Universidad de las Ciencias Informáticas

Facultad 6: Bioinformática

Título: Integración de la plataforma Tarenal con el middleware Globus Toolkit 4.

Trabajo de Diploma para optar por el título de
Ingeniero en Ciencias Informáticas

Autores: Maikel Zúñiga Súarez
Keiler Arnaldo González Torres.

Tutores: Msc. Longendris Aguilera Mendoza
Msc. Noel Moreno Lemus

Ciudad de La Habana, Cuba
Junio-2008

“Lo importante es no dejar de hacerse preguntas.”

Albert Einstein.

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Maikel Zúñiga Suárez

_____

Firma del Autor

Keiler Arnaldo González Torres

_____

Firma del Autor

Msc. Longendris Aguilera Mendoza

_____

Firma del Tutor

Msc. Noel Moreno Lemus

_____

Firma del Tutor

DATOS DEL TUTOR

Msc. Longendris Aguilera Mendoza.
Universidad de las Ciencias Informáticas, Ciudad de La Habana, Cuba.
Email: loge@uci.cu

Msc. Noel Moreno Lemus.
Universidad de las Ciencias Informáticas, Ciudad de La Habana, Cuba.
Email: noel@uci.cu

AGRADECIMIENTOS

A todos los que nos han apoyado y han confiado en nosotros. A los profesores que han contribuido a nuestra preparación, a todas las personas que de una forma u otra han ayudado a que hoy estemos aquí.

DEDICATORIA

A mis padres por apoyarme durante toda mi vida y haber confiado siempre en mí.

A mi abuela y a mi tío por aconsejarme y alentarme en los momentos difíciles.

A todas las personas que durante estos años me han brindado su amor y amistad y han contribuido a que hoy sea mejor persona.

Maikel Zúñiga Suárez

A mis padres y hermanos por haberme apoyado y confiado en mí en todo momento...

A mis amigos por estar presentes en los buenos y malos momentos…

A todos aquellos que hicieron posible que este momento ocurriera...

Keiler Arnaldo González Torres

RESUMEN

El presente trabajo propone una solución a la demanda de cálculo existente actualmente en muchos proyectos de nuestra universidad y del país. El mismo tiene como objetivo fundamental el desarrollo de una infraestructura Grid que incorpora como recurso a Tarenal, que es una plataforma de cómputo distribuido desplegada en la Universidad de las Ciencias Informáticas (UCI) y algunos centros del polo científico del oeste de la capital. Debido a la enorme cantidad de computadoras existentes en la UCI, más de 5 000, se hace necesario el uso de varias plataformas Tarenal para la gestión de las mismas. El sistema desarrollado, sobre el middleware Globus Toolkit 4, es capaz de integrar y compartir varias plataformas Tarenal con el fin de lograr un mayor aprovechamiento del potencial de cálculo que presenta nuestra universidad. También se implementaron un conjunto de portlets que serán incorporados en el Portal de Servicios del Grupo de Bioinformática con el fin de brindar a sus usuarios las funcionalidades de todo el sistema.

PALABRAS CLAVE

Sistema

Recursos

Grid

Estándar

Arquitectura

Especificación

Infraestructura

Plataforma

Cómputo

Middleware

Servicio

INTRODUCCIÓN

Desde tiempos remotos el hombre por varios motivos ha tenido la necesidad de realizar cálculos, con este objetivo se ha propuesto encontrar nuevas formas para realizar los mismos de manera más rápida y de un modo más fácil. Inicialmente se usaron artefactos manuales, y posteriormente mecánicos, hasta el surgimiento de la computadora. En la actualidad, es posible manejar todo tipo de cálculos e información por medio de computadoras, cuyo desarrollo es cada vez más vertiginoso [1].

Durante la última década del siglo XX, los avances de la ingeniería genética, la biotecnología y las nuevas tecnologías de la información, condicionaron el surgimiento de una nueva disciplina que creó vínculos indisolubles entre la informática y las ciencias biológicas: la Bioinformática [2].

Para desarrollar la Bioinformática en Cuba, la UCI creó el Grupo de Bioinformática, el cual tiene numerosos proyectos de colaboración con algunos de los principales centros biotecnológicos del país como el Centro de Ingeniería Genética y Biotecnológica (CIGB) y el Centro de Inmunología Molecular (CIM) [3].

La Bioinformática, al igual que otras disciplinas científicas maneja grandes volúmenes de información, por lo que se ha hecho necesario el uso de nuevas tecnologías para satisfacer la enorme demanda de cálculo requerida por las mismas. La UCI, con el objetivo de aprovechar al máximo sus recursos computacionales y satisfacer en gran medida esta demanda de cálculo desplegó un Sistema de Cómputo Distribuido basado en Java al que se le dio el nombre de Tarenal [4]. Este sistema cuenta fundamentalmente con dos módulos, uno cliente y uno servidor, y tiene como característica la de usar el tiempo ocioso de las computadoras configuradas como clientes. A la UCI le han seguido otros centros como el CIM que también han desplegado esta solución para aprovechar sus recursos computacionales.

La UCI es el centro a nivel nacional con la mayor cantidad de computadoras, contando actualmente con más de 6 000 PCs. La mayoría de estas máquinas son Pentium 4 y se encuentran conectadas mediante una red local con una velocidad de transferencia de 100 Mbps Debido a este gran número de computadoras existente en la UCI se hace necesario tener varias Plataformas Tarenal, sin embargo no existen herramientas para lograr un uso centralizado de todas estas plataformas.

Una solución existente en el mundo para compartir prácticamente cualquier tipo de recurso (hardware, software, datos e información, dispositivos electrónicos, etc.) a través de la red es la tecnología Grid [5]. Estos recursos compartidos se pueden utilizar para resolver problemas de gran escala, y los mismos pueden estar geográficamente dispersos.

Toda Grid es posible gracias al "Grid middleware" [5], el software especial que permite la integración de los distintos tipos de recursos que participan en la misma. El middleware es el cerebro de una Grid y va estar implementado sobre una arquitectura orientada a servicios (SOA) [6]. Para esto hace uso primario de los servicios Grid, los cuales están basados en los servicios Web, añadiendo otras funcionalidades.

Para integrar varias Plataformas de Cómputo Distribuido Tarenal en un único sistema, se decide desarrollar una infraestructura Grid donde Tarenal sería considerado un recurso a compartir. Después de un estudio [4] se decidió por varias razones que el middleware a usar para implementar la infraestructura sería Globus Toolkit 4 (GT4) [7]. Este middleware cuenta con varios componentes y servicios Grid que facilitan la implementación de dicha infraestructura. Sin embargo, estos elementos no son suficientes y se hace necesario el desarrollo de nuevos servicios Grid para lograr integrar y compartir varias plataformas Tarenal en un entorno Grid.

De ahí surge el **problema** ¿Cómo integrar varias plataformas Tarenal en una infraestructura Grid?

Objeto de estudio: La tecnología Grid

Campo de acción: Los servicios Grid

Objetivo general: Desarrollar una infraestructura Grid que incorpore a la plataforma Tarenal como un recurso a compartir.

Para lograr el mismo se han trazado los siguientes **objetivos específicos**:

- Identificar los servicios a incorporar.
- Diseñar una infraestructura sobre una arquitectura orientada a servicios.
- Implementar la infraestructura diseñada.
- Garantizar el acceso a la infraestructura Grid a través del Portal de Servicios.

Para lograr las metas planteadas se le dará cumplimiento a las siguientes tareas:

- Estudio del middleware Globus Toolkit 4.
- Estudio de la plataforma Tarenal.
- Identificación de los servicios Grid a implementar.
- Estudio de los estándares, especificaciones y herramientas existentes para el desarrollo de servicios Grid.
- Levantamiento de requisitos de los servicios a implementar.
- Diseño de los servicios a implementar.
- Implementación de los servicios.
- Despliegue de los servicios implementados.
- Realización de pruebas a nivel de unidad de los servicios implementados.
- Estudio de los estándares, especificaciones y herramientas existentes para el desarrollo de portlets.
- Identificación de los portlets a implementar.
- Implementación de los portlets.
- Despliegue de los portlets en el Portal de Aplicaciones.

El presente trabajo de diploma estará estructurado de la siguiente forma:

- ✓ **Capítulo 1** "Fundamentación teórica", donde se brinda una descripción general de la tecnología Grid y el middleware Globus Toolkit 4, así como los distintos estándares y especificaciones existente para el desarrollo de aplicaciones Grid. Finalmente se brinda una descripción de las herramientas y tecnologías usadas.
- ✓ **Capítulo 2** "Características del sistema", en el mismo se brinda una breve descripción del sistema a desarrollar, se describen los requisitos funcionales y no funcionales, se definen los actores y se presenta el diagrama de casos de uso del sistema, detallando textualmente cada uno de ellos.
- ✓ **Capítulo 3** "Diseño del sistema", donde se explican los patrones de desarrollo de software usados, se presentan los diagramas de clases del diseño y los diagramas de interacción. Finalmente se muestra el diagrama de despliegue, con el objetivo de tener una idea más clara de como quedará la infraestructura una vez desplegada.
- ✓ **Capítulo 4** "Implementación del sistema", en este capítulo se presenta el diagrama de componentes del sistema y el código fuente de las principales clases, haciendo además una breve descripción de las mismas.

CAPITULO 1: FUNDAMENTACIÓN TEÓRICA

En el presente capítulo se brinda una descripción general de la plataforma Tarenal, de la tecnología Grid y el middleware Globus Toolkit 4 (GT4) [7], así como los distintos estándares y especificaciones existentes para el desarrollo de aplicaciones Grid.

Finalmente se da una descripción de las herramientas y tecnologías a utilizar para desarrollar la solución propuesta en este trabajo.

1.1 Plataforma de Cómputo Distribuido Tarenal

Tarenal es una plataforma de cómputo distribuido desarrollada en la UCI. La misma está compuesta principalmente por un módulo cliente y uno servidor. Tarenal usa en general el tiempo ocioso de las computadoras que tengan instalado el módulo cliente y trabaja de la siguiente forma: El módulo servidor recibe los problemas y la forma de dividirlos en subtareas. Por su parte los módulos cliente cada cierto tiempo realizan peticiones de trabajo al servidor, el que se encarga de repartir las subtareas de los problemas a cada uno de los clientes. Cuando un cliente termina de realizar la tarea envía los resultados de la misma al servidor [4].

La plataforma utiliza Remote Method Invocation (RMI) [8] para el envió de mensajes. La misma es software libre y se dispone del código fuente, esto, unido a su fácil instalación hace relativamente sencilla su modificación y distribución según sea necesario. Este sistema es de propósito general, por lo que permite solucionar cualquier tipo de problema que pueda descomponerse en subtareas.

Tarenal brinda un API para la interacción de los diferentes usuarios con el sistema, los usuarios pueden ser invitados, administradores de problemas o administradores en general y van a estar organizados por grupos. A cada usuario se le puede asignar los problemas a los que tiene acceso.

Este API brinda funcionalidades según el nivel de acceso del usuario entre otras cosas para:

- Gestión de problemas, ejecuciones y resultados.
- Actualización de los módulos cliente. administración del servidor.
- Configuración de cuentas de usuario.

Este API puede ser usado por otros sistemas.

1.2 Computación Grid

A lo largo de la historia, muchos desarrollos científicos y tecnológicos han surgido como respuestas a alguna necesidad. Hace un tiempo se planteó la pregunta: ¿Cómo traer hasta un ordenador los datos que están almacenados en otras máquinas?, y como respuesta a esta necesidad surgió la "World Wide Web".

Debido a la creciente demanda de cómputo y de capacidad de almacenamiento, apareció una nueva necesidad y pregunta: ¿Cómo compartir no solo información, sino también poder de cálculo y capacidad de almacenamiento? La respuesta a esta pregunta fue precisamente la Grid. [5]

El término Grid se refiere a una infraestructura que permite la integración y el uso colectivo de ordenadores de alto rendimiento, redes y bases de datos que son de propiedad y están administrados por diferentes instituciones. Puesto que la colaboración entre instituciones envuelve un intercambio de datos, o de tiempo de computación, el propósito de la Grid es facilitar la integración de recursos computacionales [9].

1.3 ¿Cómo funciona la Grid?

La infraestructura Grid descansa sobre un software denominado “middleware”, que asegura la comunicación transparente entre los distintos ordenadores intercomunicados.

También poseen un motor de búsqueda que no sólo encontrará los datos que el usuario necesite, sino también las herramientas para analizar y la potencia de cálculo necesaria para utilizarlas. Al finalizar el proceso, la Grid distribuirá las tareas de computación a cualquier lugar de la red en la que haya capacidad disponible y enviará los resultados al usuario [5].

El funcionamiento de la Grid se basa en cinco pilares básicos [5]:

- La posibilidad de compartir recursos
- La seguridad(acceso seguro)
- El uso eficiente de los recursos
- Redes de comunicación fiables que eliminen las distancias

- Estándares abiertos

La posibilidad de compartir recursos

Esta es la idea clave de la Grid: poder utilizar recursos remotos que nos permitan realizar tareas que no podríamos llevar a cabo en nuestra máquina o centro de trabajo. Esta idea va más allá del simple intercambio de ficheros: se trata del acceso directo a software, ordenadores y datos remotos, así como acceso y control de otros dispositivos [5].

Acceso seguro

La seguridad en la Grid se centra en los siguientes aspectos [5]:

- **Política de Accesos:** Permite definir cuáles son los recursos que se van a compartir, a quién se permite el acceso a dichos recursos y bajo qué condiciones.
- **Autenticación:** Permite establecer la identidad de un usuario o de un recurso concreto.
- **Autorización:** Permite determinar si una determinada operación está permitida de acuerdo a las relaciones que se han definido previamente.

La utilización eficiente de recursos

El tercer aspecto fundamental en la tecnología Grid es el uso eficiente de los recursos. Es aquí donde radica el verdadero interés de la Grid. No importa la cantidad de recursos de los que uno disponga; siempre habrá usuarios haciendo cola para utilizarlos. Se necesitan mecanismos para repartir el trabajo de forma automática y eficiente entre una gran cantidad de recursos, reduciendo las colas de espera [5].

Redes de comunicaciones rápidas y fiables

La existencia de conexiones de alta velocidad es lo que hace posible la Grid a escala mundial. Hace diez años hubiese sido ingenuo tratar de enviar grandes cantidades de datos a través del mundo para que se pudiesen procesar más rápido en otros ordenadores. El tiempo que se tardaba en transferirlos anularía el beneficio de un procesamiento más rápido [5]

Estándares abiertos

El objetivo es conseguir que las aplicaciones que se ejecuten en una Grid puedan funcionar en cualquier otra. Debido a que la naturaleza de la Grid es compartir recursos, es comprensible que la existencia de estándares abiertos redunde en beneficio de todos los agentes participantes [5].

Actualmente, los estándares de Grid son desarrollados fundamentalmente por organismos de normalización, que están compuestos por organizaciones, firmas y desarrolladores que han guiado sus esfuerzos a este desarrollo.

1.4 Organismos de normalización

1.4.1 Global Grid Forum (GGF)

Global Grid Forum [10] es una iniciativa que reúne a los principales Centros de Investigación de EEUU y Europa, así como a las más relevantes empresas del sector, y cuyo objetivo es promover y apoyar el desarrollo, despliegue e implementación de tecnologías y aplicaciones Grid por medio de la creación y documentación de especificaciones técnicas, experiencias de usuarios y guías de implementación [11]. Entre los estándares producidos por GGF están: OGSA (Open Grid Services Architecture) [12, 13, 14,15], DRMAA (Distributed Resource Management Application API) [16] y JSDL (Job Submission Description Language) [17], entre otros.

En 2006, el Global Grid Forum se fusionó con Enterprise Grid Alliance [18], otro grupo estadounidense de estandarización Grid, con el objetivo de acelerar la adopción de las tecnologías Grid en todo el mundo, surgiendo de esta forma Open Grid Forum [19].

Los estándares y productos de GGF han sido subsumidos dentro de los estándares de OGF.

1.4.2 Open Grid Forum (OGF)

Open Grid Forum [19] surge en junio de 2006 a partir de la unión de Global Grid Forum con Enterprise Grid Alliance. Dispone de grupos de trabajo en varias áreas: de aplicaciones, de arquitectura, de cómputo, de datos, de gestión y de seguridad.

Dentro de las aplicaciones desarrolladas por OGF tenemos un API para el envío y control de trabajos (drmaa-wg), un API y servicios relacionados para check pointing (gridcpr-wg), un API para llamada de procedimientos remotos Grid (gridrpc-wg) y un api para aplicaciones de red (saga-core-wg).

1.4.3 Organization for the Advancement of Structured Information Standards (OASIS)

OASIS [20] es un consorcio internacional sin fines de lucro que orienta el desarrollo, la convergencia y la adopción de los estándares e-business. OASIS ha aprobado la versión 2.0 de UDDI (Universal Description, Discovery and Integration) [21], uno de los estándares claves en la arquitectura de servicios Web junto a XML, SOAP (Simple Object Access Protocol) [22] y WSDL (Web Services Description Language) [23].

1.4.4 World Wide Web Consortium (W3C)

W3C [24] es un consorcio internacional que produce estándares para guiar el desarrollo y expansión de la Web. OASIS es miembro de este consorcio, el cual ha participado con el OGF en el área de la estandarización de servicios Web.

1.5 Especificaciones y estándares

Como en todos los ámbitos tecnológicos, en el ámbito de la Grid es de especial importancia la existencia de estándares y especificaciones que permitan un desarrollo homogéneo. La arquitectura OGSA [12, 13, 14,15], pretende servir de base para estandarizar el desarrollo de aplicaciones Grid. El principal resultado ha sido Globus Toolkit [7].

OGSA está basado sobre la tecnología y conceptos de los servicios Web. Un servicio Web es definido por el W3C como "un sistema de software diseñado para apoyar la interoperabilidad entre ordenadores a través de una red".

Los componentes básicos de la arquitectura de servicios Web son: SOAP [22] para las comunicaciones, Web Services Description Language (WSDL) [23] para describir los servicios de red, y el protocolo Universal Description Discovery & Integration (UDDI) [21] para definir un conjunto de servicios de apoyo a la descripción y localización de las empresas, organizaciones, proveedores de servicios, los servicios disponibles, y las interfaces técnicas utilizadas para acceder a estos servicios.

1.5.1 Open Grid Services Architecture (OGSA)

La OGSA (Open Grid Services Architecture) [12, 13, 14,15], desarrollada por el Global Grid Forum (GGF) [10], es un estándar que tiene por función definir una arquitectura abierta y común para todas las aplicaciones basadas en Grid. El objetivo de esta última es estandarizar prácticamente todos los

servicios que se pueden encontrar en una aplicación Grid (servicios de gestión de recursos, servicios de seguridad, etc.) especificando un conjunto de interfaces para estos servicios [5].

1.5.2 Simple Object Access Protocol (SOAP)

SOAP [22] es un protocolo auspiciado por la W3C [24] que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML. SOAP es uno de los protocolos utilizados en los servicios Web y funciona sobre cualquier otro protocolo de Internet.

A diferencia de DCOM [26] y CORBA [25], que son binarios, SOAP usa el código fuente en XML. Esto es una ventaja ya que facilita su lectura por parte de humanos, pero también es un inconveniente dado que los mensajes resultantes son más largos.

1.5.3 Web Services Description Language (WSDL)

WSDL [23] es un formato XML que se utiliza para describir servicios Web. Describe la interfaz pública de los servicios Web y la forma de comunicación, es decir, los requisitos del protocolo y los formatos de los mensajes necesarios para interactuar con los servicios listados en su catálogo. Las operaciones y mensajes que soporta se describen en abstracto y se ligan después al protocolo concreto de red y al formato del mensaje. Así, WSDL se usa a menudo en combinación con SOAP [22] y XML Schema.

Un programa cliente que se conecta a un servicio Web puede leer el WSDL para determinar que funciones están disponibles en el servidor. Los tipos de datos especiales se incluyen en el archivo WSDL en forma de XML Schema. El cliente puede usar SOAP para hacer la llamada a una de las funciones listadas en el WSDL.

1.5.4 Universal Description Discovery and Integration (UDDI)

UDDI [21] es un estándar que define un conjunto de servicios para dar soporte a la descripción y descubrimiento de:

- Empresas, organizaciones, y otros proveedores de servicios Web.
- Los servicios Web disponibles.
- Las interfaces técnicas que pueden ser usadas para acceder a estos servicios.

UDDI está basado en un conjunto de estándares que incluye, HTTP, XML, XML Schema y SOAP, que provee una infraestructura para un software basado en servicios Web para ser publicado y buscado.

1.5.5 Web Service Resource Framework (WS-RF)

WS-RF [27] forma parte de la familia de especificaciones relativas a servicios Web publicadas por OASIS [20]. Los principales impulsores de esta especificación son IBM y la Globus Alliance.

El propósito de WSRF es definir un framework genérico para modelar y acceder a recursos persistentes usando servicios Web, logrando con esto que la definición e implementación de un servicio de este tipo y la integración y administración de varios de ellos sea más fácil.

Un servicio Web no tiene estado, es decir, no puede recordar información o mantener el estado de una invocación a otra. Esta falta de estado limita el número de usos que se le puede dar a los mismos.

Para minimizar el complejo problema de agregar un estado a los servicios Web se propuso que un servicio y su estado se mantengan completamente separados. La entidad encargada de almacenar este estado es llamada recurso, el cual cuenta con un identificador único. A la pareja de un servicio Web con un recurso se le llama WS-Resource. La dirección de un WS-Resource particular es llamada referencia final (WS-Addressing).

WSRF proporciona un conjunto de operaciones que los servicios Web compatibles pueden implementar para lograr tener estado; los clientes de los mismos se comunican con servicios WSRF que representan a recursos y que permiten almacenar y recuperar información. Los clientes invocarán el servicio añadiendo como parámetro el identificador del recurso que será utilizado durante la petición, codificado en una referencia que cumpla con WS- Addressing.

WSRF integra la siguiente colección de especificaciones para la gestión de WS-Resource:

- WS-ResourceProperties: Especifica como las propiedades de los recursos son definidas y accedidas. Un recurso esta compuesto por cero o muchas propiedades de recursos.

- WS-ResourceLifetime: Provee mecanismos básicos para gestionar el ciclo de vida de los recursos. Los recursos pueden ser creados y destruidos en cualquier momento.

- WS-ServiceGroup: Especifica exactamente como agrupar los servicios o WS-Resources

- WS-BaseFaults: Brinda una forma estándar de reportar fallas cuando algo está funcionando de manera incorrecta durante la invocación de un servicio Web

- WS-Notification: No es parte de WSRF pero están estrechamente relacionados. Permite a un servicio Web ser configurado como notificador, y a un cliente a ser consumidor o subscriptor de notificaciones. Esto significa que si ocurre un cambio en el servicio o en uno de los WS-Resources este cambio es notificado a los subscriptores.

WS-Addressing: Provee un mecanismo para direccionar servicios Web que es mucho más versátil que las URIs.

WSRF proporciona también un conjunto de operaciones estandarizadas para la consulta y modificación de propiedades del recurso representado. Estas operaciones se pueden utilizar para leer y/o modificar el estado del recurso, tal y como los métodos de un objeto comparten las variables de clase.

1.6 Servicios Grid

La base fundamental de los servicios Grid son los servicios Web (Web services), tecnología de computación de sistemas distribuidos que permite la creación de aplicaciones basándose en el modelo cliente/servidor mediante una plataforma independiente del lenguaje y utilizando protocolos abiertos como HTTP. [29]

En la Figura 1 se describe el proceso de invocación de un servicio Web. El contenedor de servicios Web se refiere a la aplicación dentro de la cuál están disponibles los mismos. Los “stubs” son piezas de software en la que delegan tanto cliente como servidor para dar el adecuado formato SOAP a sus peticiones y respuestas. Los “stubs” del servidor se generan a partir del a descripción WSDL [23], previamente definida. El desarrollo de estos servicios habitualmente no necesita considerar los detalles SOAP [22] y WSDL ya que las herramientas generan automáticamente estos “stubs”. [29].

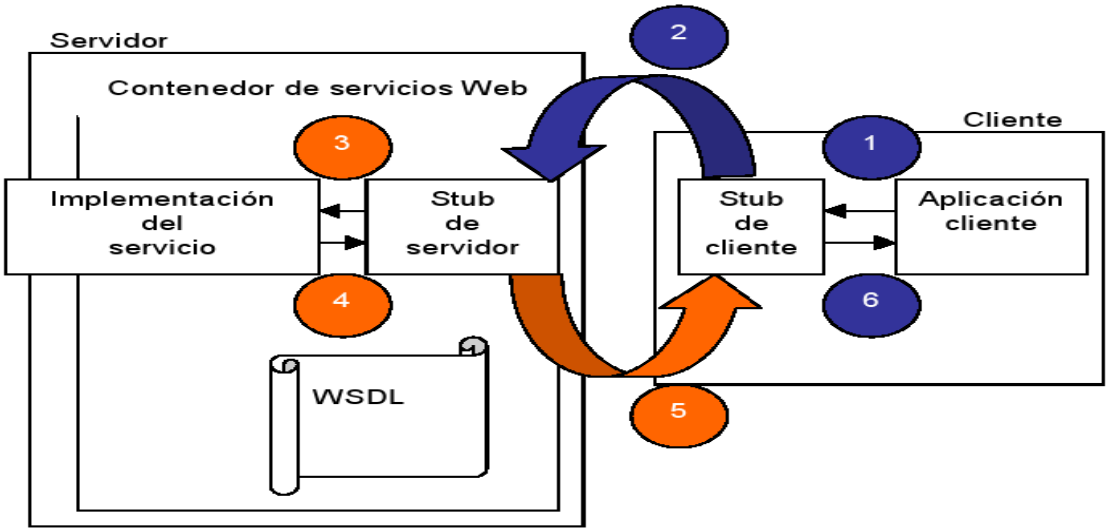

Figura 1. Invocación de un servicio Web

Las comunicaciones entre cliente y servidor siguen el siguiente protocolo:

- Cuando una aplicación cliente necesita invocar un servicio Web, llamará al stub de cliente. Este se encargará de dar el adecuado formato SOAP a la petición.
- La petición SOAP es enviada a través de la red mediante el protocolo HTTP. El contenedor de servicios Web recibe la petición SOAP y la reenvía al stub de servidor. Éste convertirá la petición SOAP en algo que la implementación del servicio (desarrollado en algún lenguaje de programación) pueda entender.
- La implementación del servicio recibe la petición del stub de servicio y lleva a cabo su trabajo.
- El resultado de este trabajo se envía al stub del servidor, que le da un formato SOAP a la respuesta
- La respuesta SOAP viaja, vía HTTP, de vuelta al cliente, cuyo stub transforma en algo que la aplicación pueda entender.
- La aplicación cliente recibe el resultado de la invocación del servicio Web.

Los servicios Grid son ampliaciones de los servicios Web, su arquitectura está especificada por el Global Grid Forum. Algunas de las extensiones que incluyen los servicios Grid y que no poseen los servicios Web son [29]:

- Memoria de estado: Permite almacenar el último resultado de una invocación, que puede ser utilizado por otro cliente. De esta manera se ahorra tiempo y trabajos redundantes.
- Servicios potencialmente transitorios: Los servicios Grid permiten utilizar el modelo fábrica/instancia. Una fábrica no es más que un servicio Grid que puede crear otros servicios Grid (instancias) cuya vida no está vinculada a la del contenedor que los alberga (son transitorios). La fábrica puede controlar el ciclo de vida de sus instancias.
- Notificaciones: Se pueden programar cliente y servidor para que determinados cambios en este último sean notificados al primero (por ejemplo, los progresos que realiza el servidor en un trabajo que ha enviado el cliente).
- Datos de servicio: Esta potente característica permite tener un conjunto estructurado de información asociada al servicio Grid. Esta información puede ser de estado (resultado de las operaciones, del tiempo de ejecución, etc.) o del propio servicio (interfaces soportadas, coste del servicio, etc.). Los elementos que contienen datos de servicio reciben el nombre de SDEs (*Service Data Elements*).
- Extensión de la interfaz: Gracias a esta característica, un servicio Grid puede llevar a cabo múltiples funciones de una manera sencilla: extendiéndose a partir de una o varias interfaces previamente existentes. Por ejemplo, un simple servicio Grid puede convertirse en fábrica y en fuente de notificaciones agregando únicamente unos términos en la línea portType (que define la interfaz) del documento WSDL [23].

Por todas estas razones podemos pensar en los servicios Grid como servicios Web de características mejoradas.

Para lograr estas funcionalidades los servicios Grid usan la especificación WSRF [27], vista en la sección anterior.

1.7 Arquitectura de la Grid

OGSA [12,13,14,15] es comúnmente usada como una arquitectura abierta y estándar, haciendo uso de los mejores aspectos del Modelo de Arquitectura de la Grid y la Arquitectura Orientada a Servicios (Modelo SOA) [6]. En esta sección se describe brevemente los modelos arquitectónicos, las capacidades definidas en OGSA así como las especificaciones de implementación que nos brinda WS-RF.

1.7.1 Modelo Arquitectónico de la Grid

El modelo arquitectónico de la Grid define las capas para proveer interoperabilidad. Esto permite al sistema Grid ofrecer un entorno compartido entre los proveedores de recursos y los consumidores a través de conjunto de protocolos comunes. Este modelo cuenta con cinco capas descritas brevemente a continuación [30]:

- Capa de tejido: Define las interfaces para acceder a los recursos locales que van a ser compartidos.
- Capa de conectividad: Define los protocolos de autenticación y comunicaciones de las transacciones de la red sobre los recursos.
- Capa de recursos: Usa los protocolos de autenticación y comunicaciones de la capa de conectividad para gestionar la compartición de recursos de forma individual, controlando las negociaciones de forma segura, instanciado y monitoreo. También usa la función de la capa de tejido para acceder y controlar los recursos locales.
- Capa colectiva: Gestiona las interacciones entre colecciones de recursos. Usa protocolos de la capa de conectividad y la capa de recursos. Además implementa protocolos de información para acceder a la estructura y estado de los recursos, así como gestiona protocolos para negociar el acceso a recursos.
- Capa de Aplicación: Incluye las diferentes aplicaciones de los usuarios, portales y herramientas de desarrollo que soporten las aplicaciones. Las aplicaciones se basan en todas las capas para ejecutarse en la Grid.

1.7.2 Service Oriented Architecture (SOA)

SOA es un paradigma para organizar recursos distribuidos que pueden estar bajo el control de diferentes dominios. Un servicio en SOA es un mecanismo para habilitar el acceso a uno o más recursos. Los servicios en SOA tienen tres propiedades fundamentales [31]:

- El acceso se provee usando una interfaz consistente prescrita con las restricciones y políticas especificadas en la descripción del servicio. Esta propiedad es usada para describir y publicar servicios.
- La localización e invocación pueden ser dinámicas: Esta propiedad es utilizada para

descubrir servicios.

- Mantienen su propio estado: Esto permite que los servicios tengan estado para el consumo o interacción.

Los principales componentes de SOA son: el proveedor, el solicitante y el registro de servicios [6]. Figura 2

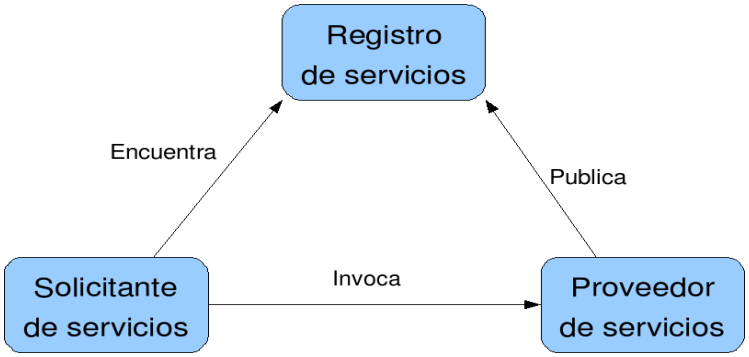

Figura 2. Elementos básicos de SOA

El proveedor de servicio es el responsable de construir el servicio, crear la descripción del mismo, publicar dicha descripción en uno o más registros, y recibir mensajes de invocación de uno o muchos solicitantes.

Por su parte los solicitantes de servicios son los que descubren las descripciones de los mismos en los registros y las usan para invocar los servicios. Cualquier consumidor de un servicio puede ser considerado como solicitante.

Finalmente los registros de servicios son los responsables de poner a disposición o publicar todas las descripciones de servicios, publicadas en él por los proveedores, permitiéndoles a los solicitantes buscar dichas descripciones.

1.7.3 Open Grid Services Architecture (OGSA)

En OGSA [12, 13, 14,15] un servicio Grid es un servicio Web que provee un conjunto de interfaces bien definidas y cumple con convenciones específicas. Lo servicios implementados bajo OGSA caen dentro de siete áreas definidas en términos de capacidades que frecuentemente requieren escenarios Grid [32]:

- Servicios de infraestructura: Hacen referencia a un conjunto de funcionalidades comunes comúnmente requeridas por servicios de más alto nivel. Como OGSA se construye sobre la tecnología de servicios Web, las interfaces de los servicios son definidas por WSDL. La infraestructura incluye estándares emergentes como WSRF, WS-Notification (WSN), Web Services Distributed Management (WSDM), y Naming (RNS y WS-Naming).

- Servicios de gestión de ejecución: Se encargan de asuntos tales como la puesta en marcha y la gestión de tareas, incluyendo la gestión de colocación, suministro y ciclo de vida. Las tareas pueden ser desde simples trabajos hasta flujos de trabajo complejos o servicios compuestos.

- Servicios de datos: Proveen funcionalidades para mover datos, gestionar copias replicadas, ejecutar consultas y actualizaciones, además permite convertir datos a nuevos formatos. La consistencia, persistencia e integridad de los datos son requisitos básicos que deben satisfacer estos servicios.

- Servicios de gestión de recursos: Brindan capacidades para gestionar recursos Grid como son autogestión de los recursos, gestión de los recursos como componentes Grid y gestión de la infraestructura de OGSA. O sea, los recursos pueden ser monitorizados, reservados, desplegados y configurados como sea necesario para cumplir con los requisitos de calidad de servicio.

- Servicios de seguridad: Facilitan la aplicación de políticas basadas en seguridad dentro de una organización virtual (VO) y soporta la compartición de recursos de manera segura. Autenticación, autorización y garantía de la integridad son funcionalidades esenciales brindadas por estos servicios.

- Servicios de autogestión: Soportan completamiento a nivel de servicios para grupos de servicios o recursos con el objetivo de reducir los costos de complejidad de gestionar del sistema. Estos servicios son esenciales en sentido de la creciente complejidad de poseer y operar una infraestructura tecnológica de información.

- Servicios de información: Proveen de manera eficiente información sobre la Grid y los recursos que la componen. Se refiere a información dinámica o eventos usados para el

monitoreo de estado. Esta información puede ser utilizada para solucionar problemas.

1.8 Los Grid middlewares

El Grid middleware es el encargado de proporcionar las herramientas que permiten que los distintos elementos (servidores, almacenes de datos, redes, etc.) participen de forma coordinada en un entorno Grid unificado [5].

El middleware, es el auténtico "cerebro" de la Grid y se ocupa de las siguientes funciones [5]:

- Encontrar el lugar conveniente para ejecutar la tarea solicitada por el usuario.
- Optimiza el uso de recursos que pueden estar muy dispersos.
- Organiza el acceso eficiente a los datos.
- Se encarga de la autenticación de los diferentes elementos.
- Se ocupa de las políticas de asignación de recursos.
- Ejecuta las tareas.
- Monitoriza el progreso de los trabajos en ejecución.
- Gestiona la recuperación frente a fallos.
- Avisa cuando se haya terminado la tarea y devuelve los resultados.

1.8.1 Globus Toolkit 4

Globus Toolkit [7] es un Grid middleware, desarrollado por Globus Alliance [33], el cual puede ser usado para programar aplicaciones basadas en la Grid incluyendo un conjunto de librerías, servicios y APIs. Provee además por defecto un contenedor denominado Apache Axis que es capaz de interpretar las peticiones SOAP [22].

En primer lugar incluye un conjunto de servicios de alto nivel que pueden ser usados para construir aplicaciones Grid. Estos servicios cumplen la mayoría de los requerimientos de OGSA [12, 13, 14, 15,32], y podemos encontrar desde servicios de monitorización y descubrimiento de recursos hasta servicios de gestión de datos. Como los grupos de trabajo de GGF [10] aún continúan trabajando en la definición de interfaces estándares para esto tipos de servicios, no podemos decir, en este sentido, que Globus Toolkit 4 es una implementación de OGSA. Aunque si es una realización de los requerimientos

de OGSA y un pequeño estándar para la comunidad de desarrolladores de Grid, mientras que GGF trabaja en la estandarización de todos los distintos servicios [11].

1.8.1.1 Seguridad en Globus Toolkit 4

La seguridad en Globus se alcanza a través de GSI (Grid Security Infrastructure) [34,35], basándose en la criptografía de clave pública. Los conceptos de clave pública y privada y el de firma digital, son básicos dentro de GSI, incluyendo GT4 en su implementación una CA (Certification Authority) [36].

Las necesidades de seguridad en Globus, en cualquier Grid en general son las siguientes [28]:

- Comunicaciones seguras (autenticación, confidencialidad e integridad de los datos) entre los componentes de la Grid.
- La necesidad de soportar un modelo de seguridad por encima de los límites organizacionales, es decir, no usar un modelo centralizado de la gestión de seguridad.
- La necesidad de un logon único (Single Sign On) con delegación de credenciales y un sistema adecuado de control de acceso a los recursos sin la necesidad de múltiples autenticaciones.

La herramienta básica en GSI es el certificado digital X.509 [38].

En la Grid se debe configurar una CA que firme los certificados que se usen en la Grid y que permita la autenticación mutua entre los componentes de la misma. Cada componente de la Grid debe confiar en la misma CA de manera que esa CA sea la que firme los certificados digitales de todos los compontes.

La confidencialidad de los datos se realiza mediante encriptación de clave asimétrica usando las claves públicas y privadas delos componentes. Estas mismas claves son usadas para la integridad de las comunicaciones [28].

El logon único de la Grid que implementa GSI se basa en el mecanismo de delegación de credenciales. Las credenciales son en este caso certificados x.509. La delegación se consigue

mediante un proxy, que no es más que otro certificado con información que lo identifica como este tipo de certificado [28].

1.8.1.2 Principales componentes de Globus Toolkit 4

Muchos de los componentes de Globus Toolkit 4 tienen su implementación basada en servicios Web sobre la especificación WSRF [27], mientras que otros no, y por compatibilidad algunos tienen ambas implementaciones.

Componentes comunes de ejecución: Son librerías y herramientas necesarias para construir servicios, ya sean Web o no. Incluye soporte para Java, C y Python.

Componentes de seguridad: Debido a la importancia que se le concede a la seguridad en una Grid, Globus provee varios componentes, que permiten entre otras cosas:

- El manejo de credenciales, donde se encuentran SimpleCA [36] y MyProxy [38,39].
- La autenticación y autorización, con lo que se controla el acceso a los servicios y recursos.
- La delegación de credenciales, donde existe un servicio que delega credenciales a un contenedor
- La autorización comunitaria, donde el Servicio de Autorización Comunitaria (CAS) [36] permite administrar políticas en Organizaciones Virtuales sobre sus recursos.

Componentes de gestión de datos: En esta parte podemos encontrar componentes, que permiten entre otras cosas la localización, transferencia y gestión de datos a través de una infraestructura Grid, usando distintos protocolos y servicios. Dentro de estos componentes podemos encontrar:

- **GridFTP:** Constituye un cliente FTP optimizado para la transferencia de grandes cantidades de datos, y de forma segura, usando el protocolo gsiftp(GridFTP= GSI + FTP). No está implementado como WS. Globus incluye una implementación del servidor (globus-gridftp-server), un cliente (globus-url-copy) y una completa API para desarrollo [40,41].
- **Reliable File Transfer (RFT):** Servicio que permite programar transferencias gsiftp (por debajo utiliza GridFTP). Tiene mayor tolerancia a fallos (usa una base de datos para guardar en memoria persistente las transferencias que se han programado) [42]. Esta

implementado como WS. Es utilizado por WS GRAM .

Componentes de monitoreo y descubrimiento: Estos componentes se encargan principalmente de la recolección, distribución, listado, archivado, y otro tipo de información procesada acerca del estado de varios recursos, servicios y configuraciones del sistema. Esta información recolectada puede ser utilizada tanto para descubrir nuevos servicios o recursos, como para monitorear el estado del sistema. Dentro de estos componentes se encuentran:

- **Index Service**, el cual permite agregar recursos de interés a la Grid [43].
- **Trigger Service**, que colecta datos de los recursos y realiza acciones basada en la información reunida [44]
- **WebMDS**, que proporciona una vista apta para un navegador de los datos colectados por el servicio de indexado [45].

Componentes de administración de ejecución: Estos componentes nos posibilitan inicializar, monitorear, administrar, programar y coordinar programas ejecutables, usualmente llamados jobs, en un entorno Grid. El principal servicio encontrado dentro de este grupo de componentes es el WS-GRAM (Web Services Grid Resource Allocation and Management) [46]
El componente WS-GRAM esta implementado sobre los servicios Web y usa el RFT para la transferencia de ficheros .

1.9 Herramientas y tecnologías

En esta sección se hace referencia a algunas herramientas, tecnologías y lenguajes de programación existentes que pudieran ser útiles para dar solución al problema planteado con el objetivo de luego de analizar las características, ventajas y desventajas de cada uno hacer una selección de los que se van a utilizar finalmente.

1.9.1 Metodologías de desarrollo de software

Uno de los principales retos que enfrentan los desarrolladores de software es el desarrollo con calidad y de manera eficiente, por lo cual supone un paso importante hacer una selección correcta de las herramientas y procesos necesarios para lograr este objetivo. Para lograr este objetivo se crearon diferentes metodologías de desarrollo de software, las cuales son un conjunto de pasos y procedimientos que deben seguirse para el desarrollo de software. Con los años y las experiencias que han ido teniendo las diferentes instituciones desarrolladoras de software se han ido mejorando algunas

metodologías de desarrollo y se han creado muchas nuevas con el objetivo de que el desarrollador pueda emplear su tiempo de manera eficiente, así como para lograr una adecuada organización y comunicación en el ambiente de trabajo de los grupos de desarrollo.

Entre la familia de metodologías existen las metodologías ágiles, las cuales intentan evitar los intricados y burocráticos caminos de las metodologías tradicionales enfocándose en la gente y los resultados. Estas metodologías recogen ventajas de las metodologías tradicionales e incorporan características nuevas que hacen que el proceso de desarrollo sea más simple basándose en que lo más importante en un proyecto es valorar más a los individuos que a los procesos y herramientas, al software que funciona más que a la documentación exhaustiva, a la colaboración del cliente más que a la negociación contractual, a la respuesta al cambio más que al seguimiento de un plan.

El manifiesto ágil [47], la base de este tipo de metodologías se compone de doce premisas:

- La prioridad es satisfacer al cliente mediante entregas de software
- Dar la bienvenida a los cambios
- Entregar frecuentemente software que funcione
- La gente de negocio y los desarrolladores deben trabajar juntos en el proyecto
- Construir el proyecto en torno a individuos motivados.
- El diálogo cara a cara es el mejor método de comunicación.
- El software que funciona es la medida principal de progreso
- Los procesos ágiles promueven un desarrollo sostenible, todos deben trabajar en paz
- La atención continua a la calidad técnica y al diseño mejoran la calidad
- La simplicidad es esencial
- Las mejores arquitecturas, requisitos y diseños surgen de los equipos organizados por sí mismos
- En intervalos regulares, el equipo busca ser más efectivo

.

Proceso Unificado Abierto o OpenUP (Open Unified Process) [48], es una de las metodologías ágiles de desarrollo de software. Es parte del marco de trabajo de procesos de Eclipse [49], un marco de trabajo de código abierto desarrollado dentro de la fundación de Eclipse.

OpenUP ofrece las mejores prácticas de una variedad de líderes en ideas sobre desarrollo de software y comunidades de desarrollo que cubren un diverso conjunto de perspectivas y necesidades de desarrollo. Preserva las características esenciales de RUP [50] que incluye el desarrollo interactivo, casos de uso y escenarios de conducción de desarrollo, la gestión de riesgo y el enfoque centrado en la arquitectura.

OpenUP/Basic es la forma más ágil y ligera de OpenUP [48], y se basa en una donación de código abierto al proceso de contenido conocido como el Proceso Unificado básico (BUP). La mayoría de las partes de RUP han sido excluidas de esta metodología y muchos elementos se han fusionado, siendo el resultado un proceso mucho más sencillo que coincide con los principios de RUP. OpenUP/Basic es aplicable a proyectos pequeños con grupos de 3 a 6 personas interesadas en el desarrollo rápido e interactivo.

Para dar solución al problema usaremos OpenUP/Basic como metodología de desarrollo.

1.9.2 Lenguaje de modelado

Para dar solución al problema planteado, usaremos como lenguaje de modelado UML. Este lenguaje unificado de modelado (UML) es el lenguaje de modelado de sistemas de software más utilizado y conocido en la actualidad. Esta respaldado por la OGM (Object Management Group) [51], pero todavía no es un estándar oficial. Es un lenguaje gráfico para especificar, construir y documentar sistemas de software. UML es un lenguaje para especificar y no para describir métodos o procesos. Se puede aplicar en una variedad de formas para dar soporte a una metodología de desarrollo de software, pero no especifica que metodología o proceso utilizar.

1.9.3 Herramientas CASE

Las herramientas Case (Computer Aided Software Engineering, Ingeniería de Software Asistida por Computadoras) son un conjunto de métodos, utilidades y técnicas que facilitan la automatización del ciclo de vida del desarrollo del sistema de información, completamente o en algunas fases. También pueden mejorar la productividad en el desarrollo de una aplicación de bases de datos. Y por productividad se entiende tanto la eficiencia en el desarrollo, como la efectividad del sistema desarrollado.

Las herramientas fueron diseñada para:

- Soportan un entorno personal dedicado.
- Utilizar Gráficos para especificar y documentar los sistemas.
- Unir todas las fases del ciclo del software.
- Utilizar la inteligencia artificial para realizar automáticamente muchas de las rutinas, tareas de desarrollo y mantenimiento del software.

Como ejemplo de herramientas CASE tenemos MagicDraw, Rational Rose, Umbrello, Visual Paradigm for UML, ArgoUML y muchos otros.

Por decisión del Grupo de Bioinformática la Herramienta CASE que se utilizará para dar solución al problema en cuestión es Visual Paradigm.

Visual Paradigm es una herramienta CASE que soporta la ultima versión del lenguaje de modelado unificado (UML) y la notación del proceso de modelado de negocio (BPMN). Brinda soporte para el modelado de UML y modelado de procesos de negocio y genera código para un gran número de lenguajes de programación. Además brinda una versión libre para uso no comercial. La herramienta fue desarrollada para una amplia gama de usuarios incluyendo Ingenieros de Software, analistas de sistemas, analistas del negocio y arquitectos de sistemas. Permite la integración con varias herramientas de Java como son Eclipse, Netbeans, Jbuilder, entre otras, además brinda una gran interoperabilidad con otras Herramientas CASE como Rational Rose.

1.9.4 Lenguaje de programación

Un lenguaje de programación es un lenguaje que permite expresar las instrucciones que han de ser ejecutadas en una computadora, permitiéndole al programador especificar de manera precisa: sobre qué datos una computadora debe operar, cómo deben ser estos almacenados y transmitidos y qué acciones debe tomar bajo determinadas circunstancias. Todo esto, a través de un lenguaje que intenta estar lo más cerca posible al lenguaje humano o natural.

Para dar solución al problema se hará uso del lenguaje de programación Java.

Java es un lenguaje de programación orientado a objetos diseñado a principios de los años 90 por la compañía Sun Microsystems Inc, con el propósito de que pudiera funcionar en redes computacionales heterogéneas, y que fuera independiente de la plataforma en la que se vaya a

ejecutar. El lenguaje toma mucha de su sintaxis de C y C++, pero tiene un conjunto de características que lo diferencias de los mismos:

- Maneja la memoria de la computadora por el programador, evitándole al mismo tener que preocuparse por apuntadores, memoria que no se esté utilizando, etc. Java realiza todo esto sin necesidad de que uno se lo indique.
- Tiene ciertas políticas que evitan la codificación de virus con este lenguaje. Existen muchas restricciones, especialmente para los applets, que limitan lo que se puede y no hacer con los recursos críticos de una computadora.
- Al compilar un programa en Java, el código resultante de compilar un programa en Java es un tipo de código binario conocido como byte code, el cual es interpretado por diferentes computadoras de igual manera. Solamente hay que implementar un intérprete para cada plataforma. De esa manera Java logra ser un lenguaje que no depende de una arquitectura computacional definida
- No requiere compilar todas las clases de un programa para que este funcione. Si realizas una modificación a una clase Java se encarga de realizar un Dynamic Binding o un Dynamic Loading para encontrar las clases.

1.9.5 Herramienta de desarrollo

Para dar solución al problema se va a hacer uso del Eclipse [49] como herramienta de desarrollo. Eclipse es un IDE (Entorno de Desarrollo Integrado) que en un principio fue diseñado y desarrollado por IBM y que luego fue lanzada a la comunidad de software libre. Presenta un entorno de desarrollo integrado en perspectivas personalizables. Las perspectivas son combinaciones formadas por vistas y editores que muestran los diversos aspectos de los recursos del proyecto y están organizados por el rol o la tarea del desarrollador. Las más grandes ventajas de Eclipse son:

- Basado en plugins: Puedes conseguir cientos de plugins para programar en C++, Perl, PHP, XML y pare de contar.. Y no solo para programar sino tambien para modelar en UML.
- Permite el desarrollo en equipo a través de CVS (Concurrent Version System) y a través de SVN con su respectivo plugin instalado
- IDE basado en archivos: Todo el contenido esta almacenado en archivos. Los recursos como clases java, archivos HTML, archivos XML(descriptores de despliegue), están almacenados en un sistema de archivo y así puedes obtener fácilmente acceso a ellos.

Para el desarrollo de los servicios Grid se hará uso del plugin MAGE-GDT[14], el cual nos abstraerá de todo el funcionamiento del servicio Grid, generando las clases “stubs” y otras, lo que permite concentrarse en la lógica del servicio.

1.10 Conclusiones

Para dar solución al problema en cuestión y haciendo uso de la metodología y herramientas vistas en este capitulo se desarrollarán un conjunto de servicios Grid que faciliten a los usuarios finales la interacción con la plataforma de cómputo distribuido Tarenal a través del Portal de Servicios del Grupo de Bioinformática.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

En el presente capítulo se da una breve descripción del sistema a desarrollar, de los requisitos funcionales y no funcionales del mismo, y de sus actores. Se presenta el diagrama de casos de uso del sistema, detallando textualmente cada uno de ellos.

2.1 Breve descripción del sistema

El sistema a desarrollar en el presente trabajo permitirá integrar varias plataformas de cómputo distribuido Tarenal en una única infraestructura basada en la tecnología Grid. Los usuarios podrán interactuar con dicha infraestructura a través del Portal de Servicios del Grupo de Bioinformática.

2.3 Definición de los requerimientos funcionales

Los requerimientos funcionales son capacidades o condiciones que un sistema debe cumplir. En nuestro sistema se identificaron los siguientes:

- **RF 1-)** Buscar problemas en servidores Tarenal.
- **RF 2-)** Gestionar ejecuciones
  - **RF 2.1-)** Crear una ejecución.
  - **RF 2.2 -)** Manipular ejecución.
  - **RF 2.2.1-)** Conocer el estado de una ejecución.
  - **RF 2.2.2-)** Conocer información de una ejecución.
  - **RF 2.2.3-)** Pausar una ejecución.
  - **RF 2.2.4-)** Reanudar una ejecución.
  - **RF 2.2.5-)** Finalizar una ejecución.
- **RF 3-)** Obtener resultados de las ejecuciones.
- **RF4-)** Guardar resultados de las ejecuciones.

2.4 Definición de los requisitos no funcionales

Los requerimientos no funcionales son las propiedades o cualidades que debe tener el sistema y que deben hacer que el mismo sea atractivo, fácil de usar y seguro, con el objetivo de brindarle al usuario final un software que además de ofrecerle todas las funcionalidades que necesite (Requisitos Funcionales), le sea agradable al usar y le brinde un nivel de confianza y seguridad, logrando así un

mayor nivel de aceptación. Estos requerimientos son divididos, según su clasificación en las siguientes categorías:

Software

- El middleware Globus Toolkit 4.0.5 con todos sus componentes básicos, incluyendo MyProxy Server deberá ser desplegado sobre la máquina virtual de java 1.5.
- El contenedor de portlets Gridsphere 2.2.10, con el proyecto GridPortlets 1.4 se deberá desplegar sobre el servidor Web Apache Tomcat 5.5.x.
- Se deberá instalar al menos una Plataforma de Cómputo Distribuido Tarenal.

Hardware

- La memoria RAM requerida es de 256 MB, aunque se recomienda 512 o superior.
- Procesadores Pentium IV.

Restricciones en el diseño y la implementación

- El lenguaje de programación en el cual se desarrollará la infraestructura es java.
- Se requiere el estándar WS-RF para el desarrollo de los servicios Grid.
- Se utilizará el plugin de Eclipse MAGE GDT para el desarrollo de los servicios Grid.
- Visual Paradigm debe ser usado como herramienta CASE.
- Globus Toolkit debe ser usado como Grid Middleware.
- La arquitectura a usar para la infraestructura debe ser SOA.

Seguridad

- El usuario necesita autenticarse para poder acceder a la infraestructura Grid.
- La información correspondiente a un usuario solo pueden ser accedida por él, manteniendo así la confidencialidad de la misma. Además la información de la autenticación de los usuarios es altamente confidencial.
- La información se encuentra siempre disponible, o sea, el usuario una vez autenticado puede acceder cada vez que lo desee a toda su información.

Apariencia o interfaz externa

- La interfaz debe ser agradable, sencilla y fácil de usar para lograr que el usuario acostumbre rápidamente a la aplicación.

Usabilidad

- El diseño de la aplicación está centrado en hacer transparente al usuario del portal Web la forma de acceder al Sistema de Cómputo Distribuido Tarenal a través de la Grid, por lo que el usuario solo necesita conocimientos básicos de informática para hacer uso de la misma.

Soporte

- Se realizarán pruebas al software una vez concluido el mismo para comprobar si todo funciona correctamente.
- Se prestarán servicios de mantenimiento a la infraestructura.

Portabilidad

- La infraestructura se podrá instalar en cualquier distribución de Linux siempre y cuando haya sido instalado el middleware Globus Toolkit 4.0.5 con la máquina virtual de java 1.5 o superior.

Confiabilidad

- La infraestructura debe poder repararse de manera eficiente en el caso de ocurrir fallas, el proceso de reparación no debe exceder las 48 horas en casos normales.
- En caso de fallo la pérdida de información debe ser muy poca o ninguna, según el tipo de fallo.

2.5 Actores del sistema

Los actores representan terceros fuera del sistema que interactúan con este. En nuestro sistema podemos encontrar los siguientes actores:

Actor	Justificación
Usuario	Es la persona que interactúa con la infraestructura a través del Portal de Servicios del Grupo de Bioinformática. Es el interesado en conocer los problemas de la plataforma Tarenal que se pueden ejecutar. Crea y manipula las ejecuciones de los problemas
Reloj	Es el encargado de automatizar el proceso de guardar los resultados de las ejecuciones.

Tabla 1. Actores del sistema

2.6 Diagrama de casos de uso del sistema

La forma en que los actores usan un sistema es representada a través de los casos de uso, los cuales van a ser fragmentos de funcionalidad que ofrece el sistema para aportar un resultado de valor para sus actores, y especifican una secuencia de acciones que el sistema debe llevar a cabo.

Orden.	Nombre del Casos de Uso	Justificación
1	Buscar problema	Necesario para conocer inicialmente cuales son los problemas de la Plataforma de Cómputo Distribuido Tarenal que podemos ejecutar a través de la infraestructura Grid
2	Gestionar ejecución	Necesario para manipular y conocer información de las ejecuciones de los problemas de la plataforma Tarenal.
3	Obtener resultado	Necesario para obtener los resultados de las ejecuciones terminadas.
4	Guardar resultado	Necesario para guardar los resultados de las ejecuciones en el servidor GridFTP designado a este fin.

Tabla 2. Casos de uso del sistema

Un diagrama de casos de uso del sistema contiene actores, casos de uso del sistema y las relaciones existente entre los mismo. En la siguiente figura se muestra el diagrama de casos de uso del sistema.

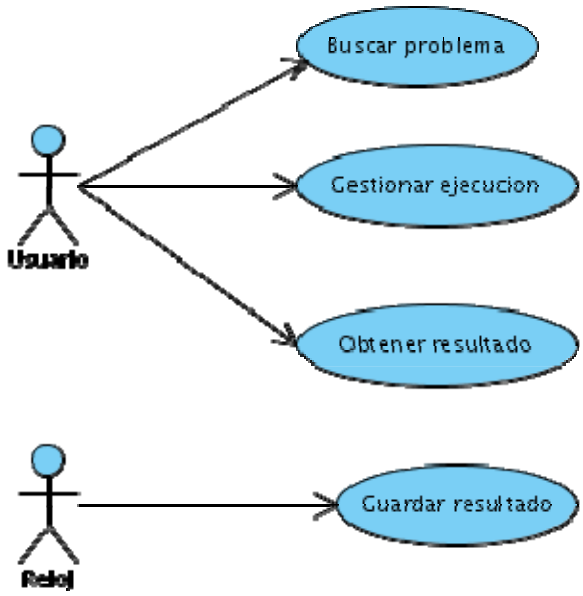

Fig. 3. Diagrama de casos de uso del sistema

2.7 Descripción textual de los casos de uso del sistema

Caso de Uso:	Buscar problema
Actores:	Usuario(Inicia)
Propósito:	Conocer los problemas que pueden ser ejecutados.
Resumen:	El caso de uso inicia cuando el usuario va a realizar la operación: **Buscar problema:** Cuando el usuario necesita hacer una búsqueda de los problemas existentes en uno o varios servidores de la Plataforma de Cómputo Distribuido Tarenal. El usuario selecciona el o los servidores e indica un criterio de búsqueda para los problemas. El sistema le muestra los problemas resultantes de la búsqueda, terminando así el CU.
Referencia:	RF1
Precondiciones:	El usuario debe estar autenticado y autorizado para usar los servicios Grid.
Pos condiciones:	Se muestran los problemas resultantes de la búsqueda
Curso Normal de los Eventos	

Acción del Actor	Respuesta del Sistema
➢ El usuario selecciona la acción buscar problemas.	➢ El sistema busca todos los servidores registrados.
	➢ El sistema muestra un listado con todos los servidores encontrados.
➢ El usuario escribe el nombre del problema que desea buscar y el o los servidores donde debe efectuarse la búsqueda.	➢ El sistema obtiene todos los problemas de los servidores seleccionados y realiza la búsqueda del problema introducido por el usuario, obteniendo los problemas asociados a dicha búsqueda.
	6- El sistema le muestra al usuario todos los problemas relacionados con su búsqueda, terminando así el CU.

Curso Alternativo de los Eventos	
Acción del Actor	**Respuesta del Sistema**
	3.1- En caso de no encontrarse ningún servidor se le informa al usuario, finalizando de esta forma al CU.
	5.1- En caso de no encontrar ningún problema el sistema notifica al usuario, terminando así el CU.
	5.2- En caso de que el usuario no haya seleccionado ningún servidor el sistema realizará la búsqueda sobre todos los problemas de todos los servidores.
	5.3- En caso de que el usuario no haya escrito el nombre del problema a buscar el sistema obtendrá como resultado de la búsqueda todos los problemas encontrados en los

	servidores seleccionados.
Prioridad:	Crítico

Tabla 3. Descripción del caso de uso Buscar Problema

Caso de Uso:	**Gestionar ejecución**
Actores:	Usuario(Inicia)
Propósito:	Manipular las ejecuciones de los problemas que se ejecutaron en la plataforma a través de la Grid.
Resumen:	El caso de uso inicia cuando el usuario va a realizar alguna de las siguientes operaciones: **Crear una ejecución:** Cuando el usuario desea crear una nueva ejecución de un problema de la Plataforma de Cómputo Distribuido Tarenal. El usuario indica el servidor Tarenal, el problema que desea ejecutar, el lugar donde se encuentran los ficheros necesarios para crear la ejecución. El sistema intentará llevar a cabo esta operación, notificando al usuario si fue posible o no realizarla, finalizando así el CU. **Conocer el estado de una ejecución:** Cuando el usuario desea conocer el estado de una ejecución que se encuentra en la plataforma y que fue iniciada a través de la Grid. El usuario indica el servidor Tarenal y el identificador de la ejecución de la cual desea conocer el estado y el sistema le muestra dicha información, finalizando de esta manera el CU. **Conocer información de una ejecución:** Cuando el usuario desea conocer información de una ejecución que se encuentra en la plataforma y que fue iniciada a través de la Grid. El usuario indica el servidor Tarenal y el identificador de la ejecución de la cual desea

	conocer la información y el sistema le muestra dicha información, finalizando de esta manera el CU. **Pausar una ejecución:** Cuando un usuario desea detener por un periodo de tiempo una ejecución que se encuentra en la plataforma y que fue iniciada a través de la Grid. El usuario indica el servidor Tarenal y el identificador de la ejecución que desea pausar, el sistema intentará pausar la ejecución y notificará al usuario si fue posible o no realizar la dicha operación, finalizando así el CU. **Reanudar una ejecución:** Cuando un usuario desea reanudar una ejecución que se encuentra en la plataforma, que fue iniciada a través de la Grid y que se encuentra actualmente pausada. El usuario indica el servidor Tarenal y el identificador de la ejecución que desea reanudar, el sistema intentará reanudar la ejecución y notificará al usuario si fue posible o no realizar la operación, finalizando así el CU. **Finalizar una ejecución:** Cuando un usuario desea finalizar una ejecución que se encuentra en la plataforma y que fue iniciada a través de la Grid. El usuario indica el servidor Tarenal y el identificador de la ejecución que desea finalizar, el sistema intentará finalizar la ejecución y notificará al usuario si fue posible o no realizar la dicha operación, finalizando así el CU.
Referencia:	RF2, RF2.1, RF2.2, RF2.2.1, RF2.2.2, RF2.2.3, RF2.2.4, RF2.2.5
Precondiciones:	El usuario debe estar autenticado y autorizado dentro de la Grid.
Pos condiciones:	En dependencia de la operación seleccionada por el usuario se crea una ejecución de un problema, se muestra la información , se muestra el estado, se detiene, se pausa o se reanuda una ejecución.

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1- El usuario selecciona cual operación desea realizar:	2- El sistema en dependencia de la operación seleccionada por el usuario realizará una de las

a) Crear una ejecución de un problema. **b) Manipular una ejecución.**	siguientes acciones: ➢ Crear una nueva ejecución: ir a la sección **"Crear nueva ejecución"**. ➢ Manipular ejecución: ir a la sección **"Manipular ejecución"**.
Sección "Crear nueva ejecución": Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1- El usuario indica el problema del cual desea crear la ejecución, el servidor Tarenal donde se encuentra, y los ficheros necesarios para la misma.	2- El sistema crea la ejecución, terminando así el CU.
Sección "Crear nueva ejecución": Curso Alternativo de los Eventos	
Acción del Actor	**Respuesta del Sistema**
	2.1- En caso de no poderse crear la ejecución el sistema notifica al usuario, finalizando así el CU.
Sección "Manipular ejecución": Flujo Normal de Eventos	
	1- El sistema busca en los servidores Tarenal registrado todas las ejecuciones creadas por el usuario.
	2- El sistema le muestra al usuario un listado con todas las ejecuciones obtenidas.
3- El usuario selecciona la ejecución que desea manipular.	

4- El usuario selecciona una de las siguientes operaciones: **a) Conocer el estado de la ejecución.** **b) Conocer información de la ejecución.** **c) Pausar la ejecución.** **d) Reanudar la ejecución.** **e) Finalizar la ejecución.**	5- El sistema dependiendo de la operación seleccionad por el usuario realizará una de las siguientes acciones: 1- Mostrar estado de la ejecución seleccionada: ir a la sección **"Mostrar estado de la ejecución"**. 2- Mostrar información de la ejecución seleccionada: ir a la sección **"Mostrar información de la ejecución"**. 3- Pausar la ejecución seleccionada: ir a la sección **"Pausar ejecución"**. 4- Reanudar la ejecución seleccionada: ir a la sección **"Reanudar ejecución"**. 5- Finalizar la ejecución seleccionada: ir a la sección **"Finalizar ejecución"**.
Sección "Manipular ejecución": Curso Alternativo de los Eventos	
Acción del Actor	**Respuesta del Sistema**
	1.1- Si el sistema no encuentra ninguna ejecución que haya sido creada por el usuario notifica al mismo, terminando así el CU.
Sección "Mostrar estado de la ejecución": Curso Normal de los Eventos	
Acción del Actor	**Respuesta del Sistema**
	1- El sistema le muestra al usuario el estado de la ejecución seleccionada, terminando así el CU.
Sección "Mostrar estado de la ejecución": Curso Alternativo de los Eventos	
Acción del Actor	**Respuesta del Sistema**
	1.1- En caso de ocurrir algún error el sistema lo notifica usuario, finalizando de esta forma el CU.
Sección "Crear información de la ejecución": Curso Normal de los Eventos	
Acción del Actor	**Respuesta del Sistema**

	1- El sistema le muestra al usuario la información correspondiente a la ejecución seleccionada, terminando así el CU.
Sección "Mostrar información de la ejecución": Curso Alternativo de los Eventos	
Acción del Actor	**Respuesta del Sistema**
	1.1- En caso de ocurrir algún error el sistema lo notifica usuario, finalizando de esta forma el CU.
Sección "Pausar ejecución": Curso Normal de los Eventos	
Acción del Actor	**Respuesta del Sistema**
	1- El sistema pausa la ejecución seleccionada por el usuario.
	2- El sistema informa al usuario que la tarea ha sido completada, terminando así el CU.
Sección "Pausar ejecución": Curso Alternativo de los Eventos	
Acción del Actor	**Respuesta del Sistema**
	1.1- En caso de ocurrir algún error el sistema lo notifica usuario, finalizando de esta forma el CU.
Sección "Reanudar ejecución": Curso Normal de los Eventos	
Acción del Actor	**Respuesta del Sistema**
	1- El sistema reanuda la ejecución seleccionada por el usuario.
	2- El sistema informa al usuario que la tarea ha sido completada, terminando así el CU.
Sección "Reanudar ejecución": Curso Alternativo de los Eventos	
Acción del Actor	**Respuesta del Sistema**
	1.1- En caso de ocurrir algún error el sistema lo notifica usuario, finalizando de esta forma el CU.
Sección "Finalizar ejecución": Curso Normal de los Eventos	
Acción del Actor	**Respuesta del Sistema**
	1- El sistema finaliza la ejecución seleccionada.

	2- El sistema informa al usuario que la tarea ha sido completada, terminando así el CU.
Sección "Finalizar ejecución": Curso Alternativo de los Eventos	
Acción del Actor	**Respuesta del Sistema**
	1.1- En caso de ocurrir algún error el sistema lo notifica usuario, finalizando de esta forma el CU.
Prioridad:	Crítico

Tabla 4. Descripción del caso de uso Gestionar Ejecución

Caso de Uso:	**Obtener resultado**
Actores:	Usuario(Inicia)
Propósito:	Obtener los resultados de las ejecuciones que se realizaron en las plataformas de cómputo distribuido Tarenal a través de la Grid.
Resumen:	El caso de uso se inicia cuando el usuario solicita al sistema obtener los resultados de las ejecuciones, el sistema le muestra al usuario los resultados de sus ejecuciones terminadas. Hecho esto el usuario puede realizar la descarga de los resultados, terminando así el CU.
Referencia:	RF3
Precondiciones:	El usuario debe estar autenticado y autorizado dentro de la Grid.
Pos condiciones:	El usuario obtiene los resultados de sus ejecuciones.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1- El usuario selecciona la operación **"Obtener resultados"**.	2- El sistema busca en los servidores Tarenal registrados los resultados de las ejecuciones terminadas del usuario.
	3- El sistema busca los resultados de las ejecuciones terminadas del usuario en el servidor gridFTP designado para almacenar las soluciones.

	4- El sistema crea el listado de los resultados obtenidos.
	5- El sistema le muestra al usuario el listado con los resultados obtenidos.
6- El usuario selecciona el o los resultados que desea obtener, terminando así el CU.	

Curso Alternativo de los Eventos	
Acción del Actor	**Respuesta del Sistema**
	4.1- En caso de no existir resultados para crear el listado el sistema notifica al usuario, terminando así el CU.
Prioridad:	Crítico

Tabla 5. Descripción del caso de uso Obtener Resultado

Caso de Uso:	**Guardar Resultado.**
Actores:	Usuario(Inicia)
Propósito:	Guardar los resultados de las ejecuciones terminadas para el servidor gridFTP designado.
Resumen:	El caso de uso se inicia cuando el reloj, a una hora determinada solicita al sistema guardar los resultados de las ejecuciones, el sistema copia todos los resultados de todos los servidores Tarenal para el servidor gridFTP designado, eliminando los resultados del servidor Tarenal a medida que son copiados. Una vez copiados todos los resultados termina el CU.
Referencia:	RF4
Precondiciones:	La hora del sistema debe coincidir con la hora planificada para la iniciación del CU.

Pos condiciones:	Los resultados serán guardados para el servidor gridFTP designado.

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1- El reloj dispara la acción “Guardar resultados”.	2- El sistema selecciona un servidor Tarenal registrado que no haya sido seleccionado anteriormente en este proceso.
	3- El sistema copia un resultado desde un Servidor Tarenal seleccionado para el servidor gridFTP designado.
	4- El sistema elimina el resultado copiado del servidor Tarenal seleccionado.

Curso Alternativo de los Eventos	
Acción del Actor	**Respuesta del Sistema**
	2.1- Si no queda ningún servidor Tarenal registrado por seleccionar culmina el CU.
	3.1- Si no queda ningún resultado que copiar en el servidor Tarenal seleccionado se vuelve al paso 2.

Prioridad:	Crítico

Tabla 6. Descripción del caso de uso Guardar Resultado

2.8 Conclusiones

En este capítulo se identificaron lo requisitos funcionales y no funcionales de la infraestructura, además se realizó el diagrama de los casos de uso del sistema y la descripción de los mismos.

CAPÍTULO 3.DISEÑO DEL SISTEMA

En este capítulo se describen brevemente los patrones utilizados y como se evidencia su utilización en el desarrollo de la infraestructura. Además se presentan los diagramas de clases del diseño que darán solución a los casos de uso identificados y descritos en el capítulo anterior y se realizarán los diagramas de interacción para los diferentes escenarios. Finalmente se presentará el diagrama de despliegue, con el objetivo de tener una idea más clara de como quedará distribuida la infraestructura una vez desplegada.

3.1 Patrones de desarrollo de software

Los patrones de desarrollo de software constituyen un conjunto de soluciones a problemas comunes. Generalmente estos problemas nos los encontramos una y otra vez en el desarrollo de aplicaciones de software, por lo que el uso de una solución aplicada anteriormente para resolver un problema parecido puede ser conveniente. En otras palabras, un patrón sería una solución probada para un problema ya conocido en un contexto dado, esta solución debe ser recurrente, logrando así que pueda ser utilizada en otras soluciones.

Con el paso del tiempo el uso de los patrones se ha ido intensificando ya que gracias a estos patrones podemos producir software más resistente a los cambios. Además permiten identificar con facilidad los problemas y la solución adecuada para los mismos, el código es reutilizable y la documentación es estándar. En términos más específicos facilitan la localización de los objetos que formarán el sistema, determinación de la granularidad adecuada y el aprendizaje y comunicación entre programadores.

3.1.1 Estilo Arquitectónico utilizado

Los patrones arquitectónicos están orientados al diseño de alto nivel. Están dirigidos a aspectos fundamentales de la estructura de un software. Especifican un conjunto predefinido de subsistemas con sus responsabilidades y un grupo de recomendaciones para organizar los distintos componentes.

Para el desarrollo de la infraestructura se implementaron un conjunto de servicios Grid haciendo uso de SOA (Ver epígrafe 1.7.2), estos servicios se encuentran divididos en dos grupos:

- Servicios específicos: Son los que interactúan directamente con la Plataforma de Cómputo Distribuido Tarenal.
- Servicios generales: Son los que hacen uso de los servicios específicos, y los que son usado a través del Portal Se Servicios del Grupo de Bioinformática.

Según lo visto en 1.7.2 en nuestra infraestructura Grid se identifican como proveedores de servicios las computadoras que tienen disponibles los servicios específicos (Figura 14). Estos servicios son registrados en el Index Service [43], que hace el papel de registro de servicios. Los solicitantes son los servicios generales quienes usan al Index Service para descubrir a los específicos y conocer detalles de como usarlos.

Para acceder a la infraestructura a través del Portal de Servicios del Grupo de Bioinformática se desarrolló una aplicación Web basada en portlets, que son componentes Web controlados por un contenedor [52]. Esta aplicación hace uso de los servicios generales (Figura 14). Para el desarrollo de dicha aplicación se hizo uso del patrón arquitectónico Modelo Vista Controlador (MVC)

El Modelo Vista Controlador separa el software en tres partes fundamentales: la lógica de negocio (Modelo), la interfaz que se brinda al usuario con toda la información que necesita (Vista) y una parte encargada de recibir e interpretar las acciones del usuario, actuando sobre el modelo y la vista (Controlador). Cada una de estas partes tiene funciones específicas dentro del software:

- Modelo:
    - Es el responsable de acceder a la capa de almacenamiento de datos.
    - Define las reglas de negocio.
    - Lleva un registro de las vistas y controladores del sistema.
    - Notifica a las vistas los cambios producidos por un agente externo.
- Controlador:
    - Recibe los eventos de entrada.
    - Contiene reglas de gestión de eventos.
- Vista:

- Recibe datos del modelo y los muestra al usuario.
- Tiene un registro de su controlador asociado.

Para el uso de este patrón la aplicación se estructuró por capas de la siguiente forma:

Modelo:

En esta capa se encuentran distribuidas todas las clases relacionadas con los servicios Grid y sus respectivos clientes, además de un conjunto de clases básicas necesarias para el funcionamiento del sistema.

Controlador:

En esta capa se encuentran agrupados los servlets, que son las clases encargadas de recibir las diferentes peticiones de los usuarios y hacer llamadas a las funciones necesarias para dar cumplimientos a estas peticiones.

Vista:

Aquí podemos encontrar los portlets con todos sus componentes, que son las clases con las que interactúa directamente el usuario desde el portal de servicio.

3.1.2 Patrones utilizados para el diseño

Los patrones de diseño son clasificados como patrones estructurales para el diseño de bajo nivel. Se ocupan de aspectos relacionados con los subsistemas, por lo que se centran en aspectos específicos.

Para el desarrollo del sistema se usaron fundamentalmente los patrones GRASP (General Responsibility Assignment Software Patterns), que describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades.

Las responsabilidades están relacionadas con las obligaciones de un objeto en cuanto a su comportamiento. Estas responsabilidades son básicamente de dos tipos:

- Conocer:
  - Conocer los datos privados encapsulados.
  - Conocer los objetos relacionados.

- Conocer las cosas que puede derivar o calcular.

- Hacer:
  - Hacer algo él mismo, como crear un objeto o hacer un cálculo.
  - Iniciar una acción en otros objetos.
  - Controlar y coordinar actividades en otros objetos.

De este grupo de patrones se hizo uso de los siguientes:

Experto: Constituye el principio básico de asignación de responsabilidades. Indica que la responsabilidad de la creación de un objeto debe ser de la clase que conoce toda la información necesaria para crearlo. Brinda como beneficio que se mantiene el encapsulamiento, los objetos utilizan su propia información para llevar a cabo sus tareas. Se distribuye el comportamiento entre las clases que contienen la información requerida, haciéndola más fáciles de entender y mantener.
El uso de este patrón en nuestro sistema se evidencia en la clase JDCSXmlWriter, a la cual se le dio la responsabilidad de codificar cualquier objeto de tipo JDCSElement en formato XML.

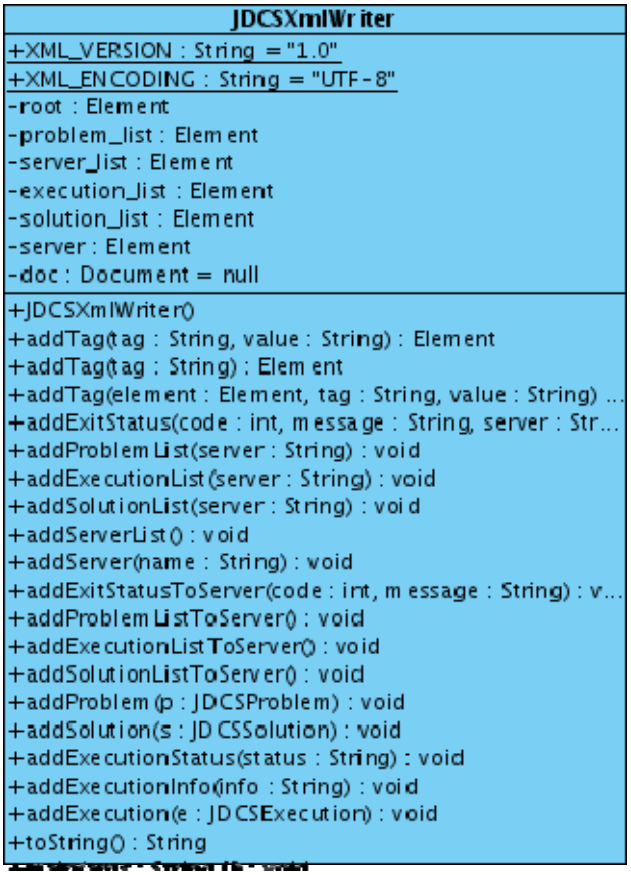

Figura 4. Diseño de la clase JDCSXmlWriter.

Creador: Ayuda a identificar quien debe ser responsable de la creación de nuevos objetos o instancias de clases. La nueva instancia deberá ser creada por aquellas clases que:

- Tiene la información necesaria para realizar la creación del objeto, o
- Usa directamente las instancias creadas del objeto, o
- Almacena o maneja varias instancias de la clase

El uso de este patrón se evidencia en la clase TarenalProblemServlet, la cual es la encargada de crear instancias de la clase JDCSGeneralProblemInformationClient

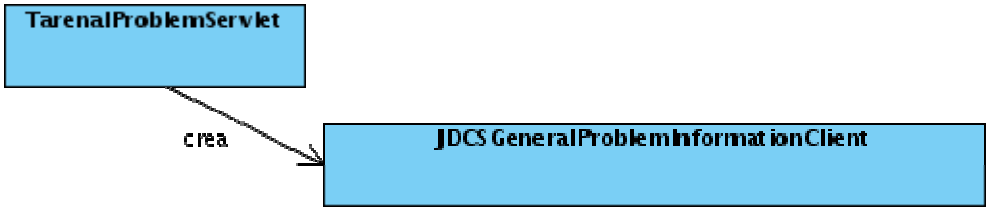

Figura 5. Relación entre las clases TarenalProblemServlet
y JDCSGeneralInformationClient

Controlador: Funciona como intermediario entre la interfaz de usuario y las clases que dan solución al problema, es la encargada de recibir los datos entrados por el usuario y enviarlos a las distintas clases según las funcionalidades implementadas.

Se puede evidenciar su uso en la clase TarenalProblemServlet, la cual recibe los datos entrados por el usuario para realizar una búsqueda de uno o varios problemas y se los envía a la clase JDCSGeneralProblemInformationClient.

3.2 Diagramas de clases del diseño

En cada diagrama de clase intervienen los servicios específicos y los generales, así como sus respectivos clientes.

3.2.1 Caso de uso Buscar Problema

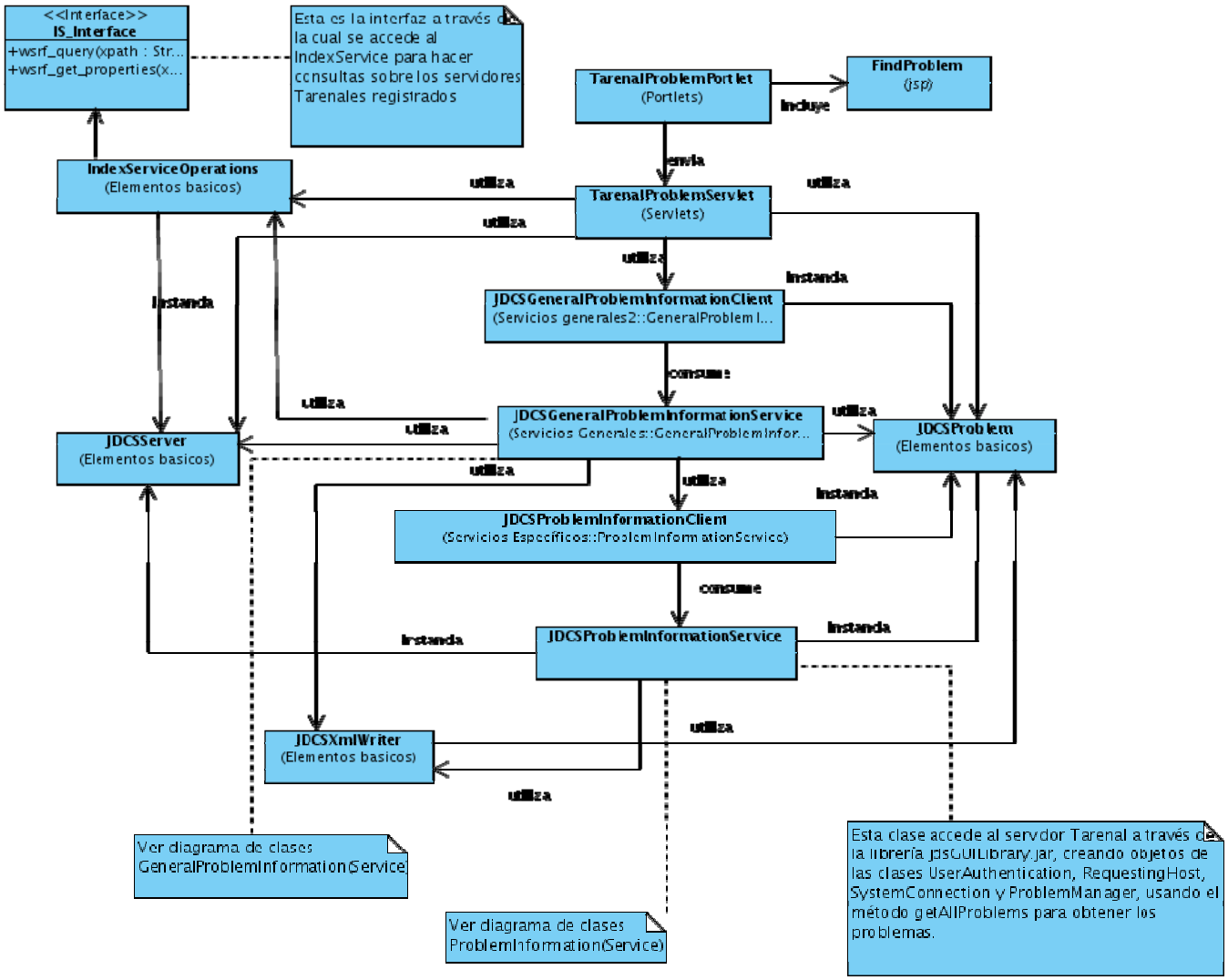

Figura 6. Diagrama de clases del diseño. Caso de uso Buscar Problema.

3.2.2 Caso de uso Gestionar Ejecución

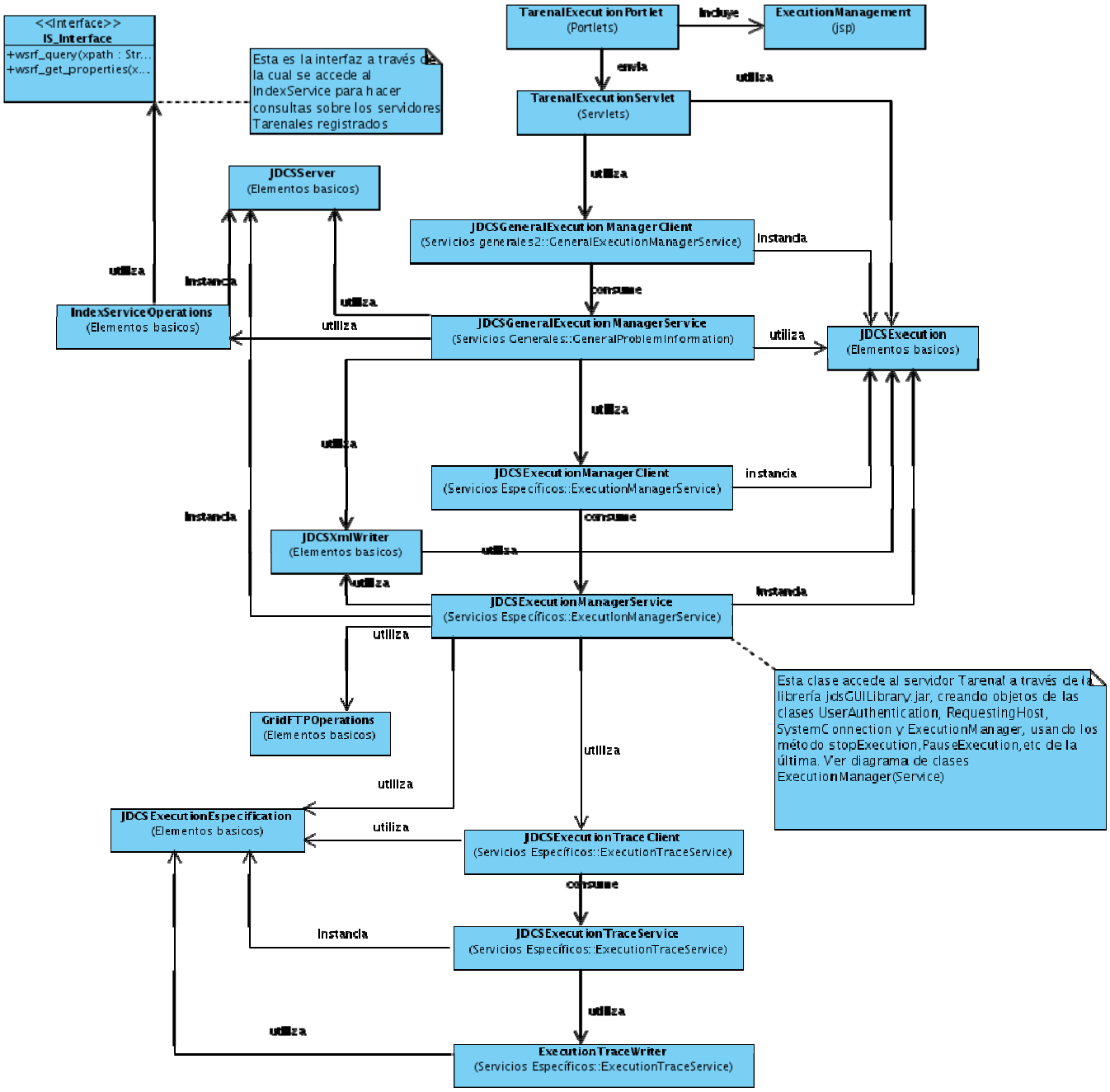

Figura 7. Diagrama de clases del diseño. Caso de uso Gestionar Ejecución.

3.2.3 Caso de uso Obtener Resultado

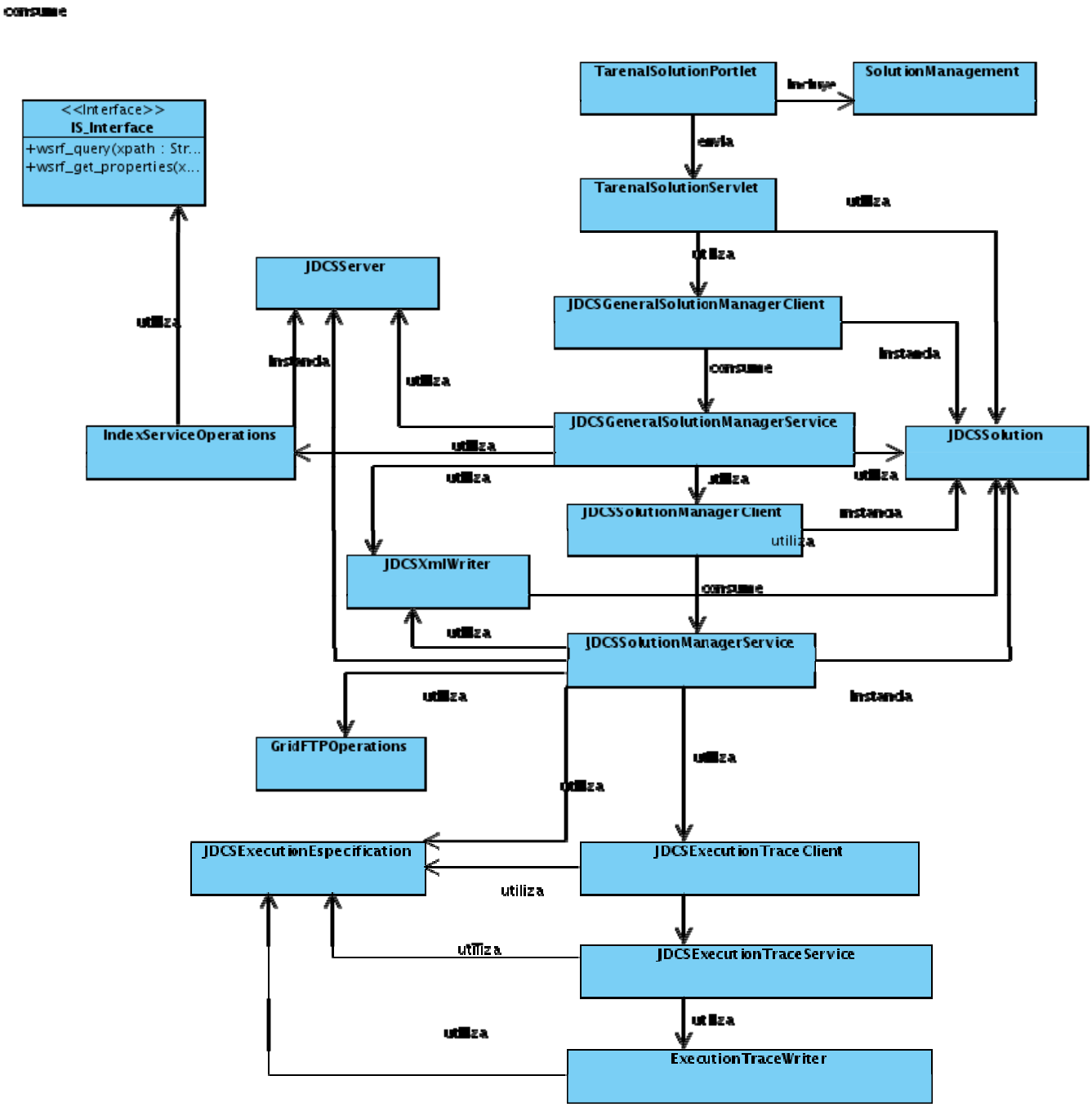

Figura 8. Diagrama de clases del diseño. Caso de uso Obtener Resultado.

3.2.4 Caso de uso Guardar Resultado

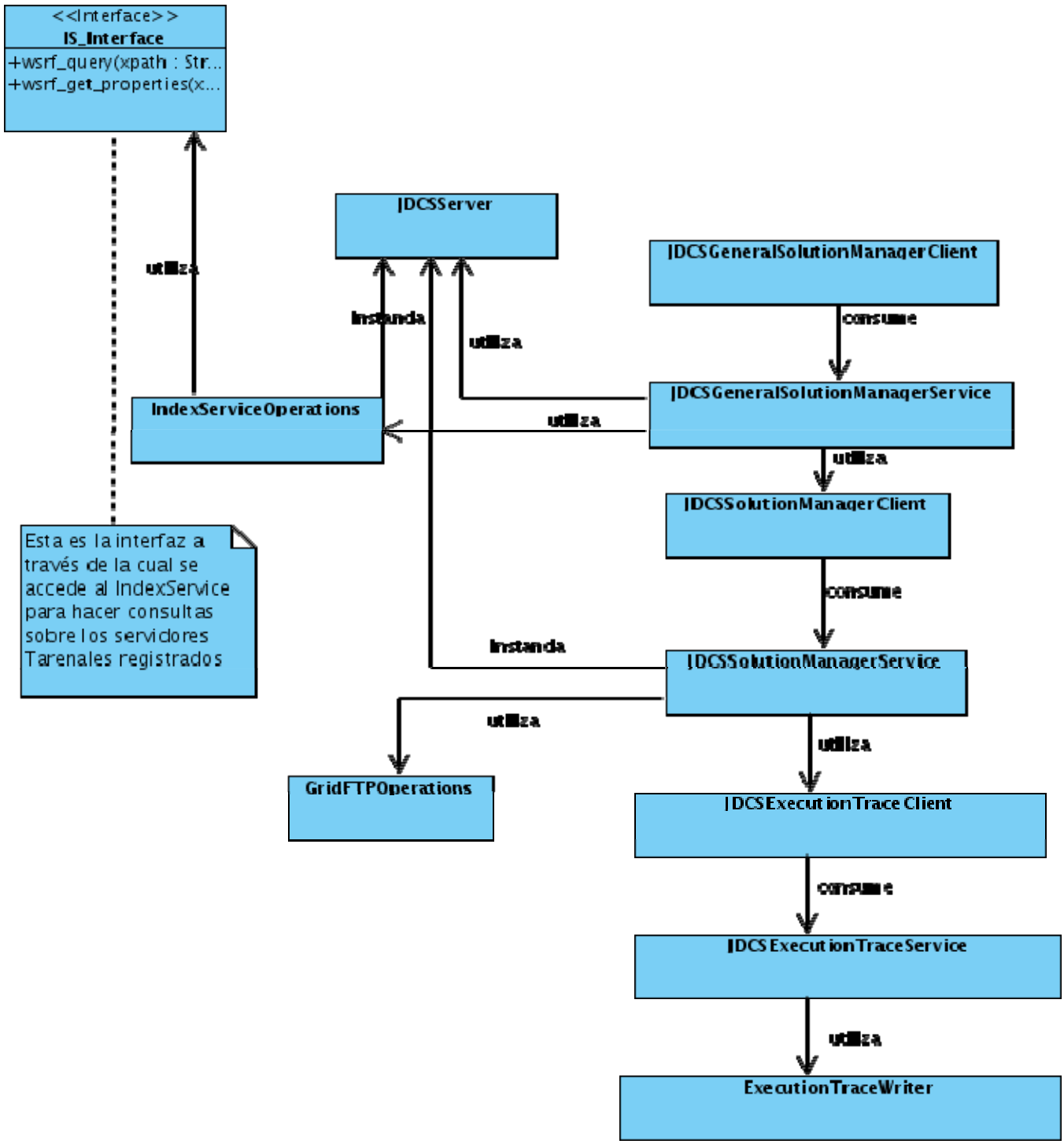

Figura 9. Diagrama de clases del diseño. Caso de uso Guardar Resultado.

3.3 Diagramas de secuencia

3.3.1 Caso de uso Buscar Problema

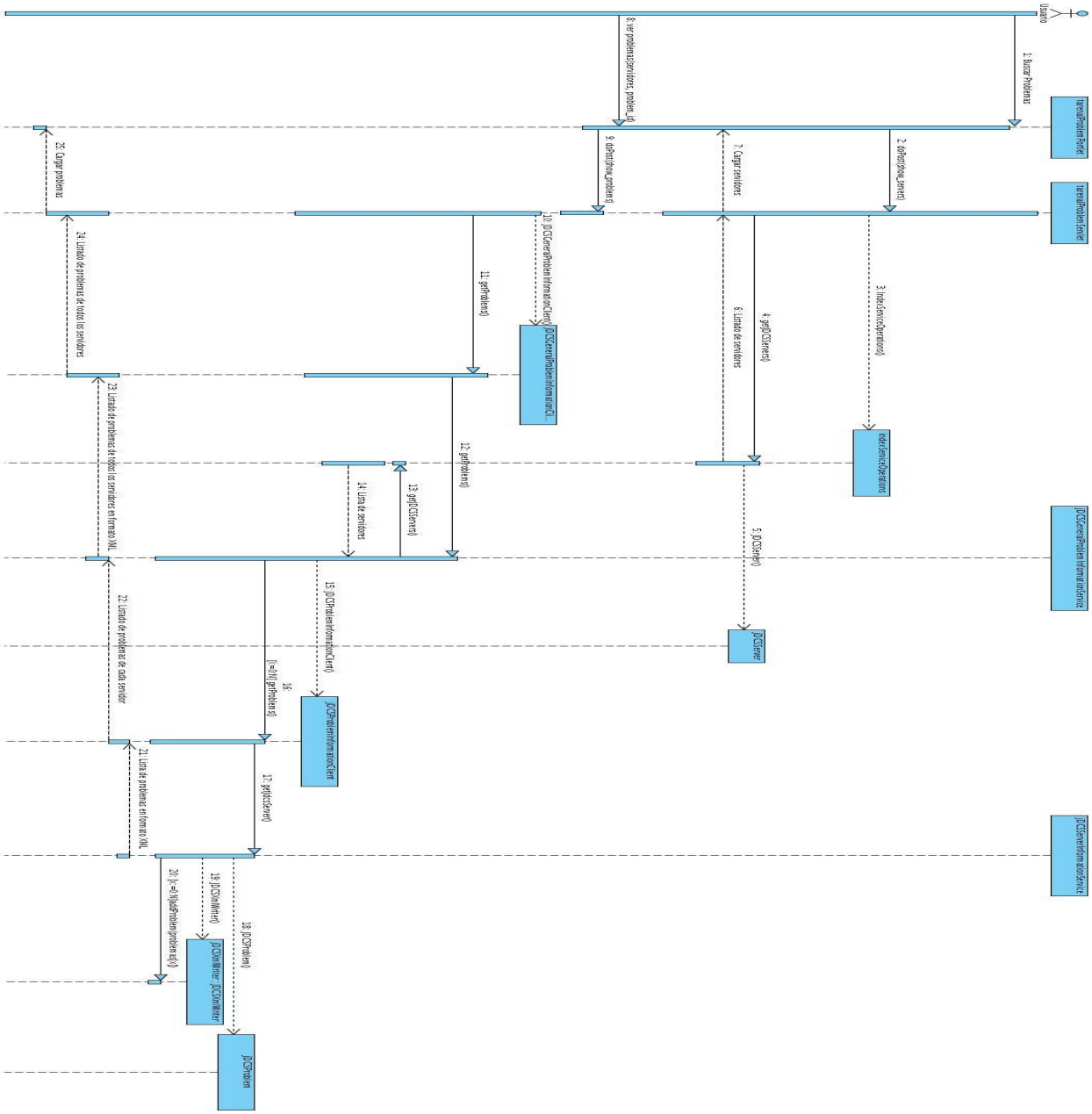

Figura 10. Diagrama de secuencia. Caso de uso Buscar Problema

3.3.2 Caso de uso Manipular Ejecución

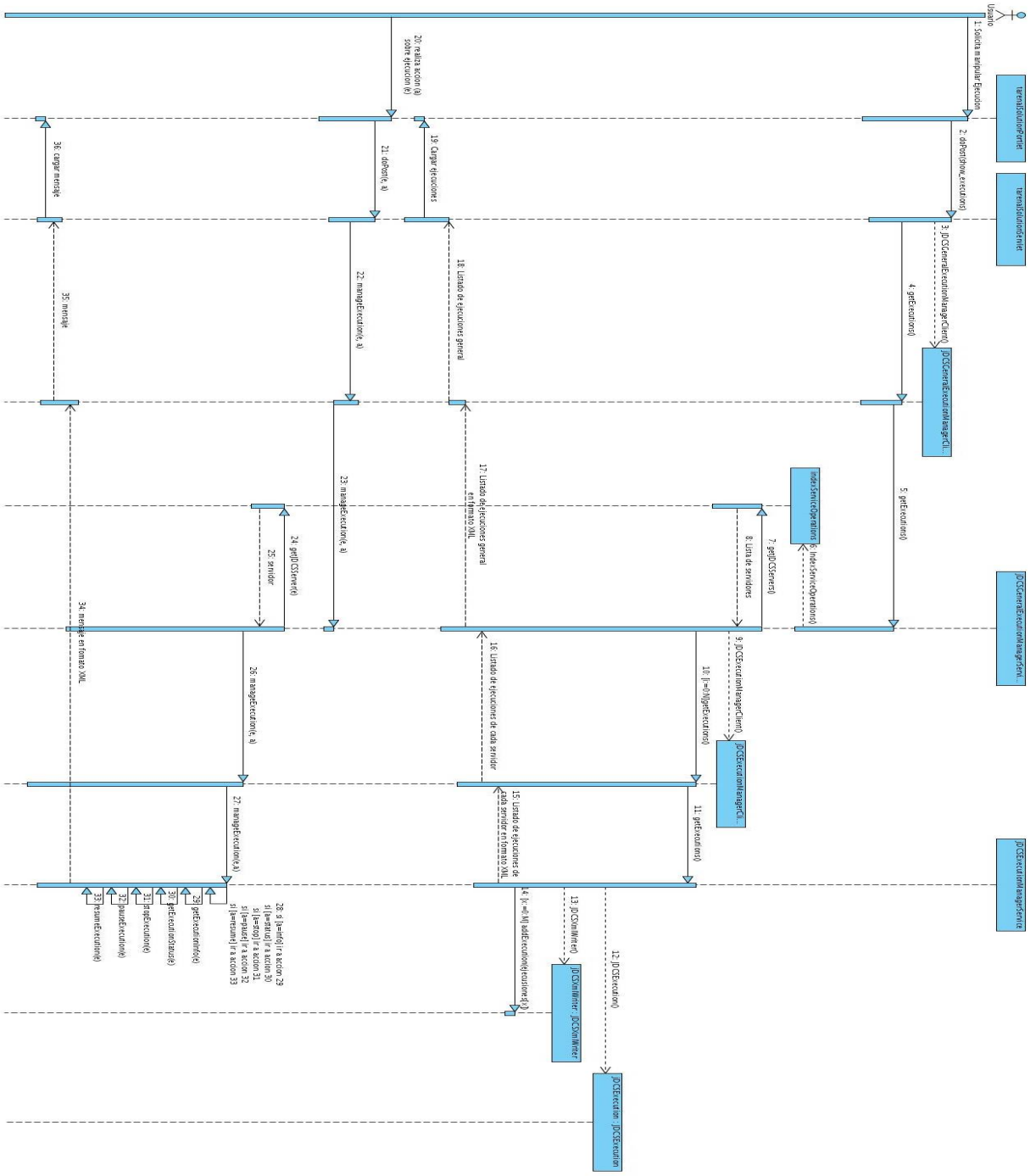

Figura 11. Diagrama de secuencia. Caso de uso Manipular Ejecución.

3.3.3 Caso de uso Obtener Resultado

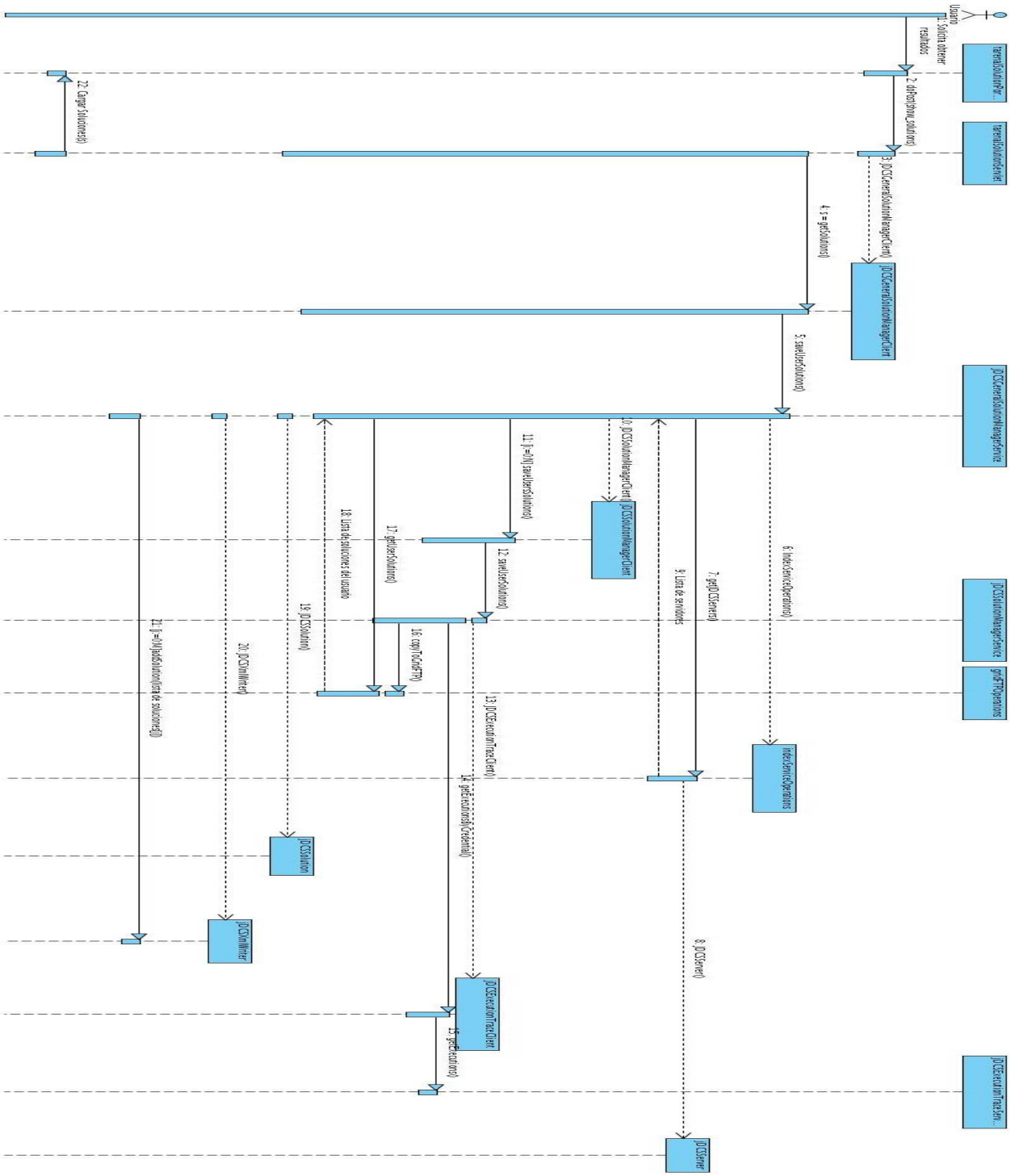

Figura 12. Diagrama de secuencia. Caso de uso Obtener Resultado.

3.3.1 Caso de uso Guardar Resultado

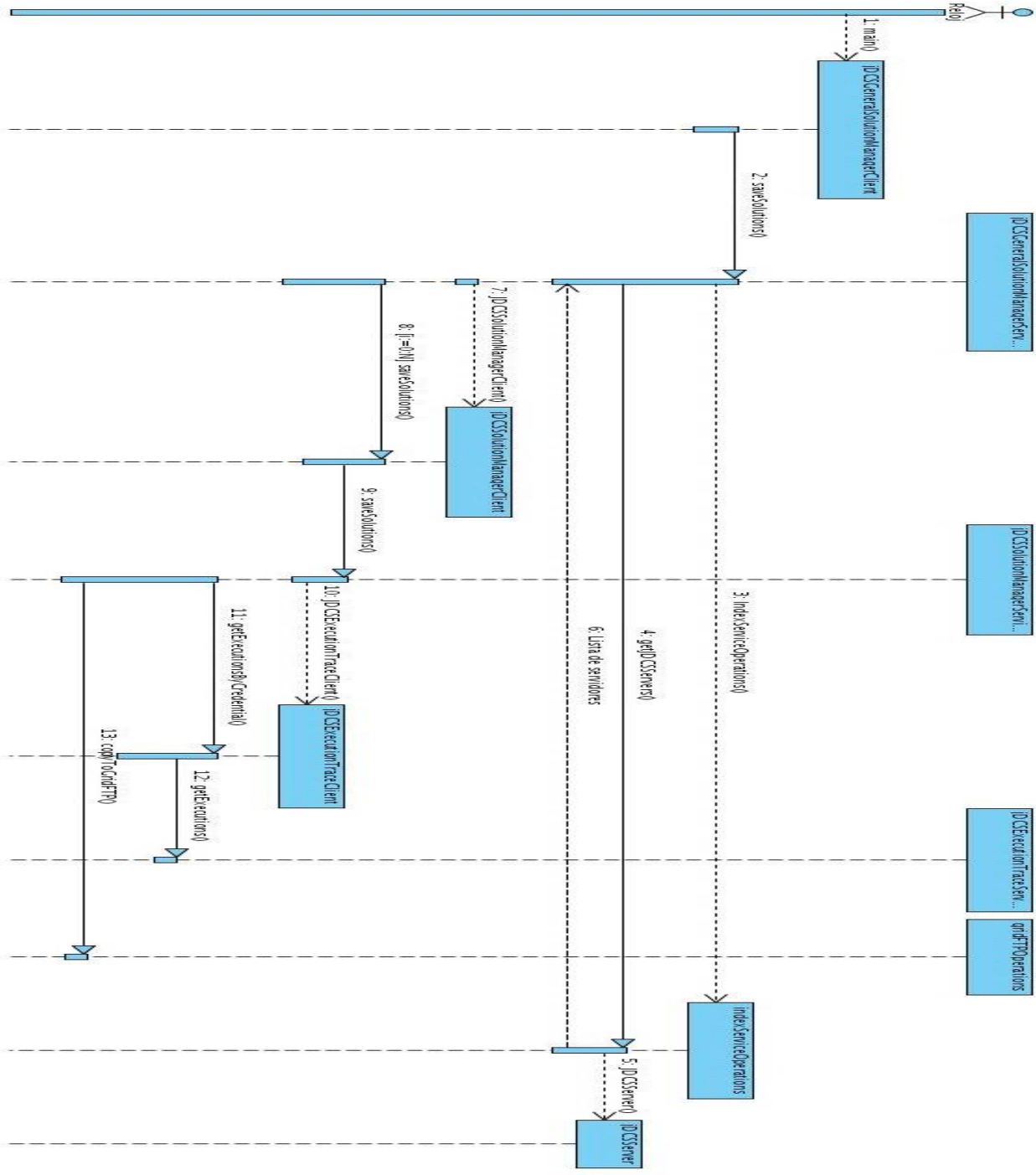

Figura 13. Diagrama de secuencia. Caso de uso Guardar Resultado.

3.4 Diagrama de despliegue

El diagrama de despliegue es un modelo de objetos que describe la distribución física del sistema, indicando cómo quedan distribuidas las funcionalidades del mismo entre los nodos de cómputo que utiliza. En la siguiente figura se muestra el diagrama despliegue del sistema desarrollado.

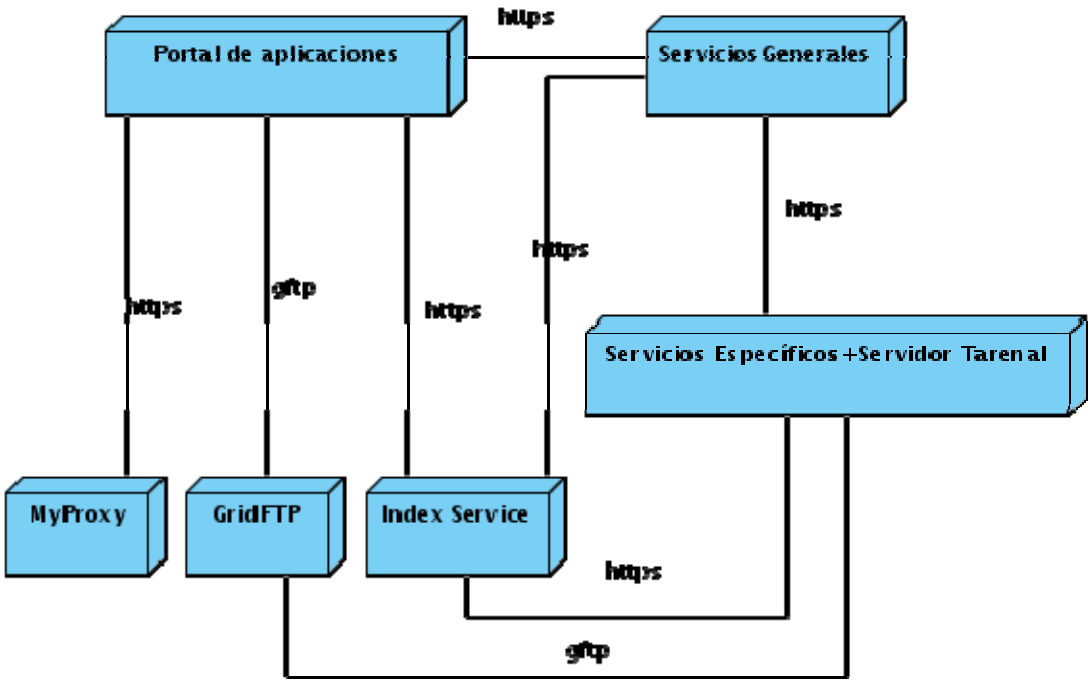

Figura 14. Diagrama de despliegue

En el portal de aplicaciones están montados los portlets que interactúan con los servicios generales.

Estos servicios generales usan el Index Service para descubrir los servicios específicos registrados en el mismo, y una vez descubierto pueden hacer uso de ellos. Los servicios específicos deben estar en el mismo nodo que se encuentra el servidor Tarenal. Estos servicios específicos interactúan con el servidor Tarenal, con el GridFTP para obtener y guardar ficheros, y con con Index Service para registrase en el mismo.

Desde el portal de aplicaciones el usuario además puede interactuar, a través de determinados portlets con el servidor MyProxy para obtener sus credenciales y con el servidor GridFTP para navegar

por el sistema de archivos del mismo. También desde este lugar se hacen consultas al Index Service para descubrir los servidores Tarenal disponibles e informarle al usuario dicha información.

3.5 Conclusiones

En el presente capítulo se identificaron los patrones de desarrollo usados para desarrollar la infraestructura. Además se presentaron los diagramas de clases del diseño para cada caso de uso del sistema y los diagramas de secuencia para cada escenario de los casos de uso. Finalmente se presentó el diagrama de despliegue del sistema.

CAPÍTULO 4: IMPLEMENTACIÓN DEL SISTEMA

En este capítulo se presenta el diagrama de componentes del sistema y los fragmentos de código fuente más relevantes.

4.1 Diagrama de componentes

En diagrama de componentes representa la separación de un sistema de software en componentes físicos y muestra las dependencias entre estos componentes. Estos componentes pueden ser simples archivos, paquetes, bibliotecas, etc. En la siguiente figura se muestra el diagrama de componentes del sistema desarrollado.

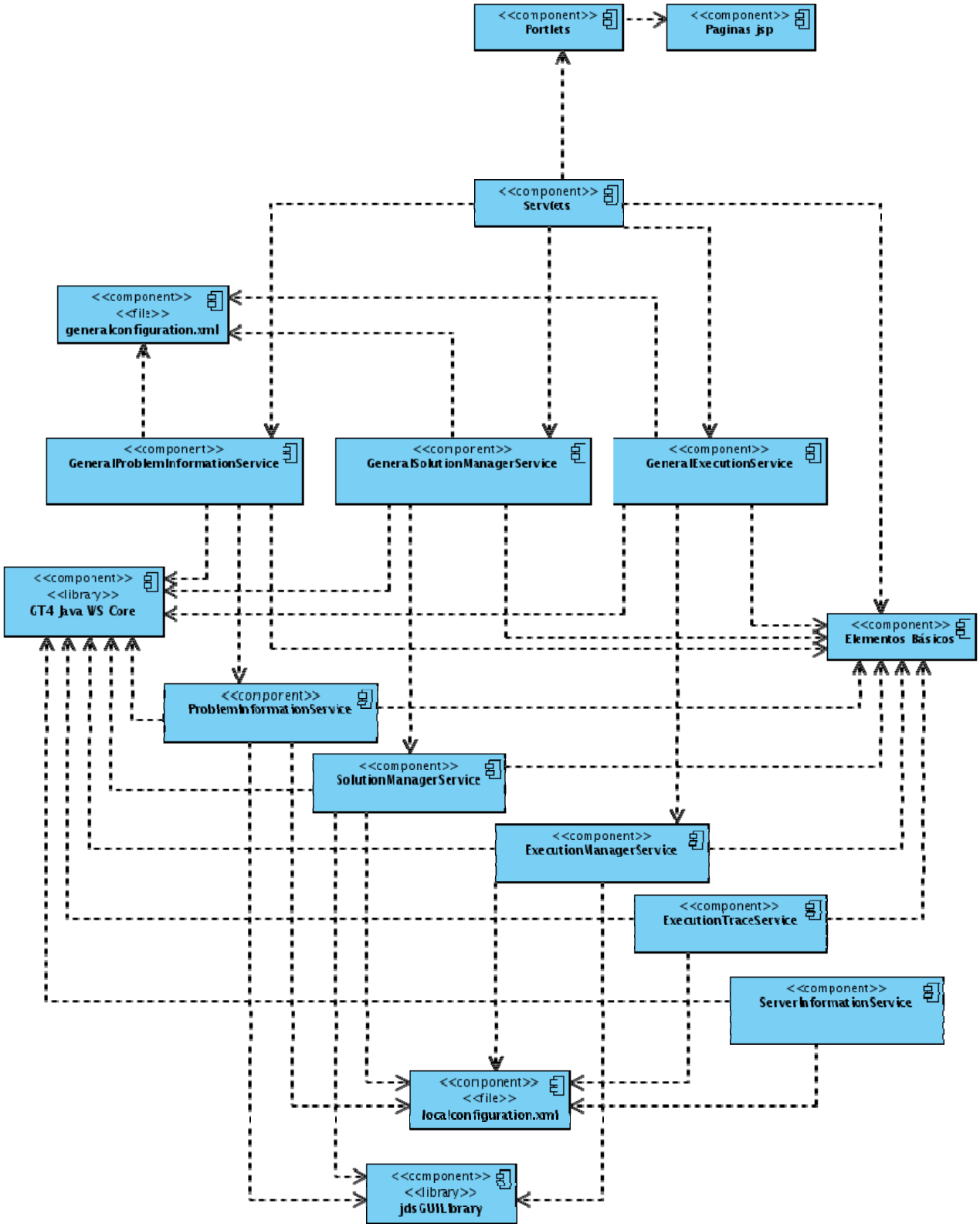

Figura 15. Diagrama de componentes

4.2 Código fuente

Para el desarrollo de la infraestructura se implementaron primeramente un conjunto de servicios Grid, a los que llamamos específicos. Estos servicios tienen como principal función interactuar con la Plataforma de Cómputo Distribuido Tarenal. Los servicios específicos desarrollados son:

- JDCSServerInformationService
- JDCSProblemInformationService
- JDCSExecutionTraceService
- JDCSExecutionManagerService
- JDCSSolutionManagerService

Cada uno de estos servicios definen un recurso el cual será el mismo para todos los clientes que lo invoquen. Este recurso tiene como propiedad o atributo una cadena de caracteres con formato XML. En el caso de JDCSExecutionTrace este XML representa la asociación entre la ejecución de un problema, la credenciales de su propietario y el problema ejecutado.

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
    <execution>
        <id>32</id>
        <credential>/O=Grid/OU=GlobusTest/OU=simpleCA-gt1/CN=System Manager</credential>
        <problem>biosys</problem>
    </execution>
</root>
```

Figura 16. Fichero XML usado por el JDCSExecutionTraceService

En el caso del resto de los servicios el XML contiene información necesaria para interactuar con la Plataforma de Cómputo Distribuido Tarenal y con el resto de los servicios específicos.

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
        <name>10.34.20.117</name>
        <rmi_port>5800</rmi_port>
        <socket_port>5801</socket_port>
        <user>root</user>
        <password>administrator</password>
        <home>/home/jds/server</home>
        <services_url>https://10.34.20.117:8443/wsrf/services/</services_url>
</root>
```

Figura 17. Fichero XML usado por los servicios específicos.

Cada servicio específicos inicializa sus recursos con el método initialize(), de la clase que representa a su recurso. Por ejemplo, en el caso de JDCSServerInformationService, su recurso es representado con la clase JDCSServerInformationResource. En este método es donde se lee el fichero XML para inicializar el atributo del recurso destinado con este fin.

```
public class JDCSServerInformationResource implements
        Resource, ResourceIdentifier, ResourceProperties, JDCSServerInformationNamespaces

{

        public void initialize() throws Exception
        {
        |...
                String a = "";
                Scanner scan = new Scanner(new File("/etc/tarenal/local_services_cfg.xml"));
                while(scan.hasNext())
                {
                        a+=scan.nextLine();
                }
                setJdcsServer(a);
        }
        ...
}
```

Figura 18. Fragmento de código donde se inicializa el atributo jdcsServer del recurso definido por el servicio JDCSServerInformationService

Una vez inicializado cada uno de los recursos de los servicios Grid, los mismo ya pueden brindar cualquiera de las funcionalidades para lo que están diseñados.

Los servicios específicos hacen uso del API de la Plataforma de Cómputo Distribuido Tarenal para interactuar con ella.

```
public class JDCSProblemInformationService {
        [...]
        public GetProblemsResponse getProblems(GetProblems complexType) throws RemoteException  {
                [...]
                //Se obtiene el recursos del servicio Grid
                JDCSProblemInformationResource re = getResource();

                //Se obtiene el atributo jdcsServer del recurso y se construye con el mismo
                //un objeto de la clase JDCSServer. Esta clase es la encargada de obtener la información necesaria
                //del contenido XML.
                JDCSServer s = new JDCSServer(re.getJdcsServer());

                //Se hace uso del objeto de la clase JDCSServer para obtener los datos necesarios para interactuar
                //con el API de la plataforma
                UserAuthentication ua = new UserAuthentication(s.getUser(),s.getPass());
                ResquestingHost ro = new ResquestingHost(s.getServer(), s.getRmi_port(), s.getSocket_port());
                SystemConnection con = new SystemConnection(ua, ro);

                //Se obtiene la lista de problemas del servidor
                Vector problem_list = con.getProblemManager().getAllProblems();
                [...]
}
```

Figura 19. Fragmento de código donde se interactúa con la plataforma Tarenal para obtener el listado de sus problemas

Estos servicios están diseñado para utilizar como mecanismo de autorización una conversación segura. Esto es necesario para delegar las credenciales de los clientes que los consuman. Servicios como JDCSExecutionManagerService hacen uso especial de esta credencial delegada para llevar un control de las ejecuciones por credencial.

```
<?xml version="1.0" encoding="UTF-8"?>
<securityConfig xmlns="http://www.globus.org">
        [...]
        <auth-method>
                <GSISecureConversation/>
        </auth-method>
        [...]
</securityConfig>
```

Figura 20. Fragmento del fichero de configuración security_descriptor.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<securityConfig xmlns="http://www.globus.org">
        [...]
        <authz value="host"/>
                <GSISecureConversation>
                        <integrity/>
                        <delegation value="full"/>
                </GSISecureConversation>
        [...]
</securityConfig>
```

Figura 21. Fragmento del fichero de configuración client-security-descriptor.xml

Una vez que un servicio sea invocado, la credencial del cliente que lo invoca es obtenida de la siguiente forma:

```
public class JDCSExecutionManagerService {

        [...]
        public GetExecutionsResponse getExecutions(GetExecutions complexType)
                        throws RemoteException {
                [...]
                        GSSCredential credential = null;
                        MessageContext context = MessageContext.getCurrentContext();
                        Subject subject = (Subject) context
                                        .getProperty(org.globus.wsrf.impl.security.authentication.Constants.PEER_SUBJECT);

                        if (subject != null) {
                                credential = JaasGssUtil.getCredential(subject);
                        }
                        if (credential == null) {
                                credential = org.globus.axis.util.Util.getCredentials(context);
                        }
                        [...]
}
```

Figura 22. Fragmento de código donde se obtiene la credencial del cliente.

Como los servicios específicos están configurados para tener como método de autenticación la conversación segura, la única forma en que podrán ser consumidos es precisamente estableciendo la misma.

```
public class JDCSExecutionManagerClient {

        [...]
        public JDCSExecutionManagerClient(String services_url,
                        GSSCredential credential) throws MalformedURIException,
                        RemoteException, ServiceException {
                [...]
                EndpointReferenceType instanceEPR;
                JDCSExecutionManagerServiceAddressingLocator locator = new JDCSExecutionManagerServiceAddressingLocator();
                instanceEPR = new EndpointReferenceType();
                instanceEPR.setAddress(new Address(instanceURI));
                JDCSExecutionManagerPortType port = locator
                                .getJDCSExecutionManagerPortTypePort(instanceEPR);

                ((javax.xml.rpc.Stub) port)._setProperty(
                                org.globus.wsrf.security.Constants.GSI_SEC_CONV,
                                org.globus.wsrf.security.Constants.SIGNATURE);

                ((javax.xml.rpc.Stub) port)._setProperty(
                                org.globus.axis.gsi.GSIConstants.GSI_MODE,
                                org.globus.axis.gsi.GSIConstants.GSI_MODE_FULL_DELEG);
                if (this.credential != null)
                        ((javax.xml.rpc.Stub) port)._setProperty(
                                        org.globus.axis.gsi.GSIConstants.GSI_CREDENTIALS,
                                        this.credential);
                [...]
        }
        [...]
}
```

Figura 23. Fragmento de código donde un cliente establece una conversación segura con un servicio Grid

Para la implementación de la infraestructura Grid también se desarrollaron un conjunto de servicios generales, los cuales descubren a los servicios específicos por medio del Index Service y hacen uso de ellos. Los servicios generales son:

- JDCSGeneralProblemInformationService
- JDCSGeneralExecutionManagerService
- JDCSGeneralSolutionManagerService

Estos servicios generales están configurados de la misma forma que los específicos, exceptuando que definen un recurso que tiene como atributo una cadena de caracteres con formato XML con información general de la infraestructura. Estos recursos son inicializado de la misma forma que los recursos de los específicos.

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
        <index_service>https://10.34.20.117:8443/wsrf/services/DefaultIndexService</index_service>
        <services_url>https://10.34.20.117:8443/wsrf/services/</services_url>
        <gridftp>
                <name>10.34.20.117</name>
                <port>2811</port>
                <solutions_path>/var/tarenal/solutions</solutions_path>
        </gridftp>
</root>
```

Figura 24. Fichero XML usado por los servicios generales.

Los servicios generales hacen uso del método getJDCSServers() de la clase IndexServiceOperations para obtener la lista de los servidores Tarenal registrados.

```
    public Vector getJDCSServers() throws Exception {
            Vector v = new Vector();
            Vector temp = new Vector();
            MessageElement[] lista = query("//*[local-name()='JdcsServer']");
            if (lista != null) {
                    for (int i = 0; i < lista.length; i++) {
                            try {
                                    NodeImpl as = ((NodeImpl) lista[i].getChildren().get(0));
                                    String ser = as.getNodeValue();

                                    JDCSServer server = new JDCSServer(ser);
                                    if(!temp.contains(server.getServer()))
                                    {
                                            temp.add(server.getServer());
                                            v.add(server);
                                    }
                            } catch (Exception e) {
                            }
                    }
            } else
                    throw new Exception("No Tarenal servers found");
            return v;
    }
```

Figura 25. Método getJDCSServers() de la clase IndexServiceOperations

```
private MessageElement[] query(String xpathExpresion) throws Exception {
        [...]
        WSResourcePropertiesServiceAddressingLocator locator = new WSResourcePropertiesServiceAddressingLocator();
        QueryResourceProperties_PortType port = null;
        port = locator.getQueryResourcePropertiesPort(endpoint);
        ((Stub) port)._setProperty(Constants.GSI_ANONYMOUS, Boolean.TRUE);
        String queryString = xpathExpresion;
        QueryExpressionType query = new QueryExpressionType();
        query.setDialect(WSRFConstants.XPATH_1_DIALECT);
        query.setValue(queryString);
        QueryResourceProperties_Element request = new QueryResourceProperties_Element();
        request.setQueryExpression(query);
        QueryResourcePropertiesResponse response = null;
        response = port.queryResourceProperties(request);
        MessageElement[] mensajes = response.get_any();
        [...]
        return mensajes;
}
```

Figura 26. Método query() de la clase IndexServiceOperations

Una vez que se conocen los distintos servidores de la Plataforma de Cómputo Distribuido Tarenal, se puede hacer uso de los respectivos clientes específicos.

```
public class JDCSGeneralProblemInformationService {
        [...]
        public GetProblemsResponse getProblems(GetProblems complexType) throws RemoteException  {
                [...]
                JDCSGeneralProblemInformationResource resource = getResource();
                IndexServiceOperations index = new IndexServiceOperations(resource.getIndexService());
                Vector servers = index.getJDCSServers();
                [...]
                for (int i = 0; i < servers.size(); i++) {
                 JDCSServer s = (JDCSServer)servers.get(i);
                 [...]
                 JDCSProblemInformationClient client = new JDCSProblemInformationClient(s.getServices_url(),credential);
                 Vector problems = client.getProblems();
                 [...]
                }
        }

}
```

Figura 27. Uso del método getJDCSServers()

Los clientes de los servicios generales son definidos de igual manera que los clientes de los servicios específicos y son consumidos por los servlets de la aplicación Web desarrollada para acceder a la infraestructura Grid.

4.3 Pantallas del sistema

Los siguientes portlets permiten el acceso .a la infraestructura Grid a través del Portal de Servicios del Grupo de Bioinformática.

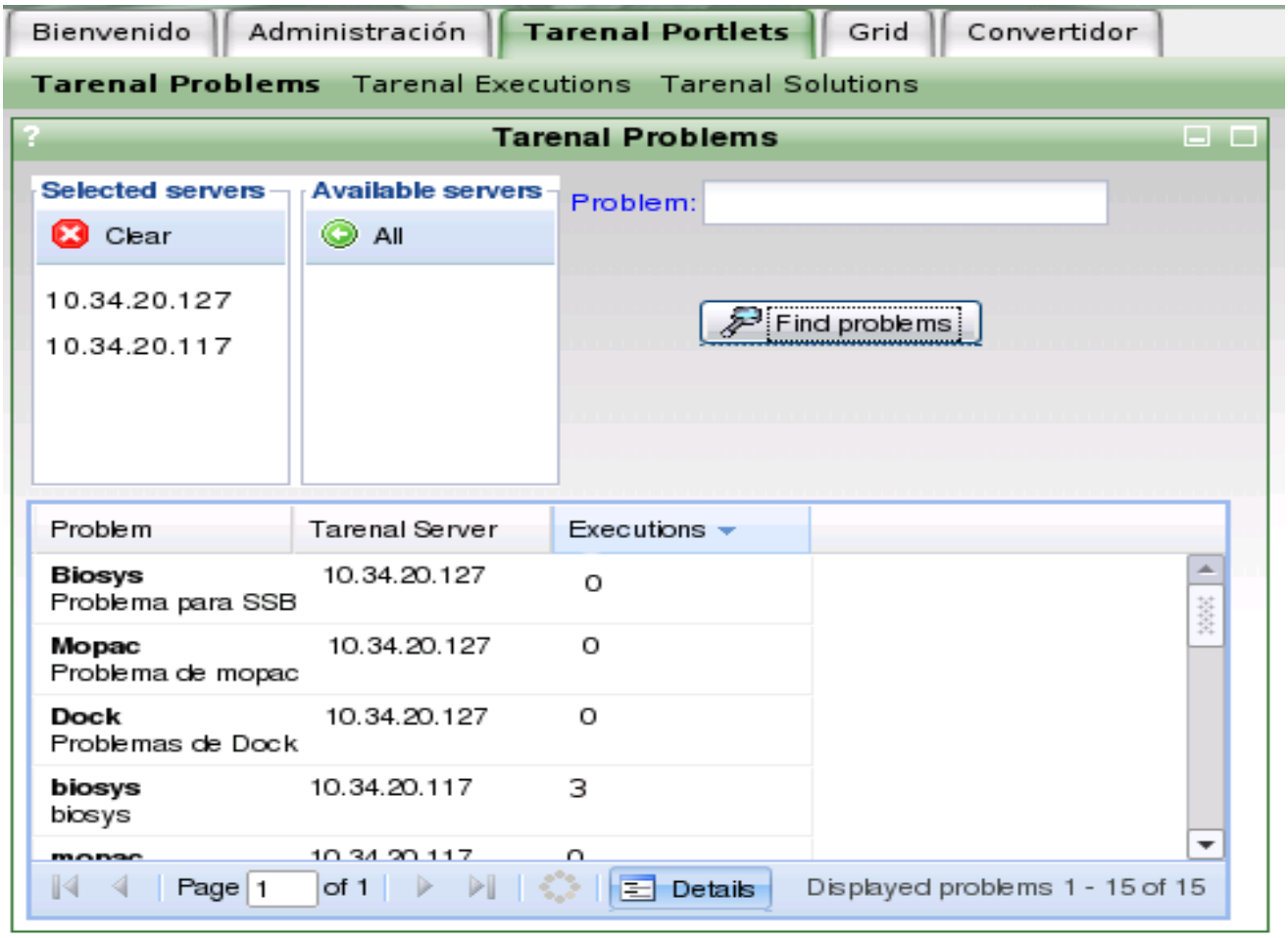

Figura 28. Portlet para buscar problemas en las plataformas Tarenal

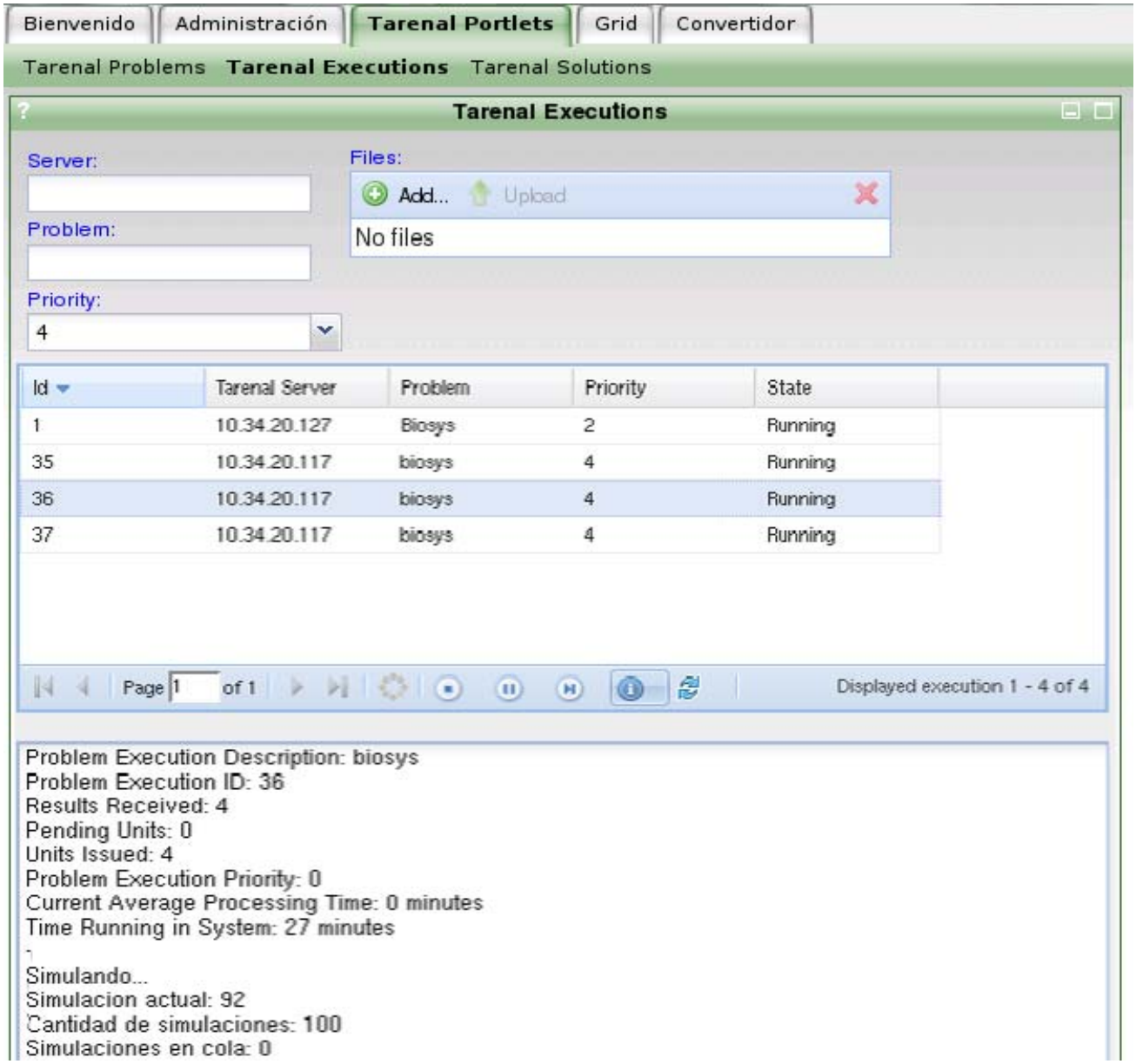

Figura 29. Portlet para crear y manipular ejecuciones de los problemas.

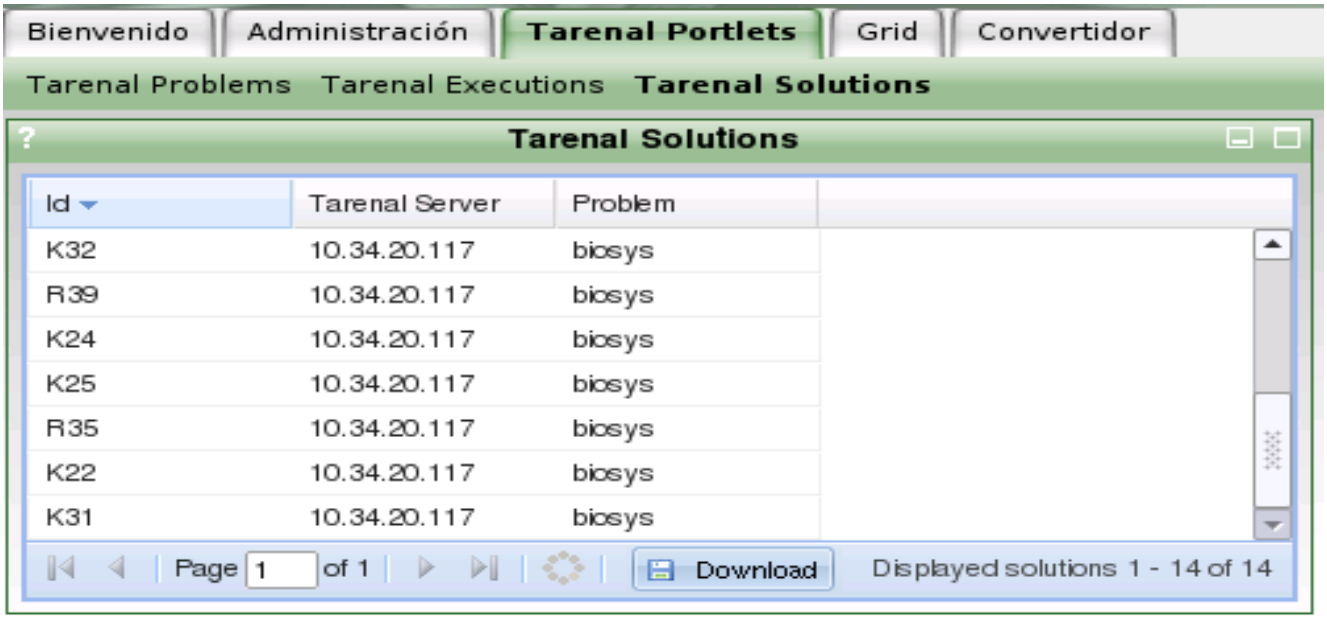

Figura 30. Portlet para obtener los resultados de las soluciones

Conclusiones

En ese capítulo se presentó el diagrama de componentes de la infraestructura. Además se presentó y explicó a grandes rasgos el código de las principales clases.

CONCLUSIONES

En el presente trabajo se realizó un estudio de los estándares y especificaciones para la implementación de servicios Grid con el objetivo de seleccionar las técnicas y herramientas necesarias para el desarrollo de los mismos.

Se diseñó una infraestructura capaz de incorporar a la Plataforma Tarenal como recurso de la Grid.

Se implementó la infraestructura diseñada, con el desarrollo y despliegue de un conjunto de servicios Grid sobre el middleware Globus Toolkit 4.

Se implementaron un conjunto de portlets para permitir a los usuarios del Portal de Servicios del Grupo de Bioinformática el acceso a la infraestructura desarrollada.

RECOMENDACIONES

- Desarrollar otros servicios Grid para ofrecer el resto de las funcionalidades que brinda el API de la Plataforma Tarenal.
- Incorporar funcionalidades para brindarle al usuario información estadística sobre los servidores Tarenal registrados.
- Incorporar a la infraestructura Grid las plataforma Tarenal de los polos científicos de la capital
- Realizar la integración con otros middlewares.

REFERENCIA BIBLIOGRÁFICA

[1] Barzanallana R. Historia de la Informática; 2005. Revisado Enero 2008. Available from: http://mundopc.net/actual/varios/historiainf/index.php

[2] González M. Bioinformática: A la búsqueda de su definición. Revisado Enero 2008. Available from: http://www.robotiker.com/revista/

[3] Rodríguez J.P.F. La bioinformática en Cuba: presente y perspectivas; 2005

[4] Aguilera L. Sistema de cómputo distribuido aplicado a la Bioinformática; 2008

[5] Sánchez F.M., Ramos A.V. Introducción a la tecnología grid. Revisado Enero 2008. Available from: http://www.astic.es/SiteCollectionDocuments/Astic/Documentos/Boletic/Boletic%2033/monografico2.pdf

[6] Mausolf J.,Subbian K. Grid Services Programming and Application Enablement; 2004.

[7] Globus Toolkit; Revisado Enero 2008. Available from: http://www.globus.org/

toolkit/

[8] Ann Wollrath R.R., Waldo J. A Distributed Object Model for the Java System; 1996. Conference on Object-Oriented Technologies. Toronto, Ontario, Canada

[9] Foster I. What is the Grid? A Three Point Checklist. Argonne National Laboratory & University of Chicago; 2002

[10] Global Grid Forum; Revisado Enero 2008. Revisado Enero 2008. Available from: http://www.ggf.org/

[11] Grid Technology Cookbook. Southeastern Universities Research Association. Revisado Enero 2008. Available from: http://www.sura.org/cookbook/gtcb/index.php

[12] Foster I., Kesselman C., Nick J.M., Tuecke S. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Revisado Enero 2008. Available from: http://www.globus.org/alliance/publications/papers/ogsa.pdf

[13] Foster I., Kishimoto H., Savva A., Berry D., Djaoui A., Grimshaw A., Horn B., Maciel F., Siebenlist F., Subramaniam R., Treadwell J. The Open Grid Services Architecture, Version 1.0. Revisado Enero 2008. Available from: http://www.gridforum.org/documents/GWD-I-E/GFD-I.030.pdf

[14] Foster I., Gannon D., Kishimoto H. Open Grid Services Architecture Use Cases. Revisado Enero 2008. Available from: http://www.gridforum.org/documents/GWD-I-E/GFD-I.029v2.pdf

[15] Kishimoto H., Treadwell J. Defining the Grid: A Roadmap for OGSA. Revisado Enero 2008. Available from: http://www.ogf.org/documents/GFD.53.pdf

[16] DRMAA. Revisado Enero 2008. Available from: http://www.drmaa.org

[17] JDSL. Revisado Enero 2008. Available from: https://forge.gridforum.org/projects/jsdl-wg/

[18] Enterprise Grid Alliance. Revisado Enero 2008. Available from: http://gridalliance.org/

[19] Open Grid Forum. Revisado Enero 2008. Available from: http://www.ogf.org/

[20] Organization for the Advancement of Structured Information Standards. Revisado Enero 2008. Available from: http://www.oasis-open.org/

[21] UDDI. Revisado Enero 2008. Available from: http://uddi.org/pubs/uddi_v3.htm#_Toc85907967

[22] SOAP version 1.2. Revisado Enero 2008. Available from: http://www.w3.org/TR/2002/WD-soap12-part0-20020626/

[23] WSDL. Revisado Enero 2008. Available from: http://www.w3.org/TR/wsdl

[24] World Wide Web Consortium. Revisado Enero 2008. Available from: http://www.w3.org/

[25] Group OM. Common Object Request Broker Architecture: Core Specification. Version 3.0.3; 2004.

[26] Box D. Essential COM. Addison-Wesley; 1997.

[27] Banks T. Web Services Resource Framework(WSRF) - Primer 1.2. OASIS. 2005. Available from: http://docs.oasis-open.org/wsrf/wsrf-primer-1.2-primer-cd-01.pdf

[28] Monroy J.A.G. Globus Toolkit. E.T.S.I.Telecomunicación- Departmento Ingenieria Sistemas Telemáticos Ciudad Universitaria s/n 28040 Madrid. Revisado Enero 2008. Available from: http://internetng.dit.upm.es/joe/Art/Globus.pdf

[29] García J.G. Utilización de grids para compresión de vídeos y como repositorio/servidor de vídeos. Universidad Politéctica de Catalunya. Revisado Enero 2008. Available from: http://upcommons.upc.edu/pfc/bitstream/2099.1/3818/2/36428-2.pdf

[30] Fellenstein C., Joseph J., Ernest M.. Evolution of grid computing architecture and grid adoption models. IBM Systems Journal, 43(4):624–645, 2004. Available from: http://www.research.ibm.com/journal/sj/434/josepaut.html

[31] MacKenzie C., Laskey K., McCabe F., Brown P., Metz R. Reference model for service oriented architecture 1.0. Technical report, OASIS Standards, 2006. Available from: http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.pdf

[32] Talia D. The open grid services architecture: Where the grid meets the web. IEEE Internet Computing, 06(6):67–71, 2002. Available from: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1067739

[33] Globus Alliance; Revisado Enero 2008. Available from: http://www.globus.org/

[34] GSI Working Group. Available from: https://forge.gridforum.org/projects/gsi-wg

[35] Foster I., Kesselman C., Tsudik G., Tuecke S., "A Security Architecture for Computational Grids," Proc. 5th ACM Conference on Computer and Communications Security Conference, pp. 83--92, 1998.

[36] GT4 Community Authorization Service (CAS) Administrators Guide. The Globus Alliance, 2005. Revisado Febrero 2008. Available from: http://www.globus.org/toolkit/docs/4.0/admin/docbook/ch14.html

[37] PKI y certificados digitales. Revisado Febrero 2008. Available from: http://www.idg.es/comunicaciones/articulo.asp?id=134356

[38] Novotny J. An Online Credential Repository for the Grid: MyProxy, et al. Proceedings of the 10th IEEE Symposium on High Performance Distributed Computing (HPDC 10), 2001. Revisado Febrero 2008. Available from http://www.globus.org/research/papers/myproxy.pdf

[39] MyProxy Credential Manager Service. Revisado Febrero 2008. Available from: http://grid.ncsa.uiuc.edu/myproxy/

[40] Allcock W., Bester J., Bresnahan J., Meder S., Plaszczak P., Tuecke S. GridFTP: Protocol Extensions to FTP for the Grid. Revisado Marzo 2008. Available from: http://www.ogf.org/documents/GFD.20.pdf

[41] Mandrichenko I., Allcock W., Perelmutov T. GridFTP v2 Protocol Description. Revisado Febrero 2008. Available from: http://www.ogf.org/documents/GFD.47.pdf

[42] Allcock W.E, Foster I., Madduri R. Reliable Data Transport: A Critical Service for the Grid. Revisado Febrero 2008. Available from: http://www.doc.ic.ac.uk/%7Esjn5/GGF/GGF11/BGBS-Allcock.pdf

[43] GT 4.0: Information Services: Index. Revisado Marzo 2008. Available from: http://www.globus.org/toolkit/docs/4.0/info/index/

[44] GT 4.0 WS MDS Trigger Service. Revisado Marzo 2008. Available from: http://www.globus.org/toolkit/docs/4.0/info/trigger/

[45] GT 4.0 WS MDS WebMDS. Revisado Marzo 2008. Available from: http://www.globus.org/toolkit/docs/4.0/info/webmds/

[46] WS GRAM Documentation. Revisado Marzo 2008. Available from: http://globus.org/toolkit/docs/3.2/gram/ws/index.html

[47] Manifiesto Ágil. Revisado Marzo 2008. Available from: http://www.agilemanifesto.org/

[48] OpenUP/Basic. Revisado Febrero 2008. Available from: http://epf.eclipse.org/wikis/openupsp/

[49] Eclipse - an open development platform. Revisado Febrero 2008. Available from: http://www.eclipse.org/

[50] Lesster P. Rational Unified Process (RUP). Revisado Febreo 2008. Available from: https://pid.dsic.upv.es/C1/Material/Documentos%20Disponibles/Introducci%C3%B3n%20a%20RUP.doc

[51] Object Management Group. Revisado Febrero 2008. Available from: http://www.omg.org/

[52] Martínez N., Hernández Y. M. Portal Web de Servicios Bioinformaticos; 2008

BIBLIOGRAFÍA

[1] Maozhen Li MB. The Grid. Core Technologies. John Wiley & Sons, Ltd; 2005. Revisado 2008.

[2] Minoli D. A Networking Aproach To Grid Computing. John Wiley & Sons, Ltd; 2005.

[3] Robbins S. Lessons in Grid Computing: The System Is a Mirror. John Wiley & Sons, Ltd; 2006.

[4] Globus Alliance; Revisado Enero 2008. Available from: http://www.globus.org/.

[5] I. Foster. Globus toolkit version 4: Software for service-oriented systems. In IFIP International Conference on Network and Parallel Computing, volume 3779 of Lecture Notes in Computer Science, pages 2–13. Springer-Verlag, 2005.

[6] Akhurst TJ. The Role of Parallel Computing in Bioinformatics. Rhodes University; 2005.

[7] Hariri S, Parashar M, editors. Tools and Environments for Parallel and Distributed Computing. Wiley; 2004.

[8] Overview of the Grid Security Infrastructure; Revisado Enero 2008. Available from: http://www.globus.org/security/overview.html.

[9] I. Foster. Designing and Building Parallel Programs. Addison Wesley; 1995.

[10] Open grid forum. Available from: http://www.ogf.org.

[11] I. Foster. What is the grid? a three point checklist. Grid Today, 1(6):22, 2002. Available from: http://www.globus.org/alliance/publications/papers.php.

[12] Tanenbaum A, Steen MV. Distributed Systems: Principles and Paradigms. Prentice Hall, Pearson Education, USA; 2002.

[13] G Couloris JD, Kinberg T. Distributed Systems - Concepts and Design, 4th Edition. Addison-Wesley, Pearson Education, UK; 2001.

[14] Berkeley Open Infrastructure for Network Computing; Revisado Enero 2008. Available from: http://boinc.berkeley.edu/.

[15] Keane T. A General-Purpose Heterogeneous Distributed Computing System. National University of Ireland Maynooth; 2004.

[16] Message Passing Interface; Revisado Enero 2008. Available from: http://www-unix.mcs.anl.gov/mpi/.

[17] Sun Microsystems I. RPC: Remote Procedure Call Protocol Specification. Version 2; 1988.

[18] Microsystem S. Java; Revisado Enero 2008. Available from: http://java.sun.com/.

[19] Grid Café:´The place for everybody to learn about the Grid; Revisado Junio 2007 Available from: http://gridcafe.web.cern.ch/gridcafe/index.html.

[20] Ian Foster CK. The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann; 1998.

[21] Sotomayor B. Computación Grid: Más allá de los superordenadores; 2006.

[22] Open Grid Services Architecture; Revisado Enero 2008.
Available from: http://www. globus.org/ogsa/.

[23] I. Foster. Designing and Building Parallel Programs. Addison Wesley; 1995.

[24] Moltó, G., Hernández, V., and Alonso, J. M. 2008. A service-oriented WSRF-based architecture for metascheduling on computational Grids. *Future Gener. Comput. Syst.*

[25] Dörnemann, T., Smith, M., and Freisleben, B. 2008. Composition and Execution of Secure Workflows in WSRF-Grids. In *Proceedings of the 2008 Eighth IEEE international Symposium on Cluster Computing and the Grid (Ccgrid).*

[26] Wasson, G. and Humphrey, M. 2005. Exploiting WSRF and WSRF.NET for Remote Job Execution in Grid Environments. In *Proceedings of the 19th IEEE international Parallel and Distributed Processing Symposium (Ipdps'05).*

[27] Reich, C., Banholzer, M., Buyya, R., and Bubendorfer, K. 2007. Engineering an Autonomic Container for WSRF-Based Web Services. In *Proceedings of the 15th international Conference on Advanced Computing and Communications* (December 18 - 21, 2007).

[28] Wang, X. D., Yang, X., and Allan, R. 2006. Flexible Grid Portlets to Access Multi Globus Toolkits. In *Proceedings of the Fifth international Conference on Grid and Cooperative Computing Workshops* (October 21 - 23, 2006).

[29] Alameda, J., Christie, M., Fox, G., Futrelle, J., Gannon, D., Hategan, M., Kandaswamy, G., von Laszewski, G., Nacar, M. A., Pierce, M., Roberts, E., Severance, C., and Thomas, M. 2007. The Open Grid Computing Environments collaboration: portlets and services for science gateways: Research Articles.

[30] Martin, A. and Yau, P. 2007. Grid security: Next steps. *Inf. Secur. Tech. Rep.*

[31] Sotomayor, B. and Childers, L. 2005 *Globus® Toolkit 4, First Edition: Programming Java Services (The Morgan Kaufmann Series in Networking)*. Morgan Kaufmann Publishers Inc.
[32] Mache, J. 2006. Hands-on grid computing with Globus Toolkit 4. *J. Comput. Small Coll*

GLOSARIO DE TÉRMINO

API: Application Programming Interface. Especificación de una librería o utilidad que documenta su interfaz y permite su uso sin conocimiento de su interior.

Ant: es una herramienta Open-Source utilizada en la compilación y creación de programas Java.

C: Lenguaje de programación estructurado de propósito general que ofrece control de flujo, estructuras sencillas y un buen conjunto de operadores. No es un lenguaje de muy alto nivel.

Certificado digital: Archivo que contiene información acerca de su propietario, que puede emplearse para identificarlo.

Criptografía: área de las telecomunicaciones que tiene como objetivo la protección de la información contra usuarios no autorizados.

Criptografía de clave pública: Consiste en claves relacionadas matemáticamente para encriptar y desencriptar.

CORBA: (Common Object Request Broker Architecture) arquitectura común de intermediarios en peticiones a objetos, es un estándar que establece una plataforma de desarrollo de sistemas dsitribuidos facilitando la invocación de métodos remotos bajo un paradigma orientado a objetos.

CVS: herramienta de control de versiones**.**

GRASP: son patrones Generales de Software para Asignación de Responsabilidades**,** los cuales se aplican en el diseño de la aplicación.

HTTP: Protocolo de comunicaciones utilizado por los programas clientes y servidores de WWW para comunicarse entre sí.

HTTPS: HTTP sobre SSL. Protocolo que permite transmitir de manera segura páginas Web.

HTML: (HyperText Markup Language) ó (Lenguaje de Marcas Hipertextuales), lenguaje en el que se programan las páginas Web.

IBM: International Business Machines, una de las mayores compañías de fabricación de hardware y software en el mundo desde los años cincuenta. Conocida coloquialmente como el Gigante Azul.

IDE: Un “Integrate Development Enviroment” es una herramienta de soporte al proceso de desarrollo de software que integra las funciones básicas de edición de código, compilación y ejecución de programas, entre otras.

JSP: Java Server Pages es la tecnología del lado del servidor para generar páginas Web de forma dinámica en el servidor, desarrollado por Sun Microsystems, basado en scripts que utilizan una variante del lenguaje java.

Máquinas virtuales: es un software que emula a un ordenador y puede ejecutar programas como si fuese un ordenador real.

SSL: Secure Sockets Layers. Un protocolo desarrollado por Netscape para asegurar los datos enviados por el navegador mediante un criptado.

Plugin: Accesorio adicional que se integra con un programa y añade nuevas funcionalidades.

Plugin: Función o utilidad generalmente muy específica. Se adiciona a algún programa para ser ejecutado. Los plugins típicos tienen la función de reproducir determinados formatos de gráficos, reproducir datos multimedia, etc.

RMI: (Java Remote Method Invocation), Invocación de métodos remotos, consiste en que un objeto acceda a un método (una de las funcionalidades) de otro objeto remoto (que esté situado en otro punto de una red).

RUP: Proceso Unificado (Rational Unified Process) metodología para el desarrollo de sistemas informáticos, dirigidos por casos de uso.

Servlet: son objetos que corren dentro del contexto de un servidor de aplicaciones.

Virtual Organization(VO): Grupo de individuos u instituciones que comparten los recursos computacionales de una Grid por un objetivo común

WSDL: Lenguaje de Descripción de Servicios Web. Un lenguaje basado en XML usado para describir servicios Web.

XML: (eXtensible Markup Language) ó (Lenguaje de marcas extencibles), es considerado como un metalenguaje de definición de documentos estructurados mediante marcas o etiquetas.

XML Schema: Lenguaje de esquema utilizado para describir la estructura y las restricciones de los contenidos de los documentos XML.