Disponible en línea en www.sciencedirect.com

ScienceDirect

Procedia Computer Science 4 (2011) 392–401

Procedia Computer Science

Conferencia Internacional sobre Ciencias Computacionales, ICCS 2011

Un marco de API web para desarrollar portales Grid

Valerio De Luca una, Italo Epicoco una, *, Daniele Lezzi una, Giovanni Aloisio una

una *Dep. Ingeniería para la Innovación, Universidad de Salento, via per Monteroni, 73100 Lecce, Italia*

Resumen

En este artículo describimos un entorno de resolución de problemas de cuadrícula que desarrollamos para aplicaciones financieras. Basamos su desarrollo en un marco de portlet que hemos desarrollado específicamente y en un conjunto de API web que encapsulan toda la lógica de cálculo y control de grid. Aunque hoy en día los portales grid se caracterizan por varios y di ff características diferentes y se implementan en muy diferentes ff Al considerar los lenguajes de programación y las tecnologías, pensamos que tienen muchos aspectos estructurales en común. Por esta razón, decidimos diseñar e implementar un conjunto de API web específicas de Grid, que llamamos GRB WAPI. A través de ellos, un desarrollador de portal no tendrá que lidiar con los detalles técnicos de la red y podrá gestionar un diseño de alto nivel. Un desarrollador de portal podrá concentrarse en algunos otros aspectos relacionados con la presentación, como la usabilidad y funcionalidad del portal. Descartamos la idea de desarrollar una biblioteca tradicional para liberar a los desarrolladores de portales de una tecnología de implementación particular. Gracias a esta elección, la lógica de presentación del portal se puede implementar en cualquier tecnología web y puede estar en un di ff servidor actual

Palabras clave: Entorno de resolución de problemas de cuadrícula, API web, REST, portales de cuadrícula, portlet

1. Introducción

Un portal Grid es una puerta de enlace basada en la web que proporciona un acceso perfecto a una variedad de recursos de backend. En general, un portal Grid proporciona a los usuarios finales una vista personalizada de los recursos de software y hardware específicos para el dominio de su problema particular. También proporciona un único punto de acceso a los recursos basados en Grid que están autorizados a utilizar. Esto permite a los científicos o ingenieros concentrarse en su área problemática al hacer de Grid una extensión transparente de su entorno informático de escritorio. Los servicios web pueden considerarse una tecnología importante para desarrollar interacciones automatizadas entre aplicaciones distribuidas y heterogéneas. Varios estándares, como WSDL (Lenguaje de descripción de servicios web), UDDI y SOAP, apoyan la definición de servicios web y su publicidad para la comunidad de usuarios potenciales. Sin embargo, se solicitan cada vez más enfoques de servicio más ligeros, especialmente para las aplicaciones web. La tecnología de API web nació solo para satisfacer esta necesidad.

En este artículo describimos el diseño de un marco de portal para desarrollar entornos de resolución de problemas de red basados en

* Autor correspondiente. Tel +39 0832 297235; fax: +39 0832 297235

Correos electrónicos: valeriodl@gmail.com (Valerio De Luca), italo.epicoco@unisalento.it (Italo Epicoco), daniele.lezzi@gmail.com (Daniele Lezzi), giovanni.aloisio@unisalento.it (Giovanni Aloisio)

Selección y / o revisión por pares bajo la responsabilidad del Prof. Mitsuhisa Sato y el Prof. Satoshi Matsuoka doi: 10.1016 / j.procs.2011.04.041

en un conjunto de portlets que se comunican con API web que encapsulan toda la lógica de cálculo y control de grid; llamamos GRB WAPI a estas API web específicas de Grid. Al usarlos, un desarrollador de portal no tiene que lidiar con los detalles técnicos de la cuadrícula concentrándose en algunos otros aspectos relacionados con la presentación, como la usabilidad y funcionalidad del portal. Descartamos la idea de desarrollar una biblioteca tradicional para liberar a los desarrolladores de portales de una tecnología de implementación particular. Gracias a esta elección, la lógica de presentación del portal se puede implementar en cualquier tecnología web y puede estar en un di ff servidor actual.

Adoptamos un estilo arquitectónico REST [26] en nuestro diseño de API web porque es liviano en comparación con los servicios web para desarrollar interacciones automatizadas entre aplicaciones distribuidas y heterogéneas. En las arquitecturas de estilo REST, las solicitudes y respuestas web se basan en la transferencia de representaciones de recursos. Un recurso puede ser esencialmente cualquier concepto coherente y significativo que se pueda abordar. Una representación de un recurso es típicamente un documento que captura el estado actual o previsto de un recurso. Las aplicaciones RESTful maximizan el uso de la interfaz bien definida y preexistente y otras capacidades integradas proporcionadas por el protocolo de red elegido y minimizan la adición de nuevas características específicas de la aplicación encima. El documento está organizado de la siguiente manera: en la sección 2 se presenta una breve descripción del estado del arte para contextualizar nuestro trabajo; la WebAPI y la arquitectura del servidor de portal se analizan en las secciones 3 y 4; la sección 5 describe el diseño del marco del portal grid basado en portlets, la sección 6 presenta un caso de uso de un portal de finanzas grid y la sección 7 concluye el documento.

2. Estado de la técnica

El desarrollo de portales de red se puede clasificar en general en dos generaciones. Los portales Grid de primera generación están estrechamente acoplados con middleware Grid como Globus. Estos portales generalmente brindan servicios de red como autenticación, administración de trabajos, transferencia de datos y servicios de información. Algunos conjuntos de herramientas representativos del portal Grid de primera generación son GRB [1] [2] [3] [4], GridPort 2.0 [6], Ninf Portal [7] y GridSpeed [8]. Una limitación importante de los portales de primera generación es la falta de personalización porque el conocimiento y la experiencia necesarios para el uso de los kits de herramientas del portal están más allá de la capacidad de la mayoría de los usuarios finales de Grid. Además, los portales Grid de primera generación a menudo ofrecen solo servicios de portal restringidos. Además,

La segunda generación de portales Grid se basa en tecnologías como los portlets para proporcionar soluciones más personalizables. Algunos marcos de portales representativos y populares son Jetspeed, WebSpheres Portal y GridSphere [5]. Al ejecutarse dentro de un contenedor de portlets, los portlets se pueden agregar o eliminar de un portal, lo que brinda a los usuarios la capacidad de personalizar los servicios Grid a nivel de portal. Los portlets de Grid son componentes independientes que se basan en servicios Grid existentes. Un portal Grid construido a partir de Grid Portlets puede permitir a los usuarios integrar los servicios proporcionados por di ff middlewares de Grid existentes. Los portales Grid de segunda generación con portlets tienen los siguientes beneficios en comparación con los portales Grid de primera generación:

- Personalización del portal: los usuarios pueden crear sus portales personalizados a partir de los portlets disponibles para satisfacer sus necesidades específicas sin la ayuda de un desarrollador de sistemas Grid.
- Servicios de Grid extensible: los portales creados a partir de portlets están acoplados de forma flexible con las tecnologías de middleware de Grid, ya que los servicios de Grid se pueden exponer como portlets estándar; un portal construido a partir de portlets proporciona a los usuarios la capacidad de integrar servicios de di ff diferentes proveedores de servicios de Grid.
- Servicios de Grid dinámico: un portal de Grid debería poder proporcionar a los usuarios la capacidad de acceder a servicios de Grid dinámicos en un entorno Grid; con este fin, se puede proporcionar un mecanismo para exponer los servicios Grid como portlets individuales que se pueden publicar y acceder a través de un portal.

Muchos portales grid basados en portlets se han desarrollado utilizando el marco del portal GridSphere. El primero se llamó portal P-GRADE Grid [9]. Otros portales de grid basados en portlets son BeSTGRID [10], CCDB Portal [11] y University of Sydney Portal [12].

Los portales Grid también se pueden agrupar en dos categorías: portales de usuarios y portales de aplicaciones. Un portal de usuario proporciona un conjunto de servicios básicos para los usuarios del portal, que incluye inicio de sesión único, envío y seguimiento de trabajos, selección de recursos y gestión de datos. Un portal de aplicaciones proporciona servicios relacionados con la aplicación para los usuarios, por ejemplo, para construir una aplicación de dominio para Grid.

Otro concepto relacionado, el llamado mashup, ha surgido recientemente; indica una forma de crear nuevas aplicaciones web mediante la combinación de recursos web existentes utilizando datos y API web. Estas herramientas o ff er oportunidades considerables como la disponibilidad de datos como servicio que, junto con la implementación de aplicaciones colaborativas, pueden ayudar a realizar la visión de la Web programable abierta. Uno de los ejemplos más comunes de API web son las API de datos de Google [13], que permiten a los programadores crear aplicaciones que leen y escriben datos de los servicios de Google. Actualmente, estos incluyen API para Google Apps, Google Analytics, Blogger, Google Base, Google Book Search, Google Calendar, Google Code Search, Google Earth, Google Spreadsheets, Google Notebook y Álbumes web de Picasa. Los servicios de Google utilizan un protocolo de datos REST para leer, escribir y modificar información en la web. Muchos servicios de Google brindan acceso externo a datos y funcionalidades a través de API que utilizan el Protocolo de datos de Google. Actualmente, el protocolo admite dos modos principales de acceso:

- AtomPub [24]: la información se envía como una colección de elementos Atom, utilizando el formato de sindicación estándar Atom para representar datos y HTTP para manejar la comunicación. El protocolo de datos de Google extiende AtomPub para procesar consultas, autenticación y solicitudes por lotes.

- JSON (notación de objetos JavaScript) [25]: la información se envía como objetos JSON que reflejan la representación de Atom.

Las bibliotecas cliente de datos de Google proporcionan herramientas y una capa de abstracción para que el usuario no tenga que lidiar con solicitudes HTTP o analizar rawXML. Las bibliotecas están disponibles para Java, JavaScript, .NET, PHP y Python. La tecnología REST es di ff diferente de los protocolos RPC como SOAP, donde HTTP es solo un sobre para los datos RPC y la mayor parte de la información está contenida dentro de la carga útil XML. La API GridRPC [14] es un intento de estandarizar e implementar un conjunto de API basado en RPC portátil y simple para Grid Computing. Un marco web importante basado en RPC para aplicaciones grid es WAGA [15]: proporciona WAGA-designer, una GUI implementada con tecnología mashup y AJAX que utiliza un conjunto de WAGAAPI envuelto con re fl exión Java y tecnología RPC. El NERSCWeb Toolkit (NEWT) [16] es una colección de API REST específicamente diseñada para desarrollar aplicaciones de cuadrícula. Las API NEWT utilizan JSON como formato de intercambio de datos: dado que el navegador del cliente analiza automáticamente cada objeto JSON como un objeto JavaScript nativo, cualquier persona que conozca HTML y JavaScript puede programar la interfaz NEWT.

En nuestro diseño de API web, elegimos XML en lugar de JSON para codificar los datos de salida, con el fin de permitir transformaciones XSLT. En particular, optamos por eximir a los clientes del análisis de datos: por esta razón, decidimos usar transformaciones XSLT del lado del servidor para construir fragmentos HTML a partir de datos de salida XML. Además, no queremos obligar a los desarrolladores de portales a usar solo JavaScript: gracias a algunos contenedores de API web o bibliotecas de enlace que implementamos en varios lenguajes de programación, permitimos que un desarrollador de portales elija un lenguaje de programación.

3. Arquitectura GRB WAPI

El diseño e implementación de GRB WAPI se basa en las Bibliotecas GRB desarrolladas en la Universidad de Salento, Italia. Las bibliotecas GRB implementan las funcionalidades básicas necesarias para acceder a varios servicios de middleware de grid. El portal GRB grid, basado en las bibliotecas GRB, se desarrolló utilizando tecnologías CGI. El objetivo de GRB WAPI es permitir una evolución de los portales basados en GRB desacoplando la interfaz web de los servicios de middleware. GRB WAPI cumple con la definición de API web, por lo que pueden invocarse a través del protocolo HTTP a través de un servidor HTTP. Las funciones de GRB WAPI toman los parámetros de entrada como una solicitud POST y / o archivos con una solicitud de datos de formulario / de varias partes y escriben su salida en formato XML en la respuesta HTTP relacionada. Con el objetivo de otorgar un buen nivel de seguridad,

```
<salida>
<resultado> </resultado>
<error id => </error>
</output>
```

los *salida* El elemento, que es la raíz del documento XML, incluye un elemento de resultado, que contiene los resultados del procesamiento, y un elemento de error, que en caso de error contiene un mensaje y su código relacionado almacenado en el atributo id (cuando el procesamiento finaliza sin errores, el error elemento contiene una cadena vacía y su atributo id tiene el valor 0). El primer GRB WAPI que debe invocarse es *grb wapi login,* que autentica a un usuario que desea acceder a los servicios de GRB WAPI: toma el nombre de usuario y la contraseña como parámetros de entrada y devuelve un identificador de sesión, que representa la autenticación del usuario con el sistema y debe pasarse como parámetro de entrada a todos los siguientes Llamadas GRB WAPI. GRB WAPI proporciona varios servicios, como administración de credenciales de usuario, definición de organización virtual, definición de recursos computacionales, envío de trabajos, monitoreo de recursos y seguimiento del estado del trabajo. Por ejemplo el *envío de trabajo de grb wapi* se puede utilizar para el envío de trabajos: toma como parámetros de entrada el identificador de sesión y un archivo JSDL que describe el trabajo a enviar de acuerdo con el estándar JSDL. Una vez enviado el trabajo, *grb wapi comprobar trabajo* podría invocarse para rastrear el estado del trabajo. los *grb wapi comprobar trabajo* requiere la identificación del identificador de sesión y la identificación del trabajo como parámetros de entrada.

4. Arquitectura del portal

Como hemos dicho, nuestra arquitectura de portal se basa en una separación importante entre la lógica de control y computación y la lógica de presentación. Como consecuencia, la arquitectura contiene dos entidades: un servidor de portal y un servidor web para alojar el GRB WAPI. Portal Server es responsable de la lógica de presentación y maneja las interfaces de usuario. El Servidor Web GRB WAPI es el responsable de la lógica de procesamiento y control en la que se basa el portal para exponer estos servicios a través del GRB WAPI. El sistema diseñado se basa en una arquitectura de tres niveles:

- el primer nivel está representado por el usuario que accede al portal a través de un navegador web;
- el segundo nivel consiste en un servidor de portal que aloja los portlets y es responsable de construir dinámicamente las páginas web que contienen los contenidos del portal;
- el tercer nivel está representado por el servidor web donde se implementa la lógica GRB WAPI; este servidor incluye también un servidor MySQL que contiene una base de datos relacional a la que acceden las WAPI de GRB.

El diagrama de implementación de UML de la figura 1a muestra la arquitectura que acabamos de describir. Las WAPI de GRB proporcionan al portal un alto nivel de flexibilidad. La capa de presentación podría desarrollarse en cualquier lenguaje y tecnología de programación web utilizando una biblioteca de enlace adecuada escrita en el lenguaje de programación elegido para invocar llamadas remotas GRB WAPI.

Nuestro portal se caracteriza por la claridad de diseño, e ffi modularidad ciente, múltiples vistas, facilidad de crecimiento y mantenimiento y potentes interfaces de usuario: adoptamos la tecnología de portlets de Java para alcanzar estos objetivos. La tecnología de portlets, como se define en la especificación JSR-168 [17], permite el desarrollo de componentes web reutilizables para crear interfaces de usuario accesibles y altamente personalizables. Un portlet es un componente web basado en Java que procesa las solicitudes de un contenedor de portlet y genera contenido dinámico, que es el denominado fragmento. Un fragmento se puede agregar con otros fragmentos para formar un documento completo, llamado página del portal. JSR-286 [18] define la especificación del portlet 2.0. Basamos el desarrollo de nuestro portal en esta segunda especificación, porque cumple con las tecnologías Web 2.0 y, en particular, porque permite manejar solicitudes HTTP asincrónicas. El contenedor de portlets que elegimos es Liferay [19], un marco de código abierto que proporciona a los usuarios páginas personalizables. Sin embargo, diseñamos nuestro portal para que sea independiente de los contenedores.

El diagrama de secuencia UML de la figura 1b muestra todas las interacciones entre los componentes de la arquitectura. Cuando un cliente solicita una página, cada portlet de esa página realiza las siguientes operaciones de manera iterativa para obtener algunos fragmentos HTML que deben ensamblarse para formar el contenido de la página:

- invoca una llamada remota GRB WAPI en el servidor WEB API, que devuelve un fragmento XML que contiene el resultado de la elaboración;
- aplica una hoja de estilo XSL al fragmento XML para ejecutar una transformación XSLT y obtener un fragmento HTML.

Estas dos operaciones están en el ciclo más interno del diagrama de secuencia. El bucle externo contiene las operaciones realizadas para formar el contenido del portlet. Fuera de este bucle, Portlet Server recopila el portlet para crear la página del portal que debe enviarse al cliente.

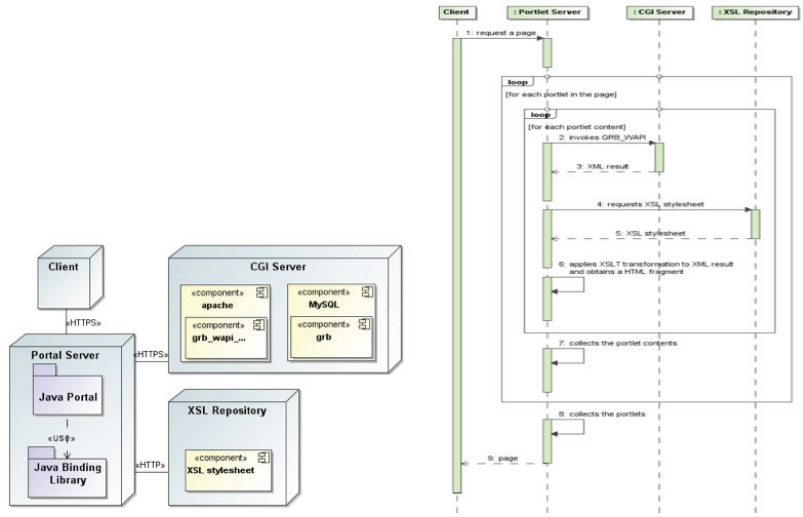

Figura 1: (a) Arquitectura del portal; (b) Interacciones entre los componentes de la arquitectura

5. Diseño del marco del portal

Basamos el desarrollo de portlets en el diseño de un marco que permite invocar métodos de biblioteca de enlaces Java y transformar la salida XML en contenido de portlets. Este marco se puede dividir en cuatro capas. La capa más baja es responsable de invocar la API de biblioteca de enlace, cuyos métodos son contenedores de llamadas GRB WAPI. Los resultados de este nivel son fragmentos XML, que a veces se aglomeran para formar nuevos fragmentos XML. Una capa superior aplica algunas transformaciones XSLT a fragmentos XML para obtener algunos fragmentos HTML. En la capa más alta, los componentes HTML se aglomeran en un contenido HTML, que puede ser todo el contenido que se muestra en el modo de portlet EDITAR o VER o se puede aglomerar con otros contenidos para formar un contenido múltiple.

5.1. Biblioteca de enlace de Java

La biblioteca de enlaces de Java que vamos a describir actúa como una capa de abstracción que facilita el desarrollo de un portal: al usarlos no es necesario invocar directamente GRB WAPI, escribir solicitudes HTTP y leer las respuestas relacionadas. La biblioteca de enlaces de Java que desarrollamos tiene una estructura de tres capas:

- la capa más alta consta de métodos que el desarrollador invocará para construir el portal;
- la capa intermedia es una envoltura que utilizan los métodos de la primera capa para invocar las WAPI de GRB;
- la capa más baja es utilizada por la capa intermedia para construir solicitudes HTTP que contactan a GRB WAPI, pasándoles los parámetros apropiados.

Los desarrolladores de portales solo necesitan usar los métodos de capa más alta, ya que las otras capas solo o ff er servicios a este. Las cuatro clases, representadas en el diagrama UML de la figura 2a, implementan la arquitectura descrita anteriormente: en particular *GRBWapiServer, GRBWapi* y *MultipartFormOutputStream* clase implementar los servicios o ff ered por las tres capas, mientras *GRBException* class representa todas las excepciones planteadas. *GRBWapiServer* class es la que usa directamente un desarrollador de portal y es una abstracción del Web Server donde se han instalado GRB WAPI: por lo que el constructor de clases toma como parámetro de entrada la URL del Web Server, mientras que casi todos los métodos de clase son envoltorios utilizados para corresponder Invocaciones de GRB WAPI. Después de la *GRBWapiServer* objeto ha sido creado, es necesario invocar un *openConnection* método para conectarse al servidor GRB WAPI y autenticar al usuario: este método toma el nombre de usuario y la contraseña como parámetros de entrada y los pasa a *grb wapi login.* Si la autenticación es exitosa, se crea un identificador de sesión:

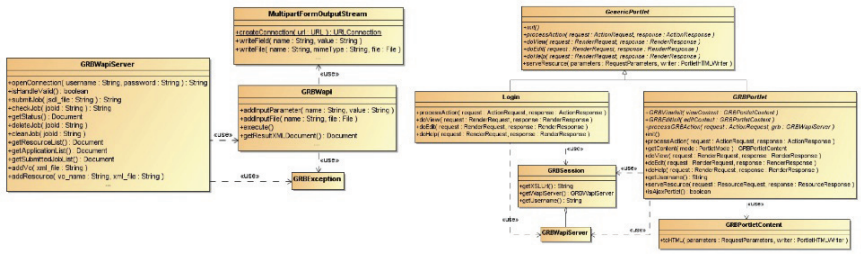

Figura 2: (a) diagrama de clases UML para la biblioteca de enlace de Java; (b) Diagrama de clases UML para la sección principal del marco

este identificador será tomado como un parámetro de entrada por GRB WAPIs invocados por todos los otros métodos de clase.

Una vez establecida la conexión, el desarrollador debe invocar el *getUsername* método para recuperar el nombre de usuario de los usuarios, el *getSessionId* método para recuperar el identificador de sesión que se ha creado y *isHandleValid* para comprobar si el identificador de sesión es válido. Todos los demás métodos que pueden ser invocados por un desarrollador proporcionan varios servicios de grid. Cada uno de ellos realiza las siguientes operaciones:

- creación de un *GRBWapi* objeto para que se invoque GRB WAPI;
- pasar los parámetros de entrada al *GRBWapi* objeto; los parámetros de entrada se pasan con el *addInputParameter*
  El método y / o los archivos de entrada XML se pasan con el método addInputFile;
- invocación del GRB WAPI real representado por el *GRBWapi* objeto a través del método de ejecución, en el caso de que no se devuelvan salidas, o el *getResultXMLDocument*

método cuando la invocación remota devuelve salidas. En el último caso, las salidas se devuelven en un documento XML representado por un *Documento* objeto. Si se produce un error remoto, se establece la etiqueta de error del documento de salida XML y la ejecución o la *getResultXMLDocument* métodos del *GRBWapi* clase lanza un *GRBException.*

Como muchos GRB WAPI también toman algunos archivos además de parámetros simples como datos de entrada, necesitábamos permitir también la carga de archivos en solicitudes HTTP, por lo que elegimos usar POSTmethod y solicitud multiparte. Como se describe en RFC 1521 [20] en la solicitud multiparte, el campo Content-Type del encabezado contiene el valor multipart / form-data y el cuerpo de la solicitud se compone de más partes, delimitadas por una cadena de límite adecuada que debe especificarse en el campo Content-Type del encabezado principal. los *MultipartFormOutputStream* la clase es utilizada por el *GRBWapi* clase para escribir la solicitud HTTP: su constructor toma como parámetros el flujo de salida donde se debe escribir la solicitud y una cadena de límite.

5.2. Clases de marco

El diagrama de clases de UML en la figura 2b ilustra la sección principal de nuestro marco de portal. *GenericPortlet* Implementos de clase abstracta *Portlet* interfaz yo ff ers una plantilla que se puede utilizar para desarrollar cualquier portlet. En nuestro marco, esta clase se especializó aún más en *GRBPortlet* clase abstracta, que a su vez se utilizó como plantilla para construir los portlets de portales de casos de uso. Nuestros portlets pueden de fi nir su VER y EDITAR contenidos específicos implementando *GRBViewInit*
y *GRBEditInit* métodos respectivamente: *viewContent* y *contenido editado* los parámetros son instancias de *GRBPortletContent*
class y se utilizan para recopilar todos los fragmentos de contenido que el portlet tiene que mostrar en modalidad VIEW y EDIT. los *procesoGRBAcción* El método define las operaciones que debe realizar el portlet cuando se genera una nueva solicitud HTTP mediante el envío de un formulario. los *isAjaxPortlet* un portlet utiliza el método para solicitudes HTTP asíncronas utilizando tecnología Ajax; esto permite que un portlet recupere datos del servidor de forma asíncrona en segundo plano sin interferir con la visualización y el comportamiento del resto de la página donde se encuentra el contenido del portlet; por lo que, por ejemplo, un envío de formulario en el portlet no provocará una actualización de toda la página.

En todos estos casos, en *GRBPortlet* clase aplicamos un patrón de diseño de método de plantilla, que sugiere de fi nir la estructura general de un algoritmo en una clase abstracta permitiendo que subclases concretas de fi nen algunos detalles del mismo. El único portlet que no se extiende *GRBPortlet* la clase es la *Iniciar sesión* portlet, que es una subclase directa del *GenericPortlet* clase abstracta: este es el portlet que el usuario debe utilizar para autenticarse. Después de que el usuario haya proporcionado su nombre de usuario

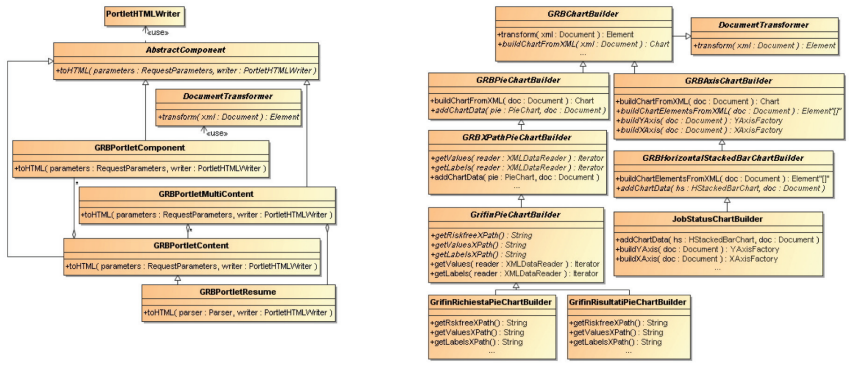

Figura 3: (a) diagrama de clases UML para el manejo de componentes de portlet; (b) Diagrama de clases UML para la construcción de gráficos.

y contraseña, *processAction* El método construye un *GRBSession* objeto, que contiene un *GRBWapiServer* ejemplo; el *openConnection* Este método se invoca en este objeto para autenticar al usuario con el servidor GRB WAPI. *GRBSession* El objeto se almacena en la sesión del portal como un atributo con ámbito de aplicación, por lo que cualquier portlet de la aplicación web puede acceder a él. Como hemos visto antes, el contenido que mostrará el portlet en modo VIEW y EDIT puede definirse como la composición de más componentes, que son fragmentos HTML generados por transformaciones XSLT aplicadas a marcas XML que representan salidas GRB WAPI. El diagrama de clases UML de la figura 3a ilustra la sección de nuestro marco que trata de estos componentes, su composición y su visualización.

ResumenComponente La clase abstracta representa un componente genérico que se puede representar en el modo VIEW o EDIT del portlet: define el *toHTML* método abstracto, que se encarga de construir un fragmento HTML y escribirlo en el contenido del portlet; este método debe ser implementado por subclases concretas, *GRBPortletComponent, GRBPortletContent* y *GRBPortletMultiContent.* Se aplica un patrón de diseño de estrategia: cada componente se crea instanciando una de las tres subclases, pero luego se considera como un *AbtractComponent* objeto; sin embargo, cuando se invoca el método toHTML para escribir el contenido del portlet, lo que realmente se ejecuta es una implementación específica para la subclase instanciada por el objeto.

GRBPortletContent La subclase representa un contenido completo que se puede representar en la modalidad VIEW o EDIT del portlet. El contenido se compone de uno o más componentes, cada uno de ellos a su vez representado por *GRBPortletComponent* clase. Entonces, además del patrón de diseño de estrategia, se aplica un patrón de diseño compuesto, que sugiere combinar más objetos dentro de otro objeto que tiene el mismo comportamiento de sus partes componentes; por lo tanto, este patrón de diseño permite manejar objetos y composiciones individuales de manera uniforme. En el caso que acabamos de describir, la clase AbstractComponent es una abstracción común tanto para objetos primitivos como compuestos: en realidad se extiende por ambos *GRBPortletComponent*

clase, que representa un objeto primitivo, y *GRBPortletContent* class, que representa un objeto compuesto. los *toHTML* método de *GRBPortletContent* clase invoca el *toHTML* método de cada objeto que forma parte del contenido para obtener los fragmentos HTML que hay que ensamblar para componer todo el contenido HTML. los *GRBPortletMultiContent* La subclase representa contenido múltiple, dado por la composición de contenidos más simples, que son instancias de *GRBPortletContent* clase. Aquí el patrón de diseño compuesto se utiliza también para *GRBPortletCon* y *GRBPortletMultiContent* clases; en este caso, *GRBPortletContent* representa el objeto primitivo, mientras que *GRBPortletMultiContent* representa el objeto compuesto. los *toHTML* método de *GRBPortletMultiContent* clase invoca el *toHTML* método de cada objeto que es parte del contenido múltiple para construir los fragmentos HTML relacionados; luego, este método ensambla los fragmentos de contenido único para construir todo el contenido múltiple HTML. Todas *ResumenComponente* las subclases usan el *PortletHTMLWriter* objeto de clase que se pasa como entrada a su *toHTML*

método, para escribir el fragmento HTML del componente que representan. los *GRBPortletComponent* invoca una implementación concreta del método de transformación de una subclase de la *Transformador de documentos* clase abstracta para ejecutar transformaciones XSLT que generan fragmentos HTML a partir de documentos XML; esta es otra aplicación del patrón de diseño de estrategia.

El diagrama de clases de UML en la figura 3b ilustra las clases responsables de construir gráficos. los *GRBChartBuilder* La clase abstracta es una subclase de la *Transformador de documentos* class y proporciona una implementación del método de transformación que crea un gráfico con los datos en el marcado XML que toma como entrada. Este método en particular invoca la *construir-*

ChartFromXML método abstracto para construir el gráfico que luego se encapsula en un marcado HTML. El diagrama informado es un árbol de varias subclases abstractas y el patrón de diseño del método de plantilla se aplica a cada nivel, porque el *buildChartFromXML* El método se descompone en operaciones más elementales, y cada una de ellas se descompone aún más dentro de las subclases. Los nodos hoja de este árbol son clases concretas *Gri fi nRichiestaPieChartBuilder, Gri fi nRisultatiPieChartBuilder* y *JobStatusChartBuilder:* las dos primeras clases se han desarrollado para construir los gráficos circulares en el caso de uso descrito en la siguiente sección; en cambio, la tercera clase se utiliza para crear el gráfico de barras horizontales que describe el historial de estado completo de un trabajo en el portlet Detalles de estado del trabajo. Las diversas implementaciones del *buildChartFromXML* Utilice JOFC2 Library [21] para construir el objeto Chart que se devuelve a la clase que invoca. Esta biblioteca es un contenedor de Open Flash Chart 2 [22], que es una biblioteca flash de código abierto que se utiliza para crear gráficos animados.

6. Estudio de caso

El caso de estudio para demostrar la e ffi La eficiencia y potencialidad de la GRB WAPI se basa en el proyecto Grid for Finance (Gri fi n) [23] financiado por el Ministerio de Investigación y Universidad de Italia (MIUR). El proyecto GriFin se centró en el desarrollo de un entorno de resolución de problemas de red para aplicaciones financieras. Las aplicaciones financieras son una de las áreas más interesantes para la comunidad científica. El desarrollo de e ffi cient y e ff Las herramientas de apoyo a la toma de decisiones eficaces en esta área son estratégicamente importantes para las instituciones financieras, ya que podrían brindar grandes oportunidades en términos de mejora de las aplicaciones existentes y desarrollo de nuevos servicios con un mayor nivel de personalización. La aplicación financiera nos brindó un caso de uso complejo y realista para desarrollar un entorno de resolución de problemas de red basado en la tecnología GRB WAPI descrita en las secciones anteriores.

–

Después de la de fi nición de requisitos y restricciones específicas, a través del portal grid, los operadores financieros pueden obtener sugerencias sobre la composición de la cartera a lo largo del horizonte de planificación de inversiones, de acuerdo con la actitud de aversión al riesgo del inversionista. Las soluciones se trazan como gráficos que representan:

- El e ffi frontera ciente que representa soluciones en términos de medida de riesgo y riqueza final en función de la actitud de aversión al riesgo.
- La evolución de la cartera a lo largo de cada escenario.

Antes de acceder a los servicios del portal, el usuario debe autenticarse a través del portlet de inicio de sesión. La autenticación de usuario se basa en Shibboleth, una herramienta de software para el inicio de sesión único web. Permite recuperar información sobre la identidad del usuario y tomar decisiones de autorización informadas para el acceso individual a recursos en línea protegidos de manera que se preserve la privacidad. Un proveedor de servicios de Shibboleth, instalado en el servidor del portal, es responsable de redirigir la solicitud del usuario a la URL externa de un servicio de descubrimiento, que a su vez permite al usuario seleccionar un proveedor de identidad. Una vez que el usuario proporciona su nombre de usuario y contraseña, el proveedor de identidad recupera algunos datos del usuario (ID de usuario, correo electrónico, nombre, apellido, etc.) y luego lo redirige al sitio del portal. A partir de ese momento, el usuario se autentica y la aplicación web puede recuperar sus datos del encabezado de solicitud correspondiente. El portlet Gri fi n es el portlet que el usuario debe utilizar para enviar un trabajo de la aplicación Gri fi n. Tres pestañas componen la interfaz de usuario del portlet: instrumentos, parámetros y reanudación. En la pestaña Instrumentos, el usuario puede seleccionar instrumentos, especificando también un valor invertido, un límite inferior y un límite superior. En la pestaña de Parámetros el usuario puede establecer el efectivo inicial, un eventual retorno mínimo y un período de tiempo en meses; También puede especificar como restricción el riesgo máximo para contrapartes, categorías y países. La pestaña Reanudar resume para el usuario los datos que ha insertado en las pestañas anteriores, para que pueda verificarlos antes de enviar el trabajo. Usamos el El portlet Gri fi n es el portlet que el usuario debe utilizar para enviar un trabajo de la aplicación Gri fi n. Tres pestañas componen la interfaz de usuario del portlet: instrumentos, parámetros y reanudación. En la pestaña Instrumentos, el usuario puede seleccionar instrumentos, especificando también un valor invertido, un límite inferior y un límite superior. En la pestaña de Parámetros el usuario puede establecer el efectivo inicial, un eventual retorno mínimo y un período de tiempo en meses; También puede especificar como restricción el riesgo máximo para contrapartes, categorías y países. La pestaña Reanudar resume para el usuario los datos que ha insertado en las pestañas anteriores, para que pueda verificarlos antes de enviar el trabajo. Usamos el El portlet Gri fi n es el portlet que el usuario debe utilizar para enviar un trabajo de la aplicación Gri fi n. Tres pestañas componen la interfaz de usuario del portlet: instrumentos, parámetros y reanudación. En la pestaña Instrumentos, el usuario puede seleccionar instrumentos, especificando también un valor invertido, un límite inferior y un límite superior. En la pestaña de Parámetros el usuario puede establecer el efectivo inicial, un eventual retorno mínimo y un período de tiempo en meses; También puede especificar como restricción el riesgo máximo para contrapartes, categorías y países. La pestaña Reanudar resume para el usuario los datos que ha insertado en las pestañas anteriores, para que pueda verificarlos antes de enviar el trabajo. Usamos el especificando también un valor invertido, un límite inferior y un límite superior. En la pestaña de Parámetros el usuario puede establecer el efectivo inicial, un eventual retorno mínimo y un período de tiempo en meses; También puede especificar como restricción el riesgo máximo para contrapartes, categorías y países. La pestaña Reanudar resume para el usuario los datos que ha insertado en las pestañas anteriores, para que pueda verificarlos antes de enviar el trabajo. Usamos el especifi

El portlet Detalles del estado del trabajo permite al usuario verificar algunos detalles del estado de sus trabajos y se activa cuando el usuario hace clic en la salida, el registro de visualización o la columna de estado del trabajo. Entonces, a través de este portlet el usuario

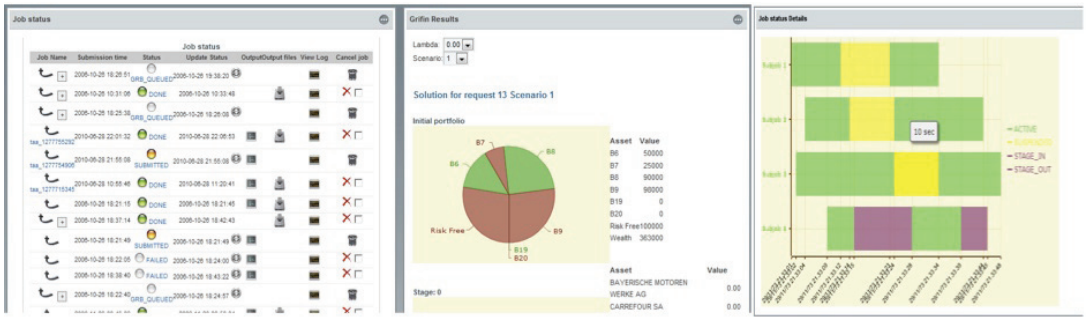

Figura 4: (a) portlet Estado del trabajo; (b) Portlet de resultados; (c) Portlet Detalles del estado del trabajo

puede descargar los archivos de salida de un trabajo enviado cuya ejecución ha finalizado, ver el registro del trabajo y, sobre todo, puede visualizar un gráfico de barras horizontales que muestra todo el historial de estado del trabajo.

7. Conclusiones y labor futura

En este artículo hemos descrito el diseño de un nuevo portal grid basado en una arquitectura multinivel: en particular se utilizó una capa back-end muy versátil, caracterizada por el conjunto de métodos remotos GRB WAPI, que permite utilizar varios lenguajes y tecnologías de programación en el capa frontal. Si bien la tecnología de portlets de Java se ha utilizado para desarrollar un caso de uso de finanzas en cuadrícula, se están desarrollando más bibliotecas vinculantes para varios lenguajes de programación (Ruby, PHP, etc.) con el fin de integrar la tecnología GRB WAPI en los marcos existentes.

El desarrollo de un portal de red basado en llamadas GRB WAPI también es un paso parcial hacia las tecnologías en la nube. Los proveedores de computación en la nube típicos ofrecen aplicaciones comerciales comunes en línea a las que se accede desde otro servicio web o software como un navegador web, mientras que el software y los datos se almacenan en servidores. GRB WAPI se puede ampliar para crear interfaces para acceder a aplicaciones SaaS (software como servicio) en entornos de nube.

Otros pasos en la dirección de la nube incluyen mejoras para hacer que el sistema sea más autónomo y transparente para los usuarios: como resultado final, el hardware, el software y los datos deben manejarse y reconfigurarse automáticamente dentro de una única plataforma que podría presentarse a los usuarios.

Referencias

[1] M. Cafaro, I. Epicoco, M. Mirto, D. Lezzi, G. Aloisio, "El motor de flujo de trabajo del corredor de recursos de la red", Concurrencia y cálculo: práctica and Experience, Número especial: 2do Taller internacional sobre gestión del flujo de trabajo y aplicaciones en entornos Grid (WaGe2007), Volumen 20, Número 15, págs. 1725 - 1739

[2] G. Aloisio, M. Cafaro, G. Carteni, I. Epicoco, S. Fiore, D. Lezzi, M. Mirto, S. Mocavero, "The Grid Resource Broker Portal", Concurrencia and Computation: Practice and Experience, Número especial sobre entornos de computación en red, Volumen 19, Número 12 (2007), págs. 1663-1670

[3] G. Aloisio, M. Cafaro, "Acceso basado en la web a la red utilizando el portal Grid Resource Broker", Concurrencia y Computación: Práctica y Experience, volumen 14, número 13-15 (2002), págs. 1145-1160, número especial sobre entornos de computación en red

[4] G. Aloisio, M. Cafaro, E. Blasi, I. Epicoco, "The Grid Resource Broker, un marco ubicuo de computación Grid", Journal of Scienti fi c Programación, Volumen 10, Número 2 (2002), págs. 113-119, Número especial sobre Grid Computing, IOS Press, Amsterdam

[5] J. Novotny, M. Russell, O. Wehrens, Gridsphere: Un marco de portal avanzado, en: EUROMICRO, IEEE Computer Society, 2004, págs. 412419.

[6] Página del proyecto GridPort, http://gridport.sourceforge.net/

[7] T. Suzumura, H. Nakada, M. Saito, S. Matsuoka, Y. Tanaka y S. Sekiguchi. 2002. El portal ninf: una herramienta de generación automática de grid portales. En Actas de la conferencia conjunta ACM-ISCOPE de 2002 sobre Java Grande (JGI '02). ACM, Nueva York, NY, EE. UU., 1-7.

[8] T. Suzumura, S. Matsuoka, H. Nakada, H. Casanova, "GridSpeed: A Grid Portal Generation Server", hpcasia, págs.26-33, High Performance Computing and Grid in Asia Pacific Region, Séptima Conferencia Internacional sobre (HPCAsia'04), 2004

[9] P. Kacsuk y G. Sipos: Flujos de trabajo de múltiples redes y múltiples usuarios en el portal P-GRADE, Journal of Grid Computing, vol. 3, n ° 3-4, Springer Editores, págs. 221-238, 2005

[10] BeSTGRID, https://www.bestgrid.org/

[11] Portal CCDB, http://ccdb-portal.crbs.ucsd.edu/

[12] Portal de la Universidad de Sydney, http://cima.chem.usyd.edu.au:8095/gridsphere/gridsphere [13] API de datos de Google, http://code.google.com/intl/it-IT/ apis / gdata /

[14] K. Seymour, H. Nakada, S. Matsuoka, J. Dongarra, C. Lee, H. Casanova, Una descripción general de GridRPC: una API de llamada de procedimiento remoto para Grid Informática

[15] Xuanhua Shi, Zhao Chen, Hai Jin, SongWu, Ke Fan, WAGA: un marco web flexible para aplicaciones de red, Universidad de Huazhong de Ciencia y Tecnología, Wuhan, China

[16] S. Cholia, D. Skinner, J. Boverhof, NEWT: Servicio ARESTful para la creación de aplicaciones web de Computación de alto rendimiento, Berkeley National Laboratorio, Berkeley, CA, EE. UU.

[17] Especificación de portlet http://jcp.org/aboutJava/communityprocess/ fi nal / jsr168 / index.html [18] Potlet Speci fi cation 2.0, http://jcp.org/en/jsr/detail?id=286

[19] http://www.liferay.com/documentation/social-o ffi ce / 1.5 / user-guide [20] RFC 1521, http://tools.ietf.org/html/rfc1521

[21] JOFC2, http://code.google.com/p/jofc2/

[22] Open Flash Chart 2, http://teethgrinder.co.uk/open- fl ash-chart-2 / [23] Portal del proyecto Gri fi n, http: //www.gri fi n.eu

[24] Protocolo de publicación Atom, http://bitworking.org/projects/atom/rfc5023.html [25] Formato de intercambio JSON, http://www.json.org/

[26] R. Fielding, REST: estilos arquitectónicos y el diseño de arquitecturas de software basadas en red, tesis doctoral, Universidad de California, Irvine, 2000, http://www.ics.uci.edu/ fi elding / pubs / dissertation / top.htm