



UNILASALLE
CENTRO UNIVERSITÁRIO LA SALLE



Curso de Bacharelado em Ciência da Computação

MICHEL DAVID DA COSTA

**ANÁLISE DE *FRAMEWORKS* PARA CONSTRUÇÃO DE PORTAIS DE GRADE E
SUA APLICAÇÃO NO APPMAN**

Canoas, julho de 2009

MICHEL DAVID DA COSTA

**ANÁLISE DE *FRAMEWORKS* PARA CONSTRUÇÃO DE PORTAIS DE GRADE E
SUA APLICAÇÃO NO APPMAN**

Trabalho de conclusão apresentado para a banca examinadora do curso de Ciência da Computação do Centro Universitário La Salle, como exigência parcial para a obtenção do grau de Bacharel em Ciência da Computação, sob orientação da Prof^ª. DSc. Patrícia Kayser Vargas Mangan.

Canoas, julho de 2009

TERMO DE APROVAÇÃO

MICHEL DAVID DA COSTA

ANÁLISE DE *FRAMEWORKS* PARA CONSTRUÇÃO DE PORTAIS DE GRADE E SUA APLICAÇÃO NO APPMAN

Trabalho de conclusão aprovado como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação do Curso de Ciência da Computação do Centro Universitário La Salle - UNILASALLE, pela seguinte banca examinadora:

Prof. MSc. Marcos Ennes Barreto

Unilasalle

Prof. MSc. Mozart Lemos de Siqueira

Unilasalle

Canoas, julho de 2009.

DEDICATÓRIA

Dedico este trabalho de conclusão aos meus pais, que ajudaram a formar as bases sólidas para todo conhecimento que agrego hoje, e que mesmo nos momentos mais difíceis dessa longa jornada, sempre dedicaram o seu amor e incentivo a cooperar com o sucesso da jornada. Aos grandes amigos que compartilharam e discutiram suas idéias comigo durante este período para termos conclusões mais coesas e sólidas. Aos familiares e colegas que colaboraram por muitos dias na pesquisa, modelagem e avaliação deste trabalho. Enfim, a todas as pessoas que contribuíram e ajudaram para a conclusão dessa caminhada, muito obrigado.

AGRADECIMENTOS

Agradeço a professora, coordenadora e orientadora Patrícia Kayser Vargas Mangan por compartilhar de seus conhecimentos e contribuir para o desenvolvimento deste trabalho. A todos os professores que, depois de muitas discussões em aula, sempre prestaram atenção para resolver dúvidas e convergir as idéias levantadas, fortalecendo a base do conhecimento posto em prática durante este trabalho.

Aos professores Abraham Lincoln, Alessandra Dahmer, Gaspare Bruno, Luís Schäffer, Márcia Franco, Marcos Barreto, Mozart Siqueira e Simão Toscani que compartilharam de seu tempo e conhecimento comigo durante estes anos de graduação, meu muito obrigado.

Aos meus grandes amigos, para os quais muitas vezes tive de reapresentar o assunto, porém contribuíram com idéias inestimáveis para a conclusão desta caminhada, Wagner Nobres, Tonismar Bernardo, Fernando Zank, Thiago Oliveira, Thiago Cabelleira, Gian Jaskulski, Jonas Friederich, Diogo El Murr, Wanderson Costa, Rafael Garbin e a todos que ofereceram seu apoio contribuindo de alguma forma a favor desse trabalho, meus agradecimentos.

RESUMO

Há algum tempo que middlewares de grade têm concebido uma forma mais alto nível de controle da grade. Esta solução se chama portal de grade, onde é possível acesso a grade de forma mais intuitiva, sem a necessidade de instalar qualquer pacote na máquina do usuário, apenas com o uso de um navegador de internet. No entanto, com diversas iniciativas, algumas até unificando-se para formar uma solução mais sólida, é difícil determinar qual seria a melhor solução para suprir a necessidade de acesso à grade do protótipo AppMan. Esta monografia apresenta uma comparação entre frameworks para construção de portais de grade e a aplicação de um destes frameworks na construção do portal para o protótipo AppMan. Desta forma, é importante avaliar o impacto que o protótipo terá sobre métricas de tempo de execução total das aplicações na grade.

PALAVRAS-CHAVE: portais, grade, frameworks para portais de grade

ABSTRACT

For some time, the grid middlewares have been designed for a higher level of grid control. This solution is called the grid portal, where you can access the grid in a more intuitive way, without the need to install any package on the user's machine, just using an internet browser. However, with several initiatives, where some of them even unified to form a more solid solution, it is difficult to determine what would be the best solution to meet the needs of accessing the AppMan's grid prototype. This text presents a comparison between frameworks for building grid portals and the application of one of these frameworks in the construction of AppMan's grid portal. Thus, it is important to assess the impact it will have on the prototype, basing on metrics of total execution time of applications in the grid.

KEYWORDS: portals, grid, grid portal frameworks

LISTA DE ABREVIATURAS E SIGLAS

AM	<i>Application Manager</i>
API	<i>Application Programming Interface</i>
BIRN	<i>Biomedical Informatics Research Network</i>
CAS	<i>Central Authentication Service</i>
CSG	<i>Candidate Set Generator</i>
DAG	<i>Directed Acyclic Graph</i>
DoS	<i>Denial of Service</i>
EGA	<i>Enterprise Grid Alliance</i>
EMS	<i>Execution Management System</i>
EPS	<i>Execution Planning Services</i>
ESG	<i>Earth System Grid</i>
EXEHDA	<i>Execution Environment for Highly Distributed Applications</i>
GGF	<i>Global Grid Forum</i>
GRAND	<i>Grid Robust Application Deployment</i>
GRID-ADL	<i>Grid Application Description Language</i>
HTML	<i>HyperText Markup Language</i>
ICENI	<i>Imperial College e-Science Networked Infrastructure</i>
ISAM	<i>Infra-estrutura de Suporte às Aplicações Móveis</i>
JSR	<i>Java Specification Request</i>
LQCD	<i>Lattice Quantum Chromodynamics</i>
MVC	<i>Model View Controller</i>
NAREGI	<i>National Research Grid Initiative</i>
NAT	<i>Network Address Translation</i>
NVO	<i>National Virtual Observatory</i>
OGCE	<i>Open Grid Computing Environment</i>
OGF	<i>Open Grid Forum</i>
OGSA	<i>Open Grid Services Architecture</i>

OGSI	<i>Open Grid Services Infrastructure</i>
OMII	<i>Open Middleware Infrastructure Institute</i>
P2P	<i>Peer-to-peer</i>
PBS	<i>Portable Batch System</i>
PPDG	<i>Particle Physics Data Grid</i>
QoS	<i>Quality of Service</i>
RMS	<i>Resource Management System</i>
SDK	<i>Software Development Kit</i>
SM	<i>Submission Manager</i>
SSH	<i>Secure Shell</i>
SOAP	<i>Simple Object Access Protocol</i>
TI	<i>Tecnologia da Informação</i>
TM	<i>Task Manager</i>
VLAN	<i>Virtual Local Area Network</i>
VO	<i>Virtual Organization</i>
WML	<i>Wireless Markup Language</i>
WSDL	<i>Web Service Description Language</i>
WS-N	<i>Web Services Notification</i>
WS-RF	<i>Web Services Resource Framework</i>
XHTML	<i>eXtensible HyperText Markup Language</i>
XML	<i>eXtensible Markup Language</i>

LISTA DE FIGURAS

Figura 1 Relacionamento entre serviços com base na arquitetura OGSA.....	21
Figura 2 Ordem de execução de uma tarefa	24
Figura 3 Principais componentes do modelo GRAND	27
Figura 4 Encapsulamento dos componentes e dependências no portal	42
Figura 5 Trecho do arquivo context.xml de configuração do Tomcat	42
Figura 6 Configuração do descritor da portlet no uPortal	43
Figura 7 Portlet de manipulação de aplicações	47
Figura 8 Portlet para download de arquivos de uma aplicação	48
Figura 9 Esquema de integração do Portal com o AppMan	49
Figura 10 Tempo (s) de cada execução via script	53
Figura 11 Tempo (s) de cada execução via portal	55
Figura 12 Diferença no tempo (s) de execução em cada submissão	56

LISTA DE TABELAS

Tabela 1 Componentes do AppMan instanciados na grade.....	52
Tabela 2 Síntese dos tempos de execução via script	53
Tabela 3 Síntese dos tempos de execução via portal.....	54
Tabela 4 Diferença no tempo de execução entre a submissão via Scripts e via Portal	55

LISTA DE QUADROS

Quadro 1 Características dos 5 tipos de aplicação para grade	19
Quadro 2 Comparação entre <i>frameworks</i> para desenvolvimento de <i>portlets</i>	39
Quadro 3 Configuração das máquinas da grade	51

SUMÁRIO

1	INTRODUÇÃO	14
1.1	Relevância	15
1.2	Objetivos.....	15
1.3	Estrutura do texto.....	16
2	CONCEITUALIZAÇÃO	17
2.1	Classificação de aplicações para grade.....	18
2.2	Funcionamento da grade.....	19
2.3	Middleware de grade	20
2.3.1	Serviços de infra-estrutura.....	21
2.3.2	Serviços de gerenciamento de execução (EMS).....	22
2.3.3	Serviços de dados	25
2.3.4	Serviços de gerenciamento de recursos	25
2.3.5	Serviços de segurança.....	25
2.3.6	Serviços de autogerenciamento	26
2.3.7	Serviços de informação	26
2.4	AppMan.....	26
2.4.1	Gerenciamento de dados.....	28
2.4.2	Particionamento da aplicação	28
2.4.3	Gerenciamento da aplicação	29
2.5	Estudo sobre adaptação do AppMan para o padrão OGSA	29
2.6	Considerações Finais	30
3	PORTAIS	31
3.1	Arquitetura de portais para grades	32
3.1.1	Interface	32
3.1.2	Serviços oferecidos por portais de grade	33
3.2	Pré-requisitos para a construção de portais para grades	34
3.3	Frameworks para construção de Portlets	36

3.3.1	GridSphere.....	36
3.3.2	OGCE	37
3.4	Considerações finais	38
4	MODELO	41
4.1	Configurações do portal.....	42
4.2	Portlets de integração com o AppMan	44
4.2.1	Controle de submissão de aplicações	44
4.2.2	Manipulação de aplicações.....	45
4.2.3	Download de arquivos da grade	47
4.3	Funcionamento	48
4.4	Considerações finais	50
5	AVALIAÇÃO	51
5.1	Execução via scripts	52
5.2	Execução via portal	54
5.3	Considerações finais	55
6	CONCLUSÃO	57
	REFERÊNCIAS	59
	APÊNDICES	63

1 INTRODUÇÃO

Grades computacionais são foco de estudos pela comunidade científica desde o final da década de 1990. Ao decorrer deste tempo, diversas soluções de grade foram criadas. Com relação a infra-estrutura de software, estas soluções se baseavam em desenvolvimento de código muitas vezes fechado, visando a execução destes apenas quando distribuído pela própria organização que o desenvolveu. Assim, alguns *middlewares* de grade foram sendo propostos como o Legion, o Globus e o Condor-G.

A necessidade de uma unificação na infraestrutura das grades se fez clara e concretizou-se com o início do projeto *Open Grid Services Infrastructure* (OGSI) (TUECKE *et al*, 2003), futuramente aperfeiçoado, dando origem à uma nova arquitetura, chamada *Open Grid Services Architecture* (OGSA) (KISHIMOTO; TREADWELL, 2005). Seguindo os padrões de protocolo expostos na definição da arquitetura OGSA a grade não mais seria vista como um componente único, mas como diversos serviços, com funcionalidades necessárias para o efetivo sucesso da grade. Através desta arquitetura é possível modularizar componentes com o uso de *Web services*. Com interfaces definidas pelo OGSA, um *Web service* pode ser reutilizado, a fim de economizar tempo e trabalho. Estes, agora, podem até mesmo ser substituídos, caso uma versão mais otimizada esteja em vista. Assim, houve uma tendência de que novos softwares, se não rigidamente seguindo o padrão OGSA, pelo menos adotassem o padrão de *Web services*.

Além de um candidato à unificação da arquitetura de grades, do ponto de vista dos usuários de uma grade computacional, é preciso obter fácil acesso à grade de forma a não necessitar conhecimentos do componente de inicialização de aplicações ou mesmo de presença física em um dos nós da grade para fazer o uso desta. A solução para tal problema veio com o desenvolvimento de portais para grades. Em um portal é fornecido ao usuário cliente uma interface gráfica e intuitiva, com recursos de visualização que podem ser melhorados com o tempo, sem a necessidade de definição de novos web services ou modificação nos mesmos.

O protótipo AppMan, baseado no modelo *Grid Robust Application Deployment* (GRAND) (MANGAN, 2006), por não ser um *middleware* de grade mas um gerenciador de

aplicações e por ter sido concebido no início da década de 2000, não segue a arquitetura OGSA definida hoje, bem como não possui um portal para submissão de aplicações. É exigido hoje no AppMan a presença física do usuário em um dos nós da grade, ou o acesso via *Secure Shell* (SSH), para que possa ser inicializada a execução de uma aplicação.

A principal frente de motivação para este trabalho é a necessidade de avaliação de *frameworks* para a construção de portais de grade, com o objetivo de determinar o *framework* que oferece maior compatibilidade na integração de um portal ao AppMan. Portanto, dadas as vantagens de um portal disponível mundialmente via web, esta análise tem como foco a indicação do *framework* mais adequado para a construção de um portal que supra as necessidades imediatas e futuras do protótipo AppMan.

O objetivo final deste trabalho ainda visa a determinação e avaliação do portal, determinando *overhead* mínimo criado pela interface oferecida como forma de portal, e a migração do *middleware* do AppMan para uma estrutura de grade que permita maior integração e controle, como é necessário para um portal.

1.1 Relevância

O presente trabalho encontra-se inserido dentro de um contexto de pesquisa maior. O AppMan é o protótipo de gerenciamento de aplicações em grade que foi construído como uma versão simplificada do modelo GRAND.

Vários esforços de pesquisa vêm sendo realizados nos últimos anos, com finalidade de aprimorar questões de modelo (SANTOS *et al*, 2005; LEMOS, 2007) e de protótipo (BERNARDO, 2008; NOBRES, 2008). Apesar disso, até o momento nenhum trabalho havia focado com maior prioridade na perspectiva dos usuários.

O fato da adoção de um portal para o AppMan trará também visibilidade em frente ao meio acadêmico, visto que o desenvolvimento de portais para grade já é adotado hoje por diversos projetos, confirmando assim as facilidades trazidas com sua implementação.

1.2 Objetivos

O objetivo geral deste trabalho é trazer uma experiência no manuseio de portais para o AppMan, consagrando-o como um serviço estável e de fácil acesso. Além disso, seguem listados alguns objetivos específicos do trabalho:

- Avaliar possibilidade de modularizar os serviços oferecidos em forma de *Web services*, criando componentes que podem ser auto-contidos ou não, dependendo do nível de autonomia;
- avaliar a possibilidade de migrar a arquitetura do AppMan para forma de serviço, facilitando sua integração em outros projetos;
- determinar um *framework* que será adequado ao AppMan, fornecendo os serviços que serão necessários para o funcionamento correto do *middleware*;
- desenvolvimento do portal para o AppMan de forma a facilitar o acesso a este *middleware*, aumentando seu uso e criando a confiança no uso do mesmo;
- avaliar a integração do AppMan ao portal.

1.3 Estrutura do texto

A estrutura desta monografia será dada inicialmente pela apresentação de termos comuns na comunidade de grades e suas definições. Será explicitado o conceito de grade e a classificação das aplicações. Dadas as definições é apresentada a forma de funcionamento de uma grade.

É estudada a arquitetura para construção de serviços de *middleware* OGSA, onde são descritos os serviços necessários para a infraestrutura da grade, ressaltando o serviço de gerenciamento de execução. Também será apresentado o AppMan e seus componentes, buscando a equidade deste aos serviços OGSA e um estudo da adaptação para esta arquitetura.

No capítulo seguinte são apresentados portais de grade em funcionamento, agregando detalhes de implantação e funcionamento dos portais, incluindo seus frameworks de construção.

Após são demonstrados detalhes do modelo, como o mapeamento dos serviços do AppMan para a arquitetura OGSA e a escolha de um framework para a construção de um portal para o AppMan. No caso de possibilidade de enquadramento do AppMan em um portal já existente, isso será feito, concedendo completa funcionalidade ao portal pré-construído através do AppMan.

No capítulo 5 é apresentada a forma de avaliação da solução, incluindo detalhes para o modo de comparação e os resultados obtidos pela avaliação.

Por fim, é apresentada a conclusão desta monografia, onde são mostrados resultados diante do trabalho como um todo, juntamente com trabalhos futuros recomendados.

2 CONCEITUALIZAÇÃO

O termo *grid* foi introduzido inicialmente no meio científico por Foster e Kesselman (1998) com a definição de uma infra-estrutura de hardware e software que provê acesso a capacidades computacionais de alto nível de forma confiável, consistente, pervasiva e econômica. Vários trabalhos vêm sendo desenvolvidos nesta área nos últimos 10 anos, gerando várias outras definições, algumas vezes conflitantes.

Outra iniciativa importante é o trabalho da comunidade Open Grid Forum (OGF) (FORUM, 2009), a qual teve origem na união das comunidades *Global Grid Forum* (GGF) e *Enterprise Grid Alliance* (EGA). Esta comunidade consiste de centenas de pessoas na indústria e pesquisa, representando em torno de 400 organizações em mais de 50 países. Segundo a comunidade OGF (ROEHRIG *et al*, 2002) grades são ambientes persistentes que habilitam aplicações de software a integrar instrumentos, visualizações, computacionais e de informação dos recursos que são gerenciados por várias organizações com localização generalizada.

Com a dificuldade de chegar a um consenso sobre como definir e verificar que um sistema distribuído é ou não uma grade, Foster (2002) propôs três pontos que definem grade como um sistema que:

- coordena recursos que não são sujeitos a um controle centralizado
- usando protocolos e interfaces padronizadas, abertas e de propósito geral
- para distribuir qualidades de serviço não trivial.

Como forma de sistematizar as pesquisas da área, Stockinger (2001) propôs a divisão do campo de pesquisas de grade em dois sub-domínios: *Computational Grid* e *Data Grid*. Enquanto que a *Computational Grid* é uma extensão natural do *cluster* computacional onde grandes aplicações têm de ser computadas em recursos de computação distribuídos, já a *Data Grid* trabalha com gerenciamento, localização e replicação de grandes quantidades de dados.

2.1 Classificação de aplicações para grade

Na literatura (FOSTER; KESSELMAN, 1998) as aplicações são classificadas em 5 tipos distintos (Quadro 1). Estes são:

Supercomputação distribuída (*Distributed supercomputing*): aplicações que usam a grade para agregar recursos computacionais para resolver problemas que não podem ser resolvidos em um único sistema por serem problemas muito grandes, como a exata simulação de processos físicos complexos;

Computação com alta taxa de entrada/saída de dados (*High-Throughput Computing*): usa os recursos da grade para agendar um grande número de tarefas independentes ou pouco dependentes, com o objetivo de aproveitar ciclos de processamento não utilizados;

Computação sob demanda (*On-Demand Computing*): usa as capacidades da grade para adequar requisitos de curto prazo para recursos que não podem estar convenientemente localizados localmente ou com muito custo pela eficácia;

Computação com uso intenso de dados (*Data-Intensive Computing*): o foco destas aplicações está em sintetizar novas informações a partir de dados mantidos em repositórios geograficamente distribuídos, bibliotecas digitais e bases de dados;

Computação colaborativa (*Collaborative Computing*): estas aplicações estão preocupadas primeiramente em habilitar interações entre pessoas. Dessa forma, faz parte da estrutura básica deste tipo de aplicação um espaço virtual compartilhado.

Quadro 1 Características dos 5 tipos de aplicação para grade

Categoria	Exemplos	Características
Supercomputação distribuída	Sistema de Dados e Informações (DIS); Dinâmicas estelares; Química quântica	Problemas muito grandes, que precisam de muito processamento, memória, etc.
Computação com alta taxa de entrada/saída de dados	Desenho de chips; Estudo de parâmetros; Problemas de criptografia	Aproveita recursos ociosos para aumentar a vazão dos dados agregados
Computação sob demanda	Instrumentação médica; Soluções por redes; Detecção de nuvens	Recursos remotos integrados com computação local, normalmente por tempo limitado
Computação com uso intenso de dados	Pesquisa atmosférica; Dados físicos; Assimilação de dados	Síntese de novas informações a partir de muitos ou grandes fontes de dados
Computação colaborativa	Desenho colaborativo; Exploração de dados; Educação	Suporta comunicação ou trabalho colaborativo entre múltiplos participantes

Fonte: Traduzido de Foster e Kesselman, 1998

Será apresentada na próxima seção o funcionamento da grade, com o objetivo de aprofundar os conhecimentos da arquitetura de um *middleware* OGSA e do *middleware* do AppMan.

2.2 Funcionamento da grade

Segundo Foster *et al* (2001), a verdadeira questão por baixo do conceito de grades é o compartilhamento de recursos de forma coordenada e a solução de problemas em organizações virtuais dinâmicas e multi-institucionais, onde o compartilhamento em questão não é uma simples troca de arquivos, mas acesso direto a computadores, software, dados e outros recursos, como é necessário por um grupo de colaboradores solucionadores de problemas e estratégias de encaminhamento de recursos emergentes na indústria, ciência e engenharia. Também de acordo com Foster *et al* (2001), este compartilhamento precisa, obrigatoriamente, de um alto nível de controle, com definições claras dos provedores de recursos e consumidores sobre o que é compartilhado, quem pode compartilhar, e as condições em que isso pode acontecer. Um conjunto de indivíduos e/ou instituições definidas por estas regras de compartilhamento formam o que é chamado de *virtual organization* (VO).

Organizações virtuais podem ter proposta, escopo, tamanho, duração, estrutura, comunidade e sociologia bastante variada. Entretanto, um estudo cuidadoso das necessidades

da tecnologia por baixo destas VOs levaram à identificação de um grande grupo de preocupações e requisitos comuns.

O estabelecimento e gerenciamento de compartilhamento de recursos entre VOs explorados dinamicamente exige uma arquitetura, chamada *Grid architecture* (arquitetura de grade), demonstrada aqui para identificar componentes fundamentais do sistema, especificando o propósito e funcionalidade destes, além de como estes interagem uns com os outros.

A arquitetura de grade (FOSTER *et al*, 2001) é uma forma de tornar possível a interoperabilidade entre VOs. Se tratando de interoperabilidade em redes de computadores, a solução seria a adoção de protocolos comuns. Entretanto, a arquitetura de grade é primeiramente uma arquitetura de protocolos, usando estes para definir o mecanismo básico em que um usuário de VO e recursos negociam, estabelecem, gerenciam e exploram relacionamentos de compartilhamento. Uma arquitetura aberta e baseada em padrões facilita a extensibilidade, interoperabilidade, portabilidade e compartilhamento de código. Também é possível a construção de *Application Programming Interfaces* (APIs) e *Software Development Kits* (SDKs) para prover a abstração necessária para criar uma grade usável. Em conjunto, esta tecnologia e arquitetura constitui o que é normalmente chamado de *middleware*.

2.3 Middlewares de grade

Um *middleware* contém todos os serviços necessários para o suporte de um conjunto comum de aplicações em uma rede distribuída (AIKEN *et al*, 2000). Neste podem ser considerados os protocolos usados por cada serviço, como padronização do meio de comunicação entre serviços. Serviços são um ponto chave da arquitetura da grade, pois um serviço é definido por completo pelo protocolo que este usa para se comunicar e o comportamento implementado.

A arquitetura baseada em serviços adotada em *middlewares* de grade hoje é o resultado do desenvolvimento do conceito apresentado no artigo de Foster *et al* (2002) pela comunidade Global Grid Forum. Esta arquitetura, chamada de *Open Grid Services Architecture* (OGSA), é adotada como padrão de interoperabilidade por diversos projetos (KISHIMOTO; TREADWELL, 2005) como Grid Computing Project (BG), NextGrid, National Research Grid Initiative (NAREGI), Open Middleware Infrastructure Institute (OMII), UniGrids e Gridbus.

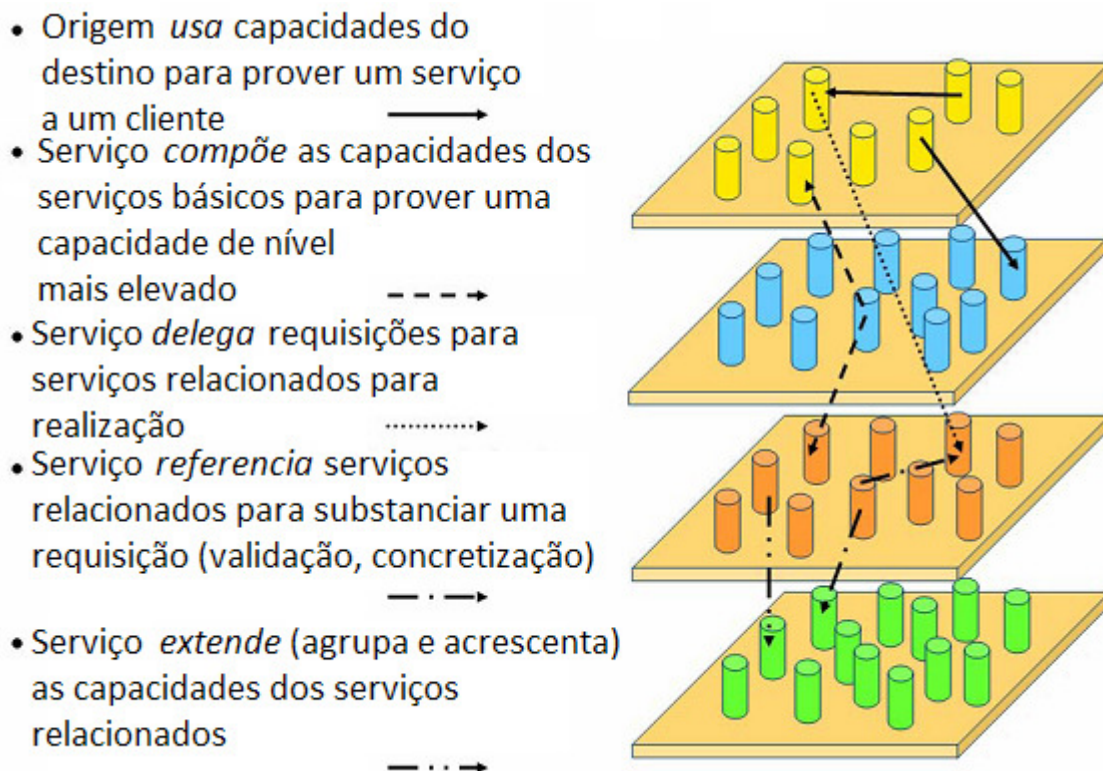


Figura 1 Relacionamento entre serviços com base na arquitetura OGSA.

Fonte: Traduzido de Foster *et al* (2006)

De acordo com o Grid Forum (FOSTER *et al*, 2006), os serviços básicos necessários para fornecer as capacidades de grade são referidos com *infrastructure services* (serviços de infraestrutura) ou da *Grid fabric*. Também é assumido que a especificação do *Web Services Resource Framework* (WS-RF) será parte da versão inicial do *Grid fabric*. A Figura 1 mostra como os serviços podem se relacionar de acordo com o OGSA, considerando que serviços que oferecem maiores capacidades devem usar outros serviços como apoio, os quais devem fornecer serviços básicos para a execução do serviço.

As próximas seções apresentam alguns aspectos principais destes serviços, conforme apresentado em Foster *et al* (2006).

2.3.1 Serviços de infra-estrutura

De acordo com a OGSA (FOSTER *et al*, 2006), serviços são providos como forma de *Web Services*, adotando assim o uso do *Web Service Description Language* (WSDL) como linguagem de descrição para permitir a interface entre os serviços. Também é escolhido o

XML como linguagem padrão de descrição e representação e SOAP como o formato de troca de mensagens primário.

É usado na OGSA o protocolo padrão *WS-Security*, que permite aos serviços o transporte de *tokens* de forma segura para fins de autenticação, autorização e proteção da mensagem. Em alguns casos, onde é necessária proteção da mensagem de ponta a ponta, pode ser usada a criptografia de XML e assinatura digital junto com ou no lugar de segurança a nível de transporte, como TLS ou IPsec.

Finalmente, são assumidas as especificações do *WS-Notification* (WS-N) para capacitar notificações e técnicas para eventos definidas na especificação. Este tipo de mecanismo de notificação permite assinaturas e notificação de mudanças de componentes de estado.

2.3.2 Serviços de gerenciamento de execução (EMS)

Os serviços de gerenciamento de execução se preocupam com problemas relacionados com o a instanciação e gerenciamento de unidades de atividade, a fim de completá-las. Qualquer tipo de aplicação, seja baseada no OGSA ou não, é considerada uma unidade de atividade.

Em aplicações OGSA será cuidado o tipo de serviço necessário, se este já existe e pode ser usado, no caso de não existir ainda, onde este pode ser alocado, como deve ser configurado, como serão fornecidos os recursos necessários a este serviço, que tipo de acordos este serviço pode fazer e que tipo de acordos ele precisa.

No caso de uma aplicação legada, existe uma classe de aplicações que recebe informações de entrada (normalmente em arquivos) e parâmetros e gera saídas. Este tipo de aplicações é executada no que se chama “*embarrassingly-parallel parameter space studies*” (estudos de escopo de parâmetro trivialmente paralelizáveis). Em aplicações legadas também existem questões a serem tratadas como onde a aplicação será executada, a forma que os dados e executáveis irão para o *stage* (local da ação), o que ocorre se a execução falhar, se será reiniciada e como.

Assim que a execução estiver inicializada, também é trabalho do EMS gerenciá-la e monitorar esta execução até o seu término, bem como tomar as decisões cabíveis de acordo com o estado do término da execução.

Para a questão do EMS suas funcionalidades foram quebradas em múltiplos problemas, solucionáveis por componentes substituíveis. Algumas implementações de grade não

precisam usar todos os serviços ou podem encapsular alguns dentro de outros e não disponibilizá-los diretamente. Os sub-serviços determinados foram os seguintes:

- *Resources* (recursos): modelam o processamento, armazenamento, executáveis, gerenciamento de recursos e provisionamento;
- *Job management* (gerência de tarefa) e
- *Resource selection services* (serviços de seleção de recursos): de forma coletiva, decide onde executar uma atividade.

2.3.2.1 Recursos

Recursos são definidos em um *container* de serviços, que contém entidades em execução, indiferente se são tarefas ou *Web services*. *Containers* possuem as propriedades dos recursos, descrevendo informações estáticas, como o tipo de executáveis aceitos (versão do Sistema Operacional (SO), bibliotecas instaladas, políticas, e ambiente seguro), tanto quanto informações dinâmicas, como a quantidade de carga e informações de *quality of service* (QoS).

2.3.2.2 Gerência de tarefa

Uma tarefa, neste contexto, incorpora e estende a noção de uma tarefa convencional. A tarefa encapsula tudo que é conhecido em uma unidade de atividade. A tarefa é a melhor unidade gerenciável. Ele representa a forma de manusear uma unidade de atividade: não é o mesmo que a aplicação que está executando, ou a visão da execução de uma unidade de tarefa. A tarefa mantém o *log* do estado da execução, comprometimento e acordos com recursos, requisitos da tarefa, metadados do usuário (credenciais), e quantas vezes a tarefa foi iniciada. Estes são armazenados sob a forma de um *job document* (documento de tarefa). Este documento pode ser exposto como um recurso de propriedade da tarefa.

2.3.2.3 Serviços de seleção de recursos

Existem três tipos de serviços que são úteis à seleção e agendamento de recursos:

- *Execution Planning Services* (EPS) (serviço de planejamento de execução)
- *Candidate Set Generator* (CSG) (gerador de conjuntos candidatos)
- *Reservation Services* (serviços de reserva)

O serviço de planejamento escala as execuções, onde uma escala é um relacionamento entre serviço e recurso, com possibilidade de restrições de tempo. Uma escala pode ter execuções alternativas, que, basicamente, informam “se esta parte da escala falhar, tente esta outra no lugar”. O EPS usa serviços de informação juntamente com o CSG. Por exemplo, primeiro é buscado um conjunto de recursos candidatos, após, são buscados dados atuais dos recursos envolvidos a partir de um serviço de informação, e por último é montada a escala.

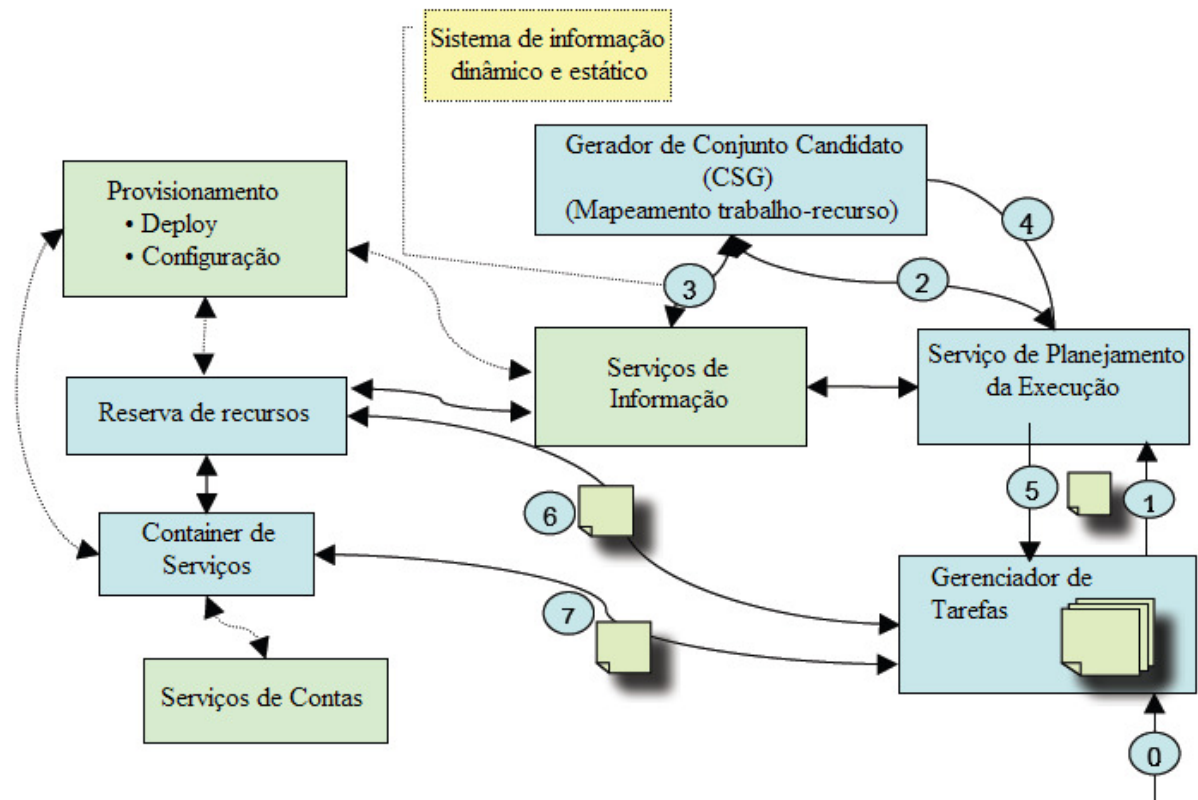


Figura 2 Ordem de execução de uma tarefa

Fonte: Traduzido de Foster *et al* (2006)

Na geração de conjuntos candidatos a idéia é simples: determinar um conjunto de recursos em que uma unidade de atividade pode ser executada. Neste caso estão sendo buscados recursos que têm possibilidade de execução, e não onde exatamente será a execução. Este serviço atende pré-requisitos de aplicação (espaço em disco necessário) e questões de segurança e confiança (certificação de código e de recurso).

Serviços de reserva gerenciam a reserva de recursos, interagem com serviços de conta (caso haja custo pela reserva) e revogam reservas. A reserva de um recurso tem como representação um documento de acordo que está assinado.

A Figura 2 ilustra a ordem da interação entre os componentes explicados, decorrente da submissão de uma tarefa ao gerenciador de tarefas, através do uso dos serviços de informação, planejamento e geração de conjuntos candidatos.

2.3.3 Serviços de dados

Estes serviços são responsáveis pela gerência, acesso e atualização de recursos de dados, bem como a transferência de dados entre recursos. Também é possível gerenciar os metadados que descrevem os dados, em particular, a proveniência dos dados. Estes serviços podem receber uma especificação para determinar como trabalhar com certo esquema de dados, indicando em que recurso buscar determinado dado.

Não é cuidado pelo serviço de dados os tipos de dados ou a semântica dos mesmos, pois este trabalha com dados genéricos.

2.3.4 Serviços de gerenciamento de recursos

No gerenciamento de recursos são usadas algumas formas para gerenciar os recursos em uma grade. Em OGSA existem três tipos de gerenciamento que envolvem recursos:

- Gerenciamento de recursos físicos e lógicos: *restart* de servidor, ou configuração de VLAN em um *switch* de rede;
- Gerenciamento de recursos OSGA da grade, expostos através de interfaces de serviço: reservas de serviço, submissão de tarefas e monitoramento;
- Gerenciamento da infra-estrutura OSGA da grade, exposta através de interfaces de gerenciamento: monitoramento de um serviço de registro.

2.3.5 Serviços de segurança

Serviços de segurança OGSA facilitam o cumprimento de políticas relacionadas à segurança de uma organização virtual. O propósito do cumprimento destas políticas é para permitir o cumprimento de objetivos de alto nível de negócio. Por exemplo, pode ser forçada a integridade e confidencialidade de mensagens, autenticação de entidades com interação, um mínimo de robustez na autenticação, log e auditoria seguros, separação de responsabilidades, detecção de invasores, verificação de política de autorização, mínimo de privilégios,

mecanismos obrigatórios de controle de acesso, nível de confiabilidade do ambiente, isolamento da aplicação, anulação de ataques DoS, redundância, e treinamento.

2.3.6 Serviços de autogerenciamento

O autogerenciamento foi uma forma concebida para diminuir os custos de possuir uma infra-estrutura operacional de TI. Em um ambiente autogerenciável componentes do sistema, como componentes de hardware (computadores, redes, dispositivos de armazenamento) ou software (sistemas operacionais e aplicações comerciais), são automaticamente configurados, mantidos e otimizados. Este tipo de serviço permite às organizações o funcionamento com menos recursos humanos e menos custos. Em um sistema autogerenciado um novo recurso é simplesmente implantado, e depois ocorre a otimização.

Embora seja esperado que os atributos necessários para o autogerenciamento estejam presentes nos serviços, nem todos os serviços em um sistema autogerenciável demonstram todos ou sequer um subconjunto destas propriedades.

2.3.7 Serviços de informação

Na arquitetura OGSA, a habilidade de acessar e manipular informações sobre aplicações, recursos e serviços no ambiente de grade de forma eficiente é uma importante capacidade. A informação tratada aqui se refere a dados dinâmicos ou eventos usados para monitoramento de estado, como dados relativamente estáticos usados para descoberta, e qualquer dado que é registrado.

Clientes do serviço de informação incluem, embora não seja limitado, a serviços de gerenciamento de execução, serviços de conta, serviços de determinação de problema, serviços de reserva de recursos, serviços de carga de recursos, e monitoramento de aplicação.

2.4 AppMan

O AppMan é um protótipo baseado no modelo GRAND, definido por Mangan (2006) com foco na submissão de aplicações com um grande número de tarefas, onde é abordado também o controle da execução. Este modelo define o gerenciamento de dados, o particionamento da aplicação e um modelo de gerenciamento para a aplicação, que juntos

favorecem o sucesso do modelo ao permitir a distribuição da carga no processo de submissão e controle.

Alguns componentes foram definidos (Figura 3) para o cumprimento do modelo, são eles o Gerenciador de Aplicação (*Application Manager* ou *AM*), que serve como ponto de entrada para a submissão de aplicações, Gerenciador de Submissão (*Submission Manager* ou *SM*), responsável pelo encaminhamento das tarefas, e por último, mas não menos importante, o Gerenciadores de Tarefas (*Task Manager* ou *TM*) que é encarregado de executar as tarefas nos nós da grade através de um Sistema de Gerenciamento de Recursos (*Resource Management System* ou *RMS*).

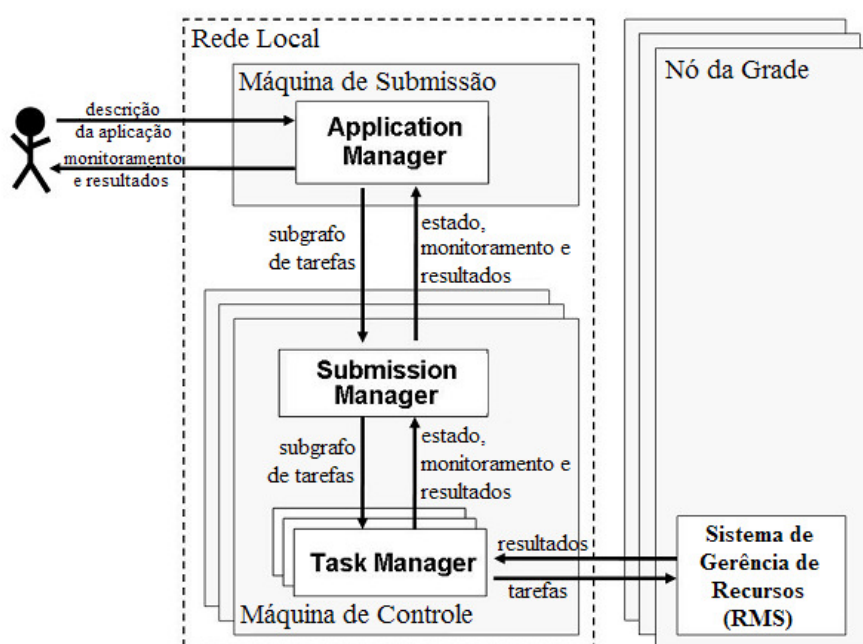


Figura 3 Principais componentes do modelo GRAND

Fonte: Traduzido de Vargas (2006)

O projeto GRAND recentemente teve otimizações de código e modelagem do protótipo AppMan através do trabalho de Nobres (2008), agregando conceitos de orientação a objetos e padrões de projeto. O trabalho buscava avaliar se tais conceitos poderiam ser aplicados mesmo em sistemas onde o desempenho é crucial para o sucesso na execução. Outro trabalho que acrescentou funcionalidades é o apresentado em Bernardo (2008), que possibilitou o uso de diferentes sistemas de gerenciamento de recursos.

2.4.1 Gerenciamento de dados

O gerenciamento de dados é realizado pelo AM, que controla o envio de dados e tarefas bem como o resultado das tarefas executadas. Este gerenciamento é feito sob três visões:

- *Stage in*: transferência dos dados para o local onde estes dados serão necessários;
- Controle do envio de dados: contempla problemas relacionados ao envio de grandes volumes de dados;
- Escalonamento baseado na localidade: evita transferências desnecessárias de dados transientes e conseqüente degradação de desempenho.

A submissão da aplicação pela máquina de submissão (*submit machine*) dispara ou utiliza gerenciadores previamente inicializados pelo sistema, que têm como função fazer o escalonamento das tarefas (de acordo com os requisitos do usuário). As máquinas pesquisadas são priorizadas de acordo com a localidade dos dados.

Uma otimização no gerenciamento de dados de grades, envolvendo transferência de dados em um modelo *P2P* (*peer-to-peer*) foi desenvolvida por Reis (2009), comparando o desempenho deste tipo de transferência com o serviço do GridFTP. No estudado realizado por Reis, a transferência de dados na grade é beneficiada pelo uso do serviço GridFTP invés da transferência de dados distribuída proporcionada pelo modelo *P2P*.

Uma vez que os dados chegam ao SM, este se encarrega da distribuição dos mesmos. Desta forma é liberada a carga da máquina de submissão, já que normalmente é a máquina de trabalho do usuário.

2.4.2 Particionamento da aplicação

Uma aplicação paralela, no contexto do projeto GRAND, é formada por tarefas que podem possuir dependências entre si, porém sem troca de mensagens. As dependências entre tarefas são determinadas pelos dados de entrada e saída (normalmente arquivos de dados), explicitados no arquivo descritor da aplicação. Este arquivo descritor utiliza a linguagem *Grid Application Description Language* (GRID-ADL) (MANGAN, 2006) como linguagem adotada e necessária para descrever a aplicação a ser submetida.

Nesta linguagem é possível descrever a aplicação como um conjunto de tarefas em um formato similar a uma linguagem de script como por exemplo AWK. A partir a indicação das tarefas, o AppMan infere a aplicação como um grafo, onde cada nó representa uma tarefa e

cada aresta a sua precedência. O grafo será, obrigatoriamente, um grafo acíclico dirigido (DAG ou *Directed Acyclic Graph*) para permitir a sua execução como utilizado em outros sistemas como o Condor (THAIN *et al*, 2003). Através do particionamento deste grafo, o sistema divide automaticamente o mesmo em sub-grafos, permitindo que estes sub-grafos sejam submetidos e executados de forma distribuída.

2.4.3 Gerenciamento da aplicação

O disparo e controle da aplicação é realizado através de uma hierarquia de gerenciadores, onde o AM é encarregado de receber um arquivo de entrada (descrição da aplicação), particionar suas tarefas em sub-grafos enviando-os a diferentes SMs, e, por último, mostrar de forma amigável informações sobre o estado da execução da aplicação.

Os SMs por sua vez têm a finalidade de determinar onde alocar recursos para a execução das tarefas com base em informações dinâmicas sobre os recursos, acompanhar o andamento da aplicação, recuperar falhas, manter um registro persistente (*log*), criar e monitorar *TMs*, e avaliar a continuidade da execução com base na ocorrência de falhas.

Um estudo feito por Lemos (2006) desenvolveu um modelo de monitoramento de recursos de forma hierárquica que, embora não esteja integrado está adaptado para o AppMan, usando um modelo hierárquico de transmissão de informações de monitoramento semelhante à arquitetura hierárquica de submissão do AppMan.

Task Managers por sua vez, são responsáveis por se comunicar com o escalonador de um determinado domínio para garantir a execução das tarefas, a ordem de execução de acordo com as dependências de dados, controlar a transferência de dados garantindo os dados necessários de entrada e o recebimento dos de saída.

2.5 Estudo sobre adaptação do AppMan para o padrão OGSA

Segundo a pesquisa realizada sobre os serviços OGSA, o AppMan é um candidato ao serviço de gerenciamento de execução. Hoje o protótipo executa em um ambiente de execução direcionado à computação pervasiva (EXEHDA), e se baseia em recursos deste middleware. Os principais serviços do EXEHDA utilizados são os serviços de informação, de segurança, de gerenciamento de recursos e de infraestrutura os quais tem relação direta com a arquitetura OGSA. Também é usado pelo protótipo serviços do sistema operacional para prover capacidades caracterizadas nos serviços de dados da arquitetura OGSA.

O trabalho de migração do AppMan para a arquitetura OGSA deve, além de converter a submissão de aplicações para o padrão de serviço de gerenciamento de execução, criar serviços de suporte como o serviço de dados, gerenciamento de recursos, segurança e informação. Estes serviços devem ser condizentes com o padrão OGSA, para manter sua compatibilidade com outras grades que se comunicam através da mesma arquitetura.

Considerando que o protótipo do AppMan está sendo utilizado em outros trabalhos e a migração para o formato OGSA implicaria grandes modificações que poderiam comprometer a integração dos esforços, optou-se por manter a execução do protótipo AppMan através de scripts, mantendo apenas um plano de migração para o padrão OGSA no Apêndice K. Tal decisão direcionou a escolha do framework de portal a ser utilizado conforme será discutido a seguir.

2.6 Considerações Finais

Neste capítulo foi apresentada a arquitetura de *middleware* de grade OGSA, adotada atualmente por projetos de grande influência entre a comunidade científica, demonstrada pela sua adoção em projetos livres, como Globus (2009), Business Grid Computing Project (SAVVA *et al*, 2004), NextGRID (2009), OMII-UK (2009), UniGrids (ANTIPOLIS, 2005), GridBus (GRIDS, 2009).

Também foi estudada a forma como o modelo GRAND trata da submissão de aplicações e com a análise das funcionalidades de serviços de gerenciamento de execução (EMS), foi possível determinar os pontos a serem trabalhados no caso de uma migração para a arquitetura OGSA. Considerando o foco deste trabalho ser a construção de um portal, esta análise fica registrada como pontos a serem tratados em caso de migração para o padrão.

No capítulo 3 serão estudados *containers* de portais, suas capacidades e *portlets* pré-definidas para facilitar na construção de um portal para o AppMan. Contemplando as *portlets* pré-definidas, será necessário o uso de *portlets* para interação com o AppMan e, conseqüentemente, frameworks para a construção destas, os quais também serão analisados buscando vantagens e desvantagens de acordo com o uso. A opção deste trabalho em continuar utilizando scripts para execução obriga a construção de portlets específicas, uma vez que os frameworks para portais mais atuais tem portlets para acesso ao OGSA.

3 PORTAIS

A comunidade acadêmica tradicionalmente constrói seus sistemas a fim de realizar suas pesquisas. Com o aumento das complexidades de pesquisa e, principalmente, a necessidade de utilizar infra-estruturas mais complexas como as grades computacionais, os sistemas utilizados vêm amadurecendo com o crescente número de comunidades científicas que confiam em códigos de terceiros (comunidades *opensource* e aplicações comerciais) invés de desenvolver seus próprios códigos. Estes softwares, entretanto, são muitas vezes mais complexos, dificultando a adoção por um pesquisador não iniciado. Diminuir a fronteira inicial necessária para permitir a iniciação de um usuário em ambientes de grade é um desafio técnico e social, ainda mais com falta de mão-de-obra especializada na área.

Interfaces de portais baseadas em navegadores (*browsers*) de web provêm acesso a uma grande variedade de recursos, serviços, aplicações e ferramentas para organizações privadas, públicas e comerciais (THOMAS *et al*, 2005b). Portais computacionais científicos são úteis para aplicações em larga escala de projetos científicos. Além disso, portais com suporte a grades podem distribuir soluções complexas de grade a usuários sempre que estes tiverem acesso a um navegador da web com uma conexão à internet, sem a necessidade de baixar ou instalar software especializado ou se preocupar com configurações de rede, firewall, e políticas de porta. Conseqüentemente, estes portais têm se provado como mecanismos efetivos para a exposição de recursos computacionais e sistemas distribuídos para comunidades de usuários em geral, sem forçá-los a lidar com as complexidades dos sistemas adjacentes (THOMAS *et al*, 2005b).

Atualmente existem portais utilizados na comunidade científica, direcionados à diferentes áreas do conhecimento, como por exemplo química (ANTIPOLIS, 2005; GRIDCHEM, 2009; LQCD, 2009), astronomia (NVO, 2009), física (CACTUS, 2009; PPDG, 2009; SCIDAC, 2009), biologia (BIRN, 2009), nanotecnologia (NANOHUB, 2009), geofísica (GEONGRID, 2009; QUAKEsim2, 2009) e clima e tempo (ESG, 2009).

Neste capítulo será apresentado o atual estado da arte em portais de grades e uma comparação entre frameworks para criação de *portlets* para portais que usam containers compatíveis com a especificação JSR-168. A conceitualização de portlets e containers

compatíveis com a especificação JSR-168 é realizada na seção 3.1.2, onde são descritos os serviços oferecidos por um portal. No caso dos frameworks estudados, é importante ressaltar que o *framework* GridSphere é parte integrante do portal GridSphere, que usa o container GridSphere, enquanto o framework OGCE é disponibilizado em conjunto com o portal uPortal, que usa o container Apache Pluto (APACHE, 2009a).

3.1 Arquitetura de portais para grades

Portais de grade são facilmente confundíveis com páginas de projetos e outras aplicações disponíveis na web. No entanto, existem características, como a comunicação com um middleware de suporte juntamente com a interface visual do usuário, que deixam clara a diferença entre outros portais web.

Para esta interconexão, é possível interagir com a arquitetura de *middlewares* OGSA, o que já é feito em alguns portais como ICENI (MCGOUGH *et al*, 2005), Textile Grid Portal (BUSUOLI, 2009), ePhysics (BEESON *et al*, 2005). Também é possível interagir com a grade através de *middlewares* baseados na execução de scripts, como é o caso de versões iniciais do Globus e atualmente do AppMan, executando *scripts* através de um usuário local, o qual é usado também para a execução do portal.

Um pré-requisito importante na execução de um portal é o seu *container*. Um portal é composto por *portlets*, que podem ou não interagir entre si. Para gerenciar tais *portlets* são usados *containers* de portlets, dentre eles o GridSphere e Apache Pluto (usado pelo portal uPortal), que serão estudados e comparados na seção 3.3.

3.1.1 Interface

Sendo o portal a porta de entrada para diversos serviços, não exclusivos à grade, seu processo de autenticação deve permitir uma credencial única válida para todos os serviços, além de permitir ao usuário a customização da visualização dos serviços de interesse ao mesmo. Ao se autenticar no portal da grade, deve ser buscado para o usuário o atual contexto do mesmo, como suas anotações, pastas de navegação, referências e logs de notificação. Também deve ser apresentado os grupos de ferramentas usados pelo usuário para efetuar atividades remotas, organizados muitas vezes em abas, da forma como o mesmo organizou previamente.

Portais de grade possuem requisitos próximos aos de portais orientados a consumidores (p. ex. Google, Yahoo, CNN). Serviços normalmente possuem suporte a um contexto, interfaces baseadas em navegadores, diferente de clientes desktop, as conexões passam pelo servidor do portal, superando problemas de firewall e/ou NAT. Também é um requisito ter páginas ativas disponíveis para usuários anônimos, autenticados ou usuários com permissão especial (p. ex. administrador). Em portais científicos também há o suporte das aplicações de grade no ambiente do usuário (THOMAS *et al*, 2005b). De maneira específica, estes portais devem gerenciar credenciais, inicializar aplicações, gerenciar arquivos e esconder complexidades da grade, como a forma de submissão da aplicação, seja por script ou web service.

3.1.2 Serviços oferecidos por portais de grade

Segundo Thomas *et al* (2005b), portais de grade normalmente oferecem serviços básicos e geralmente necessários, como:

- Segurança: o usuário pode se autenticar a partir de um navegador de internet usando credenciais como usuário e senha e gerenciar seus dados cadastrais ou de sessão;
- gerência de dados: provê acesso a arquivos e metadados de arquivos locais e remotos, oferecendo transferência de arquivos para a grade;
- submissão de tarefas: habilita a submissão de tarefas à grade, bem como o monitoramento das mesmas;
- serviços de informação: acesso a diretórios e ferramentas de status;
- interface com a aplicação: possibilita a abstração dos detalhes da grade através de interfaces úteis à aplicação;
- colaboração: portais servem como porta de entrada para que organizações virtuais compartilhem recursos;
- fluxo de tarefas (workflow): apresenta tarefas ao usuário e assume a responsabilidade controlar a seqüência de execução caso existam dependências;
- visualização: provê ferramentas que permitem ao usuário o acesso aos dados, renderizando e visualizando recursos e pode prover um nível de visualização de dados ou pode ser usado para inicializar ferramentas mais avançadas.

Os serviços de portais de grades podem ser construídos usando frameworks de arquitetura orientada a serviço (SOA), que estão sendo usados para construir alguns sistemas de grade em escala global. Como esta arquitetura é baseada em mensagens, como SOAP, invés de interfaces de aplicação, estes serviços podem ser integrados em uma variedade de frameworks e, assumindo que a semântica das mensagens não mude, esta arquitetura provê um ambiente de programação bastante flexível. A arquitetura de grades vem sendo redefinida a fim de tomar vantagem desta capacidade, tornando-se uma coleção de clientes para serviços remotos da grade.

Estes clientes para serviços da grade são apenas adaptados para a visualização em um portal web. No portal são apresentadas *portlets* (JCP, 2003) que têm papel fundamental de apresentar informação e receber dados para novas requisições. Os usuários podem optar por integrar as *portlets* mais interessantes aos seus trabalhos de forma customizada através do portal. Dentre as *portlets* mais úteis para os usuários (THOMAS *et al*, 2005a) estão as que permitem acesso a uma base de dados, chamam um web service, e conectam a um serviço da grade. Pela visão administrativa é fácil incluir uma nova *portlet* e, além disso, diferentes grupos de desenvolvedores podem contribuir com *portlets* que podem se conectar com o portal. O uso de *portlets* é suportado e reutilizado em diversos projetos como o OGCE (2009b), Jetspeed (2009), uPortal (JA-SIG, 2006), GridSphere (WEHRENS; BECK-RATZKA, 2006), dentre projetos de grandes organizações como IBM, SUN e Oracle.

Através do uso de um modelo para componentes bem definido é possível abrir um portfólio de componentes distribuíveis para acesso à grade e possibilitar ao usuário a customização e organização do seu ambiente.

3.2 Pré-requisitos para a construção de portais para grades

Um portal é uma aplicação web que normalmente provê personalização, autenticação única, agregação de conteúdo a partir de fontes diferentes e serve a camada de apresentação de sistemas de informação (JCP, 2003). Existem portais que não se baseiam na personalização da interface e são usados de forma fixa para comunicação com a grade, como o portal G-Monitor (GRIDS, 2009) do projeto GridBus (2009), no entanto este tipo de portal não possui formas de integração de conteúdo, o que o exclui do estudo em questão.

A personalização do portal pode ser realizada em diversos níveis, como pelo administrador de conteúdo de uma determinada parte do portal, pelo administrador ou pelo

usuário, buscando melhor organização do conteúdo determinado por ele como útil. O portal pode também relacionar o usuário de login com credenciais da grade ou limitar o acesso deste às *portlets* através de grupos. Agregação é o ato de integrar conteúdo de deferentes fontes em uma única página web. Um portal pode ter como característica a habilidade de prover conteúdo customizado aos seus usuários de forma sofisticada. As páginas do portal podem ter um conjunto diferente de *portlets* definindo o conteúdo para diferentes usuários.

Uma *portlet* é um componente web, gerenciado por um *container* de *portlets*, que processa requisições e gera conteúdo dinâmico. *Portlets* são usadas por portais como componentes plugáveis da interface de usuário que provêem uma camada de apresentação aos sistemas de informação, que no caso, se tratam dos serviços da grade.

O conteúdo gerado pela *portlet*, chamado fragmento, é um pedaço de uma linguagem de marcação (HTML, XHTML, WML) que tem conformidade com certas regras e pode ser agregada com outros fragmentos para formar um documento completo. O conteúdo de uma *portlet* é normalmente agregado com o conteúdo de outras *portlets* para formar a página do portal. O ciclo de vida de uma *portlet* é gerenciado pelo *container* de *portlets*.

Clientes web interagem com portlets através de um fluxo de requisições e respostas, como em uma arquitetura cliente/servidor. Normalmente, usuários interagem com o conteúdo produzido por *portlets*, por exemplo, seguindo *hyperlinks* ou submetendo formulários, resultando em ações de portlets sendo recebidas pelo portal, que são encaminhadas pelo mesmo para as *portlets* responsáveis pela interação dos usuários.

Para fazer o uso de portlets é necessário ter um *container* apropriado para a *portlet*. Existem dois padrões para a implementação de container de portlets utilizando tecnologia Java atualmente, o JSR-168, que considera a especificação 1.0 de portlets, ou o JSR-286, considerando a especificação 2.0 de portlets (JCP, 2008).

Neste trabalho, os dois containers de *portlets* analisados (GridSphere e Apache Pluto), se baseiam na JSR-168, que se apresenta mais estável no momento. Um container de *portlets* serve para conter as *portlets* e gerenciar seus ciclos de vida. Também é fornecido por este um armazenamento persistente para as preferências da *portlet*. O container de *portlets* recebe requisições do portal para executar requisições nos *portlets* servidos pelo mesmo.

O container de *portlets* não é responsável por agregar o conteúdo produzido pelas *portlets*. É uma responsabilidade do portal a agregação das *portlets*. Um portal e o container de *portlet* podem ser construídos juntos, como um único componente, como é o caso do GridSphere (WEHRENS; BECK-RATZKA, 2006), ou como dois componentes separados,

que é o caso do portal uPortal (JA-SIG, 2006), que usa Apache Pluto como container de *portlets*.

3.3 Frameworks para construção de Portlets

Como observado, *portlets* compreendem aplicações web que podem ser agregadas em um portal, que está rodando sobre um container de *portlets*. Desta forma, dois frameworks de auxílio no desenvolvimento de *portlets* para aplicações de grade serão apresentados a seguir. São eles o GridSphere (2009a), que partiu da iniciativa do projeto GridLab (finalizado oficialmente em abril de 2005), e o *Open Grid Computing Environment (OGCE) Portal*.

Cada um dos *frameworks* possui características específicas, como facilitadores para o desenvolvimento e *portlets* pré-prontas que serão estudadas a seguir.

3.3.1 GridSphere

GridSphere é um projeto *open source* e gratuito, disponibilizado sob a licença Apache 2.0. O projeto utiliza a linguagem Java como padrão para desenvolvimento de *portlets* e tem como dependência o servidor de aplicação Apache Tomcat, no qual é homologado. Este projeto é considerado 100% de acordo com a especificação Java JSR-168, correspondente à especificação de *portlets* 1.0.

Sua maior contribuição no desenvolvimento de novas *portlets* é a facilidade na criação de um projeto para hospedar, compilar e realizar a instalação do mesmo no container. Como por exemplo, basta um comando para a construção da estrutura necessária para desenvolver uma *portlet*: “ant new-project”, embora o projeto criado na versão atual (3.1) possua um erro no momento de disponibilização, o qual é facilmente revertido (chamando a tarefa “deploy” invés de “install”). Outra contribuição importante para o desenvolvimento de novos projetos é a integração com projetos conhecidos no mercado, como Hibernate3, HSQLDB e o Spring Framework. Também há a integração com um projeto de controle padrão *Model View Controller* (MVC), chamado Struts (APACHE, 2009b), permitindo interconexão do modelo MVC com o container de *portlets*.

Ao criar o projeto a partir do template, também é criada uma *portlet* padrão, que pode ser usada como exemplo para o desenvolvimento. Também são fornecidos exemplos sob a forma de wiki (GRIDSPHERE, 2009b) para diminuir a curva de aprendizagem sobre o desenvolvimento de *portlets*.

Existe, no entanto, algumas desvantagens com estas facilidades de desenvolvimento, dentre elas, não fica claro o suficiente que deve ser criado um arquivo em “~/gridsphere/portlets/\$CONTEXTO”, sendo \$CONTEXTO o contexto em que a portlet desenvolvida está disponível. Também, por conta da automatização, o novo projeto não pode ser usado diretamente em outro container de *portlet* que não seja o próprio do GridSphere, pois seu descritor de aplicação (web.xml) é modificado para a integração com o container e, ao mesmo tempo, garantindo exclusividade do container GridSphere, além de usar os chamados *visual beans* e a biblioteca GridSphere User Interface (UI), que não são padrão no desenvolvimento de portlets.

É fornecido juntamente com o portal GridSphere, *portlets* de núcleo para o desenvolvimento de um portal, como login, logout, configuração de localização, configuração de perfil e customização de layout, além de portlets administrativas para a criação de usuários, grupos, gerência de portlets e customização do layout do portal. Também estão em desenvolvimento projetos que provêm portlets para uso com o *framework* GridSphere (2009c), como SRB Portlets, bluesquid, CMAG Portal, GAMA Portlet e Community Scheduler Framework.

3.3.2 OGCE

OGCE é um projeto *open source* e pode ser redistribuído se notificado, embora possua uma série de licenças estilo Apache/BSD (OGCE, 2009a). O projeto OGCE, diferentemente do GridSphere, não possui container de *portlet* próprio nem mesmo portal próprio, por isso, além do servidor de web Apache Tomcat, que também está na sua lista de servidores homologados, deve ser instalado juntamente com o uPortal, que faz o uso do Apache Pluto (APACHE, 2009a) como container de *portlets*. Neste caso, *portlets* padrão para o gerenciamento do portal são fornecidos pelo uPortal, enquanto que o OGCE fornece *portlets* para grade e meios para o desenvolvimento de outras *portlets* para grade.

Neste caso, também é possível a criação de um projeto modelo, no entanto, quem se responsabiliza por isso é o próprio portal, enquanto que o processo de integração com o OGCE se mantém manual. Dessa forma é possível determinar quais bibliotecas devem ser empacotadas com a *portlet* desenvolvida, pois estas somente são anexadas quando são necessárias, esclarecendo dúvidas sobre bibliotecas compartilhadas ao instalar em outro container de *portlets* qualquer. Também estão disponíveis exemplos de configuração de *portlets* para o portal, disponibilizados na web sob a forma de wiki (JA-SIG, 2009).

Foi observado que o descritor da aplicação web (web.xml) não precisa de qualquer modificação para ser instalado neste portal, pois as modificações necessárias são realizadas em tempo de instalação, permitindo que o mesmo pacote seja oferecido a outro container, sem prejudicar a interoperabilidade da *portlet*. Dessa forma, a configuração do contexto em que a *portlet* está instalada neste portal não é feita em arquivos, mas sim ao gerenciar a *portlet* e, embora pouco clara, permite a configuração em tempo de execução.

Embora o projeto Pluto da Apache já tenha releases compreendendo a versão 2.0 de *portlets*, esta versão ainda está em fase de testes e não é suportada pelo container, porém, a versão 1.0 da especificação de *portlets* já é 100% compatível.

São fornecidos pelo framework OGCE facilitadores (chamados *bridges* ou pontes) para conexão com outras tecnologias como JSF, Struts e Velocity, permitindo a melhor escolha para determinar como interagir com o usuário. Levando em consideração *portlets* pré-desenvolvidas, são fornecidas *portlets* pelo portal uPortal para gerenciamento de usuários, layout, canais e *portlets* instaladas, enquanto que o OGCE (2009b) fornece *portlets* para comunicação com alguns serviços Globus da grade pré-existentes, como, GRAM, GridFTP, GridShib, Resource Prediction Service e Resource Discovery Service.

3.4 Considerações finais

Ao descrever as vantagens e desvantagens dos frameworks para construção de *portlets* foi realizada uma comparação, no entanto, ao desenvolver em ambos os frameworks, constata-se que nada impede, após uma análise detalhada das dependências de cada um deles, que ambos sejam usados de forma simultânea.

Para a comparação entre os frameworks foram analisados os seguintes critérios que foram sistematizados no Quadro 2:

- licença: determina se o software pode ser usado em instituições educacionais ou comerciais;
- gratuito: se não existem custos de utilização;
- portabilidade: invés de portabilidade entre plataformas, esta é portabilidade entre portais, pois determina se será possível instalar as *portlets* desenvolvidas em um portal já em uso;
- facilitadores: determina se o desenvolvimento será atrelado a uma tecnologia específica ou se pode ser utilizadas outras tecnologias de diferentes desenvolvedores;

- JSR-168: o mínimo de compatibilidade necessária para executar uma portlet, deve ser garantido pelo portal uma confiabilidade aceitável com esta especificação;
- JSR-286: para o desenvolvimento de futuras portlets, será interessante não limitar as possibilidades, permitindo assim escolher um container mais próximo de futuros padrões;
- serviços de grade: com trabalhos de colegas em andamento, é interessante poder, por exemplo, permitir o uso de GridFTP através apenas da inclusão de uma portlet com esta capacidade.

Quadro 2 Comparação entre *frameworks* para desenvolvimento de *portlet*s

	GridSphere	OGCE
Licença	Apache 2.0	Apache/BSD
Gratuito	Sim	Sim
Portabilidade	Indiretamente	Sim
Facilitadores	Sim: Struts Bridge	Sim, Bridges: JSF, Struts, Velocity
JSR-168	Sim	Sim
JSR-286	Não	Instável na versão 2.4 (implementado pelo Apache Pluto)
Serviços de grade	Sim: SRB Portlets, bluesquid, CMAG Portal, GAMA Portlet, Community Scheduler Framework	Sim: Globus, GRAM, GridFTP, GridShib, Resource Prediction Service e Resource Discovery Service

Fonte: Autoria própria, 2009

Tendo estes critérios em pauta, em especial a portabilidade, para permitir a instalação em diversos containers de portlet e a possibilidade de construção de portlets nas tecnologias JSF e Velocity, a comparação entre os frameworks presente no Quadro 2 é determinante para a escolha do framework OGSA para a implementação, mesmo que seja possível a execução das portlets desenvolvidas também no container do GridSphere.

A portabilidade do *framework* GridSphere é indireta, pois ao iniciar o desenvolvimento de uma *portlet* com seu facilitador é criada uma dependência indireta com as bibliotecas do portal. O desenvolvimento de *portlet*s exige classes para controle de *portlet*s, porém, com facilitadores esta limitação é vencida, pois a comunicação do controlador da *portlet* com o *framework* de controle desejado é realizada através deste facilitador e, desta forma, quanto maior o número de facilitadores, menos restrito será o desenvolvimento.

Com relação à conformidade com a especificação JSR-286 que define *portlet*s 2.0, na versão atual do OGCE é possível a sua utilização, embora não recomendada, pois a especificação não está madura o suficiente tampouco corretamente atendida pelo container Apache Pluto.

Quanto aos serviços da grade, o *framework* do GridSphere relaciona seus projetos, os quais permitem integrações de *portlets* com grades específicas, enquanto o *framework* OGCE oferece *portlets* que já possuem estas capacidades, como o serviço GridFTP, que pode ser integrado ao portal AppMan devido ao desenvolvimento de forma paralela pelo colega Reis (2009).

4 MODELO

Partindo do objetivo de atender a necessidade de acesso aos recursos da grade de forma remota, sem maiores modificações na máquina do usuário e independentemente de plataforma, a solução proposta para a construção do portal será apresentada neste capítulo. Considerando-se os dois *frameworks* para a construção de *portlet*s analisados no capítulo anterior, se optou pelo uso do *framework* OGCE. Dentre as vantagens do uso deste *framework* estão bibliotecas que permitem a conexão com outros projetos de mercado (JSF, Velocity, Struts) e fácil portabilidade, sem implicar em dependências com o núcleo de *portlet*s de qualquer portal, baseando-se unicamente na especificação. Embora o GridSphere atenda às questões básicas necessárias para o desenvolvimento do modelo como a conformidade com a especificação JSR-168, é mais limitado em relação a outros aspectos como a instalação em outros portais e facilidades para desenvolvimento, como a falta de suporte ao facilitador de visualização Velocity, adotado no desenvolvimento do modelo.

Através do uso do *framework* OGCE, as limitações deste serão assumidas pela *portlet* modelada para o AppMan. Também serão parte das limitações da *portlet* as restrições no desenvolvimento orientado à web, que é suscetível ao navegador. Foram observadas durante o uso do protótipo AppMan dificuldades para integração por serem usados arquivos de texto para determinação de aplicação em execução e momento da finalização da aplicação, que juntamente com o serviço de LDAP utilizado pelo ambiente de execução, permitem identificar as aplicações em execução na grade. A linguagem de programação escolhida para o desenvolvimento é Java, pois os portais analisados e seus *frameworks* são desenvolvidos em Java. Além disso, o próprio AppMan está implementado em Java, o que deve facilitar na integração do portal com o AppMan.

Os requisitos identificados para o modelo e a forma como será realizada a interconexão com os componentes necessários serão apresentados nas seções subseqüentes.

4.1 Configurações do portal

Para que seja possível o acesso a um portal, inicialmente é necessário um servidor web, como já foi ressaltado na Seção 3.4, constatando que é possível usar o servidor Apache Tomcat para ambos os portais. Este servidor é parte do projeto Apache e de livre distribuição atualmente sob a versão 2.0 da licença Apache. O encapsulamento do portal, container e *portlet* é dado conforme a Figura 4, onde fica clara a dependência entre os componentes, bem como a forma em que será possível a execução das portlets do AppMan em outros portais, como o portal do GridSphere.

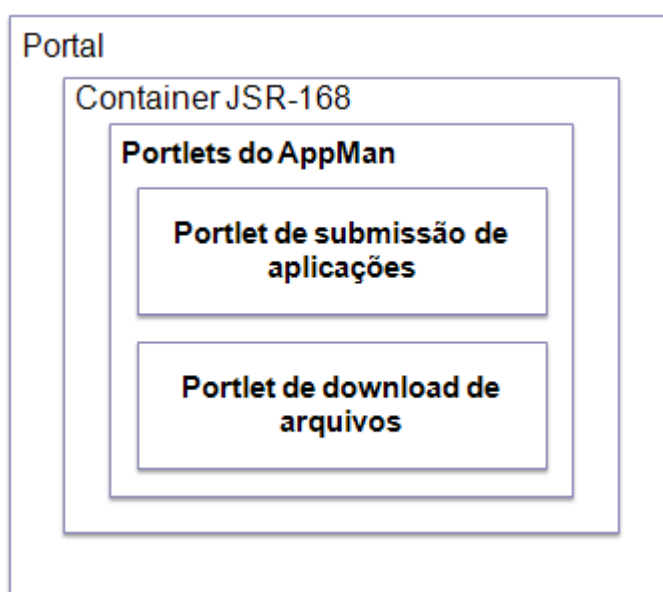


Figura 4 Encapsulamento dos componentes e dependências no portal

Fonte: Autoria própria, 2009

Em relação ao servidor web, o portal é uma aplicação capaz de integrar com outras aplicações web, o que pode ser considerado uma falha de segurança se o servidor não for configurado corretamente. Para realizar esta configuração no servidor Tomcat, é necessária a configuração do atributo *crossContext* do arquivo */conf/context.xml*, como mostra a Figura 5.

```
<Context crossContext="true">
  <WatchedResource>WEB-INF/web.xml</WatchedResource>
</Context>
```

Figura 5 Trecho do arquivo context.xml de configuração do Tomcat

Fonte: Autoria própria, 2009

Mesmo assim, o serviço web não fornece qualquer informação sobre quais aplicações web estão sendo executadas ou o contexto destas nem mesmo para a aplicação web do portal, que faz uso da propriedade *crossContext*. Por isso, para que o portal localize as *portlets* e seus contextos, no portal GridSphere (Apêndice C) são usados arquivos de configuração, enquanto que no portal uPortal (Apêndice B) a localização das *portlets* e respectivos contextos é configurada em tempo de execução, através da área administrativa do portal.

A configuração dos contextos para o GridSphere se dá através de arquivos localizados em *~/.gridsphere/portlets*. Neste diretório, basta criar um arquivo vazio com o nome do contexto, onde a extensão é opcional e pode ser uma prioridade numérica para o carregamento, caso haja alguma dependência entre *portlets*. Já no uPortal a configuração é realizada no descritor da *portlet*, encontrado na área administrativa do portal, como apresenta a Figura 6.

Portlet Descriptor: Enter the Portlet web-application path and portlet name.

Options	User can Modify?	General Settings
?	<input type="checkbox"/>	Framework Portlet: [example - false] false
?	<input type="checkbox"/>	Portlet Web Application Path: [example - /pluto-testsuite] /appman-portlets
?	<input type="checkbox"/>	Portlet Name: [example - test-port-et-1] appman-jobsubmission

< Back Next > Review Cancel

Figura 6 Configuração do descritor da portlet no uPortal

Fonte: Autoria própria, 2009

Ambos os portais precisam de uma base de dados para manter registros de usuário e customizações realizadas pelos usuários e administradores. Esta base é armazenada em um banco de dados configurável pelo servidor web, no entanto, a configuração padrão usa o banco de dados HSQLDB inicializado como servidor, permitindo acesso mesmo após a

finalização do servidor web. Este mesmo servidor de banco de dados pode ser usado para armazenamento de dados referente às *portlets* desenvolvidas para o AppMan.

4.2 Portlets de integração com o AppMan

A finalidade de um portal para grade é permitir fácil interação com os componentes da grade a partir de um computador remoto, sem a necessidade de instalação de qualquer pacote na máquina do usuário, apenas através do navegador da web. Além deste objetivo principal, o portal pode oferecer muitos outros recursos pertinentes ao bom uso da grade.

A modelagem das *portlets* de integração com o AppMan visa a instalação em diferentes portais, não apenas nos sugeridos pelos *frameworks* analisados, como também nos portais da Sun (GlassFish), IBM (WebSphere) ou Oracle (OracleAS Portal Version). Para esta compatibilidade as *portlets* foram desenvolvidas utilizando o *framework* OGCE, que está em conformidade com a especificação JSR-168. Para que a *portlet* seja instalada em um portal, este portal deve se basear em um container de *portlets* compatível com a especificação JSR-168.

As *portlets* de integração apresentam recursos como lista de aplicações em execução, aplicações finalizadas, dados relativos ao tempo de execução, busca de arquivos da grade em tempo de execução e pós-execução. A *portlet* também gerencia a execução de uma aplicação por vez, controlando a taxa de submissão de aplicações para a grade, e evitando intercalação de tarefas de aplicações distintas, o que poderia tornar os arquivos de log confusos e o tempo de execução de uma aplicação maior do que o esperado. Estes recursos serão apresentados nas seções subseqüentes, com maiores explicações sobre o desenvolvimento, usabilidade e utilidade de cada recurso.

4.2.1 Controle de submissão de aplicações

O portal, como um serviço, deve estar disponível a qualquer momento e, dessa forma, deve aceitar a submissão de aplicações independentemente do estado de execução das aplicações na grade. O aceite da aplicação por parte do portal não indica que a execução da mesma será imediata.

Para este controle foi desenvolvido um componente de planejamento de submissão, que verifica se os pré-requisitos para a execução da aplicação no AppMan estão satisfeitos. Os pré-requisitos são: não devem existir aplicações em execução, pois pode haver uma aplicação

inicializada como script e não gerenciada pelo portal; e se existem aplicações submetidas ao portal, porém não submetidas ao AppMan. Não devem existir aplicações em execução na grade, pois a intercalação destas deixaria os registros de log inconsistentes tanto quanto o tempo de execução na grade. Enquanto existirem aplicações registradas no portal, o componente de planejamento é encarregado de submeter estas à grade uma por vez.

Uma vez a aplicação submetida à grade, não há bloqueio no processo responsável pela submissão, pois a execução passa para a grade e o processo é liberado. Para esta submissão é usada a Infra-estrutura de Suporte às Aplicações Móveis (ISAM), que é parte do ambiente de execução EXEHDA. O ISAM faz uso apenas de um serviço de *gateway* do EXEHDA, para inicializar a execução no ambiente e, logo após, retorna a execução. Sem o bloqueio do processo de submissão, não há forma segura de saber quando a aplicação finalizou sua execução e, por isso, foram necessárias modificações no código do AppMan para realizar a integração com o portal.

A modificação realizada no AppMan permite que um parâmetro opcional seja enviado no momento de inicialização da aplicação. Este parâmetro é o identificador único atrelado à aplicação no momento da submissão ao portal. Este identificador é gerado pelo banco de dados (BD) do portal (Apêndice G), e é usado pelo AppMan para atualizar o registro no BD com o identificador da aplicação no *middleware* EXEHDA. Este identificador também é usado para registrar o momento de finalização da execução da aplicação na grade. O componente de planejamento do portal por sua vez, verifica se o AppMan sinalizou o fim da aplicação a cada período de tempo pré-determinado pela *portlet* e, dessa forma, uma aplicação subsequente pode demorar até este tempo limite de verificação para ser inicializada.

4.2.2 Manipulação de aplicações

Um recurso interessante para portais de grade é manter histórico das aplicações submetidas, bem como dados de controle. No caso do AppMan, uma vez que este é inicializado no ambiente de execução, lhe é concedido um identificador da aplicação, porém, para o portal isso não é o suficiente, pois há a necessidade de mapear aplicações antes da inicialização da mesma na grade, o que levou a criar um identificador único para as aplicações no momento da submissão.

Também é guardado o usuário do portal que submeteu a aplicação. Dependendo da configuração do portal, no caso de uma instalação para testes, a submissão de tarefas pode ser disponibilizada mesmo antes da autenticação no portal, o que levará ao não armazenamento

do usuário que submeteu a aplicação. Em uma grade em produção tal alternativa deve ser descartada no momento de configuração por questões de segurança.

Para registro de inicialização e finalização das aplicações são armazenadas datas de início e fim da execução, sendo a data de início determinada pelo componente de planejamento, ao identificar uma aplicação eleita para submissão à grade, enquanto que a data de fim é atribuída pelo final da execução do AppMan. Com estas informações disponíveis, também é apresentado o tempo de duração de determinada aplicação. O estado atual da aplicação também é determinado a partir da data de início e fim, convergindo para os seguintes estados:

- Aguardando: não há data de início ou fim;
- Executando: existe data de início, porém não há data de fim;
- Finalizado: datas de início e fim preenchidas.

Também é apresentado um indicador de sucesso da execução. Este indicador pode ter valor negativo se for identificada uma falha em qualquer ponto da execução, como no momento de submissão da aplicação ao *middleware*, que pode não estar em execução, ou, ao finalizar a aplicação, se uma das tarefas instanciadas pelos TMs não tiver data de finalização, o que caracteriza falha não detectada na execução.

Lista de tarefas em execução						
<input type="checkbox"/>	Identificador	Tarefa	Usuário que submeteu	Data de início	Data de término	Sucesso
<input type="checkbox"/>	15	t10.dag	admin	09/05/2009 02:28:14	09/05/2009 02:29:01	Finalizado Sim
<input type="checkbox"/>	17	t10.dag	admin	09/05/2009 02:36:16	09/05/2009 02:36:56	Finalizado Sim
<input type="checkbox"/>	18	t10.dag	admin	10/05/2009 02:21:27	10/05/2009 02:22:46	Finalizado Sim
<input type="checkbox"/>	22	t10.dag	admin	10/05/2009 03:14:23	10/05/2009 03:16:38	Finalizado Sim
<input type="checkbox"/>	23	t10.dag	admin	10/05/2009 03:17:54	10/05/2009 03:18:14	Finalizado Sim
<input type="checkbox"/>	25	t10.dag	admin	10/05/2009 03:50:55	10/05/2009 03:50:55	Finalizado Não
<input type="checkbox"/>	28	t10.dag	admin	10/05/2009 04:03:45	10/05/2009 04:04:09	Finalizado Sim
<input type="checkbox"/>	32	t10.dag	admin	10/05/2009 04:51:47	10/05/2009 04:52:11	Finalizado Sim
<input type="checkbox"/>	33	t10.dag	admin	10/05/2009 04:52:27	10/05/2009 04:52:51	Finalizado Sim
<input type="checkbox"/>	35	t10.dag	admin	10/05/2009 04:56:48	10/05/2009 04:57:12	Finalizado Sim
<input type="checkbox"/>	36	t10.dag	admin	10/05/2009 04:57:29	10/05/2009 04:57:52	Finalizado Sim
<input type="checkbox"/>	37	t10.dag	admin	10/05/2009 04:58:09	10/05/2009 04:58:33	Finalizado Sim

Ações
<ul style="list-style-type: none"> • Nova tarefa • Atualizar status • Remover tarefas marcadas • Informações

Figura 7 Portlet de manipulação de aplicações

Fonte: Autoria própria, 2009

É possível observar a *portlet* em questão na Figura 7, apresentando as informações descritas no formato de colunas para cada aplicação. Nesta *portlet* também há a possibilidade de exclusão de aplicações. No caso de exclusão, é solicitada a confirmação e, ao excluir uma aplicação que está em execução, também é cancelada a execução da mesma no AppMan, evitando aplicações com estado inconsistente com a visualização do portal.

4.2.3 Download de arquivos da grade

Uma vez que as aplicações são submetidas em formato GRID-ADL, em arquivos de extensão DAG, o download do arquivo submetido ou das informações geradas pelas tarefas pode ser útil na depuração de algum problema específico de uma tarefa ou mesmo para a validação do arquivo submetido.

Para suprir esta necessidade foi desenvolvida uma *portlet* (Figura 8) para download de arquivos de uma determinada aplicação.

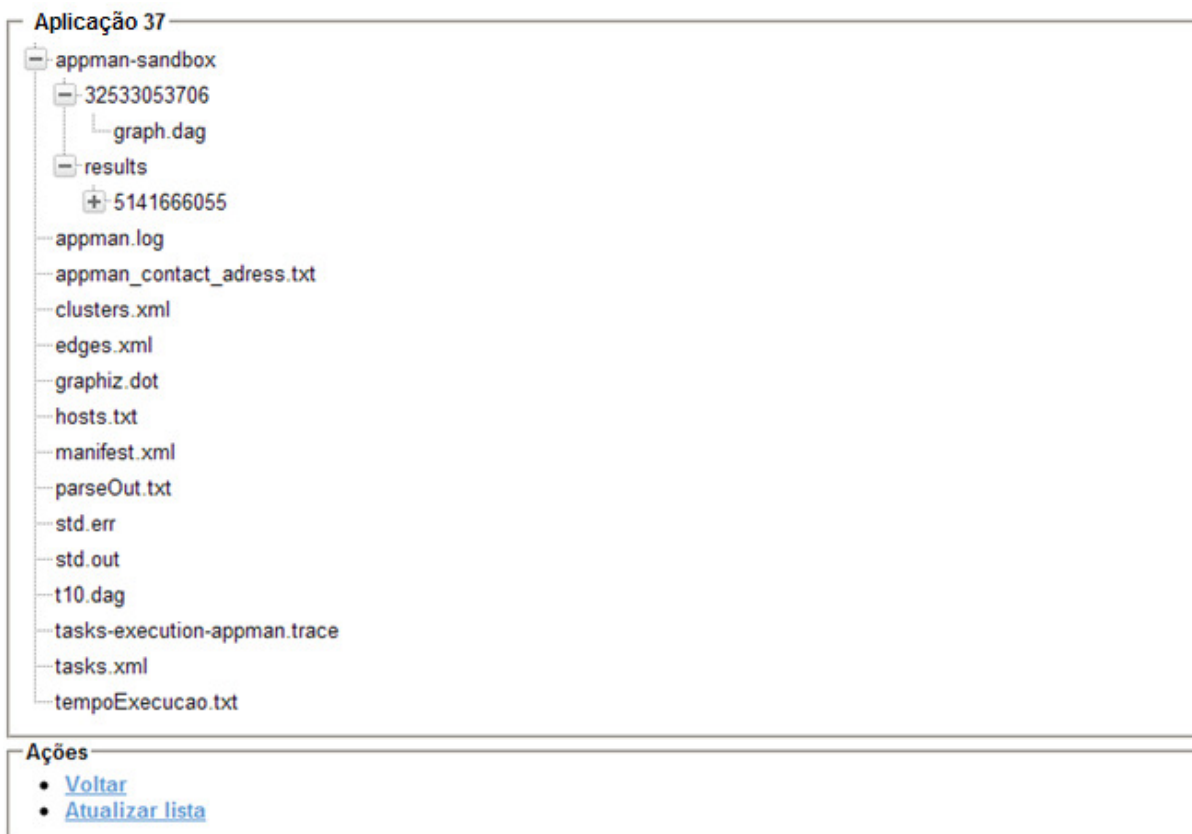


Figura 8 Portlet para download de arquivos de uma aplicação

Fonte: Autoria própria, 2009

Existe neste caso um detalhe de implementação, pois como será explicado na próxima seção, o AppMan usa apenas um diretório para armazenar arquivos de registros e da grade, no entanto o portal deve mostrar aplicações que não estão em execução, por isso, é realizada uma junção dos arquivos de aplicações já executadas com os arquivos da aplicação em execução, se for o caso da visualização de uma aplicação em execução.

4.3 Funcionamento

O funcionamento do portal depende da forma como este foi integrado ao AppMan e, conseqüentemente, ao funcionamento do AppMan em seu ambiente de execução. As interações entre o portal e o AppMan podem ser observadas na Figura 9.

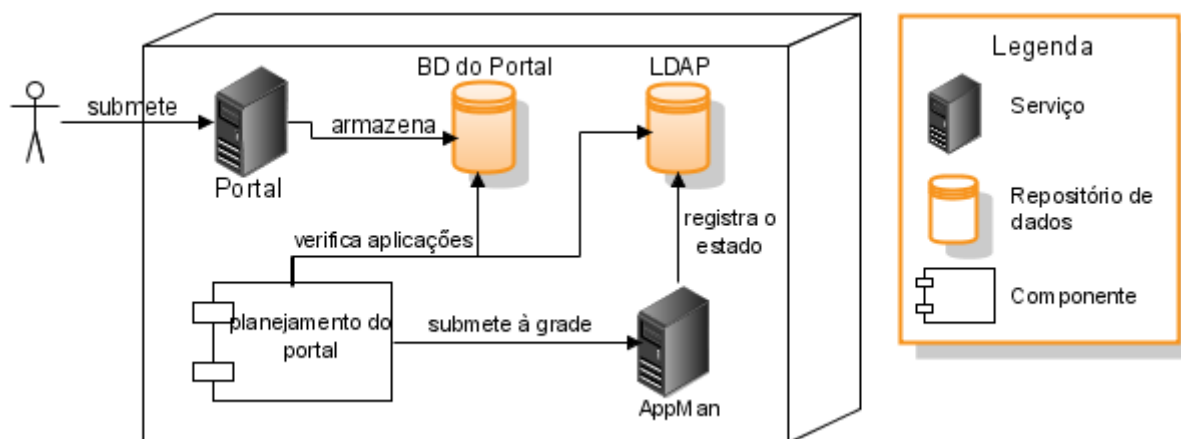


Figura 9 Esquema de integração do Portal com o AppMan

Fonte: Autoria própria, 2009

Neste esquema é apresentado um componente que pertence ao ambiente de execução EXEHDA, o qual o AppMan faz uso, que é o repositório LDAP. Este componente é utilizado pelo serviço de informações e também registra o estado das aplicações em execução.

Inicialmente, uma aplicação deve ser submetida à grade usando o portal. Assim que esta aplicação for aprovada, será gerado um identificador único para o mesmo através do banco de dados do portal. O estado da aplicação passará para “aguardando” e o arquivo descritor será gravado em um diretório de controle, configurado pelo portal. Este diretório de controle também contempla o identificador único da aplicação, para não haver sobreposição de arquivos das aplicações.

O componente de planejamento garante algumas pré-condições para a submissão de aplicações à grade:

- Aplicações não inicializadas: inicialmente, são buscadas aplicações não inicializadas no banco de dados do portal;
- Aplicações em estado “executando”: não devem ser submetidas aplicações enquanto houver uma aplicação executando na grade, para evitar conflito de arquivos, já que apenas um diretório é usado para a saída de arquivos durante a execução;
 - São pesquisadas aplicações com estado “executando” no banco de dados do portal; e
 - Também são pesquisadas aplicações com estado “executando” no LDAP, pois pode haver uma aplicação em execução inicializada via script;

Por fim, após o cumprimento das pré-condições, as aplicações não inicializadas são submetidas à grade em ordem cronológica.

Assim que a execução de uma determinada aplicação finalizar e o componente de planejamento identificar tal estado, os arquivos desta aplicação são movidos para o diretório de controle, para que o diretório de saída padrão possa ser usado por uma nova aplicação, pois apenas um diretório é usado como saída padrão durante a execução da grade. Este diretório contém, além de um diretório para os arquivos gerados pela aplicação, informações de depuração e notificação do estado das tarefas em execução na aplicação.

4.4 Considerações finais

Suprindo as necessidades básicas do usuário de grades através de um portal para o AppMan, pode-se ter no futuro mais adeptos deste gerenciador de aplicações, uma vez que é conveniente ao usuário a facilidade de acesso e manipulação das aplicações através de um navegador de internet, sem qualquer instalação na máquina do usuário. Entretanto, esta infraestrutura deve ser da mais alta confiabilidade, procurando evitar assim qualquer manutenção ou necessidade de acesso direto à grade.

Os componentes usados no portal estão sujeitos a falhas. Por exemplo, a indisponibilidade do portal ocorrerá, evitando o acesso ao mesmo como um todo, caso o banco de dados do portal seja finalizado ou falhar enquanto o portal ainda está em funcionamento. No caso do *middleware* ISAM/EXEHDA utilizado no AppMan, caso o serviço não estiver inicializado, as aplicações permanecerão no estado aguardando, já que não será possível a submissão. Caso houver uma falha na execução no AppMan ou falha no ambiente EXEHDA, a aplicação em execução será mantida no estado em execução indefinidamente, até a exclusão da mesma no portal. Este último aspecto ocorre porque o modelo GRAND foi planejado para ter um Sistema de Gerenciamento de Recursos (RMS) no controle dos *clusters* ou redes locais, os quais já possuem tratamento a falhas. Tendo estas perspectivas em mente foi realizada a avaliação do portal, com a finalidade de detectar os impactos da instalação do portal, possibilitando a melhor escolha, seja ela com acesso via portal ou scripts, dependendo do caso em estudo.

5 AVALIAÇÃO

A avaliação do portal do AppMan leva em consideração aspectos como infraestrutura, configuração e métricas, para medição de *overhead* causado pela sua execução. As métricas e configurações apresentadas neste capítulo foram realizadas com sucesso em máquinas presentes no laboratório 24 horas do Unilasalle, com a configuração apresentada no Quadro 3, interligadas na mesma rede de computadores.

Quadro 3 Configuração das máquinas da grade

Processador	Pentium 4 2.26GHz
Memória	768MB
Sistema Operacional	Xubuntu 8.10 "Intrepid Ibex" - Release i386

Fonte: Autoria própria, 2009

Conforme o Apêndice A, foi configurado inicialmente o AppMan na máquina de acesso à grade, onde é inicializado o componente *Application Manager*. Nesta máquina foi instalado a versão 1.5 do JDK da Sun, o servidor LDAP e o servidor NFS. Também foi configurado nesta máquina o servidor LDAP, o servidor NFS e o arquivo de serviços do EXEHDA (*exehda-services.xml*). A configuração do arquivo *gridnodes.properties*, que determina as máquinas que serão usadas por cada componente também foi feita, além de ter de ser recompilado o projeto do AppMan e o arquivo gerado movido para o diretório configurado no serviço BDA do EXEHDA. Nas demais máquinas apenas o JDK e o EXEHDA foram instalados.

Também foi instalado e configurado o portal na máquina de acesso à grade. Embora seja possível a instalação em uma segunda máquina, devido à escassez de recursos, o portal foi instalado na mesma máquina de ponto de entrada na grade. Conforme o Apêndice B e Apêndice D, foi instalado o *container* de portlets uPortal e inicializado. Antes de inicializar o portal foi inicializada a base de dados correspondente, pois esta é necessária para carregamento de informações sobre customização e permissões de acesso. Também foi realizada a compilação do projeto *appman-portlets* e sua configuração pelo arquivo *~/appman-portlets/env.properties*. É possível visualizar as telas das portlets do AppMan em

uso nos portais uPortal e GridSphere através do Apêndice J, embora a avaliação foi realizada com uma instalação do portal uPortal.

As aplicações enviadas à grade são bastante simples, para testar apenas o tempo de controle de aplicações na grade, obtendo um resultado que evidencie o custo adicional (*overhead*) do portal, pois com aplicações simples, grande parte do tempo de execução da grade é consumido pelo *overhead* gerado pela simples submissão e controle da aplicação. A aplicação que será utilizada está disponível no Apêndice E.

Inicialmente serão apresentados resultados obtidos da execução de aplicações via script, para avaliação do tempo normal de execução da grade. Enquanto que na seção 5.2 será avaliada a execução das aplicações quando submetidas através do portal.

5.1 Execução via scripts

A execução de aplicações através de scripts pode ser realizada através do arquivo *run-appman.sh*, encontrado no projeto *appman-project*, que basicamente executa o seguinte comando a partir do diretório de instalação do exehda: `bin/isam-run appman-bin/appman-console.isam -- appman-bin/dag/t10.dag`

Foram realizadas execuções com 2, 4 e 6 nós interligados na grade. Destas execuções, a partir do particionamento das tarefas, temos um determinado número de componentes inicializados baseados neste particionamento automático. A quantidade inicializada de cada um destes componentes está descrita na Tabela 1.

Tabela 1 Componentes do AppMan instanciados na grade

Número de nós	Quantidade de AMs	Quantidade de SMs	Quantidade de TMs
2	1	1	2
4	1	1	4
6	1	1	6

Fonte: Autoria própria, 2009

Conforme visto na seção 2.4, o componente AM serve como ponto de entrada na submissão de aplicações, o SM é responsável pelo encaminhamento das tarefas, e o TM é encarregado de executar as tarefas nos nós da grade através de um sistema de gerenciamento de recursos (RMS).

Para cada execução, a partir dos logs foram buscados os totais de tempos de execução e foi também confirmado o sucesso na execução das tarefas. Foram elaboradas 31 execuções

para obter um total de 30 execuções após descartar a primeira execução por questões de *caching* e otimização da execução da aplicação. Com 30 execuções os resultados se mostraram com baixo desvio padrão, considerando esta quantidade um bom número de execuções. Para o total das 31 execuções conforme Apêndice H, foi isolado o tempo de execução da primeira execução, para confirmação de impacto sobre o tempo total, calculado o desvio padrão e média aritmética de todas as execuções e o desvio padrão e média aritmética para todas as execuções com exceção da primeira, como pode ser observado na Tabela 2 e Figura 10.

Tabela 2 Síntese dos tempos de execução via script

Número de nós	Primeira execução	Média	Desvio padrão	Média s/ 1ª	Desvio padrão s/ 1ª
2	66.183	65.624,3	111,2	65.605,6	94,4
4	51.292	50.663,1	129,6	50.642,1	112,8
6	41.248	40.660,5	112,7	40.640,9	94,2

Fonte: Autoria própria, 2009

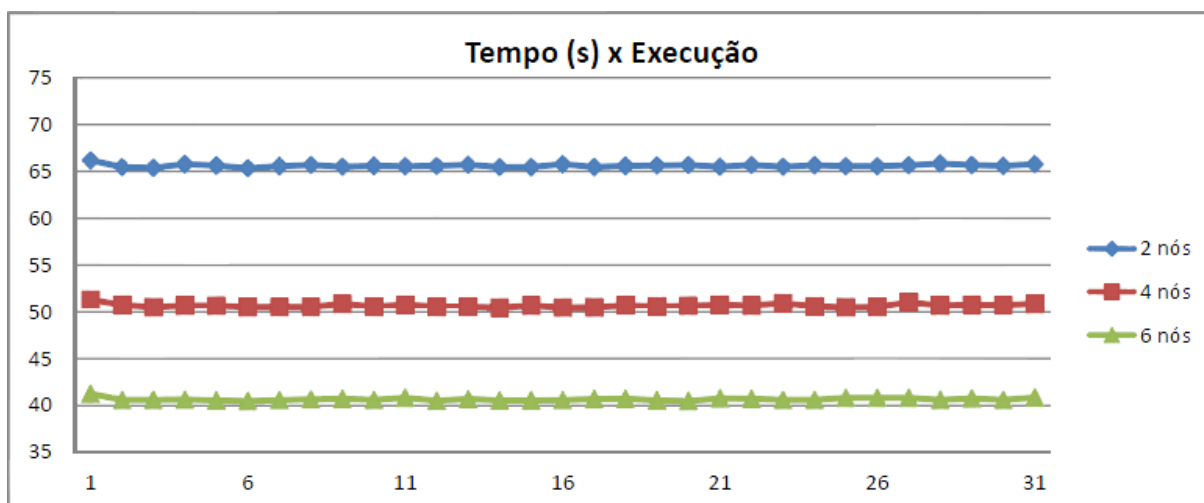


Figura 10 Tempo (s) de cada execução via script

Fonte: Autoria própria, 2009

A Figura 10 revela que as execuções ocorreram com normalidade ao longo do tempo, tendo uma pequena elevação no tempo de execução da primeira aplicação, por questões de otimização e configurações de *buffer* de certas partes da execução, consumindo em torno de 500 a 600 milissegundos a mais que os tempos das aplicações subseqüentes.

5.2 Execução via portal

Uma vez o portal configurado e inicializado, o AppMan já pode receber aplicações através do mesmo. O componente de planejamento é inicializado juntamente com o portal e, se houverem aplicações na fila, estas serão encaminhadas à grade.

A submissão destas aplicações pelo portal é relativamente simples, pois não é possível criar a aplicação a partir do portal, logo, deve ser submetido diretamente um arquivo na extensão DAG, que descreve as tarefas a serem executadas. A aplicação submetida através do portal é a mesma que foi submetida através de scripts, disponível no Apêndice E.

Para cada execução, a partir da tabela de controle do portal, foram buscados os totais de tempos de execução e foi também confirmado o sucesso na execução das tarefas. Para o total de 31 execuções conforme Apêndice I, igualmente à execução via scripts, foi isolado o tempo de execução da primeira execução, para confirmação de impacto sobre o tempo total por questões de *caching* e otimização da execução da aplicação. Também é calculado o desvio padrão para identificação de anomalias e a média aritmética de todas as execuções para identificar o tempo médio de execução pelo portal. Estas informações podem ser observadas na Tabela 3 e na Figura 11, o tempo de execução de cada aplicação.

Tabela 3 Síntese dos tempos de execução via portal

Número de nós	Primeira execução	Média	Desvio padrão	Média s/ 1ª	Desvio padrão s/ 1ª
2	71.629	70.814,4	159,9	70787,3	129,5
4	56.850	55.890,7	169,5	55858,8	134,7
6	46.652	45.855,7	174,6	45829,1	146,9

Fonte: Autoria própria, 2009

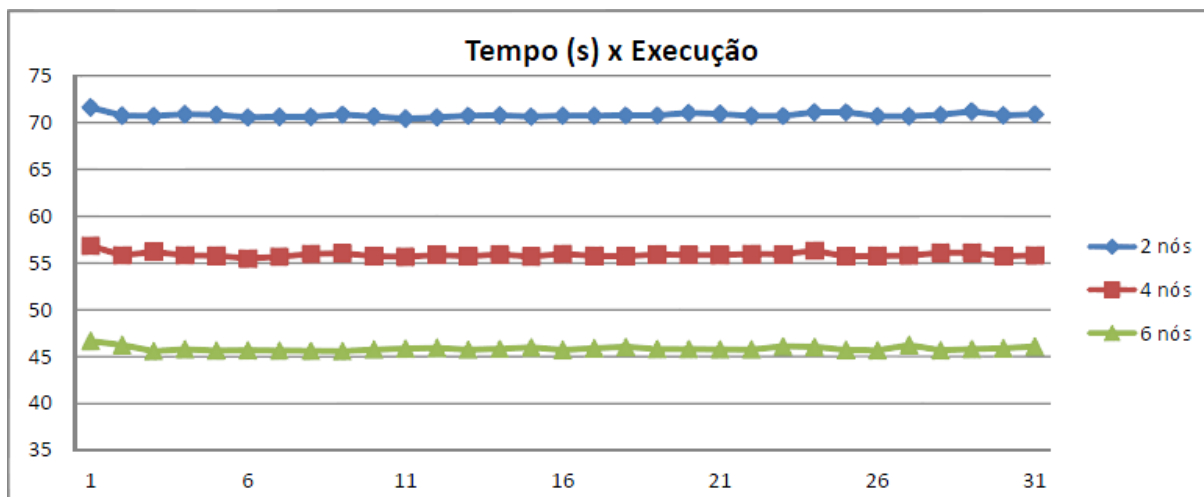


Figura 11 Tempo (s) de cada execução via portal

Fonte: Autoria própria, 2009

Também é possível observar que a primeira execução das aplicações no ambiente através do portal possui maior impacto e acaba por gerar tempos levemente superiores às execuções subsequentes, com *overhead* de 800ms a 1s. No caso da execução do portal também há otimização de buffers internos, o que leva a um tempo menor nas execuções subsequentes, entretanto, ainda há uma leve divergência entre o *overhead* da primeira execução pelo portal com a primeira execução através de scripts. Esta divergência pode ser atribuída à carga na máquina de submissão, onde o portal está sendo executado, levando a maior consumo de memória e tempo para adaptar a primeira execução.

5.3 Considerações finais

Através das avaliações realizadas na execução do AppMan com a ajuda de scripts e da instrumentação do portal foram obtidos dados cruciais para a determinação do *overhead* implicado pelo uso do portal. Ao visualizar a Tabela 4 observa-se a diferença nas médias de execução das duas formas de submissão avaliadas.

Tabela 4 Diferença no tempo de execução entre a submissão via Scripts e via Portal

Nº de nós	Médias		Diferença
Nº de nós	Scripts	Portal	
2	65.624,3	70.814,4	5.190,14
4	50.663,1	55.890,7	5.227,60
6	40.660,5	45.855,7	5.195,20

Fonte: Autoria própria, 2009

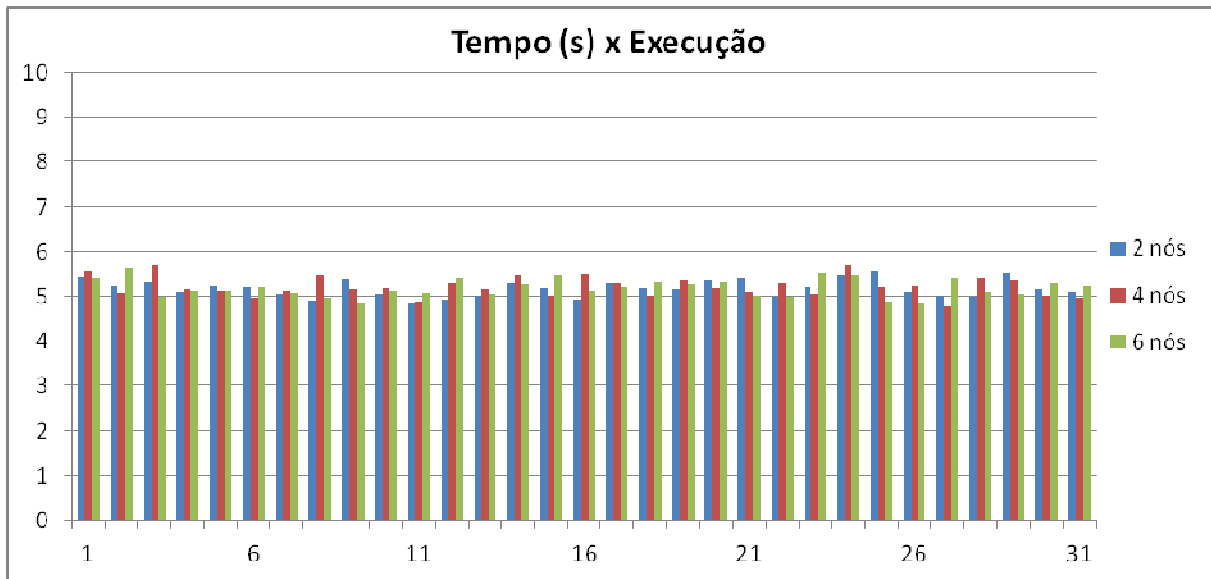


Figura 12 Diferença no tempo (s) de execução em cada submissão

Fonte: Autoria própria, 2009

Esta diferença computada entre o tempo de execução via script e o tempo de execução via portal deve ser próxima de uma constante para cada aplicação, independente de quantos nós estão sendo usados na grade. Esta constante é de escopo de ambiente, ou seja, para cada ambiente e aplicação o *overhead* possuirá um valor próximo de uma constante. É possível perceber o comportamento constante da aplicação na Figura 12, onde a diferença dos tempos de execução com dois, quatro ou seis nós, é muito próxima de 5s.

6 CONCLUSÃO

Este trabalho apresentou duas abordagens para a construção de um portal para o protótipo AppMan do modelo GRAND: a transformação do AppMan em um serviço OGSA para adotar uma portlet já existente ou a criação de portlets para a submissão e controle das aplicações via portal sem alterar significativamente o protótipo. O conhecimento do funcionamento do protótipo e sua interação com o middleware EXEHDA determinou o descarte da migração para um serviço OGSA, já que seriam necessárias modificações muito significativas no protótipo o que prejudicaria a integração deste trabalho com as outras pesquisas em andamento no projeto.

Após uma seleção prévia dos principais *frameworks* para construção de portais, foi realizada uma comparação dos frameworks GridSphere e OGCE com a finalidade de determinar o framework mais adequado a ser usado na construção de um portal para grade. Alguns fatores, como portabilidade e interconexão com outros frameworks, foram decisivos na escolha do framework OGCE, guiando a modelagem da solução para o portal do AppMan.

Durante a concepção do modelo, foi feito o levantamento dos requisitos bem como foi modelada a integração do portal com o AppMan de forma a permitir que a sua execução através de scripts permaneça. Após a concepção e alguns testes da solução a mesma foi avaliada e constatou-se que o portal acabou por gerar um *overhead* relativamente pequeno no ambiente em questão e dessa forma, lembrando as vantagens da solução do portal, é considerado de grande viabilidade o uso do mesmo para manipular a grade.

Para o bom aproveitamento do portal, este deve estar disponível grande parte do tempo, e para isso deve-se ter um cuidado especial com a infra-estrutura. No decorrer do desenvolvimento deste trabalho foram descobertas carências do protótipo, ocasionadas principalmente pelo uso de uma versão desatualizada do middleware ISAM/EXEHDA. Ao longo do desenvolvimento das portlets, também foram feitas correções e otimizações no AppMan, permitindo o desenvolvimento e execução de forma mais estável. Esta também pode ser considerada uma contribuição importante principalmente para o projeto GRAND, mas que também teve impacto no portal propriamente dito.

A limitação mais crítica detectada era o alto grau de instabilidade e freqüente “travamento” gerado pelo ambiente de execução. O principal motivo desta instabilidade era o método de detecção de terminação da aplicação que utilizava um mecanismo de *polling*, onde é buscado o estado da execução a cada intervalo de tempo. Este problema não é mais um limite, porém, qualquer aplicação que deve permanecer executando por um longo período de tempo, como é o caso do ambiente de execução EXEHDA, deve se preocupar com o consumo de memória.

Ao que diz respeito do consumo de memória, foi detectada uma falha no ambiente de execução, conhecida popularmente como *memory leak* (BOND; MCKINLEY, 2006) onde, a cada execução do AppMan, em torno de 5MB de memória são mantidos alocados, limitando o tempo de uso contínuo do protótipo. A correção deste problema exigiria a alteração do código do ISAM/EXEHDA, o qual fica como um importante trabalho futuro a ser realizado para aumentar a escalabilidade do AppMan.

As *portlets* implementadas para o AppMan são o mínimo necessário para a usabilidade de um portal de grades. São sugeridas algumas implementações de *portlets* para trabalhos futuros, como uma *portlet* para a construção do descritor da aplicação como um arquivo DAG de forma visual, uma vez que é difícil para algumas áreas desenvolver este tipo de informação. Também é interessante a construção de uma *portlet* para a visualização do estado das tarefas em execução na grade, uma vez que este recurso já existe ao executar a grade por via de scripts.

A averiguação da perda de memória informada nas limitações seria de grande valia para a sua correção, contribuindo assim para a estabilidade da grade e conseqüentemente, do portal. A correção deste problema depende estritamente do ambiente de execução EXEHDA, porém ao averiguar o problema é possível que este tenha fundamento em algum dos módulos, que podem ser reimplementados de forma correta.

Por fim, mesmo diante dos resultados positivos e avaliações de melhorias que foram encontrados por este trabalho, é considerado importante o desenvolvimento de novas pesquisas sobre o modelo GRAND, para fins de avaliação de desempenho e otimizações da grade, como por exemplo, em algumas situações onde é utilizada a técnica de *polling* para busca de estado. E para fins de comparação, também seria válido o acompanhamento da instalação das *portlets* do portal do AppMan em outros containers de *portlets* que venham a ser necessários.

REFERÊNCIAS

AIKEN, B., STRASSNER, J., CARPENTER, B., FOSTER, I., LYNCH, C., MAMBRETTI, J., MOORE, R. e TEITELBAUM, B. "Network Policy and Services: A Report of a Workshop on Middleware". Disponível em: <<http://www.ietf.org/rfc/rfc2768.txt>>. Acessado em: Junho de 2009.

ALLIANCE, Globus. "The Globus Alliance". Disponível em: <<http://www.globus.org/>>. Acessado em: Março de 2009.

Antipolis, Sophia. Unicore-UniGrids: Activities and strategies for Open Source Grids. GridCoord Workshop. 2005.

APACHE. "Apache Pluto Home Page". Disponível em: <<http://portals.apache.org/pluto/>>. Acessado em: Junho de 2009.

APACHE. "Apache Struts Home Page". Disponível em: <<http://struts.apache.org/>>. Acessado em: Junho de 2009.

BEESEON, Brett, MELNIKOFF, Steve, VENUGOPAL, Srikumar e BARNES, David G. A Portal for Grid-enabled Physics. Australasian Workshop on Grid Computing and e-Research, Australian Computer Society. 2005.

BERNARDO, Tonismar Régis. **Graduação:** Integração do Sistema AppMan de Gerenciamento de Aplicações para Ambiente de Grade com Diferentes Sistemas de Gerenciamento de Recursos. UNILASALLE. Canoas, RS, Brasil. 2008.

BIRN. "BIRN - Biomedical Informatics Research Network". Disponível em: <<http://www.nbirn.net/>>. Acessado em: Maio de 2009.

Bond, Michael D. e MCKINLEY, Kathryn S. Bell: Bit-Encoding Online Memory Leak Detection. ASPLOS'06. San Jose, California, USA. 2006.

BUSUOLI, Massimo. "BEinGRID: Textile Grid Portal ". 2009.

CACTUS. "Cactus Code". Disponível em: <<http://www.cactuscode.org/>>. Acessado em: Maio de 2009.

ESG. "Earth System Grid (ESG)". Disponível em: <<http://www.earthsystemgrid.org/>>. Acessado em: Maio de 2009.

FORUM, Open Grid. "Open Grid Forum Overview". Disponível em: <http://www.ogf.org/About/OGF_At-A-Glance.pdf>. Acessado em: Junho de 2009.

FOSTER, I., KISHIMOTO, H., SAVVA, A., BERRY, D., DJAOUI, A., GRIMSHAW, A., HORN, B., MACIEL, F., SIEBENLIST, F., SUBRAMANIAM, R., TREADWELL, J. e REICH, J. Von. The Open Grid Services Architecture, Version 1.5. Open Grid Forum. 2006.

FOSTER, Ian. "What is the Grid? A Three Point Checklist." Grid Today 1(6). 2002.

FOSTER, Ian e KESSELMAN, Carl. Computational Grids. The Grid: Blueprint for a New Computing Infrastructure. California San Francisco, USA, Morgan Kaufmann: 15-52. 1998.

FOSTER, Ian, KESSELMAN, Carl, NICK, Jeffrey e TUECKE, Steven. "The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration." 2002.

FOSTER, Ian, TUECKE, Steven e KESSELMAN, Carl. "The Anatomy of the Grid: Enabling Scalable Virtual Organizations." Intl J. Supercomputer Applications. 2001.

GEONGRID. "GEONGRID". Disponível em: <<http://www.geongrid.org/>>. Acessado em: Maio de 2009.

GRIDBUS. "The Gridbus Middleware". Disponível em: <<http://www.gridbus.org/middleware/>>. Acessado em: Junho de 2009.

GRIDCHEM. "Computational Chemistry Grid". Disponível em: <<https://www.gridchem.org/>>. Acessado em: Fevereiro de 2009.

GRIDS. "G-Monitor: Grid Resource Broker Portal for Managing and Monitoring Application Execution on Global Grids". Disponível em: <<http://www.gridbus.org/gmonitor/>>. Acessado em: Junho de 2009.

GRIDSPHERE. "GridSphere Funding Sponsors". Disponível em: <<http://www.gridsphere.org/gridsphere/gridsphere/guest/sponsors>>. Acessado em: Maio de 2009.

GRIDSPHERE. "GridSphere Portal: Portlet Development". Disponível em: <<http://docs.gridsphere.org/display/gs30/Portlet+Development+Guide>>. Acessado em: Maio de 2009.

GRIDSPHERE. "GridSphere Portlets". Disponível em: <<http://www.gridsphere.org/gridsphere/gridsphere/guest/portlets/r/>>. Acessado em: Maio de 2009.

JA-SIG. "uPortal 3 project workplan". Disponível em: <http://www.ja-sig.org/wiki/download/attachments/2240/workplan_public.pdf?version=1>. Acessado em: Maio de 2009.

JA-SIG. "Guide to Working with Portlets in uPortal". Disponível em: <<http://www.ja-sig.org/wiki/display/UPM31/04+Working+with+Portlets+in+uPortal>>. Acessado em: Maio de 2009.

JCP. JSR 168: Portlet Specification. Java Community Process. Disponível em: <http://developers.sun.com/portalserver/reference/techart/jsr168/pb_whitepaper.pdf>. Acessado em: Maio de 2009.

JCP. JSR 286: Portlet Specification 2.0. Java Community Process. Disponível em: <<http://cds-esd.sun.com/ESD43/JSCDL/portlet/2.0-fr/portlet-2.0-fr.zip>>. Acessado em: Maio de 2009.

JETSPEED. "Jetspeed 2 Home Page". Disponível em: <<http://portals.apache.org/jetspeed-2/>>. Acessado em: Junho de 2009.

KISHIMOTO, H. e TREADWELL, J. "Defining the Grid: A Roadmap for OGSA™ Standards: Version 1.0." 2005.

LEMOS, Daniel da Trindade. **Graduação:** Monitoramento de Recursos em Ambientes de Grade. UNILASALLE. Canoas, RS, Brasil. 2006.

LEMOS, Daniel da Trindade. Monitoramento de Recursos em Ambientes de Computação em Grade. ERAD 2007. Porto Alegre, RS, Brasil, Editora da UFRGS. 7ª Escola Regional de Alto Desempenho. 2007.

LQCD. "Lattice QCD Portal". Disponível em: <<http://lqcd.jlab.org/>>. Acessado em: Fevereiro de 2009.

MANGAN, Patrícia Kayser Vargas. **Pós-graduação:** GRAND: Um Modelo de Gerenciamento Hierárquico de Aplicações em Ambiente de Computação em Grade. Universidade Federal do Rio de Janeiro. Rio de Janeiro. 2006.

MCGOUGH, A. Stephen, LEE, William e DARLINGTON, John. "ICENI II Architecture." Proceedings of the UK e-Science All Hands Conference 2005. 2005.

NANOHUB. "nanoHUB: Simulation, Education, and Community for Nanotechnology". Disponível em: <<http://nanohub.org/>>. Acessado em: Maio de 2009.

NEXTGRID. "NextGRID: Architecture for Next Generation Grids". Disponível em: <<http://www.nextgrid.org/>>. Acessado em: Junho de 2009.

NOBRES, Wagner. **Graduação:** Análise do Impacto da Aplicação de Métricas de Qualidade de Software Orientado a Objetos no Desempenho de Sistemas Distribuídos de Alto Desempenho. UNILASALLE. Canoas, RS, Brasil. 2008.

NVO. "NVO: National Virtual Observatory". Disponível em: <<http://www.us-vo.org/>>. Acessado em: Maio de 2009.

OGCE. "OGCE - Licenses". Disponível em: <<http://www.collab-ogce.org/ogce/index.php/Licenses>>. Acessado em: Maio de 2009.

OGCE. "OGCE: Main Page". Disponível em: <http://www.collab-ogce.org/ogce/index.php/Main_Page>. Acessado em: Maio de 2009.

OMII-UK. "omii-uk: Software Solutions for e-Research". Disponível em: <<http://www.omii.ac.uk/>>. Acessado em: Maio de 2009.

PPDG. "Particle Physics Data Grid". Disponível em: <<http://www.ppdg.net/>>. Acessado em: Maio de 2009.

QUAKESIM2. "QuakeSim2". Disponível em: <<http://gf7.ucs.indiana.edu:8080/gridsphere/gridsphere>>. Acessado em: Maio de 2009.

REIS, Rodrigo de Souza. **Graduação:** Um modelo de otimização no gerenciamento dos dados para computação em grade. UNILASALLE. Canoas, RS, Brasil. 2009.

ROEHRIG, M., ZIEGLER, Wolfgang e WIEDER, Philipp. Grid Scheduling Dictionary WG (SD-WG). 2002.

SANTOS, Lucas A. S., REBONATTO, Marcelo T., VARGAS, Patrícia Kayser e GEYER, Cláudio F. R. Uma Proposta de Escalonamento Colaborativo de Aplicações num Ambiente de Computação em Grade. WSCAD 2005. Rio de Janeiro, RJ, Brasil. VI Workshop em Sistemas Computacionais de Alto Desempenho: p. 137-144. 2005.

SAVVA, Andreas, SUZUKI, Toshiyuki e KISHIMOTO, Hiro. "Business Grid Computing Project Activities." FUJITSU Sci. Tech. J. 40(2): 252-260. 2004.

SCIDAC, DOE. "National FusionGrid". Disponível em: <<http://www.fusiongrid.org/>>. Acessado em: Junho de 2009.

STOCKINGER, Heinz. Distributed database management systems and the data grid. Eighteenth IEEE Symposium on Mass Storage Systems. Hyatt Regency Islandia, San Diego, USA. 2001.

THAIN, D., TANNENBAUM, T. e LIVNY, M. Condor and the Grid. Grid Computing: Making The Global Infrastructure a Reality. John Wiley. 2003.

THOMAS, M., BARKER, C. J., BOISSEAU, J., DAHAN, M., REGNO, R., ROBERTS, E., SETH, A., URBAN, T. e WALLING, D. Experiences on Building a Component-Based Grid Portal Toolkit. Concurrency & Computation: Practice & Experience. 2005a.

THOMAS, M. P., BURRUSS, J., CINQUINI, L., FOX, G., GANNON, D., GILBERT, L., LASZEWSKI, G. von, JACKSON, K., MIDDLETON, D., MOORE, R., PIERCE, M., PLALE, B., RAJASEKAR, A., REGNO, R., ROBERTS, E., SCHISSEL, D., SETH, A. e SCHROEDER, W. "Grid portal architectures for scientific applications. In: Institute of Physics Publishing." Journal of Physics: Conference Series 16 (2005), SciDAC 2005.: 596-600. 2005b.

TUECKE, S., CZAJKOWSKI, K., FOSTER, I., FREY, J., GRAHAM, S., KESSELMAN, C., MAQUIRE, T., SANDHOLM, T., SNELLING, D. e VANDERBILT, P. Open Grid Services Infrastructure (OGSI) Version 1.0. GGF 2003. 2003.

WEHRENS, Oliver e BECK-RATZKA, Alexander. "GridSphere Project". Disponível em: <http://gks06.fzk.de/slides/G-Sphere-gek-Beck_Ratzka.pdf>. Acessado em: Maio de 2009.

APÊNDICE A Instalação do AppMan

Para a instalação do AppMan temos como pré-requisitos um servidor LDAP, um servidor NFS, a máquina virtual da Sun (JDK 1.5) e o projeto apache-maven. Seguem instruções para a instalação das dependências e configurações em um Linux Xubuntu 8.10.

Baixa a aplicação do repositório no googlecode.

```
$ svn checkout http://tcc-michel.googlecode.com/svn/trunk/AppMan/ tcc-
michel-read-only
```

Para facilitar a instalação, configuração e execução, devem ser definidas as seguintes variáveis de ambiente.

```
$ export APPMAN_HOME=/home/aluno/AppMan
$ export APPMAN_PROJECT=$APPMAN_HOME/appman-project
$ export EXEHDA_HOME=$APPMAN_PROJECT/exehda
```

Solicita instalação do servidor LDAP.

```
$ sudo apt-get install slapd ldap-utils db4.2-util
```

Deve ser alterado o arquivo de inicialização do SLAPD, localizado em /etc/default/slapd, pois a forma de configuração atualmente usada pelo LDAP é diretamente nos registros do LDAP, e usaremos configuração em arquivo.

```
$ sudo vi /etc/default/slapd
```

Configurar apenas a propriedade SLAPD_CONF, descrita abaixo.

```
SLAPD_CONF=/etc/ldap/slapd.conf
```

Renomear a pasta de configuração slapd.d, para usar a configuração do arquivo slapd.conf.

```
$ sudo mv /etc/ldap/slapd.d /etc/ldap/slapd.d.bkp
```

Copiar o arquivo exemplo \$APPMAN_HOME/config/slapd.conf para o diretório onde foi instalado o LDAP (ex: /etc/ldap/). Este arquivo de configuração é próximo do arquivo fornecido até as versões 2.2 do openldap, porém foram incluídas algumas configurações pertinentes à instalação do EXEHDA. A senha pré-configurada neste arquivo para o usuário rootdn="cn=admin,dc=exehda" é 1234.

```
$ sudo cp $APPMAN_HOME/config/slapd.conf /etc/ldap/
```

Copiar o arquivo \$APPMAN_HOME/config/exehda.schema para o diretório configurado no slapd.conf (ex: /etc/ldap/schema/exehda.schema).

```
$ sudo cp $APPMAN_HOME/config/exehda.schema /etc/ldap/schema/
```

Editar ou sobrescrever o arquivo /etc/ldap/ldap.conf com o arquivo fornecido no projeto (\$APPMAN_HOME/config/ldap.conf), que já contém a modificação abaixo.

```
$ sudo vi /etc/ldap/ldap.conf
```

Adicionar a linha:

```
BASE dc=exehda
```

Adicionar as entidades de inicialização do EXEHDA (\$APPMAN_HOME/config/init.ldif) ao LDAP. Para isso será necessário parar o LDAP e, após, inicializá-lo.

```
$ sudo /etc/init.d/slapd stop
```

Se foi realizada uma nova instalação, deve ser removido o conteúdo gerado ao instalar.

```
$ sudo rm -rf /var/lib/ldap/*
```

Insere o novo conteúdo.

```
$ sudo slapadd -l $APPMAN_HOME/config/init.ldif
```

Corrige as permissões no banco de dados.

```
$ sudo chown -R openldap.openldap /var/lib/ldap
```

Inicializa o daemon do LDAP

```
$ sudo /etc/init.d/slapd start
```

A configuração do LDAP pode ser verificada com a execução do seguinte comando.

```
$ ldapsearch -xLLL -b "dc=exehda"
```

A saída deve ser próxima do que segue abaixo.

```
----- Saída gerada - início
```

```
dn: dc=exehda
objectClass: dcObject
objectClass: organizationalUnit
dc: exehda
ou: Configuracao para EXEHDA
```

```
dn: cn=admin,dc=exehda
objectClass: simpleSecurityObject
objectClass: organizationalRole
cn: admin
description: LDAP administrator
----- Saída gerada - fim
```

Instalar e configurar o servidor NFS.

```
$ sudo apt-get install nfs-common nfs-kernel-server
```

Incluir a seguinte linha no arquivo /etc/exports do nó base, para permitir a troca de arquivos entre os nós do AppMan. Note que aqui não estamos usando a variável de ambiente \$EXEHDA_HOME, pois este arquivo será lido pelo usuário do NFS, que não possui a variável definida.

```
/home/aluno/AppMan/appman-project/exehda/log *(rw, sync, subtree_check)
```

Reiniciar os serviços do NFS

```
$ sudo /etc/init.d/nfs-kernel-server restart
```

```
$ sudo /etc/init.d/nfs-common restart
```

Nos demais nós deve ser instalado o cliente NFS e deve ser montado o diretório de log, onde 10.30.5.145 é o endereço do nó base.

```
$ sudo apt-get install nfs-common
```

```
$ mkdir -p $EXEHDA_HOME/log
```

```
$ sudo mount 10.30.5.145:$EXEHDA_HOME/log $EXEHDA_HOME/log
```

Instalar a máquina virtual Java da Sun. A versão a ser instalada é 1.5, pois existe um bug relativo ao classloading do ISAM ao executar na versão 1.6. O download da máquina virtual pode ser realizado diretamente do site da Sun: <http://java.sun.com/javase/downloads/5/jdk>

Se não for realizado automaticamente, a variável JAVA_HOME deve ser configurada para o diretório onde foi instalado a JDK.

```
$ export JAVA_HOME=/opt/jdk1.5.0_19/
```

Para executar o ambiente EXEHDA deve ser configurado o arquivo \$EXEHDA_HOME/bin/exehda-services.xml. Neste arquivo são definidos os perfis

de execução de cada nó, bem como id e nome do agrupamento(célula). Este arquivo está pré-configurado para usar o perfil "base" como o da máquina servidor do ambiente. Deve ser configurado o servidor de LDAP, o diretório onde ficarão disponíveis os arquivos de inicialização via protocolo BDA, e o serviço de discovery, que deve apontar para a máquina servidor do ambiente. Nos nós cliente devem ser configurados os serviços CIB e BDA para apontar para o servidor do ambiente.

O EXEHDA deve ser inicializado a partir do diretório de log, pois este será usado para a troca de arquivos entre os nós.

```
$ cd $EXEHDA_HOME/log
```

Concede permissão de execução ao arquivo exehda.

```
$ chmod +x $EXEHDA_HOME/bin/exehda
```

Existem configurações como o tamanho máximo da memória da VM, que podem ser realizadas no arquivo exehda.env.

```
$ vi $EXEHDA_HOME/bin/exehda.env
```

Executa o EXEHDA.

```
$ $EXEHDA_HOME/bin/exehda --profile base
```

Para inicializar o EXEHDA nos nós basta atribuir um perfil ao nó desejado.

```
$ cd $EXEHDA_HOME/log
```

```
$ $EXEHDA_HOME/bin/exehda --profile nohl
```

Deve ser especificado no arquivo \$APPMAN_PROJECT/src/main/resources/gridnodes.properties quais perfis do EXEHDA serão tratados como SMS e após, deve ser compilado e disponibilizado o appman-project. Para a compilação e disponibilização será necessário instalar o maven a partir do endereço <http://maven.apache.org/download.html>.

Após o download, descompacte e inclua o diretório apache-maven-X/bin no PATH.

```
$ export PATH=/opt/apache-maven-2.1.0/bin:$PATH
```

Execute as instruções do arquivo \$APPMAN_PROJECT/readme-maven.txt.

Compilando e disponibilizando o AppMan.

```
$ mvn -f $APPMAN_PROJECT/pom.xml install
```

Uma aplicação (DAG) pode ser inicializada na grade via script usando o executável run-appman.sh, como mostra abaixo.

Primeiro, conceda permissão de execução aos arquivos necessários.

```
$ chmod +x $EXEHDA_HOME/appman-bin/run-appman.sh $EXEHDA_HOME/bin/isam-run
```

Executa aplicação t10.dag.

```
$ $EXEHDA_HOME/appman-bin/run-appman.sh $EXEHDA_HOME/appman-bin/dag/t10.dag
```

APÊNDICE B Instalação do portal uPortal

O uPortal possui dependências com os seguintes projetos da Apache: Tomcat, Ant e Maven. No entanto, estas dependências estão inclusas no download da versão *Quick Start* do projeto, que será usada aqui. Também está disponível a versão *Developers Quick Start*, onde a única diferença é a inclusão das pastas de controle de versão (.svn) no pacote.

Baixe e descompacte o uPortal 3.1.x a partir do endereço abaixo.
<http://www.jasig.org/uportal/download>
 Para descompactar arquivos da extensão .tar.gz pode ser usado o comando abaixo.
`$ gunzip -c <arquivo> | tar -xvf -`

Para a instalação a variável de ambiente JAVA_HOME deve estar configurada para um caminho sem espaços.

A inicialização é realizada a partir do diretório em que foi descompactado o projeto.
`$ cd /home/aluno/uPortal-3.1.1-quick-start`

Inicializa a base de dados HSQL e o servidor Tomcat em plano de fundo.
`$./ant.sh start`

O projeto uPortal utiliza uma forma de autenticação única chamada Central Authentication Service (CAS), que exige o conhecimento do endereço externo de acesso ao portal. Este endereço aponta inicialmente para localhost:8080 e para sua modificação devem ser alteradas todas as ocorrências deste nos arquivos web.xml e security.properties do contexto uPortal.

```
$ vi /home/aluno/uPortal-3.1.1-quick-start/apache-tomcat-6.0.18/webapps/uPortal/WEB-INF/web.xml
$ vi /home/aluno/uPortal-3.1.1-quick-start/apache-tomcat-6.0.18/webapps/uPortal/WEB-INF/classes/properties/security.properties
```

APÊNDICE C Instalação do portal GridSphere

Para a instalação do GridSphere será necessária a instalação dos pré-requisitos Apache Tomcat 5.5.x e Apache Ant 1.5+.

Baixe e descompacte o GridSphere 3.1 a partir do endereço abaixo.
<http://www.gridisphere.org/gridsphere/gridsphere/guest/download>
 Para descompactar arquivos da extensão .tar.gz pode ser usado o comando abaixo.

```
$ gunzip -c <arquivo> | tar -xvf -
```

Baixe e descompacte o Apache Tomcat 5.5.x (Core) a partir do endereço abaixo.
<http://tomcat.apache.org/download-55.cgi>

Baixe e descompacte o Apache Ant 1.5+ a partir do endereço abaixo.
<http://ant.apache.org/bindownload.cgi>

Configure variáveis de ambiente para a localização dos projetos.

```
$ export ANT_HOME=/opt/apache-ant-1.7.1
$ export CATALINA_HOME=/opt/apache-tomcat-5.5.27
$ export PATH=$ANT_HOME/bin:$PATH
```

Por conta da correção de um bug na versão 5.5.27 do Tomcat, é necessário corrigir os fontes do GridSphere ou configurar a seguinte variável de ambiente.

```
$ export CATALINA_OPTS="-Dorg.apache.jasper.compiler.Parser.STRICT_QUOTE_ESCAPING=false"
```

Por conta do mesmo bug relatado acima, deve ser configurada a propriedade gridsphere.useprecompiledjsp para "false" no arquivo /home/aluno/gridsphere-3.1/build.properties.

```
$ vi /home/aluno/gridsphere-3.1/build.properties
gridsphere.useprecompiledjsp=false
```

Vá para o diretório onde o GridSphere foi descompactado.

```
$ cd /home/aluno/gridsphere-3.1
```

 Execute o comando de instalação.

```
$ ant install
```

Inicialize o servidor Apache Tomcat.

```
$ $CATALINA_HOME/bin/catalina.sh run
```

APÊNDICE D Instalação das portlets do AppMan

Para manter a compatibilidade com diferentes containers, foram seguidos os padrões da JSR-168 e, assim não foi modificado o arquivo web.xml, conforme o container do GridSphere exige. Para isso, está sendo suprida uma versão deste arquivo com extensão “.gridsphere”, para ser usado neste container. No caso do uPortal as modificações são realizadas no web.xml em tempo de deploy.

Para o deploy das portlets será necessário compilar o portal. Os fontes estão disponíveis no googlecode e podem ser baixados através do comando abaixo:

```
$ svn checkout http://tcc-michel.googlecode.com/svn/trunk/AppMan/ tcc-
michel-read-only
```

Para facilitar a instalação, é sugerida a configuração das seguintes variáveis de ambiente.

```
$ export APPMAN_HOME=/home/aluno/AppMan
$ export APPMAN_PORTLETS=$APPMAN_HOME/appman-portlets
```

Copie o arquivo env.properties para o diretório ~/.appman-portlets.

```
$ mkdir ~/.appman-portlets
$ cp $APPMAN_PORTLETS/config/user.home/.appman-portlets/env.properties
~/.appman-portlets
```

Configure as variáveis exehda.home, appman.portlets.job.dir, exehda.ldap.* e db.* do arquivo env.properties de acordo com o ambiente local.

Rode os scripts de instalação da base de dados (\$APPMAN_HOME/config/db-init.sql). Para isso, compile as classes do projeto.

```
$ mvn -f $APPMAN_PORTLETS/pom.xml compile
Instale a partir do script, passando URL JDBC, usuário e senha.
$ $APPMAN_HOME/config/db-install.sh "jdbc:hsqldb:hsqldb://localhost:8887"
"sa" ""
```

Para instalar no container GridSphere 3.1, siga os passos abaixo.

Configure a variável de ambiente do servidor Apache Tomcat.

```
$ export CATALINA_HOME=/opt/apache-tomcat-5.5.27
```

Crie um arquivo chamado appman-portlets em ~/.gridsphere/portlets.

```
$ touch ~/.gridsphere/portlets/appman-portlets
```

Execute o maven para o arquivo pom-gridsphere.xml, empacotar e configurar o ambiente para a execução.

```
$ mvn -f $APPMAN_PORTLETS/pom-gridsphere.xml pre-integration-test
```

Inicialize o servidor web e acesse http://host:port/gridsphere. Configure a portlet usando usuário e senha admin, através do menu "Layout".

Para instalar no uPortal, siga os passos abaixo.

Desconfigure a variável CATALINA_HOME, para não confundir o uPortal, fazendo com que o servidor inicializado seja o configurado para o GridSphere.

```
$ unset CATALINA_HOME
```

Configure a variável de ambiente do uPortal.

```
$ export UPORTAL_HOME=/home/aluno/uPortal-3.1.1-quick-start
```

Empacote o projeto appman-portlets.
\$ mvn -f \$APPMAN_PORTLETS/pom.xml package

Execute o deploy do uPortal.
\$ cd \$UPORTAL_HOME/uPortal-3.1.1
\$../ant.sh deployPortletApp -DportletApp=\$APPMAN_PORTLETS/target/appman-portlets.war

Inicialize o servidor web e de BD do uPortal.
\$ cd \$UPORTAL_HOME
\$../ant.sh start

Acesse <http://localhost:8080/uPortal> usando usuário e senha admin e configure a portlet no "Portlet Manager". Na configuração de "Portlet Descriptor", usar "/appman-portlets" para "Portlet Web Application Path" e, dependendo da portlet a ser configurada, usar "appman-jobsubmission" ou "appman-filebrowser" em "Portlet Name". Após, use a opção "Add Content" para adicionar a portlet configurada.

APÊNDICE E Aplicação submetida à grade

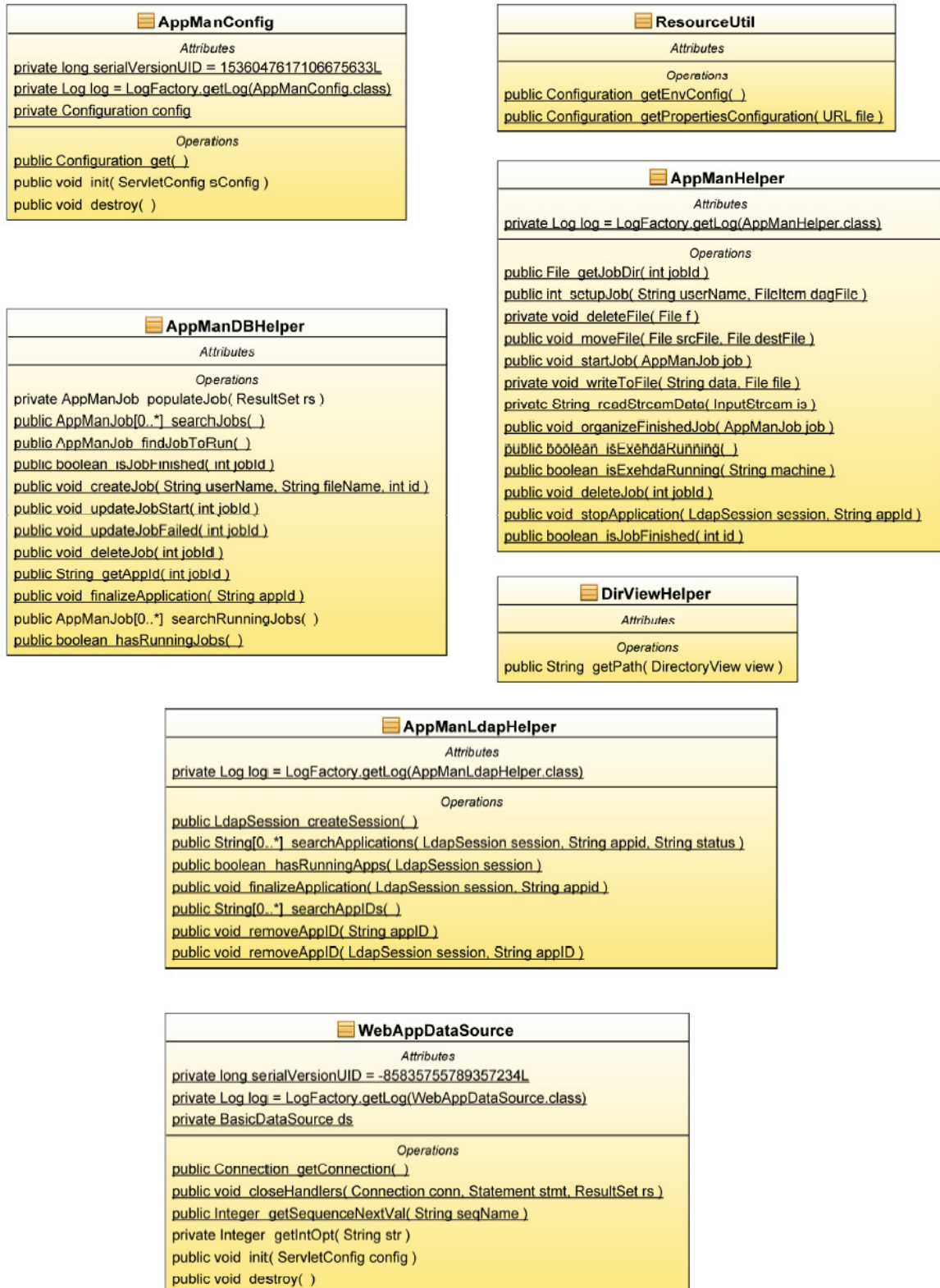
O arquivo “/home/aluno/file.in” é um arquivo em branco, que serve apenas para auxiliar no mapeamento de dependências realizado pela grade.

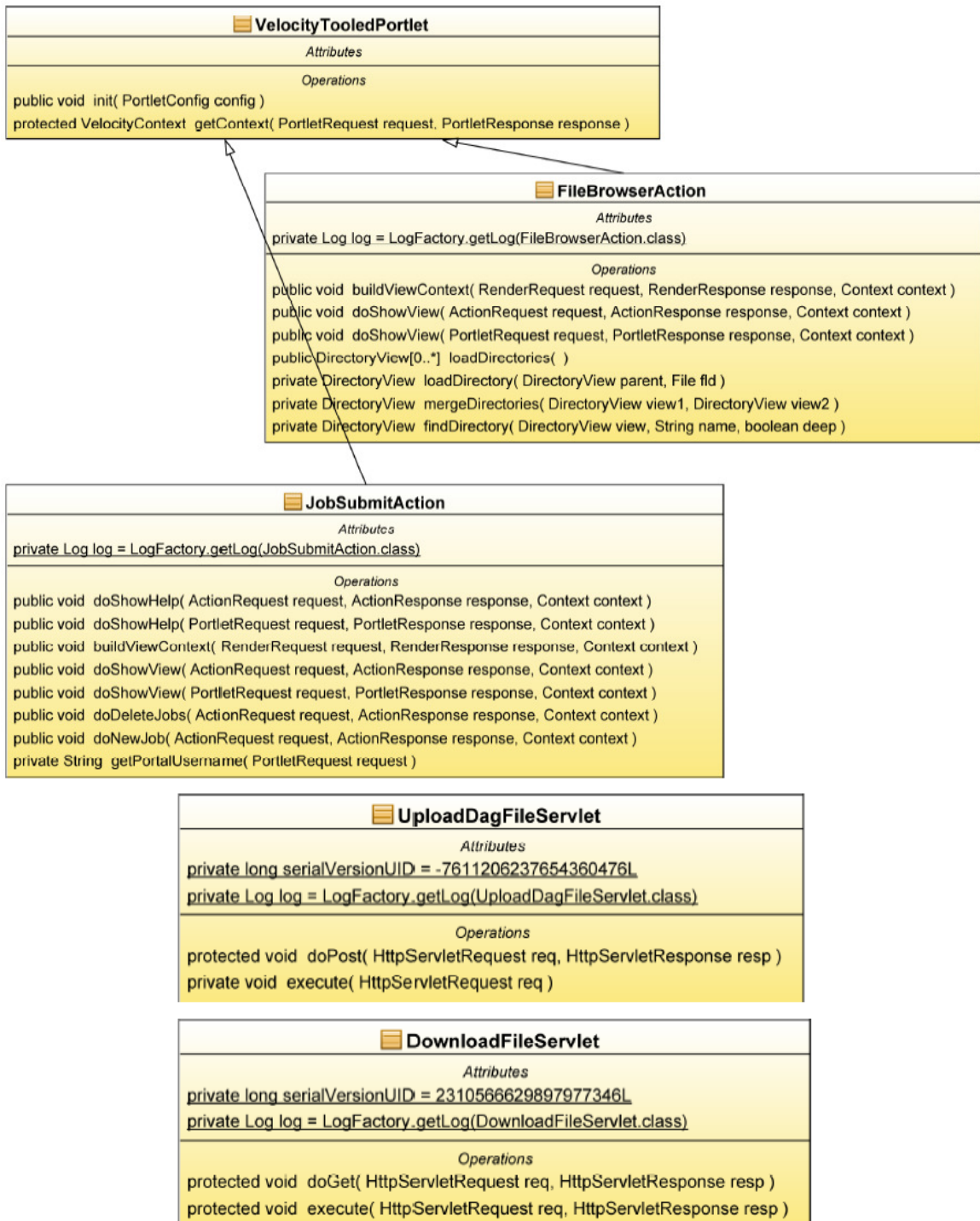
```
graph independent
task 0 -e "cat file.in > t.out.0" -i "/home/aluno/file.in" -o t.out.0
task 1 -e "cat file.in > t.out.1" -i "/home/aluno/file.in" -o t.out.1
task 2 -e "cat file.in > t.out.2" -i "/home/aluno/file.in" -o t.out.2
task 3 -e "cat file.in > t.out.3" -i "/home/aluno/file.in" -o t.out.3
task 4 -e "cat file.in > t.out.4" -i "/home/aluno/file.in" -o t.out.4
task 5 -e "cat file.in > t.out.5" -i "/home/aluno/file.in" -o t.out.5
task 6 -e "cat file.in > t.out.6" -i "/home/aluno/file.in" -o t.out.6
task 7 -e "cat file.in > t.out.7" -i "/home/aluno/file.in" -o t.out.7
task 8 -e "cat file.in > t.out.8" -i "/home/aluno/file.in" -o t.out.8
task 9 -e "cat file.in > t.out.9" -i "/home/aluno/file.in" -o t.out.9
```

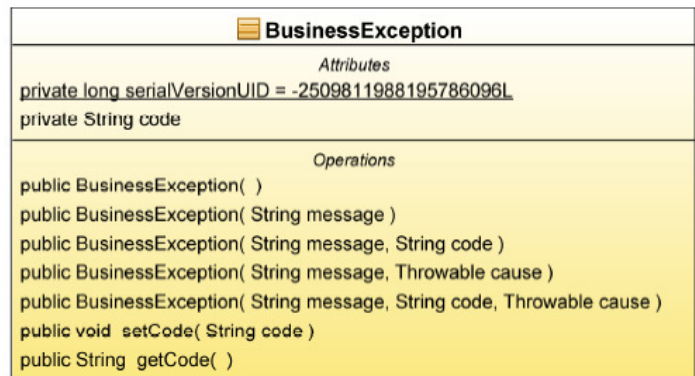
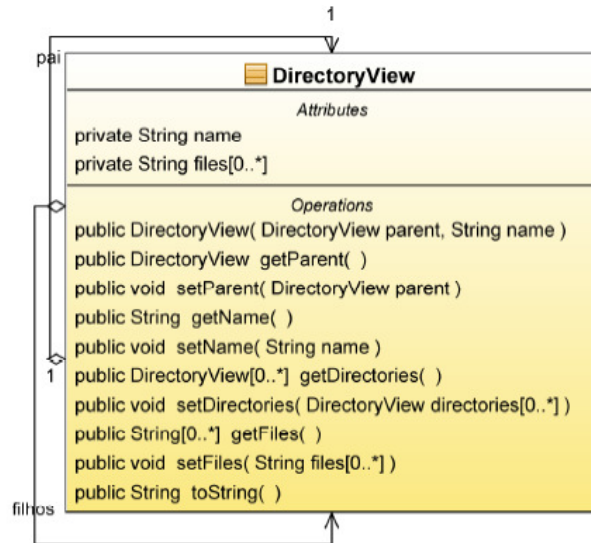
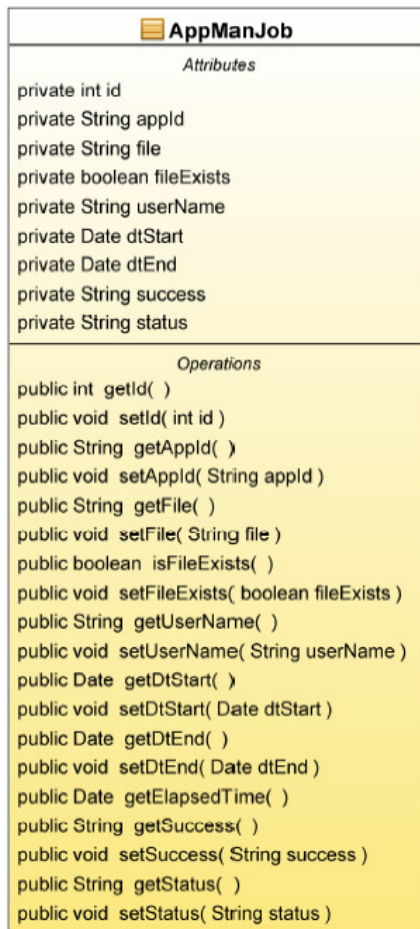
APÊNDICE F Diagrama de classes do Portal do AppMan

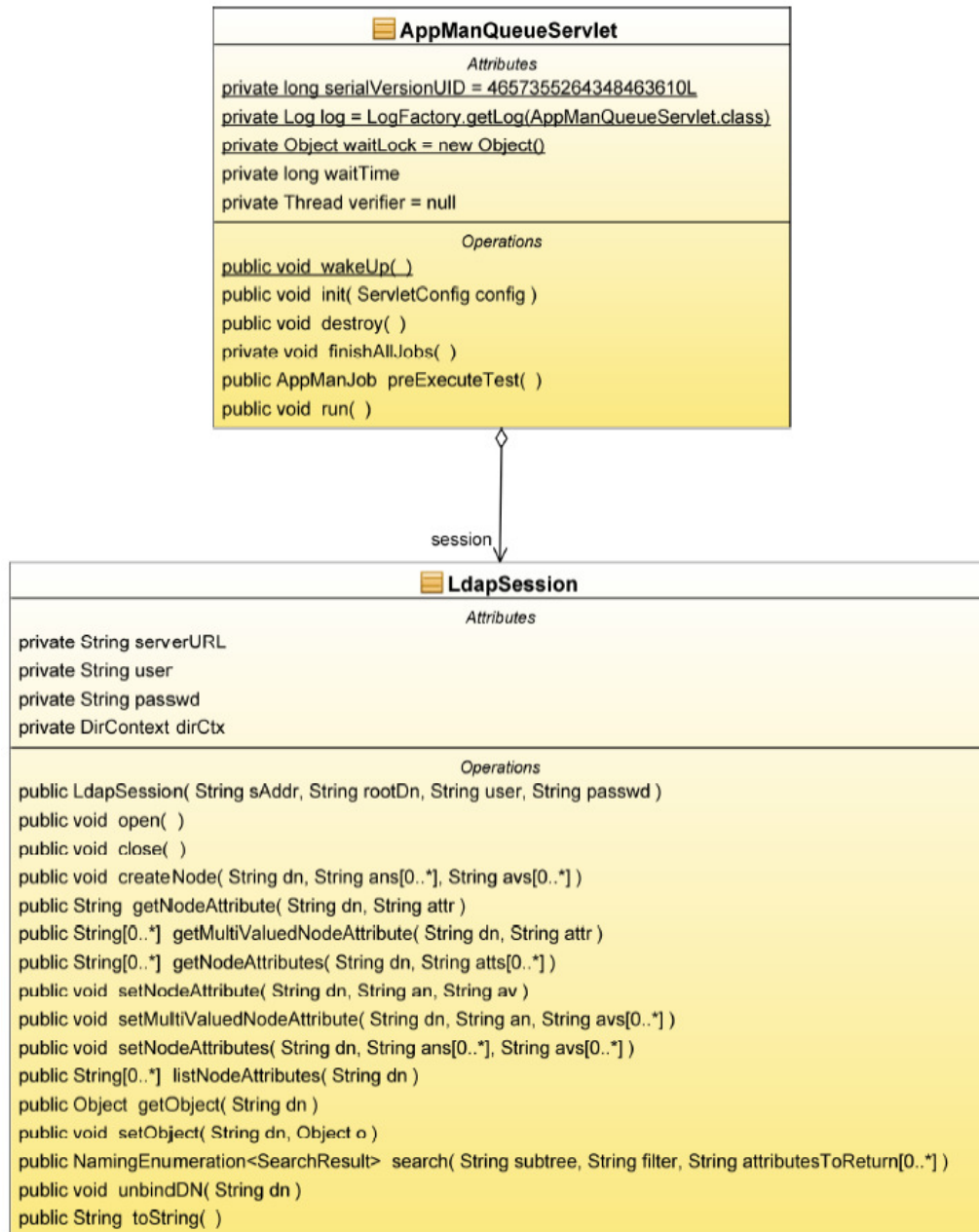
Todas as classes descritas neste apêndice foram desenvolvidas para o portal do AppMan, e encontram-se disponíveis no repositório de versões abaixo.

<http://tcc-michel.googlecode.com/svn/trunk/>









APÊNDICE G Diagrama ER da Portlet do AppMan**APPMAN_JOB**

JOB_ID: INTEGER NOT NULL
USERNAME: VARCHAR2(255) NOT NULL
FILE: VARCHAR2(255) NOT NULL
DTSTART: DATE NULL
DTEND: DATE NULL
SUCCESS: VARCHAR2(1) NULL
EXEHDA_APP_ID: VARCHAR2(255) NULL
DELETED: CHAR NOT NULL

APÊNDICE H Duração da execução das aplicações via scripts

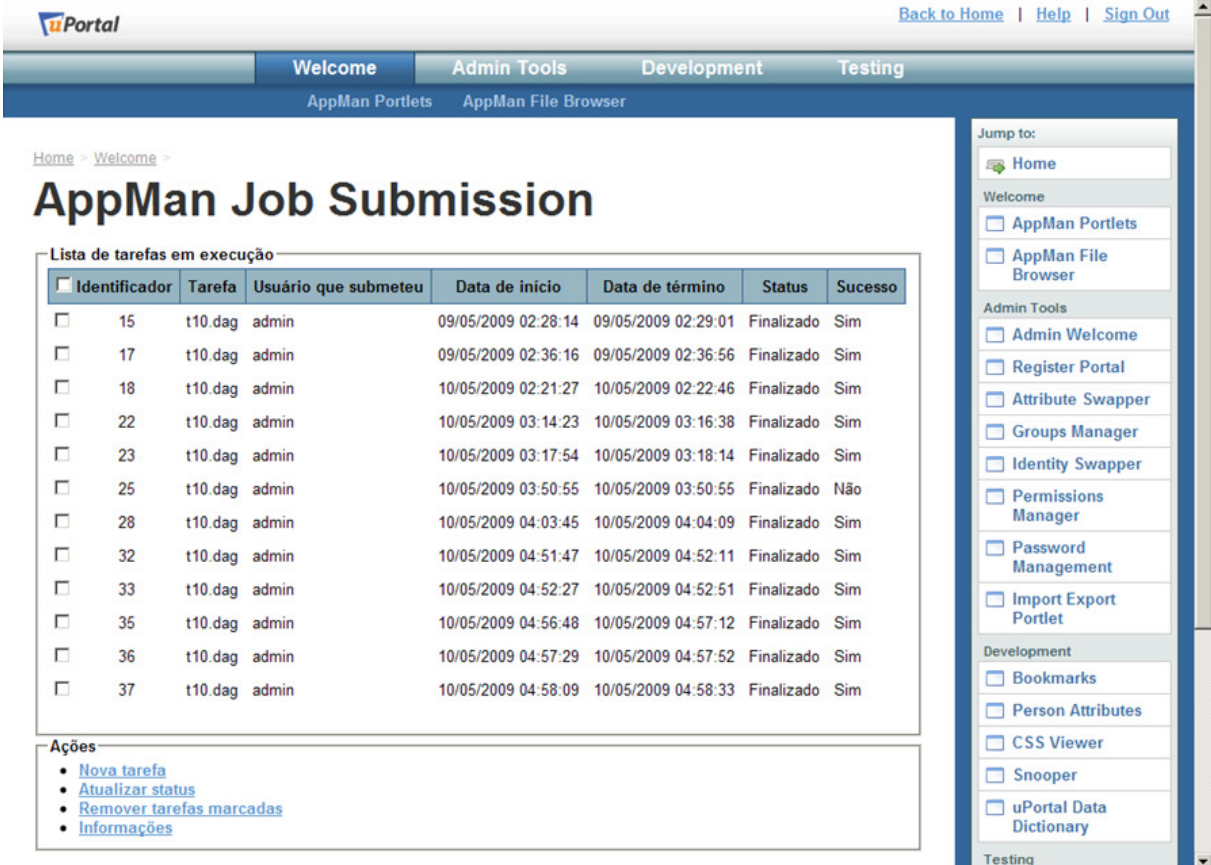
Execução	2 nós	4 nós	6 nós
1	66183	51292	41248
2	65493	50726	40590
3	65385	50490	40583
4	65773	50664	40638
5	65621	50651	40524
6	65355	50517	40471
7	65561	50527	40578
8	65705	50524	40669
9	65508	50861	40727
10	65590	50548	40592
11	65572	50768	40802
12	65609	50545	40505
13	65722	50559	40690
14	65483	50455	40545
15	65470	50669	40526
16	65799	50471	40568
17	65465	50484	40678
18	65593	50682	40723
19	65628	50576	40521
20	65697	50652	40461
21	65509	50750	40740
22	65704	50666	40731
23	65517	50918	40589
24	65654	50588	40602
25	65584	50488	40812
26	65583	50526	40819
27	65667	51026	40796
28	65832	50664	40587
29	65686	50726	40733
30	65615	50696	40602
31	65789	50846	40825

APÊNDICE I Duração da execução das aplicações via portal

Execução	2 nós	4 nós	6 nós
1	71629	56850	46652
2	70743	55820	46225
3	70713	56196	45588
4	70888	55828	45762
5	70857	55785	45648
6	70578	55481	45688
7	70617	55667	45664
8	70620	55978	45629
9	70883	56037	45600
10	70649	55747	45733
11	70430	55660	45875
12	70564	55860	45920
13	70760	55727	45739
14	70782	55926	45830
15	70671	55688	46001
16	70754	55976	45704
17	70764	55776	45901
18	70795	55719	46056
19	70783	55924	45804
20	71052	55854	45794
21	70928	55847	45774
22	70709	55971	45731
23	70734	55962	46105
24	71121	56315	46058
25	71124	55707	45705
26	70680	55779	45687
27	70682	55809	46218
28	70842	56087	45682
29	71204	56094	45789
30	70782	55733	45892
31	70909	55810	46072

APÊNDICE J *Portlets* em execução nos portais uPortal e GridSphere

Portlets em execução no portal uPortal.



uPortal [Back to Home](#) | [Help](#) | [Sign Out](#)

Welcome Admin Tools Development Testing

AppMan Portlets AppMan File Browser

Home > Welcome >

AppMan Job Submission

Lista de tarefas em execução

<input type="checkbox"/>	Identificador	Tarefa	Usuário que submeteu	Data de início	Data de término	Status	Sucesso
<input type="checkbox"/>	15	t10.dag	admin	09/05/2009 02:28:14	09/05/2009 02:29:01	Finalizado	Sim
<input type="checkbox"/>	17	t10.dag	admin	09/05/2009 02:36:16	09/05/2009 02:36:56	Finalizado	Sim
<input type="checkbox"/>	18	t10.dag	admin	10/05/2009 02:21:27	10/05/2009 02:22:46	Finalizado	Sim
<input type="checkbox"/>	22	t10.dag	admin	10/05/2009 03:14:23	10/05/2009 03:16:38	Finalizado	Sim
<input type="checkbox"/>	23	t10.dag	admin	10/05/2009 03:17:54	10/05/2009 03:18:14	Finalizado	Sim
<input type="checkbox"/>	25	t10.dag	admin	10/05/2009 03:50:55	10/05/2009 03:50:55	Finalizado	Não
<input type="checkbox"/>	28	t10.dag	admin	10/05/2009 04:03:45	10/05/2009 04:04:09	Finalizado	Sim
<input type="checkbox"/>	32	t10.dag	admin	10/05/2009 04:51:47	10/05/2009 04:52:11	Finalizado	Sim
<input type="checkbox"/>	33	t10.dag	admin	10/05/2009 04:52:27	10/05/2009 04:52:51	Finalizado	Sim
<input type="checkbox"/>	35	t10.dag	admin	10/05/2009 04:56:48	10/05/2009 04:57:12	Finalizado	Sim
<input type="checkbox"/>	36	t10.dag	admin	10/05/2009 04:57:29	10/05/2009 04:57:52	Finalizado	Sim
<input type="checkbox"/>	37	t10.dag	admin	10/05/2009 04:58:09	10/05/2009 04:58:33	Finalizado	Sim

Ações

- [Nova tarefa](#)
- [Atualizar status](#)
- [Remover tarefas marcadas](#)
- [Informações](#)

Jump to: Home

Welcome

- ☐ AppMan Portlets
- ☐ AppMan File Browser

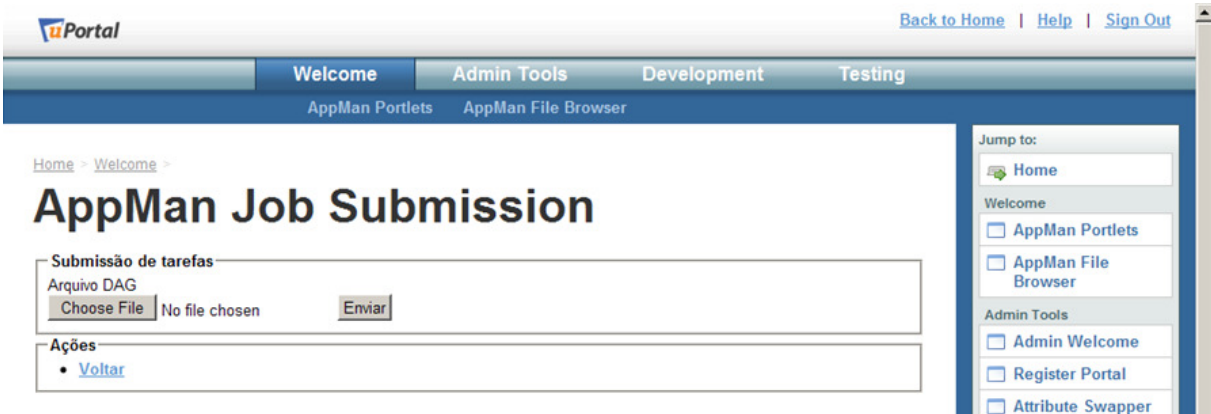
Admin Tools

- ☐ Admin Welcome
- ☐ Register Portal
- ☐ Attribute Swapper
- ☐ Groups Manager
- ☐ Identity Swapper
- ☐ Permissions Manager
- ☐ Password Management
- ☐ Import Export Portlet

Development

- ☐ Bookmarks
- ☐ Person Attributes
- ☐ CSS Viewer
- ☐ Snooper
- ☐ uPortal Data Dictionary

Testing



uPortal [Back to Home](#) | [Help](#) | [Sign Out](#)

Welcome Admin Tools Development Testing

AppMan Portlets AppMan File Browser

Home > Welcome >

AppMan Job Submission

Submissão de tarefas

Arquivo DAG

No file chosen

Ações

- [Voltar](#)

Jump to: Home

Welcome

- ☐ AppMan Portlets
- ☐ AppMan File Browser

Admin Tools

- ☐ Admin Welcome
- ☐ Register Portal
- ☐ Attribute Swapper

[Back to Home](#) | [Help](#) | [Sign Out](#)

WelcomeAdmin ToolsDevelopmentTesting
AppMan PortletsAppMan File Browser

[Home](#) > [Welcome](#) >

AppMan Job Submission

Informações

Esta portlet é parte integrante do trabalho de conclusão de curso de Michel David da Costa

Configurações em uso (arquivo `~/appman-portlets/env.properties`):

appman.portlets.work.dir	/home/michel/.appman-portlets
exehda.home	/home/michel/share/appman-project/exehda
appman.bin	/home/michel/share/appman-project/exehda/appman-bin
exehda.isam.run	/home/michel/share/appman-project/exehda/bin/isam-run
exehda.log.dir	/home/michel/share/appman-project/exehda/log
appman.isam.console	/home/michel/share/appman-project/exehda/appman-bin/appman-console.isam
appman.portlets.job.dir	/home/michel/.appman-portlets/job
exehda.gatekeeper	localhost:29901
appman.home	/home/michel/share/appman

Ações

- [Voltar](#)

Jump to:

- [Home](#)

Welcome

- ☐ AppMan Portlets
- ☐ AppMan File Browser

Admin Tools

- ☐ Admin Welcome
- ☐ Register Portal
- ☐ Attribute Swapper
- ☐ Groups Manager
- ☐ Identity Swapper
- ☐ Permissions Manager
- ☐ Password Management
- ☐ Import Export Portlet

Development

- ☐ Bookmarks
- ☐ Person Attributes
- ☐ CSS Viewer
- ☐ Snooper
- ☐ uPortal Data Dictionary

Testing

[Back to Home](#) | [Help](#) | [Sign Out](#)

WelcomeAdmin ToolsDevelopmentTesting
AppMan PortletsAppMan File Browser

[Home](#) > [Welcome](#) >

AppMan File Browser

Tarefa 37

- appman-sandbox
 - 32533053706
 - graph.dag
 - results
 - 5141666055
- appman.log
- appman_contact_adress.txt
- clusters.xml
- edges.xml
- graphviz.dot
- hosts.txt
- manifest.xml
- parseOut.txt
- std.err
- std.out
- t10.dag
- tasks-execution-appman.trace
- tasks.xml
- tempoExecucao.txt

Ações

- [Voltar](#)
- [Atualizar lista](#)

Jump to:

- [Home](#)

Welcome

- ☐ AppMan Portlets
- ☐ AppMan File Browser

Admin Tools

- ☐ Admin Welcome
- ☐ Register Portal
- ☐ Attribute Swapper
- ☐ Groups Manager
- ☐ Identity Swapper
- ☐ Permissions Manager
- ☐ Password Management
- ☐ Import Export Portlet

Development

- ☐ Bookmarks
- ☐ Person Attributes
- ☐ CSS Viewer
- ☐ Snooper
- ☐ uPortal Data Dictionary

Testing

Portlets em execução no portal GridSphere.

gridsphere portal framework

Welcome , admin admin [Administration](#) [Content](#) [Layout](#) [Profile](#) [Home](#) [Logout](#)

Home

AppMan Job Submission

Lista de tarefas em execução

<input type="checkbox"/> Identificador	Tarefa	Usuário que submeteu	Data de início	Data de término	Duração	Status	Sucesso	
<input type="checkbox"/>	0	t10.dag	admin	07/06/2009 08:45:07	07/06/2009 08:45:07	00:00.000	Finalizado	Não
<input type="checkbox"/>	1	t10.dag	admin	07/06/2009 09:01:40	07/06/2009 09:01:57	00:17.156	Finalizado	Sim
<input type="checkbox"/>	2	t10.dag	admin	07/06/2009 09:13:46	07/06/2009 09:14:04	00:17.619	Finalizado	Sim
<input type="checkbox"/>	3	t10.dag	admin	07/06/2009 09:14:53	07/06/2009 09:15:10	00:17.217	Finalizado	Sim

Ações

- Nova tarefa
- Atualizar status
- Remover tarefas marcadas
- Informações

June 15, 2009

powered by gridsphere

gridsphere portal framework

Welcome , admin admin [Administration](#) [Content](#) [Layout](#) [Profile](#) [Home](#) [Logout](#)

Home

AppMan Job Submission

Submissão de tarefas

Arquivo DAG

No file chosen

Ações

- Voltar

June 15, 2009

powered by gridsphere

gridsphere portal framework

Welcome , admin admin [Administration](#) [Content](#) [Layout](#) [Profile](#) [Home](#) [Logout](#)

Home

AppMan Job Submission

Informações

Esta portlet é parte integrante do [trabalho de conclusão de curso de Michel David da Costa](#)
 Configurações em uso (arquivo `~/appman-portlets/env.properties`):

appman.portlets.work.dir	/home/aluno/.appman-portlets
exehda.home	/home/aluno/AppMan/appman-project/exehda
appman.bin	/home/aluno/AppMan/appman-project/exehda/appman-bin
exehda.isam.run	/home/aluno/AppMan/appman-project/exehda/bin/isam-run
exehda.log.dir	/home/aluno/AppMan/appman-project/exehda/log
appman.isam.console	/home/aluno/AppMan/appman-project/exehda/appman-bin/appman-console.isam
appman.portlets.job.dir	/home/aluno/.appman-portlets/job
exehda.ldap.server	localhost
exehda.ldap.rootDN	dc=exehda
exehda.ldap.user	cn=admin,dc=exehda
exehda.ldap.password	****
db.driverClassName	org.hsqldb.jdbcDriver
db.url	jdbc:hsqldb:hsqldb://localhost:8887
db.username	sa
db.password	****
db.minIdle	0
db.maxIdle	2
db.maxActive	16

gridsphere portal framework

Welcome , admin admin [Administration](#) [Content](#) [Layout](#) [Profile](#) [Home](#) [Logout](#)

Home

AppMan File Browser

Tarefa 3

- [-] appman-sandbox
 - [-] 14642272375
 - graph.dag
 - [-] results
 - 27251104544
- appman.log
- appman_contact_adress.txt
- clusters.xml
- edges.xml
- graphviz.dot
- hosts.txt
- manifest.xml
- parseOut.txt
- std.err
- t10.dag
- tasks-execution-appman.trace
- tasks.xml
- tempoExecucao.txt

Ações

- [Voltar](#)

Apêndice K Plano de migração do AppMan para o padrão OGSA

Com a análise realizada no funcionamento do AppMan e na arquitetura OGSA foi possível construir um plano de migração para esta arquitetura. Segue abaixo uma lista de passos que devem ser cumpridos para adaptar à nova arquitetura.

1. Determinar um servidor HTTP para web services, pois são usados web services para comunicação.
2. Substituir serviço de *discovery* realizado pelo ambiente de execução pela implementação do web service de serviços de informação de acordo com a especificação WS-RF do OGSA.
3. Implementar serviço de segurança para transferência dos demais serviços de acordo com a especificação WS-Security.
4. Abstrair serviço de execução remota, substituindo-o pelo serviço de recursos, usando web services compatíveis com o perfil básico (*OGSA WSRF Basic Profile*).
5. Abstrair gerenciamento de execução do AppMan e assim sua execução hierárquica, mantendo-o como um web service EMS, também implementado através o perfil básico.
6. Substituir serviços de seleção de recursos, implementados pelo ambiente de execução e algoritmo de *round-robin* do AppMan, por um web service que irá realizar consultas ao serviço de informação para determinar os recursos disponíveis.
7. Substituir manipulação dos dados no AppMan por um web service que irá manter os dados em um repositório configurável, abstraindo assim a forma de acesso e serviço vinculado ao acesso dos dados.