UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

UM PORTLET GENÉRICO PARA CONSTRUÇÃO DE PORTAIS WEB PARA GRADES COMPUTACIONAIS UTILIZANDO O FRAMEWORK GRIDSPHERE

TRABALHO DE GRADUAÇÃO

Matheus Anversa Viera

Santa Maria, RS, Brasil

2008

UM PORTLET GENÉRICO PARA CONSTRUÇÃO DE PORTAIS WEB PARA GRADES COMPUTACIONAIS UTILIZANDO O FRAMEWORK GRIDSPHERE

por

Matheus Anversa Viera

Trabalho de Graduação apresentado ao Curso de Ciência da Computação
da Universidade Federal de Santa Maria (UFSM, RS), como requisito
parcial para a obtenção do grau de
Bacharel em Ciência da Computação

Orientador: Profª Drª Andrea Schwertner Charão

Trabalho de Graduação Nº 252
Santa Maria, RS, Brasil

2008

Universidade Federal de Santa Maria
Centro de Tecnologia
Curso de Ciência da Computação

A Comissão Examinadora, abaixo assinada,
aprova o Trabalho de Graduação

UM PORTLET GENÉRICO PARA CONSTRUÇÃO DE PORTAIS WEB PARA GRADES COMPUTACIONAIS UTILIZANDO O FRAMEWORK GRIDSPHERE

elaborado por
Matheus Anversa Viera

como requisito parcial para obtenção do grau de
Bacharel em Ciência da Computação

COMISSÃO EXAMINADORA:

Profª Drª Andrea Schwertner Charão
(Presidente/Orientador)

Prof. Dr. Benhur de Oliveira Stein (UFSM)

Prof. MSc. João Carlos Damasceno Lima (UFSM)

Santa Maria, 31 de janeiro de 2008.

AGRADECIMENTOS

Gostaria de agradecer primeiramente a Deus por ter me guiado ao longo desses 4 anos de faculdade e me ajudado a concluir o curso. Gostaria de fazer um agradecimento especial à minha mãe pelos ensinamentos e valores passados para mim e que com isso aprendi a nunca desistir, independente das dificuldades encontradas. Além disso, agradecer a ela por estar presente em toda a minha graduação (mesmo que por telefone), mas me ajudando e me incentivando sempre. Agradeço também às minhas irmãs, principalmente a Marjara, por me dar apoio de todas as formas durante os 4 anos de faculdade (além de me visitar e fazer festa comigo em Santa Maria). Além de agradecimento coloco este trabalho como uma forma de homenagem para meus avôs Dorvalino (in memorian) e Gentil (in memorian) que sempre acreditaram no meu potencial e me incentivaram a lutar pelos meus objetivos.

Agradeço também, aos professores que conseguiram transmitir o conhecimento durante o curso. Um agradecimento especial à professora Andrea, por me orientar durante 2 anos em pesquisas no LSC e no grupo PET, e nesse período me auxiliar no desenvolvimento de artigos científicos, linhas de pesquisa, sempre apresentando sugestões construtivas e principalmente por esclarecer e ajudar em todos os tipos de problemas (e pela paciência e compreensão durante esse tempo). Gostaria também de fazer um agradecimento para os professores Benhur e Caio, por serem meus orientadores em um dos projetos que participei. A esses professores, além dos professores Ceretta, Márcia, Cândia e Iara, gostaria de agradecer por serem amigos e por nos transmitirem conhecimentos que vão além de linguagens de programação, mas valores que podem ser levados para o resto da vida (agradecendo também as participações nos churrascos da turma).

Agradeço ao pessoal da minha turma, em especial ao Rodolfo, Cristiano, Tiago, Márcio, Leandro e Eduardo, que além de serem meus colegas de trabalhos e pesquisas, serem grandes amigos. Pelas parcerias de festas (e que festas), gostaria de agradecer o Rodolfo, o Cristiano e o Leandro, apesar de meu apartamento sempre ficar uma bagunça nas concentrações. Agradeço aos demais colegas da turma que de alguma forma fizeram parte de minha vida ao longo desses 4 anos.

Agradeço também a todos os meus amigos que estiveram comigo durante esse período, não podendo esquecer da Nathália que ingressou comigo no curso, mas acabou transferindo a faculdade para o curso de Medicina Veterinária. Entre meus amigos, um agradecimento especial para a Evelise, que contribuiu e é parte dessa minha conquista.

Enfim, agradeço a todos!

"Ninguém pode construir em teu lugar as pontes que precisarás passar para atravessar o rio da vida. Ninguém, exceto tu, só tu. Existem, por certo, atalhos sem número, e pontes, e semideuses que se oferecerão para levar-te além do rio, mas isso te custaria a tua própria pessoa: tu te hipotecarias e te perderias. Existe no mundo um único caminho por onde só tu podes passar. Aonde leva? Não perguntes, siga-o" — FRIEDRICH NIETZCHE

RESUMO

Trabalho de Graduação
Curso de Ciência da Computação
Universidade Federal de Santa Maria

UM PORTLET GENÉRICO PARA CONSTRUÇÃO DE PORTAIS WEB PARA GRADES COMPUTACIONAIS UTILIZANDO O FRAMEWORK GRIDSPHERE

Autor: Matheus Anversa Viera
Orientador: Profª Drª Andrea Schwertner Charão
Local e data da defesa: Santa Maria, 31 de janeiro de 2008.

O processamento paralelo e distribuído é usado atualmente como alternativa para problemas que necessitam de um grande poder computacional. Como ambiente para este tipo de processamento, destacam-se as plataformas de grades computacionais, que permitem que usuários separados geograficamente utilizem recursos computacionais provenientes de outros computadores que estejam ligados à grade. A utilização de ambientes de grades computacionais, entretanto, não é uma tarefa trivial, pois exige a utilização de ferramentas avançadas e que o usuário possua conhecimentos em processamento distribuído e sistemas operacionais. Esse problema pode ser contornado através de portais Web, que provêem uma interface amigável e de alto nível a serviços avançados de uma grade. Tipicamente, um portal permite que diferentes usuários autenticados configurem experimentos e acompanhem execuções de aplicações na grade, utilizando *portlets* especialmente concebidos para estes fins. A criação de *portlets* e portais é no entanto uma tarefa trabalhosa, que geralmente precisa ser repetida a cada nova aplicação a ser executada em uma grade. Nesse contexto, o presente trabalho propõe o desenvolvimento de um *portlet* genérico que gere portais para grades computacionais através do *framework* GridSphere. Através do GridSphere é possível desenvolver uma interface Web que traz facilidades de manuseio de processos executando em uma grade, bem como o monitoramento dos mesmos. Além disso, através da tecnologia de *portlets*, o GridSphere permite que os usuários configurem a aparência do portal e as funcionalidades que desejem acessar. Este trabalho pretende, portanto, facilitar a criação de portais que permitam que o acesso e a usabilidade de grades computacionais seja feita de forma simples e produtiva.

Palavras-chave: Grades computacionais, Portais de grades, GridSphere, Portlets.

ABSTRACT

Graduation Work
Graduate Program in Computer Science
Federal University of Santa Maria

A GENERIC PORTLET FOR BUILDING GRID PORTALS USING GRIDSPHERE FRAMEWORK
Author: Matheus Anversa Viera
Advisor: Prof[a] Dr[a] Andrea Schwertner Charão

Parallel and distributed processing is used nowadays as an efficient approach to problems that require high computing power. Computational grids stand out as an environment for such distributed execution. These platforms allow users that are geographically apart to use computer resources from other computers connected to the grid. However, using computational grids is not an easy task, because it requires the user to have some knowledge of distributed processing and operational systems, and also the use of advanced tools. Web portals that provide a friendly interface and a high level of advanced services to a grid can avoid this problem. Typically, a portal allows different authenticated users to configure experiments and to follow the execution of the application in the grid, using portlets specially designed for this purpose. The creation of portals and portlets is, however, a hard task, since it has to be repeated for every new application in a grid. In this context, this work proposes the development of generic portal able to generate portals to computional grids through the GridSphere framework. Through GridSphere it is possible to develop a Web interface that brings advantages in processes handling being executed in a grid, as well as in monitoring them. Besides this, through portlets technology, GridSphere allows users to configure the portal's look and the tools they wish to access. This work intends to facilitate the creation of portals that allow the access and use of computional grids to be done in a simple and productive way.

Keywords: computional grids, Portal of grid, GridSphere, Portlets.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	Aplication Programing Interface
MPI	Message Passing Interface
JSP	JavaServer Pages
JSR	Java Specification Request
JET	Java Emitter Templates
ASC	Astrophysics Simulation Collaboratory
JVM	Java Virtual Machine
XML	Extensible Markup Language

SUMÁRIO

1 INTRODUÇÃO

O processamento paralelo e distribuído é freqüentemente utilizado para resolver problemas que demandam um grande poder computacional ou processamento de grandes volumes de dados. A tecnologia de grades computacionais provê uma plataforma que permite a execução desses programas paralelos e distribuídos.

Computação em grade é um modelo amplamente utilizado para computação distribuída, que permite o compartilhamento de recursos computacionais entre usuários conectados através de uma rede de computadores (KESSELMAN; FOSTER, 1999). Entre esses recursos pode-se citar: processamento, memória, armazenamento de dados ou qualquer outro tipo de dispositivo que possa prover um poder computacional. A idéia principal em computação em grade é a utilização de um poder computacional, proveniente muitas vezes de computadores ociosos, fornecendo ao usuário da grade um acesso transparente a estes recursos, independente da localização geográfica dos computadores interligados.

Existem, atualmente, várias ferramentas que permitem explorar esta abordagem em computação distribuída, provendo um ambiente de grade, tais como Boinc (ANDERSON, 2004), OurGrid (OurGrid, 2008), XtremWeb (FEDAK et al., 2001), Globus Toolkit (I. Foster; C. Kesselman, 1997), entre outras. O acesso e manuseio dessas ferramentas por parte do usuário torna-se complexo, pois deve-se executar diversos comandos para poder iniciar um procedimento na grade. Em muitos casos, essa é um dos grandes problemas na utilização das grades, pois os usuários têm que possuir um grau de conhecimento elevado para não executar um comando errado e bloquear um procedimento. Quebrando essa barreira, conta-se atualmente com **portais para grades computacionais**, que além de facilitar o acesso por parte dos usuários, tornam as tarefas administrativas e de gerenciamento da grade mais eficientes. Esses portais têm por objetivo tornar ainda mais transparente o acesso a uma grade, muitas vezes sem necessitar a instalação de algum *software* no com-

putador pessoal, apenas pelo acesso a uma interface Web. Essa facilidade também passa a ser utilizada por parte do administrador, que pode fazer um controle da grade remotamente, apenas conectando-se ao portal. Além disso, informações úteis sobre o estado dos computadores disponíveis, andamento do processamento de uma requisição e velocidade da rede, passam a ser melhor disponibilizadas através desses portais.

Nesse contexto, há um crescimento na utilização de diversos *frameworks* para desenvolvimento de portais, tais como, Legion Grid Portal (NATRAJAN et al., 2002), Grid Portal Development Kit (NOVOTNY et al., 2002) e GridSphere (GRIDSPHERE PROJECT, 2007).

O GridSphere destaca-se por ser um *framework* de código aberto e por utilizar-se da tecnologia de *portlet*, que são componentes visuais dentro de uma interface Web que provêem pequenos aplicativos que podem gerar informações ou disponibilizar acesso a serviços. Recentes estudos (YANG, 2006; ZHANG; KELLEY; ALLEN, 2006) mostram as facilidades oferecidas pela utilização desta tecnologia, que fragmenta uma interface Web nesses pequenos componentes. Cada um desses componentes pode ser considerado como uma interface Web completa, e com isso podem ser adicionadas certas especificações, disponibilizando informações e acesso a serviços da grade.

Uma das motivações para o desenvolvimento deste trabalho é o fato de existir uma vasta quantidade de ferramentas para computação em grade. A construção de um portal para um ambiente de computação em uma grade específica pode trazer restrições quanto à utilização da mesma, pois algumas dessas ferramentas não possuem funcionalidades desejadas pelos usuários. Com a utilização do GridSphere e sua tecnologia de *portlets*, que são pequenos fragmentos Web que permitem visualizar informações ou acesso a recursos, o presente trabalho propõe-se a realizar o desenvolvimento de um *portlet* que funcionara como um gerador de novos portais conforme as especificações do usuário repassadas para esse *portlet*. O principal objetivo do mesmo é criar novos *portlets* para submissão de *jobs* em algumas grades previamente configuradas, ou aplicações que deseja-se executar distribuidamente. Após as informações do usuário serem submetidas para o *portlet* que funcionara como gerador, o mesmo irá criar automaticamente um *portlet* que estará pronto para ser utilizado para a grade que foi selecionada. Esses novos *portlets* irão interagir com ambientes de execução dessas grades já configuradas, tornando simples a interação do usuário com o ambiente distribuído. Portanto, o presente trabalho tem por objetivo,

criar pequenas aplicações (*portlets*) automaticamente através de uma interface Web.

O presente trabalho está dividido em 5 capítulos. O capítulo 2 descreve conceitos de portais para grades com uma rápida explanação sobre grades computacionais, continuando com portais para grade e apresentando o GridSphere. No capítulo 3 há uma descrição do projeto de desenvolvimento do *portlet* proposto, detalhando o ambiente a que se propõe sua execução, juntamente com uma descrição de seu projeto e sua implementação. A seguir, o capítulo 4 traz uma avaliação sobre um *portlet* gerado para a grade OurGrid, mostrando seus benefícios quando comparado a utilização da grade com sua interface original. Por fim, apresenta-se a conclusão do trabalho e algumas sugestões de atividades futuras.

2 PORTAIS DE GRADES: CONCEITOS E FERRAMENTAS

Esse capítulo apresenta uma breve revisão sobre grades computacionais, bem como uma explanação sobre portais para grades computacionais, citando tipos e algumas vantagens na sua utilização. Concluindo o capítulo é apresentado o *framework* GridSphere, mostrando sua estrutura e funcionamento.

2.1 Grades computacionais

As pesquisas tecnológicas, atualmente, demandam de um grande poder computacional. Isso deve-se ao fato de muitas simulações de problemas complexos estarem sendo realizadas em diversas áreas, e necessitarem de um processamento de grandes volumes de dados. Com isso, o processamento distribuído torna-se necessário para obter respostas mais rápidas aos problemas. Nesse contexto, a computação em grade demonstra-se uma alternativa promissora, pois utiliza-se, em muitos casos, do poder computacional de *cluster* conectados às grades, ou por ciclos ociosos de computadores pessoais.

A computação em grade surgiu da mesma forma que a Internet, com seu desenvolvimento nas comunidades acadêmicas e de pesquisa. A criação dela deu-se ao fato de pesquisas estarem sendo feitas de forma conjunta, porém com seus pesquisadores geograficamente longe um dos outros. Com isso, notou-se a importância de criar um ambiente que pudesse prover poder computacional, além de disponibilizar recursos e resultados de forma dinâmica.

Uma das primeiras ferramentas de destaque para processamento distribuído foi a biblioteca MPI (SNIR et al., 1998) (Message Passing Interface). Essa biblioteca servia para fornecer comunicação entre processadores, em muitos casos utilizando-se de *clusters* como ambiente. Com o passar do tempo, surgiram projetos de infra-estrutura que

visavam facilitar a utilização de recursos disponíveis nesses ambientes através da distribuição e coordenação de processos, dando início às grades computacionais. Entre esses projetos pode-se citar o Globus (I. Foster; C. Kesselman, 1997) e o Legion (NATRAJAN et al., 2002). Essas infra-estruturas passaram a ser aperfeiçoadas com a agregação de novas funcionalidades. Entre elas pode-se citar: escalonadores, que têm o objetivo de otimizar a distribuição de processos; monitoramento de recursos, tais como, memória, rede e processador.

Com isso, as grades computacionais podem ser consideradas como uma evolução da computação distribuída. De uma forma geral, computação em grade pode ser definida como vários usuários separados geograficamente e podendo utilizar recursos computacionais provenientes de outros computadores que não sejam os seus (KESSELMAN; FOSTER, 1999). Segundo Foster uma grade pode ser definida como uma infra-estrutura de computadores e programas que provêem um acesso confiável, coerente, difundido e econômico, com potencial computacional final (FORSTER; KESSELMAN, 1999).

Em meio a esse contexto, uma grade, deve ser confiável e transparente quando utilizada, não interessando onde serão processados ou armazenados os dados. Quando um usuário faz uma requisição ele obtém a resposta, sendo o processamento realizado em algum computador independente de sua localização geográfica.

Atualmente, há vários tipos de classificações para grades existentes (ALLEN et al., 2003), sendo que algumas merecem destaque. Primeiramente, as grades que preocupam-se com compartilhamentos de informações de forma distribuída (*Data-Grids*), nas quais os dados são armazenados e acessados de forma transparente quanto a sua localização. Outro tipo de grade que pode-se citar são as *Grids* computacionais, as quais preocupam-se em agrupar o máximo de recursos ou compartilhar ciclos de processamento para aumentar o poder computacional. Uma outra classificação para grades são as chamadas *Grids* interativos, que permitem que o usuário relacione-se com as aplicações que estão executando na grade.

2.2 Portais de grade computacionais

Atualmente, existe uma grande quantidade de plataformas que fornecem ambientes para computação em grade. Porém a utilização desses ambientes, em diversos casos, não é uma tarefa trivial. As grades computacionais muitas vezes possuem suas particularidades

quando se refere a submeter tarefas ou obter informações. Somente usuários experientes acabam por utilizar esse poder de computação distribuído, e após um estudo detalhado do funcionamento e peculiaridades de cada grade.

Em meio a esse cenário surgem os chamados portais para grades. Esses portais têm por objetivo disponibilizar uma plataforma de integração para os usuários finais, fornecendo serviços e recursos através de uma interface Web (NOVOTNY et al., 2002). Através dessa interface não há mais necessidade de utilizar linhas de comandos através de terminais para poder executar algum processo na grade.

Além de trazer a facilidade de os usuários poderem lançar suas tarefas em uma grade, os portais têm por objetivo mostrar detalhes a respeito da grade. Informações como número de CPU's disponíveis, velocidade de carga da rede, entre outras, podem ser úteis para tomada de decisões que melhorem o desempenho de utilização da grade. Portanto, o objetivo desses *frameworks* é auxiliar no desenvolvimento de portais que tragam um ambiente Web seguro que agregue e compartilhe conteúdo - informação, serviços e aplicações - com usuários de uma forma geral (SUN MICROSYSTEMS, 2002). Ainda, os portais provêm um único ponto de acesso a recursos computacionais, tais como: *clusters*, servidores de dados, aplicações, instrumentos científicos e a serviços computacionais (FOSTER, 2002).

Uma das vantagens desses portais possuírem uma interface Web, é que os usuários já estão familiarizados com esses ambientes. As tarefas de submissão de *jobs*, visualização de estado de um recursos na grade, entre outras, acaba tornando-se intuitiva. Usuários com pouca experiência na utilização de computação em grade podem utilizar esse processamento distribuído de forma simples.

Outra vantagem da utilização desses portais é que a submissão de tarefas e visualização de informações pode ser feita somente através da interface Web. A Web tornou-se um aspecto atrativo para o desenvolvimento desses portais porque é relativamente baixo o custo para adição de novas funcionalidades. Qualquer computador do tipo desktop possui navegadores instalados e a maioria deles possuem suporte a recursos multimídia e conteúdo dinâmico através de *plugins*. Com isso, não há necessidade de instalação de qualquer tipo de *software* no computador pessoal para utilizar a grade, bastando apenas acessar o portal.

Outro mecanismo interessante que os portais oferecem é a validação ao uso do por-

tal, e conseqüentemente ao uso da grade. Restrições podem ser feitas para usuários, e com isso consegue-se uma melhor administração. Consegue-se controlar quais tarefas podem ser submetidas, além dos administradores obterem estatísticas sobre o uso do portal, podendo tomar decisões visando um melhor aproveitamento da grade. As tarefas administrativas tornam-se, também, facilitadas através do portal.

Entre os portais para grade existentes, há dois tipos de classificação (YANG, 2006): baseados em *portlets* e não-baseados em *portlets*.

- **não-baseados em portlets**: esses tipos de portais provêem um acesso uniforme aos recursos da grade. Normalmente eles são baseados em: navegadores; aplicações que fazem requisições HTTP ao servidor através de *browser* do cliente; recursos *backend* que incluem recursos computacionais, base de dados, etc. Um exemplo que pode ser citado é o portal *Astrophysics Simulation Collaboratory* (ASC) (RUSSELL et al., 2002).

- **baseados em portlets**: *portlets* são componentes Web fragmentados que podem ser visualizados, quando em conjunto, como uma interface Web completa. Em outras palavras, eles são especificamente projetados para serem agregados em um contexto de uma página composta. Cada *portlet* produz fragmentos de página que são combinados com outros fragmentos de outros *portlets*, ao qual todos juntos formam o portal. Um exemplo de *framework* para portais baseado em *portlets* é o GridSphere (GRIDSPHERE PROJECT, 2007).

2.3 Framework GridSphere

O projeto GridSphere foi desenvolvido como parte do projeto GridLab (GRIDLAB, 2006), fundado pela European Commission. É um projeto de código aberto, e tem como principal finalidade a criação de um *framework* para o desenvolvimento de portais para grade computacional baseado no conceito de *portlet* (GRIDSPHERE PROJECT, 2007).

Dois modelos de desenvolvimento de *portlet* são suportados pelo GridSphere. Os modelos garantem que sejam desenvolvidos *portlets* e executem em software da IBM, Sun, Oracle, entre outras. Esses modelos são baseados nas API's IBM WebSphere v4.1 e JSR (Java Specification Requests) 168 (PROCESS SPECIFICATION, 2003). Essas especificações tornaram-se necessárias devido a um grande número de organizações implantarem portais.

Com isso a utilização do *framework* GridSphere possibilita a construção de *portlets* Web dentro dessas especificações, para que possam ser utilizadas por outros desenvolvedores.

Como exemplo de *portlet*, pode-se visualizar a figura 2.1.

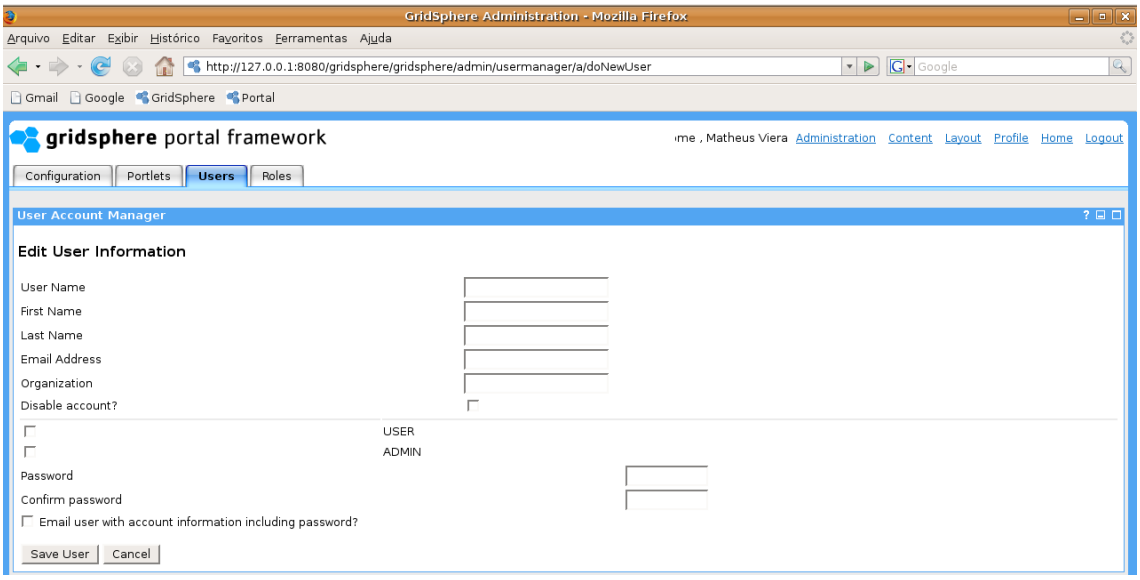

Figura 2.1: Exemplo de *portlet* GridSphere

Esse *portlet* encontras-se disponível com o *framework*, integrando suas ferramentas administrativas. Através das informações repassadas, pode-se criar novos usuários que irão acessar o portal e definir suas permissões de utilização, escolhendo se ele vai ser do tipo usuário (USER) ou administrador (ADMIN). Pode-se observar na figura, as abas *Configuration*, *Portlets* e *roles*, que são *portlets* pertencentes ao *framework* GridSphere, também fazendo parte das tarefas administrativas do portal.

2.3.1 Arquitetura do GridSphere

O *framework* GridSphere é executado através de um servidor *Tomcat*. O *Tomcat* surgiu dentro do projeto Apache Jakarta e é um *software* de código aberto que prove um servidor de aplicações Java para Web. Esse servidor permite a utilização das tecnologias Java Servlet e JavaServer Pages (JSP) (THE APACHE SOFTWARE FOUNDATION, 2007).

Os componentes do GridSphere podem ser divididos basicamente em portal, *portlets* e *container*.

O **portal** é uma interface que agrega todas as aplicações do tipo *portlets*. O portal permite também que os usuários possam editar as configurações de visualização e incluir

ou retirar *portlets* conforme sua escolha. Já os *portlets* são os componentes Web que fazem parte do portal. Eles podem fornecer informações ou prover recursos, sendo os mesmos gerenciados pelo *portlet container*. A interface destes *portlets* é desenvolvida através *GridSphere UI Beans*, que faz uma combinação entre bibliotecas de *tags* e *beans*, tornando fácil a construção deles. O *container* localiza-se entre o portal em si e os *portlets* e fornece um ambiente de execução para os *portlets*.

A arquitetura do GridSphere é do tipo cliente/servidor e pode ser visualizada na figura 2.2.

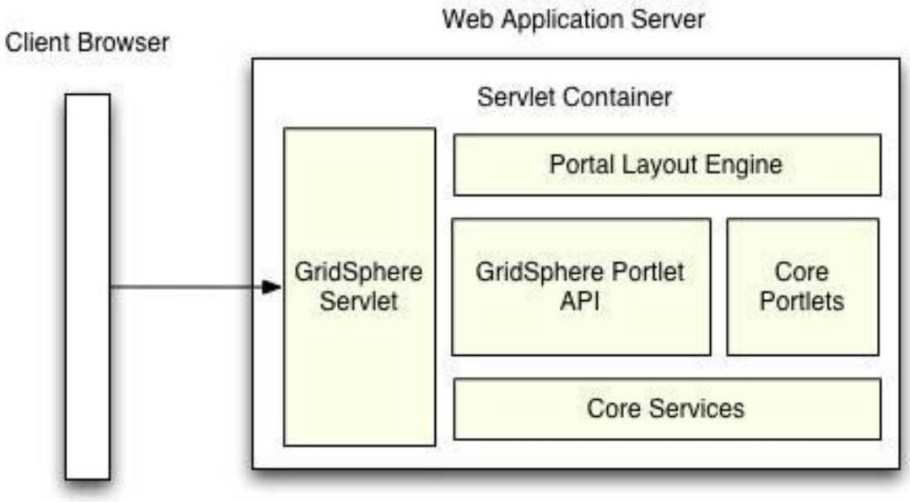

Figura 2.2: Arquitetura do Portal GridSphere (NOVOTNY; RUSSELL; WEHRENS, 2004)

O GridSphere *servlet* e o *Portal Layout Engine* fazem uso dos *Core Portlets* e *Core Services*. Através de um navegador, o usuário conectado ao portal faz uma requisição. Com isso o GridSphere *servlet* é invocado, e após, através do *Portal Layout Engine* os dados são formatados para a saída padrão do usuário.

2.3.2 *Portlets*

O conceito de *portlet*, já citado anteriormente, é uma aplicação em tecnologia baseada em componentes Web visuais que podem servir tanto para fornecer informações através de aplicações ou prover ao usuário acesso a outros tipos de serviços. Eles são gerenciados por um *portlet container* que processa requisições e gera conteúdo dinâmico. Esses *portlets* são definidos por uma API padrão e fornecem um modelo para o desenvolvimento de novos componentes de portais que podem ser compartilhados e trocados por vários *por-*

tlets containers, estando eles dentro da especificação JSR 168 ou API's IBM WebSphere v4.1.

O GridSphere através da tecnologia de *portlets* permite que haja uma interação por parte dos usuários com o portal. Os *portlets* podem ser editados para alterações no que refere-se a sua aparência e certas funcionalidades, desde que os mesmos permitam configurá-los. Eles também podem ser minimizados dentro do portal, ocultando a sua visualização. Outro aspecto interessante é que novos *portlets* podem ser adicionados ao portal, fornecendo novas funcionalidades para o usuário.

A figura 2.3 mostra detalhes da hierarquia e diagrama de seqüência, de um *portlet* seguindo a especificação JSR 168.

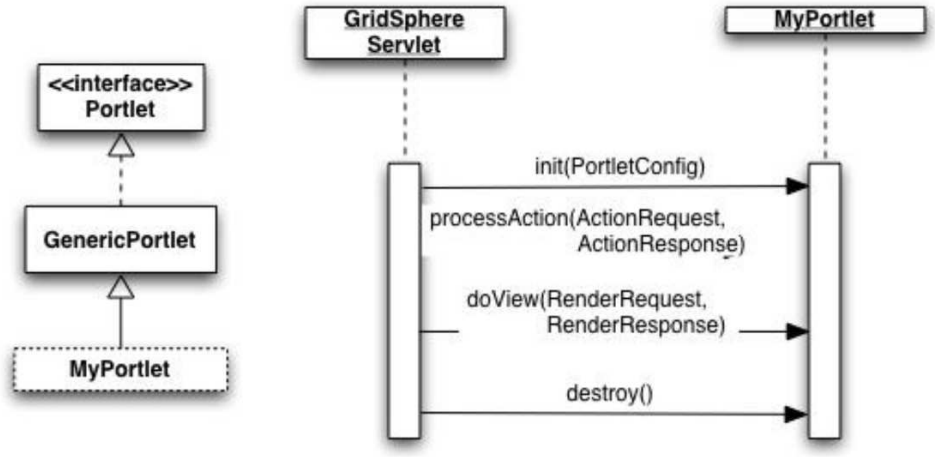

Figura 2.3: Portlet JSR 168 hierarquia e diagrama de seqüência (NOVOTNY; RUSSELL; WEHRENS, 2004)

Conforme a figura 2.3, os *portlets* possuem métodos de ciclo de vida. O método *init()*, que inicializa as configurações do *portlet*, busca as informações de configuração em um arquivo descritor.

Existe também métodos *processAction()*, que respondem a uma ação do usuário. Quando essa ação é executada, o método é invocado para executar uma ação na forma *back-end*, como por exemplo, uma atualização de uma base de dados. Essa ação realizada é apenas para o *portlet* que recebeu o evento de ação, independente do número de *portlets* existentes no portal.

Exitem, também, os métodos *Renders*, que fornecem um acessório de otimização para os *portlets*, quando não há necessidade de executar uma ação. Através do *container* há

uma gerência sobre esses.

O *container* também gerencia outros controladores que representam o estado de um *portlet*, que são o *portlet mode* e *window state*. Esses *portlets* invocam os métodos do tipo *Renders* que fornece um acessório de otimização para os *portlets*, quando não há necessidade de executar uma ação.

Por fim, o método *destroy()* libera os recursos inicializados com o método *init()*.

Um *portlet* pode ser implementado como uma subclasse de *GenericPortlet* e ter seus métodos customizados conforme as necessidades do usuário.

A especificação de *portlets* permite que uma aplicação do tipo *portlet* possa ser empacotada num arquivo .war e implantada em servidor de aplicação J2EE. No diretório WEB-INF está localizado o arquivo descritor *portlet.xml* para esse *portlet*.

O GridSphere possui os chamados *core portlets* com funcionalidades básicas que podem ser fornecidas ao portal. Entre elas pode-se citar, segundo Novonty (NOVOTNY; RUSSELL; WEHRENS, 2004):

- *Login*: permite o acesso de usuário, mediante nome ou e-mail e senha;
- *Logout*: realiza a saída do portal;
- *Account Request*: interface que o usuário pode requisitar uma nova conta e algumas configurações de grupo que deseja juntar-se;
- *Account Management*: somente usuários com permissão de administração podem acessar esse *portlet* que permite controlar características das demais contas de usuários;
- *User Management*: permite aos administradores aprovar ou recusar requisições de novas contas ou filiações em grupos;
- *Layout Configuration*: Configurações de visualizações de *portlets*, como localização e aparência;
- *Portlet Subscription*: através desse *portlet* os usuários podem adicionar ou remover *portlets* de seu espaço de trabalho;
- *Local File Manager*: os usuários, através deste recurso, têm acesso a um sistema de arquivos virtual onde podem editar, enviar e copiar arquivos para o portal;

O GridSphere utiliza o conceito de controle de acesso baseado em papéis (*role based access control*) (GRIDSPHERE PROJECT, 2007). Os usuários recebem um papel que define o grau de liberdade para realizar operações dentro do portal. A criação de usuários foi previamente mostrada na figura 2.1 e ilustra o tipo de usuário que pode ser criado. Grupos de usuários também podem ser criados sendo que esses grupos possuem um nome, uma descrição e os *portlets* que fazem parte do grupo. Como cada usuário pode ter acesso a mais de um grupo, a sua liberdade para realizar ações dentro do portal vai depender do seu papel.

3 DESENVOLVIMENTO DO PORTLET

Este capítulo descreve questões de projeto relacionadas ao desenvolvimento do *portlet* proposto e da geração do novo *portlet*. Primeiramente apresenta-se uma visão geral do comportamento do *portlet* inicial, descrevendo seu comportamento e a arquitetura do sistema. Na seqüência, apresenta-se a implementação do *portlet*.

3.1 Visão Geral

3.1.1 Comportamento do portlet

O *portlet* inicial provê uma interface gráfica em que o usuário terá a possibilidade de moldar seu novo portal através de campos em que ele poderá preencher com informações como: nome do novo portal, caminho para o *job* e etc. Essa interface inicial do *portlet* não poderá ser alterada, mas trará um grau de liberdade para usuário poder moldar seu novo portal.

O acesso a esse *portlet* é feito somente por usuários cadastrados no portal GridSphere. O *portlet* que será gerado através das informações do usuário, fornecidas ao *portlet* inicial, também só poderá ser utilizado por usuários cadastrados. Com isso, as tarefas administrativas tornam-se facilitadas.

3.1.2 Arquitetura do sistema

Um grande problema existente é a geração do novo portal de forma dinâmica. Buscou-se resolver esse problema de duas maneiras. Uma delas, consta na construção de uma classe que, assim que o *portlet* inicial armazena as informações na base de dados, ela busca essas informações e molda-se como o novo *portlet*. Uma das desvantagens nessa abordagem é que toda vez que o novo portal fosse inicializado, uma busca na base de dados deveria ser feita e novamente gerado o portal.

A outra solução encontrada, e que foi implementada como arquitetura do sistema, é utilizar-se de um gerador de códigos através de *templates*. Com isso, códigos JSP e arquivos fontes de Java serão gerados e após compilados para *bytecode* para serem interpretados pela JVM. Com isso o novo portal passa a ser um *portlet* já implementado conforme as construções de *portlets* através do GridSphere, e quando for inicializado não necessitará buscar informações sobre o seu comportamento na base de dados.

A figura 3.1 ilustra o funcionamento do ambiente para geração dos códigos.

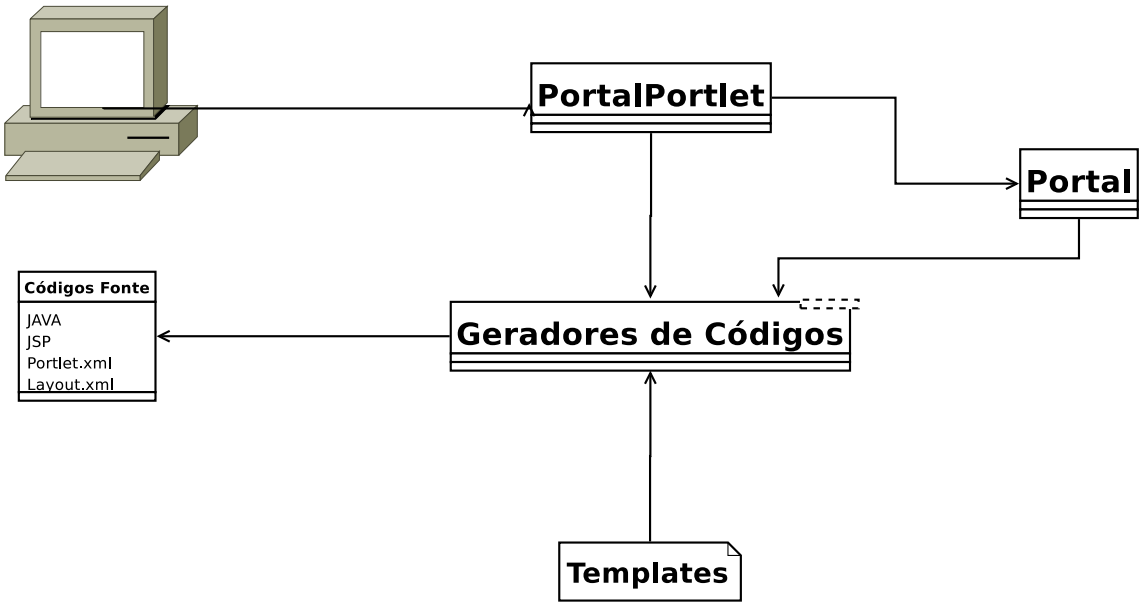

Figura 3.1: Gerador de códigos

Os geradores de códigos, mostrado na figura, são classes construídas através da API JET (Java Emitter Templates). Através do JET é possível construir *templates* para geração de códigos Java, XML, SQL, etc. Esses *templates*, geram classes Java que permitem a criação de códigos fontes.

Conforme a figura 3.1, primeiramente o *portlet* coleta as informações do usuário para a geração do novo *portlet* e as armazena em uma classe chamada *Portal*. Após, classes previamente criadas através dos *templates*, são instanciadas e coletam as informações contidas na classe *Portal*, referente ao que o usuário digitou. Como saída, arquivos de código fonte Java e JSP são gerados para compor o novo *portlet*. Também são gerados arquivos essenciais para a criação de um novo projeto que criará o novo *portlet*.

Outros pontos importantes a serem observados, ainda em se tratando da arquitetura do sistema, referem-se a cuidados com a execução de aplicações em ambientes distribuídos e às particularidades existentes em grades computacionais. Aplicações sendo executadas

em ambientes distribuídos, podem ser realizadas através de programas paralelos executados em pequenas tarefas distribuídas entre os processadores que compõem o ambiente. Outra possibilidade refere-se à execução de aplicações em que alterna-se os dados de entrada do programa. Em ambos os casos existem preocupações quanto à parâmetros a serem fornecidos para a execução do programa.

Pela existência de uma grande quantidade de ferramentas de grades computacionais, encontra-se também particularidades entre elas. Algumas, por exemplo necessitam que o tipo de sistema operacional seja passado como parâmetro quando for executar uma tarefa na grade. Outras mantêm a configuração e a execução de *jobs* através da edição de arquivos. É importante que esses detalhes referentes às particularidades das grades sejam abstraídos dos usuários que irão utilizar o portal.

3.2 Implementação

O processo de desenvolvimento do gerador de códigos está sendo implementado utilizando a API JET. Inicialmente, gera-se um arquivo *template* que servirá como um descritor do funcionamento de uma classe. Esse arquivo possui uma linguagem de marcação simples, similar a do JSP. Utilizando o JET Builder, o arquivo *template* é traduzido para uma classe Java. Essa nova classe, descreve a estrutura de comportamento do código a ser gerado. Nela, é criado automaticamente um método *generate*. Esse método funciona como o gerador de código que foi descrito no aquivo *template*. A construção dos arquivos *templates* permite que qualquer recurso da linguagem Java seja utilizado, pois esse arquivo será traduzido para uma classe Java. Com isso, foi necessário que classes fossem instanciadas dentro desses arquivos, que após traduzidos, permitiram que informações provenientes dos atributos de classes fossem utilizadas.

A fim de ilustrar o comportamento do *template*, a figura 3.2 mostra um trecho de código desse arquivo:

As *tag's package, imports*, e *class*, na figura 3.2, contém as informações da classe que irá ser gerada através do JET Builder.

Pode-se ainda observar na figura 3.2, que a classe *Portal* foi instanciada dentro do arquivo *template*, e com isso, o método *generate* da classe geradora de códigos, conterá essa classe instanciada, podendo acessar os métodos. No trecho de código mostrado, os nomes do *package* e da classe serão obtidos da classe *Portal*, que contêm as informações

```
<%@ jet package="templates" imports="java.util.*
    org.gridsphere.portal.portlets.Portal"

class="Classtemplate" %>

<%Portal portal = new Portal();%>
package <%stringBuffer.append(portal.getPackage());%>;

public class <%stringBuffer.append(portal.getNomeClasse());%>
                  extends ActionPortlet {
    public void init(PortletConfig config) throws PortletException {
    ...
```

Figura 3.2: Exemplos de código template

que foram digitadas pelo usuário.

Seguindo esse padrão, as informações necessárias para a geração do código fonte Java e JSP foram criadas através de arquivos *templates* para gerarem as classes geradoras. Foram construídos *templates* para a criação dos arquivos: *portlet.xml* e *loggedin.xml*. O arquivo *portlet.xml* contém as informações necessárias para que o *portlet* possa ser exibido dentro no GridSphere. Essas informações são: descrição do *portlet*, nome do *portlet*, a classe do *portlet*, e o seu título. Além dessas informações, algumas outras, necessárias para o funcionamento do *portlet*, são acrescentadas conforme o padrão do GridSphere. O arquivo *loggedin.xml*, contém as informações sobre os *portlets* que estão sendo executados no portal. Para isso faz-se necessário alterar esse arquivo, acrescentando as informações sobre o novo *portlet*, para que ele seja visualizado e executado também após a sua criação.

Outra maneira de inserir o novo *portlet*, para que possa ser executado, é selecionando a opção de *layouts* contida dentro do portal. É a mesma maneira que editar o arquivo *loggedin*, mas através de uma interface. Porém, com isso o novo *portlet* não seria criado de forma dinâmica.

A figura 3.3 mostra o *portlet* implementado pela classe *PortalPortlet*, que é responsável por adquirir as informações sobre o novo *portlet*. Esse *portlet* inicial é uma interface Web construída através de uma página JSP, desenvolvida com o *GridSphere UI Beans*. Essa página JSP é interpretada pelo servidor *TomCat*.

Os campos "Nome do projeto"e "short name"são utilizados para a criação do novo projeto através da ferramenta *ant*, utilizada pelo GridSphere. Os outros campos são informações referentes ao nome do novo *portlet* e nome da classe Java e JSP a ser gerado. Essa

Figura 3.3: *Portlet* Inicial

interface conta também com opções de escolha dos *middlewares* de grade XtremWeb e OurGrid, ou programas MPI. Essas opções são suportadas pelo *portlet* gerador, para criação do novo *portlet*. As particularidades de cada um dos *middlewares* suportados e os programas MPI, serão descritos na subseção seguinte.

3.2.1 Suportes oferecidos pelo portlet gerador

O XtremWeb (CAPPELLO et al., 2005) foi desenvolvido pelo *Laboratoire de Recherche en Informatique da Université Paris-Sud*, na França. É uma ferramenta de código aberto para pesquisa em torno de plataformas de computação distribuída, incluindo grades, computação global e sistemas *peer-to-peer*. O XtremWeb é divido em três componentes (XtremWeb, 2008): servidor, cliente e trabalhador. O servidor é a parte central, responsável pela gerência do sistema, recebendo as tarefas dos clientes e escalonando para os trabalhadores. O cliente é a interface do usuário com o sistema, responsável pela submissão de aplicações, podendo também visualizar informações sobre a execução de aplicações na grade. O trabalhador é a parte distribuída do sistema, responsável pela execução das tarefas.

Para a execução de uma aplicação no XtremWeb faz-se necessário carregar primeiramente a aplicação no cliente. Para isso necessita-se saber o sistema operacional que o XtremWeb está rodando e a arquitetura a qual o código binário da aplicação foi compilado. Essas informações, além do caminho para o código do cliente, são solicitadas quando a opção XtremWeb for selecionada conforme a figura 3.3. Optou-se por deixar padrão para "linux"o valor do sistema operacional "ix86"o valor do campo da arquitetura que a aplicação foi compilada. Para a execução de uma aplicação, ainda é necessário definir uma tarefa com essa aplicação, selecionando a aplicação e os parâmetros utilizados por ela. Mas isso não fica a cargo do *portlet* gerador, e sim do *portlet* gerado para essa

grade.

A grade OurGrid é diferente em alguns aspectos quando comparado ao XtremWeb. O OurGrid (OurGrid, 2008) é uma ferramenta de código aberto desenvolvida pela Universidade Federal de Campina Grande, para execução de aplicações em que não exista comunicação entre as tarefas, conhecidas como *bag-of-task*. O OurGrid é divido em três componentes: MyGrid, Peer e User Agent. O Mygrid é a interface do usuário responsável pela execução e monitoramento dos *jobs*. O Peer é o responsável pela descoberta de trabalhadores e alocação de trabalhos para os mesmos executarem. O User Agent é o responsável por fazer com que um computador funcione como trabalhador. Ele também define os mecanismos de transferência de arquivos e execução remota entre os trabalhadores e o computador que executa o MyGrid.

Uma das diferenças do OurGrid em relação ao XtremWeb é que no OurGrid, as configurações são feitas através de arquivos de descrição. Para criar a grade, executar *jobs* e etc., é necessário modificar ou criar alguns arquivos. Quando a opção de grade OurGrid é selecionada no *portlet* gerador, conforme a figura 3.3, o caminho absoluto onde o MyGrid está instalado e o tipo de sistema operacional que a grade está configuradas são solicitados para o usuário. Por padrão, deixou-se o campo do sistema operacional com o valor "linux". Com essas informações já pode-se gerar um *portlet* para a grade OurGrid.

Em caso de escolha da opção MPI, nenhuma informação extra é solicitada. Com o *portlet* gerado para rodar programas MPI, as informações de parâmetros para aplicação, entre outras, são solicitadas no novo *portlet*.

3.2.2 Geração do novo portlet

Após o preenchimento dos campos e da escolha do tipo de grade que o novo *portlet* irá dar suporte, as informações são repassadas para a classe *PortalPortlet*, através de um botão submeter. Esse botão dispara um método do tipo *ProcessAction* que responde a essa ação. Dentro desse método, a classe *Portal* é então instanciada e armazena todas as informações repassadas pelo usuário. Ao final do método *ProcessAction*, correspondente a submissão das informações para o *PortalPortlet* a classe *Executa* é então instanciada e recebe como parâmetro o objeto *Portal*. Essa classe é responsável por realizar o processo de criação dos códigos fontes, criação do novo projeto e arquivos necessários para o funcionamento do novo *portlet*.

Inicialmente a classe *Executa* instancia a classe *geradorCodigoJava*, que como o próprio nome diz, é a classe responsável por gerar o código fonte Java do novo *portlet*. Essa classe foi criada através dos *templates* e recebe como parâmetro para o método *generate*, o objeto *Portal*. Com isso o gerador de códigos verifica as informações contidas no objeto *Portal* e cria o novo código fonte Java, conforme o que foi repassado para o *PortalPortlet* e a escolha de grade ou MPI. Esse novo arquivo gerado é armazenado em um diretório e, posteriormente, será movido para o diretório pertencente ao projeto do novo *portlet*. A classe *Executa* então cria uma instância da classe *geradorCodigoJSP*, também passando como parâmetro para o método *generate*, o objeto *Portal*. Com isso tem-se os códigos fontes Java e JSP do novo *portlet*.

A arquivo *portlet.xml* é gerado através da uma classe *geradorPortletXML* de maneira semelhante aos outro geradores. Esse arquivo, como citado anteriormente, descreve algumas informações necessárias para o funcionamento do novo *portlet*. Entre essas informações está o pacote e nome da nova classe Java gerada através da classe *geradorCodigoJava*.

A classe *Executa*, então, adiciona informações ao arquivo *loggedin.xml*. Esse arquivo contém as informações sobre os *portlets* que estão sendo executados dentro do portal GridSphere. Faz-se então necessário adicionar as do novo *portlet*. Inicialmente, o método *geraLayoutXML*, faz uma cópia do arquivo *loggedin.xml* para o diretório que está armazenando os arquivos criados através dos geradores. O método então instancia a classe *geradorLayoutXML* e cria as informações necessárias para que o novo *portlet* possa ser executado, armazenando-as em uma *String*. Após, o arquivo *loggedin.xml* é percorrido até o local onde essas informações devem ser adicionadas e então acrescenta-se o código gerado ao arquivo.

Por fim, o método *geraInformacoes* da classe *Executa*, adiciona algumas informações necessárias ao arquivo *build.properties* contido no diretório do GridSphere.

Com todos os códigos fontes gerados, arquivos e informações necessárias, a classe *Executa* executa um *script* que irá criar o novo projeto no GridSphere. Quando for gerado o novo *portlet* através do comando *ant new-project* do GridSphere, o mesmo fara uma leitura no arquivo *build-newproject.xml*, que é utilizado para descrever o processo de construção e suas dependências. Esse arquivo foi alterado para que as informações que eram solicitadas para o usuário digitar sejam retiradas do arquivo *build.properties*,

manipulado pelo método *geraInformações* da classe *Executa*.

Após a criação, o script move os arquivos fontes Java e JSP, além do arquivo *portlet.xml* para dentro dos respectivos diretórios do projeto criado, e o arquivo *loggedin.xml* para o servidor *Tomcat* no local onde o portal GridSphere localiza-se. Então, o script executa os comandos necessários para que o novo projeto seja compilado e copiado para o servidor *Tomcat*. O script ainda interrompe a execução do servidor *Tomcat* e inicializa-o novamente para que as novas informações contidas nele sejam reconhecidas.

Com isso, o novo *portlet* estará executando no portal GridSphere. Um exemplo de um *portlet* criado pode ser observado na figura 3.4.

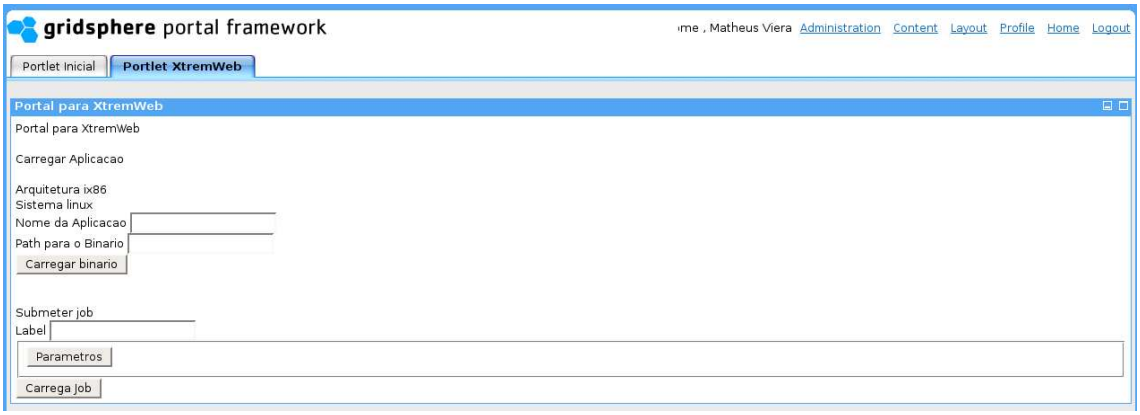

Figura 3.4: *Portlet* XtremWeb gerado

Foi escolhido o XtremWeb para geração do *portlet*. Conforme as informações repassadas para o *portlet* inicial pode-se observar na figura 3.4 a arquitetura e o tipo de sistema operacional ao qual a grade XtremWeb irá executar.

4 AVALIAÇÃO

Este capítulo tem como objetivo avaliar um *portlet* criado para a grade OurGrid. A fim de avaliar os benefícios trazidos pela utilização do *portlet*, foi utilizada uma aplicação na área de meteorologia.

Este capítulo está organizado em seções. Primeiramente é apresentado uma comparação entre as ferramentas que o *portlet* da suporte. A seguir, é exposto o funcionamento do OurGrid. Após é feita uma breve explanação da aplicação que vai executar através do *portlet*. Por fim, é mostrado o *portlet* executando sobre a grade Ourgrid, mostrando seus benefícios.

4.1 Comparativos dos middlewares e MPI

Conforme mencionado anteriormente, o *portlet* gerador da suporte aos *middlewares* para grade OurGrid e XtremWeb, além de permitir que seja criado um *portlet* para execução de programas MPI. As 3 ferramentas possuem algumas semelhanças quanto a submissão de *jobs*, ou no caso de MPI, a execução de programas no ambiente distribuído. O XtremWeb, contudo, necessita inicialmente que o código binário da aplicação seja adicionado ao sistema, para depois poder submeter o *job*. Diferentemente, o OurGrid necessita somente que o arquivo de descrição do *job* seja criado e submetido. O programas MPI são executados em ambiente distribuído através de um único comando.

As tabelas 4.1 e 4.2 apresentam os comandos necessários para execução de tarefas nas ferramentas.

O *portlet* gerador da suporte a criação de novos *portlets* para quaisquer uma das ferramentas expostas na tabela. Mas os testes foram realizados somente com o *middleware* OurGrid, pelo fato de *middleware* XtremWeb não estar devidamente configurado e por falta de tempo.

Tabela 4.1: Comandos necessários para carregar aplicação

	carregar aplicação
XtremWeb	–xwaddapp <nome> <cpu> <os> <binário>
OurGrid	não necessita
MPI	não necessita

Tabela 4.2: Comandos necessários para executar job

	submeter *job* (tarefa)
XtremWeb	–xwsubmit <nome> <label> <parâmetros>
OurGrid	mygrid addjob <job description file>
MPI	mpirun -np<num de processos> <binário> <argumentos>

4.2 OurGrid

Esta seção tem por objetivo descrever o funcionamento da ferramenta OurGrid, mostrando suas características e o que é necessário para a sua execução.

Como já descrito anteriormente, o OurGrid é dividido entre os componentes: MyGrid, Peer e User Agent. O MyGrid , interface com o usuário, conecta-se à grade definida pelo *Grid Description File*. Esse arquivo define os Peers responsáveis pela alocação de trabalhadores para execução dos *jobs*. A execução desses *jobs* é feita através da criação de um arquivo e adicionando-o a grade pelo Mygrid. O arquivo de descrição, (*Job Description File*), contem informações como: *label, init, remote e final*, além de outras referentes a sistema operacional, quantidade mínima de memória e etc..O *label* define um nome para o *job*. *Init* define os comandos que devem ser executados antes do inicio da execução do *job*, como a cópia da aplicação e seus argumentos para o trabalhador. *Remote* descreve os comandos que serão executados remotamente, como a própria execução da aplicação ou descompactação de algum arquivo. Já o *final* traz as informações referentes ao que deve ser executado após o termino do *job*, como cópia dos resultados para o cliente.

O Peer é responsável pela alocação dos trabalhos e descoberta dos trabalhadores que iram executar esses trabalhos. O mecanismo utilizado para isso é através do *Site Description File* (arquivo de descrição site). Esse arquivo define os trabalhadores que serão contatados na inicialização do Peer e traz algumas informações sobre sistema operacional e como as copias dos arquivos para os trabalhadores e execução do *job* é realizada.

O User Agent é quem possibilita que um computador funcione como trabalhador. Ele provê o acesso aos recursos da máquina para receber as tarefas, e após executa-las, enviar os resultados.

O OurGrid, portanto, possui sua interface de configuração através da criação ou edição de alguns arquivos. Com essa grade pré-configurada, a submissão de *jobs*, passa a ser feita através da criação do arquivo *Job Description File*. Com um ou mais *jobs* executando na grade, existe a possibilidade de visualizar o estado da grade através do MyGrid. Com a conclusão de um *job*, pode-se visualizar o resultado da execução através do arquivo que foi definido, dentro do *Job Description File*, para conter esse resultado.

4.3 Aplicação

A aplicação utilizada para os testes no novo *portlet*, foi desenvolvida no Laboratório de Micrometeorologia (LuMet) da Universidade Federal de Santa Maria. Aplicações nessa área geralmente demandam de um grande poder computacional, pois há um volume grande de dados a serem analisados e processados. A aplicação em questão, foi desenvolvida em Fortran77 e é um programa seqüencial que analisa e interpreta dados coletados por sensores micrometeorológicos (NEVES, 2006).

A aplicação processa dados coletados durante um dia em uma estação meteorológica, a uma taxa de 16 coletas por segundo. Em um dia, são coletadas 1.382.400 amostras, gerando um arquivo de aproximadamente 100MBytes. Esse arquivo é processado em partes, ou janelas, de 2^{15} linhas. Após o processamento de cada janela, os resultados da análise salvos em arquivos, sendo que ao final do processamento tem-se aproximadamente 70MBytes de resultados.

Para executar essa aplicação no OurGrid, necessita-se apenas que seja passado como argumento para o programa, o nome do arquivo que contém os dados a serem processados.

4.4 Portlet para OurGrid

Através do *portlet* gerador, descrito no capítulo anterior, foi criado um *portlet* para a grade OurGrid. A figura 4.1 ilustra o *portlet* que foi gerado.

Esse *portlet* possui algumas informações que devem ser preenchidas para que seja gerado o arquivo *Job Description File*, necessário para executar um *job* na grade. Com o simples preenchimento dos campos, a geração do arquivo passa a ser feita sem que o

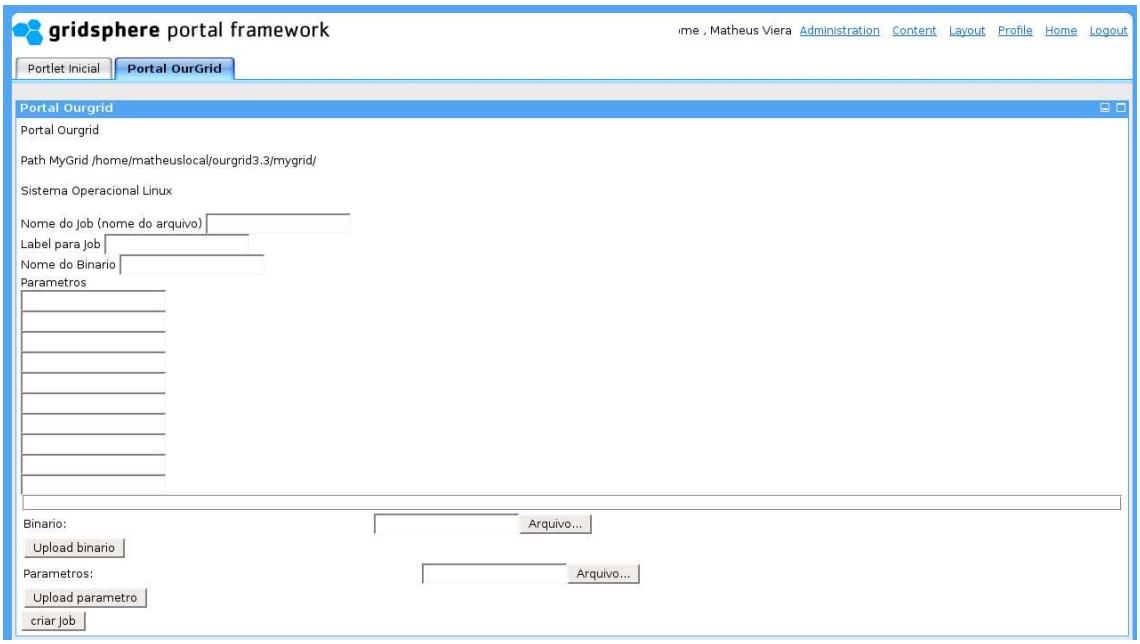

Figura 4.1: *Portlet* OurGrid

usuário necessite saber a estrutura do arquivo. Os *uploads* do arquivo binário da aplicação e de seus parâmetros pode ser feita de forma simples. No caso dos parâmetros, pode-se tanto submeter um *upload* de um arquivo quanto preencher os campos com valores.

A tela seguinte, após a submissão dos dados para o *portlet* e a criação do *job description file*, permite que informações referentes ao estado da grade sejam visualizadas. Além disso, pode-se verificar a saída do programa que executou na grade, após a sua finalização. Essa tela é ilustrada na figura 4.2.

Pode-se observar na figura 4.2 o estado da grade, bem como a saída da execução do programa da meteorologia.

Uma das vantagens da utilização do *portlet* para o *middleware* OurGrid, é que ele possuí uma interface simples de ser utilizada, quando comparado a interface original de execução do Ourgrid. O OurGrid não possuí interface gráfica que permita a sua configuração e execução. Para isso, alguns conhecimentos de comandos são necessários para a execução de tarefas na grade.

Um dos principais benefícios do *portlet* para o OurGrid, é que ele gera o arquivo de descrição do *Job*. Com isso, o usuário não precisa saber nem um tipo de comando de sistema operacional, nem se preocupar em como a grade irá funcionar. Basta que ele forneça as informações referentes ao *job* que deseja executar. Outro benefício, refere-se a visualização de estado da grade e dos resultados obtidos com a execução de uma aplicação

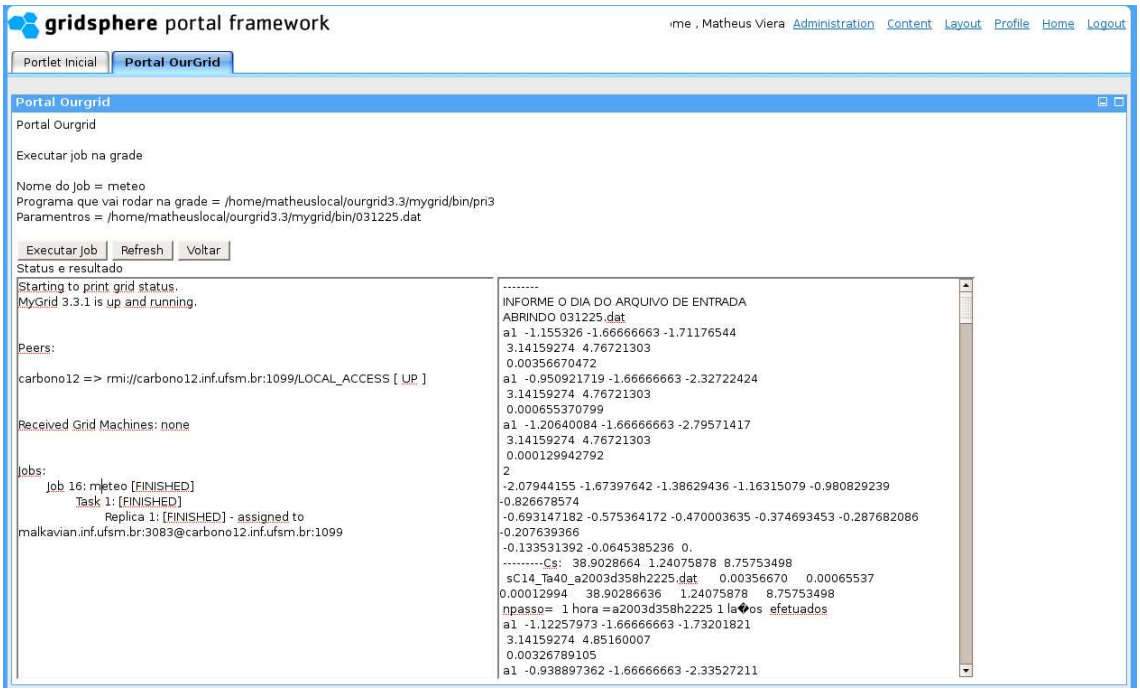

Figura 4.2: Estado da grade e resultado da execução

nela.

Quando comparado com a interface original do OurGrid, o *portlet* não permite a visualização de *jobs* executados anteriormente. Na interface original, basta navegar no diretório do MyGrid e visualizar os arquivos de saída. O *portlet* poderia ser melhorado nesse sentido, apenas com a adição de um mecanismo de *log*, que permitiria que o usuário selecionasse a saída de um *job* executado anteriormente, e pudesse visualiza-lo.

5 CONCLUSÃO

Este trabalho apresentou a construção de um *portlet* gerador de *portlets* para grades computacionais ou execução de aplicações de forma distribuída, através do *framework* GridSphere. Este trabalho pode ser utilizado para facilitar a utilização de ambientes distribuídos através de uma interface de simples utilização.

Primeiramente, foi realizado um estudo do comportamento de *portlets* criados através do *framework* GridSphere, buscando entender o seu funcionamento e os benefícios que poderiam ser alcançados com a sua utilização. Após foi realizado a construção de um *portlet* gerador de novos *porltets*, de forma dinâmica, para as grades OurGrid, XtremWeb e para execução de aplicações em ambientes distribuídos utilizando MPI.

Com os novos *portlets* sendo gerados dinamicamente, fez-se uma avaliação sobre um *portlet* para a grade Ourgrid, executando uma aplicação da área da meteorologia. Pode observar-se que através do *portlet*, as tarefas de lançar um *job* na grade e visualizar o seu estado e resultados finais, foram facilitadas. Com isso usuários com pouco experiência na utilização de grades computacionais já podem executar suas aplicações de forma distribuída através desse *portlet* no portal GridSphere.

O *framework* GridSphere, demonstrou-se bastante útil para a construção dos *portlets* para alcançar os objetivos desse trabalho. Além disso, os *core portlet* presentes na ferramenta auxiliam as tarefas administrativas do portal e por conseqüente da grade, pois somente usuários cadastrados no portal podem utilizar os *portlets* implementados. Porém, um obstáculo encontrado com a construção dos *portlets*, foi a dificuldade de troca dinâmica de informações sem que toda a página fosse atualizada. Esse obstáculo teve de ser contornado com a criação de JavaScript's (FLANAGAN, 2002) que poderiam ser interpretados pelo navegador, sem a necessidade de mandar informações para o *servlet*. Com um suporte a técnica AJAX (JavaScript e XML assíncrono) (GARRETT, 2005), que

permite trafegar-se apenas os dados que realmente foram alterados, sem a necessidade de atualizar a página inteira, não existiria esse problema.

Este trabalho pode ser melhorando com a modificação do *portlet* para geração de novos *portlets* para outros tipos de grades computacionais. Ou até mesmo, a criação de um *portlet* que permita a configuração dessas grades que foram usadas no desenvolvimento do trabalho, permitindo, por exemplo, a inclusão ou exclusão de trabalhadores.

REFERÊNCIAS

ALLEN, G.; GOODALE, T.; RUSSELL, M.; SEIDEL, E.; SHALF, J. Classifying and enabling Grid applications. **Chapter**, [S.l.], v.23, p.601–614, 2003.

ANDERSON, D. P. BOINC: A system for public-resource computing and storage. In: GRID, 2004. **Anais...** IEEE Computer Society, 2004. p.4–10.

CAPPELLO, F.; DJILALI, S.; FEDAK, G.; HERAULT, T.; MAGNIETTE, F.; NÉRI, V.; LODYGENSKY, O. Computing on large-scale distributed systems: xtremweb architecture, programming models, security, tests and convergence with grid. **Future Generation Computer Systems**, [S.l.], v.21, n.3, p.417–437, 2005.

FEDAK, G.; GERMAIN, C.; NIRI, V.; CAPPELLO, F. **XtremWeb: a generic global computing system, ccgrid2001, workshop on global computing on personal devices, may 2001**. [S.l.]: IEEE Press, 2001.

FLANAGAN, D. **JavaScript: the definitive guide**. [S.l.]: Éditions O'Reilly, 2002.

FORSTER, I.; KESSELMAN, C. The Grid: blueprint for a new computing infrastructure. **Morgan Kaufiuiann, San Francisco, CA**, [S.l.], v.211, 1999.

FOSTER, I. What is the Grid? A Three Point Checklist. **Grid Today**, [S.l.], v.1, n.6, p.22–25, 2002.

GARRETT, J. Ajax: a new approach to web applications. **Adaptive Path**, [S.l.], v.18, 2005.

GRIDLAB. **GridLab**: a grid application toolkit and testbed. [S.l.]: GridLab, 2006.

GRIDSPHERE PROJECT. **GridSphere Portlet Reference Guide**. [S.l.]: GridSphere, 2007.

I. Foster; C. Kesselman. Globus: A metacomputing infrastructure toolkit. **International Journal of Supercomputer Applications and High Performance Computing**, [S.l.], v.11, n.2, p.115–128, 1997. DisponÃvel em: ftp://ftp.globus.org/pub/globus/papers/globus.pdf.

KESSELMAN, C.; FOSTER, I. The Grid: blueprint for a new computing infrastructure. **Morgan Kaufmann**, [S.l.], 1999.

NATRAJAN, A.; NGUYEN-TUONG, A.; HUMPHREY, M.; HERRICK, M.; CLARKE, B.; GRIMSHAW, A. The Legion Grid Portal. **Concurrency and Computation: Practice and Experience**, [S.l.], v.14, n.13-15, p.1365–1394, 2002.

NEVES, M. V. **Paralelização de uma aplicação para análise de dados meteorológicos utilizando uma abordagem peer-to-peer**. Trabalho de Graduação - Curso de Ciência da Computação - UFSM.

NOVOTNY, J. et al. The Grid Portal Development Kit. **Concurrency and Computation: Practice and Experience**, [S.l.], v.14, n.13-15, p.1129–1144, 2002.

NOVOTNY, J.; RUSSELL, M.; WEHRENS, O. GridSphere: an advanced portal framework. **Euromicro Conference, 2004. Proceedings. 30th**, [S.l.], p.412–419, 2004.

OurGrid. **OurGrid WebSite**. [S.l.]: OurGrid, 2008.

PROCESS SPECIFICATION. **The Java Community Process, JSR 168**. 2003.

RUSSELL, M.; ALLEN, G.; DAUES, G.; FOSTER, I.; SEIDEL, E.; NOVOTNY, J.; SHALF, J.; LASZEWSKI, G. von. The Astrophysics Simulation Collaboratory: a science portal enabling community software development. **Cluster Computing**, [S.l.], v.5, n.3, p.297–304, 2002.

SNIR, M.; OTTO, S.; HUSS-LEDERMAN, S.; WALKER, D.; DONGARRA, J. MPI: the complete reference (vol. 1). **The MPI Core, Massachusetts Institute of Technology, Cambridge, Mass**, [S.l.], 1998.

SUN MICROSYSTEMS. **Sun Technical Computing Portal**. [S.l.]: Sun, 2002.

THE APACHE SOFTWARE FOUNDATION. **Apache Tomcat**. [S.l.]: Apache, 2007.

XtremWeb. **XtremWeb WebSite**. [S.l.]: XtremWeb, 2008.

YANG, X. Survey of Major Tools and Technologies for Grid-enabled Portal Development. **Proceedings of the UK e-Science All Hands Meeting 2006**, [S.l.], 2006.

ZHANG, C.; KELLEY, I.; ALLEN, G. Grid Portal Solutions: a comparison of gridportlets and ogce. **special issue GCE05 of Concurrency and Computation: Practice and Experience**, [S.l.], 2006.

APÊNDICE A PROCESSO DE INSTALAÇÃO

Este apêndice descreve o processo necessário para a instalação do *framework* GridSphere, apresentando os pré-requisitos para seu funcionamento.

A.1 Pré-requisitos

- Java JDK 1.4.2 ou superior (para versão do GridSphere 3.0 ou superior é necessário Java JDK 1.5 ou superior)
- Jakarta Ant 1.6 ou superior
- Jakarta Tomcat Servlet Container Tomcat 4.1 ou superior (é recomendado Tomcat 5.0.25 ou superior)

A.2 Instalação

O processo de instalação é composto pelas seguintes etapas:

1. É necessário configurar a variável de ambiente JAVA_HOME e adicionar o binário do Java no PATH.
2. Download do GridSphere e seus arquivos fontes no endereço:

   http://www.gridsphere.org/gridsphere/gridsphere/guest/downloadGS/r/

   Descompacte em seu local de preferência e crie a variável de ambiente GRIDSPHERE apontando para essa localização.
3. Download do Ant no endereço:

   http://ant.apache.org/

   Descompacte em seu local de preferência e crie a variável de ambiente ANT_HOME apontando para essa localização.

Adicione ANT_HOME/bin na variável de ambiente PATH.

4. Download da distribuição do Tomcat 4.1 ou superior (caso use o Tomcat 5.5 ou superior é necessário os pacotes: Admin, Deployer e Compat) no endereço:

   http://tomcat.apache.org/

   Descompacte em seu local de preferência e cria a variável de ambiente CATALINA_HOME apontando para essa localização.

   A configuração do servidor Tomcat é feita seguindo os passos desse endereÃ§o:

   http://www.gridsphere.org/gridsphere/docs-GS-2.2.X/UsersGuide/UsersGuide.html#N1007D

   Seguindo esses passos o portal GridSphere estará pronto para ser utilizado.

5. Download do projeto Portal Genérico e de seus arquivos fontes no endereço:

   svn://paple.inf.ufsm.br/matheus-portlet

   Copie o diretório portal para dentro do diretório projects, contido no gridsphere.

   Dentro do diretório projects/portal, basta executar os comandos *ant install* e *ant deploy* para que o *portlet* gerador esteja pronto para ser utilizado.

   Quando o *portlet* estiver sendo usado, após submeter as informações para o servidor, deve-se atualizar a página para que o novo *portlet* apareça no portal do GridSphere.