y el Federador de Datos.

En la definición de la arquitectura tanto del Federador Link-ALL como de los utilitarios se tomó como referencia el modelo de Java 2 Enterprise Edition (J2EE) [1] para sistemas de información basados en computadora. De la misma forma en el diseño de los componentes se destaca el uso de patrones, con el objetivo de incorporar buenas prácticas probadas en sistemas reales y así contribuir a lograr una solución flexible, robusta y fácil de extender [2].

1.1. Arquitectura del Federador Link-ALL

A continuación se describe la arquitectura global del sistema construido. Para describirla se utiliza el conjunto de vistas definido por el proceso RUP (4+1 Architecture Views of Rational Unified Process).

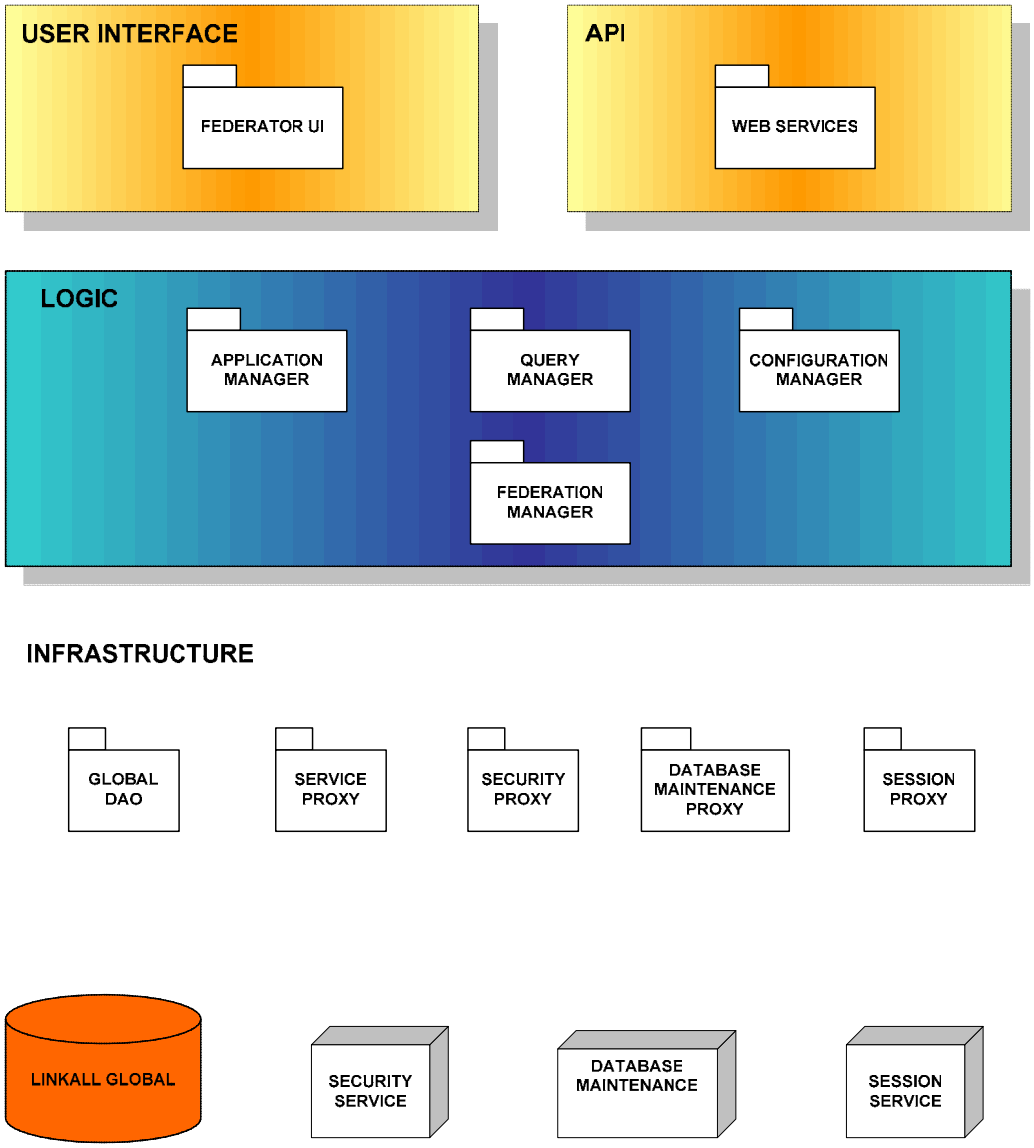


Fig. 1.2 - Arquitectura del Federador Link-ALL.

El Federador Link-ALL presenta una arquitectura de tres capas en donde la lógica se encuentra dividida en componentes de acuerdo a su función.

En la capa del cliente (no representada en la figura) se encuentran los componentes que se ejecutan en la máquina del cliente. Estos componentes son los clientes Web (clientes finos) y las aplicaciones que consumen los Web Services ofrecidos por el Federador Link-ALL. Los clientes Web son clientes HTML híbridos ya que poseen tecnología JavaScript. Esta tecnología provee cierta inteligencia a las páginas posibilitando realizar validaciones básicas (por ejemplo chequeo del formato de datos ingresados y campos obligatorios) antes que la información sea enviada al Federador. Esto tiene como ventaja la disminución de la carga del servidor Web donde se encuentra y el tráfico en la red (debido a que se necesita enviar menos información). Los clientes que consumen los Web Services ofrecidos por el Federador se encuentran actualmente representados por federadores remotos y

herramientas de navegación como es FedBrowser, pero eventualmente podría ser cualquier aplicación capaz de consumir dichos Web Services.

En la capa de presentación se encuentran los Web Services de apoyo a la Federación así como también Servlets y páginas JSP que generan las páginas y contenido dinámico de las mismas.

En la capa lógica se encuentran los componentes de negocio que permiten implementar la lógica del Federador Link-ALL, es decir sus funcionalidades. Los componentes de esta capa son Enterprise Java Beans (EJBs) y Plain Java Objects (POJOs). Los EJBs utilizados son Session Beans sin estado que actúan a modo de fachada, manejando las conversaciones con los clientes.

En la capa de infraestructura se encuentran los componentes de acceso a recursos como la base de datos Global de Link-ALL y otros componentes externos al Federador como los servicios de seguridad, sesión y acceso a la base de datos Administrativa. En esta capa se encuentran también los proxies que enmascaran la comunicación con los federadores remotos.

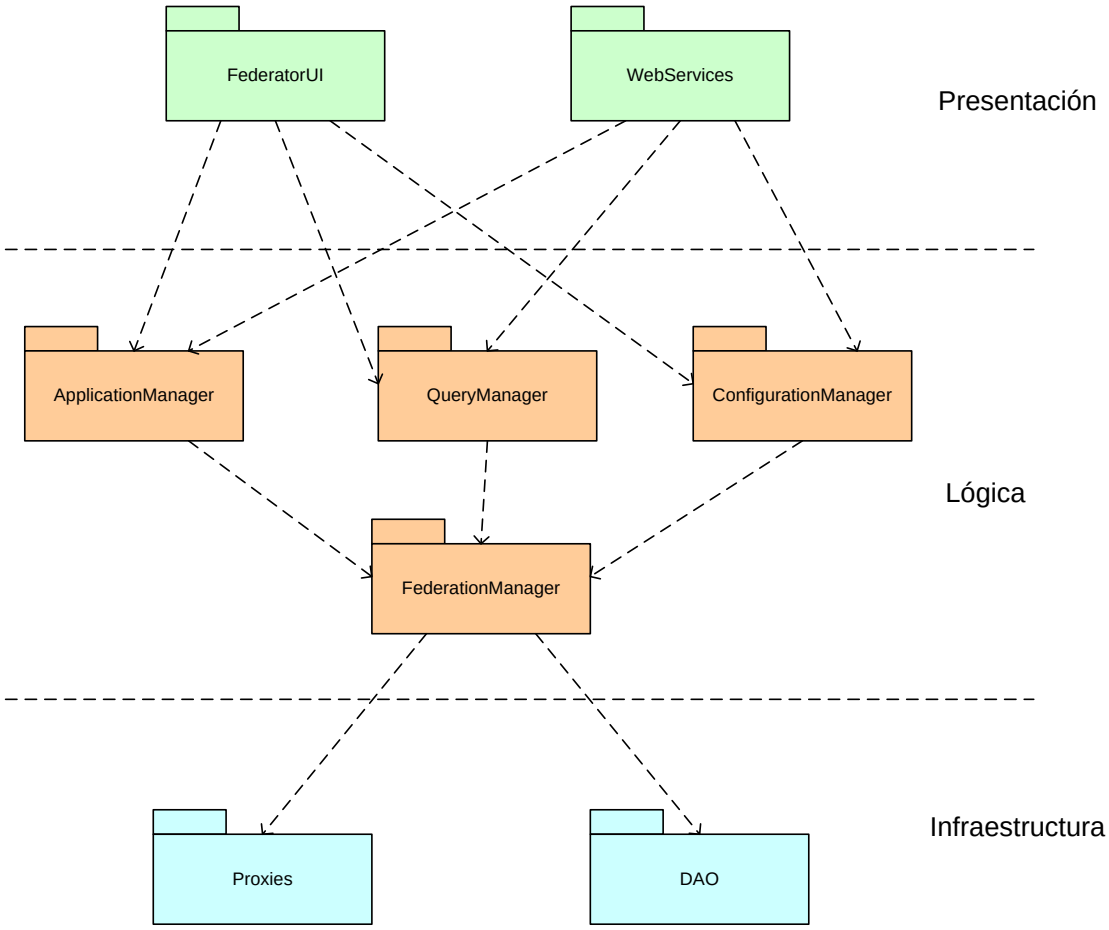


Fig. 1.3 - Vista lógica del Federador Link-ALL.

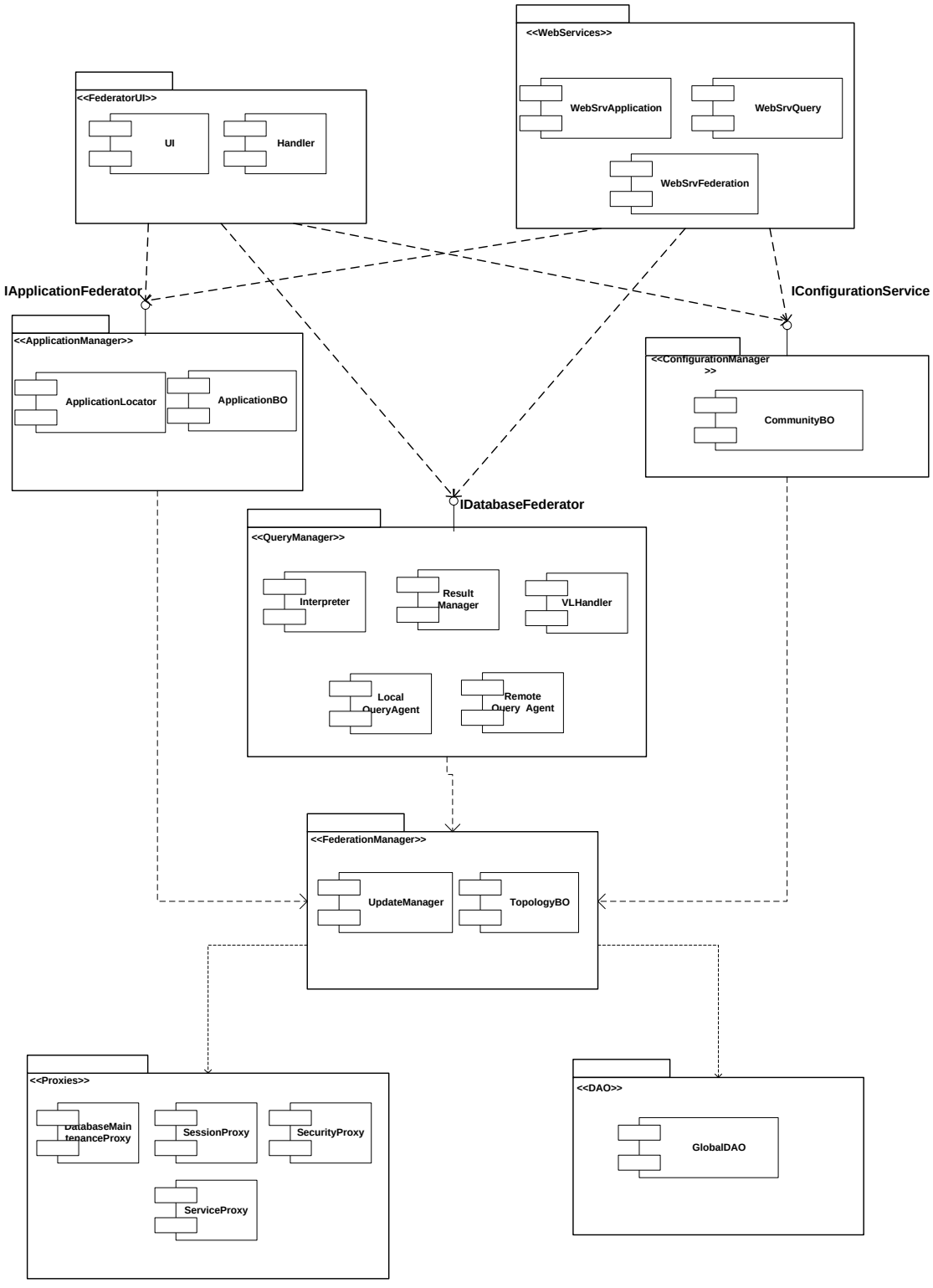


Fig. 1.4 - Vista de Implementación del Federador Link-ALL.

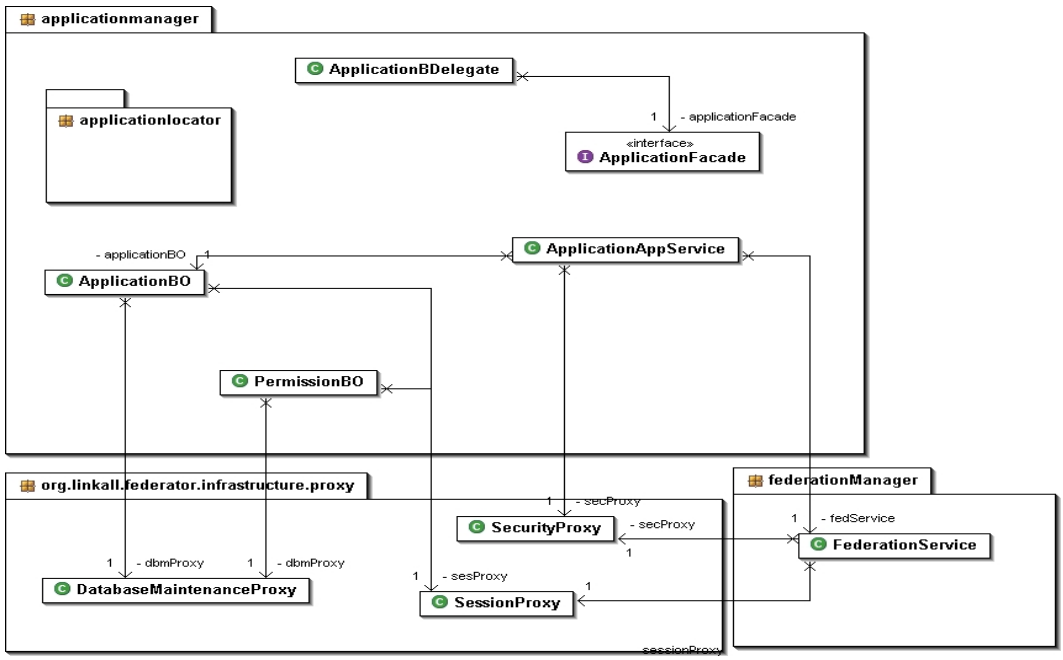
1.2. Subsistemas del Federador de Aplicaciones

El Federador de Aplicaciones es uno de los dos componentes más importantes del Federador Link-ALL. Está compuesto por los subsistemas ApplicationManager y FederationManager, donde éste último brinda también los servicios de federación al Federador de Datos. En las secciones siguientes se proporcionará una descripción de los mismos.

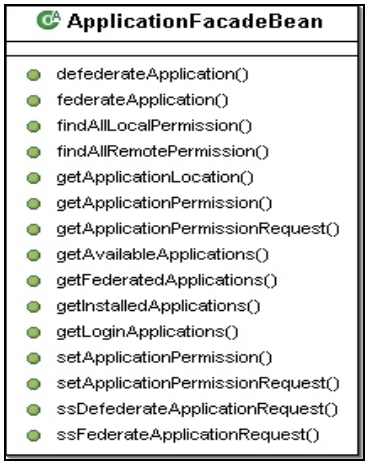
1.2.1. ApplicationManager

Este subsistema brinda los servicios necesarios para realizar la gestión de aplicaciones. Entre sus principales funciones se encuentran la de permitir la federación y desfederación de aplicaciones instaladas en el servidor Link-ALL, obtener las aplicaciones que tiene acceso un determinado usuario y rutear pedidos de aplicaciones de manera transparente.

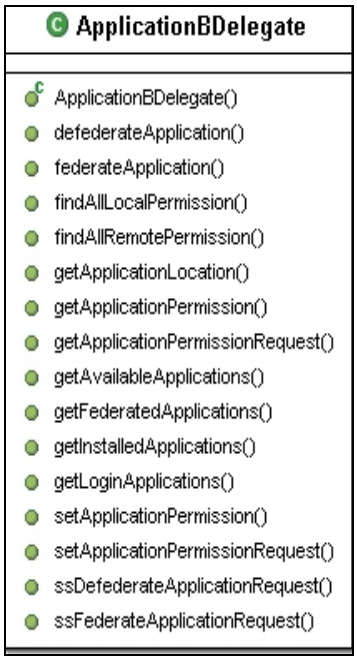
Este componente se integra al subsistema FederationManager (que se detallará más adelante) quien le provee de servicios relativos a la federación. Cuenta además con un SessiónBean sin estado que maneja las conversaciones con los clientes y objetos de negocio que brindan operaciones básicas sobre las aplicaciones.



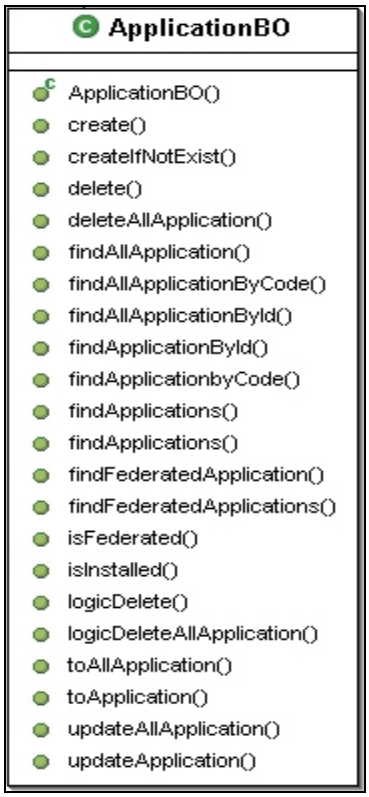
A continuación se presenta un detalle de las principales clases que componen el subsistema ApplicationManager:



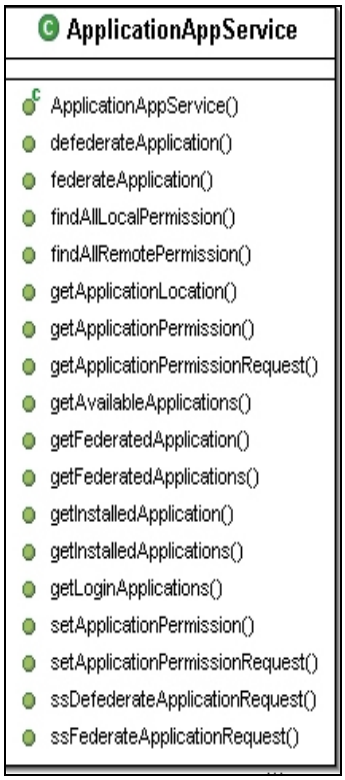
Session Bean de manejo de Aplicaciones.
Es un Session Bean sin estado que actúa como fachada. Su principal objetivo es brindar flexibilidad y claridad en el diseño ya que desacopla el cliente de los objetos de negocio de bajo nivel (por ejemplo ApplicationBO), conteniendo y centralizando el acceso a los mismos.



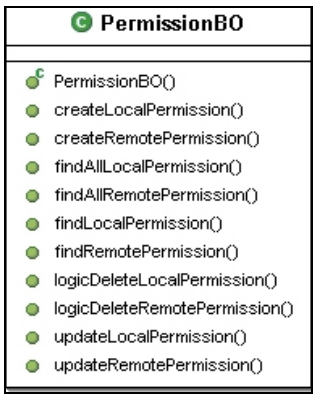
El objetivo principal del ApplicationBDelegate es el de ocultar la complejidad de la comunicación remota con los servicios de negocio que manejan aplicaciones.
Realiza el acceso a la fachada del Federador de Aplicaciones utilizando para ello el componente Service Locator y encapsula los errores que se produzcan al utilizar dicho componente.
El cliente podría acceder a la fachada directamente a través de la JNDI pero es recomendable utilizar esta clase ya que además de reducir el acoplamiento cliente-negocio brinda caché de referencias remotas, lo cual mejora la performance.



ApplicationBO es un objeto de negocio de bajo nivel encargado de encapsular y administrar los datos referidos a aplicaciones conjuntamente con su comportamiento y persistencia.
Es un objeto reutilizable que busca separar la lógica de persistencia de la lógica de negocio.



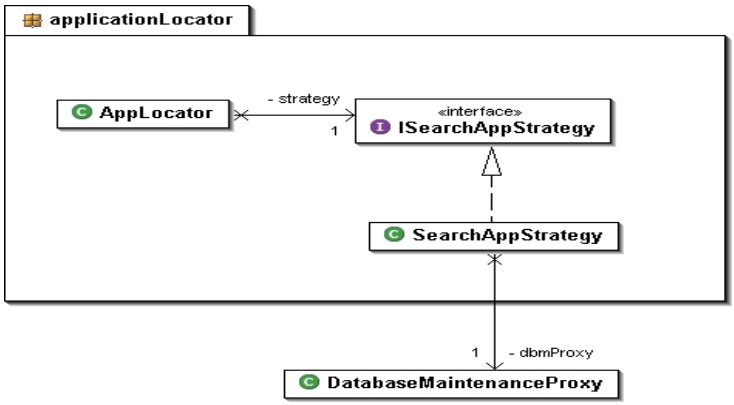
Esta clase implementa la lógica del subsistema ApplicationManager que involucra a varios objetos de negocio. Dicha lógica no es incluida en la fachada (ApplicationFacade) dado que puede ser necesaria para varias fachadas, con la consiguiente duplicación de código. De la misma forma no es incluida en los objetos de negocio de bajo nivel (ApplicationBO, NodeBO, TopologyBO) dado que incrementa el acoplamiento entre ellos, reduciendo la reusabilidad y mantenibilidad del código.



PermisssionBO es un objeto de negocio de bajo nivel encargado de encapsular y administrar los datos referidos a permisos conjuntamente con su comportamiento y persistencia.
Brinda operaciones básicas sobre los permisos de acceso a aplicaciones y tablas de la Federación.

1.2.1.1. ApplicationLocator

Este subsistema es el responsable de la ubicación de las aplicaciones, por lo tanto cumple un rol fundamental en el caso de uso getApplicationLocation. Su objetivo principal es hacer posible la fácil inclusión de nuevas estrategias de localización de aplicaciones.



Interfaz que deben cumplir las estrategias de localización de aplicaciones.

Estrategia de localización por defecto. Implementa la interfaz ISearchAppStrategy y consiste básicamente en buscar la aplicación solicitada (dado su código) en las aplicaciones locales, si no se encuentra se busca dicha aplicación en las remotas al nodo (una vez escogida la aplicación se hace el pedido al nodo remoto para que éste retorne su ubicación). Notar que la aplicación podría estar instalada y federada por más de un nodo. Esta estrategia devuelve la ubicación de la primera aplicación encontrada (o sea que coincide el código).
Estrategias de ruteo de aplicaciones más complejas podrían contemplar otras variables para así retornar la ubicación de la aplicación que sea más conveniente para el usuario que la solicita. Como ejemplo se podría tomar el nodo con mejor ancho de banda, más cercano geográficamente, etc y que tiene federada dicha aplicación.

Localizador de aplicaciones. Su principal objetivo es obtener la ubicación de la aplicación solicitada, utilizando para ello la estrategia que ha sido configurada previamente.

1.3. Subsistemas del Federador de Datos

El Federador de Datos es el otro componente fundamental del Federador Link-ALL. Sus principales subsistemas son QueryManager y FederationManager.

1.3.1. QueryManager

La principal función de éste subsistema es la de permitir la federación de datos existentes en el servidor Link-ALL. Esto incluye la actualización de datos locales y la posibilidad de consultar datos distribuidos a lo largo de toda la federación de manera transparente al usuario. Éste componente se integra al subsistema FederationManager quien le provee servicios de acceso a la federación.

Para cumplir con sus funciones el subsistema cuenta con un Session Bean sin estado que actúa como fachada para los servicios brindados por QueryService; un generador automático de identificadores para las nuevas consultas; un despachador de consultas remotas (QueryRemoteDispatcher) y siete subsistemas más que se detallarán posteriormente (agent, caché, interpreter, datadirectory, security, resultmanager, vlhandler).

El subsistema Agent está compuesto por los agentes de consulta. El agente de consultas locales (LocalQueryAgent) tiene como objetivo atender las consultas locales (contexto local, o sea consultas realizadas por ejemplo por aplicaciones) que llegan al Federador. Si la consulta es de selección, puede llegar a consultar a la base de datos Global utilizando un adapter JDBC (DefaultJDBCAdapter) y consultar a los nodos remotos utilizando al despachador de consultas remotas (QueryRemoteDispatcher). El agente de consultas remotas (RemoteQueryAgent) es el encargado de atender las consultas remotas (contexto remoto, o sea consultas recibidas por ejemplo de nodos

remotos) que llegan al Federador. Realiza la consulta recibida localmente utilizando al adapter JDBC y luego retorna los resultados. Ambos agentes implementan la interfaz QueryAgent que especifica dos métodos: uno para procesar consultas y otro para obtener una determinada página de una consulta realizada.

El subsistema Caché tiene como objetivo proveer servicios de caching al Federador de Datos. Cuenta con dos componentes principales: LocalCache y RemoteCache. El primero se utiliza para cachear las consultas recibidas por el Federador de Datos. Almacena información acerca de las consultas que el Federador recibió (ya sea de contexto local o remoto) incluyendo la colección de páginas. RemoteCache almacena información acerca de las consultas que el Federador de Datos realizó a federadores remotos (por ejemplo incluye el identificador de consulta que le dio el federador remoto, número de secuencia actual, host y puerto donde se encuentra dicho federador, etc.).

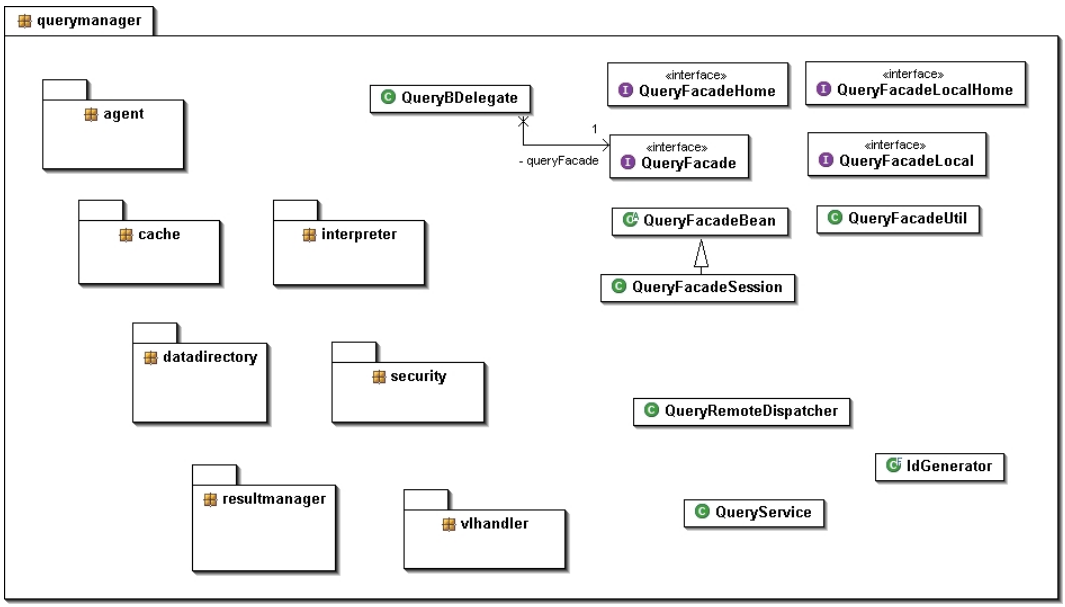
El subsistema Interpreter tiene como principal objetivo la interpretación de la consulta recibida por el Federador. El objeto SqlHelper es el encargado de parsear las consultas SQL, transformando la cadena de caracteres de la consulta en un objeto que la representa y viceversa.

El directorio de datos, DataDirectory, tiene como objetivo brindar al Federador de Datos información acerca de la distribución de datos en la federación. Es decir en base a una determinada consulta y un determinado criterio, es capaz de informar adonde realizar la consulta (que servidores son potenciales proveedores de datos). El directorio de datos implementado actualmente en el Federador utiliza un criterio de decisión basado en permisos, es decir que retorna los servidores a los que se debería enviar la consulta fijándose para eso en los permisos que posee el usuario sobre las tablas remotas involucradas en la consulta.

El subsistema Security tiene como principal componente a DataSecurityManager. Este es el encargado de brindar funcionalidades que permitan apoyar el esquema de seguridad del Federador de Datos (a nivel de software). Su funcionalidad principal es la de reescribir consultas anexándole las restricciones que correspondan.

ResultManager es el subsistema encargado de la administración de resultados. En una consulta de selección pueden existir eventualmente muchos proveedores de datos, por lo tanto también podrán existir muchos resultados que necesitarán ser integrados y divididos en páginas. Esto hace imperativa la necesidad de un componente que administre tal cantidad de resultados, desligando al resto del Federador de Datos de tales funciones. Es aquí entonces que el ResultManager cumple su rol clave.

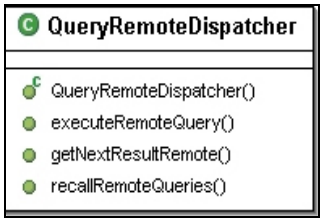
El subsistema VLhandler es el administrador de listas de valores (lista de filas). Está compuesto por clases e interfaces que permiten implementar el patrón Value List Handler [2]. Entre sus componentes se encuentran adapters para realizar consultas utilizando JDBC e Hibernate [3] y además funcionalidades adicionales de caché de las páginas de resultados almacenadas en memoria.



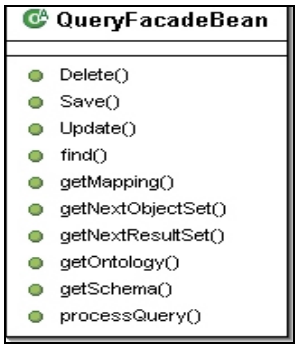
El objetivo principal del QueryBDelegate es el de ocultar la complejidad de la comunicación remota con los servicios de negocio que manejan consultas.

Realiza el acceso a la fachada del Federador de Datos utilizando para ello el componente Service Locator y encapsula los errores que se produzcan al utilizar dicho componente.

QueryService es una clase Java encargada de los servicios internos del Federador de Datos. Los dos más importantes son el de procesar consultas y obtener páginas.



Este componente es el encargado de despachar y administrar las consultas de datos a nodos remotos.

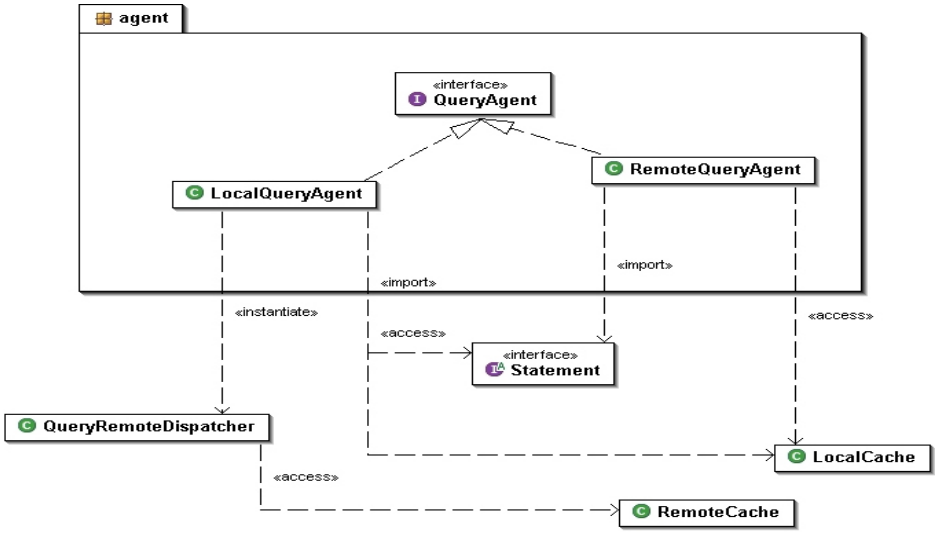


Session Bean de manejo de consultas.

Es un Session Bean sin estado que actúa como fachada. Su principal objetivo es brindar flexibilidad y claridad en el diseño ya que desacopla el cliente de los objetos de negocio de bajo nivel, conteniendo y centralizando el acceso a los mismos.

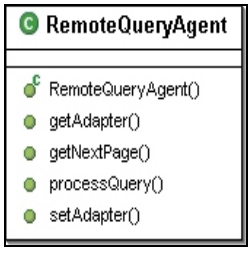
1.3.1.1. Agent

Subsistema cuyos componentes principales son los agentes de consulta. Existen dos agentes uno local y otro remoto, cuyo cometido son atender las consultas locales y remotas respectivamente.



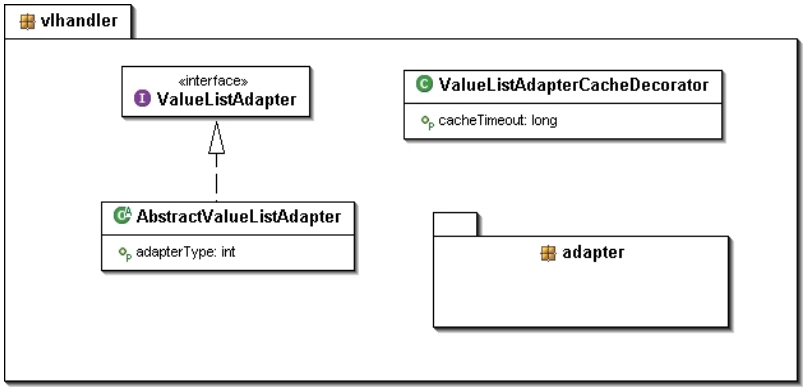
Interfaz que deben cumplir los agentes de consulta. Posee dos métodos, uno para procesar consultas y otro para obtener páginas.

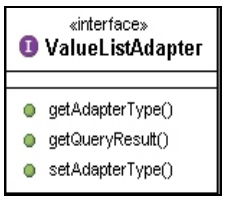
Agente de consultas locales. Atiende las consultas locales que llegan al Federador (consultas con contexto local).



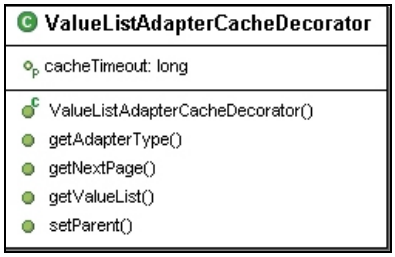
Agente de consultas remotas. Atiende las consultas remotas que llegan al Federador (consultas con contexto remoto).

1.3.1.2. Value List Handler (Vlhandler)





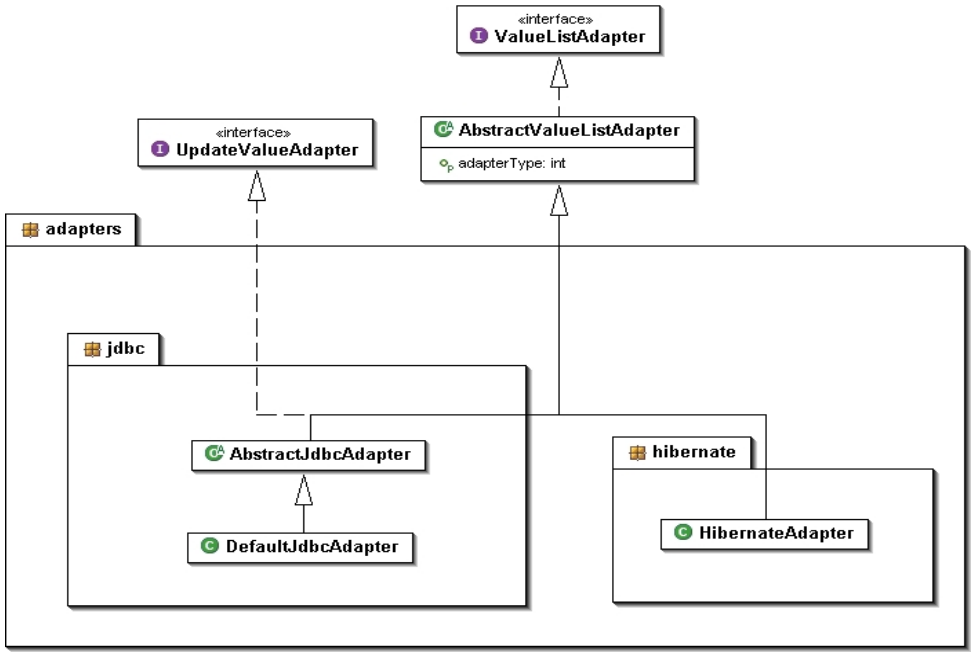
Adapter del manejador de listas de valores (manejador de páginas).

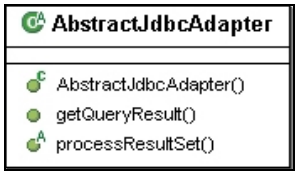


Decorator para el ValueListHandler. Provee la funcionalidad adicional de manejar tiempo de vida de los resultados cacheados.

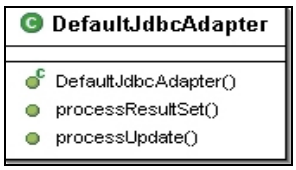
1.3.1.3. Adapters

Los componentes principales de este subsistema son los adaptadores a las fuentes de datos utilizando distintos tipos de conexiones.

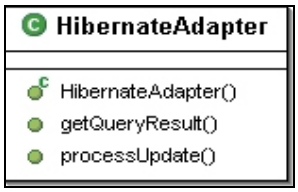




Adapter que permite realizar consultas de selección utilizando una conexión JDBC. Es una clase abstracta dado que el procesamiento del ResultSet obtenido es realizado por las clases que la extienden.



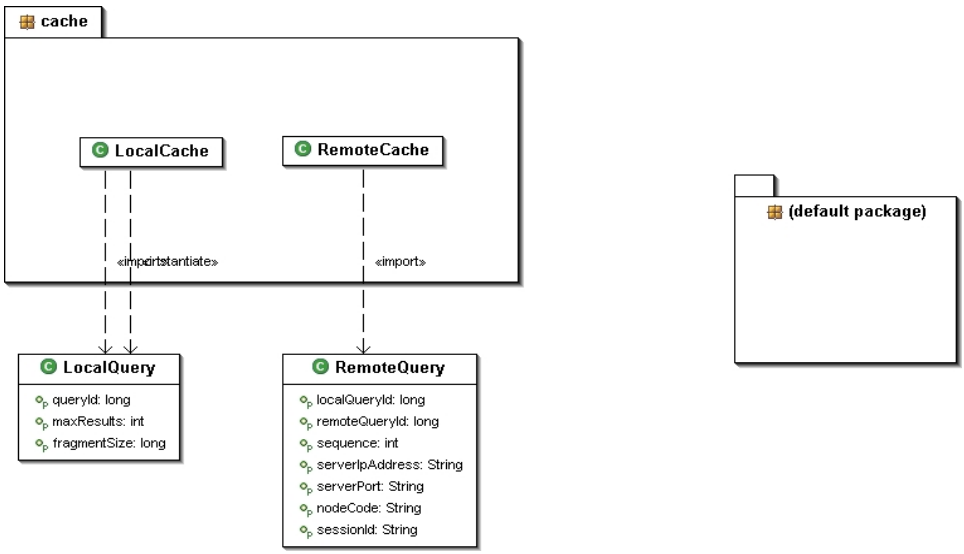
Adapter que permite ejecutar consultas de selección o actualización utilizando una conexión JDBC.

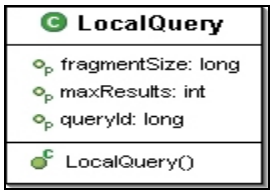


Adapter que permite realizar consultas de selección o actualización utilizando el framework Hibernate.

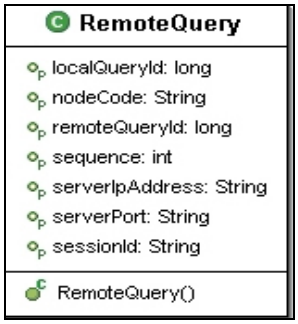
1.3.1.4. Caché

Este subsistema provee el servicio de caching de objetos al Federador de Datos. Permite almacenar en memoria información sobre las consultas que recibe el Federador así como también las que el realiza.

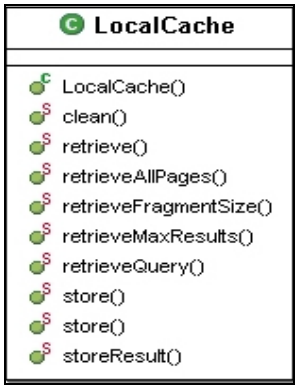




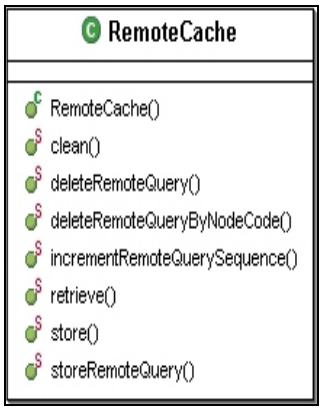
Consulta local. Mantiene información sobre las consultas que "recibe" el Federador de Datos.



Consulta remota. Mantiene información sobre las consultas que "realiza" el Federador de Datos a federadores remotos.



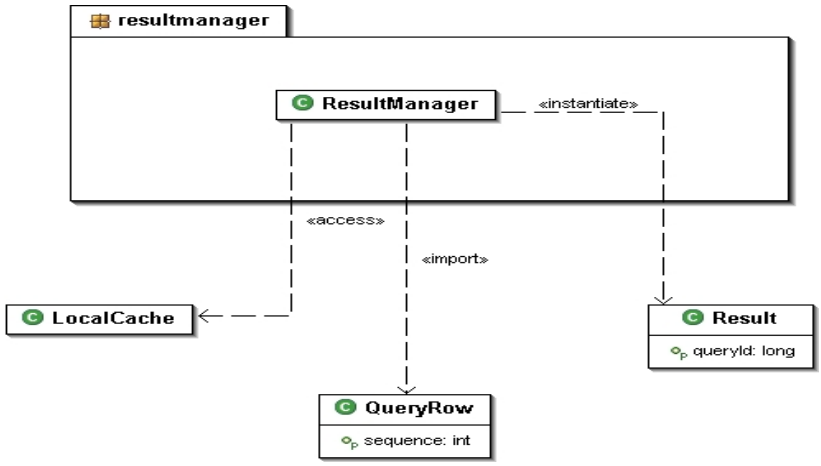
Caché local. Almacena en memoria consultas locales, es decir consultas que el Federador de Datos recibe.

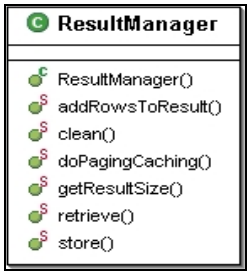


Caché remoto. Almacena en memoria consultas remotas, es decir consultas que el Federador de Datos realiza.

1.3.1.5. ResultManager

Este subsistema provee los servicios de administración de resultados al Federador de Datos. Maneja los resultados de las consultas siendo el encargado de paginarlas y almacenarlas en caché.

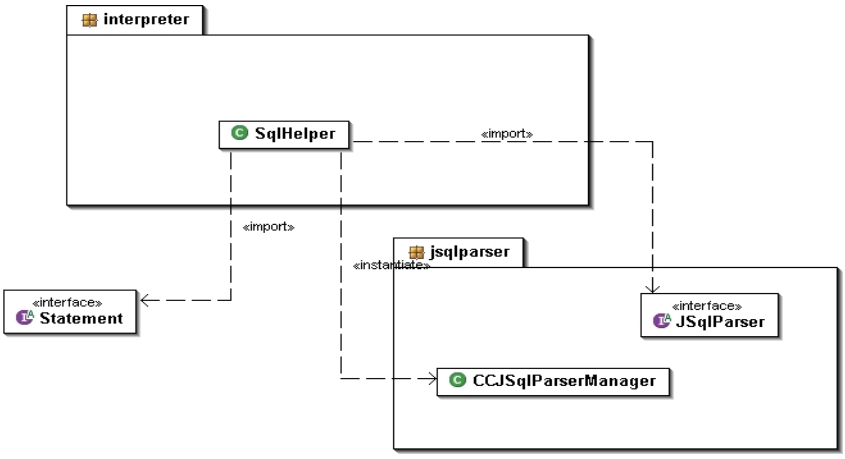


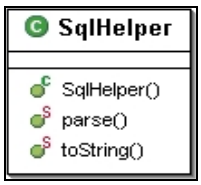


Esta clase es quien maneja los resultados (Result).

1.3.1.6. Interpreter

Este subsistema provee los servicios de parsing al Federador de Datos.

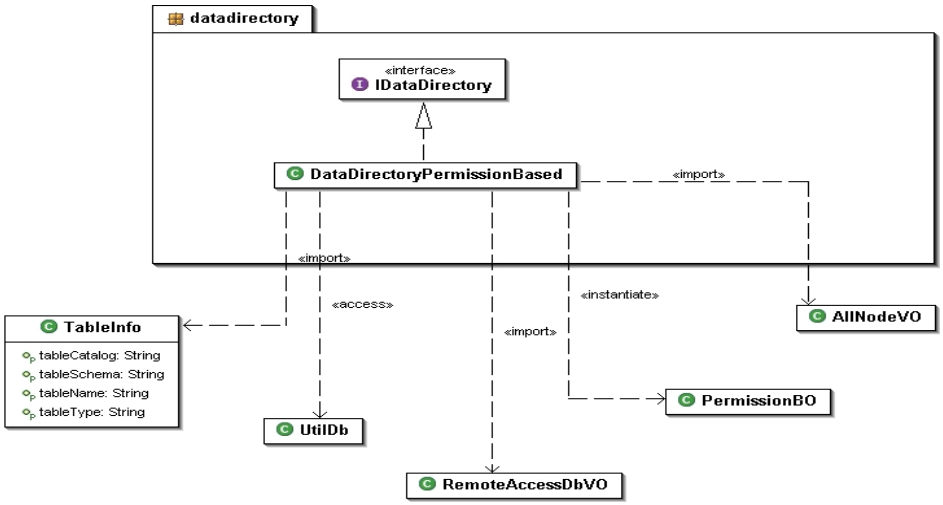




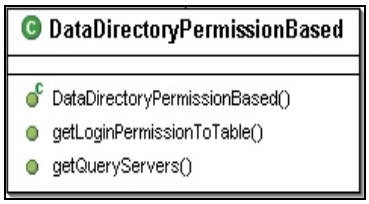
Esta clase permite convertir sentencias SQL en objetos y viceversa.

1.3.1.7. DataDirectory

Este subsistema es básicamente el directorio del Federador Datos. Su función principal es indicar a que nodos remotos consultar en base a la consulta realizada.



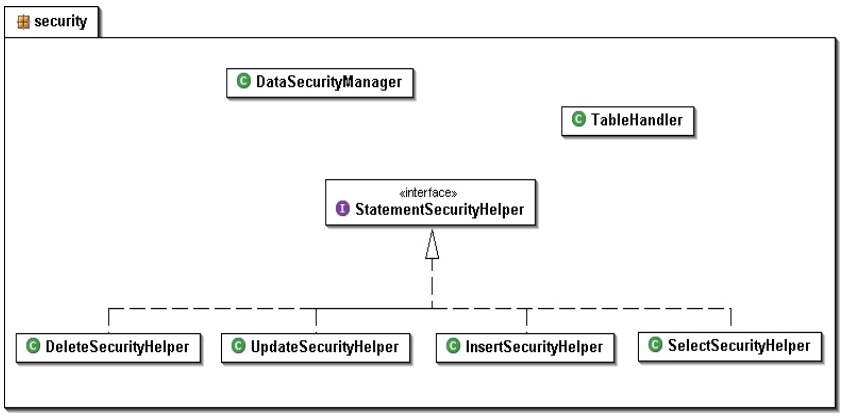
Interfaz que deben cumplir las posibles implementaciones para el directorio de datos.



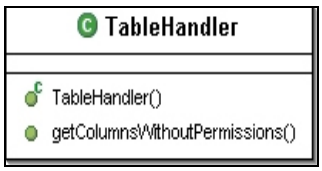
Directorio de Datos basado en permisos. Es una implementación posible al directorio donde los nodos a consultar se obtienen en base a los permisos que el grupo tiene sobre las tablas.

1.3.1.8. Security

Subsistema encargado de hacer cumplir las restricciones de seguridad a nivel de software para el Federador de Datos.

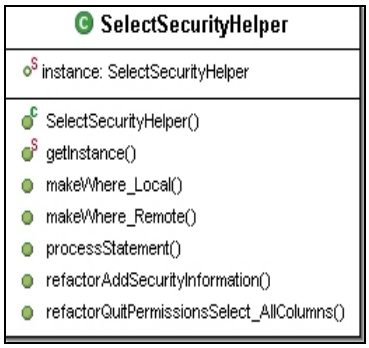


Esta clase procesa las consultas que llegan al Federador de Datos anexando la información de seguridad en caso de ser necesario.



Manejador de Tablas. Provee funciones de ayuda sobre las tablas.

Interfaz que deben cumplir las clases de ayuda del componente de seguridad para el procesamiento de sentencias.



Clase de ayuda del componente de seguridad que permite procesar sentencias Select.

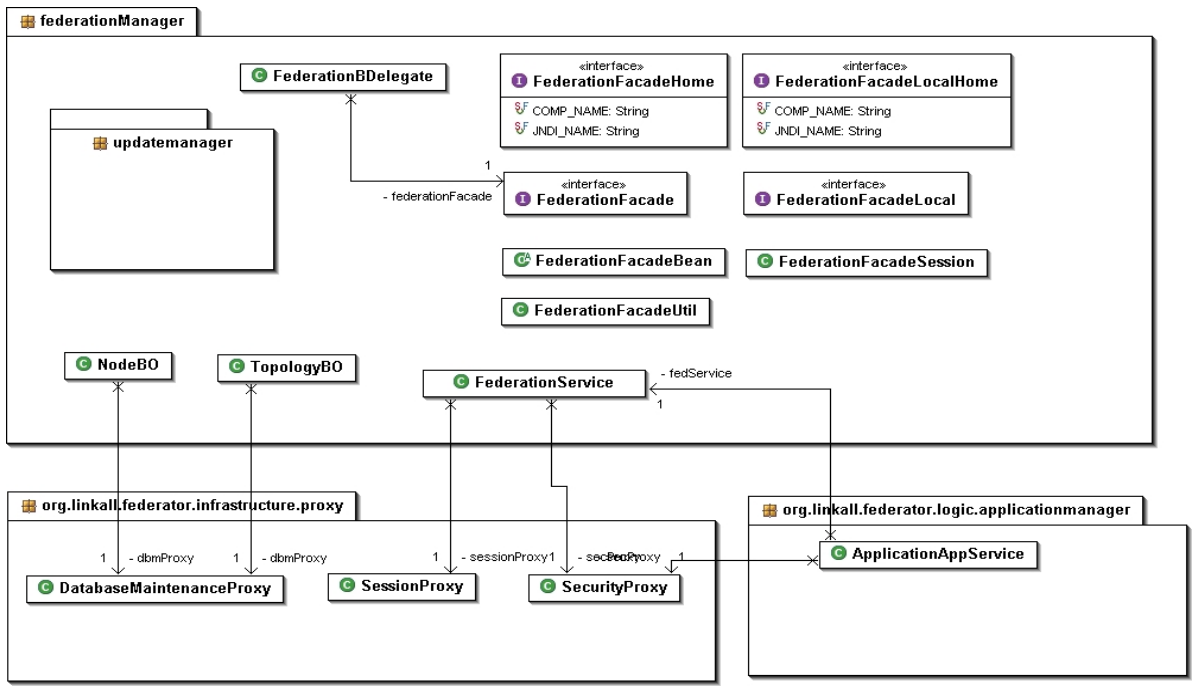
1.4. Subsistemas comunes

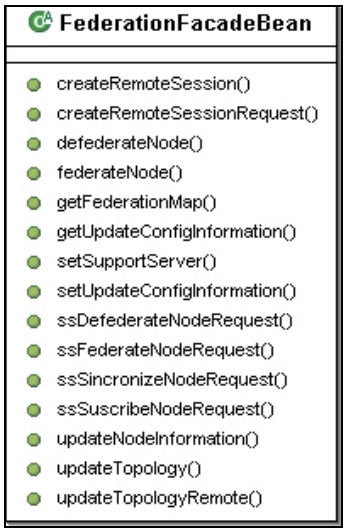
En las secciones siguientes se presenta una descripción de los subsistemas y las principales clases comunes al Federador de Aplicaciones y al Federador de Datos.

1.4.1. FederationManager

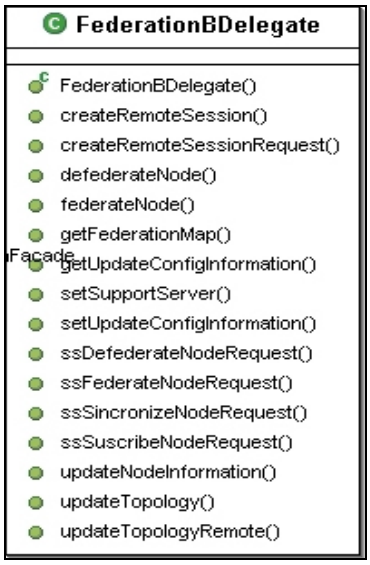
Este subsistema brinda servicios de federación a los subsistemas ApplicationManager y QueryManager. Es el encargado también de mantener la información de la federación que posee el nodo en sincronía con el resto de la red Link-ALL. Entre sus principales funciones se encuentran la de contener e informar cambios ocurridos en el nodo al resto de la Federación (a través del Servidor de Soporte) y también es el encargado de recibir y procesar paquetes de actualización de la federación que llegan al nodo utilizando para esto al subsistema UpdateManager. Otras funcionalidades que provee son las que hacen posible el alta y baja de un servidor Link-ALL en la federación.

Para llevar a cabo tales operaciones cuenta entre sus componentes con un Session Bean sin estado y objetos de negocio de bajo nivel que brindan operaciones básicas sobre los nodos (NodeBO) y la topología (TopologyBO).

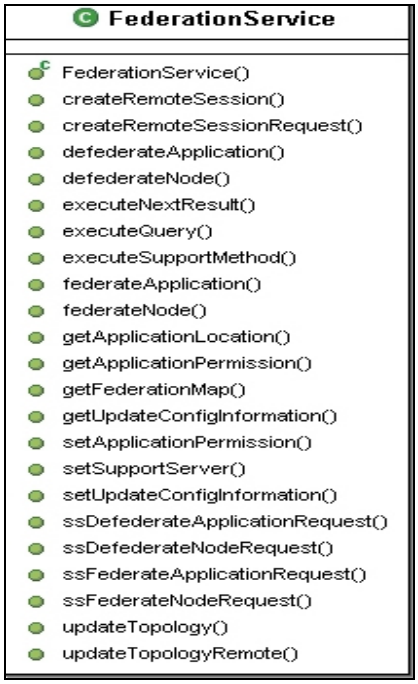




Session Bean de manejo de la Federación.
Es un Session Bean sin estado que actúa como fachada. Su principal objetivo es brindar flexibilidad y claridad en el diseño ya que desacopla el cliente de los objetos de negocio de bajo nivel (por ejemplo TopologyBO) , conteniendo y centralizando el acceso a los mismos.

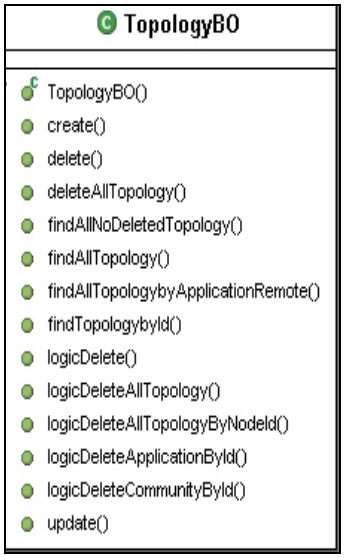


El objetivo principal del FederationDelegate es el de ocultar la complejidad de la comunicación remota con los servicios de negocio que manejan la federación.
Realiza el acceso a la fachada de los servicios de federación utilizando para ello el componente Service Locator y encapsula los errores que se produzcan al utilizar dicho componente.



Esta clase implementa la lógica del subsistema FederationManager que involucra a varios objetos de negocio.

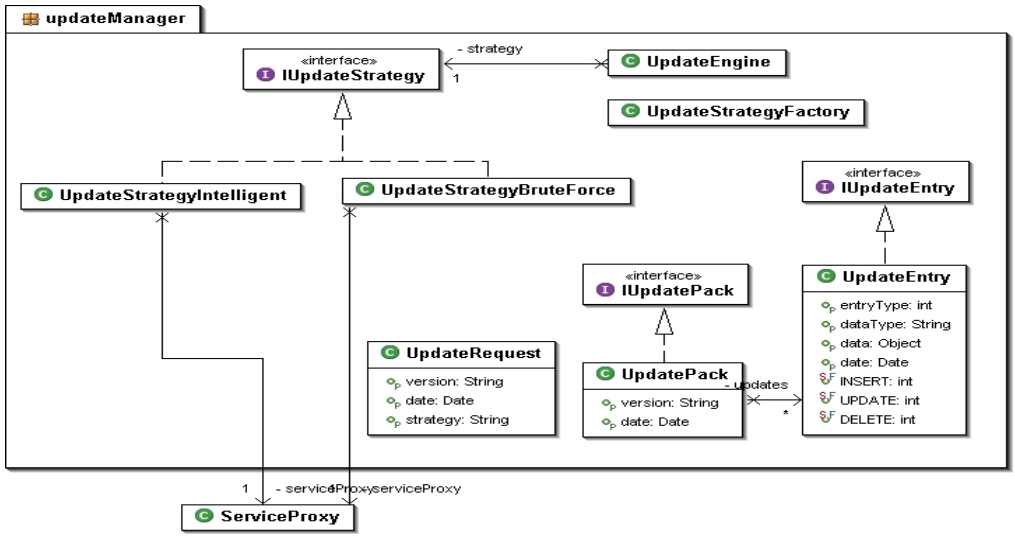
NodeBO es un objeto de negocio de bajo nivel encargado de encapsular y administrar los datos referidos a los nodos de la Federación Link-ALL conjuntamente con su comportamiento y persistencia.



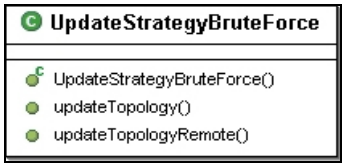
TopologyBO es un objeto de negocio de bajo nivel encargado de encapsular y administrar los datos referidos a entradas de la topología (información de la federación) conjuntamente con su comportamiento y persistencia.

1.4.1.1. UpdateManager

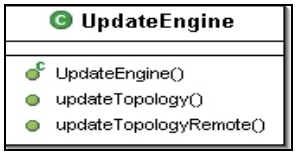
Este subsistema miembro de FederationManager es el encargado de la administración de las actualizaciones de la información de la federación (actualizaciones de la topología).



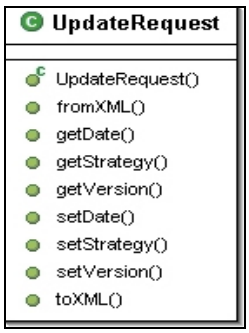
Interfaz que deben cumplir las estrategias de actualización de la información de la federación.
Básicamente especifica que las estrategias deben proveer dos métodos correspondientes entre sí, uno de invocación local (updateTopololgy) y otro de invocación remota (updateTopologyRemote).



Estrategia de actualización Fuerza Bruta. Esta estrategia implementa la interfaz IUpdateStrategy. Como su nombre lo indica sigue una metodología fuerza bruta para actualizar la información de la federación. Esta consiste básicamente en descargar TODA la información de la federación y reemplazarla por la local.

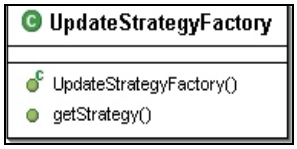


Motor de las actualizaciones. Este objeto permite realizar la actualización de la información de la federación, contra el Servidor de Soporte, utilizando una estrategia configurable por el usuario.

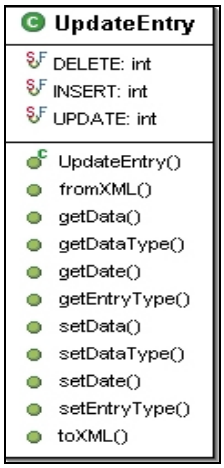


Pedido de actualización. Este objeto encapsula información sobre el pedido de actualización de la información de la Federación que realiza un nodo al Servidor de Soporte. Actualmente se utiliza de manera simple, pero el nodo podría especificar en él la versión de actualización que desea (o información de cambios de la federación posteriores a la última actualización, indicada en el campo Date) así como también la estrategia a utilizar para la transferencia.

Paquete de actualización. Básicamente consiste en un conjunto de entradas (UpdateEntry) de actualización. Permite especificar fecha, tamaño y versión del paquete en cuestión.
Además incluye métodos para su fácil serialización y deserialización a XML.



Factory de estrategias de actualización. Se incluye básicamente para facilitar la inclusión de nuevas estrategias de actualización.

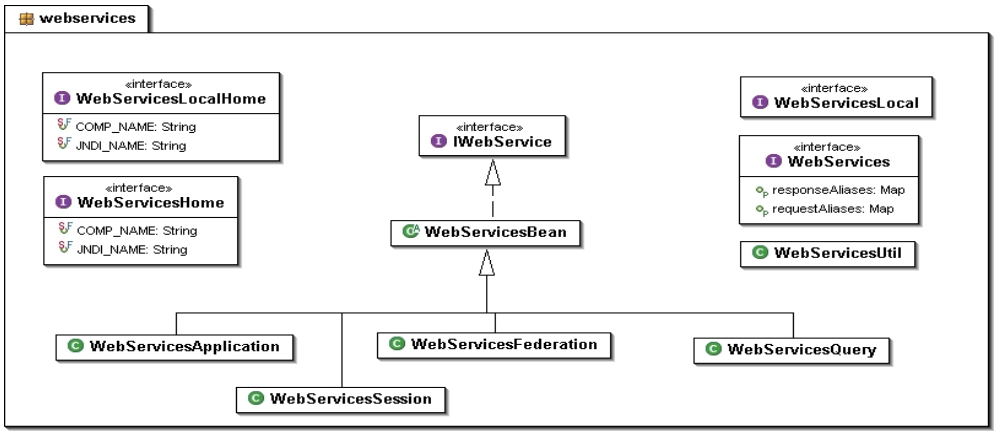


Entrada del paquete de actualización. Consiste en un elemento de información del paquete de actualización. Permite especificar el tipo de la entrada, la fecha, el tipo de dato y el dato (información de actualización).

1.4.2. WebServices

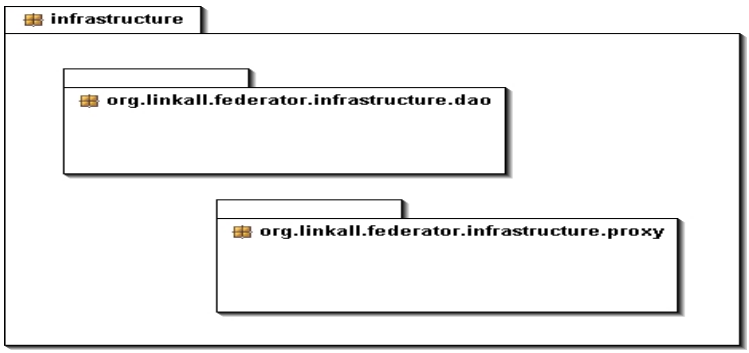
Este subsistema expone las funcionalidades del Federador Link-ALL a través de Web Services.

Los servicios Web expuestos son suministrados por tres componentes principales de éste subsistema: WebServicesApplication, WebServicesQuery y WebServicesFederation. Cada uno de éstos expone servicios proporcionados por la lógica del negocio: ApplicationManager, QueryManager y FederationManager respectivamente.



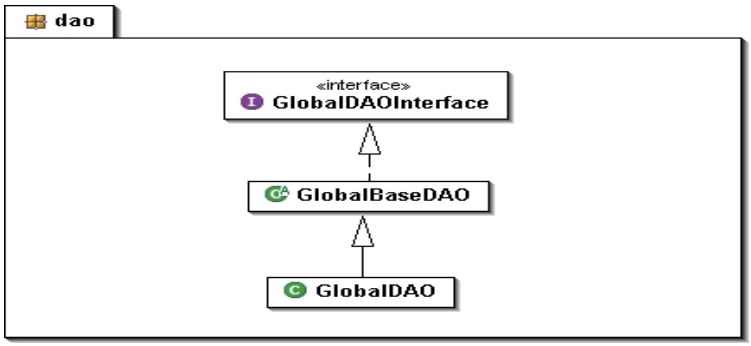
1.4.3. Infrastructure

El principal objetivo de éste subsistema es el de abstraer al Federador Link-ALL del acceso a componentes externos y a la fuente de datos. Para ello cuenta con dos componentes esenciales que se detallarán a continuación.



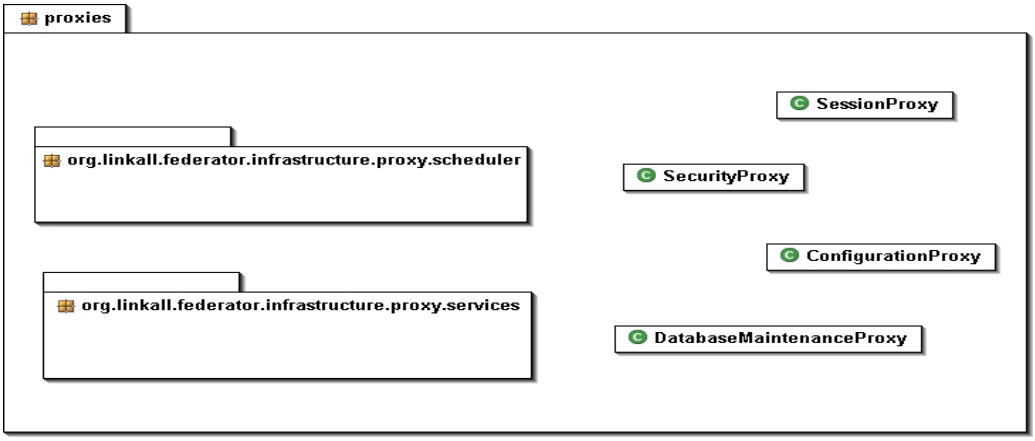
1.4.3.1. DAO

Este subsistema está compuesto por el objeto que provee el acceso a la Base de Datos Global de Link-ALL: GlobalDAO. Su objetivo principal es abstraer al Federador de Datos del acceso directo a la fuente de datos.



1.4.3.2. Proxies

Este subsistema comprende principalmente los proxies remotos de acceso a componentes externos al Federador Link-ALL. Entre sus componentes se encuentran el proxy de acceso al servicio de seguridad (SecurityProxy), al servicio de manejo de sesiones (SessionProxy), a la base administrativa (DatabaseMaintenanceProxy), al planificador de trabajos y por último a los servicios prestados por otros federadores (ServiceProxy).



Esta clase actúa como Proxy remoto del componente que administra la seguridad del Servidor Link-ALL.
Los componentes del Federador evitan así el uso directo del servicio, disminuyendo el acoplamiento entre ambos. El Proxy además de enmascarar la comunicación también realiza manejo de errores (en caso de que ocurra un error tanto en el Proxy como en el servicio de seguridad la única excepción que tendrá el Federador es SecurityProxy-Exception).

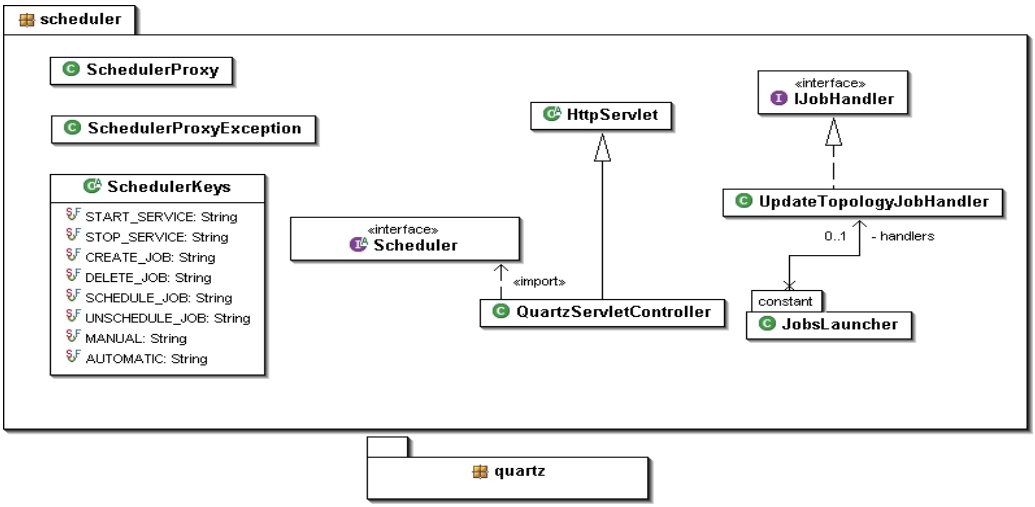
Esta clase actúa como Proxy remoto del componente que administra las sesiones del Servidor Link-ALL.
Los componentes del Federador evitan así el uso directo del servicio, disminuyendo el acoplamiento entre ambos. El Proxy además de enmascarar la comunicación también realiza manejo de errores (en caso de que ocurra un error tanto en el Proxy como en el servicio de manejo de sesiones la única excepción que tendrá el Federador es SessionProxyException).

Esta clase actúa como Proxy remoto del componente que administra el acceso a la Base de Datos Administrativa (NAD) del Servidor Link-ALL.
Los componentes del Federador evitan así el uso directo del servicio, disminuyendo el acoplamiento entre ambos. El Proxy además de enmascarar la comunicación también realiza manejo de errores (en caso de que ocurra un error tanto en el Proxy como en el acceso al NAD la única excepción que tendrá el Federador es DatabaseMaintenanceProxy-Exception).

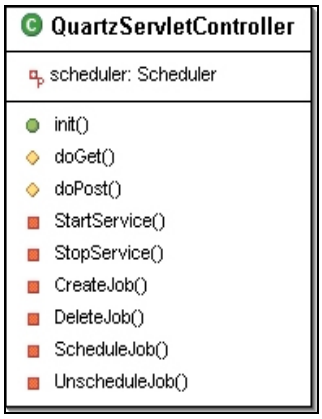
Esta clase actúa como Proxy remoto del componente que administra la configuración del Servidor Link-ALL.
Los componentes del Federador evitan así el uso directo del servicio, disminuyendo el acoplamiento entre ambos. El Proxy además de enmascarar la comunicación también realiza manejo de errores (en caso de que ocurra un error tanto en el Proxy como en el servicio de administración de configuración la única excepción que tendrá el Federador es ConfigurationProxyException).

1.4.3.2.1. Scheduler

Este subsistema brinda los servicios de planificación de trabajos (Jobs) al Federador Link-ALL. Para cumplir con sus funciones cuenta entre sus componentes principales con Quartz [4], un scheduler open source, el cual es invocado a través de un Proxy remoto, que pretende desacoplar al Federador del uso particular de dicho framework. Entre las posibilidades que brinda es el de ejecutar cualquier tipo de trabajos, iniciar y detener el servicio de planificación según se necesite y por sobre todo la capacidad de recordar trabajos planificados después de una caída o apagado del servidor de aplicaciones. Otra característica a mencionar es que la declaración de los disparadores de las tareas (los que determinan en que momento se ejecutarán) se pueden especificar utilizando una expresión tipo UNIX, lo que le brinda flexibilidad a dicho mecanismo.



Detalle de las principales clases que componen el subsistema Scheduler:

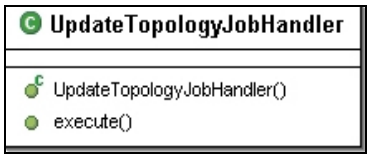


Esta clase maneja el planificador de trabajos (Quartz). Sus principales funciones son iniciar y detener el servicio del scheduler, crear, eliminar, programar y desprogramar trabajos (jobs).
Es un Servlet dado que está configurado para iniciar conjuntamente con el servidor de aplicaciones. Esto tiene como objetivo hacer persistentes los trabajos agendados a pesar de las caídas del servidor.

Proxy remoto del Scheduler. El objetivo de esta clase es independizar al Federador del framework elegido para que brinde los servicios de planificación de trabajos (Quartz). Todas las llamadas del Federador al planificador se realizan a través de este Proxy lo cual permitiría cambiar el componente que brinda el servicio o implementar uno propio.

Esta interfaz es la que deben cumplir todas las tareas que serán planificadas por el Scheduler.

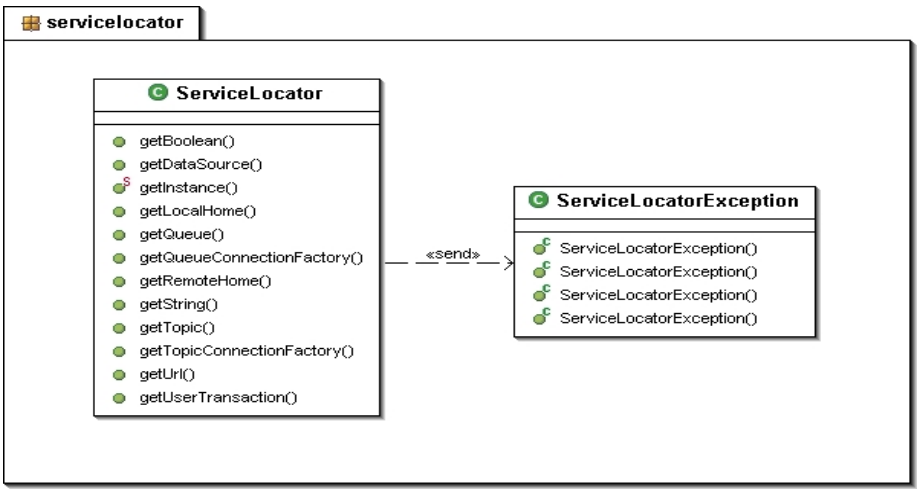
Lanzador de trabajos. Esta clase es la que es “despertada” por el planificador en el momento agendado. La acción y los parámetros a utilizar son establecidos por QuartzServletController. Una vez que la clase despierta realiza la acción que le fue ordenada (por ejemplo actualizar la información de la topología).

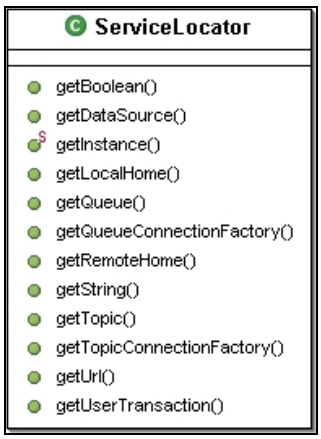


Esta clase implementa la interfaz IJobHandler. Actualiza la información de la topología una vez que JobsLauncher es despertado por el planificador.

1.4.3.3. Service Locator

Este subsistema permite implementar y encapsular la búsqueda a los servicios. El service locator oculta los detalles de implementación, reduce la complejidad, provee un solo punto de control y mejora la performance al proveer facilidad de caching de referencias.





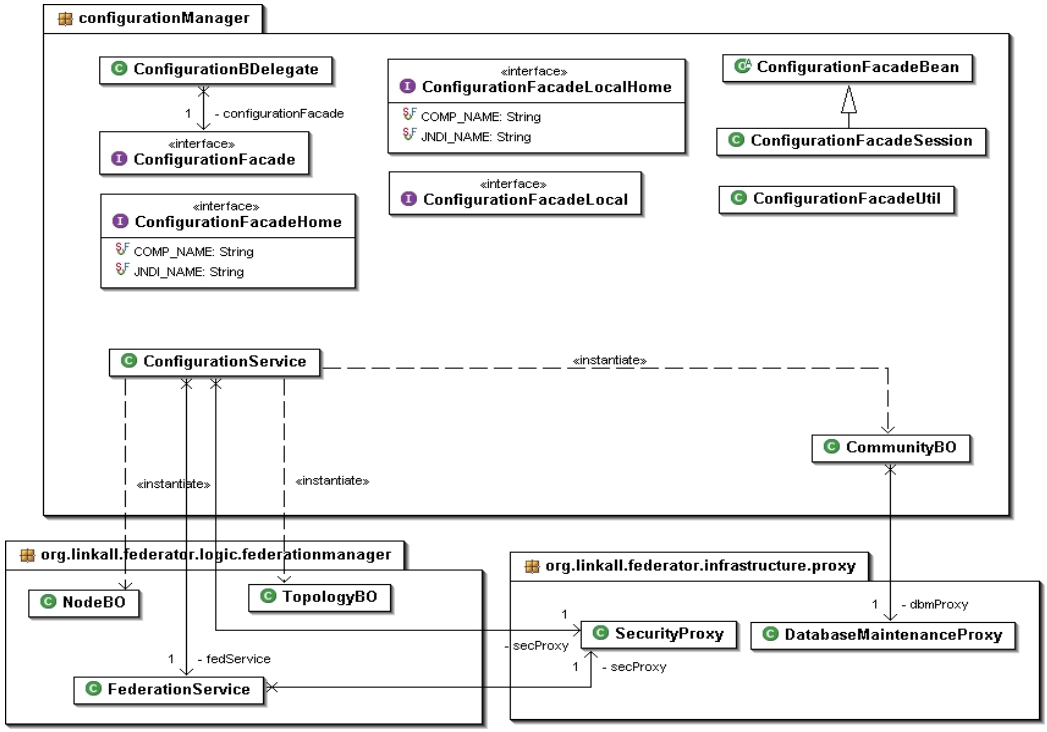
Esta clase brinda los servicios de búsqueda y caching al subsistema.

1.5. Subsistemas de Configuración

Se describe a continuación los subsistemas que brindan servicios de configuración al Federador Link-ALL.

1.5.1. ConfigurationManager

Una de las funcionalidades importantes que brinda este subsistema es la gestión de Comunidades existentes en el Servidor Link-ALL.

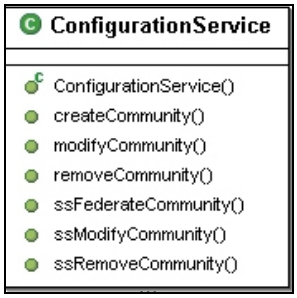


Detalle de las principales clases que componen el subsistema ConfigurationManager:

Session Bean de Configuración.

Es un Session Bean sin estado que actúa como fachada. Su principal objetivo es brindar flexibilidad y claridad en el diseño ya que desacopla el cliente de los objetos de negocio de bajo nivel (por ej. CommunityBO) , conteniendo y centralizando el acceso a los mismos.
En funcionalidades complejas, el cliente puede llegar a requerir múltiples llamadas a los componentes de negocio de bajo nivel, con la consecuente sobrecarga en la red y acoplamiento entre ambos. Al existir esta “fachada” el cliente sólo invoca un método de la misma y es ésta quien realiza las demás interacciones.

CommunityBO es un objeto de negocio de bajo nivel encargado de encapsular y administrar los datos referidos a comunidades conjuntamente con su comportamiento y persistencia.



Este clase implementa la lógica del subsistema que involucra a varios objetos de negocio.

2. FedBrowser

El navegador de la federación (FedBrowser) permite visualizar el estado actual de la federación. Se trata de una aplicación J2EE, integrada a la plataforma Link-ALL que permite al usuario tener una idea visual del conjunto de servidores, comunidades y aplicaciones que forman parte de la federación, brindando a su vez información de cada uno.

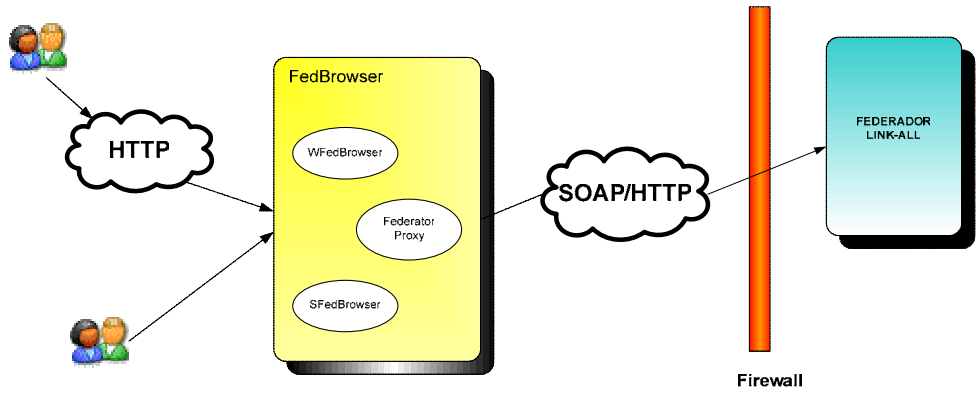


Fig. 2.1 – Vista de Instalación, FedBrowser.

2.1. Arquitectura de FedBrowser

El navegador de la federación presenta una arquitectura de tres capas: Presentación, Lógica y Recursos como se muestra a continuación:

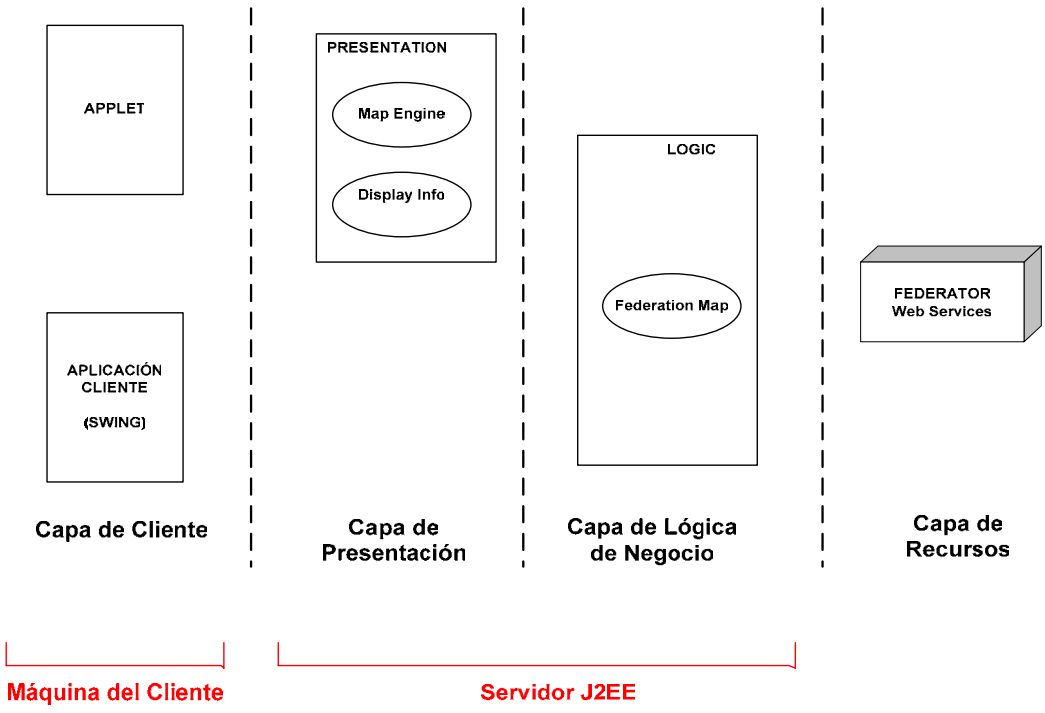


Fig. 2.2 - Arquitectura de FedBrowser.

En la capa del cliente se encuentran los componentes que se ejecutan en la máquina del cliente. Entre los mismos se encuentran un Applet y una aplicación de escritorio Swing que permiten visualizar la información de la federación.

La representación visual del estado de la federación se logra a través de un proceso de tres fases: obtención de la información, procesamiento y despliegue. A continuación se detalla dicho proceso brindando detalles técnicos de los pasos del mismo:

Obtención de la información de la federación.

Para obtener la información de la federación FedBrowser utiliza un Proxy remoto de acceso (FederatorProxy) a los servicios del Federador. Dicho Proxy es un cliente del Web Service ofrecido por el Federador Link-ALL que permite obtener el mapa de la federación actual en formato XML.

La configuración de la comunicación con el Federador Link-ALL se encuentra en un archivo de configuración denominado fedbrowserconfig.properties.

Procesamiento.

Una vez obtenido el XML del mapa de la federación el mismo es deserializado y almacenado en memoria en un objeto que lo representa denominado FederationMap. Este objeto posee métodos para cargar el mapa a partir de un paquete de actualización obtenido del Federador, así como también para realizar consultas sobre sus datos almacenados.

Despliegue.

El rol más importante en la etapa de despliegue lo tiene la herramienta TouchGraph [5] quien es la encargada de dibujar cada una de las entidades del mapa de la federación en la pantalla. Esta herramienta necesita que le provean la información en un determinado formato para poder desplegarla. El método formatToFedBrowser del objeto FederationMap es un método especial que permite “formatear” la información de la Federación contenida en el mapa a un formato XML que la aplicación entienda.

La figura siguiente presenta una vista lógica de la arquitectura de FedBrowser mostrando sus principales subsistemas:

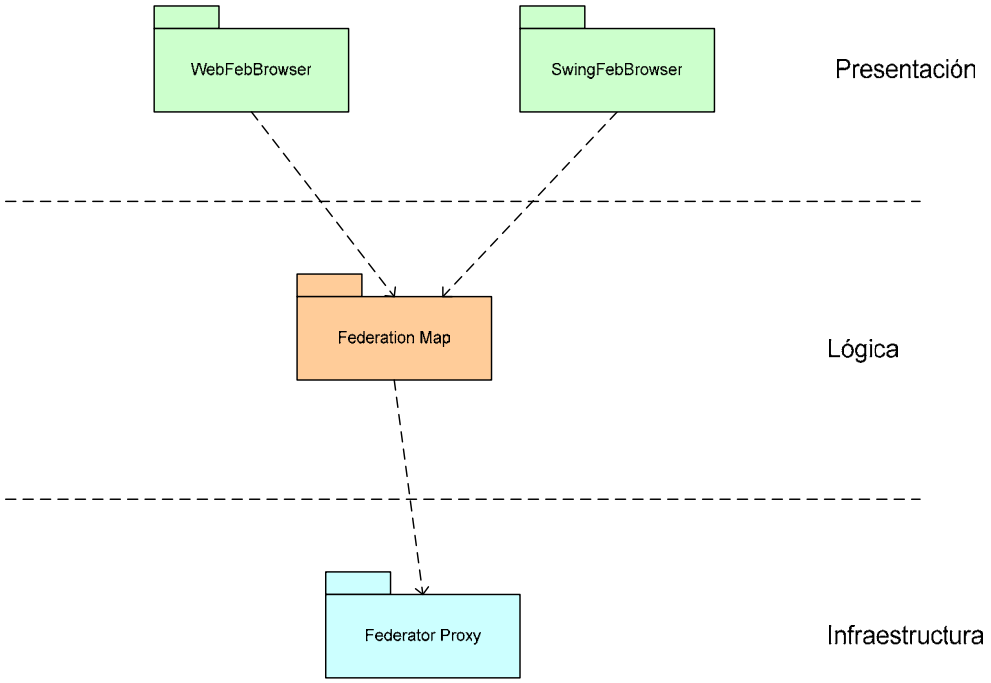


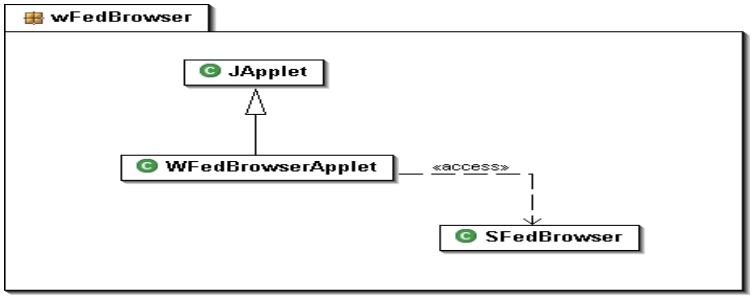
Fig. 2.3 - Vista Lógica de FedBrowser.

2.2. Subsistemas

En los párrafos siguientes se presenta una descripción de los subsistemas y las principales clases que componen FedBrowser.

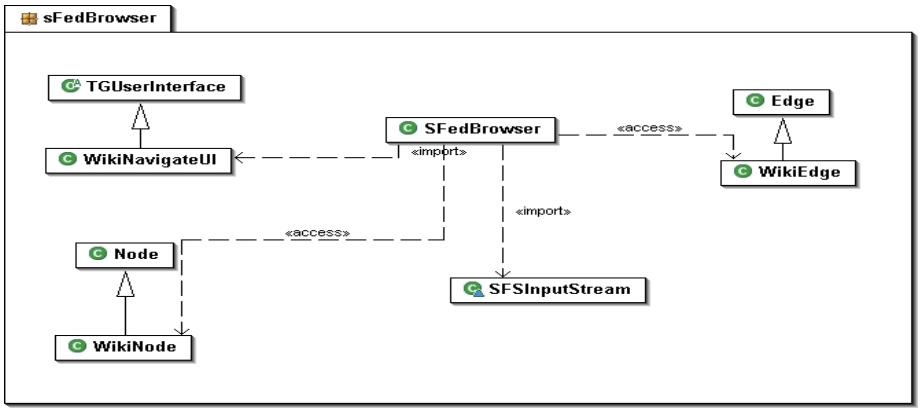
2.2.1. WFedBrowser

WebFedBrowser es el navegador Web de la federación. Este subsistema es el que hace posible que el estado actual de la federación pueda ser monitoreado remotamente utilizando una página Web.
Su componente principal es un Applet de Java (WFedBrowserApplet) que permite visualizar la información que despliega SFedBrowser (ver más adelante).



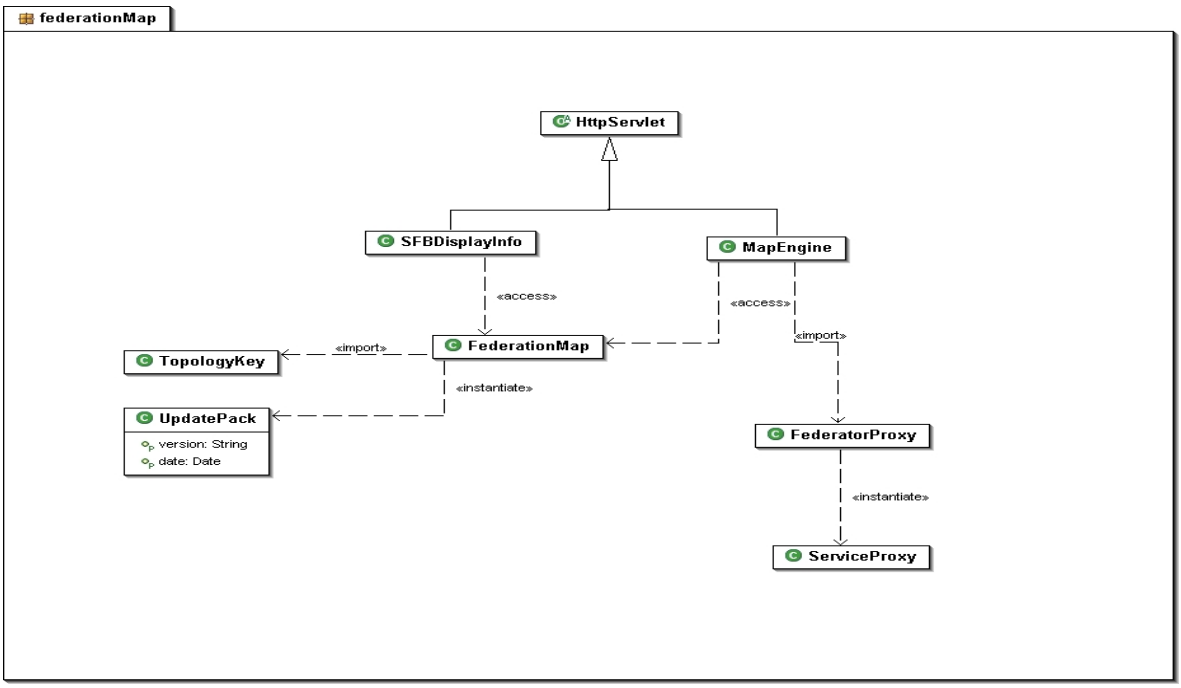
2.2.2. SFedBrowser

SFedBrowser es el navegador Swing de la federación. Este subsistema posibilita el monitoreo de la federación utilizando una interfaz de usuario Swing. Puede ser utlizada tanto de forma local como remota dado que la información que despliega la obtiene de un servlet (MapEngine) (ver más adelante).

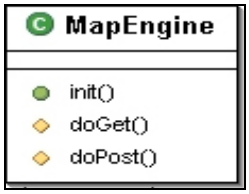


2.2.3. FederationMap

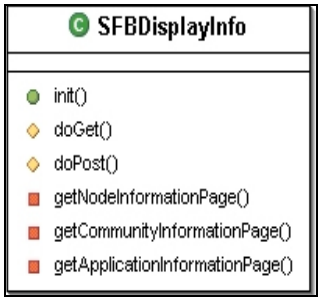
Este subsistema es fundamental en la lógica de negocio del navegador de la Federación. Los subsistemas encargados del despliegue (WFedBrowser y SFedBrowser) se valen de él para lograr sus objetivos. Es responsable de obtener la información actual de la federación cuando le sea solicitado, almacenarla en memoria, proveer operaciones de consulta sobre dicha información y además formatearla para que los subsistemas de despliegue la entiendan.



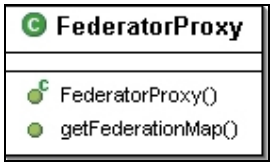
A continuación se presenta un detalle de las principales clases que componen el subsistema FederationMap:



Motor del mapa de la federación. Es un Servlet que tiene un rol importante en FedBrowser ya que es el encargado de obtener la información de la federación a través del Proxy (FederatorProxy). Luego debe cargar el mapa de la federación (FederationMap) y por último debe devolver en un formato adecuado la información para que sea desplegada en pantalla.



Servlet que es invocado para obtener información de un determinado objeto (nodo, aplicación, comunidad) que está siendo visualizado. Para obtener la información que se le solicita el componente utiliza la información de FederationMap (ver más adelante) y devuelve como resultado una página con los datos obtenidos.



Proxy remoto de acceso al Federador. Este objeto es utilizado para acceder al Federador, independizando de dicho acceso al FedBrowser.

Esta clase permite mantener el mapa de la federación almacenado en memoria. Posee métodos para cargar el mapa a partir de un paquete de actualización obtenido del Federador, así como también para realizar consultas sobre sus datos almacenados.
El método formatToFedBrowser es un método especial que permite “formatear” la información de la Federación contenida en el objeto a un formato XML que la aplicación que lo utiliza entienda.

3. Referencias

[1] Java™ 2 Platform Enterprise Edition Specification v1.4 http://java.sun.com/j2ee/, 06/04/2005

[2] Deepak Alur, John Crupi, Dan Malks, Core J2EE™ Patterns: Best Practices and Design Strategies Second Edition, Prentice Hall, 2003

[3] Hibernate site http://www.hibernate.org, 30/04/2005

[4] OPENSYMPHONY web site http://www.opensymphony.com/quartz, 22/01/2005

[5] TouchGraph WikiBrowser v1.02 http://www.touchgraph.com, 05/03/2005

PROYECTO DE GRADO
FEDERADOR LINK-ALL

Diseño de Casos de Uso

Mayo 2005

Instituto de Computación - Facultad de Ingeniería
Universidad de la República

Estudiantes: Fabricio Alvarez
Rodolfo Amador

Tutores: Federico Piedrabuena
Raúl Ruggia

Tabla de Contenido

1. Introducción

Este documento trata de describir como fueron diseñadas e implementadas las principales funcionalidades del Federador. Para el Federador de Aplicaciones se describen, rutear pedido de ejecución de una aplicación y federar aplicación. En el Federador de Datos se describe procesar consulta. También se detallan federar servidor y actualizar mapa de la federación que son funcionalidades de configuración de la federación.

No se incluyeron todas las funcionalidades porque muchas de ellas son similares a alguna de las que se describen en este documento.

2. Funcionalidades

Las funcionalidades del Federador fueron desarrolladas tomando en cuenta el diseño presentado en la Documentación Técnica, donde se describe la arquitectura del Federador [1]. En general las aplicaciones tienen el siguiente comportamiento: la interfaz del usuario (UI) invoca al método del Business Delegate. El Business Delegate invoca utilizando la interfaz remota al SessionBean que actúa de fachada, es decir, brinda transparencia al cliente sobre los servicios y objetos de negocio de bajo nivel, contiene y centraliza el acceso a los mismos, las fachadas fueron implementadas como SessionBean sin estado [2].

Los servicios implementan la lógica del negocio pudiendo invocar a otros servicios u objetos de negocio de bajo nivel. Los Buissines Objects son objetos reutilizables que buscan separar la lógica de persistencia de la lógica del negocio. Finalmente la persistencia y las invocaciones a otros servicios de la plataforma Link-ALL se realizan por medio de proxies, estos también enmascaran la comunicación con los demás federadores.

2.1. Rutear Aplicación

Descripción:

Un usuario invoca la ejecución de una aplicación, el Federador rutea el pedido independientemente que la aplicación esté instalada en el servidor donde se hizo el pedido o no y devuelve la URL de la misma. Las aplicaciones pueden ser públicas o privadas, para que un usuario pueda acceder a una aplicación privada, éste deberá tener permisos para su ejecución.

Una vez que llega el pedido de ejecución una aplicación al Federador Link-ALL, éste busca la aplicación entre las aplicaciones instaladas, si la encuentra entonces procede a verificar los permisos de ejecución que posee el usuario. Si la aplicación es pública o el usuario tiene permisos, el Federador devuelve la dirección de la misma.

En el caso de que el usuario no tenga permisos sobre la aplicación o ésta no se encuentre instalada en el nodo donde se realizó el pedido (aplicación remota), se buscan el o los servidores en los que el usuario puede acceder a la aplicación de acuerdo a los permisos remotos. Una vez localizado un servidor donde está instalada la aplicación, el Federador le solicita la creación de una sesión remota para el usuario, y retorna la dirección de la aplicación incluyendo la URL del servidor.

Diseño e implementación:

El usuario selecciona la aplicación que desea ejecutar de la lista de aplicaciones disponibles. Como cualquier evento en la interfaz de usuario provoca una llamada al Controller y éste selecciona e invoca al manejador que atenderá el pedido, que en el caso de rutear aplicación es el LocateHandler. El handler se encarga de obtener los parámetros ingresados por el usuario en la página JSP a través del request. Para ésta funcionalidad los parámetros obtenidos son el código de la aplicación y el identificador de la sesión del usuario. A partir del código de la aplicación se crea un Value Object de aplicación, y a partir del identificador de la sesión se crea un Value Object de contexto, con estos

parámetros el handler invoca al método getApplicationLocation del ApplicationBDelegate. Esto se muestra en el diagrama de secuencia de la figura 1.1.

El BusinessDelegate utiliza el ServiceLocator para obtener una referencia a la Fachada de los servicios de negocio referidos a aplicaciones (ApplicationFacade). Una vez obtenida la referencia se realiza una invocación vía RMI a dicha fachada, que es un Session Bean sin estado. A partir de aquí el Session Bean se encarga de llevar a cabo la solicitud, invocando a los servicios correspondientes; que en este caso es el método getApplicationLocation del servicio ApplicationAppService.

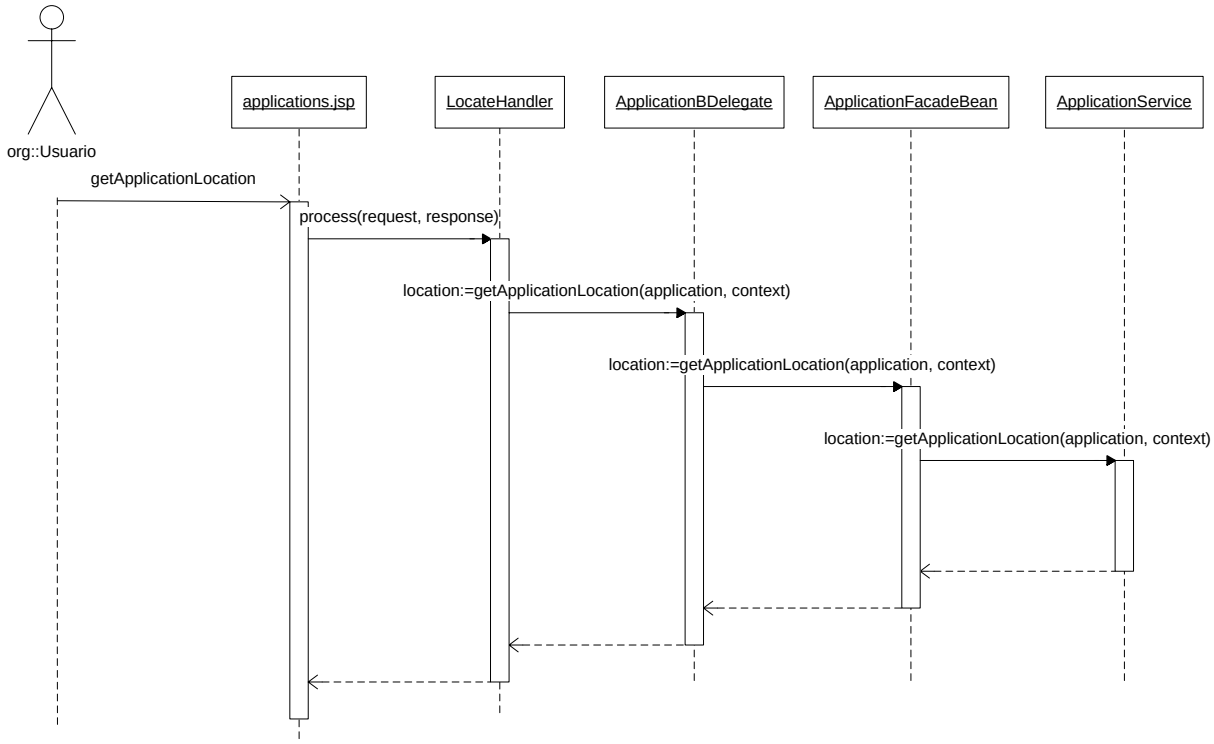


Fig. 1. 1 - Rutear aplicación, hasta el ApplicationService.

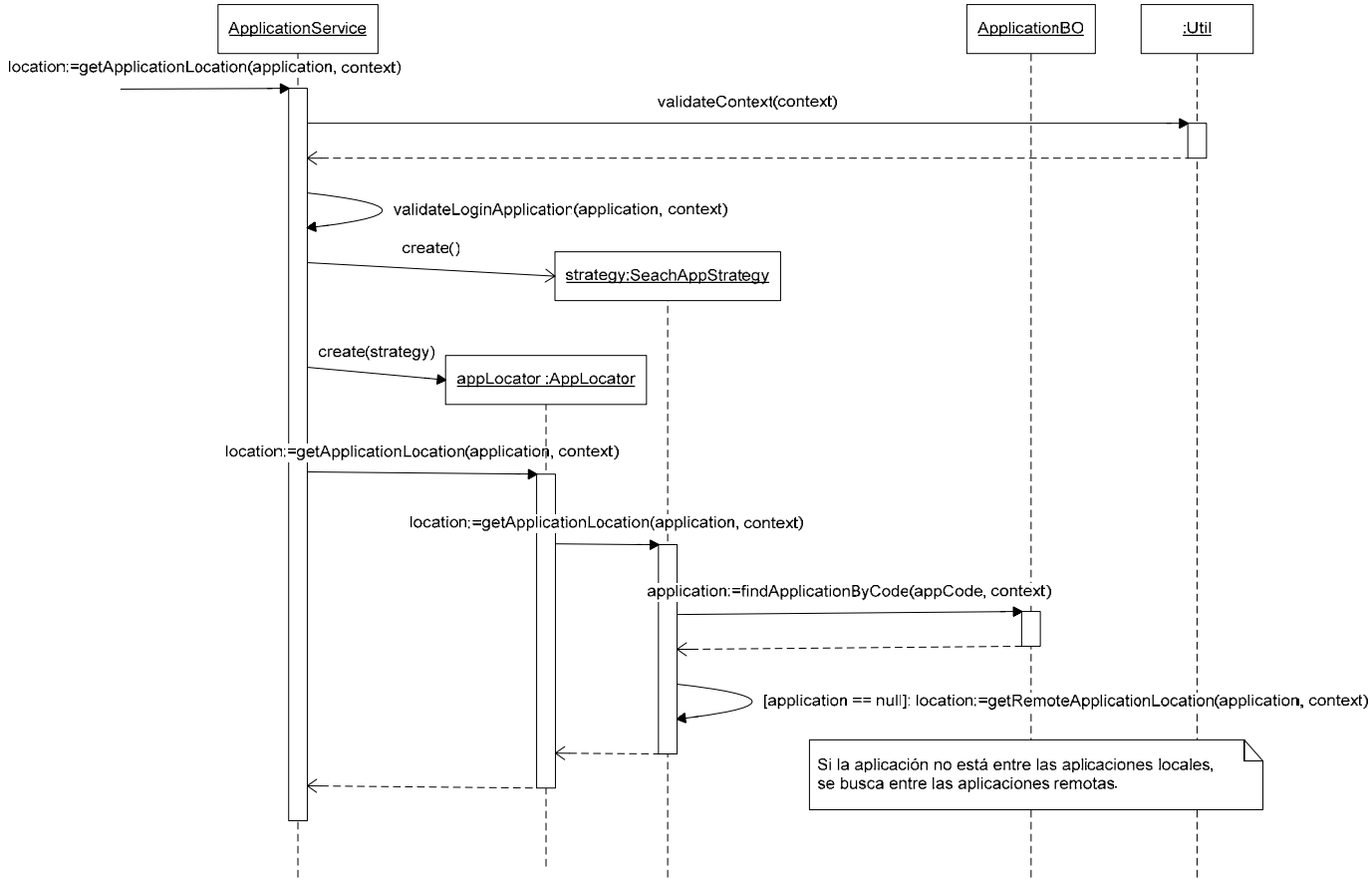


Fig. 1.2 - Lógica del negocio de Rutear Aplicación

La figura 1.2 muestra el diseño de la lógica del negocio de rutear aplicación. Primeramente se valida el contexto de la llamada y después se chequean los permisos del usuario sobre la aplicación. Aunque se omitió en la figura por razones de claridad, estas validaciones se hacen utilizando los servicios de la plataforma Link-ALL por intermedio de los proxies correspondientes.

La interfaz ISearchAppStrategy es implementada por la clase SeachAppStrategy y se encarga de localizar la aplicación requerida. Aquí se ve la utilización del patrón Strategy [2], existiendo la posibilidad de incorporar fácilmente nuevas estrategias para la localización de la aplicación.

La estrategia implementada consiste básicamente en buscar las aplicaciones entre las locales; si la encuentra retorna la URL de la aplicación. En caso de no encontrarla, busca entre las aplicaciones remotas que estén federadas y que el usuario tenga permisos.

Para localizar se toma en cuenta los permisos que el usuario posee para acceder a aplicaciones remotas. En los mismos se indica la comunidad y el nodo que otorgó el acceso. Si el nodo y la aplicación estén federados se solicita al servidor donde está instalada la aplicación que cree una sesión remota para el usuario.

Por su parte la figura 1.3 muestra la localización de la aplicación remota hasta la llamada al Web service. Éste invoca el método getApplicationLocationRemote, si el resultado es correcto, devuelve la dirección de la aplicación que incluirá la URL del nodo y retorna también el identificador de la sesión remota.

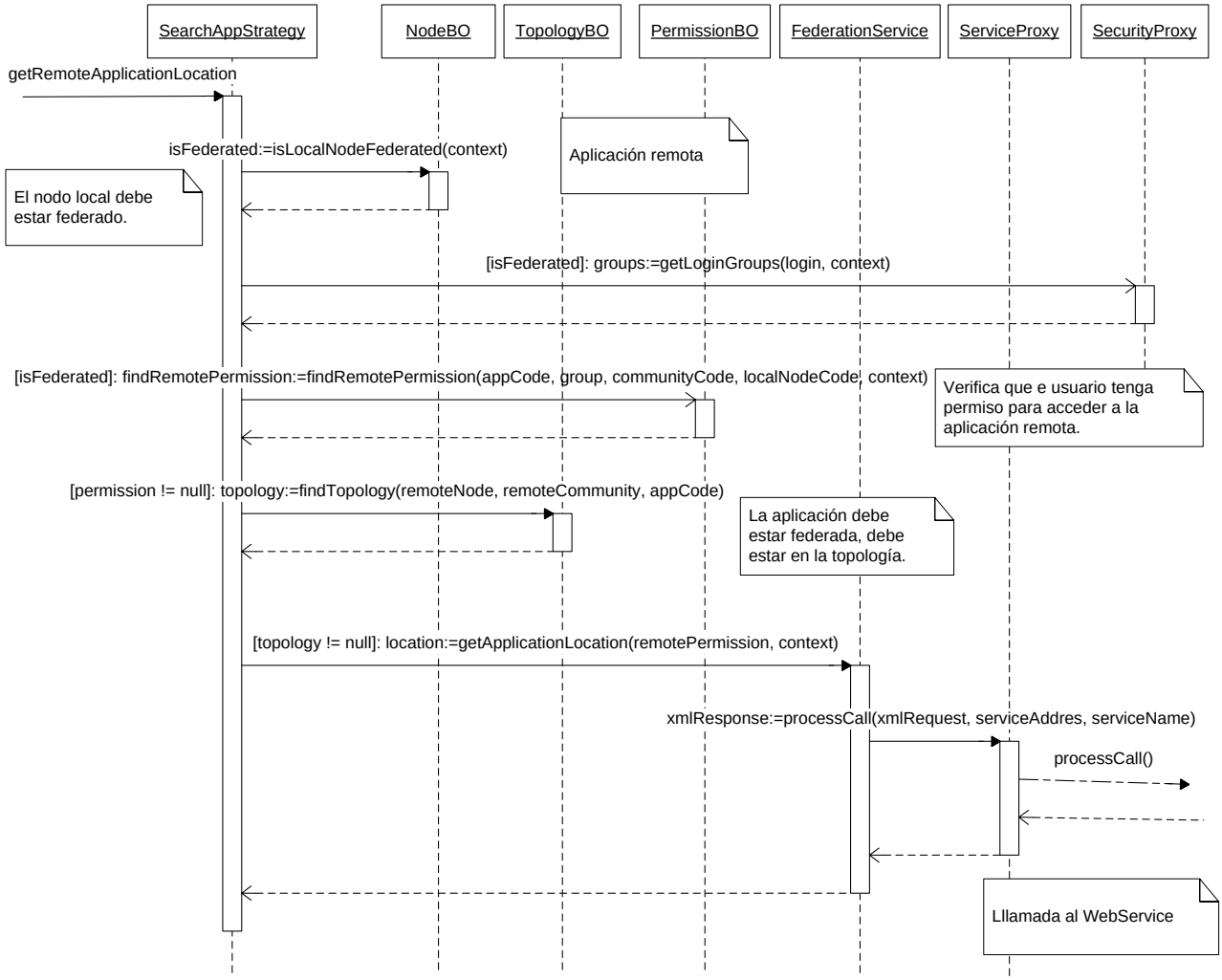


Fig. 1.3 Rutear aplicación remota

En el servidor donde está instalada la aplicación la llamada es atendida por el Web Service. El mismo se encarga de invocar al BusinessDelegate para rutear aplicación remota. Cuando la invocación llega al ApplicationAppService, primero se verifican los permisos del grupo del nodo remoto, que hace la solicitud de rutear la aplicación. Al igual que con las aplicaciones locales se crea una estrategia para obtener la URL de la aplicación. Al recibir un contexto remoto, la estrategia solo busca la aplicación entre las aplicaciones locales y crea una sesión remota para devolver junto con la URL, a la que incluye la dirección del nodo local. Esto se puede apreciar en la figura 1.5.

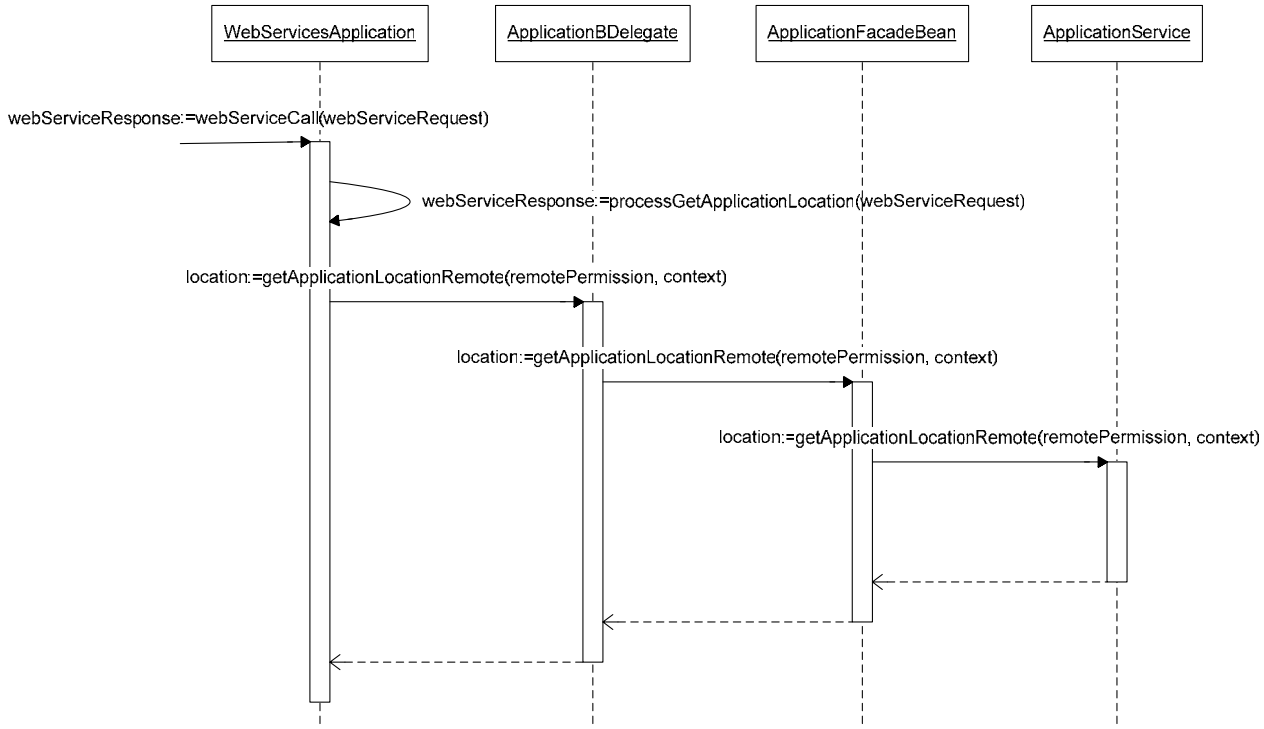


Fig. 1.4 - Rutear apliación remota llamada del webservice hasta el applicationService

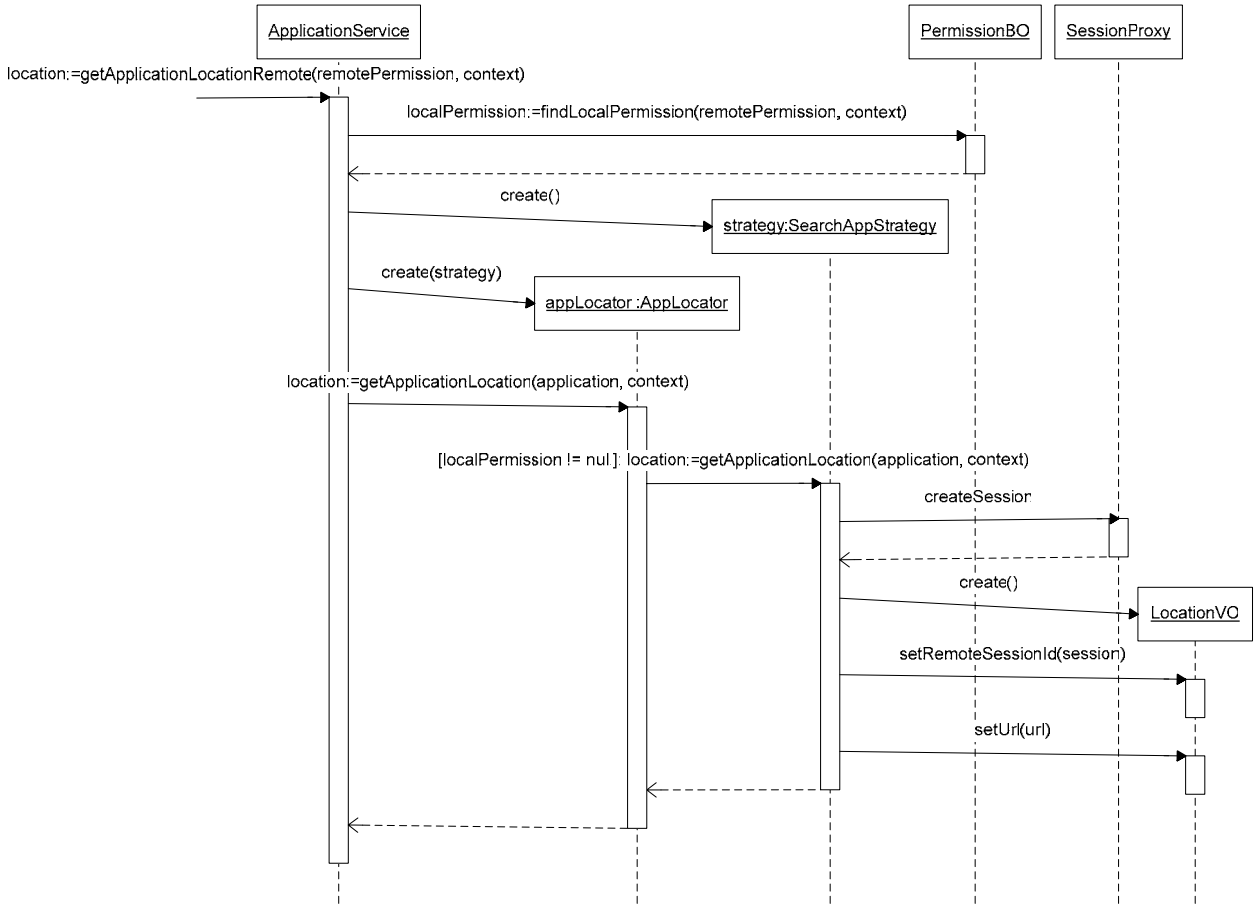


Fig. 1.5 - Diagrama de secuencia rutear aplicación remota en el nodo remoto

2.2. Federar Aplicación

Descripción:

Consiste en exportar o publicar una aplicación en la federación. Un usuario administrador de la configuración del Federador puede indicar si la federación de la aplicación es inmediata a la instalación de la misma por el componente Link-ALL encargado de tales funciones (Deployer). Si la federación no es inmediata, la aplicación quedará instalada en el Servidor y podrá ser federada manualmente por el administrador del nodo. La federación de la aplicación consiste básicamente en comunicar al servidor de soporte que la aplicación está federada y puede ser utilizada desde otros servidores, es decir se da de alta la misma en la topología de la red Link-ALL.

Diseño e implementación:

Si el nodo está federado y configurado para federar aplicaciones entonces el Federador envía la información de la aplicación al servidor de soporte para que éste la de de alta en la topología. La invocación a esta operación se hace a través del ApplicationBDelegate, éste solicita el método federateApplication del ApplicationFacadeBean y éste el método análogo al ApplicationAppService. En el ApplicationAppService se hacen las validaciones del contexto, de los permisos del usuario para federar. Se verifica además que el nodo esté federado, que la aplicación esté instalada y no esté ya federada, esto lo muestra la figura 2.1. Si pasa todas las validaciones entonces se invoca al FederationService que se encarga de la comunicación con el servidor de soporte.

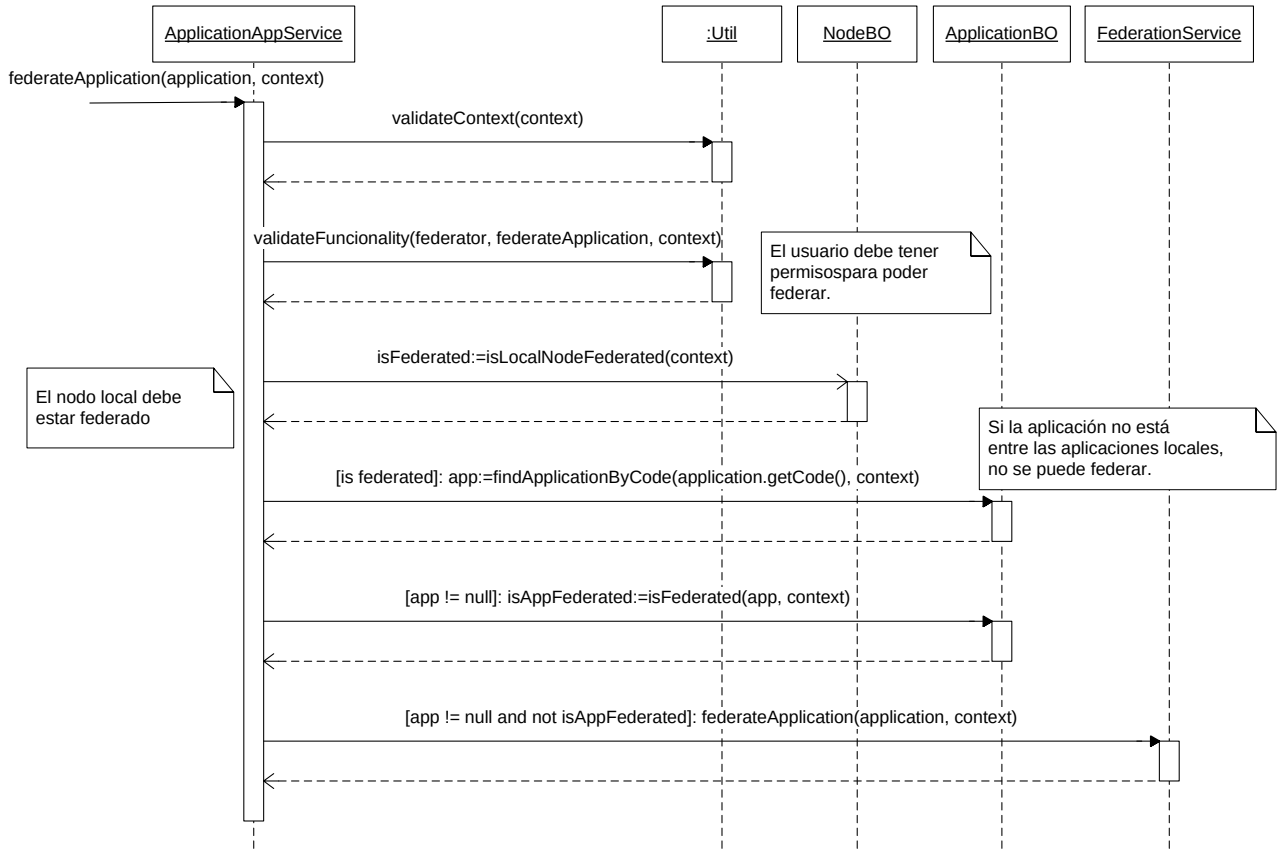


Fig. 2.1 - Federar Aplicación - Validaciones

Si el nodo que federa la aplicación no es servidor de soporte, entonces se crea un WebServiceRequest, que es un objeto para intercambiar información entre los webservices, en este caso indica que se federa una aplicación y cuál es la aplicación que se federa, finalmente se envía el pedido al servidor de soporte (Ver figura 2.2). Si el nodo es servidor de soporte simplemente ingresa la aplicación en la topología.

Por otro lado, el servidor de soporte atiende el pedido del Web Service y solicita el método ssFederateApplication del ApplicationBDelegate e invoca al ApplicationAppService a través de la fachada. En el service se hacen las validaciones y la aplicación se agrega a la topología local como se muestra en el diagrama de secuencia de la figura 2.3.

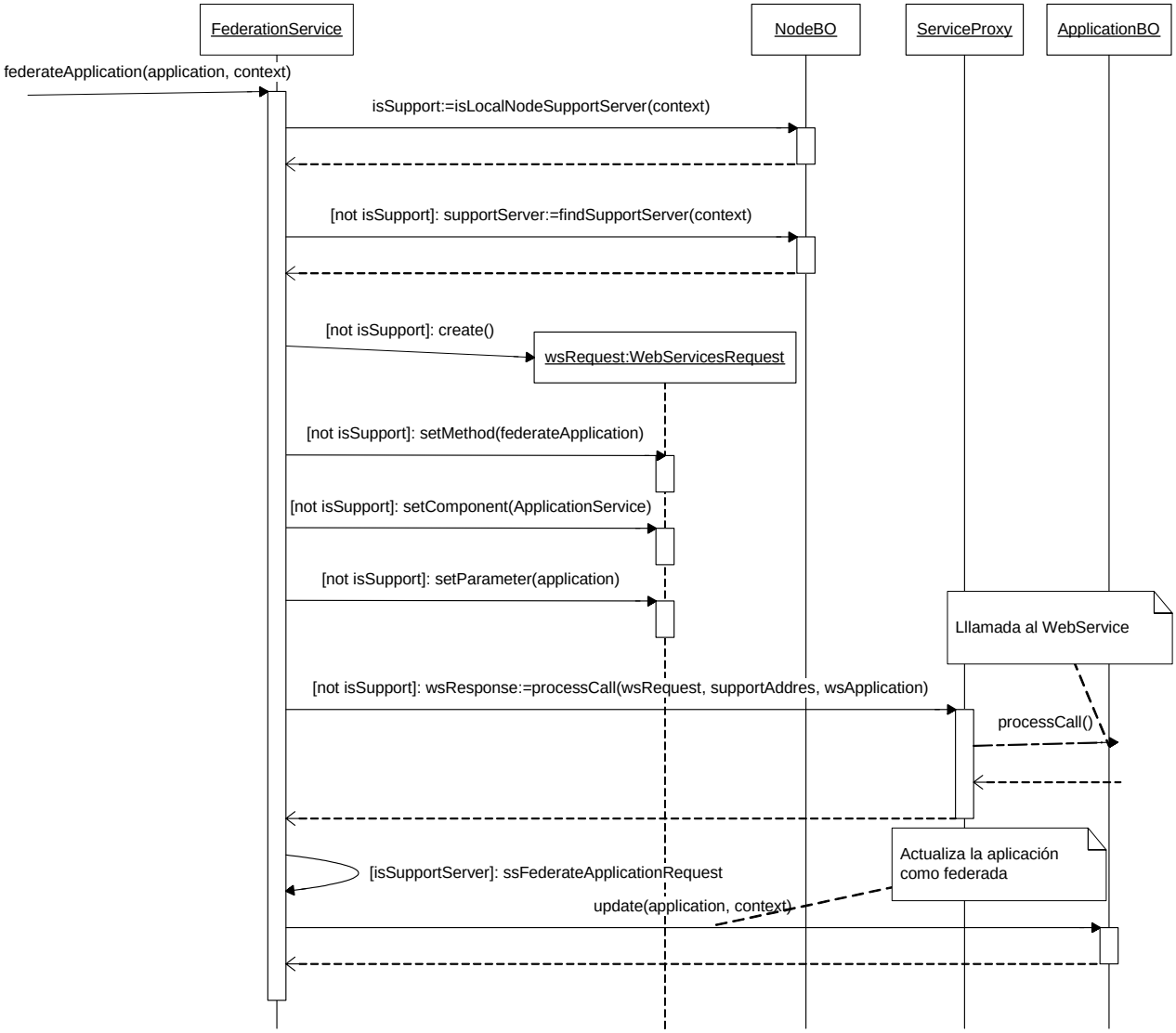


Fig. 2.2 - Federar aplicación, llamada al Web Service.

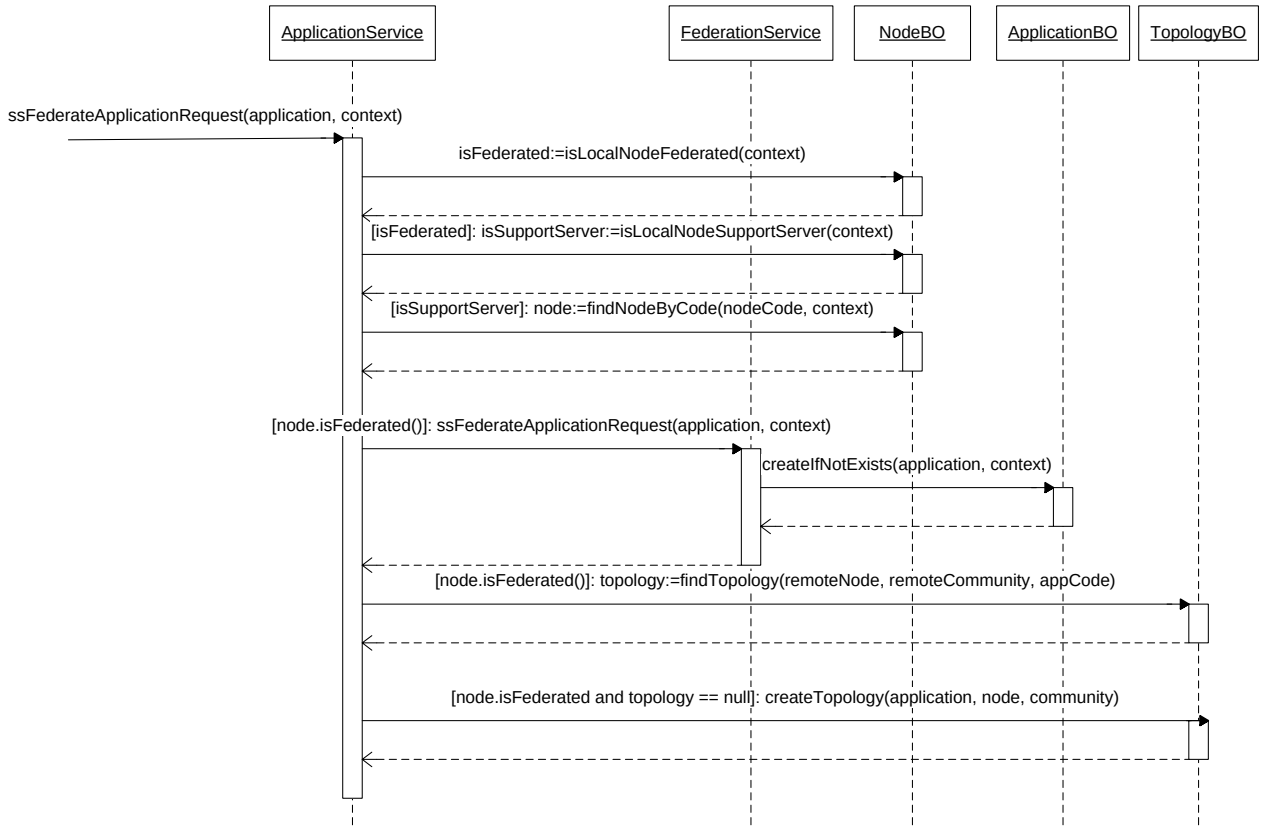


Figura 2.3 Federar aplicación en el nodo de soporte

Si bien no era un requerimiento, a los efectos de facilitar las pruebas y la visualización de esta funcionalidad, se diseñó una página web para que el administrador del nodo pueda federar una aplicación manualmente, a partir de la lista de aplicaciones federadas el usuario puede seleccionar una aplicación y federarla.

2.3. Federar Servidor

Descripción:

Esta operación permite a los servidores integrarse a la Federación Link-ALL. Un determinado servidor puede federarse como servidor de soporte o servidor común. La federación de un nodo de soporte implica que el federador ingrese en la topología del servidor a este nodo y sus comunidades, en cambio un nodo común solicita federarse al servidor de soporte, para esto el nodo debe tener un servidor de soporte configurado con anterioridad, del mismo modo que en el caso anterior se ingresa el nodo y sus comunidades en la topología del servidor de soporte, finalmente cuando el nodo ha sido federado recibe la información de la topología Link-ALL desde el nodo de soporte. Esta funcionalidad es invocada por el administrador del Servidor

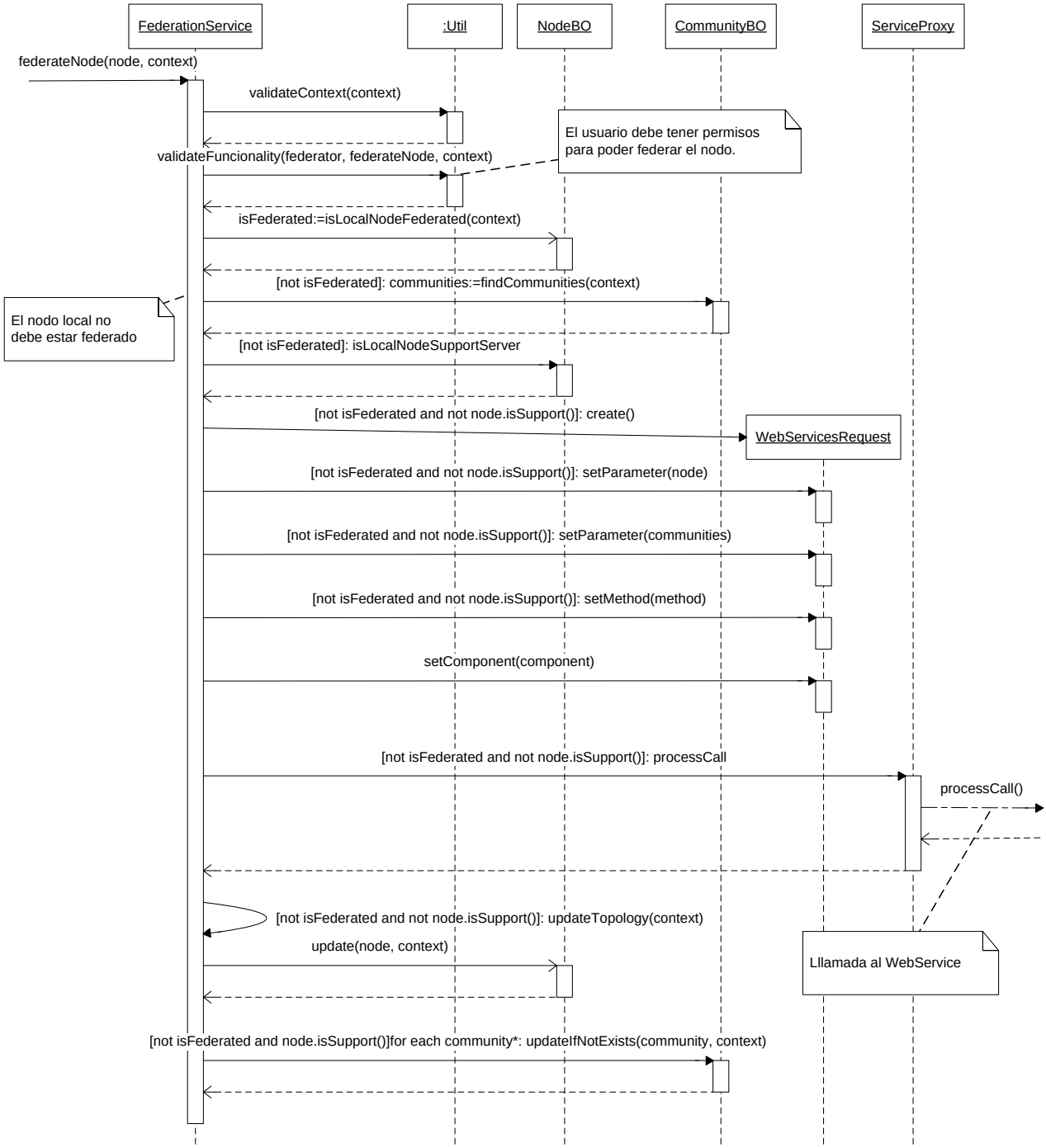


Fig. 3.1 - Federar Servidor

Diseño e implementación:

Luego de ingresar la información del servidor el usuario confirma el envío de los datos y el handler FederateNodeHandler es quien atiende el pedido. Obtiene los datos del nodo y de la sesión del usuario a partir del request y crea el Value Object del nodo a federar. Entonces realiza el pedido al FederationBDelegate, quien a su vez invoca al FederationFadeBean y éste invoca al FederationService, de manera similar que en el resto de las funcionalidades. En la figura 3.1 se visualuza el diagrama de secuencia que muestra el funcionamiento de federateNode a partir del

FederacionService. Primero se validan el contexto y los permisos del usuario para federar, luego se verifica que el nodo no esté ya federado.

Si se federa el nodo como servidor de soporte, entonces se da de alta en la federación y se agregan las comunidades del nodo en la topología.

Si no se federa como nodo de soporte, se invoca al Web Service del servidor de soporte, y se envían las comunidades del nodo, que también serán federadas. En el servidor de soporte por su lado si el nodo no está federado se ingresa en la topología y también se ingresan las comunidades que este nodo posea. Esto se ve en el diagrama de la figura 3.2.

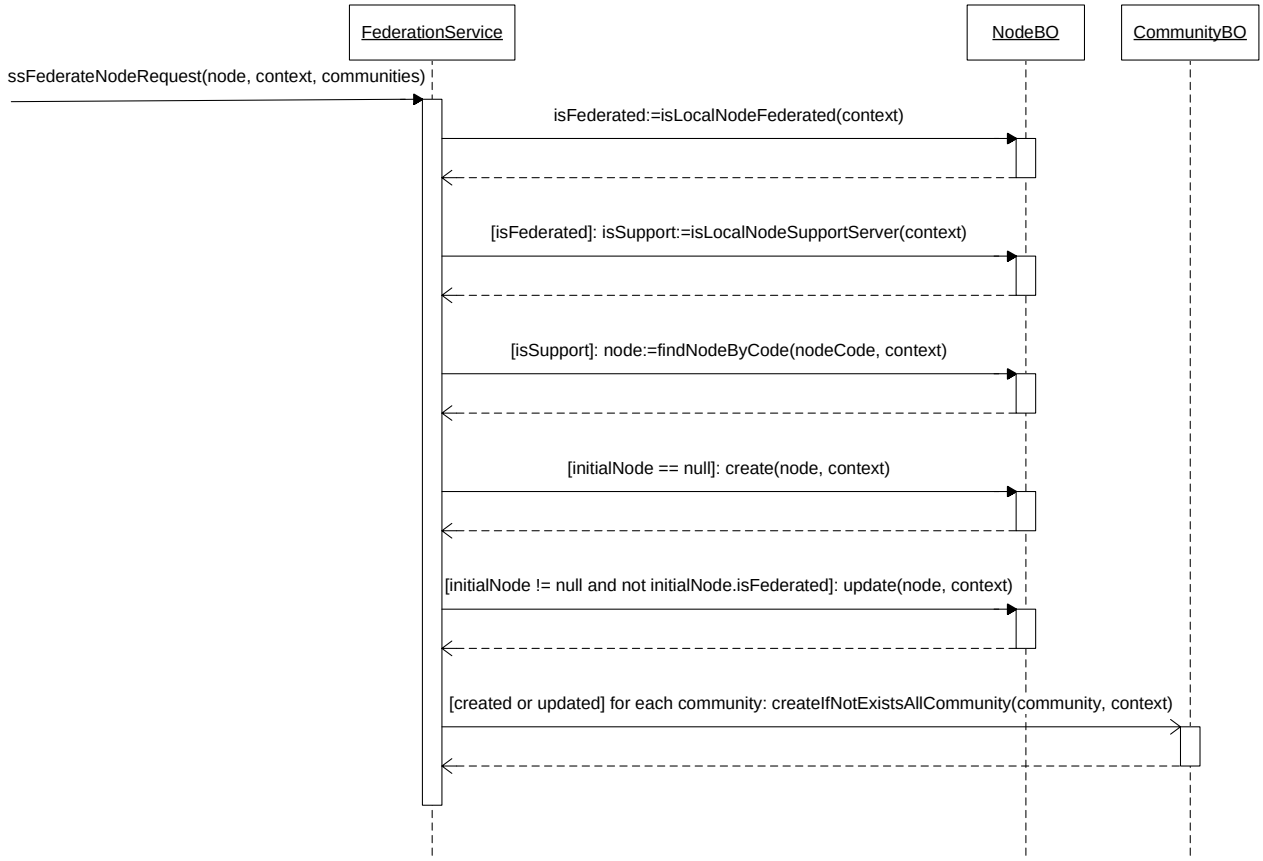


Fig. 3.2 - Federar Servidor en el servidor de soporte

2.4. Actualizar Topología

Descripción:

La actualización del mapa de la federación se hace desde el servidor de soporte hacia un nodo común, ya que todas las operaciones que un nodo efectúe en la federación son informadas o realizadas contra el servidor de soporte por lo que éste mantendrá actualizada la topología de la federación.

Pueden existir varios mecanismos para actualizar el mapa de federación en un nodo, según las características del nodo se pueden emplear uno u otro. El mecanismo de actualización debe ser configurable por el usuario administrador del servidor. Las actualizaciones podrán ser por separado por ejemplo, actualizar solo los servidores, actualizar servicios, solo aplicaciones o solo datos, o cualquier combinación.

- Mecanismos manuales de actualización

    1) Un primer mecanismo consiste en que el administrador del servidor ingrese a la herramienta administrativa y seleccione actualizar el mapa de la federación con algún criterio de filtrado, y reemplazar el mapa local por el del servidor de soporte.

    2) El administrador ingresa a la herramienta administrativa y selecciona actualizar la información de la federación, pero el servidor de soporte envía solamente la información de las modificaciones desde la sincronización anterior.

- Mecanismos automáticos

    3) El administrador configura el servidor para que cada cierto intervalo de tiempo el Federador actualice la información desde el servidor de soporte reemplazando la información local.

    4) El administrador configura el servidor para que cada cierto intervalo de tiempo (horas, días) el nodo sincronice con el servidor de soporte, y este le envíe las actualizaciones que ocurrieron desde la actualización anterior.

    5) El administrador configura el servidor para que las actualizaciones se realicen en línea. El nodo se registra en el servidor de soporte para que éste le envíe las actualizaciones al momento que ocurren. Esta actualización a diferencia de las otras ocurre desde el servidor de soporte hacia los nodos que se subscriban. Esta opción solo podrá ser configurada en los servidores con conexión permanente.

Para los mecanismos 2 y 4, el servidor de soporte deberá llevar un control de versiones sobre los paquetes de actualización y de esta forma los nodos podrán actualizar los cambios ocurridos.

Aunque en la descripción de los mecanismos de actualización de la información de la federación siempre se mencionó al servidor de soporte, esto se podrá realizar desde cualquier servidor con conexión permanente que brinde el servicio de actualización.

Aún cuando la información de la federación no esté actualizada, el Federador deberá funcionar normalmente con la información que tiene en el mapa local.

Diseño e implementación:

De los mecanismos mencionados anteriormente solo fueron implementadas la actualización manual por parte del administrador (mecanismo 1) y la actualización automática como describe el mecanismo 3. El diseño del subsistema en base a estrategias, permite la incorporación de distintas estrategias.

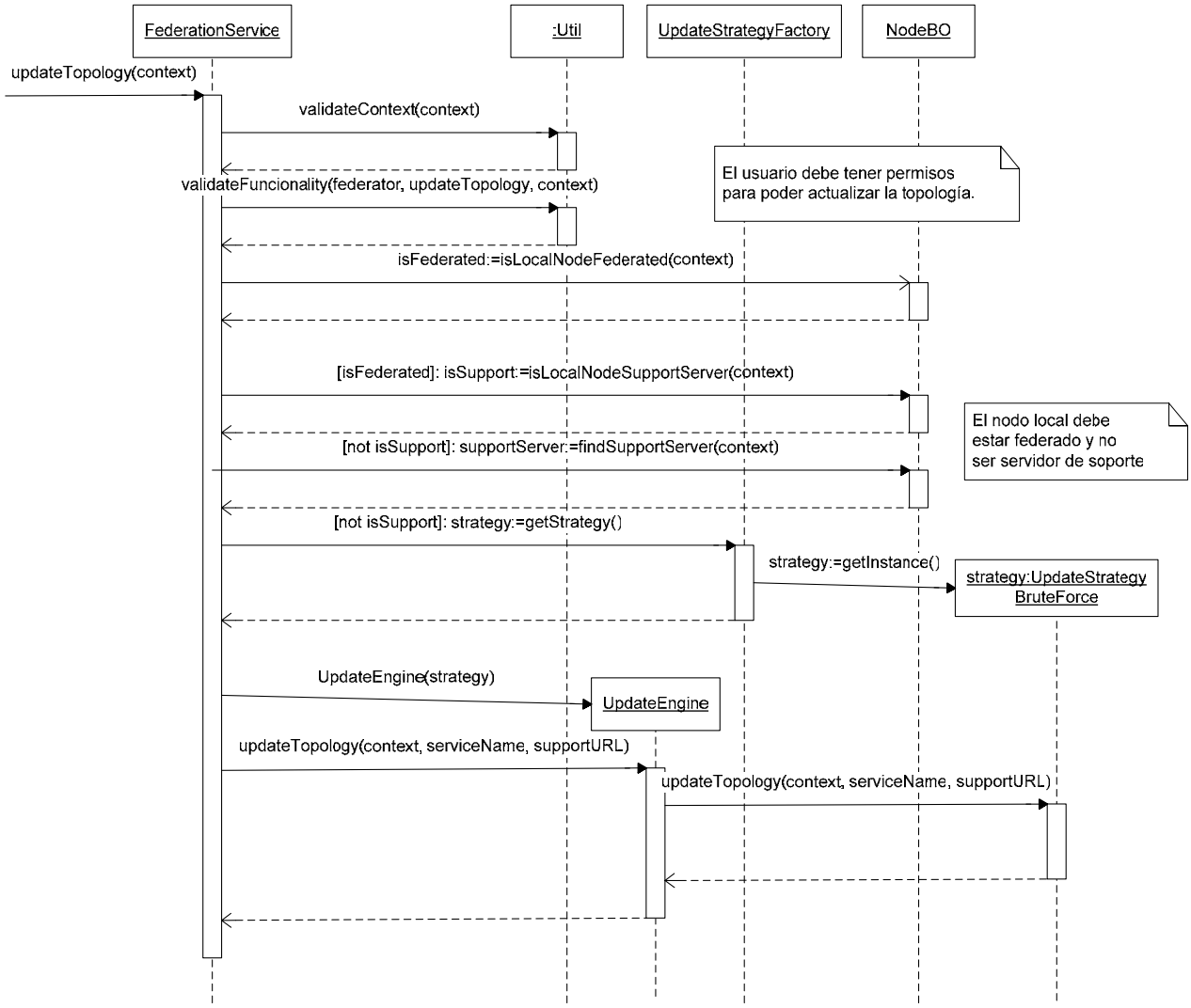


Fig. 4.1 - Actualizar Topología padrón de estrategias

El diagrama de secuencia de la figura 4.1 muestra como se utiliza el patrón de estrategias para la actualización de la topología. El FederationService invoca al motor de actualización, al que previamente se ha creado con una estrategia de actualización, que en este caso es actualización por “fuerza bruta”. Esta estrategia puede ser fácilmente reemplazada por otra más inteligente simplemente implementando la interfaz IUpdateStrategy. Por ejemplo, una actualización que modifique solo los cambios que hubo, sin tener que reemplazar toda la topología local. Las estrategias deben implementar dos métodos updateTopology, y updateTopologyRemote, que es el método que atiende el pedido en el servidor de soporte.

La figura 4.2 muestra el diseño de la estrategia Fuerza bruta para actualizar la topología en el nodo local. Básicamente borra la topologia actual conservando el nodo local, obtiene los datos de la topología del servidor de soporte e ingresa estos datos a la topología local. La figura 4.3 muestra el procesamiento de una entrada en particular de la topología.

En el servidor de soporte el pedido del webService como en los demás casos es atendido por el WebService, éste invoca al FederationBDelegate que a su vez es atendido por FederationFacadeBean. La fachada utiliza el servicio del FederationService y este sigue el comportamiento expresado en el diagrama de la figura 4.4.

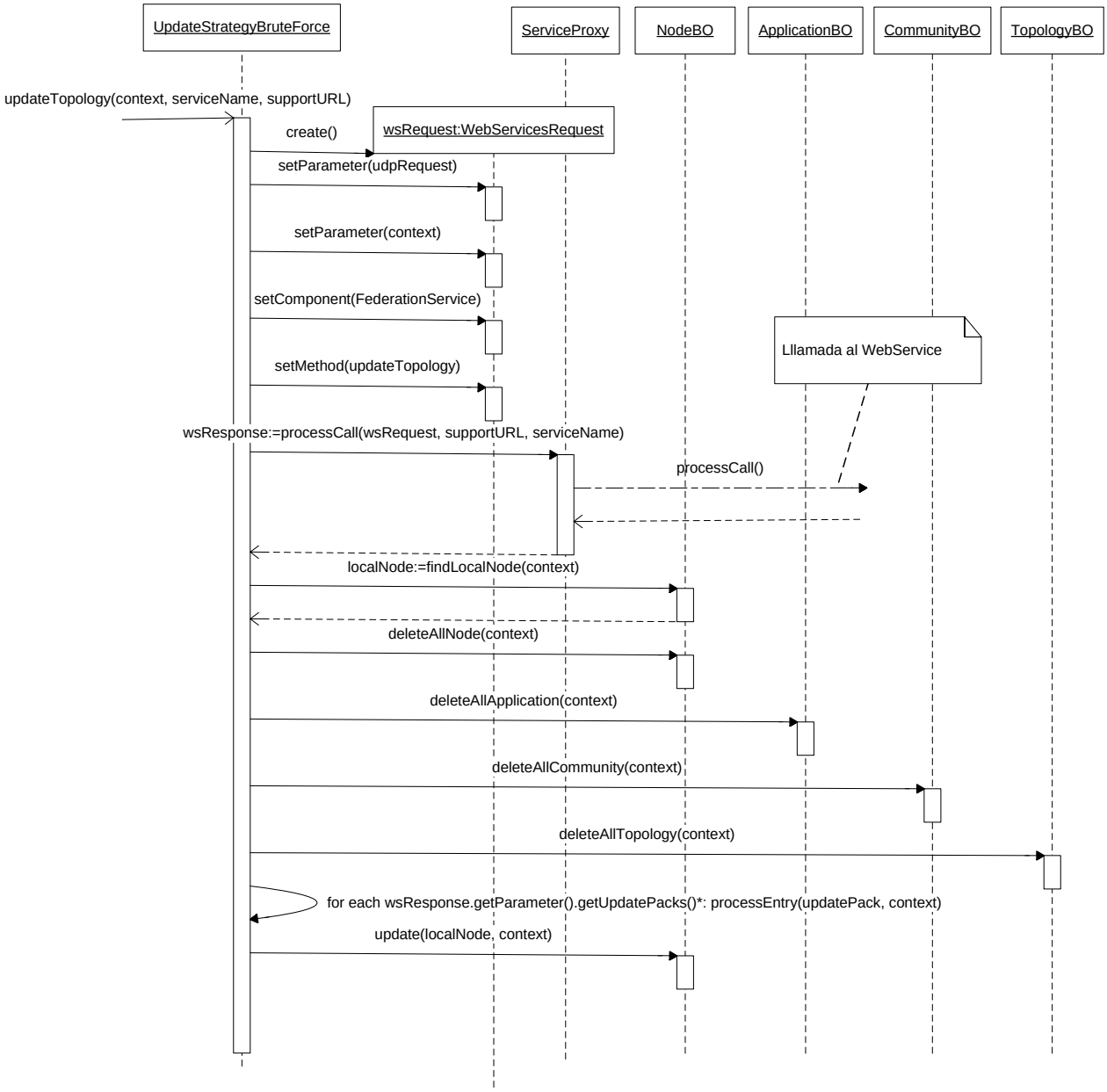


Fig. 4.2 - Actualizar Topología estrategia fuerza bruta en el nodo local

En el servidor de soporte también se sigue un patrón de estrategias para atender el pedido de la actualización de la topología (figura 4.4), la estrategia implementada fuerza bruta, lo que hace es obtener todos los datos de la topología, aplicaciones, comunidades nodos y entradas en la tabla topología, con cada uno arma un paquete de actualización, y es el paquete serializado el que envía como respuesta al web service, esto se representa en el diagrama de la figura 4.5.

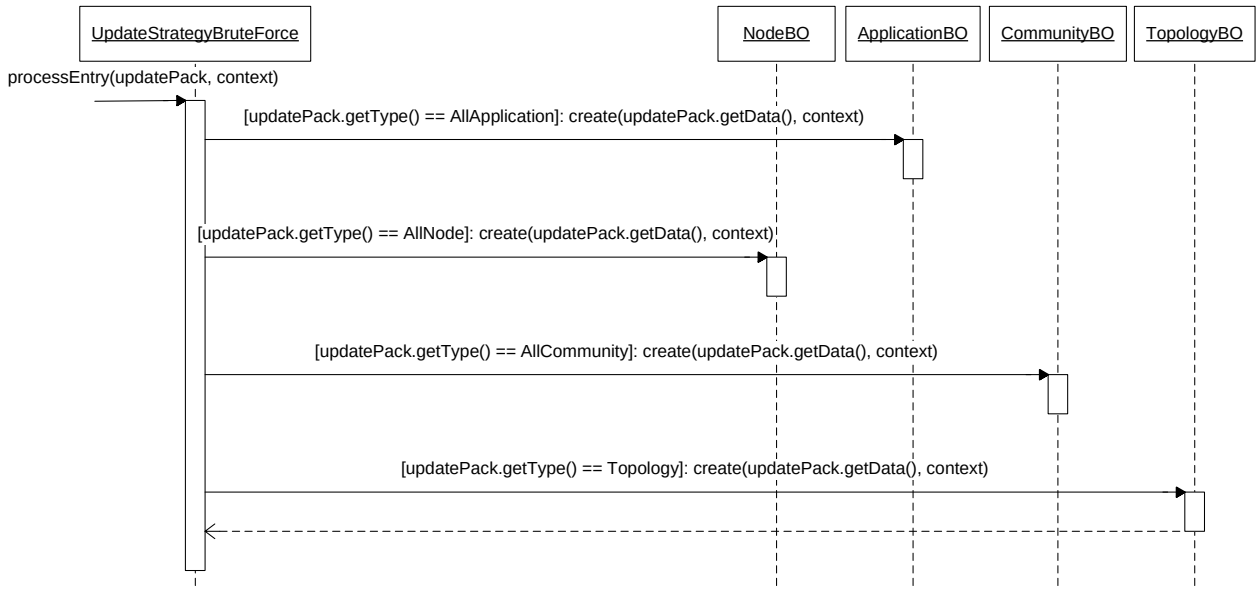


Fig. 4.3 - Actualizar Topología estrategia “fuerza bruta” procesamiento de entradas del servidor de soporte en el nodo local

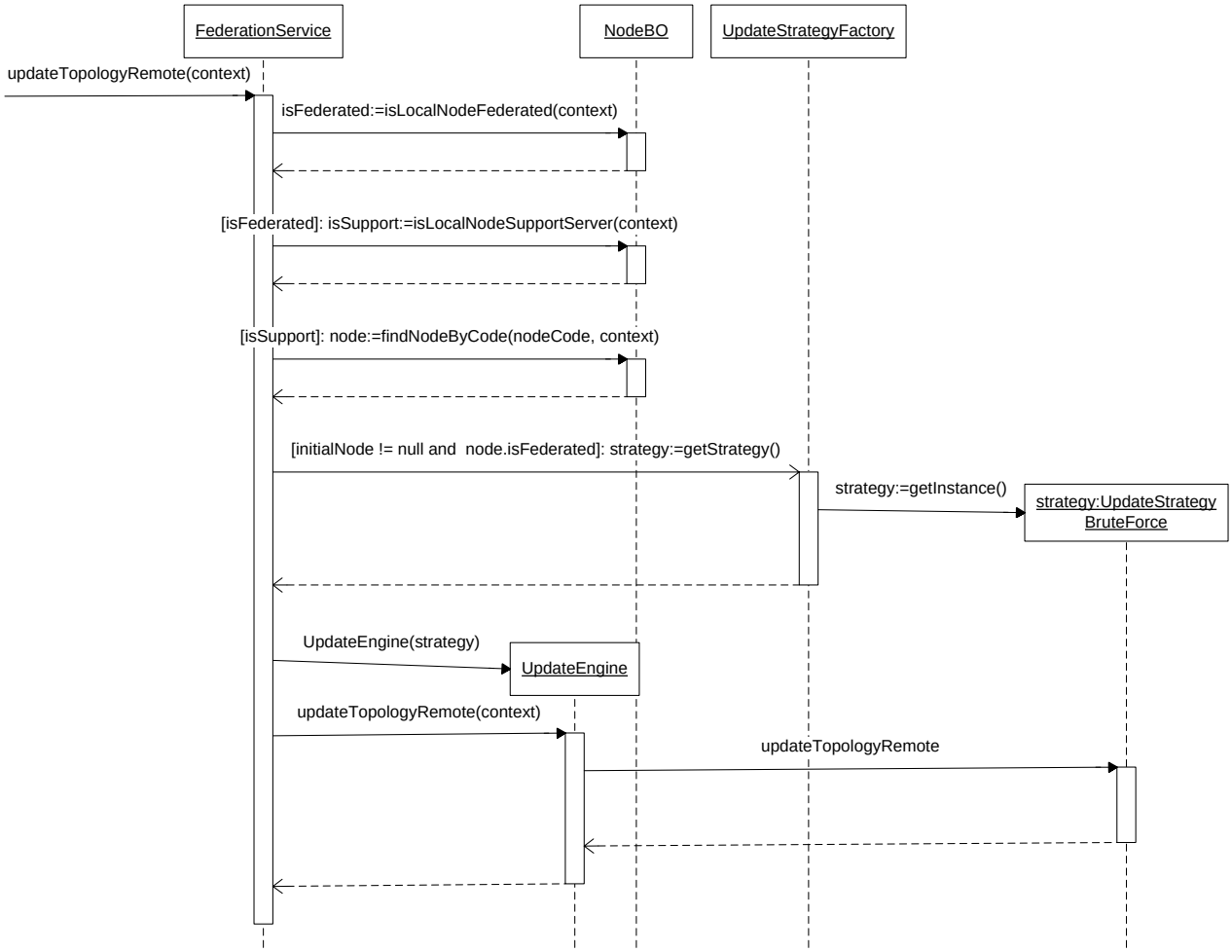


Fig. 4.4 - Actualizar Topología Atención del pedido en el servidor de soporte, patrón de estrategias

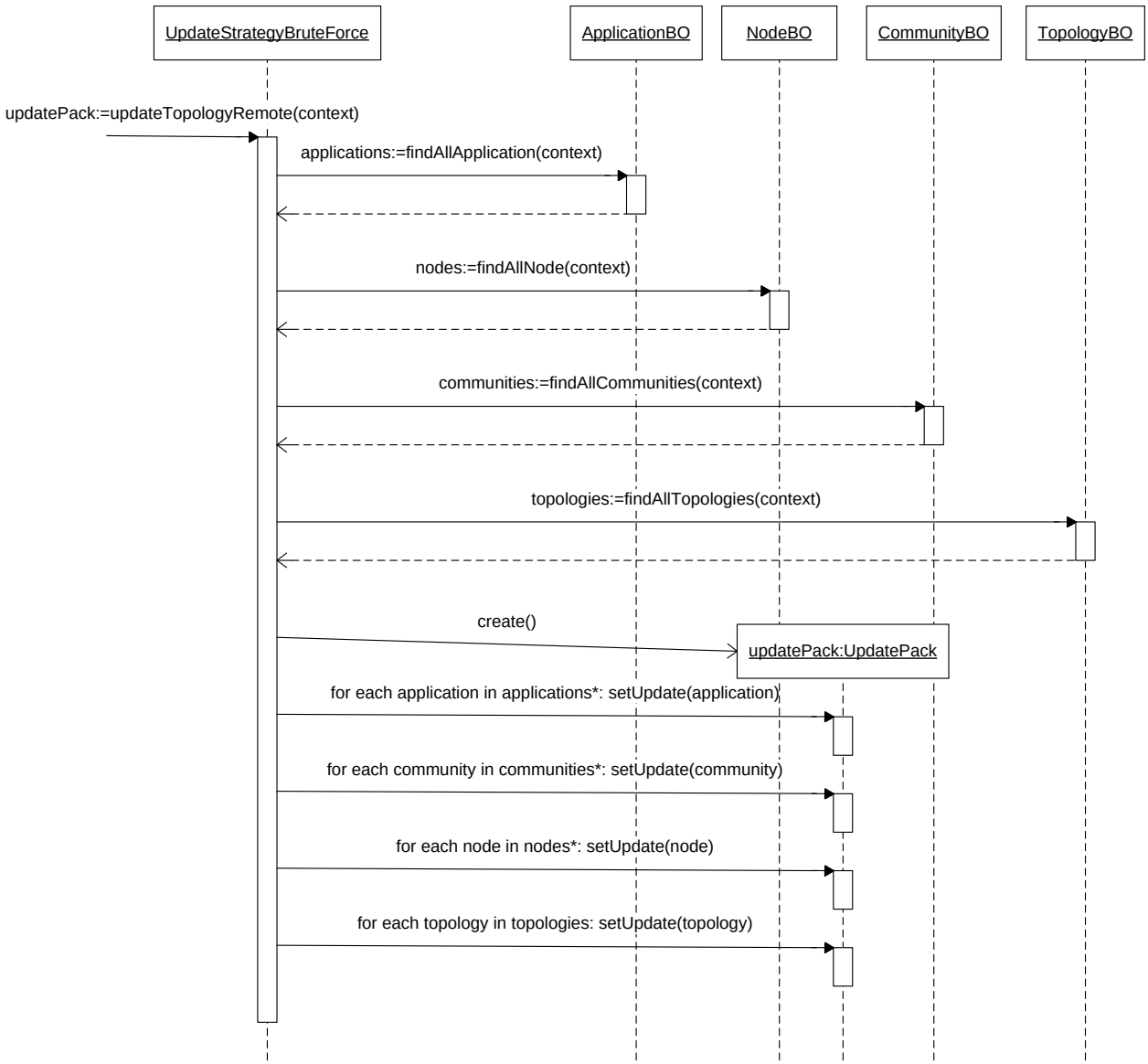


Figura 4.5 Actualizar Topología en el servidor de soporte, obteniendo los packs de actualización

2.5. Procesar Consulta

Descripción:

Esta operación permite al Federador de Datos ejecutar la consulta solicitada y retornar el resultado. La consulta a ejecutar podría ser una consulta SQL o una consulta en formato de Hibernate. El Federador será capaz de procesar tanto consultas de actualización como de selección de datos. Las consultas de selección serán ejecutadas en múltiples fuentes de datos (en la federación) y las consultas de inserción y actualización serán ejecutadas en una única fuente de datos (base de datos local). Para determinar donde se ubican las fuentes de datos el Federador de Datos se vale de la información de la topología de la red Link-ALL. Las consultas podrán ser ejecutadas o retornarán resultados siempre y cuando respeten las restricciones de seguridad correspondientes.

Las consultas de selección se hacen por medio del Federador de datos, ya que se realizan sobre toda la federación, es decir se devuelven los datos locales que cumplan las condiciones, pero también la consulta se ejecuta sobre otros servidores. El Federador de datos recoge los datos obtenidos tantos locales como remotos, y maneja la paginación de los mismos. Cada Federador presenta el mismo esquema el cual es compartido a la federación.

Diseño e implementación:

Se solicita al Federador de Datos la ejecución de una consulta SQL. Para ello se invoca al QueryBDelegate, quien a través del Service Locator realiza la llamada vía RMI al SessionBean sin estado encargado de las consultas (QueryFacade) y que actúa como fachada de los servicios de negocio. Esto se muestra en el diagrama de secuencia de la figura 5.1.

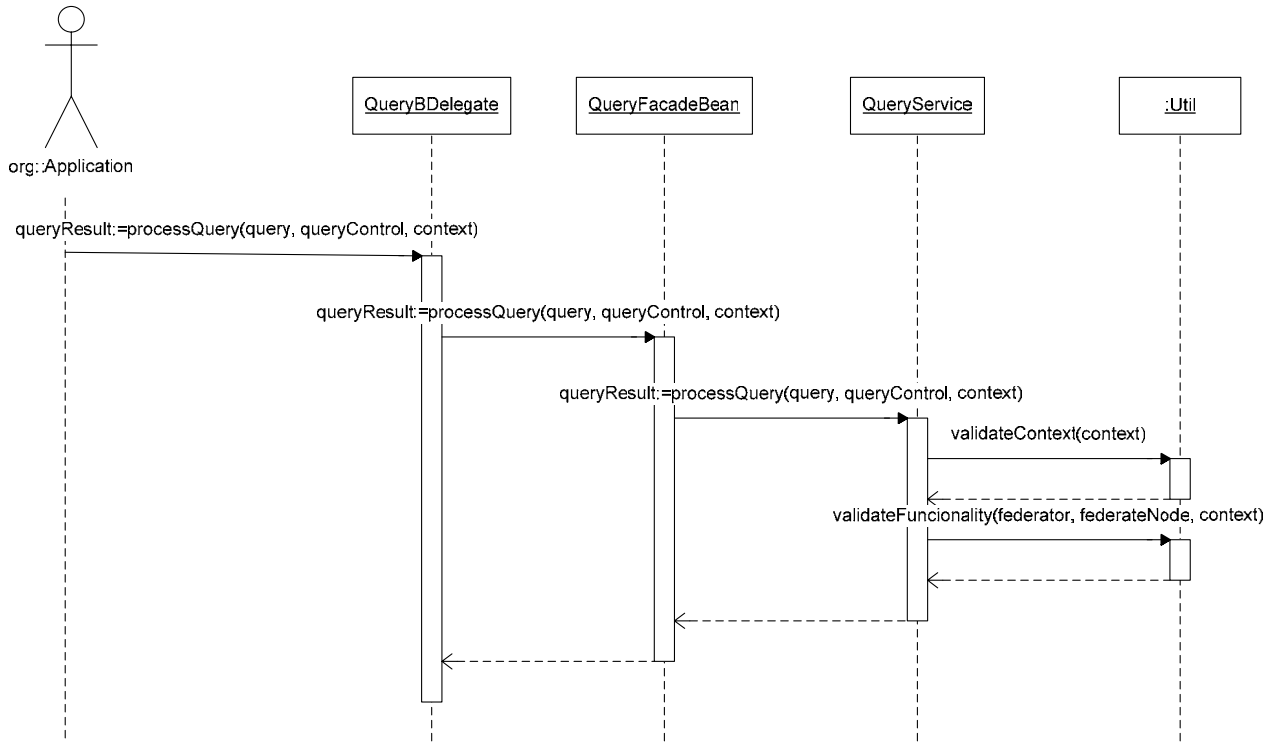


Fig. 5.2 - Procesar consulta, invocación al Business Delegate

Una vez que el servicio ha recibido una consulta, las acciones que se toman dependen del tipo de consulta recibida (selección o actualización) y el contexto (local o remoto). Si la consulta recibida es de actualización entonces se instancia un adaptador JDBC (DefaultJDBCAdapter) quien es el encargado de realizar la inserción, actualización o eliminación necesaria. Anteriormente dicha consulta es modificada donde se le anexa la información de seguridad si corresponde.

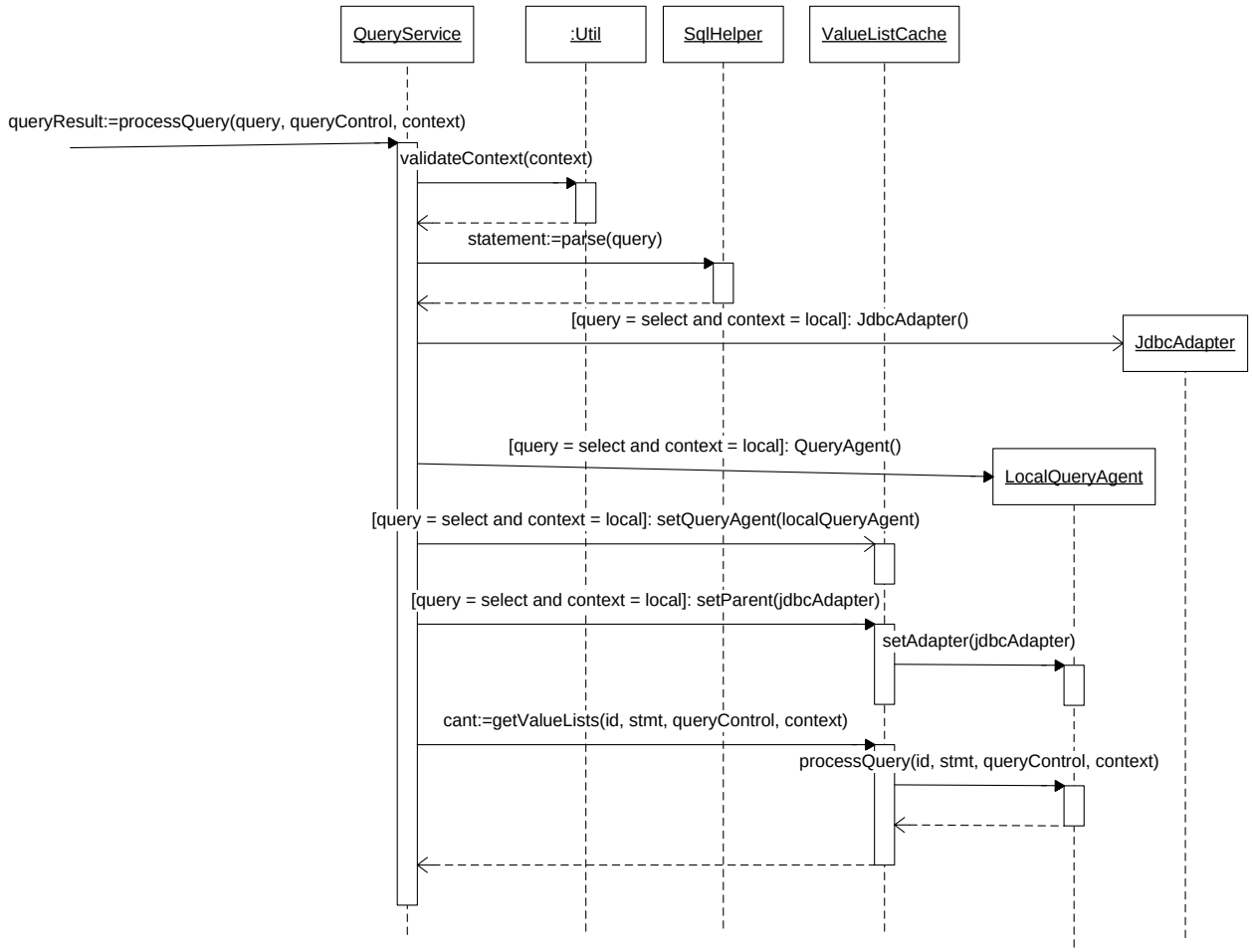


Fig. 5.2 Procesar consulta, lógica del negocio

Si la consulta es de selección realizará las acciones necesarias el administrador de listas de resultados (Vlhandler) utilizando para ello un agente de consulta local o remoto, dependiendo de las características del contexto de la llamada. Al administrador de listas de valores se le puede establecer un tiempo máximo de vida al caché en que almacena las páginas obtenidas.

Si el contexto es local, la consulta es manejada por el agente encargado de las consultas locales (LocalQueryAgent). Este agente realiza la consulta en la base de datos Global local después de haber sido examinada por el componente encargado de la seguridad (DataSecurityManager). Posteriormente la consulta original es enviada a los nodos remotos a los que el usuario tenga permisos de acceso. El conjunto de estos nodos se obtiene a través del directorio de datos (quien examina los permisos del grupo que realiza la consulta). La consulta a la base de datos local es realizada utilizando el adaptador JDBC DefaultJDBCAdapter. Una vez que se han conseguido tanto los resultados locales como remotos, el administrador de resultados (ResultManager) se encarga de unirlos y dividirlos en fragmentos según lo solicitó el cliente, para luego paginarlos y almacenarlos en memoria.

Si el contexto es remoto, la consulta es manejada por el agente encargado de las consultas remotas (RemoteQueryAgent). La misma es analizada por DataSecurityManager y luego se ejecuta en la base de datos Global local. Por último los resultados son administrados por ResultManager quien pagina y almacena en memoria los resultados.

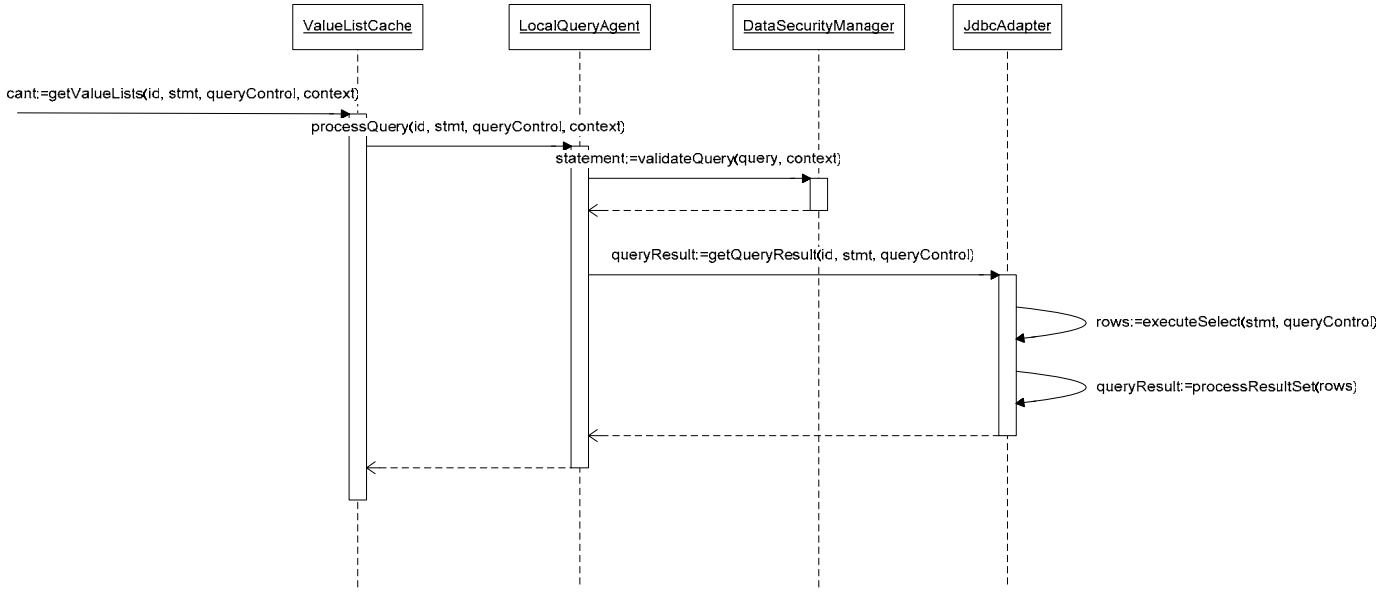


Fig. 5.3 - Obtener valor de la lista de valores retornados

3. Referencias

[1] Documentación Técnica, Fabricio Alvarez, Rodolfo Amador, Proyecto Federador Link-ALL 2004.

[2] Documento de Estado del Arte y Tecnologías, Fabricio Alvarez, Rodolfo Amador, Proyecto Federador Link-ALL 2004.

PROYECTO DE GRADO
FEDERADOR LINK-ALL

Verificación

Mayo 2005

Instituto de Computación - Facultad de Ingeniería
Universidad de la República

Estudiantes: Fabricio Alvarez
Rodolfo Amador

Tutores: Federico Piedrabuena
Raúl Ruggia

Tabla de contenido

1. Verificación del sistema

1.1. Objetivos

El objetivo de la verificación del sistema es comprobar que el producto realizado cumple adecuadamente con las funcionalidades requeridas y que lo hace satisfaciendo cierto nivel de calidad. Debido a su relevancia dentro del federador ciertas funcionalidades requerían nivel de calidad cinco y otras de nivel de calidad tres.

En el plan de verificación del sistema se establecieron los casos de uso, requerimientos funcionales y no funcionales que serían verificados [1]. A continuación se detalla la lista de estos requerimientos:

- Retornar las aplicaciones de un usuario
- Rutear un pedido de ejecución una aplicación
- Federar una aplicación
- Desfederar una aplicación
- Solicitar permisos de acceso a una aplicación
- Otorgar permisos de acceso a una aplicación
- Denegar acceso a una aplicación
- Alta de un servidor en la Federación
- Baja de un servidor en la Federación
- Visualizar y consultar el Mapa de la Federación
- Actualizar el Mapa de la Federación
- Alta de Comunidades
- Baja de Comunidades
- Actualizar datos de Comunidades
- Consultas de selección de datos
- Consultas de actualización de datos
- Solicitar permisos de acceso a datos
- Otorgar permisos de acceso a datos
- Denegar acceso a datos

En el documento de casos de prueba de aceptación se estableció que las siguientes funcionalidades requerían nivel de calidad cinco:

- Retornar las aplicaciones de un usuario
- Rutear un pedido de ejecución una aplicación
- Federar una aplicación
- Desfederar una aplicación
- Alta de un servidor en la Federación
- Baja de un servidor en la Federación
- Actualizar el Mapa de la Federación

Para lograr un nivel de calidad cinco es necesario cumplir con las siguientes condiciones:

- Relevar los requerimientos para el sistema
- Validar y priorizar los casos de uso generados a partir de los requerimientos.
- El diseño debe incluir una especificación de los componentes a construir.
- Realizar tests de los casos de usos, test funcional del sistema, test de aceptación, tests de regresión, de componentes y de integración de componentes
- Establecer el cubrimiento que deben tener los casos de pruebas sobre los requerimientos, casos de uso y componentes
- Realizar el plan de la verificación de los tests a realizar y llevar un seguimiento de las todas fallas encontradas que sean reproducibles
- Automatizar en lo posible los casos de pruebas.

- Tomar métricas sobre el cubrimiento y los test realizados que ayuden a determinar el nivel de calidad alcanzado.
- No se deben encontrar errores en la prueba de aceptación
- La cantidad de fallas por caso de prueba de componentes debe ser menor a 5%
- La cantidad de fallas sobre la cantidad de casos de pruebas de casos de uso debe ser inferior a 10 % en lo que respecta a las funcionalidades seleccionadas para este nivel de calidad.

1.1.1. Alcance

Dentro del plan se estableció que el alcance de las pruebas de verificación y validación abarcaba las pruebas unitarias, de integración, de sistema y funcionales correspondientes al Federador. Las pruebas de performance, carga, volumen, esfuerzo (estrés), seguridad, control de acceso, fallas, recuperación y configuración fueron comprendidas dentro del alcance pero su realización quedaba sometida a los tiempos que se manejaran durante el transcurso del proyecto.

1.1.2. Priorizar los requerimientos

La intención fue reducir el esfuerzo sin desmejorar la calidad del producto final. Debido a esto se determinaron qué funcionalidades de la aplicación serían más intensamente testeadas que otras. Para ello se empleó una técnica combinada basada en User Profiles (perfiles de usuario) y Análisis de Riesgo a lo que se agregó el nivel de calidad requerido de las funcionalidades.

User Profiles: Consiste básicamente en testear las partes más usadas con un rango amplio de entradas, y no tanto para aquellas poco usadas. Esto puede ayudar a asegurar la satisfacción del usuario. Se realizó un análisis de las operaciones del sistema, teniendo en cuenta la frecuencia con que se utiliza la operación y qué tan crítica es para el mismo.

Esta técnica fue combinada con la de análisis de riesgo, teniendo presente para ello los riesgos de negocio que puedan existir. El resultado de la clasificación fue obtenido aplicando una estrategia de promedio (conservativa en algunos casos).

Se presenta a continuación el ranking de testing obtenido para los casos de uso del Federador:

Funcionalidad	Nivel de Calidad	Frecuencia	Criticidad	Riesgo	Prioridad para testing
Rutear un pedido de ejecución de aplicación	5	Alta	Alta	Medio	Alta
Devolver lista de aplicaciones del Usuario	5	Alta	Alta	Medio	Alta
Federar una Aplicación	5	Media	Alta	Medio	Alta
Desfederar una Aplicación	5	Baja	Alta	Medio	Alta
Solicitar permisos de acceso a una aplicación	3	Media	Baja	Bajo	Baja
Otorgar permisos de acceso a una aplicación	3	Media	Baja	Bajo	Baja
Denegar acceso a una aplicación	3	Media	Baja	Bajo	Baja
Alta de un Servidor en la Federación	5	Baja	Alta	Alto	Alta
Baja de un Servidor de la Federación	5	Baja	Media	Medio	Media
Visualizar y consultar el Mapa de la Federación	3	Alta	Media	Bajo	Media
Actualizar Mapa de la Federación	5	Media	Alta	Alto	Alta
Alta de Comunidades	3	Baja	Baja	Medio	Baja
Baja de Comunidades	3	Baja	Baja	Medio	Baja
Actualizar datos de	3	Media	Baja	Medio	Baja

Comunidades					
Consultas de selección de datos	3	Alta	Alta	Alto	Media
Consultas de actualización de datos	3	Media	Alta	Medio	Media
Solicitar permisos de acceso a datos	3	Media	Baja	Bajo	Baja
Otorgar permisos de acceso a datos	3	Media	Baja	Bajo	Baja
Denegar acceso a datos	3	Media	Baja	Bajo	Baja

2. Pruebas Realizadas

Dado que un testing total de todos los componentes usualmente no es posible fue necesario utilizar técnicas que permitieran minimizar los escenarios y casos de verificación en una forma efectiva, posibilitando lograr además el cubrimiento más amplio posible con el menor esfuerzo. Se combinaron varias técnicas de verificación dado que ha sido probado que es más efectivo que utilizar una sola. Esta combinación incluyó un análisis funcional, partición en clases de equivalencia y valores límites.

El análisis funcional involucra analizar el comportamiento esperado del sistema de acuerdo a las especificaciones funcionales y generar uno o más procedimientos de testing para cada función o rasgo del Sistema. La partición en clases de equivalencia implica identificar rangos de entradas y condiciones iniciales que se esperan produzcan el mismo resultado.

Cumpliendo con el alcance propuesto fueron realizadas las pruebas de componentes, de integración de componentes, de casos de uso y del sistema. Las pruebas de performance, carga, volumen, esfuerzo, configuración y demás no pudieron ser realizadas debido a los tiempos manejados para la implementación del proyecto.

Para el seguimiento de errores se evaluaron las herramientas Bugzilla [8] y BugCollector [9]. En un principio se instaló y probó el Bugzilla pero éste es pensado para proyectos de mayor escala (mayor numero de desarrolladores). Otra herramienta evaluada fue BugCollector es de fácil instalación y uso pero requiere de una licencia; por lo que finalmente se decidió utilizar una planilla Excel con los campos relevantes para el seguimiento de los errores.

2.1. Pruebas de componentes y de integración

Las pruebas de componentes incluyeron pruebas automáticas sobre los componentes que se consideraron críticos como ser ApplicationManager, FederationManager y QueryManager [2]. Las pruebas automáticas sobre componentes se realizaron con JUnit [3], que es una herramienta de fácil uso y se puede integrar sencillamente con el IDE Eclipse [6]. A partir de ellas se pudo realizar una métrica del cubrimiento de los tests de componentes. Esta medida se realizó con la herramienta automática DJUnit [12].

2.2. Pruebas de requerimientos y/o casos de uso

Los tests de casos de uso incluyeron varios casos de prueba por cada escenario de los mismos. En las funcionalidades de nivel de calidad cinco se diseñaron casos de prueba utilizando la técnica de partición en clases de equivalencia. Algunas de estas pruebas fueron realizadas automáticamente, pero en ellas no se incluyó la interfaz gráfica. Las pruebas de la interfaz gráfica, así como también las que incluían comunicación entre servidores, fueron realizadas manualmente.

Para ejemplificar se detallan a continuación los casos de prueba de los casos de uso considerados más relevantes del federador de aplicaciones *rutear un pedido de ejecución una aplicación* y *federar aplicación*.

2.2.1. Rutear pedido de ejecución de una aplicación

Un usuario solicita la ejecución de una aplicación, si el usuario tiene permisos de acceso a la aplicación se devuelve la dirección de la aplicación, ya sea local o remota, si la aplicación es remota, además de la dirección de la aplicación se devuelve la dirección del servidor donde está instalada y el identificador de la sesión remota creada en este servidor.

Las entradas son el código de la aplicación y el contexto de la invocación.

Por el contrato de la funcionalidad [4] se asume que el contexto es válido, por lo que las clases de equivalencia identificadas corresponden a la aplicación a rutear:

1. codigo de aplicación nulo
2. código de aplicación vacío
3. código de aplicación de menos de 50 caracteres (es el tamaño del atributo en la BD)
   i) código de aplicación existente
      a. instalada localmente
         – aplicación federada
         – aplicación no federada
      b. aplicación remota
         – aplicación federada
         – aplicación no federada
   ii) código de aplicación no existente

4. código de aplicación con más de 50 caracteres.

A partir del contexto se puede validar si el usuario tiene o no permisos para acceder a la aplicación por lo que también existen las clases de equivalencia: {usuario con permisos, usuario sin permisos}

En este caso de uso se identificaron 4 escenarios:

1. Servidor origen federado, servidor destino federado, comunicación correcta
2. Servidor origen federado, servidor destino federado, problemas en la comunicación
3. Servidor origen no federado
4. Servidor origen federado, servidor destino no federado, comunicación correcta.

Basándose en las clases de equivalencia de la aplicación y los escenarios identificados se diseñaron los siguientes casos de prueba para rutear aplicación:

Esce-nario	Entrada	Salida esperada	Num. Caso
1	Aplicación local, federada Usuario con permisos	URL de la aplicación solicitada.	1
1	Aplicación local, no federada Usuario con permisos	URL de la aplicación solicitada.	2
1	Aplicación local, federada Usuario sin permisos	Error, el usuario no tiene permisos para ejecutar la aplicación.	3
1	Aplicación local, no federada Usuario sin permisos	Error, el usuario no tiene permisos para ejecutar la aplicación.	4
1	Aplicación remota, federada en top. local y federada en top. remota, Usuario con permisos.	URL de la aplicación. Sesión remota creada.	5
1	Aplicación remota, federada en top. local y no federada en top. remota, Usuario con permisos.	Error, la aplicación no se encuentra federada en el servidor remoto.	6
1	Aplicación remota, federada en top. local, no instalada en servidor remoto Usuario con permisos.	Error, la aplicación no se encuentra instalada en servidor remoto.	7
1	Aplicación remota, no federada en top. local Usuario con permisos.	Error, la aplicación no se encuentra federada.	8
1	Aplicación remota, federada en top. local, Usuario sin permisos	Error, el usuario no tiene permisos para ejecutar la aplicación.	9
1	Aplicación remota, no federada en top. local, Usuario sin permisos	Error, el usuario no tiene permisos para ejecutar la aplicación.	10
1	Código de aplicación no existente.	Error, código de aplicación incorrecto.	11

1	Código de Aplicación no válido: cadena nula	Error, código de aplicación incorrecto.	12
1	Código de Aplicación no válido: cadena vacía	Error, código de aplicación incorrecto.	13
1	Código de Aplicación no válido: cadena de más de 50 caracteres	Error, código de aplicación incorrecto.	14
1	Aplicación nula	Error, aplicación nula.	15
2	Aplicación local Usuario con permisos	URL de la aplicación solicitada.	16
2	Aplicación remota, federada local Usuario con permisos.	Error, No se pudo conectar con el servidor de la aplicación	17
3	Aplicación local Usuario con permisos	URL de la aplicación solicitada.	18
3	Aplicación remota, federada local Usuario con permisos.	Error, servidor local no federado	19
4	Aplicación remota, federada local Usuario con permisos.	Error, servidor remoto no federado	20

2.2.2. Federar aplicación

Se comparte a la federación una aplicación instalada en el servidor, se debe informar al nodo de soporte. Esta funcionalidad tiene como datos de entrada el código de la aplicación y el contexto de la llamada. Puede ser invocada directamente por el deployer al instalar una aplicación o manualmente por el administrador.

Las entradas son el código de la aplicación y el contexto de la invocación [4].

Al igual que en rutear aplicación, por el contrato de la funcionalidad se asume que el contexto es válido, por lo que las clases de equivalencia identificadas corresponden a la aplicación a federar y éstas son similares a las clases de equivalencia de rutear aplicación. De la misma forma se puede dividir al usuario en usuario con permisos para federar y usuario sin permisos para federar.

Los escenarios identificados para federar aplicación son:

1. Servidor de soporte, federado
2. Servidor no federado
3. Servidor común, federado, con conexión, servidor de soporte federado
4. Servidor común, federado, con conexión, servidor de soporte no federado
5. Servidor común, federado, con problemas de conexión.

A partir de las los escenarios identificados y de las clases de equivalencia mencionadas se diseñaron los siguientes casos de prueba:

Esce-nario	Condiciones de entrada	Salida esperada	Núm. de caso
1	Aplicación instalada y no federada Usuario con permisos para federar	Aplicación ingresada en la topología de la red Link All.	1
1	Aplicación instalada y federada Usuario con permisos para federar	Error aplicación ya federada	2
1	Aplicación instalada y no federada Usuario sin permisos para federar	Error, el usuario no tiene permisos para federar aplicaciones.	3
1	Aplicación instalada y federada Usuario sin permisos para federar	Error, el usuario no tiene permisos para federar aplicaciones.	4
1	Aplicación no instalada Usuario con permisos para federar	Error, código de aplicación incorrecto.	5
1	Código de Aplicación no válido: cadena nula	Error, código de aplicación incorrecto.	6
1	Código de Aplicación no válido: cadena	Error, código de aplicación incorrecto.	7

	vacía		
1	Código de Aplicación no válido: cadena de más de 50 caracteres	Error, código de aplicación incorrecto.	8
1	Aplicación nula	Error, aplicación nula.	9
2	Aplicación instalada y no federada Usuario con permisos para federar	Error, servidor no federado	10
3	Aplicación instalada, no federada local , no federada SS Usuario con permisos para federar	Aplicación ingresada en la topología local y del SS.	11
3	Aplicación instalada, no federada local, federada SS Usuario con permisos para federar	Aplicación ingresada en top local y mensaje indicando que ya esta federada en SS	12
3	Aplicación instalada, federada local Usuario con permisos para federar	Error aplicación ya federada	13
3	Aplicación instalada y no federada Usuario sin permisos para federar	Error, el usuario no tiene permisos para federar aplicaciones.	14
4	Aplicación instalada no federada local Usuario con permisos para federar	Error, servidor de soporte no esta federado	15
5	Aplicación instalada, no federada local Usuario con permisos para federar	Error, no se pudo establecer la comunicación con el SS	16

A continuación se describen los tests realizados sobre los demás casos de uso de nivel de calidad cinco.

Retornar las aplicaciones de un usuario

Los casos de pruebas consistieron básicamente en asignar permisos a un usuario sobre determinadas aplicaciones y comprobar que en el resultado estén todas las aplicaciones asignadas y solamente estas aplicaciones. En las pruebas se incluyeron aplicaciones locales y remotas, usuarios que pertenecían a uno o más grupos, servidores federados y no federados y las combinaciones entre estas entradas.

Alta de un servidor en la Federación

Las entradas para federar un servidor son los datos del servidor y un contexto válido. Para generar los casos de prueba se consideraron las diferentes clases de equivalencia para las entradas y las condiciones de ejecución. Por ejemplo el usuario puede tener o no permisos para federar el servidor, el nodo puede federarse como nodo común o como servidor de soporte. Si se federa como nodo común puede o no tener configurado el servidor de soporte contra el que se federará. Los resultados esperados varían de acuerdo a las entradas y condiciones de ejecución. Si en el test el servidor se federa como nodo común el resultado se debe verificar tanto en el nodo como en el servidor de soporte.

Baja de un servidor en la Federación

La baja de un servidor en la federación solo tiene como entrada un contexto válido ya que solo se da de baja el servidor local, por lo tanto las posibilidades son que el usuario tenga o no tenga permisos para dar de baja el servidor. También se realizaron casos de prueba teniendo en cuenta las mismas condiciones de ejecución que el alta del servidor. Los resultados esperados dependen de las condiciones de entrada, si en el test el nodo no es servidor de soporte se debe chequear en ambos nodos.

Actualizar el Mapa de la Federación

La entrada para el caso de uso actualizar mapa de la federación es simplemente un contexto válido, por lo que las clases de equivalencia se limitan a que el usuario tenga o no permisos para la funcionalidad. Las condiciones de ejecución son las mismas que para el caso anterior. Los casos de prueba se realizaron en base a combinaciones de la entrada y las condiciones de ejecución. Para que el resultado de la operación sea exitoso el nodo debe estar federado y no ser servidor de soporte,

además el usuario debe tener permisos para poder actualizar el mapa. En este caso para que el test no falle el mapa de la federación obtenido debe ser equivalente al mapa del servidor de soporte.

De manera similar fueron realizados los casos de pruebas para los demás casos de uso de la aplicación, los que no serán detallados en este documento por su extensión.

2.3. Pruebas de regresión

El proyecto fue realizado en cuatro iteraciones, en cada iteración se desarrollaron diferentes funcionalidades y se ejecutaron tests para las funcionalidades implementadas en esa iteración, estos tests fueron reutilizados y sirvieron como tests de regresión en las siguientes iteraciones. En el pasaje entre las iteraciones, algunos de los tests fueron modificados a raíz de los cambios de versión en la plataforma Link-ALL, que en ciertos casos implicaron cambios en las tablas de la base de datos administrativa y por lo tanto cambio en el juego de datos utilizados para correr los tests.

2.4. Pruebas del sistema

Las pruebas del sistema incluyeron fundamentalmente las pruebas funcionales, realizándose mayor énfasis en los casos de uso prioritarios, en las que se testeó el federador como un todo, realizándose ciclos de pruebas que incluyeron varias funcionalidades.

Se realizaron tests que incluyeron pruebas con un solo servidor para testear sobretodo el funcionamiento de un servidor de soporte y otras que involucraban la comunicación entre más de un servidor para simular el funcionamiento real de la red Link-All.

Para realizar estas pruebas se utilizaron las plataformas Windows 2000, Windows XP, Suse Linux 9.0 y los navegadores Mozilla/5.0 (Firefox 1.0), Internet Explorer 6.0, y Konqueror.

2.5. Pruebas de Aceptación

Se determinó como pruebas de aceptación, los casos de prueba de casos de uso y del sistema que involucren a las funcionalidades que requieran nivel de calidad cinco.

Una de las pruebas de aceptación se realizó al finalizar la tercera iteración, luego de la implementación de todas las funcionalidades del federador de aplicaciones. Dicha prueba incluyó un caso de prueba general del sistema, que abarcó todos los casos de uso. Durante la prueba se marcaron algunos errores menores, relacionados principalmente con la interfaz de usuario.

3. Resultados Obtenidos

3.1. Cubrimiento:

3.1.1. Cubrimiento de Especificación de Requerimientos.

El cubrimiento que se estableció aceptable en cuanto a la Especificación de Requerimientos es tener al menos un caso de prueba por cada requerimiento especificado en el alcance del producto. Por lo visto en las secciones anteriores se puede decir que los requerimientos del sistema fueron cubiertos aceptablemente.

Para realizar la medición y ver que tanto se cumple con el criterio establecido se realizó un análisis de los requerimientos para ver cual está testeado y se tomó en cuenta la partición de las entradas y salidas. En este análisis se observó que existe más de un caso de prueba para cada requerimiento funcional del sistema, y que en su mayoría se generaron casos para cada clase de equivalencia relevante.

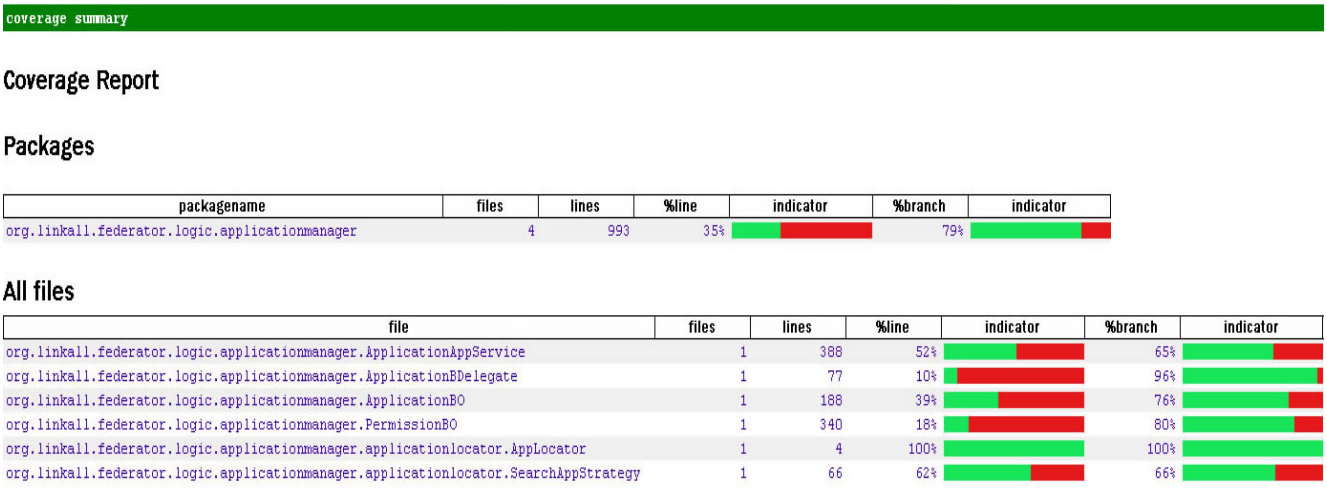
3.1.2. Cubrimiento de Casos de Uso.

El criterio que se consideró aceptable para el cubrimiento de Casos de Uso es tener casos de prueba para todos los escenarios posibles de cada caso de uso. Para los casos de uso considerados como críticos además se debían generar casos de prueba utilizando la técnica de partición en clases de equivalencia. Como se muestra en la sección 5.2.2, donde se describen los casos de pruebas de las funcionalidades críticas, se cumplió aceptablemente con el criterio de cubrimiento establecido.

3.1.3. Cubrimiento de Componentes.

El criterio que se estableció en cuanto a componentes fue cubrir todas las funcionalidades de aquellos componentes considerados críticos. Se resolvió establecer además un cubrimiento de un 75% en las ramas de decisión. El testing de los componentes no-críticos se realizó conjuntamente con el de los casos de uso en que los mismos están involucrados.

A continuación se presentan los resultados de cubrimiento de los test automáticos sobre componentes realizados con JUnit [5], y el cubrimiento medido con la herramienta DJUnit [6]:



coverage summary

Coverage Report

Packages

packagename	files	lines	%line	indicator	%branch	indicator
org.linkall.federator.logic.applicationmanager	4	993	35%		79%	

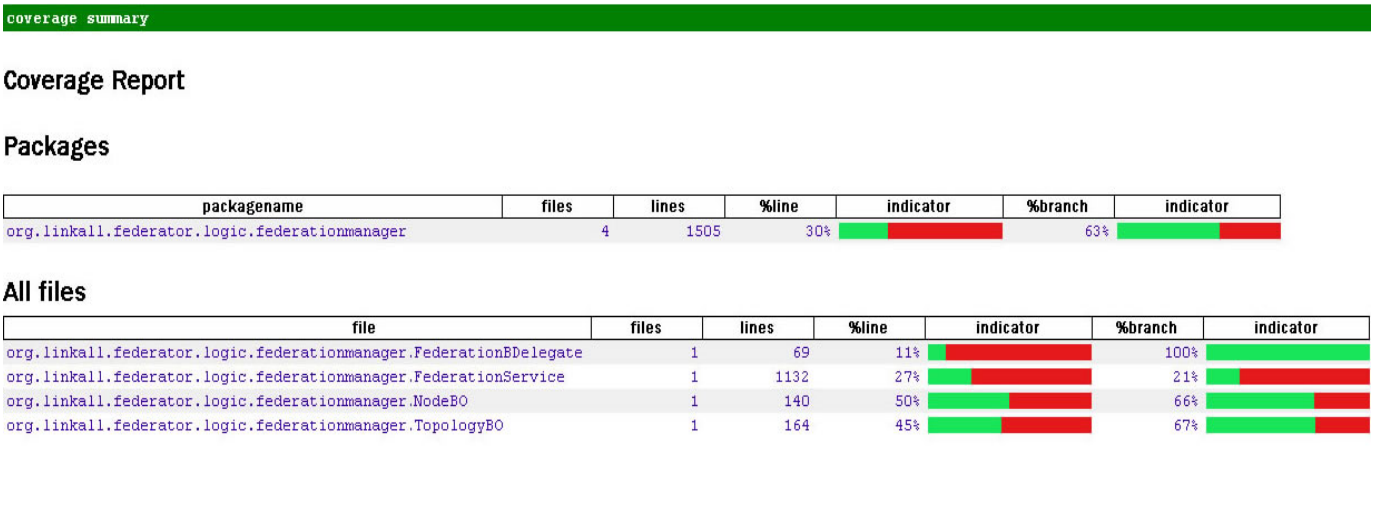
All files

file	files	lines	%line	indicator	%branch	indicator
org.linkall.federator.logic.applicationmanager.ApplicationAppService	1	388	52%		65%	
org.linkall.federator.logic.applicationmanager.ApplicationBDelegate	1	77	10%		96%	
org.linkall.federator.logic.applicationmanager.ApplicationBO	1	188	39%		76%	
org.linkall.federator.logic.applicationmanager.PermissionBO	1	340	18%		80%	
org.linkall.federator.logic.applicationmanager.applicationlocator.AppLocator	1	4	100%		100%	
org.linkall.federator.logic.applicationmanager.applicationlocator.SearchAppStrategy	1	66	62%		66%	

Fig. 3.1 - Resultados del cubrimiento de pruebas automáticas para ApplicationManager

Los resultados presentados en la figura 3.1 muestran que para el componente ApplicationManager los tests automáticos cubrieron un 35 % de líneas de código y un 79 % de las ramas de decisión. Para el caso del FederationManager el cubrimiento de líneas de código fue un 30% y las ramas de

decisión un 63% en los tests automatizados como muestra la figura 3.2. En el caso del componente QueryManager el porcentaje del cubrimiento de las líneas de código fue de un 22% y las decisiones fueron cubiertas en un 84% como muestra la figura 3.3.



coverage summary

Coverage Report

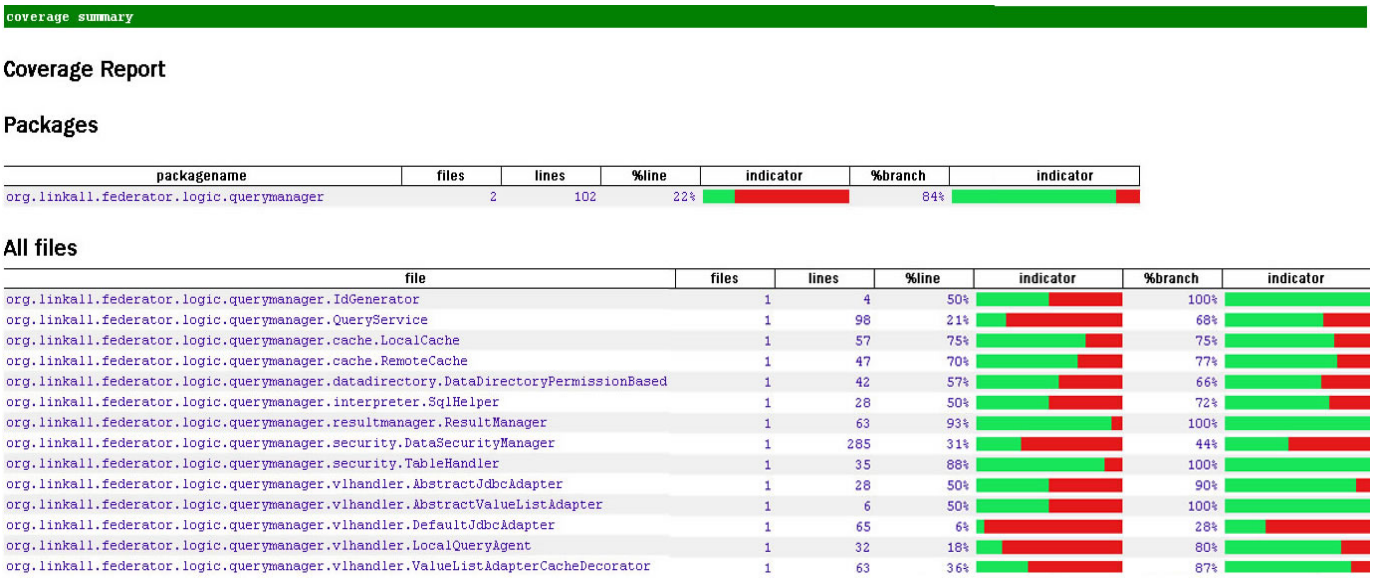
Packages

packagename	files	lines	%line	indicator	%branch	indicator
org.linkall.federator.logic.federationmanager	4	1505	30%		63%	

All files

file	files	lines	%line	indicator	%branch	indicator
org.linkall.federator.logic.federationmanager.FederationBDelegate	1	69	11%		100%	
org.linkall.federator.logic.federationmanager.FederationService	1	1132	27%		21%	
org.linkall.federator.logic.federationmanager.NodeBO	1	140	50%		66%	
org.linkall.federator.logic.federationmanager.TopologyBO	1	164	45%		67%	

Fig. 3.2 - Resultados del cubrimiento de pruebas automáticas para FederationManager



coverage summary

Coverage Report

Packages

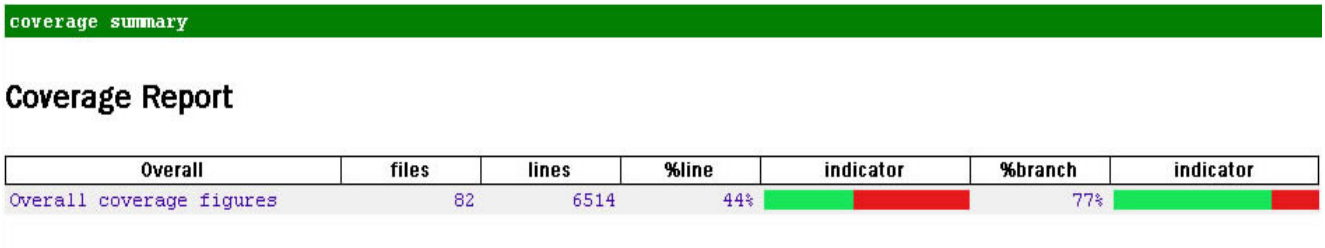
packagename	files	lines	%line	indicator	%branch	indicator
org.linkall.federator.logic.querymanager	2	102	22%		84%	

All files

file	files	lines	%line	indicator	%branch	indicator
org.linkall.federator.logic.querymanager.IdGenerator	1	4	50%		100%	
org.linkall.federator.logic.querymanager.QueryService	1	98	21%		68%	
org.linkall.federator.logic.querymanager.cache.LocalCache	1	57	75%		75%	
org.linkall.federator.logic.querymanager.cache.RemoteCache	1	47	70%		77%	
org.linkall.federator.logic.querymanager.datadirectory.DataDirectoryPermissionBased	1	42	57%		66%	
org.linkall.federator.logic.querymanager.interpreter.SqlHelper	1	28	50%		72%	
org.linkall.federator.logic.querymanager.resultmanager.ResultManager	1	63	93%		100%	
org.linkall.federator.logic.querymanager.security.DataSecurityManager	1	285	31%		44%	
org.linkall.federator.logic.querymanager.security.TableHandler	1	35	88%		100%	
org.linkall.federator.logic.querymanager.vlhandler.AbstractJdbcAdapter	1	28	50%		90%	
org.linkall.federator.logic.querymanager.vlhandler.AbstractValueListAdapter	1	6	50%		100%	
org.linkall.federator.logic.querymanager.vlhandler.DefaultJdbcAdapter	1	65	6%		28%	
org.linkall.federator.logic.querymanager.vlhandler.LocalQueryAgent	1	32	18%		80%	
org.linkall.federator.logic.querymanager.vlhandler.ValueListAdapterCacheDecorator	1	63	36%		87%	

Fig. 3.3 - Resultados del cubrimiento de pruebas automáticas para QueryManager

En promedio los tests automáticos de componente cubrieron un 44 % de las 6514 líneas de código y un 77% de decisión como lo muestra la figura 3.4, con lo que se aproxima en gran medida con el criterio de aceptación establecido.



coverage summary

Coverage Report

Overall	files	lines	%line	indicator	%branch	indicator
Overall coverage figures	82	6514	44%		77%	

Fig. 3.4 - Resultados del cubrimiento promedio de pruebas automáticas para componentes

Es importante aclarar que para todos los componentes, se corrieron casos de pruebas que no fueron automatizados, por lo tanto el cubrimiento real de los componentes es mayor al porcentaje obtenido de los tests automáticos.

3.2. Resultados de los tests

3.2.1. Pruebas de componentes

La tabla 4.1 muestra los resultados obtenidos en los tests de componentes en la primera, tercera y cuarta iteración. Estos resultados muestran que el porcentaje de fallas sobre casos de prueba fue disminuyendo a medida que se avanzaba en las iteraciones. Para el caso de las pruebas de componentes realizadas sobre ApplicationManager y FederationManager en la última iteración se llegó a que la cantidad de fallas sobre la cantidad de casos de pruebas fue en promedio de 4%, menor al 5% requerido.

Cabe aclarar que en la cuarta iteración a los subsistemas mencionados no se le agregaron funcionalidades, solo se hicieron pequeñas modificaciones para integrarlo a la versión 1.8.2 de la plataforma link-all.

Tabla 4.1 Resultados de las pruebas por componentes en las iteraciones 1, 3 y 4 para los componentes ApplicationManager y FederationManager.

Iteración	Componente	Casos de prueba	Fallas	Fallas /Casos
1	ApplicationManager	21	12	57 %
1	FederationManager	6	2	33 %
3	ApplicationManager	42	6	14.28 %
3	FederationManager	35	3	8.57 %
4	ApplicationManager	42	2	4.76
4	FederationManager	35	1	2.86 %

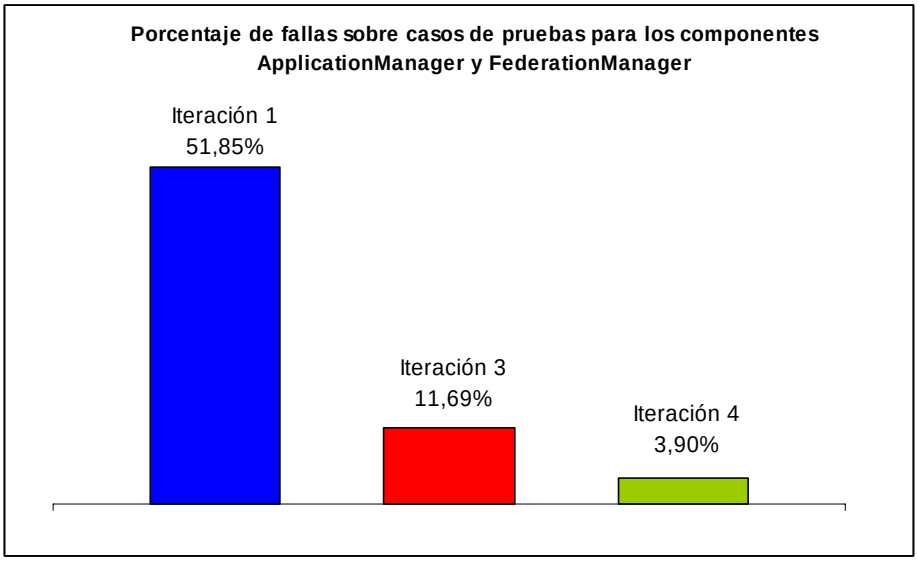


Fig. 3.5 - Muestra la cantidad de fallas sobre casos de pruebas para los componentes ApplicationManager y FederationManager en las iteraciones 1, 3 y 4.

3.2.2. Pruebas de Casos de uso

Los resultados de los tests de casos de uso también muestran que cada iteración se fue reduciendo el porcentaje de fallas sobre cantidad de pruebas realizadas.

En la primera iteración se implementaron las funcionalidades rutear aplicación, obtener aplicaciones de usuario, federar y desfederar aplicación. De un total de 20 casos de pruebas para los caso de uso fueron encontrados 10 errores los cuales se reportaron y corrigieron antes de pasar a la segunda iteración.

En la segunda iteración se agregaron las funcionalidades actualizar mapa de la federación, federar y desfederar servidor, además se integró con la plataforma Link-All 1.6. Se ejecutaron en total 78 casos de prueba para estas funcionalidades y para las implementadas en la iteración anterior de los cuales se encontraron 27 fallas.

Las demás funcionalidades del federador de aplicaciones fueron implementadas en la tercera iteración y se integró con la versión 1.8 de la plataforma Link-All. Los casos de pruebas realizados para las nuevas funcionalidades no fueron tan exhaustivos debido a que estos casos de uso no requerían nivel de calidad cinco. Varios de los tests realizados para las funcionalidades implementadas en iteraciones anteriores tuvieron que ser modificados debido al cambio de versión de la plataforma, ya que se modificaron tablas y se debió modificar el juego de datos de entrada.

La cantidad de tests de casos de uso realizados en la tercera iteración fue 106 y éstos detectaron un total de 13 fallas en el sistema.

La siguiente tabla muestra la cantidad de tests funcionales realizados en cada iteración, la cantidad de fallas encontradas y los porcentajes de fallas por casos de prueba.

Resultados de las pruebas de todos los casos de uso por iteración

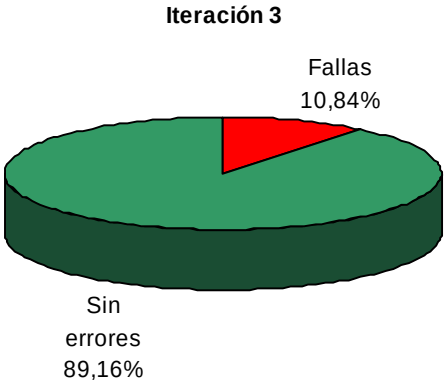
Iteración	Funcionalidades	Total Tests	Total Fallas	Porcentaje
Primera	4	20	10	50 %
Segunda	7	78	27	35 %
Tercera	14	106	13	12 %
Cuarta	19	121	14	11.5 %

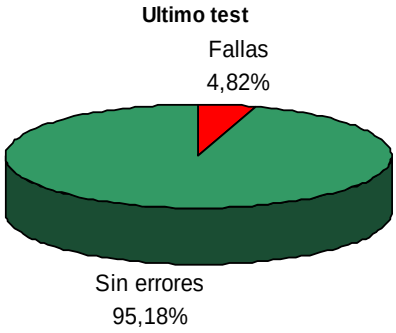
Resultados de las pruebas de casos de uso que requieren nivel de calidad 5, iteración 3

Caso de uso	Casos de prueba	Fallas	Fallas /Casos
Rutear aplicación	20	4	20 %
Obtener aplicaciones del usuario	6	0	0 %
Federar Aplicación	16	0	0 %
Desfederar Aplicación	16	1	6.25%
Federar Nodo	11	2	18 %
Desfederar Nodo	8	1	12.5 %
Actualizar Topología	6	1	16.6 %
Total de estos casos	**83**	**9**	**10.84 %**

Resultados finales de las pruebas de casos de uso que requieren nivel de calidad 5

Caso de uso	Casos de prueba	Fallas	Fallas /Casos
Rutear aplicación	20	2	10 %
Obtener aplicaciones del usuario	6	0	0 %
Federar Aplicación	16	0	0 %
Desfederar Aplicación	16	1	0%
Federar Nodo	11	1	9 %
Desfederar Nodo	8	0	0 %
Actualizar Topología	6	0	0 %
Total de estos casos	**83**	**4**	**4.82 %**





Las gráficas anteriores muestran la cantidad de fallas que hubieron en los pruebas de los casos de uso que requerían nivel de calidad cinco, en la tercera iteración y en los últimos test realizados para esas funcionalidades.

3.3. Conclusiones

En base a los resultados obtenidos se puede concluir que el producto cumple con los requerimientos planteados y posee las características de calidad exigidas.

4. Referencias

[1] Documento de Requerimientos y Casos de Uso, Fabricio Alvarez, Rodolfo Amador, Proyecto Federador Link-ALL 2004.

[2] Documentación Técnica, Fabricio Alvarez, Rodolfo Amador, Proyecto Federador Link-ALL 2004.

[3] Documento de Estado del Arte y Tecnologías, Fabricio Alvarez, Rodolfo Amador, Proyecto Federador Link-ALL 2004.

[4] Pablo Garbuzi, Federico Piedrabuena, Laura González, Link-ALL System Architecture Document, Versión 1.8, Instituto de Computación, Facultad de Ingeniería, Universidad de la República. Febrero 2005

[5] Vincent Massol, Ted Husted, JUnit in Action, Manning Publications Co, 2004

[6] DJUNIT http://works.dgic.co.jp/djwiki/Viewpage.do?pid=@646A556E6974, 18/11/2004

[7] ECLIPSE http://www.eclipse.org/, 30/03/2005

[8] Bugzilla http://www.bugzilla.org/, 10/12/2004

[9] BugCollector http://www.bugcollector.com/, 13/12/2004

PROYECTO DE GRADO
FEDERADOR LINK-ALL

Manual de Usuario e Instalación

Mayo 2005

Instituto de computación - Facultad de Ingeniería
Universidad de la República

Estudiantes: Fabricio Alvarez
Rodolfo Amador

Tutores: Federico Piedrabuena
Raúl Ruggia

Tabla de contenido

1. Introducción

Este manual proporciona una guía con la información necesaria para instalar y configurar el Federador Link-ALL.

1.1. Acerca de la versión

Esta versión del Federador y sus utilitarios brindan mecanismos de federación a la plataforma Link-ALL [1].

1.2. Limitaciones

Esta liberación no cubre un escenario con más de un Servidor de Soporte.

1.3. Archivos y descripción

Binarios:

appdbFederator.jar – Federador Link-ALL, versión 1.0.5.
federator.war – Interfaz Web del Federador Link-ALL.
fedbrowser1.war – FedBrowser, navegador de la federación.
linkall.war – Interfaz Web de Link-ALL.
wsfederator.war – Web Service del Federador (Axis).

Descriptores de instalación:

linkall-global-ds.xml – Descriptor de instalación de la base de datos linkall_global.
linkall-admin-ds.xml – Descriptor de instalación de la base de datos linkall_admin.

Docs:

javadocs_1.0.5.zip – Javadocs.

2. Instalación

2.1. Requerimientos del Sistema

El Federador Link-ALL ha sido testeado con la configuración especificada a continuación, pero puede ser ejecutado en versiones anteriores o posteriores del software mencionado.

2.1.1. Requerimientos de Software

Para ejecutar el Federador Link-ALL es necesario por lo menos:

- JDK 1.4.2
- JBoss 3.2.3
- PostgreSQL 7.3.4

2.2. Instalación

2.2.1. Instalar las bases de datos Globales y Administrativas

En PostgreSQL:

Usuario y Grupo

- Crear un grupo de usuarios de nombre linkall.
- Crear un usuario llamado linkall, con password linkall.
- Agregar el usuario linkall al grupo linkall.

Base de datos Administrativa

- Crear una base de datos denominada linkall_admin.
- Ejecutar los scripts incluidos en el archivo linkall_admin.sql sobre la base de datos linkall_admin. Esto creará la estructura de la base de datos e insertará la información requerida.
- Verificar que el grupo linkall tiene privilegios sobre esta base de datos.

Base de datos Global

- Crear una base de datos denominada linkall_global.
- Ejecutar los scripts incluidos en el archivo linkall_global.sql sobre la base de datos linkall_global. Esto creará la estructura de la base de datos e insertará la información requerida.
- Verificar que el grupo linkall tiene privilegios sobre esta base de datos.

2.2.2. Instalar el Federador Link-ALL y los utilitarios

Copiar los archivos binarios y descriptores de instalación (ver sección anterior Binarios) en la carpeta JBOSS_HOME\server\default\deploy\ donde JBOSS_HOME es la carpeta de instalación de JBoss.

3. Configuración

En las secciones siguientes se presentan los detalles de configuración del Federador Link-ALL y el utilitario FedBrowser.

3.1. Configuración del Federador Link-ALL

La configuración del Federador Link-ALL que se almacena en un archivo se detallará en la sección siguiente.

3.2. Archivo de Configuración

3.2.1. Ubicación del archivo de configuración

El archivo de configuración se denomina ***fedconfig.properties*** y se encuentra en la carpeta ***configuration*** dentro del JAR ***appdbFederator.jar*** como se muestra a continuación:

appdbFederator.jar\org\linkall\federator\configuration\fedconfig.properties

3.2.2. Secciones del archivo de configuración del Federador

El archivo se encuentra dividido por secciones, a continuación se detalla cada una de las secciones con el significado de los parámetros del archivo:

3.2.2.1. Sección Actualización de Datos de la Federación

En esta sección se detalla información sobre la estrategia de actualización de la información de la Federación.

FED.update_strategy: Nombre de la estrategia de actualización (por defecto fuerza bruta).

3.2.2.2. Sección Federación de Aplicaciones

En esta sección se detalla información sobre la federación de aplicaciones.

FED.federate_after_install: Indica si la aplicación se federa inmediatamente después de la instalación.

3.2.2.3. Sección Datos del Servidor de Soporte

En esta sección se debe detallar el nombre del Host y el Puerto donde se encuentra instalado el Servidor de Soporte.

FED.host_support_server: Nombre del Host o dirección IP del Servidor de Soporte.
FED.port_support_server: Puerto de acceso a los Web Services del Servidor de Soporte.

3.2.2.4. Sección Punto de acceso al Web Service del Federador

En esta sección se debe detallar el protocolo de comunicación utilizado para acceder a los Web Services del Federador Link-ALL además del camino de acceso a los mismos.

FED.url_protocol: Protocolo de comunicación.
FED.url_service: Camino de acceso a los Web Services del Federador Link-ALL.

3.2.2.5. Sección Controlador del planificador de trabajos

En esta sección se debe detallar como controlar el framework encargado de la planificación de trabajos.

FED.path_scheduler: Camino relativo del Servlet controlador de Quartz.

3.2.2.6. Sección Base de Datos Global

En esta sección se detalla información sobre la base de datos global:

DBGLOBAL.schema_name: Nombre del esquema de la base de datos Global.

3.2.3. Ejemplo de archivo de configuración del Federador Link-ALL

A continuación se presenta el archivo de configuración por defecto que presenta el Federador:

```
#####################################################################
# FEDERADOR                                                         #
#####################################################################

#Estrategia de actualizacion de la Federacion
FED.update_strategy = upd_bruteForce

#Indica si la federación de una aplicación se realiza inmediato a la instalación
FED.federate_after_install = yes

#Datos del Servidor de Soporte
FED.host_support_server =
FED.port_support_server =

#Punto de acceso al WebService del Federador
FED.url_protocol = http
FED.url_service = /wsfederator/services/webservicesfederator

#Path relativo del servlet controlador de Quartz
FED.path_scheduler = /federator/QuartzServletController

#Nombre del esquema de la base de datos Global
DBGLOBAL.schema_name = public
```

3.3. Configuración de FedBrowser

La configuración del navegador de la federación consiste en indicar la forma de acceder al Federador Link-ALL ya que ésta herramienta debe consumir el Web Service que le proporciona el XML del Mapa de la Federación. La configuración del componente se almacena en un archivo de configuración como se detalla en la sección siguiente.

3.4. Archivo de Configuración

3.4.1. Ubicación del archivo de configuración

La configuración de acceso al Federador Link-ALL se encuentra en el archivo de configuración denominado ***fedbrowserconfig.properties*** y se encuentra en la carpeta ***configuration*** dentro del WAR ***fedbrowser1.war*** como se muestra a continuación:

fedbrowser1.war\WEB-INF\classes\configuration\fedbrowserconfig.properties

3.4.2. Secciones del archivo de configuración de FedBrowser

El archivo se encuentra dividido por secciones, a continuación se detalla cada una de las secciones con el significado de los parámetros del archivo:

3.4.2.1. Sección Datos del Federador

En esta sección se debe detallar el nombre del Host y el Puerto donde se encuentra instalado el Federador Link-ALL.

FEDBROWSER.host_federador: Nombre del Host o dirección IP del servidor donde se encuentra instalado el Federador Link-ALL.

FEDBROWSER.port_federador: Puerto de acceso a los Web Services del Federador Link-ALL (corresponde al puerto del servidor Web donde está instalado el Federador).

3.4.2.2. Sección Punto de acceso al Web Service del Federador

En esta sección se debe detallar el protocolo de comunicación utilizado para acceder a los Web Services del Federador Link-ALL además del camino de acceso a los mismos.

FEDBROWSER.url_protocol: Protocolo de comunicación.
FEDBROWSER.url_federator_web_service: Camino de acceso a los Web Services del Federador Link-ALL.

3.4.3. Ejemplo de archivo de configuración de FedBrowser

A continuación se presenta el archivo de configuración por defecto que presenta FedBrowser:

```
######################################################################
# FEDBROWSER                                                         #
######################################################################

#Datos del Federador
FEDBROWSER.host_federador = 127.0.0.1
FEDBROWSER.port_federador = 8080

#Punto de acceso al WebService del Federador
FEDBROWSER.url_protocol = http
FEDBROWSER.url_federator_web_service = /wsfederator/services/webservicesfederator
```

4. Manual de Usuario

4.1. Ejecutar la aplicación

Levantar el servidor JBoss y abrir un navegador en la dirección http://localhost:8080/linkall. En caso que el servidor sea remoto se debe reemplazar localhost por la URL del servidor, de la misma forma se debe cambiar 8080 si se configuró otro puerto para el servidor. Luego de ingresar esta dirección se presenta la página inicial de la plataforma Link-ALL como se ve en la figura 4.1.

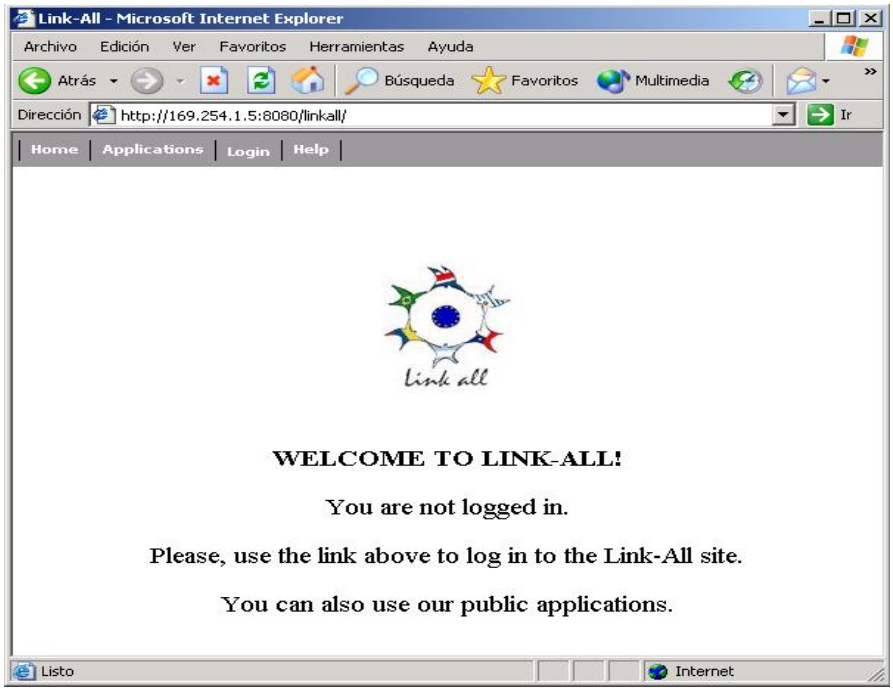


Fig. 4.1 - Página inicial de la plataforma Link-ALL.

4.2. Ingresar al Federador

Para poder acceder a la interfaz Web del Federador es necesario ingresar a la plataforma Link-ALL con el login del usuario administrador del servidor. Para ello seleccionar “Login” en el menú superior, a continuación se presenta una página que solicita usuario, contraseña y comunidad del usuario como muestra la figura 4.2.

Fig. 4.2 - Acceso a la plataforma Link –ALL.

Una vez ingresado al sistema Link-ALL, se presenta una página de bienvenida y en el menú superior aparecen las siguientes opciones:

Seleccionar “Applications” para obtener la lista de aplicaciones disponibles del usuario, entre estas aplicaciones el administrador encuentra al Federador Link-ALL.

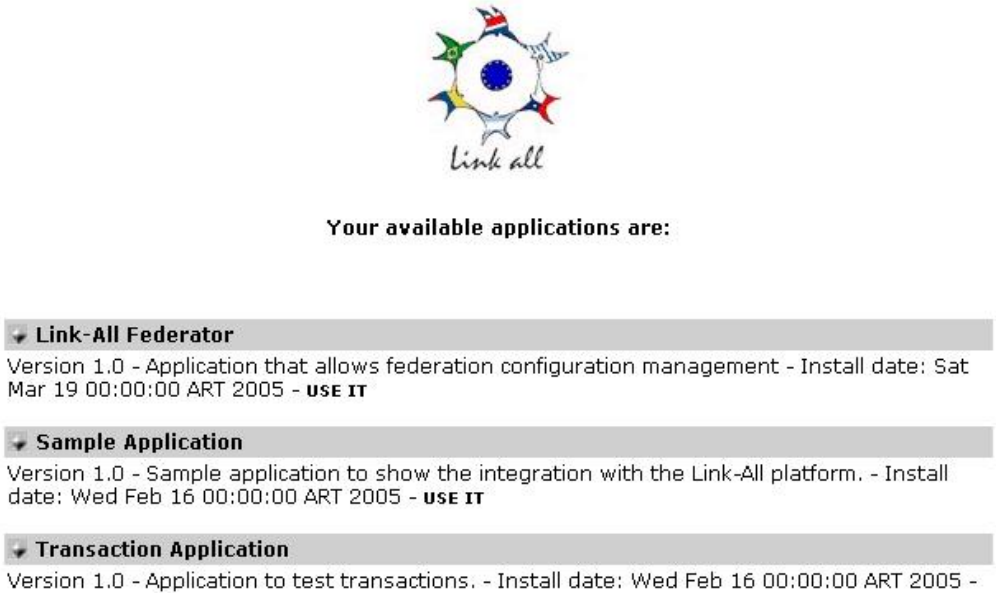


Fig. 4.3 - Lista de aplicaciones disponibles para el administrador.

Seleccionar la aplicación Link-ALL Federator. A continuación se desplegará una presentación y un link que permitirá ingresar a la página inicial de configuración del Federador como se muestra en la siguiente figura:

Fig. 4.4 - Página inicial del Federador Link-ALL.

Al ingresar se desplegará un menú superior y un menú lateral.

- **Home**: Retorna a la página inicial – configuración del federador.
- **Federation**: Brinda opciones para la federación por ejemplo visualizar mapa, actualizar mapa.
- **Node**: Federar y desfederar servidor.
- **Community**: Listar, crear, modificar y eliminar comunidades del nodo.
- **Application**: Listar, federar y desfederar aplicaciones. Solicitar, otorgar y denegar permisos sobre aplicaciones.
- **Data**: Permite solicitar, otorgar, y denegar permisos sobre datos. Presenta un ejemplo de las funcionalidades del federador sobre los datos.
- **Contact**: Información de contacto.

En el menú lateral se presentan las mismas opciones que en el menú superior, a las que se le agrega la opción de Configuración del Federador.

4.2.1. Establecer el servidor de soporte de un nodo

Al iniciar el Federador Link-ALL por primera vez no está configurado el servidor de soporte. Si se determina que el servidor va a ser un nodo común se deberá primeramente establecer el servidor de soporte ingresando sus datos.

Para esto se debe ingresar a la opción de *Federator Configuration* desde la página inicial. Luego se debe seleccionar la opción de *Support Server*.

Se presentará una página como la que se ve en la figura 4.5, donde se deben ingresar los datos requeridos para configurar el servidor de soporte.

Una vez ingresada la información el nodo puede ser federado.

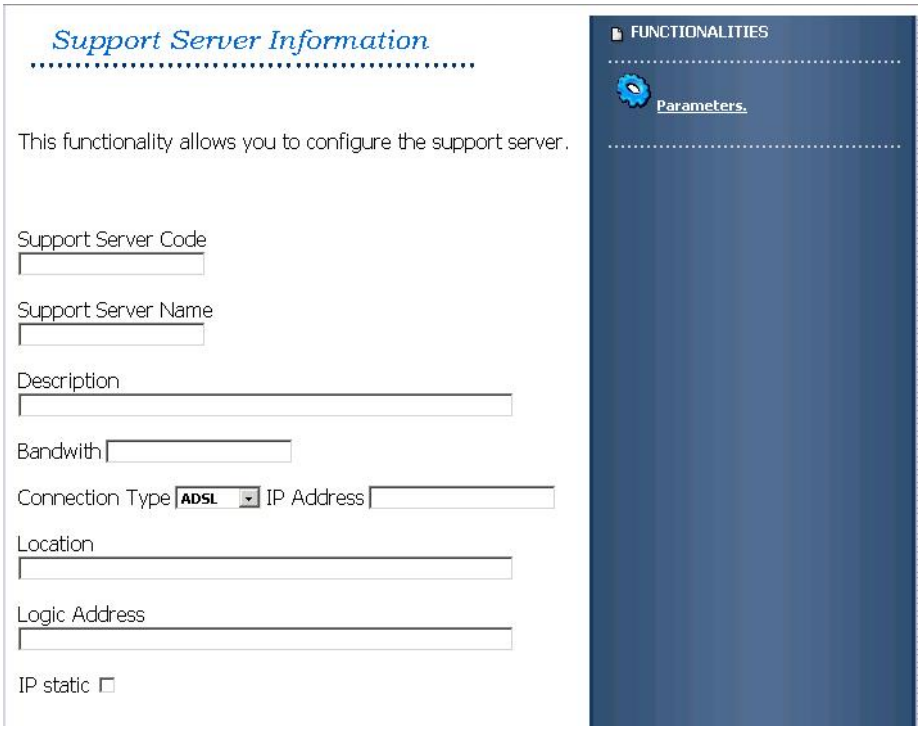


Fig. 4.5 - Datos a ingresar para configurar el servidor de soporte.

4.3. Federación

4.3.1. Actualizar Mapa de la Federación

Esta opción solo actualizará el mapa de la federación si el servidor no es un nodo de soporte.

Seleccionar tipo de actualización:

- Manual
- Automática

Para la actualización manual presionar el botón que se muestra en la siguiente figura:

En caso de configurar la actualización automática, indicar fecha, hora para la actualización, frecuencia con la que se repite y finalmente registrar la configuración.

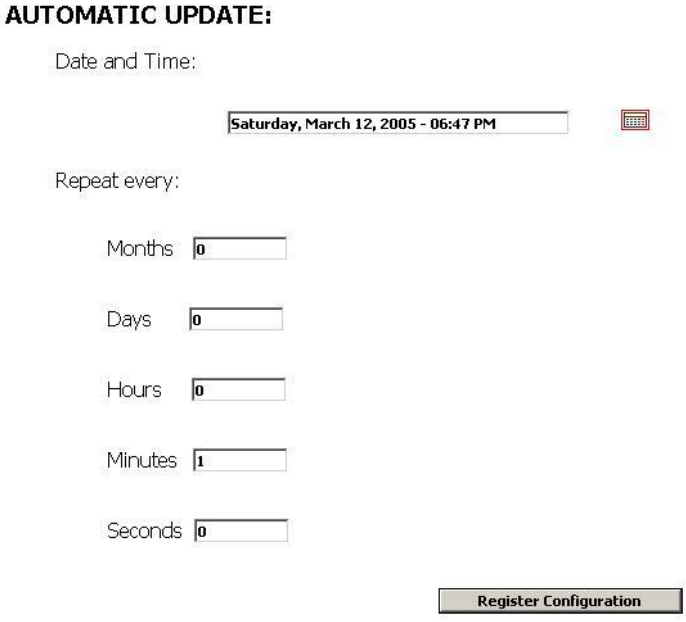


Fig. 4.6 - Registrar configuración automática

4.3.2. Visualizar Mapa de la federación

Para visualizar el mapa de la federación seleccionar la opción *Federation map* del menú *Federation*. Esto desplegará el mapa de la federación existente en el nodo local. Se visualizará un mapa similar al de la siguiente figura:

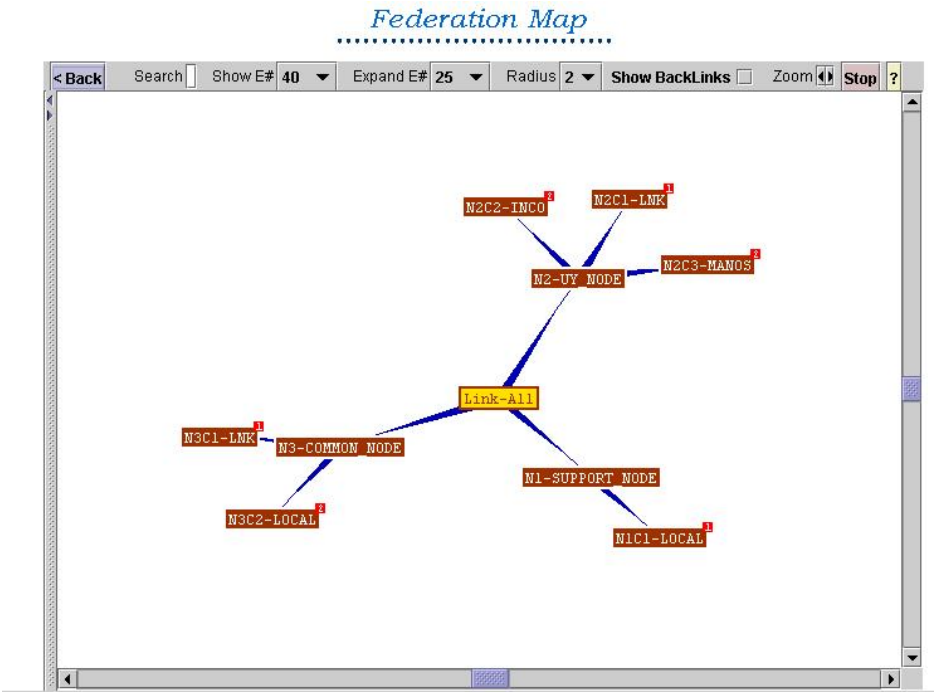


Fig. 4.8 - Mapa de la Federación

En el mapa se ve que la red está conformada por tres nodos cuyos códigos son UY_NODE, COMMON_NODE y SUPPORT_NODE. También se ven las comunidades que tiene federada cada nodo y el número de aplicaciones que ésta federa. Por ejemplo en el servidor UY_NODE las comunidades INCO, LNK y MANOS tienen aplicaciones federadas. El número que aparece a la derecha del código de la comunidad es la cantidad de aplicaciones que ésta federa.

Para ver las aplicaciones que federa una comunidad presionar con el botón derecho del ratón y seleccionar *Expand Node*.

Fig. 4.9 - Ver aplicaciones federaras por una comunidad

Para obtener los detalles de cualquier elemento en el mapa hacer doble clic sobre el elemento.

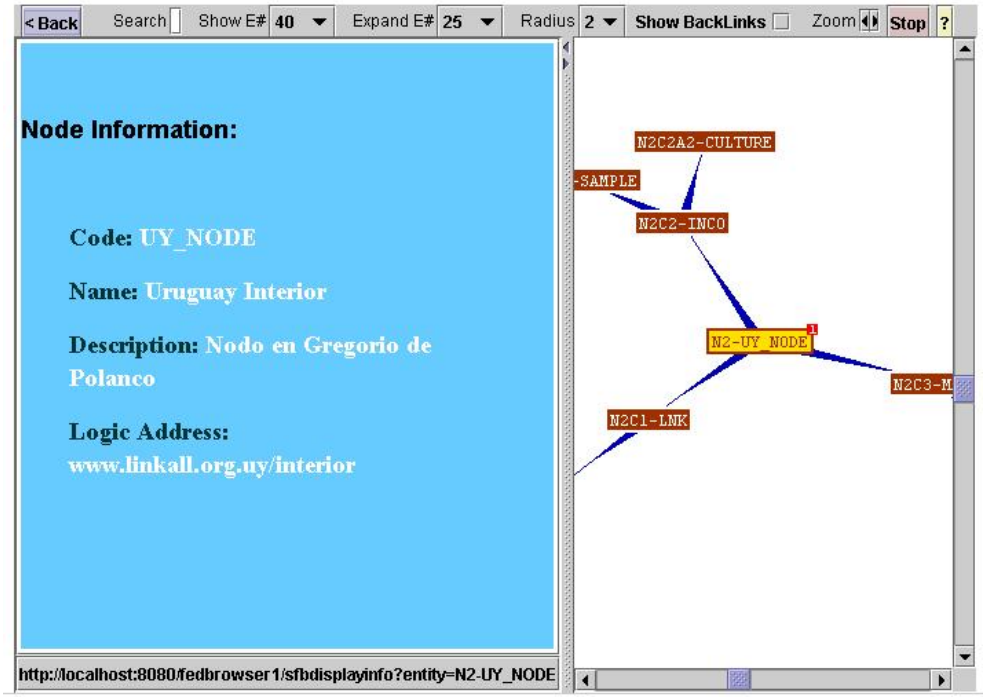


Fig. 4.10 - Detalles de un elemento en el mapa de la federación

4.4. Nodo

4.4.1. Federar Nodo

4.4.1.1. Federar Servidor de Soporte

Ingresar a la opción *Federate Local Node* y completar la información correspondiente. Se debe seleccionar el checkbox *Support Server* y no es necesario configurar previamente un servidor de soporte.

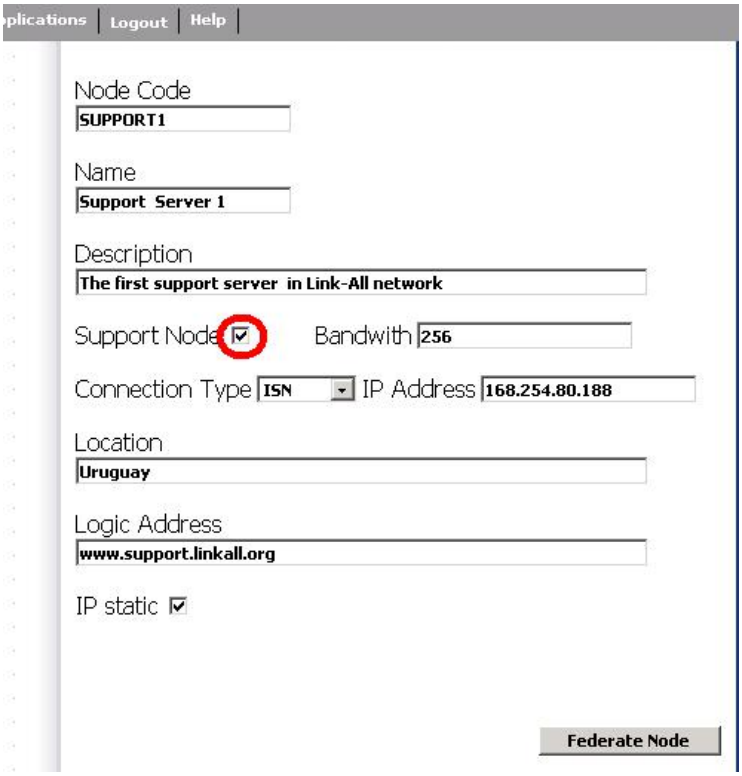


Fig. 4.11 - Federar Servidor

4.4.1.2. Federar servidor común

Ingresar los datos de la misma forma que para federar un nodo de soporte pero no seleccionar el checkbox *Support Node*. Para federar un servidor común debe estar configurado el servidor de soporte ya que es contra éste que el nodo se federa.

4.4.2. Desfederar Nodo

Seleccionar *Defederate Node* del menú Node. El sistema solicitará la confirmación de la operación.

4.5. Comunidades

4.5.1. Listar comunidades

Seleccionar la opción *Communities* del menú superior o lateral, a continuación seleccionar *Communities Administration*. Se desplegará la lista de comunidades existentes en el servidor. Esta página permite crear, eliminar y modificar comunidades.

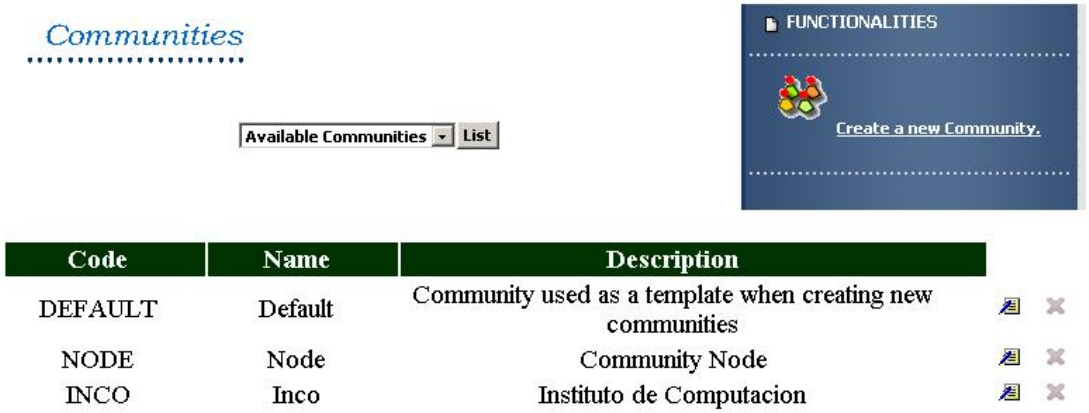


Fig. 4.12 - Página para la administración de comunidades

4.5.2. Crear Comunidades

Seleccionar *Create new Community* del menú lateral de la página anterior. Se despliega una página que solicita el ingreso del código, nombre y descripción de la comunidad a crear. Ingresar los datos solicitados y confirmar.

4.5.3. Modificar datos de Comunidades

Seleccionar el boton *update* a la derecha de la comunidad que se desea modificar en la lista de comunidades. Se despliega una página donde se podrá modificar el nombre y la descripción de la comunidad. Luego de modificados los datos se deberá confirmar.

4.5.4. Eliminar Comunidades

Seleccionar el boton *delete* a la derecha de la comunidad que se desea eliminar en la página donde se listan las comunidades. Se presenta un mensaje de confirmación para la eliminación de la comunidad.

4.6. Aplicaciones

4.6.1. Federar aplicaciones

Seleccionar la opción *Applications to Federate* del menú *Application*. A continuación se presenta una lista de las aplicaciones del servidor que no están federadas, como lo muestra la figura 4.13. Para federar una aplicación, pulsar el botón a la derecha de los datos de la aplicación a federar.

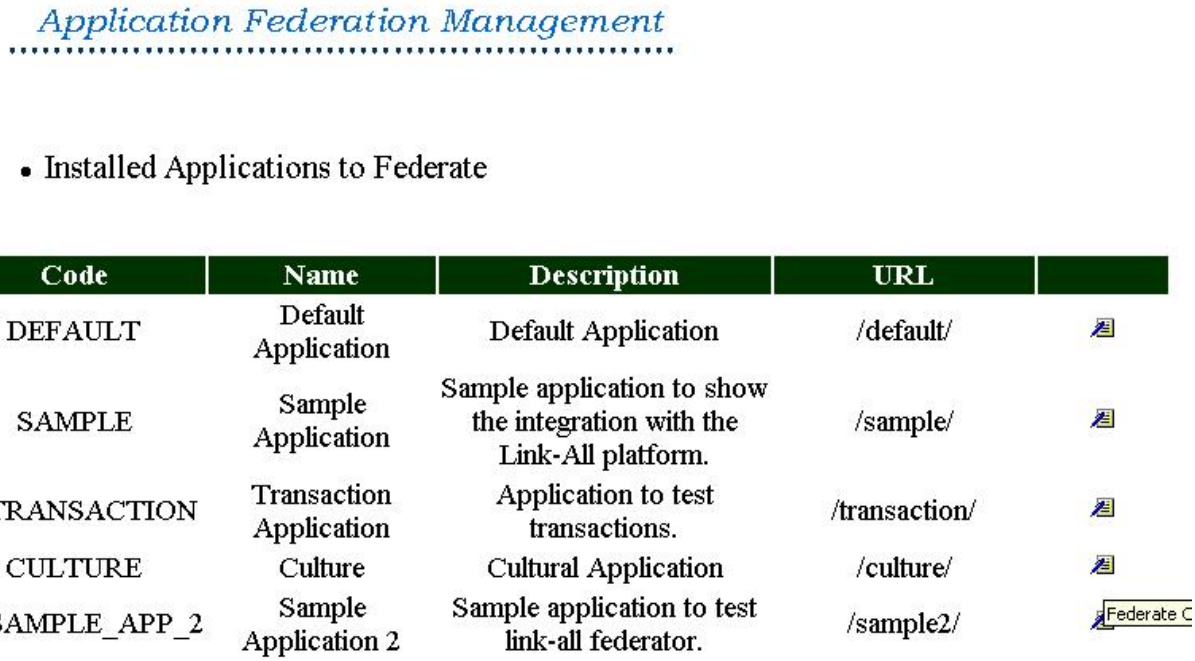


Fig. 4.13 - Lista de aplicaciones del servidor que pueden ser federadas

4.6.2. Desfederar aplicación

El mecanismo es similar al de federar aplicación. Seleccionar la opción *Applications to Defederate* del menú *Application*, a continuación se presenta una lista de las aplicaciones del servidor que están federadas. Presionar el botón *desfederar* a la derecha de los datos de la aplicación que se desea desfederar.

4.7. Datos

Las funcionalidades que se describen a continuación permiten testear las operaciones del Federador de Datos Link-ALL. Estas pruebas se realizan sobre la tabla lnk_actor.

4.7.1. Consulta de selección de datos

En la figura 4.14 se muestra la página inicial del Federador de Datos. Al seleccionar el link *Actors* en el menú lateral o el link *Actor of Link-ALL* en la página, se realizará la consulta que permitirá obtener la información de todos los actores de la red Link-ALL (*select * from lnk_actor*).

Fig. 4.14 – Página inicial del Federador de Datos

Si el nodo está federado el resultado de la consulta involucra datos de otros servidores y se toman en cuenta los permisos que el usuario tenga para la tabla *lnk_actor*. Luego de realizada la consulta se presentará una página como la de la figura 4.15.

Fig. 4.15 – Resultado de la consulta sobre la tabla lnk_actor

La paginación utilizada es de cinco resultados por página con un máximo de 100 resultados. Los botones y permiten la navegación hacia la siguiente y anterior página de resultados respectivamente. A su vez los botones y permiten ir a la primer y última página de resultados. La consulta realizada integra los resultados locales y remotos de manera transparente.

4.7.2. Consulta de inserción, actualización y eliminación

Para realizar una consulta de inserción de datos en la tabla de actores, seleccionar el link *Add new actor* en la página de resultados. A continuación se solicitarán los datos para el nuevo registro: el nombre del actor, tipo, descripción y comunidad. Estos son los mismos datos que para la consulta de actualización. Presionar el botón *Create Actor* y se insertarán los datos ingresados.

Fig. 4.16 Actualización de datos en la tabla lnk_actor

Para actualizar los datos de un registro se debe seleccionar el botón a la derecha de los datos del registro que se desea modificar en la página de resultados. A continuación se presenta la página mostrada en la figura 4.16.

Luego de ingresar los datos para confirmar la actualización se debe presionar el botón *Update actor*.

Para eliminar un registro presionar el botón a la derecha del mismo. Se desplegará un mensaje solicitando confirmar la eliminación de los datos.

La inserción, actualización y eliminación se realizan localmente.

4.8. Seguridad

4.8.1. Solicitar permiso de acceso a un recurso

El mecanismo de solicitar permisos es similar tanto para aplicaciones como para tablas.

Ingresar al menú *Federation*, *Application/Data* y por último seleccionar la opción *Solicit Access to application/table*. A continuación se despliega la página presentada en la figura 4.17 para las aplicaciones o 4.18 para las tablas. En ésta se solicita el código del servidor, comunidad, aplicación/tabla sobre la que se solicita permiso de acceso (Remote Data) y la comunidad y grupo local para el que se solicita el permiso (Local Data).

Ingresar los datos y confirmar la operación.

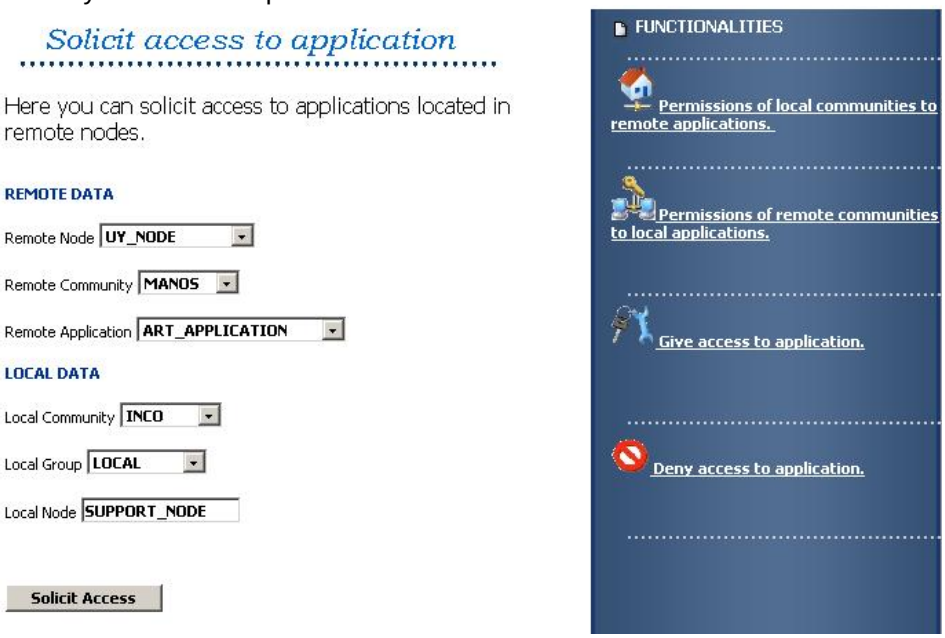


Fig. 4.17 - Solicitar permisos de acceso a una aplicación

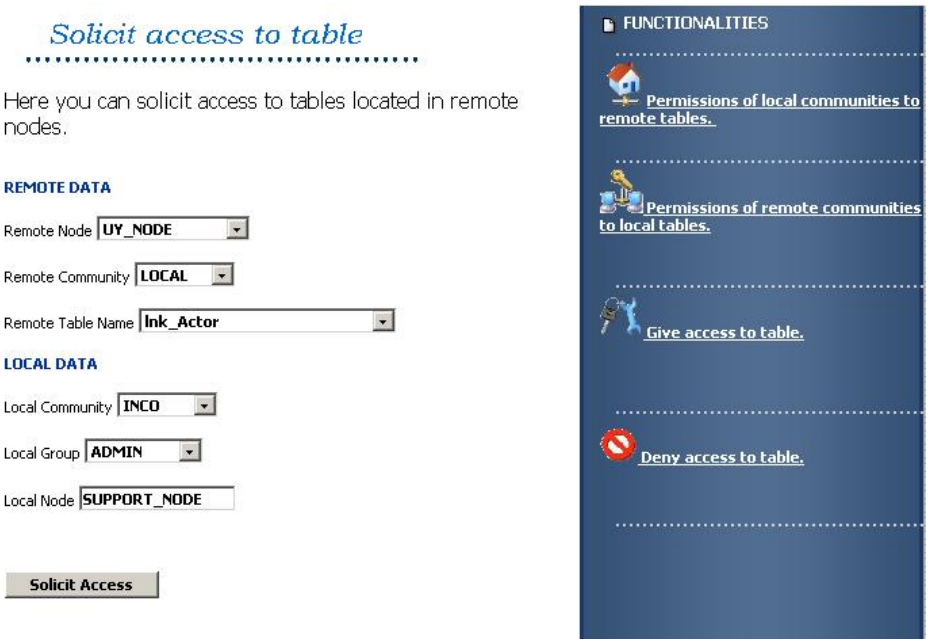


Fig. 4.18 - Solicitar permisos de acceso a una tabla

4.8.2. Otorgar permiso de acceso a un recurso

Seleccionar la opción Give *access to application* del menú *Application* o desde el menú *Federation*. A continuación se despliega la página donde se solicita los datos del servidor, comunidad, y grupo remoto al que se le otorgan permisos (Remote Data). También se solicita los datos de la comunidad y aplicación local (Local Data) que otorga dichos permisos.

Se podrá aprobar solicitudes de permisos pendientes. Para esto sólo se debe ingresar el identificador de la solicitud (Ver listar Permisos).

4.8.3. Denegar acceso a un recurso

Para revocar permisos se solicita ingresar el identificador del permiso a denegar (Ver Listar permisos) y luego confirmar la operación.

4.8.4. Listar Permisos

Se pueden realizar consultas de los permisos que tienen las comunidades remotas sobre los recursos locales (seleccionar *Permissions of remote communities to local applications*) así como también los permisos que tienen las comunidades locales sobre los recursos remotos (seleccionar *Permissions of local communities to remote applications)*.

En ambas consultas se retornan tanto las solicitudes pendientes como los permisos activos.

5. Referencias

[1] Link-ALL web site http://www.Link-ALL.org, 15/03/2005

PROYECTO DE GRADO
FEDERADOR LINK-ALL

Manual Técnico

Mayo 2005

Estudiantes: Fabricio Alvarez
Rodolfo Amador

Tutores: Federico Piedrabuena
Raúl Ruggia

Tabla de contenido

1. Introducción

En las secciones siguientes se presenta información acerca de cómo utilizar la API del Federador Link-ALL. Se presentan también posibles puntos de extensión, conjuntamente con detalles técnicos de cómo implementar tales extensiones.

2. Federador de Aplicaciones

Esta sección presenta como utilizar las funcionalidades del Federador de Aplicaciones.

2.1. Invocar al federador de aplicaciones

Para invocar al Federador de Aplicaciones es necesario crear una instancia del ApplicationBDelegate [1] y luego realizar las llamadas correspondientes. El BusinessDelegate será el encargado de realizar las llamadas RMI a la fachada del Federador de Aplicaciones.

Ejemplo:

```
ApplicationBDelegate appFederator = new ApplicationBDelegate();
```

Nota:

No es recomendable pero sería posible invocar directamente a la fachada del Federador de Aplicaciones vía RMI y no a través del ApplicationBDelegate (éste es un punto de acceso centralizado que contribuye a mejorar la performance debido al uso de caché de referencias). Dicha invocación podría ser así:

```
Hashtable hashProps = new HashTable();
InitialContext ic;

hashProps.put(InitialContext.INITIAL_CONTEXT_FACTORY,
"org.jnp.interfaces.NamingContextFactory");
hashProps.put(InitialContext.PROVIDER_URL, "jnp://127.0.0.1:1099");

 ic = new InitialContext(hashProps);

ApplicationFacadeHome applicationFacadeHome = (ApplicationFacadeHome)
ic.lookup("ApplicationFacade");
ApplicationFacade applicationFacade = (ApplicationFacade) applicationFacadeHome.create();
```

2.2. Federar una aplicación

Suponiendo creados el value object [1] con los datos de la aplicación a federar y el value object con los datos del contexto de la llamada, la invocación sería así:

```
// Invoca la federación de aplicaciones a través del BusinessDelegate
ApplicationBDelegate appBD = new ApplicationBDelegate();
appBD.federateApplication(application, contextVO);
```

2.3. Desfederar una aplicación

Suponiendo creados el value object con los datos de la aplicación a desfederar y el value object con los datos del contexto de la llamada, la invocación sería así:

```
// Invoca la federación de aplicaciones a través del BusinessDelegate
ApplicationBDelegate appBD = new ApplicationBDelegate();
appBD.defederateApplication(application, contextVO);
```

2.4. Retornar aplicaciones del login

Suponiendo creados el value object con los datos del Login y el value object con los datos del contexto de la llamada la invocación sería así:

```
ApplicationBDelegate appBD = new ApplicationBDelegate();
return appBD.getLoginApplications(loginVO, contextVO);
```

2.5. Rutear una aplicación

Suponiendo creados el value object con el código de la aplicación solicitada y el value object con los datos del contexto de la llamada la invocación sería así:

```
ApplicationBDelegate appBD = new ApplicationBDelegate();
return appBD.getApplicationLocation(applicationVO, contextVO);
```

3. Federador de Datos

Esta sección presenta como utilizar las funcionalidades del Federador de Datos.

3.1. Invocar al federador de Datos

Para invocar al Federador de Datos es necesario crear una instancia del QueryBDelegate [1] y luego realizar las llamadas correspondientes. El BusinessDelegate será el encargado de realizar las llamadas RMI a la fachada del Federador de Datos.

Ejemplo:

```
QueryBDelegate dbFederator = new QueryBDelegate();
```

Nota:

No es recomendable pero sería posible invocar directamente a la fachada del Federador de Datos vía RMI y no a través del QueryBDelegate (éste es un punto de acceso centralizado que contribuye a mejorar la performance debido al uso de caché de referencias). Dicha invocación podría ser así:

```
Hashtable hashProps = new HashTable();
InitialContext ic;

hashProps.put(InitialContext.INITIAL_CONTEXT_FACTORY,
"org.jnp.interfaces.NamingContextFactory");
hashProps.put(InitialContext.PROVIDER_URL, "jnp://127.0.0.1:1099");

 ic = new InitialContext(hashProps);

QueryFacadeHome queryFacadeHome = (QueryFacadeHome) ic.lookup("QueryFacade");
QueryFacade queryFacade = (QueryFacade) queryFacadeHome.create();
```

3.2. Ejecutar una consulta

Suponiendo creada la consulta SQL a ejecutar, el value object con la información de control y el value object con los datos del contexto de la llamada, la invocación sería así:

```
QueryBDelegate dbFed = new QueryBDelegate();
resultVO = dbFed.processQuery(sqlQuery, controlVO, ctxVO);
```

3.3. Obtener siguiente resultado

Suponiendo creados el value object con la información de control sobre la consulta y el value object con los datos del contexto de la llamada, la invocación sería así:

```
QueryBDelegate dbFed = new QueryBDelegate();
resultVO = dbFed.getNextResultSet(controlVO, ctxVO);
```

4. Puntos de extensión (Detalles Técnicos)

4.1. Incorporar nuevas estrategias de actualización de la información de la Federación.

Es posible incorporar fácilmente nuevas estrategias de actualización de la información de la Federación. Actualmente se provee una estrategia de actualización denominada “Fuerza Bruta”. Como su nombre lo indica es una estrategia simple en donde el Servidor de Soporte envía al nodo solicitante toda la información que posee sobre la topología (información de la federación). Una posibilidad de extender y a su vez mejorar el mecanismo de actualización sería incorporar metodologías que permitan enviar menos información al nodo para reducir tráfico en la red por ejemplo. Además el subsistema encargado de las actualizaciones transfiere “paquetes” de actualización genéricos serializados entre los servidores, los cuales podrían ser implementados de manera diferente, siempre y cuando cumplan con la interfaz adecuada.

A continuación se brindan los detalles de cómo implementar nuevas estrategias de actualización de la topología.

4.1.1. Detalles técnicos.

El componente encargado de manejar las actualizaciones de la información de la Federación es el Motor de Actualizaciones (UpdateEngine) según la estrategia establecida. Las nuevas estrategias a desarrollar deben cumplir con la interfaz IUpdateStrategy, es decir que deben proveer dos métodos: updateTopology y updateTopologyRemote. El primer método es utilizado para actualizar la topología local desde es el servidor de soporte según la estrategia indicada. Recibe tres parámetros: el contexto en que se realiza la llamada (ContextVO), el nombre del servicio del servidor de soporte que brinda las actualizaciones (serviceName) y la URL del Servidor de Soporte del cual se descargará la información. UpdateTopologyRemote es un método utilizado por el Servidor de Soporte y es el que se invoca cuando llegan pedidos de actualización remotos desde los nodos. Su función consiste en construir y retornar un paquete de actualización con la información solicitada.

Es importante aclarar que conjuntamente con el cambio de estrategia de actualización se podría cambiar la estructura del paquete de actualización que se transfiere entre el servidor de soporte y el nodo. El nuevo paquete implementado debe respetar la interfaz IUpdatePack, la cual básicamente consiste en que todo paquete debe ser capaz de serializarse a XML y reconstruirse a partir del mismo. Otro aspecto técnico a aclarar es sobre el pedido de actualización que realiza el nodo. Este se encapsula en el objeto UpdateRequest y permite detallar entre otras cosas la versión de pedido y estrategia que solicita el nodo para la actualización. Esto hace posible que puedan coexistir diferentes estrategias en el Servidor de Soporte; si el nodo no especifica una el Servidor de Soporte le indica la estrategia utilizada y utiliza la que tiene por defecto para devolver la información; en cambio si el nodo solicita una estrategia en particular el Servidor de Soporte sería capaz de contestarle utilizando la estrategia y el paquete adecuado.

Al crear una instancia del componente UpdateEngine se debe pasar como parámetro la nueva estrategia de actualización desarrollada y a partir de ese momento es el motor el que actualizará o retornará la topología utilizando la nueva estrategia.

A modo de ejemplo se presenta la lógica de la estrategia por defecto (Fuerza Bruta) que utiliza el federador:

ESTRATEGIA Fuerza Bruta:

NODO:

a. Construye el pedido de actualización de topología (UpdateRequest).
b. Envía el pedido de actualización al Servidor de Soporte
c. Si tuvo éxito, elimina todas las entradas de la topología e inserta las nuevas.

SERVIDOR DE SOPORTE:

a. Busca todos los datos de la topología.
b. Construye un paquete de actualización.
c. Serializa a XML y lo envía.

4.2. Incorporar nuevas estrategias de ruteo de aplicaciones

Nuevas estrategias de ruteo pueden ser desarrolladas e incorporadas con facilidad. Actualmente el Federador cuenta con una estrategia por defecto de ruteo de pedidos de aplicaciones. Se podrían escribir nuevas estrategias que tomen en cuenta prioridades, estadísticas de acceso, características de la aplicación en cuestión como ancho de banda del servidor donde están instaladas, etc para así retornar la aplicación “óptima” (según un determinado criterio) al solicitante.

A continuación se brindan los detalles de cómo implementar nuevas estrategias de ruteo de pedidos de aplicaciones.

4.2.1. Detalles técnicos

El componente encargado de rutear las aplicaciones es el Localizador de aplicaciones (ApplicationLocator) según la estrategia configurada. Las nuevas estrategias a desarrollar deben cumplir con la interfaz ISearchAppStrategy, es decir que deben tener un método denominado getApplicationLocation que recibe dos parámetros: los datos de la aplicación solicitada (ApplicationVO) y el contexto en que se realiza la llamada (ContextVO). La función retorna como resultado los datos de la ubicación de la aplicación (LocationVO) que seleccionó el federador al analizar la Federación.

Al momento de instanciar el ApplicationLocator se debe pasar como parámetro la nueva estrategia de ruteo desarrollada y a partir de este momento es el Localizador quien rutea según la nueva estrategia.

A modo de ejemplo se presenta la lógica de la estrategia por defecto que utiliza el federador:

ESTRATEGIA utilizada (Default):

CONTEXTO LOCAL:

1. Busca la aplicación solicitada en las aplicaciones locales.
2. Si la encuentra, retorna los datos de la PRIMERA aplicación cuyo código coincida con el solicitado
3. Si la aplicación no es local, examina la topología de la red para determinar que nodo posee una aplicación que tenga como identificador el código pasado como parámetro.
4. Una vez determinados los nodos, solicita al PRIMERO que tenga la aplicación la ubicación de la misma.

CONTEXTO REMOTO:

1. Busca la aplicación solicitada en las aplicaciones locales.
2. Si la encuentra, retorna los datos de la PRIMERA aplicación cuyo código sea igual al solicitado. Crea una sesión en el nodo devolviendo su identificador.
3. Si no la encuentra, retorna la ubicación en NULL. Habría una inconsistencia en la federación ya que el nodo solicitante tenía indicada en su topología que éste tenia la aplicación.

4.3. Posibilidad de reintentar pedidos fallidos a otros federadores

Se podría brindar la posibilidad de hacer parametrizable cuantas veces se desea reintentar un determinado pedido en caso de que falle (por ejemplo debido a que el federador remoto no puede atender el servicio o el servidor ha sido apagado).

Para esto solo es necesario definir un nuevo tipo de tarea que implemente la interfaz IJobHandler y planificarla para que sea ejecutada por el Scheduler cuantas veces se necesite.

5. Referencias

[1] Documentación Técnica, Fabricio Alvarez, Rodolfo Amador, Proyecto Federador Link-ALL 2004.