João Pedro Rodrigues Dias Granja

Sistema de congiração de ambientes de deployment de Aplicações Web

UMinho | 2009

Universidade do Minho
Escola de Engenharia

João Pedro Rodrigues Dias Granja

Sistema de conguração de ambientes de deployment de Aplicações Web

Outubro de 2009

Universidade do Minho
Escola de Engenharia

João Pedro Rodrigues Dias Granja

Sistema de conguração de ambientes de deployment de Aplicações Web

Mestrado em Engenharia Informática

Trabalho efectuado sob a orientação do
Professor Doutor António Nestor Ribeiro

Outubro de 2009

Resumo

Cada vez mais, a Internet assume-se como uma rede global de convergência de aplicações e serviços das mais diversas áreas. Com a transição das aplicações para a Internet a qualidade destas já não está apenas dependente da construção do software, estando intimamente ligada à qualidade da infra-estrutura onde as aplicações vão assentar. Devido à natureza dos sistemas Web é essencial ter uma infra-estrutura suficientemente flexível para se poder adaptar a uma maior carga, ou a uma mudança nas políticas de negócio. Outro factor que pode condicionar o correcto funcionamento das aplicações é o tipo de instalação e configuração a que vão ser sujeitas. De facto, a instalação e configuração manual de aplicações é uma tarefa cada vez mais inexequível, dada a sua morosidade e dificuldade inerente à sua eventual complexidade, decorrente da crescente complexidade das soluções tecnológicas. Estes problemas são acrescidos em ambientes muito dinâmicos, como é o caso dos sistemas Web, necessitando, na maior parte das situações, um elevado conhecimento técnico. Esta necessidade de conhecimento técnico advém, sobretudo, do excesso de meta-informação presente, quer na configuração das infra-estruturas Web, quer na configuração das próprias aplicações Web.

Neste trabalho estudou-se a problemática em torno, por um lado, da construção e gestão de infra-estruturas Web e, por outro, da gestão e *deployment* de aplicações Web JEE. Desenvolveu-se uma ferramenta que fosse capaz, em determinados casos, de automatizar e promover, de forma facilitada, a construção e alteração de uma infra-estrutura, bem como auxiliar o utilizador em todo o processo de deployment de uma aplicação Web JEE. Esta foi sujeita a cenários de testes, de forma a validar a sua capacidade em auxiliar o utilizador nas tarefas acima mencionadas, sendo que os resultados obtidos foram bastante animadores. Além disso, a ferramenta permite, em muitos casos, reduzir o tempo gasto na execução dessas tarefas.

Palavras-Chave: JEE, Deployment, Disponibilidade, Aplicações Web, Escalabilidade, Infra-estruturas

Abstract

Internet is taking off increasingly as a global network of convergence of applications and services of different areas. With the transition of applications to the Web the quality of those does not depend exclusively on the software development. It's closely linked to the quality of the infrastructure where applications will be deployed. Owing to the nature of the Web systems it is essential to have a flexible infrastructure in order to adapt it to a higher load or to a change in business policies. Another factor that may influence the proper functioning of the applications is the installation and configuration type they are going to be submitted. The manual installation and configuration of the applications is more and more an unfeasible task, because of its slowness and difficulty inborn to its complexity. This is duo to the growing complexity of the technological solutions. These problems become more serious in dynamic environments, as it is the case of Web systems, so a high technician knowledge is required. This need of technical knowledge stems mainly because of the existence of too much meta-information in the configuration of the Web infrastructures as well as in the configuration of the Web applications.

In this work the problem of the construction and management of an infrastructure, on one hand, and the management and deployment of JEE Web applications , on the other hand, were studied. A tool was developed in order to automate and promote the construction and changing of infrastructures, as well as assisting the user in the whole deployment process of the JEE Web application. This tool was tested to evaluate its capacity to help the user in the tasks mentioned above, and the results were very encouraging. In addition, the tool allows, in many cases, to reduce the time spent in carrying out those tasks.

Keywords: JEE, Deployment, Availability, Web Applications, Scalability, Infrastructures

Agradecimentos

A realização deste trabalho não seria possível sem a colaboração e compreensão de várias pessoas que me incentivaram e apoiaram ao longo da sua execução. Sem querer omitir ou ser injusto, gostaria de reconhecer particularmente algumas dessas pessoas.

Começo por agradecer ao meu orientador, Prof. António Nestor Ribeiro, pela disponibilidade e apoio demonstrados.

A todos os meus colegas e amigos do Departamento de Informática, pelo ambiente criado, companheirismo e entreajuda demonstrados.

Ao Fábio pela amizade e pelos pertinentes comentários que contribuíram para a melhoria deste trabalho.

À Eliana pelo amor, carinho e imensa paciência nesta recta final.

Por último, não poderia deixar de agradecer à minha família, e em particular aos meus pais, pelo permanente estímulo e carinho, sem eles nada disto seria possível.

Conteúdo

Lista de Figuras

Lista de Tabelas

Acrónimos

A

ADL Architecture Description Language, p. 26.

AMP Autonomic Management Process, p. 21.

AOP Aspect Oriented Programming, p. 21.

APIs Application Programming Interface, p. 6.

APR Apache Portable Runtime, p. 53.

ASM Application Server Module, p. 25.

B

BARK Bean Automatic Reconfiguration frameworK, p. 24.

C

COTS Commercial Off-The-Shelf, p. 2.

CTS Compatibility Test Suite, p. 52.

D

DELADAS DEclarative LAnguage for Describing Autonomic Systems, p. 21.

DSD Deployable Software Description, p. 22.

DTP Data Transfer Process, p. 45.

E

`EIS` Enterprise Information Systems, p. 12.

`EJB` Enterprise Java Beans, p. 9.

`ERP` Enterprise resource planning, p. 12.

F

`FTP` File Transfer Protocol, p. 45.

I

`IDL` Interface Definition Language, p. 30.

J

`JDBC` Java Database Connectivity, p. 7.

`JCA` JEE Connector Architecture, p. 12.

`JEE` Java Platform Enterprise Edition, p. 7.

`JME` Java Platform Micro Edition, p. 7.

`JMS` Java Message Service, p. 9.

`JMX` Java Management Extensions, p. 53.

`JNDI` Java Naming and Directory Interface, p. 12.

`JSE` Java Standard Edition, p. 7.

`JSF` JavaServer Faces, p. 8.

`JSP` JavaServer Pages, p. 8.

`JSTL` JavaServer Pages Standard Tab Library, p. 9.

`JTA` Java Transaction API, p. 12.

`JVM` Java Virtual Machine, p. 28.

L

`LDAP` Lightweight Directory Access Protocol, p. 12.

M

`MTBF` Mean Time Between Failures, p. 34.

`MTTR` Mean Time To Repair, p. 34.

N

`NIS` Network Information Service, p. 12.

`NVT` Network Virtual Terminal, p. 45.

O

`ORM` Object-Relational Mapping, p. 55.

P

`PDA` Personal Digital Assistants, p. 20.

`PI` Protocol Interpreter, p. 45.

`POJOs` Plain Old Java Objects, p. 52.

`PVUs` Processor Value Units, p. 57.

R

`RAID` Redundant Array of Independent Drives, p. 41.

`RAIDb` Redundant Array of Inexpensive Databases, p. 79.

`RMI` Remote Method Invocation, p. 18.

S

`SCP` Secure Copy, p. 47.

`SGBD` Sistemas de Gestão de Bases de Dados, p. 37.

`SMP` Symmetric Multiprocessor, p. 40.

`SNMP` Simple Network Management Protocol, p. 87.

`SOA` Service Oriented Architecture, p. 53.

`SOAP` Simple Object Access Protocol, p. 53.

`SQL` Structured Query Language, p. 9.

`SSH` Secure Shell, p. 27.

`SSL` Secure Sockets Layer, p. 27.

`SVN` Subversion, p. 93.

W

`WCF` Windows Communication Foundation, p. 59.

Capítulo 1

Introdução

Desde a sua criação até aos dias de hoje, a importância da Internet tem crescido rapidamente. Muito mais do que um canal de comunicação, a Internet é um meio de distribuição de informação. Efectivamente, a sua velocidade de disseminação torna-a numa "auto-estrada"da informação e, em virtude dessa sua capacidade, o número de plataformas Web está a aumentar consideravelmente.

Actualmente, as empresas de Tecnologia de Informação concedem uma maior importância de como e onde é efectuada a instalação e configuração das aplicações que desenvolvem, uma vez que a arquitectura[1] (infra-estrutura) onde vai ser instalada a aplicação é um factor essencial para a sua qualidade.

Para uma aplicação ter sucesso, já não basta ter sido bem desenvolvida, sem conter erros e responder da forma esperada a todas as especificações para que foi desenvolvida. Com o aparecimento e expansão das aplicações Web conceitos como a disponibilidade ou escalabilidade ganharam maior importância. Estes só podem ser alcançados, de forma satisfatória, se a arquitectura que suporta a aplicação estiver de acordo com as necessidades da mesma.

Cada vez mais, as aplicações são compostas por um elevado número de componentes que oferecem e necessitam de serviços de outros componentes [Dea07]. As abordagens de Engenharia de Software baseadas em componentes têm sido consolidadas ao longo dos tempos, sendo que, a sua essência consiste em combinar componentes para formar aplicações mais complexas. Segundo [Kru98], um componente pode ser considerado um elemento significativo, independente e uma parte substituível de um sistema, que cumpre uma função clara no contexto de uma arquitectura bem definida.

Em cada componente é possível identificar uma série de propriedades básicas [GTWJ03], nomeadamente, identidade própria, ou seja, cada componente possui um identificador

[1] Arquitectura física da instalação

único que o permite distinguir de outros que se encontrem no mesmo ambiente. Sem esta propriedade, a reutilização de componentes seria praticamente impossível. São modulares, permitem o encapsulamento de dados, customizáveis, polimórficos, inter-operáveis, podendo, ainda, ser compostos.

Os componentes comerciais são normalmente conhecidos por *Commercial Off-The-Shelf* (COTS). Estes são software de terceiros [Voa98], disponíveis para o público e podem ser comprados ou licenciados [Obe98]. Uma desvantagem deste tipo de componentes consiste na dependência para com a entidade que o desenvolveu, uma vez que, caso seja descoberto um erro, é necessário comunicar à empresa a quem o componente foi adquirido, de modo a que esta possa corrigi-lo e lançar uma nova versão.

No entanto, os erros e as limitações do componente podem ser descobertos de uma forma mais célere, dado que, mais organizações podem estar a utilizá-lo, o que aumenta o número de pessoas que interagem com o componente. Outra vantagem reside na potencial incorporação de novas tecnologias nos sistemas de forma mais rápida, uma vez que os COTS, aquando do seu desenvolvimento, podem ter embutidas essas novas tecnologias.

Com a reutilização de componentes em múltiplos sistemas torna-se possível reduzir os recursos, o tempo necessário para o desenvolvimento dos mesmos, além de permitir a diminuição do período de testes [RM01] e, ainda, aumentar a produtividade das pessoas responsáveis pelo seu desenvolvimento [Hui99]. É de salientar que um componente deve ser adequadamente testado [Wey98] e bem documentado, de forma a minimizar os riscos da sua reutilização num novo contexto, sendo que, aquando da sua reutilização, há que ter em conta certas características do componente, nomeadamente:

- Portabilidade - caracteriza como os componentes interagem com a plataforma subjacente. De acordo com as tecnologias utilizadas aquando do desenvolvimento do componente, este pode apenas funcionar em determinadas plataformas [Gil06].
- Eficiência - indica a capacidade de um componente ter o desempenho apropriado, tendo em conta os recursos que utiliza [SKG08].
- Confiabilidade - pode ser considerada a probabilidade de um componente falhar num determinado período de tempo [SKG08].
- Capacidade de manutenção - descreve a capacidade de um componente ser modificado. Como em alguns casos o código fonte do componente não está disponível, este tem de ser adaptado e reconfigurado para poder ser reutilizado [SKG08].

A não reutilização dos componentes disponíveis vai implicar a reinvenção de soluções, sendo que só se justifica quando o custo de desenvolvimento é inferior ao custo da pos-

sível aquisição do componente, ou quando a qualidade da nova solução desenvolvida for bastante superior à já existente [Szy02].

A actualização dos sistemas baseados em componentes também se torna menos problemática, uma vez que, normalmente, será apenas necessário substituir o componente que sofreu a actualização [Ste98], desde que este respeite as interfaces da versão anterior, de modo a que o sistema possa continuar a funcionar normalmente, sem alterar outros componentes. Caso isso não seja possível, urge que sejam descritas novas dependências nas especificações do novo componente.

Apesar de todas as vantagens que a Engenharia de Software baseada em componentes possa ter, há que ter sempre em conta que um sistema que usa componentes herda os defeitos e limitações dos mesmos. Assim, a confiabilidade e segurança de um sistema está dependente da confiabilidade e segurança de cada componente que esteja a ser utilizado.

1.1 Motivação

Com a Internet a tornar-se rapidamente num centro de aplicações cada vez mais complexas e diversificadas, é fundamental garantir um padrão de qualidade nas plataformas Web disponibilizadas. Através da Engenharia de Software baseada em componentes foi possível, não só facultar maior confiabilidade às plataformas Web, como também, permitir que estas se adaptem mais rapidamente a mudanças.

A passagem das aplicações para a Internet faz com que estas não se encontrem única e exclusivamente dependentes delas próprias. Não basta que as aplicações sejam bem desenvolvidas para terem o desempenho esperado e cumprirem os requisitos para as quais foram projectadas. É, igualmente, essencial ter em consideração a infra-estrutura que as irá alojar. De facto, sem uma infra-estrutura robusta, o risco da plataforma Web ficar indisponível com um considerável aumento do volume de pedidos ou a falha de um servidor da infra-estrutura, é elevada. Deste modo, torna-se necessário construir uma infra-estrutura capaz de se adaptar a diferentes fluxos de carga, devendo ser o mais escalável possível e com mecanismos que melhorem a sua disponibilidade. No entanto, esta construção está longe de ser um processo trivial, sendo, portanto, fundamental a escolha do tipo de servidores que a vão constituir, bem como, as tecnologias que irão estar presentes. Esta escolha está fortemente condicionada pelo tipo de aplicações Web que a infra-estrutura deverá alojar.

Para além da aplicação Web e da infra-estrutura, outro factor que pode condicionar o bom funcionamento de uma plataforma Web, é a forma como as aplicações são instaladas e configuradas na infra-estrutura. A crescente complexidade das aplicações pode aumentar

o risco de problemas na sua instalação e configuração, na medida em que:

- A aplicação está sujeita a incompatibilidade entre componentes;
- Incompatibilidade com outras aplicações que estão a correr no mesmo ambiente;
- O aparecimento de novas versões podem tornar a aplicação incompatível com os ambientes em que estão integradas;
- As aplicações estão cada vez mais dependentes de outras aplicações ou serviços, nomeadamente Sistema Operativo, Middleware, Base de Dados, entre outros [Dea07], e
- A actualização de um dos seus componentes pode inviabilizar a utilização do mesmo numa outra aplicação. [PDC01].

Por outro lado, pode ser feita a instalação e configuração de uma determinada aplicação em diversas fases do seu ciclo, nomeadamente, quando esta ainda está em desenvolvimento, na fase de testes ou já em produção. Neste sentido, cada fase pode conter um conjunto de requisitos diferentes [EKKP05].

Na fase de desenvolvimento, factores como a segurança e a disponibilidade da infra-estrutura não são prioritários, podendo ser retirados da arquitectura algumas *firewalls* e redundância [EKKP05] de componentes. No entanto, devemos considerar que na fase de testes nenhum dos factores anteriormente citados deve ser negligenciado, uma vez que, é imprescindível ter a certeza que o desempenho da aplicação está dentro dos parâmetros desejados, e torna-se prioritário que estes testes sejam realizados numa infra-estrutura o mais próximo possível da infra-estrutura final.

O deploy de aplicações é um processo complexo que abrange todas as actividades a serem efectuadas a partir do desenvolvimento dessa mesma aplicação até à instalação e manutenção da mesma [CE00].

A instalação e configuração dos ambientes que concretizam as arquitecturas das aplicações Web - redundantes e escaláveis - é uma actividade complexa que exige um considerável conhecimento técnico, constituindo-se, assim, uma tarefa relevante na área da Engenharia de Software.

1.2 Objectivos

O tema deste trabalho assenta na mais valia de proporcionar uma ferramenta gráfica, em ambiente Web, na qual se possa, de forma facilitada, desenhar e configurar os ambientes

nos quais as aplicações irão funcionar.

As aplicações que serão suportadas nestes ambientes são aplicações Web multi-camada baseadas em tecnologia JEE, com os necessários compromissos com a escalabilidade, redundância e disponibilidade.

Pretende-se ter como resultado final deste trabalho uma ferramenta que auxilie o utilizador em todo o processo de construção e configuração de uma infra-estrutura, bem como na instalação e configuração de uma aplicação numa infra-estrutura, automatizando, sempre que possível as tarefas referidas. É, ainda, objectivo que a ferramenta, além de configurar os ambientes, possa, também, em tempo de execução, e através de mecanismos de monitorização, obter algumas métricas referentes aos servidores, nomeadamente, utilização do processador e memória, entre outras, garantindo que as aplicações tenham o desempenho pretendido. A ferramenta deve, ainda, dar a possibilidade de reestruturar a arquitectura onde estão instaladas as aplicações, disponibilizando mecanismos que permitam, ao utilizador, acrescentar novos nós e parametrizá-los, dado que podem ser necessárias alterações para melhorar a qualidade de algumas métricas das aplicações Web, designadamente, um mais rápido acesso. Deste modo, a ferramenta resultante deverá permitir efectuar operações de optimização de desempenho, sem que seja necessária a produção ou alteração de código fonte.

De modo a cumprir com os objectivos, a presente dissertação divide-se em duas partes, uma primeira de contextualização teórica e, de seguida a componente prática. Na parte teórica apresentamos um estudo da problemática do Deployment de aplicações JEE, bem como o estudo de um modelo de infra-estrutura base que irá suportar as aplicações Web multi-camada. A componente prática, de prova de conceito, assenta no desenvolvimento de um protótipo demonstração de uma ferramenta com as características acima mencionadas.

1.3 Considerações linguísticas

Tratando-se de um trabalho com uma forte componente tecnológica, é praticamente inevitável o uso de expressões ou designações que não fazem parte do léxico da língua portuguesa. De facto, certos conceitos não apresentam uma designação uniforme em língua portuguesa, por vezes fruto de diversas traduções de um mesmo termo em inglês. Procuramos, sempre que possível, utilizar os termos em português, privilegiando as variantes com aceitação mais generalizada pela comunidade científica. O recurso a termos em inglês foi feito quando se considerou necessário para uma melhor compreensão do conceito em questão e para uma melhor interpretação das figuras apresentadas. Nesses casos houve

o cuidado de, na primeira vez que ocorrem, o sinalizar a itálico de forma a destacar a sua génese. Um caso particular de um termo em inglês muito utilizado ocorre com a palavra *deployment*, que pode ser definido como a instalação e configuração de uma aplicação. Por ser uma palavra com uma carga semântica forte e aceite pela comunidade para designar as tarefas anteriormente explicitadas, utiliza-se o termo em inglês, por se considerar que para os especialistas da área é mais significativo que qualquer uma das possíveis traduções em Português.

1.4 Organização do Documento

Para além do capítulo introdutório, estruturamos a presente dissertação em seis capítulos, descritos de seguida.

Tendo em conta que esta dissertação é focada na plataforma JEE, no capítulo 2 é feita uma introdução à mesma. Neste capítulo são abordados assuntos como as aplicações multi-camada, a constituição dos servidores aplicacionais, necessários para executar as aplicações JEE e, ainda, as *Application Programming Interfaces* (`APIs`) JEE que são disponibilizadas.

No capítulo 3 é elaborado um levantamento bibliográfico de vários estudos relacionados com a presente dissertação. Neste são apresentados estudos e ferramentas sobre a escalabilidade dos servidores e das aplicações J2EE, abordagens para o apoio da instalação e configuração de aplicações e, ainda, sobre monitorização e gestão de aplicações.

O capítulo 4 apresenta um estudo em torno da problemática da construção e gestão de infra-estruturas. São apresentados alguns fundamentos teóricos importantes como confiabilidade, disponibilidade e escalabilidade, bem como uma análise sobre a arquitectura de uma infra-estrutura JEE. Temas como a monitorização da infra-estrutura e a comunicação remota com os servidores da mesma são, igualmente, abordados.

No capítulo 5 o foco é o ambiente de deployment, nomeadamente, o estudo de alguns processos de deployment presentes na literatura. São apresentados alguns servidores aplicacionais JEE, bem como técnicas e ferramentas para o deployment e gestão de aplicações JEE.

O capítulo 6 apresenta os passos e decisões necessárias ao desenvolvimento da ferramenta, sendo, ainda realizada, uma descrição exaustiva das funcionalidades do mesmo. Os testes elaboradas com a ferramenta e respectiva avaliação são a temática do capítulo 7.

No último capítulo apresentamos um balanço global do trabalho realizado com as devidas considerações finais.

Capítulo 2

Enterprise Java

Actualmente o Java é uma das linguagens mais utilizadas e serve para desenvolver qualquer tipo de aplicação, designadamente: aplicações Web, desktop, servidores, mainframes, jogos, aplicações móveis, entre outras.

Existem três tipos de plataformas Java, as *Java Standard Edition* (`JSE`), as *Java Platform Enterprise Edition* (`JEE`) e as *Java Platform Micro Edition* (`JME`).

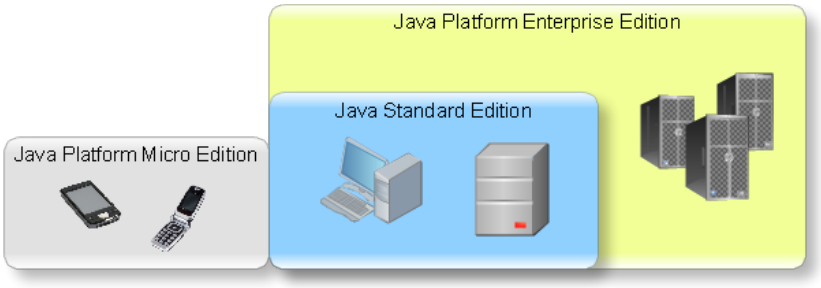


Figura 2.1: Plataforma Java

Java Platform Enterprise Edition, é um dos três tipos de plataformas Java, e define-se como um conjunto de especificações e regras com suporte ao desenvolvimento de aplicações robustas e escaláveis. Construída no topo da versão Java Standard Edition aproveita a vantagem de muitas das suas características, como a portabilidade "Write Once, Run Anywhere", a API *Java Database Connectivity* (`JDBC`) para o acesso a Base de Dados, tecnologia CORBA para interacção com recursos empresariais existentes, entre outros. O `JEE` reúne um conjunto de serviços - `APIs` e protocolos - que oferecem funcionalidades para o desenvolvimento de aplicações multi-camada, baseadas em componentes, e tem por objectivo a redução do custo e complexidade de desenvolvimento das mesmas.

Enquanto que o Java SE é utilizado para o desenvolvimento de aplicações para computadores pessoais ou workstations com arquitecturas mono ou multiprocessador, o Java

EE está focado nas aplicações do tipo cliente-servidor para redes, intranets e Internet [Mar06], como pode ser observado na Figura 2.1. Outra diferença resido no facto de, ao contrário das aplicações JSE, as aplicações JEE necessitam de um servidor aplicacional para serem executadas.

2.1 Aplicações multi-camada (multi-layer)

A arquitectura JEE permite o desenvolvimento de aplicações divididas em camadas com funcionalidades específicas. A capacidade de construir aplicações multi-camada proporciona uma maior e mais fácil capacidade de manutenção da aplicação, possibilidade de reutilização dos componentes por parte de outras aplicações e torna a aplicação mais escalável.

Na sua maioria, uma aplicação multi-camada é constituída por uma camada de apresentação, uma camada de negócio e uma camada de dados. De seguida são apresentadas breves descrições sobre cada uma das camadas lógicas.

2.1.1 Camada de Apresentação

A camada de apresentação (*presentation layer*) é responsável por fornecer um meio de comunicação entre o utilizador e os serviços disponibilizados pelos sistema, tendo, ainda, a responsabilidade de interagir com outras camadas da aplicação.

Esta pode ser constituída por uma aplicação cliente que acede ao servidor aplicacional JEE para possibilitar a interacção do mesmo com o utilizador. De salientar que a aplicação cliente não tem de ser necessariamente desenvolvida em Java.

Uma outra forma de interagir com o utilizador e através de uma camada Web, onde o utilizador, através do browser, acede remotamente aos serviços disponibilizados. A camada Web tem a responsabilidade de recolher o dados e os pedidos introduzidos pelo utilizador, e depois de apresentar-lhe os resultados provenientes da camada de negócio. Outra função, de grande importância, desta camada é a necessidade de gerir e garantir a coerência dos dados de sessão do utilizador.

Para o desenvolvimento da camada Web são disponibilizadas várias tecnologias JEE, nomeadamente, os *Servlets* que são classes java que processam pedidos e constroem respostas de forma dinâmica. As *JavaServer Pages* (JSP) permitem a criação simples de conteúdo Web que possui tanto componentes estáticos como dinâmicos. As páginas JSP são compostas por código HTML/XML misturado com etiquetas especiais para colocar código Java. A *JavaServer Faces* (JSF) é uma framework que tem como objectivo

simplificar o desenvolvimento de interfaces Web, através de um conjunto de componentes pré-definidos. Possui,ainda, mecanismos para validação de entradas e conversão de dados. A *JavaServer Pages Standard Tab Library* (JSTL) que encapsula funcionalidades essenciais e comuns a muitas páginas JSP, designadamente, iteradores, tags para manipulação de ficheiros XML, tags *Structured Query Language* (SQL), entre outras.

Além disso, é exequível utilizar componentes JavaBeans para guardar temporariamente dados das paginas da aplicação.

2.1.2 Camada de Negócio

A camada de negócio contém os componentes que disponibilizam toda a lógica de negócio de uma aplicação. Esta camada dá a conhecer a lógica de negócio à camada de apresentação, ou mesmo, a outras aplicações remotas, através de interfaces bem definidas.

Para a implementação da camada de negócio podemos utilizar *Enterprise Java Beans* (EJB).

Existem três tipos de EJB: os Session Beans, os Entity Beans e os Message Driven Beans.

Os Session Beans representam processos síncronos de negócio. Estes, podem ser classificados em dois tipos:

- *Stateful Session Beans* - mantém um relacionamento (diálogo) continuado com um cliente. O estado é mantido durante a sessão e é criada uma instância para cada cliente que invoca o construtor do Bean.

- *Stateless Session Beans* - não mantém o estado durante a sua interacção com o cliente. São mais escaláveis, por serem menos "pesados", uma vez que não precisam de guardar estado.

Os Entity Beans representam entidades guardadas em suporte persistentes, são utilizados na camada de dados / camada de persistência. Já os Message Driven Beans representam processos de negócios assíncronos e são invocados via *Java Message Service* (JMS) ou através de outro sistema de mensagens.

2.1.3 Camada de Dados

A camada de dados (data layer ou camada de persistência) permite isolar o acesso aos dados. Esta possui os componentes responsáveis pela lógica inerente à persistência e recuperação de informações de algum tipo de repositório de dados. O objectivo desta camada

é fornecer uma abstracção capaz de encapsular toda a lógica de acesso aos dados, deixando transparente à camada de negócio como são realizadas tais operações, permitindo que esta não esteja dependente da origem ou estrutura sob a qual os dados estão armazenados.

Para a implementação desta camada podemos utilizar Hibernate [BK06], TopLink [Spe03], OpenJPA [HBB+08], entre outros.

2.2 Servidor aplicacional JEE

Como foi dito anteriormente, as aplicações JEE desenvolvidas, para correr, necessitam de estar instaladas e configuradas num servidor aplicacional.

Os servidores aplicacionais com a especificação JEE oferecem diferentes tipos de *containers*, que disponibilizam serviços aos componentes, nomeadamente gestão de ciclo de vida, segurança, persistência de dados, entre outros, como pode ser observado na Figura 2.2. Estes containers ajudam a diminuir a complexidade de desenvolvimento dos componentes, uma vez que o programador pode concentrar os seus esforços no desenvolvimento da lógica de negócio da aplicação, sendo que a *lógica não funcional* da mesma é feita automaticamente.

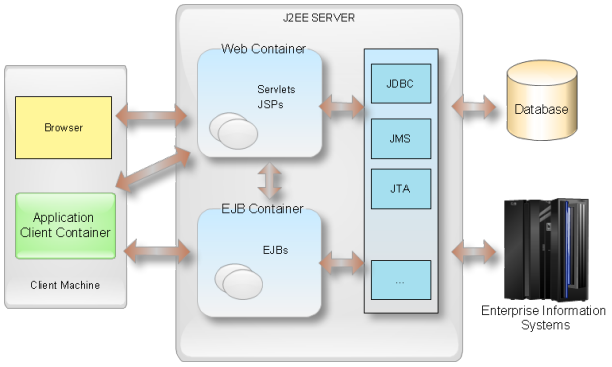


Figura 2.2: Arquitectura JEE

Os containers são uma interface entre um componente e a funcionalidade de baixo nível da plataforma que suporta o mesmo componente[1]. De seguida são descritos alguns deles:

- *Application client container* - efectua a gestão da execução de componentes das aplicações cliente.
- *Web container* - interface entre componentes Web e servidor Web. É responsável pelo ciclo de vida dos componentes Web, remete pedidos para os componentes da

[1] http://java.sun.com/j2ee/1.4/docs/tutorial/doc/Overview3.html

aplicação e disponibiliza interfaces para dados. Possui suporte para Servlets, que são classes Java que podem ser carregadas dinamicamente e executar sob um servidor Web, e para páginas `JSP`.

- *Enterprise JavaBeans (EJB) container* - contém e executa componentes `EJB` e fornece um conjunto de serviços para esses componentes, nomeadamente, gestão do ciclo de vida do componente, gestão de persistência, gestão de transacções, entre outros. Possui suporte para Sessions Beans, Entity Beans e Message-Driven Beans.

A abstracção dos containers permite que o comportamento dos componentes seja especificado em tempo de configuração (deployment), possibilitando, desta forma, uma maior flexibilidade.

2.3 Principais APIs do JEE

A plataforma JEE inclui várias `APIs` que fornecem acesso a base de dados, transacções, processamento de XML, serviço de nomes, entre outros, facilitando, deste modo, o desenvolvimento da aplicação. A Figura 2.3 ilustra todas as APIs[2] existentes em cada container para o desenvolvimento de uma aplicação JEE.

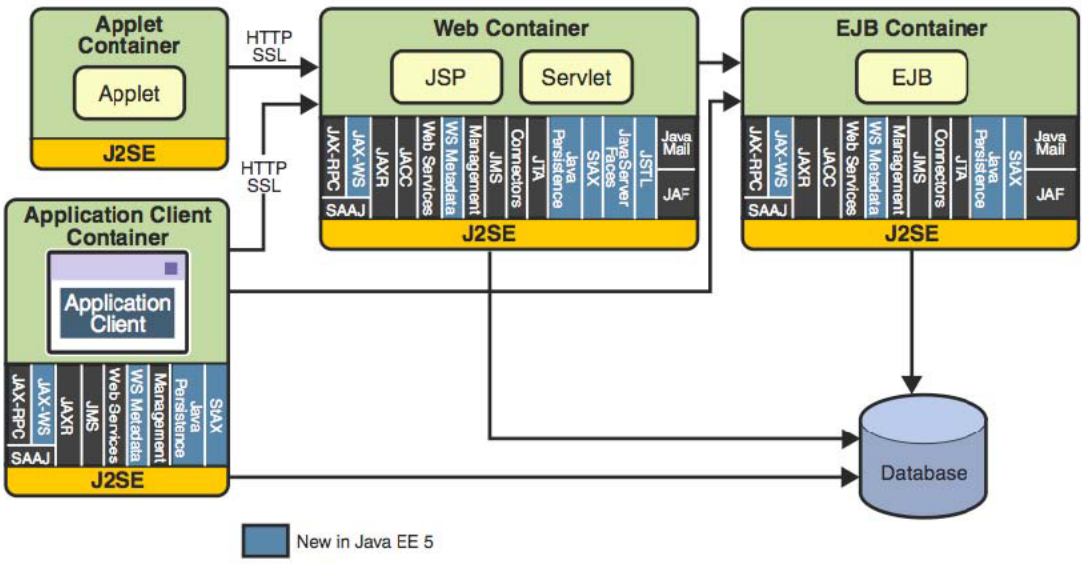


Figura 2.3: APIs existentes em cada container JEE

Alguns dos mais importantes serviços disponibilizados pela plataforma JEE são:

[2] `http://java.sun.com/j2ee/1.4/docs/tutorial/doc/Overview3.html`

- *Java Database Connectivity* (JDBC) - API de acesso a base de dados. Com esta API, para além da possibilidade de conectividade com vários motores de base de dados, podemos gerir transacções, executar Stored Procedures e ainda, examinar e modificar os resultados obtidos nas declarações "Select" [SSJ02].

- *JEE Connector Architecture* (JCA) - Esta API tem como função criar um padrão para a ligação entre plataformas JEE e *Enterprise Information Systems* (`EIS`), como por exemplo, *Enterprise resource planning* (`ERP`). A `JCA` fornece ainda mecanismos escaláveis e seguros para ajudar essa mesma integração.

- *Java API for XML Processing* (JAXP) - É a API utilizada para o processamento de ficheiros XML. Estes podem ser processados utilizando SAX ou DOM, bem como XSLT.

- *Java Naming and Directory Interface* (JNDI) - Permite aceder aos serviços de nomes e directórios. É através do `JNDI` que podemos localizar os componentes Enterprise Java Beans.

  Tal como muitas outras especificações em Java, é projectada para ser independente da implementação do serviço. Ou seja, os clientes irão aceder a directórios *Lightweight Directory Access Protocol* (`LDAP`) e *Network Information Service* (`NIS`) através do mesmo código Java.

- Java Transaction API (JTA) - Conjunto de interfaces para a gestão de transacções. A `JTA` fornece uma forma dos componentes JEE gerirem suas próprias transacções.

2.4 JEE Deployment

Quando apareceram os primeiros servidores aplicacionais, eram estes que decidiam como as aplicações corriam e a forma como eram instaladas e configuradas. Correr o mesmos EJBs em distintos servidores aplicacionais, podia ser uma tarefa extremamente complexa, uma vez, que cada servidor tinha os seus parâmetros de deployment e as configurações para cada um eram diferentes.

Com a chegada da versão 1.4 do JEE as `APIs` de instalação e configuração, das aplicações desenvolvidas, foram mais especificadas [Exe04]. Nessa proposta é visível uma separação clara entre a plataforma JEE e a ferramenta utilizada para a instalação e configuração das aplicações.

De forma a poder ser instalada e configurada num servidor aplicacional JEE, a aplicação tem de ser de um dos seguintes tipos [Suna]:

- Enterprise Java Beans JAR - contêm classes de `EJB`.
- WAR - contêm aplicações Web, nomeadamente, Serverls e páginas `JSP`.
- RAR - contêm classes de JEE CA Connecter. Estas classes permitem o acesso a EIS externos através de componentes JEE.
- EAR - permite o empacotamento de Enterprise Java Beans JAR e Web Components JAR (WARs).

Todos estes tipos de empacotamentos de aplicações JEE incluem ficheiros descritores XML, que contêm informações sobre a aplicação. E é precisamente, através destas descrições que os containers sabem como devem gerir a aplicação [FC05].

O assunto dos descritores será abordado mais detalhadamente no Capitulo 5.

Capítulo 3

Trabalho Relacionado

Este capítulo apresenta os trabalhos relacionados que foram estudados e que formam a base teórica para este projecto.

Dividiremos este capítulo em três partes. Na primeira, são apresentados alguns estudos sobre a escalabilidade dos servidores e das aplicações JEE. A segunda parte dá a conhecer ferramentas e abordagens para o apoio da instalação e configuração de aplicações. Por fim, são descritas técnicas e ferramentas que permitem a monitorização e gestão de aplicações.

3.1 Escalabilidade dos Servidores e das Aplicações JEE

A capacidade que uma plataforma Web em adaptar-se às constantes variações de *workload* que vai receber é, cada vez mais, de extrema importância [BH04]. Estas variações, obrigam que as plataformas sejam escaláveis o suficiente para manter os padrões de desempenho exigidos. Tendo em consideração que a escalabilidade não está apenas ligada à forma a como uma aplicação é desenvolvida e quais as tecnologias que utiliza, mas também à arquitectura na qual está inserida, vários estudos foram elaborados que testam não só a escalabilidade da tecnologia JEE, mas também, a escalabilidade de alguns dos servidores JEE existentes.

P. Brebner e J. Gosper em [BG03] realizaram vários testes de forma a constatar até que ponto a tecnologia JEE é escalável. Para a realização desses mesmos testes foi utilizada a benchmark ECperf [Mic]. Esta benchmark mede a performance e a escalabilidade dos servidores aplicacionais JEE.

A partir dos resultados obtidos foi possível perceber que proporcionar a mesma qualidade de serviço a um número maior de utilizadores, torna-se exponencialmente mais

dispendioso, sendo que estes custos estão maioritariamente relacionados com o hardware necessário. Assim, e de acordo com o que foi concluído, torna-se mais vantajoso em termos de rácio de desempenho, a escalabilidade horizontal (Secção 2.3) do que a vertical (Secção 2.3).

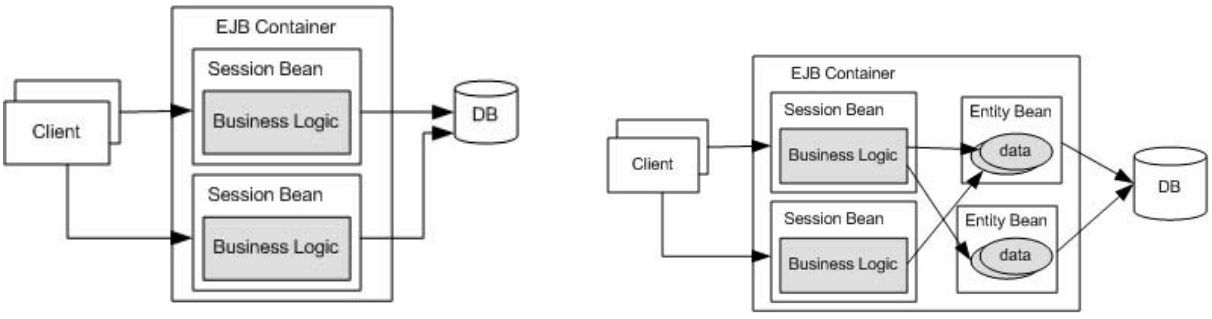
Um estudo comparativo entre duas formas de obter a escalabilidade num servidor aplicacional JEE, utilizando técnicas de replicação ou técnicas de particionamento, foi elaborado em [SDH06]. A replicação consiste em ter uma cópia completa da aplicação que está a correr em todos os nodos existentes do servidor aplicacional, enquanto, o particionamento tem como base a fragmentação da aplicação pelos vários nodos.

Para efectuar os testes foi utilizada a benchmark RUBis [Uni], que simula uma plataforma de leilões idêntica ao eBay. Como servidor `EJB` foi escolhido o JOnAS e utilizado um middleware para permitir escalabilidade na Base de Dados, de modo a que esta não ficasse saturada antes do servidor `EJB`.

Foi possível perceber que uma solução com replicação consegue ter um melhor desempenho do que uma com particionamento de `EJB`s, sendo que, este resultado se pode dever ao facto de na solução com particionamento serem necessárias mais chamadas remotas. É de salientar que a solução que obteve os melhores resultados, neste estudo, foi uma solução híbrida entre a replicação e o particionamento. No entanto, esta solução implica um conhecimento profundo da aplicação que vai ser utilizada.

No caso de [Liu02] foi feita uma investigação para tentar mediar a performance e escalabilidade de `COTS` baseada numa análise da arquitectura e tecnologia dos mesmos.

A benchmark utilizada para fazer os testes consiste num *'ping'* que utiliza toda a infra-estrutura `JEE`. Esta benchmark foi implementada em duas arquitecturas distintas: numa foi utilizada uma arquitectura *Stateless Session Bean*; e na outra uma arquitectura *Stateless Session Bean Facade* [Mar02]. Um esquema dessas duas diferentes arquitecturas pode ser observado na Figura 3.1.

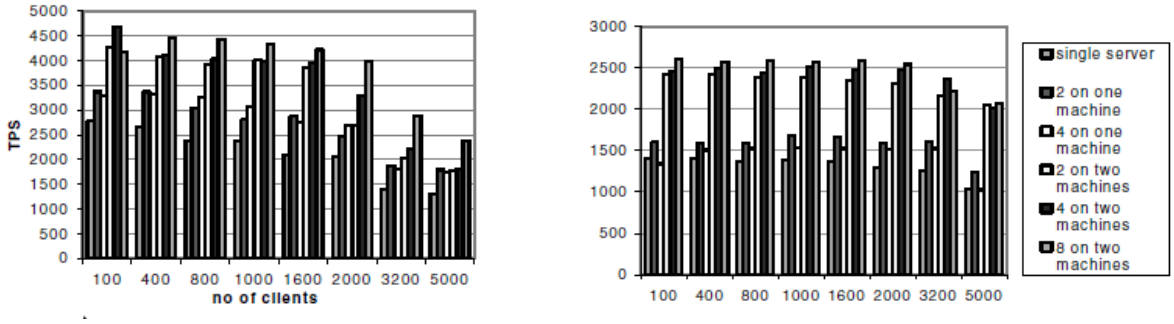


(a) Arquitectura Stateless Session Bean (b) Arquitectura Stateless Session Bean Facade

Figura 3.1: Diferentes Arquitecturas da Benchmark [Liu02]

Como podemos verificar na arquitectura Stateless Session Bean, o session bean contém, não só o código da camada de negócio, mas também o código para aceder à Base da Dados. Na arquitectura Stateless Session Bean Facade, o código referente aos acessos à Base de Dados foi colocado num entity bean.

Ambas as implementações irão ser testadas com uma carga que vai desde 100 até 5000 clientes. Estas vão ver colocadas em diferentes configurações de infra-estruturas, para deste modo verificar até que ponto a infra-estrutura tem efeitos no desempenho e escalabilidade das aplicações.



(a) Session Bean

(b) Session Bean Facade

Figura 3.2: Resultados obtidos com diferentes configurações de clustering [Liu02]

Na Figura 3.2 são mostrados os resultados obtidos nos testes efectuados. Como é possível perceber a implementação com Stateless Session Bean conseguiu ter throughputs mais elevados. Quanto à configuração da infra-estrutura, é possível observar bastantes variações de acordo com o número de clientes, sendo que, na sua maioria, a infra-estrutura constituída por 8 instâncias de servidores dividas por dois computadores obteve os melhores resultados.

Neste estudo foi possível perceber que a escalabilidade e o desempenho da solução depende não só da arquitectura da infra-estrutura que a suporta, mas também da arquitectura utilizada no desenvolvimento da aplicação.

Em [MZ05] é desenvolvida uma investigação de modo a compreender de que forma o clustering dos servidores aplicacionais tem influência na escalabilidade e performance de uma aplicação. Para o efeito foi utilizada como aplicação a benchmark ECperf [Mic] e como servidor aplicacional foi escolhido o JBoss [Red].

Foram efectuados diferentes testes para um número variado de servidores, designadamente, a replicação apenas dos session beans pelos vários servidores, replicação de todos os beans, base de dados distribuídas, entre outros.

Com estes testes, chega-se à conclusão que diferentes arquitecturas de deployment podem melhorar a performance e a escalabilidade das aplicações sem ser necessário alterar

o código destas.

Na investigação elaborada em [CMZ02] tenta-se compreender os efeitos do método utilizado na implementação da aplicação, do design do container e da eficiência da camada de comunicações na performance e escalabilidade dos servidores aplicacionais JEE.

Para efectuar as experiências, foi elaborado uma plataforma Web de leilões, sendo que esta foi desenvolvida utilizando 5 implementações diferentes, cada uma com um método específico. Como servidores aplicacionais foram utilizados o JBoss ou o JOnAS, com diferentes configurações, designadamente, JBoss 2.4.4, JBoss 2.4.4 - optimized calls, JOnAS 2.4.4 - `RMI` e JOnAS 2.4.4 - Jeremie.

Nestes testes o sistema vai estar sujeito a dois tipos de operações, um *Browsing Mix* que é constituído apenas por operações de leitura e um *Bidding Mix* onde 15% das acções são leitura-escrita.

As Figuras 3.3 e 3.4 apresentam os resultados obtidos para as diversas combinações possíveis entre os métodos utilizados para implementar a aplicação e as configurações dos servidores aplicacionais JEE.

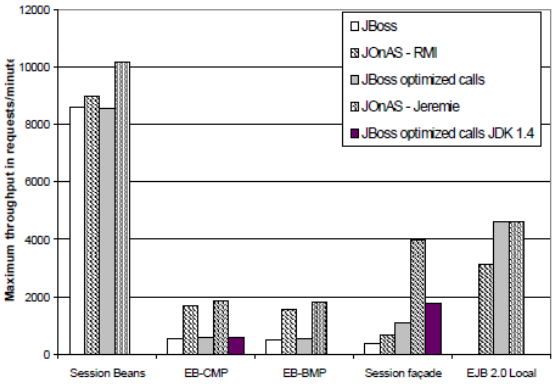


Figura 3.3: Interacções/Min - Browsing Mix [CMZ02]

Os resultados mostram que a implementação com Session Beans obteve o melhor throughput em ambos os tipos de operações, sendo que, a diferença para com os outros tipos de implementações foi bastante significativa. Por outro lado a separação do código de acesso aos dados dos servlets e a colocação deste em entity beans, EB-CMP e EB-BMP, obtiveram os piores resultados.

Quanto ao desempenho das várias configurações dos servidores aplicacionais é de destacar os resultados obtidos com o JOnAS -Jeremie, que é o servidor aplicacional JOnAS juntamente com uma implementação optimizada do Java RMI, Jeremie, como camada de comunicação. Em todos os testes, foi com esta configuração que se obtiveram os melhores resultados, excepção feita, ao teste elaborado com a utilização de `EJB` 2.0, como é visível

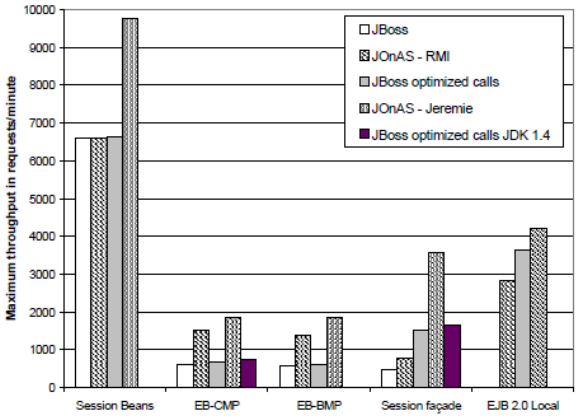


Figura 3.4: Interacções/Min - Bidding Mix [CMZ02]

na Figura 3.3.

Como foi possível constatar pelos diferentes estudos acima citados, a infra-estrutura JEE pode ser bastante escalável, sendo possível obter a performance necessária para o sucesso de uma aplicação. No entanto, é necessário conhecer a natureza da aplicação e fazer um estudo cuidado da eventual carga de pedidos que irá ter de suportar, para que, deste modo, seja feita uma escolha correcta, quer da arquitectura da própria aplicação a ser desenvolvida, quer da arquitectura da infra-estrutura que irá receber a aplicação.

3.2 Instalação e configuração de Aplicações

O processo de instalação e gestão de aplicações é definido como uma sequência actividades interligadas que têm como objectivo disponibilizar uma aplicação para uso [HMA06]. Com o aumento da complexidade das aplicações, estes processos tornaram-se, também, cada vez mais complicados. Na presente secção são apresentadas algumas ferramentas e técnicas de apoio à instalação, configuração e gestão de aplicações.

Em [MB04] é proposto um ambiente de deployment para aplicações onde é suportado a instalação, configuração, reconfiguração e desinstalação. Podemos distinguir três entidades: servidores de aplicações (*applications servers*), servidor de deployment (*deployment server*) e clientes (*client site*).

Os servidores de aplicações funcionam como um repositório de aplicações e respectiva metadata. A metadata é especificada num documento XML que contém informações sobre a versão e nome da aplicação. Pode ainda conter informações relativas a dependências da aplicação, especificações como Sistema Operativo, espaço em disco necessário, entre outras.

Os servidores de deployment são os responsáveis por todo o processo de instalação das aplicações nos clientes. De forma a que essa instalação seja feita, são necessárias descrições, quer da aplicação, quer da máquina cliente que a irá receber.

Como pode ser observado na Figura 3.5, que apresenta um esquemático da arquitectura do ambiente de deployment e das entidades que o formam, o servidor de deployment funciona como um intermediário entre os servidores de aplicações e a máquina cliente.

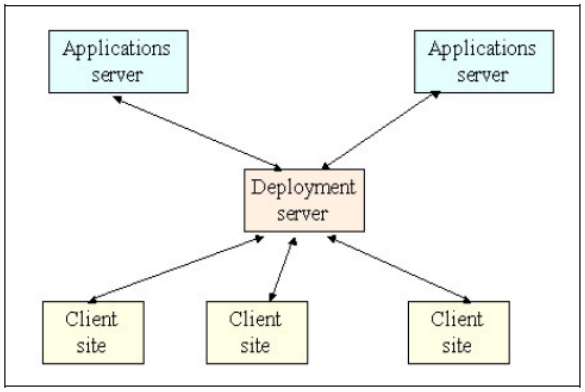


Figura 3.5: Arquitectura do ambiente de deployement ORYA [MB04]

A OSGi Service Platform [Alla] é uma plataforma para o deployment de serviços para uma variedade de dispositivos, nomeadamente, PDAPDAs, telemóveis, servidores, entre outros.A unidade base de deployment é designada por *bundle*. Este pode ser visto como um componente e pode estar dependente de outro *bundle* para funcionar.

A framework é, ainda, responsável por manter a consistência dos serviços ao controlar a relação de dependência entre os módulos instalados.

R. Sabharwa e J. Guijarro, em [SG08], apresentam uma ferramenta denominada *Avalanche*. Esta ferramenta permite a instalação distribuída de aplicações.

Todo o ciclo de vida do software é considerado pela ferramenta, designadamente, instalação, configuração, upgrades e desinstalação. Como mecanismo de deployment utiliza a framework SmartFrog [Sma], que é utilizada para descrever e gerir sistemas distribuídos.

Além disso, esta ferramenta disponibiliza um módulo de monitorização do sistema, que permite, entre outras coisas, verificar a carga que o processador está a ter.

No caso de [DKMy04] é-nos apresentado um sistema desenvolvido para suportar o deployment de aplicações, que oferecem serviços distribuídos.

Neste sistema é introduzido o conceito de *Thin Server*, servidores que podem ser colocados na rede do sistema. Estes complementam a infra-estrutura existente, de modo a aumentar a usabilidade e eficiência da mesma.

Este framework disponibiliza, ainda, mecanismos de Binding, que permitem aos componentes comunicar com o código, data e processos locais, além de possibilitar a comunicação entre nodos e entre componentes.

Kirby et all, em [KDM04], propõem uma framework para deployment e gestão autónoma de aplicações distribuídas baseadas em componentes.

Nesta framework é proposta uma linguagem declativa, *DEclarative LAnguage for Describing Autonomic Systems* (`DELADAS`), para especificar determinadas condições do deployment, nomeadamente, mapeamento dos componentes pelos servidores, o tipo de comunicação entre os componentes, os recursos que são necessários estarem disponíveis para o correcto funcionamento da aplicação, entre outros.

O mecanismo de gestão e deployment (ADME), presente na framework, tenta cumprir os objectivos descritos, dando a conhecer as soluções possíveis. Essas soluções estão na forma de ficheiros XML conhecidos por *Deployment Description Documents*.

Cada aplicação instalada leva consigo um mecanismo, *Autonomic Management Process* (`AMP`), responsável por monitorizar os componentes presentes no servidor. O `AMP` é capaz de gerar mensagens quando determinado evento acontece, enviando posteriormente informações a uma instância do ADME, o *Monitoring ADME* (MADME), como podemos observar na Figura 3.6. Esta ao receber as informações irá verificar se as condições especificadas anteriormente ainda estão a ser cumpridas.

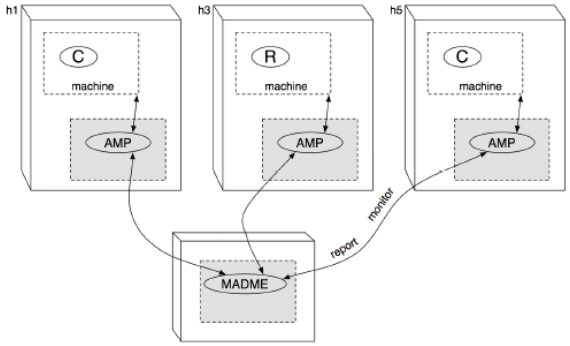


Figura 3.6: Gestão autónoma de aplicações [KDM04]

Uma proposta para verificar se os requisitos de um componente de software estão a ser cumpridos num determinado cenário de deployment é apresentada em [GD02].

A descrição dos requisitos funcionais e não funcionais dos componentes, como performance, segurança, recursos utilizados, entre outras, é feita com *Aspect Oriented Programming* (`AOP`) [JN04, GL03]. Esta informação é guardada em ficheiros XML, sendo analisada, depois do deployment, por agentes que irão verificar, através de testes automáticos, se todos os requisitos do componente estão a ser cumpridos.

Após a realização dos testes são apresentados vários relatórios com os resultados obtidos.

Em [KB05] é apresentada uma framework, *DYVA*, que foi concebida para se adequar à reconfiguração e instalação dinâmicas da maioria das tecnologias de componentes existentes. Esta framework baseada na abordagem model-driven, pode ser personalizada, através de plugins, para uma tecnologia de componentes específicas. A personalização torna-se necessária para se poder utilizar na prática o sistema, uma vez que, segundo os autores, a framework foi desenvolvida para ser genérica, sendo necessário personalizá-la para uma tecnologia específica, de modo a usufruir das suas capacidades.

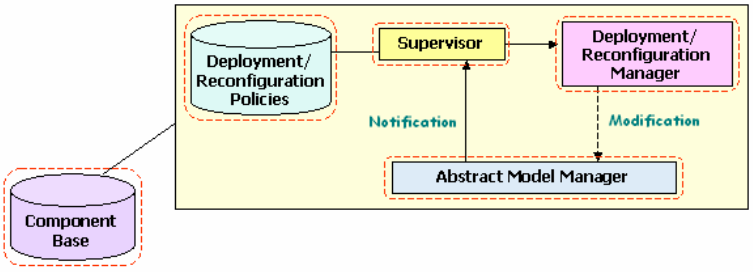


Figura 3.7: Arquitectura da framework DYVA [KB05]

Na Figura 3.7 podemos observar a arquitectura da framework DYVA. Esta é constituída por um módulo que implementa e fornece as rotinas referentes à reconfiguração e instalação dinâmicas (Deployment/Reconfiguration manager). Supervisor que tem como função permitir a auto-reconfiguração e auto-instalação de aplicações. E ainda, um módulo que tem as directrizes que tornam possível uma reconfiguração autónoma por parte das aplicações, sem a intervenção de algum actor externo (Reconfiguration policies).

Uma arquitectura genérica para o desenho e instalação de componentes de gestão e configuração autónomas é apresentada em [PA06]. Esta framework fornece a capacidade binding dinâmico e permite a substituição de um componente por outro em runtime, sem afectar a confiabilidade do sistema.

Software Dock [HHW99] é uma framework de deployment baseada em agentes que tem por objectivo fomentar a cooperação entre as várias pessoas/empresas que desenvolvem o software e entre essas mesmas pessoas/empresas e os utilizadores do software. Para descrever e especificar o software, esta framework utiliza o formato *Deployable Software Description* (`DSD`). Esta descrição é interpretada por agentes que efectuam a instalação e configuração do software.

Uma outra framework de deployment baseada em agentes móveis é descrita em [SJ02]. Esta framework, TACOMA, contém repositórios onde estão armazenados as últimas ver-

sões dos pacotes de software. Estes pacotes podem ser utilizados para efectuar actualizações ou novas instalações nos hosts que estão registados no serviço de repositórios.

Em [HMA06] é apresentada uma abordagem para o planeamento do deployment. Esta abordagem, baseada em grafos, é independente de qualquer tecnologia de componentes, elaborando o plano de deployment das comunicações solicitadas pelos componentes de uma aplicação e os recursos disponíveis nos host onde irão ser instaladas as aplicações.

Para que esse planeamento possa ser efectuado é necessário especificar qual a aplicação a ser instalada, o ambiente onde se pretende instalá-la e ainda especificar condições, impostas pelo utilizador, para a instalação da aplicação. Um exemplo de uma condição que pode ser imposta pelo utilizador é o nível de segurança nas comunicações.

No caso de [HW04] é apresentada uma framework para gestão de reconfigurações dinâmicas. Com esta framework é possível observar os custos, de uma mudança em runtime. Um conjunto de algoritmos de reconfiguração estão presentes na framework, podendo o utilizador escolher qual utilizar, baseando-se em análise comparativas dos mesmos.

A framework é constituída por três camadas: *Change Driver*, *Reconfiguration Manager* e *Application*, como pode ser observado na Figura 3.8. Os pedidos de reconfiguração são submetidos para a camada *Change Driver*, sob a forma de um script escrito em OpenRecML, que é uma linguagem de configuração baseada em XML.

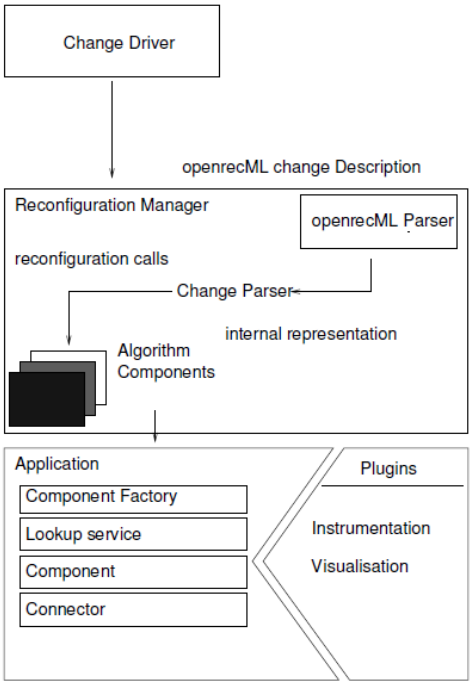


Figura 3.8: Arquitectura da Framework OpenRec [HW04]

Um ambiente para especificar, manipular, visualizar e estimar arquitecturas de de-

ployment para sistemas distribuídos é apresentado em [MRMBM04]. Com este ambiente é possível explorar as várias possibilidades de deployment para um determinado sistema permitindo desta forma perceber quais as arquitecturas que levarão a uma maior disponibilidade do sistema em causa.

Torna-se, também, exequível avaliar a sensibilidade a mudanças em parâmetros específicos, designadamente, confiabilidade de um determinado link na rede, e também às condições impostas no deployment, por exemplo, a necessidade de dois componentes terem de ficar em host diferentes.

A. Akkerman et all, em [ATK05], apresentam uma infra-estrutura para o deployment automático de aplicações JEE. Problemas como a especificação da conectividade entre os componentes e os seus efeitos na instalação e configuração dos mesmos, bem como as dependências que os componentes podem ter a nível dos serviços disponibilizados pelo servidor aplicacional, são abordados.

Como podemos constatar observando a Figura 3.9, a infra-estrutura é constituída por uma rede que detém múltiplos servidores aplicacionais. Cada um dos servidores aplicacionais contem um *Agent Service*, que comunicam com uma instância do *Replication Management Service* que corre em um nodo da rede da infra-estrutura.

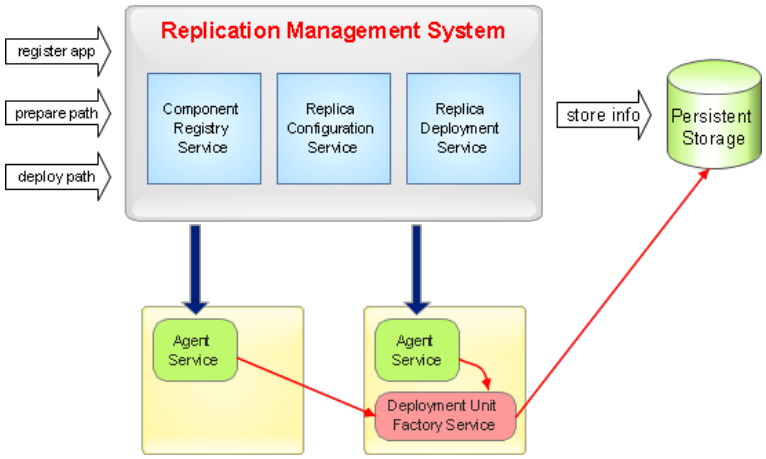


Figura 3.9: Arquitectura da infra-estrutura em [ATK05]

Em [RAC+02] é apresentado um protótipo de uma framework designada *Bean Automatic Reconfiguration frameworK* (BARK), que foi concebido com o intuito de facilitar a gestão e a automatização de todas as actividades do ciclo de vida do deployment de um EJB. Esta framework permite o download de pacotes e instalação dos mesmos no EJB container, sendo, ainda, possível a edição dos descriptores desses pacotes, para, deste modo, ter um maior controlo sobre o sistema. Além disso, é possível executar, não só deployments individuais, mas também scripts que efectuam múltiplos deployments. Estes

scripts são ficheiros XML que definem as sequências dos comandos a executar pelo BARK.

Na Figura 3.10 podemos observar as entidades que compõem a arquitectura da BARK. O *Application Server Module* (ASM) auxilia o servidor aplicacional EJB a gerir os componentes. É responsável por todo o processamento que esteja relacionado com o ciclo de vida do deployment. O *Repository* é uma storage que armazena os pacotes de software. É aqui que os produtores de software colocam as novas versões dos seus componentes. Já o *Workbench* disponibiliza uma ferramenta para controlar os ASMs.

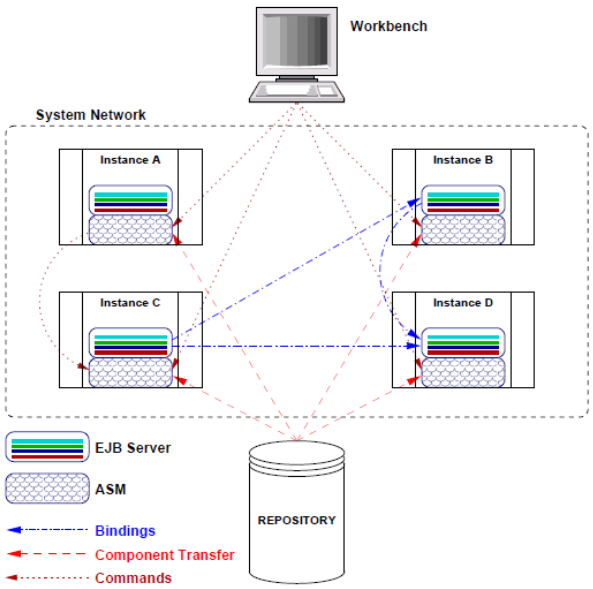


Figura 3.10: Arquitectura da Framework BARK [RAC+02]

Uma framework para testar um componente quando este é instalado é apresentado em [BP03], e tem como objectivo aumentar a fiabilidade que se pode ter nele. Esta framework permite aos utilizadores especificarem e documentarem uma bateria de testes que depois pode ser utilizada, num ambiente também disponibilizado.

L. Gayard et all, em [GAG+06], propõem uma ferramenta, CosmosLoader, que automatiza o processo de construir/unir uma aplicação em runtime, através de class loaders alteradas para o efeito. Com esta ferramenta é possível instanciar, de forma automática, os componentes e liga-los para formar uma aplicação em runtime.

A CosmosLoader recebe um ficheiro XML com a descrição da configuração dos componentes e ao interpretá-la liga e instância os componentes de acordo com a descrição lida. Com esta descrição é possível garantir que a aplicação final é composta pelos componentes e respectiva versões correctas.

Uma formalização das dependências do deployment é apresentado em [BD07]. Esta formalização tem como objectivo garantir a instalação e desinstalação de componentes

sem riscos para o sistema.

Outra framework baseada em métodos formais é apresentada em [LS06]. Esta fornece formalismos para quase todas as actividades do processo de deployment de software, designadamente, instalação e reconfiguração do sistema.

No caso de [BD06] são apresentados modelos de deployment, uma arquitectura e ainda a formalização das dependências de instalação.

Em [HWM+04] é apresentada uma abordagem para o deployment e re-deployment de serviços, baseada em modelos de arquitectura de software.

A arquitectura de software descreve a estrutura de um sistema com componentes, conectores e restrições [SG96]. Esta descrição funciona como uma ponte entre os requisitos e a implementação do software. De modo a efectuar a descrição do sistema é proposta a utilização da ABC/ADL, uma *Architecture Description Language* (`ADL`) com suporte para composição de componentes. A pessoa responsável pelo deployment tem de submeter as informações necessárias, quer da aplicação, quer dos servidores, de modo a que a instalação possa ser realizada com sucesso.

Como prova de conceito foi elaborada uma ferramenta, CADTool, que auxilia no processo de deployment, disponibilizando funcionalidades para:

- Visualizar os modelos de arquitectura de software.
- Visualizar os servidores e as suas capacidades no momento, especificamente, utilização do processador e da memória.
- Drag-and-drop de componentes.
- Cálculo automático de métricas como tempo de resposta e *throughput*.

Caso a descrição da arquitectura não seja disponibilizada, a ferramenta em questão pode automaticamente construir a arquitectura a partir dos componentes. Por conseguinte, a arquitectura gerada vai estar desprovida de informações derivadas da fase de desenvolvimento.

É de salientar, ainda, que foi utilizado como servidor aplicacional JEE o PKUAS [MH04].

3.3 Monitorização e gestão de Aplicações

Com o aumento da complexidade dos sistemas e das infra-estruturas onde assentam, é de extrema importância ter mecanismos que permitam obter dados sobre o seu funcionamento, através dos quais é possível evitar determinados problemas. Com a temática da

monitorização a ter cada vez mais importância, são várias as ferramentas existentes de apoio à monitorização e gestão de aplicações.

O Nagios [Nag] é talvez das ferramentas de monitorização de sistemas mais conhecidas. Este permite a monitorização dos serviços de rede e dos recursos que estão a ser utilizados em cada servidor. De notar que toda a monitorização remota pode ser efectuada através de *Secure Shell* (SSH) ou *Secure Sockets Layer* (SSL), tornando-a segura.

A partir do Nagios podemos obter vários relatórios de disponibilidade e configurar acções correctivas para os problemas ocorridos no sistema, podendo alertar os administradores quando ocorre um através de diferentes formas, por exemplo, email ou SMS.

Devido à sua enorme capacidade e o facto de ser open-source contribui para que existam muitas outras ferramentas de monitorização baseadas nele.

O Zabbix [SIA] é um exemplo de uma ferramenta que possui funcionalidades herdadas do Nagios e do Cacti, esta é utilizada para monitorizar a performance e a disponibilidade de uma infra-estrutura. Tal como o Nagios, pode obter várias informações sobre os servidores, nomeadamente, espaço disponível em disco, memória RAM utilizada, entre outros. Para a obtenção destes dados, são utilizados agentes que são instalados em cada servidor.

Todas as configurações e os dados adquiridos através da monitorização são armazenados numa base de dados centralizada.

No caso do Zenoss [Zen], é uma ferramenta, também open-source, desenvolvida em Python/Zope, que permite a monitorização de servidores, aplicações e da rede envolvente de uma infra-estrutura, testando, de entre outras, a sua disponibilidade e performance. Possui uma arquitectura de três camadas, sendo a primeira referente à apresentação dos dados, a segunda tem como função recolher toda as informações requeridas, e finalmente, a terceira que tem como função armazenar os dados em formato persistente. Esta ferramenta também possui funcionalidades que permitem alertar o administrador do sistema.

InfraRED [Ltd] é uma ferramenta de monitorização da performance de uma aplicação JEE. Esta ferramenta colecciona várias métricas e disponibiliza os resultados para análise. Para efectuar esta monitorização é utilizado *Aspect Orient Programming* (AOP) [JN04, GL03].

A InfraRED tem, ainda, a capacidade de alertar para determinados problemas de performance, servindo como guia para encontrar a possível causa para os mesmos.

A empresa Hyperic [Hyp], recentemente adquirida pela SpringSource[1], apresenta um sistema para gestão e monitorização de uma infra-estrutura j2ee, que possui suporte quer para servidores aplicacionais comerciais, quer para soluções open-source.

[1] http://www.springsource.com/

A ferramenta possibilita a monitorização de diversos factores nomeadamente o consumo de processador, os recursos a ser utilizados pela *Java Virtual Machine* (JVM) e o desempenho da Base de Dados, tendo ainda a capacidade de alertar caso ocorra algum evento fora do normal. Uma das vantagens desta ferramenta é a possibilidade desta conseguir descobrir o hardware e o software utilizado na infra-estrutura.

Os elementos chave da sua arquitectura são o servidor HQ que é o gestor central de toda a monitorização e gestão, e o agente HQ que reúne os valores de métricas como disponibilidade, utilização, desempenho, entre outras, da máquina onde foi instalado e envia essas mesmas informações para o servidor HQ, como podemos observar na Figura 3.11. Assim, torna-se necessário instalar um agente HQ em cada máquina que queremos gerir e monitorizar.

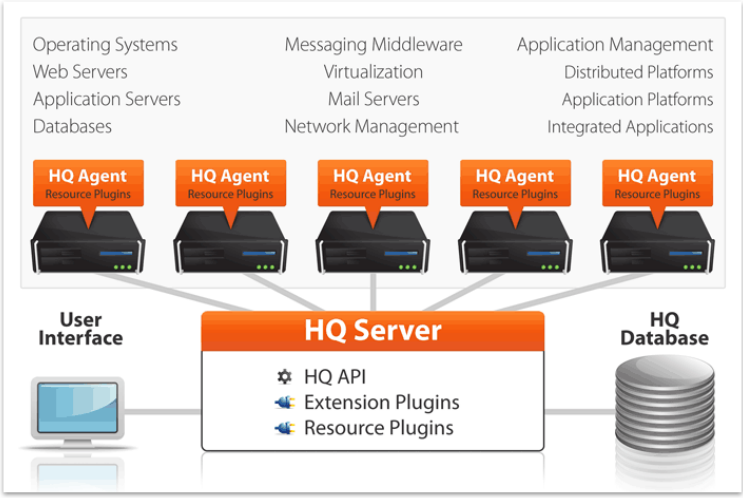


Figura 3.11: Arquitectura da Ferramenta Hyperic HQ [Hyp]

Uma das desvantagens desta ferramenta, até ao momento da escrita desta dissertação e segundo a informação disponibilizada no site da mesma, é o facto de não ser compatível com nenhum servidor aplicacional com certificação por parte da Sun para Java EE 1.5.

O Jopr [Hat] é uma solução open-source para projecto JBoss que permite, não só, a monitorização da disponibilidade e da performance dos servidores, facultando, ainda, um sistema de alerta que permite avisar os administradores, mas, também, a gestão de vários períodos do ciclo de vida de uma aplicação. Além disso, permite efectuar o deployment em servidores aplicacionais JBoss.

Jiang Guo *et all* em [GLP06] propõem uma ferramenta para analisar e monitorizar aplicações JEE. Através dos diagramas de sequência apresentados pela ferramenta será possível detectar a base de um problema de performance. É ainda possível utilizar esta ferramenta para fazer supervisionar a performance, não só de aplicações JEE, mas também

de aplicações desenvolvidas em Java puro.

Esta ferramenta permite especificar a informação que é retirada em runtime, designadamente, qual o tipo de entidades que devem ser monitorizadas, qual a parte do código dessas entidades a ser monitorizada, qual o tipo de informação a retirar, entre outras. Para retirar os dados pretendidos, agentes de monitorização estão espalhados pelas várias camadas do sistema JEE, como se pode observar na Figura 3.12.

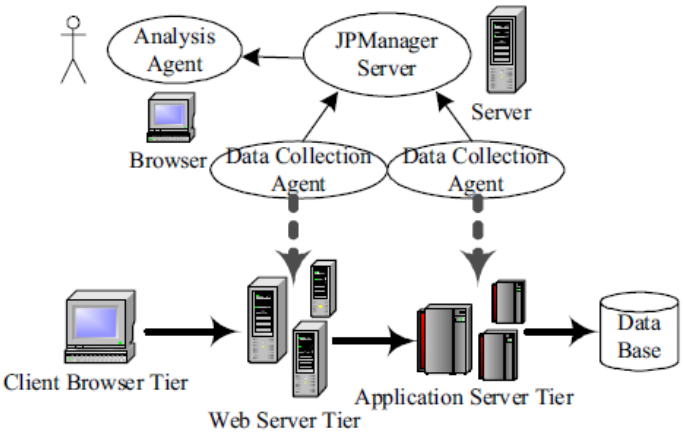


Figura 3.12: Arquitectura da Ferramenta JPManager [GLP06]

Uma framework para supervisionar automaticamente a performance de sistemas baseados em componentes, com especial atenção a sistemas JEE, é apresentada em [DMM04].

Esta framework é constituída por dois módulos principais, um para monitorização e diagnóstico, sendo responsável por adquirir, em run-time, informação sobre a performance dos componentes, assim como do ambiente onde a aplicação está inserida (*monitoring and diagnosis*). Tendo, ainda, a função de analisar a informação recolhida. O segundo módulo é responsável pela resolução dos problemas de desempenho encontrados pelo primeiro módulo (*application adaptation*).

A solução proposta pelos autores para a resolução dos problemas de desempenho encontrados, baseia-se na utilização de múltiplas implementações dos componentes, cada uma optimizada para uma situação específica. O segundo módulo quando recebe indicações do problema faz uma análise da situação e activa o componente que mais se adequa à mesma.

Uma pesquisa sobre sistemas de gestão de performance de aplicações JEE foi descrita em [GLP04]. Aqui são apresentadas algumas ferramentas comerciais e ainda são discutidos quais os requisitos que uma ferramenta deste género deve possuir.

Segundo os autores, é necessário que estes sistemas possuam suporte para monitorização e análise. Métricas como o tempo de resposta e o *throughput* são essenciais para

se obter informações sobre a performance de uma determinada aplicação. Os resultados apresentados por estes sistemas devem permitir que os problemas sejam detectados rapidamente e de forma precisa.

As ferramentas analisadas nesta pesquisa foram:

- DiagnoSys;
- Dirig Application Performance Platform (APP);
- Quest Central;
- VERITAS Indepth for J2EE;
- JDBInsight.

Jun Li e Keith Moore em [LM07] apresentam uma framework de apoio à elaboração de testes unitários aos componentes de sistemas distribuídos baseados em componentes, como são os casos de sistemas desenvolvidos utilizando CORBA ou JEE/RMI. Esta framework automaticamente equipa a aplicação, ampliando o compilador de *Interface Definition Language* (`IDL`), de forma a inserir o código referente à monitorização nos módulos de stub e skeleton gerados. Com os resultados obtidos é possível construir um grafo que reflecte as relações de comunicação ao nível dos componentes.

Capítulo 4

Infra-estrutura

As infra-estruturas onde assentam os serviços Web assumem uma importância vital nos dias de hoje. Com a Internet a impor-se, cada vez mais, como um canal de comunicação para o mundo e uma óptima aliada do marketing, muitos serviços estão a ser disponibilizados através dela. Um estudo da Universidade de Berkeley na California[1] sobre a quantidade de informação gerada mundialmente, revelou que, em 2003, cada pessoa gerou cerca de 800 MBytes de informação por ano. Actualmente, com a facilidade de acesso e uma maior velocidade nas ligações, estes números tendem a aumentar de forma considerável.

Com esta abertura a um *"mercado sem geografia"* são vários os casos de plataformas Web a ficarem inacessíveis, devido ao volume de carga a que foram sujeitas. Em alguns casos, esta situação não tem qualquer tipo de consequência directa para a entidade responsável pela plataforma. Noutros, porém, esta situação pode levar à perda de utilizadores/clientes e, mesmo, a perdas financeiras.

Se casos há onde o problema está na própria aplicação, muitos deles acontecem devido ao deficiente planeamento da infra-estrutura. Assim, de modo a suportar a complexidade dos requisitos das aplicações e o eventual volume de carga que podem gerar, é fundamental um planeamento cuidado da infra-estrutura onde as mesmas vão assentar.

São vários os exemplos de sistemas que ficam indisponíveis devido a um elevado volume de pedidos. Um dos mais recentes é o Monopoly City Streets[2] que não dia do lançamento ficou indisponível para frustração de muitos utilizadores. A Hasbro juntamente com a Google, responsáveis pelo jogo, viram-se obrigadas a re-estruturar o sistema para aguentar um maior volume de dados, bem como reiniciar todos os dados do jogo, para que todos os jogadores tivessem as mesmas oportunidades.

[1] http://www2.sims.berkeley.edu/research/projects/how-much-info-2003/
[2] http://www.monopolycitystreets.com/

Outro caso, relativamente recente, é o Twitter[3], uma mistura de rede social com *microblogging*. A rápida popularização deste sistema foi de tal forma acentuada que este não aguentou o volume de pedidos ficando, em várias ocasiões, indisponível.

A capacidade de estimar a carga que uma infra-estrutura deve suportar pode ser uma tarefa muito delicada, sendo necessário não só um conhecimento aprofundado das aplicações que irá suportar, mas, também, do tipo de público alvo e respectivo comportamento.

Conceitos como escalabilidade, disponibilidade, segurança e integridade dos dados são fundamentais para qualquer plataforma Web. Uma infra-estrutura deve ser robusta, sendo capaz de continuar em funcionamento mesmo na ocorrência de uma falha no sistema, por exemplo, devido a uma avaria no disco ou a um corte de energia. Além disso, deve ser suficientemente flexível para se poder adaptar a uma maior carga ou a uma mudança nas políticas de negócio sem grandes custos.

4.1 Confiabilidade

Dada à inevitável presença e ocorrência de falhas, é impossível classificar os sistemas computacionais 100% correctos.

Um sistema pode avariar pelas mais diversas razões, por exemplo, porque não cumpre a sua especificação ou porque esta não descreve adequadamente a sua funcionalidade, ou, até, devido a um erro no próprio desenho da solução.

Tendo em conta que os sistemas computacionais assumem, cada vez mais, um papel central em diversas funções críticas, e reconhecendo que estes não estão livres de eventuais avarias, torna-se necessário aumentar os seus requisitos de confiabilidade e segurança, de modo a poderem suportar os efeitos de falhas, avarias acidentais e acessos não autorizados.

A confiabilidade é a qualidade de um sistema que nos permite confiar, justificadamente, no serviço que fornece. Ou seja, é o modo de poder quantificar, prever, prevenir, tolerar o efeito de perturbações ao seu comportamento.

O nível de confiabilidade de um sistema caracteriza-se por uma série de atributos [AA04]:

- Fiabilidade (*reliability*), consiste na probabilidade de um sistema apresentar um serviço correcto, de forma contínua, durante um determinado intervalo de tempo.
- Confidencialidade (*confidenciality*), diz respeito à probabilidade de, num determinado intervalo de tempo, não ser divulgada informação indevida.

[3] http://twitter.com/

- Disponibilidade (*availability*), é a probabilidade de um sistema apresentar um serviço correcto, num determinado instante de tempo.
- Integridade (*integrity*), refere-se à probabilidade de não ocorrerem alterações impróprias de estado, num determinado intervalo de tempo.
- Segurança (*safety*), consiste na probabilidade do sistema não apresentar uma avaria que tenha consequências negativas para os seus utilizadores, num determinado intervalo de tempo.
- Capacidade de manutenção (*manutenability*), representa a capacidade de um sistema poder ser reparado, regressando ao estado de serviço correcto, num determinado intervalo de tempo, após a detecção de uma avaria.

Claro está que, dependendo do tipo de funcionalidades do sistema, os vários atributos acima mencionados podem ter diferentes graus de importância. Um atributo que está sempre presente é a disponibilidade, na medida em que é sempre necessária, se bem que em grau variável. Na secção seguinte abordamos este assunto de forma mais exaustiva.

Na Figura 4.1 está representada uma taxonomia de confiabilidade e segurança, onde podemos observar os atributos que caracterizam a confiabilidade, as ameaças à mesma e os meios que possibilitam aumentá-la.

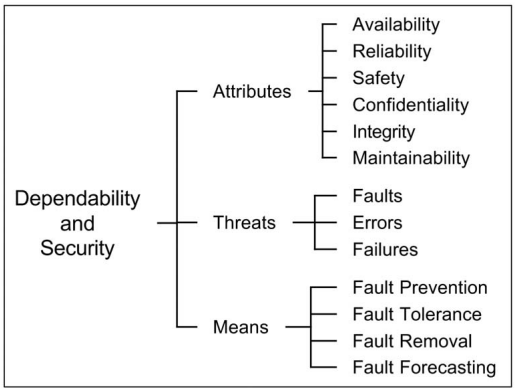


Figura 4.1: Confiabilidade e Segurança [AA04]

Como podemos observar, os meios desenvolvidos para atingir os vários atributos de confiabilidade e segurança podem ser agrupados em quatro grandes categorias [AA04]. Prevenção de falhas (*fault prevention*), tolerância a falhas (*fault tolerance*), remoção de falhas (*fault removal*) e previsão de falhas (*fault forecasting*).

A prevenção de falhas tem por objectivo prevenir a ocorrência ou introdução de falhas. A tolerância a falhas visa fornecer um serviço de forma correcta, mesmo na presença de

falhas. A remoção de falhas, como o próprio nome indica, tem como objectivo reduzir o número e/ou gravidade de falhas e a previsão de falhas estima o número presente, ocorrência futura e consequências prováveis das falhas.

Apesar de não ser possível garantir, na totalidade, que estas situações indesejáveis não ocorram, é, no entanto, prioritário assegurar que as suas consequências se mantenham dentro dos limites aceitáveis.

4.2 Disponibilidade

Graças à expansão do mundo digital, é possível encontrar hoje um crescente número de sistemas baseados em aplicações Web, que necessitam de estar sempre, ou quase sempre, disponíveis. Portanto, é essencial que as arquitecturas que os suportam garantam a sua disponibilidade.

A disponibilidade de um sistema pode ser definida como a probabilidade de este estar operacional num determinado instante de tempo [BP75] e baseia-se no correcto funcionamento dos vários sistemas presentes na arquitectura do sistema.

Para calcularmos a disponibilidade de um sistema temos de ter em conta dois factores: o primeiro é o tempo médio entre falhas (`MTBF`) e o segundo é o tempo médio necessário para recuperar/reparar o sistema (`MTTR`).

Assim sendo, a fórmula para calcular a disponibilidade de um sistema é [MS03]:

$$A = \frac{MTBF}{MTBF + MTTR}$$

Pela fórmula acima transcrita, chegamos a duas conclusões [MS03]:

- À medida que o `MTTR` se aproxima do 0, a disponibilidade do sistema aumenta para os 100%
- Quanto maior for o `MTBF`, menos impacto tem o `MTTR` no cálculo da disponibilidade do sistema

No entanto, é de salientar que a disponibilidade de um sistema não é um assunto tão linear como a fórmula o apresenta. Um sistema é composto por vários componentes e cada um deles tem um determinado `MTBF` e `MTTR`. Todavia, um componente que tenha um `MTBF` de 5000,000 horas não significa que irá falhar passado 57 anos de uso, podendo, falhar a qualquer momento [WWS04]. Assim, esta fórmula dará apenas valores aproximados, os quais podem ser usados como referência.

São várias as causas - umas mais frequentes que outras - que podem levar um sistema a ficar indisponível [MS03], nomeadamente, falhas no Software, Hardware, erro Humano, desastres naturais, entre outras, como podemos observar na Figura 4.2.

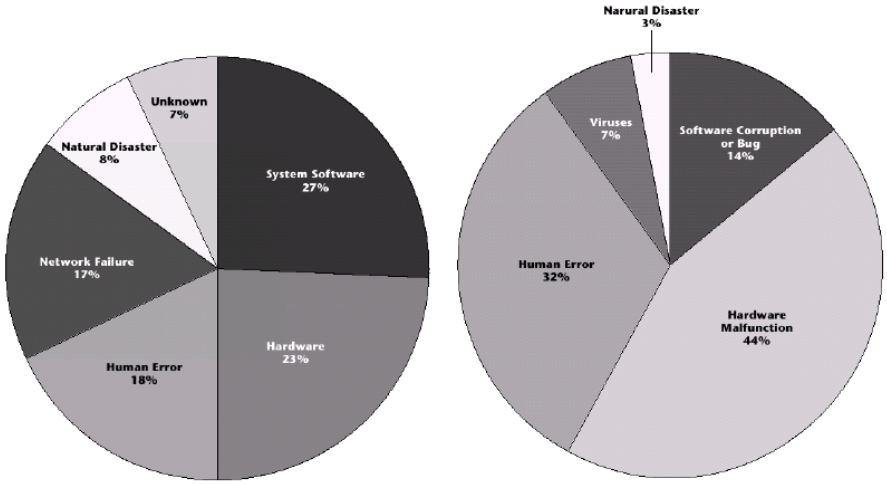


Figura 4.2: Causas da Indisponibilidade [MS03]

Os erros de Software manifestam-se, muitas vezes, como bloqueios, término inesperado ou respostas erradas. Estes erros podem ser o reflexo de uma má implementação, ou, então, estar relacionados com a falta de recursos da infra-estrutura onde este se encontra instalado.

Já os erros de Hardware ficam a dever-se, por exemplo, a erros na RAM, cabos danificados ou à deficiente refrigeração interna, causada pelo mau funcionamento das ventoinhas. Outro factor que contribui para a ocorrência deste tipo de erros são os Discos.

No entanto, nem todos os erros estão relacionados com o Software ou Hardware. Uma percentagem de erros deve-se à intervenção humana e ao ambiente que rodeia todo o sistema. Assim, falhas de energia, desastres naturais, ou, mesmo, um ataque malicioso por parte de um utilizador, podem ser a causa da indisponibilidade de um sistema.

Como a indisponibilidade de um sistema pode causar danos directos ou indirectos, torna-se imprescindível saber a partir de que ponto é garantida a disponibilidade de um sistema necessária a cumprir os requisitos para os quais foi planeado.

Uma forma de quantificar e medir a disponibilidade de um sistema é através da contagem do número de noves. Esta medida traduz-se na percentagem de tempo que um sistema está activo e responde correctamente a pedidos (uptime, em oposição a downtime).

A Tabela 4.1 relaciona a percentagem de Uptime com o Downtime.

Tabela 4.1: Níveis de Alta Disponibilidade

Uptime (%)	Downtime / dia	Downtime / mês	Downtime / ano
95	72.00 minutos	36 horas	18.26 dias
99	14.40 minutos	7 horas	3.65 dias
99.9	86.40 segundos	43 minutos	8.77 horas
99.99	8.64 segundos	4 minutos	52.60 minutos
99.999	0.86 segundos	26 segundos	5.26 minutos

De relevar que a alta disponibilidade tem um custo, se assim não fosse todos os sistemas estariam 99,999% das vezes disponíveis. Portanto, quanto maior for a necessidade de disponibilidade, maior é o custo para a atingir, uma vez que é necessário proteger todo o sistema contra uma maior variedade de falhas.

Torna-se, então, prioritário identificar qual é a taxa de disponibilidade esperada pela aplicação, e, posteriormente procurar identificar quais as técnicas, tecnologias e modelos que podem dar a flexibilidade necessária para que a aplicação mantenha a taxa de disponibilidade desejada.

4.3 Escalabilidade

O crescimento do número de serviços disponibilizados e, principalmente, do número de utilizadores desses serviços fez com que a escalabilidade dos servidores aplicacionais e das aplicações se tornasse uma das principais preocupações no que toca a questões relacionadas com a arquitectura dos sistemas.

A Escalabilidade é um requisito não funcional de um sistema que mede a sua flexibilidade em se adaptar rapidamente a mudanças de intensidade de utilização, continuando a cumprir os objectivos de negócio pré-definidos, nomeadamente, o tempo de resposta do sistema.

Existem vários modos para medir a escalabilidade. Normalmente uma aplicação e-business deve ter em conta [BH04]:

- Capacidade de processamento : taxa a que um determinado processo pode ser executado.
- Capacidade de informação : quantidade de armazenamento num determinado nível do sistema. Por exemplo, capacidade do disco necessária pelo sistema.
- Conectividade : número de acessos efectivos que podem ser feitos ao sistema.

É de salientar que a escalabilidade não depende só da forma e das técnicas utilizadas aquando do desenvolvimento do software, estando igualmente relacionada com a arquitectura da infra-estrutura onde o sistema foi instalado e configurado. Por mais que um sistema seja escalável, este é limitado pela capacidade do Hardware onde assenta. Desta forma, são dois os métodos para adicionar recursos ao sistema [BG03] :

- Escalabilidade vertical (*Scale-up*) - Consiste em melhorar os componentes de hardware da actual solução. Implica acrescentar recursos a um único nó do sistema, mais precisamente, aumentar o número de processadores ou memória. É a forma mais simples de aumentar a escalabilidade de um sistema.

  Uma das desvantagens desta estratégia diz respeito ao eventual custo inerente ao aumento de capacidade do nó, e à possível necessidade de desligar o sistema para fazer o upgrade.

- Escalabilidade horizontal (*Scale-out*) - Implica adicionar mais nós a um sistema. Um exemplo pode ser adicionar mais um servidor aplicacional ao sistema.

  Com o aumento do número de nós, aumenta o grau de complexidade de gestão dos mesmos, assim como, a preocupação em manter um estado coerente entre os nós.

Na relação entre custos e benefícios, é mais comum que a escalabilidade horizontal ganhe, não só devido aos custos reais, mas também devido aos benefícios de segunda ordem, como a melhoria da disponibilidade.

Com as crescentes mudanças no mundo digital, e com condições de negócio a mudarem constantemente, é essencial a capacidade de adaptação ao crescimento. Assim a escalabilidade é, cada vez mais, um requisito indispensável para as infra-estruturas que servem de base para as aplicações Web.

4.4 Arquitectura Física de uma Infra-Estrutura Java EE

Uma Infra-Estrutura capaz de acolher aplicações Web Java EE divide-se, normalmente, em três camadas (*tiers*) distintas. A primeira camada é composta pelos Servidores Web. Da segunda fazem parte os servidores aplicacionais JEE que, como mencionado no Capítulo 2, são necessários para executar as aplicações JEE. Portanto, inevitavelmente estes têm de estar presentes. A última camada é constituída por *Sistemas de Gestão de Bases de Dados* (`SGBD`).

A existência de uma camada com Servidores Web permite a implementação de mecanismos de distribuição de carga, caching ou clustering. Esta camada física permite, ainda, a apresentação de conteúdo estático, quer seja uma página HTML ou uma imagem, de uma forma mais rápida, delegando os pedidos que necessitam de uma resposta gerada dinamicamente para o nível imediatamente abaixo, ou seja, para os servidores aplicacionais.

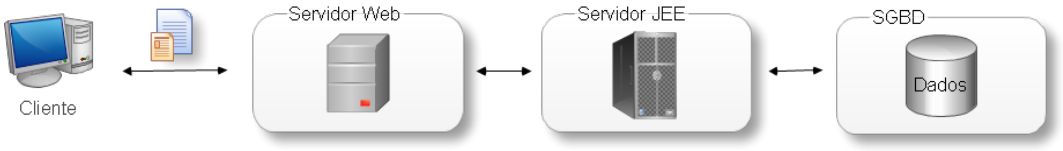


Figura 4.3: Infra-estrutura Simples

A Figura 4.3 representa uma infra-estrutura simples para uma aplicação Web JEE, que contém todos as camadas físicas referidas.

Apesar de ser válida, esta infra-estrutura não tem em conta alguns factores enumerados acima, nomeadamente a escalabilidade e disponibilidade, uma vez que, a título de exemplo, se houver uma falha no servidor Web, todo o sistema fica indisponível.

Ainda relativamente à Figura 4.3, podemos observar que esta não tem qualquer tipo de protecção adicional, designadamente, firewalls, o que constitui um risco para todo o sistema. Com um número crescente de informação crítica disponível online, é essencial garantir que esta seja somente acedida por utilizadores com permissões para tal.

No passado, a questão segurança da informação era muito mais simples, pois grande parte da informação encontrava-se sobre a forma de inúmeros documentos físicos que podiam ser guardados num local seguro. Porém, hoje a maioria da informação encontra-se digitalizada e é um risco deixá-la sem protecção alguma, uma vez que a maioria dos computadores possui uma ligação à internet.

> "It's easy to run a secure computer system. You merely have to disconnect all dial-up connections and permit only direct-wired terminals, put the machine and its terminals in a shielded room, and post a guard at the door."
>
> F.T. Grampp e R.H. Morris

Um estudo conduzido pelo professor Doutor Michel Cukier[4], realizado pela Universidade de Maryland, demonstrou que ocorre um ataque, por parte de agentes maliciosos, em cada 39 segundos, o que vem reforçar, ainda mais, a necessidade de construir uma infra-estrutura segura.

[4]http://www.enre.umd.edu/faculty/cukier.htm

Depois da análise feita à infra-estrutura apresentada podemos concluir que o nível de confiabilidade que lhe podemos atribuir é baixo, uma vez que, a título de exemplo, uma falha numa das camadas vai impedir o correcto funcionamento do sistema. Deste modo, podemos concluir que a infra-estrutura apresentada apenas deve ser utilizada num limitado número de sistemas computacionais, designadamente sistemas que não desempenhem funções críticas.

Na Figura 4.4 podemos observar uma infra-estrutura mais complexa, comparativamente à apresentada anteriormente. Nesta infra-estrutura é evidente a existência de redundância em todas as suas camadas. Esta é a chave para podermos obter uma infra-estrutura tolerante a faltas e, ao mesmo tempo, escalável [Ash04].

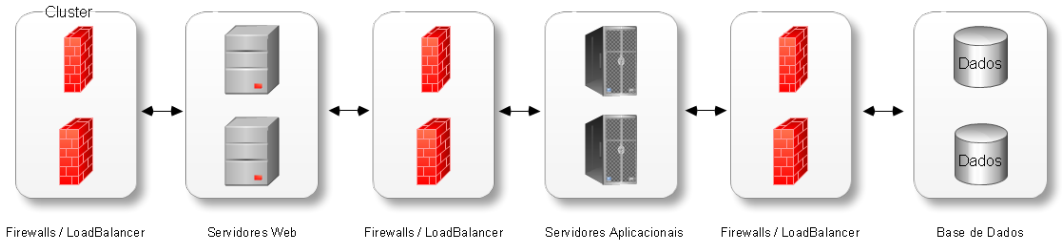


Figura 4.4: Infra-estrutura Base

No entanto, é de notar que a redundância dos componentes acarreta complexidade e custos ao sistema, implicando a coordenação entre as réplicas do componente, sendo esta uma tarefa, de uma forma geral, bastante complexa. O desempenho do sistema, em determinados casos, também pode ser afectado, uma vez que a coordenação entre componentes redundantes introduz complexidade no processamento do sistema.

Outro aspecto a salientar relativamente a esta infra-estrutura é a presença de *firewalls*, que visam aumentar a segurança da mesma. No entanto, a segurança adquirida com a utilização das firewalls apenas se torna eficaz se as *firewalls* estiverem bem configuradas.

Nas duas infra-estruturas apresentadas não está representada a estrutura da rede que permite a comunicação entre os servidores das mesmas, e a sua comunicação com o exterior. Esta estrutura é, igualmente, de extrema importância, pois sem a comunicação com os servidores nada seria possível. No entanto, todo o seu planeamento foge do âmbito da tese, partindo-se do pressuposto que esta estrutura já se encontra construída.

Como foi possível constatar anteriormente[5] não basta que a aplicação seja bem desenvolvida e cumpra todos os requisitos para os quais foi projectada, para corresponder à expectativa dos utilizadores. Efectivamente, há que ter em consideração todos os factores acima mencionados, para, desta forma, se obter uma infra-estrutura capaz de corresponder

[5] Ver secção 3.1

aos requisitos da aplicação e às expectativas dos utilizadores.

4.5 Escalabilidade e Disponibilidade da Camada de Dados

Nos últimos anos o crescimento de aplicações Web com um grande volume de dados tem-se acentuado, sendo o acesso a esses vital para as aplicações [Joh02, Hig01]. Estes acessos têm de ser feitos de forma eficiente, sendo necessário garantir a integridade e consistência dos dados [Joh02]. Esta celeridade de resposta é um dos factores chave que podem levar ao sucesso ou insucesso de uma aplicação Web [Hig01].

A performance de uma aplicação JEE pode ser deteriorada devido ao volume de dados que uma base de dados aguenta [CM03]. Torna-se, por conseguinte, necessário arranjar soluções que possam garantir a escalabilidade e disponibilidade da Camada de Dados.

Uma solução para a obtenção da escalabilidade e disponibilidade necessária passaria por colocar a camada de dados numa máquina *Symmetric Multiprocessor* (`SMP`) [CM03, JMZ04]. Uma máquina deste género é normalmente constituída por vários processadores iguais, onde a memória e o disco são partilhados. Contudo, o problema desta solução reside nos elevados custos que a mesma comporta.

Além do preço, este sistema tem outras desvantagens, designadamente o facto de conter um single point of failure no sistema operativo, aquando da manutenção ou upgrade do hardware é, normalmente, necessário parar toda a máquina, o que torna indisponíveis os serviços alojados por esta. Muitas das vezes, é também necessário fazer alterações à própria aplicação, de modo a esta poder tirar total partido da capacidade da infra-estrutura.

Em detrimento das máquinas `SMP`, o clustering começou a ganhar maior expressividade graças a uma melhor relação entre a performance obtida e o preço da infra-estrutura [JMZ04]. Como foi possível constatar na secção anterior, é possível com clustering atingir os níveis de escalabilidade e disponibilidade desejados, ideia reforçada pelo estudo em [Chi01].

No que toca ao clustering de base de dados podemos ter duas topologias distintas [DKS]:

- *Shared-Nothing Clusters*
- *Shared-Stored Clusters*

Na topologia *Shared-Nothing* temos um grupo de servidores independentes, interligados, com memória e discos independentes, como pode ser observado na Figura 4.5, podendo cada um deles ter uma cópia integral da base de dados ou então esta estar particionada entre os diversos nodos [DKS, Mul02]. A propagação dos Updates pelas Base de Dados podem ser feitos pelo próprio servidor de Base de Dados ou então através de um middleware [SP09].

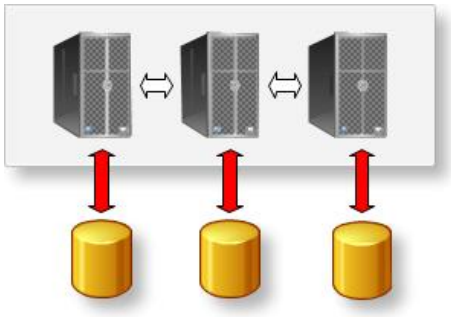


Figura 4.5: Topologia Shared-Nothing

São várias as propostas de soluções que utilizam esta topologia como pode ser observado em [SP09].

No caso da topologia Shared-Stored Clusters todos os nodos partilham o mesmo espaço de armazenamento [DKS, Mul02], como ilustra a Figura 4.6. Neste caso, torna-se necessário recorrer a técnicas de *Redundant Array of Independent Drives* (`RAID`) para obtermos redundância de dados [SP09].

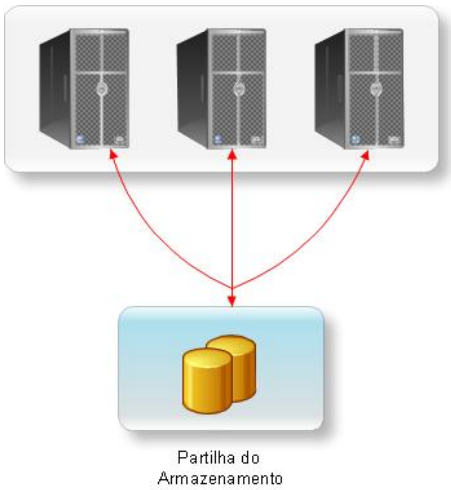


Figura 4.6: Topologia Shared-Store

Existem várias soluções que prometem a escalabilidade e disponibilidade da camada de dados necessária para que a resposta do sistema esteja de acordo com os parâmetros

exigidos. Na subsecção seguinte serão apresentadas algumas dessas soluções. De salientar que muitas das soluções apresentadas são tecnologias, uma vez que o estudo de técnicas de replicação de Base de Dados foge do teor desta dissertação.

4.5.1 Soluções para a Camada de Dados

Como referido anteriormente, o acesso contínuo e rápido aos dados tornam-se um meio de fazer a diferença entre as várias plataformas Web. Deste modo, várias soluções apareceram para garantir o desempenho esperado e, ao mesmo tempo, ser possível obter escalabilidade e disponibilidade.

Podemos dividir essas soluções em duas categorias distintas. Na primeira encontram-se as soluções específicas para um determinado Sistema de Gestão de Bases de Dados. Apesar de poderem tirar o máximo partido do `SGBD`, uma das desvantagens é o facto de serem restritas a um `SGBD` específico.

Já a segunda é constituída por soluções que formam uma camada intermédia que funciona como uma ponte entre a camada de negócio e a camada de Dados. Estas soluções têm como principal vantagens a capacidade de trabalharem com diferentes `SGBD`.

O Oracle Real Application Clusters [Incb] e IBM DB2 Integrated Cluster Environment [Che04] são duas soluções comerciais que utilizam uma Storage Area Network para assegurar tolerância a faltas e escalabilidade.

O MySQL Cluster [RT04] fornece uma arquitectura de clustering de bases de dados tolerante às falhas para implementar bases de dados críticas de alta disponibilidade. Esta solução é utilizada com uma topologia shared-nothing.

Associado à Base de Dados PostgreSQL existe o Slony-I [Wie05], onde a replicação é feita através de Triggers. No entanto uma desvantagem deste sistema é o facto de este não realizar a replicação da estrutura da Base de Dados, o que pode fazer como que haja incoerência entre as réplicas, no caso de haver updates à estrutura da Base de Dados e estas não serem propagadas.

E ainda, o Postgres-R [KA00] onde replicação é feita de maneira diferente do Slony-I, na medida em que, existe um gestor de replicação que envia as alterações efectuadas às outras réplicas.

Apesar das soluções acima explanadas garantirem escalabilidade da camada de dados, toda a infra-estrutura ficava presa ao respectivo Sistema de Gestão de Bases de Dados. Assim, uma outra forma de obtermos a escalabilidade e a disponibilidade desejadas na

camada de dados é utilizar um middleware para fazer a ligação entre a camada de negócio e a camada de dados.

O Sequoia [Inca], continuação do Projecto C-JDBC, é um middleware open-source, que utiliza a topologia share-nothing e oculta a complexidade do clustering à aplicação. Para a aplicação, é como se estivesse a ligar a uma base de dados [JMZ04]. Uma das vantagens desta solução é, precisamente, o facto de não ser necessário alterar código na aplicação para se ligar ao middleware.

Este middleware é compatível com todas as Base de Dados que possuem um driver `JDBC` e é feito inteiramente em Java aumentando, deste modo, a sua portabilidade.

Com características idênticas ao Sequoia temos o HA-JDBC [Fer], sendo este também feito em java e possuindo suporte para todas as Base de Dados com um driver `JDBC`. Algumas diferenças entre estas duas soluções são:

- O Sequoia possui Log de recuperação
- No HA-JDBC é possível, de forma automática, reactivar os nodos que ficaram inactivos. No Sequoia, este processo é manual
- No Sequoia há o suporte para base de dados heterogéneas, ou seja, é possível termos uma camada de dados constituída por, por exemplo, um motor de base de dados MySQL e outro PostgreSQL.
- No Sequoia existe suporte para *RAIDb-0*, *RAIDb-1* e *RAIDb-2*, enquanto que no HA-JDBC apenas o *Mirroring* é suportado
- O HA-JDBC tem um suporte total às várias versões do JDBC.
- O Sequoia disponibiliza, nativamente, pool de conexões.

No que toca ao SQL Relay [Fir], este consiste num middleware específico para sistemas Unix e Linux. Uma desvantagem em relação aos outros é a necessidade de se ter de alterar a aplicação original, para que esta comunique com middleware. Para isso são disponibilizadas várias `APIs` de acordo com a linguagem a ser utilizada. Tem suporte para *C*, *C++*, *Java*, *PHP*, *Ruby*, entre outras.

O projecto GORDA (Open Replication of DAtabases) [JPR+07], é um projecto de investigação europeu, financiado pela Fundação de Ciência e Tecnologia e pela Comissão Europeia, tem como propósito fomentar a investigação e o desenvolvimento na área da replicação de servidores de Base de Dados, promovendo a interoperabilidade dos Sistema

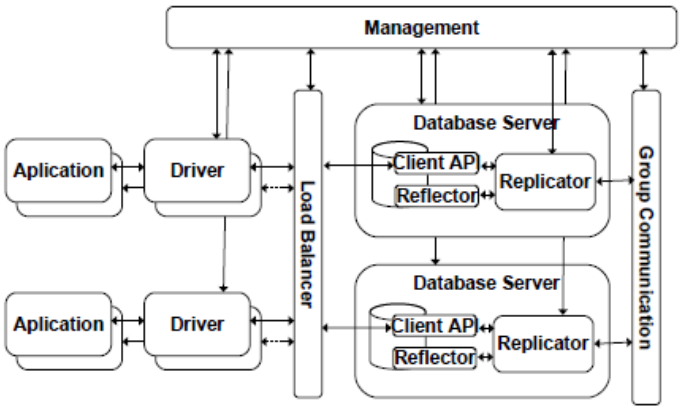


Figura 4.7: Arquitectura do Projecto GORDA [JPR+07]

de Gestão de Bases de Dados e dos protocolos de replicação definindo uma arquitectura e protocolos genéricos.

Como podemos observar na Figura 4.7, a arquitectura do sistema de gestão das base de dados replicadas é complementada por uma arquitectura de gestão genérica [JPR+07].

Através das várias técnicas para obter escalabilidade da camada de dados acima citados, é possível perceber que existem soluções específicas, quer para um determinado motor de base de dados ou, então, soluções mais genéricas e adaptáveis. Assim, torna-se pertinente desenvolver um estudo cuidado da situação para escolher o método mais adequado à mesma.

4.6 Comunicação com os Servidores

Com o grande aumento das aplicações Web, as infra-estruturas onde assentam tiveram que, inevitavelmente, crescer, de modo a responder as necessidades exigidas. Visto que os servidores que constituem a infra-estrutura normalmente não se encontrarem disponíveis localmente, torna-se necessário ter mecanismos que permitam o acesso remoto aos mesmos para, desta forma, ser construída e configurada a infra-estrutura.

Nas subsecções seguintes são apresentados alguns mecanismos que permitem a comunicação, local ou remota, entre computadores.

4.6.1 Telnet

Telnet é um protocolo cliente-servidor standard que permite a comunicação entre computadores ligados numa rede.

Este protocolo apoia-se sobre uma ligação TCP/IP e um *Network Virtual Terminal* (`NVT`), que define como a informação é enviada pela rede.

O cliente Telnet recebe a informação do utilizador, de seguida codifica-a para o formato `NVT` e envia-a para o servidor Telnet que se encontra em execução no computador remoto. Por seu turno, quando este recebe a informação, a mesma é transformada para o formato do computador onde vai ser acedida [Koz05]. Este processo está representado na Figura.

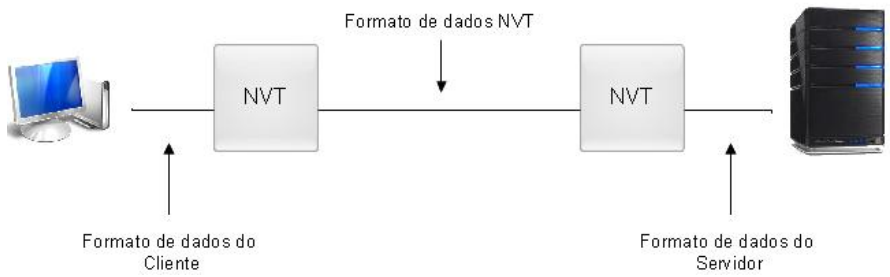


Figura 4.8: Comunicação entre computadores via Telnet

É de salientar que os dados, bem como as passwords, são enviados sem qualquer tipo de encriptação.

4.6.2 FTP

Como o próprio nome indica, o protocolo *File Transfer Protocol* (`FTP`) é um protocolo de transferência de ficheiros, tendo em conta restrições de acesso e propriedades dos mesmos.

Uma das diferenças deste protocolo em relação a outros reside no facto de utilizar duas ligações TCP, em detrimento de uma [Koz05]. Essas duas ligações são:

- **Ligação de Controlo**

  Criada quando uma sessão `FTP` é estabelecida, é utilizada unicamente para transmitir informações de controlo, designadamente, comandos FTP e respectivos resultados.

- **Ligação de Dados**

  Criada quando é necessário enviar ficheiros do cliente para o servidor, ou vice-versa. Quando a transferência de ficheiros acaba, a ligação termina.

Como existem dois canais de comunicação, existem, igualmente, um responsável distinto para cada canal. Assim, o *Protocol Interpreter* (`PI`) é um componente com a função de gerir a ligação de controlo. O *Data Transfer Process* (`DTP`) é responsável por enviar e receber os dados, trocados entre o cliente e o servidor.

Para além dos componentes referidos anteriormente - que estão presentes quer no cliente, quer no servidor - do lado do cliente existe, ainda, um outro componente responsável por interagir com o utilizador [Koz05].

Os componentes que fazem parte do servidor e do cliente, assim como, os dois canais de comunicação entre ambos, são ilustrados na Figura 4.5.

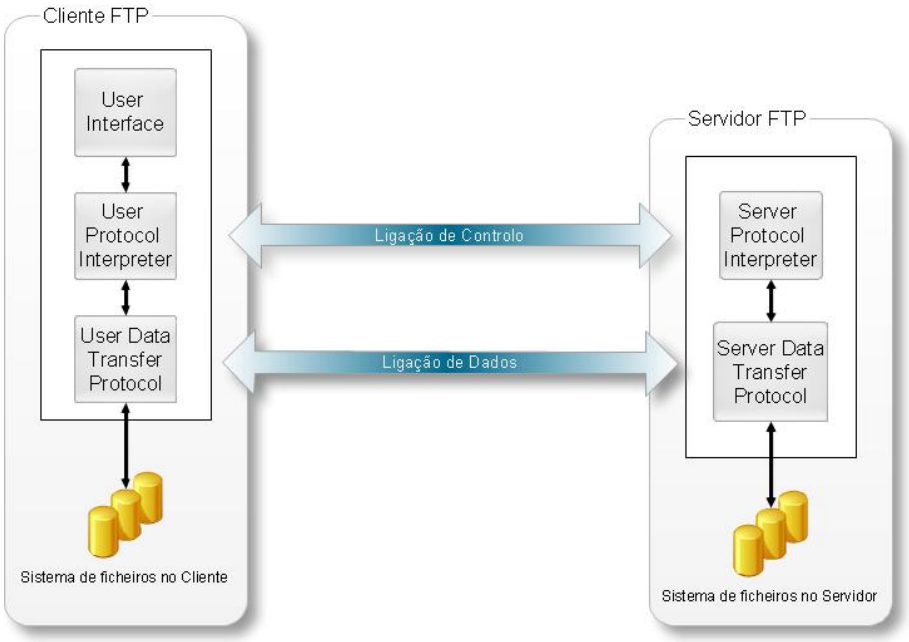


Figura 4.9: Protocolo FTP

4.6.3 SSH

O *Secure SHell* (`SSH`) é um protocolo, baseado na arquitectura cliente/servidor que permite que a troca de dados entre dois dispositivos seja feita de através de uma comunicação segura [BSB05].

Existem duas versões deste protocolo, SSH-1 e SSH-2, sendo que ambas consideram três aspectos [BSB05]:

- **Autenticação**

  Determinar, com exactidão, a identidade da pessoa que se pretende autenticar. Quando tentamos aceder a uma conta remota, o `SSH` exige uma prova digital da nossa identidade, por exemplo, username e password.

- **Encriptação**

  Consiste em codificar os dados de tal forma que estes sejam imperceptíveis a quem os tentar interceptar, protegendo, desta forma, a informação que passa através da rede.

- **Integridade**

  Garante que a informação que é transmitida chega ao seu destino de forma inalterada. O `SSH` possui mecanismos para verificar se houve alguma modificação nessa informação antes de chegar ao seu destino.

Na Figura 4.10, podemos observar uma ilustração da funcionalidade dos aspectos previamente referidos.

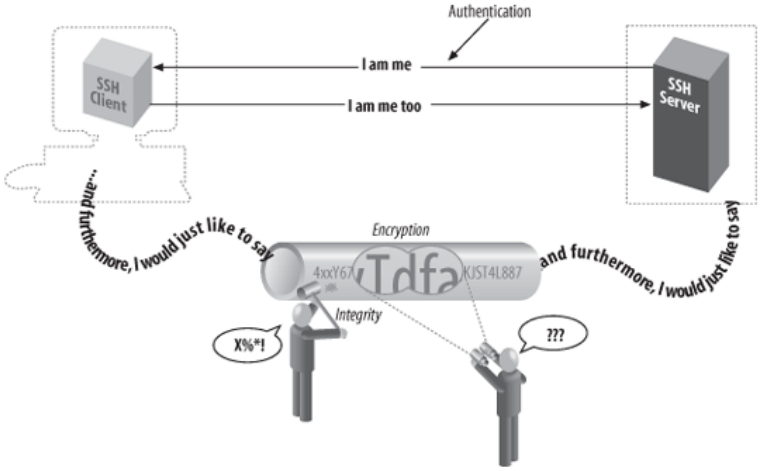


Figura 4.10: Protocolo SSH [BSB05]

Para além de ser possível iniciar uma sessão num dispositivo remoto de forma segura, o `SSH` permite, ainda, a transferência de ficheiros de forma segura. Através do protocolo *Secure Copy* (`SCP`) os ficheiros são automaticamente encriptados a quando da sua saída e desencriptados à chegada à máquina destino.

Com a capacidade de encapsular sockets (port forwarding), é possível aumentar a segurança dos serviços existentes, como, por exemplo, telnet ou `FTP` [BSB05].

4.7 Monitorização de uma Infra-estrutura

Com o aumento da complexidade das infra-estruturas aumenta a necessidade de metodologias e soluções que auxiliem a sua gestão e monitorização, em todos os níveis, âmbitos e

camadas que a constituem.

Um sistema de monitorização tem como finalidade supervisionar uma infra-estrutura, de forma a recolher informação relevante sobre o comportamento da mesma, desde o hardware até às próprias aplicações. Toda essa informação é utilizada para detectar problemas e comportamentos estranhos que possam por em risco o bom funcionamento da infra-estrutura.

Marinescu et al [MLCS90] propuseram um modelo teórico baseado no paradigma evento – acção, que ajuda na compreensão de como um sistema de monitorização consegue reagir e actuar em resposta a determinados eventos. A problemática da monitorização é apresentada sob a forma de ciclo básico[MLCS90]:

Ocorrência de evento; Avaliação de condição; Execução de acção

Como é perceptível, através do ciclo apresentado, aquando da detecção de um evento, segue-se um processamento que tem como propósito avaliar a situação e, de acordo com essa avaliação, pode desencadear uma eventual reacção, que pode, ou não, exercer alguma forma de controlo sobre o sistema que está a ser monitorizado.

A definição dos tipos de eventos, das condições a avaliar e das acções a efectuar, dependem dos objectivos da monitorização e da natureza dos sistemas a monitorizar.

No caso concreto de uma infra-estrutura `JEE` que vai alojar aplicações Web, existe uma quantidade diversificada de situações/acontecimentos em que os administradores devem ser notificados, designadamente [BC08, Jos07]:

- Quando um servidor deixa de estar acessível;
- Quando algum processo importante bloqueia, ou simplesmente, não iniciou correctamente;
- Quando existe uma sobrecarga nos recursos, nomeadamente, falta de disco, utilização excessiva do processador, entre outros.

Sem uma monitorização automatizada, detectar as situações mencionadas pode ser extremamente moroso, e, em determinados casos, quando a infra-estrutura se encontra distribuída por diversos locais, ser praticamente inexequível.

Um sistema de monitorização bem implementado e configurado ajuda não só a detectar mais rapidamente os problemas, mas também, a preveni-los.

Capítulo 5

Ambiente de Deployment

À medida que cresce a dimensão das aplicações, a fase de deployment adquire um maior relevo, deste modo, pretendemos, neste capítulo, aprofundar a temática do deployment. Para tal, começamos por apresentar diversas sequências de actividades que constituem o processo de deployment. De seguida, listamos vários servidores aplicacionais, que são essenciais para a execução das aplicações JEE, bem como os ficheiros descritores presentes nas aplicações, a partir dos quais os servidores aplicacionais obtêm informação sobre como devem geri-las. No final, são abordadas várias estratégias para o deployment e gestão de aplicações JEE.

5.1 Processo de Deployment

O processo de deployment de software pode ser considerado como uma sequência de actividades para colocar as aplicações desenvolvidas num determinando ambiente, designadamente, um computador pessoal, um servidor, entre outros, de forma a poderem ser utilizadas pelos utilizadores.

Reconhecendo que cada software é único, a sequência de actividades a serem seguidas no seu processo de deployment é difícil de definir. Portando, apresentamos, de seguida, algumas sequências de actividades do deployment, que podem ser encontradas em várias referências bibliográficas.

Em [LS06] o processo de deployment começa com uma actividade designada por *Shipping*, que engloba o empacotamento dos componentes do sistema e a transferência do pacote resultante do local de desenvolvimento para o ambiente de deployment. Segue-se a *Installation* que consiste na instalação do sistema no ambiente de deployment. A *Update* acontece se houver algum tipo de alterações e seja necessário reconfigurar/actualizar o sistema. Finalmente, há uma outra actividade que não é mais do que executar o sistema

instalado e/ou actualizado.

Na OSGi Service Platform [Alla] o processo de deployment inclui as actividades *Install*, *Update* e *Uninstall*.

Na especificação elaborada pela Object Management Group [Gro], a primeira actividade designa-se por *Packaging*, e tem como objectivo o empacotamento do sistema, juntamente com as suas configurações. A *Installation* é a transferência do pacote de software, resultante da primeira actividade, para um repositório de software. Na actividade *Configuring* podem ser acrescentadas algumas configurações funcionais aos pacotes de software presentes no repositório. Um planeamento de como e onde o sistema vai ser executado na infra-estrutura do ambiente de deployment é elaborada no *Planning*. Na actividade *Preparation* são transferidos os componentes do sistema para os respectivos servidores da infra-estrutura, de acordo com o planeamento elaborado. Finalmente, o software é colocado em execução, sendo esta actividade designada por *Launching*.

Na sequência de actividades apresentada por Merle e Belkhatir, em [MB04], uma aplicação pode ser instalada, actualizada, ou, ainda, ser alvo de reconfigurações. De modo a poder ser executada, uma aplicação tem de ser activada em primeiro lugar. Quando já não for necessária, a aplicação pode ser desinstalada. Todas estas actividades e suas interligações estão representadas na Figura 5.1.

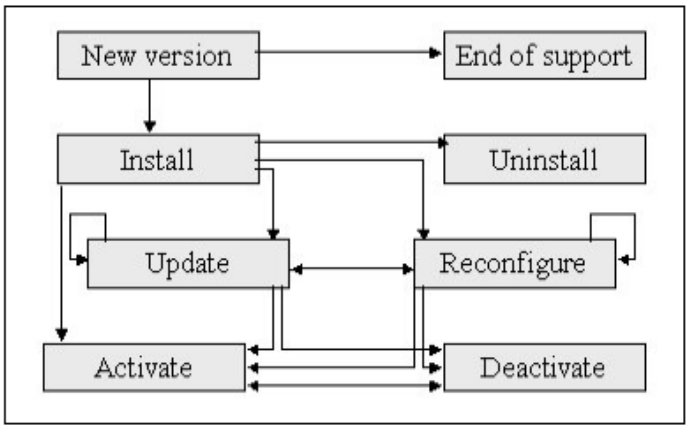


Figura 5.1: Processo de deployment presente no ambiente ORYA [MB04]

Uma sequência de actividades em tudo semelhante à anterior é apresentada por A. Ketfi e N. Belkhatirem em [KB05]. A única excepção é a presença de uma actividade anterior à instalação designada *Configuration*. Nesta acção são seleccionados os componentes apropriados para a plataforma onde o sistema vai ser colocado, sendo empacotados para posterior instalação.

Matthew J. Rutherford *et all* em [RAC+02] para além das actividades de *Install*,

Activate, *Deactivate*, *Reconfigure* e *Update* referidas anteriormente, adicionaram uma nova actividade denominada *Release*, que é executada antes da *Install* e tem como objectivo empacotar todos os componentes e ficheiros de configuração necessários para a instalação do sistema. A *Adapt*, em tudo idêntica à *Reconfigure*, tem como principal diferença o facto de esta se destinar a manter a integridade do sistema face a alterações na infra-estrutura que o suporta. A actividade que se encarrega de retirar o sistema da infra-estrutura é denominada *Remove*. Existe, ainda, a acção *Retire* que faz com que um sistema fique indisponível para o deployment.

O processo de deployment presente em [Hey06] é constituído por cinco actividades distintas. A primeira denomina-se *Acquiring*, onde os componentes de um sistema e respectiva metadata são colocados num repositório. No *Plannig* é desenvolvido um plano de deployment que tem em consideração as especificações, quer da aplicação, quer da infra-estrutura onde esta vai ser colocada e as condições/restrições definidas pelo utilizador. Neste plano está definida a localização dos componentes do sistema na infra-estrutura. Após o planeamento e de acordo com o mesmo, é feita a instalação do sistema na infra-estrutura, o que é designado por *Installation*. De seguida, e caso seja necessário, são feitos alguns ajustes a determinadas propriedades do sistema, *Configuration*. Por último, inicia-se o sistema acabado de instalar e configurar, actividade que é intitulada de *Execution*.

Como foi possível constatar, existem diferentes sequências de actividades, sendo perceptível algumas semelhanças entre elas e, em alguns casos, a existência de uma maior divisão do processo de deployment.

5.2 Servidores Aplicacionais JEE

Dada a crescente diversidade de servidores aplicacionais disponíveis, torna-se necessária uma maior ponderação aquando da escolha da melhor solução. Para fazer uma boa escolha é necessário ter em conta diversos aspectos, designadamente, quais as APIs presentes no servidor aplicacional ou a existência de funcionalidades de clustering, bem como mecanismos que facilitem e permitam uma melhor gestão dos mesmos.

É imprescindível que os servidores aplicacionais possuam mecanismos de clustering, de forma a ser possível obter uma melhor escalabilidade e disponibilidade. De referir que esta característica está intimamente ligada com o tipo e objectivos das aplicações que vão ser colocadas no servidor aplicacional.

Para além de ser necessário ter em consideração os mecanismos que o servidor aplicacional possui para obter escalabilidade e disponibilidade,é, também, essencial ter em conta o desempenho do servidor aplicacional, e perceber se estão presentes mecanismos

para o melhorar.

A presença de um bom sistema de monitorização é igualmente importante, de modo a se evitarem determinados problemas e, no caso de algum surgir, perceber, o mais rapidamente possível, qual a raiz desse problema.

Outro aspecto a ter em consideração - apesar de não ser um requisito obrigatório - é a certificação JEE atribuída pela Sun aos servidores aplicacionais. Para efectuar esta certificação a Sun possui o J2EE *Compatibility Test Suite* (`CTS`).

Esta certificação tem várias vantagens, nomeadamente:

- Garante que o Servidor Aplicacional é compatível com a especificação JEE
- Permite que os programadores possam reutilizar os componentes migrando-os de um Servidor de Aplicações para outro
- Possibilita que o Servidor Aplicacional aceite componentes vindos de outro servidor compatível com a especificação

Para além dos aspectos enumerados anteriormente, existe ainda a questão dos custos da solução, que pode, em muitos dos casos, ser um dos mais relevantes, uma vez que algumas das soluções são bastante dispendiosas.

Nas subsecções seguintes são apresentados alguns servidores aplicacionais, e descritas algumas características dos mesmos.

5.2.1 JBoss

JBoss Application Server [Red] é um servidor aplicacional JEE open-source, gerido pela JBoss Enterprise Middleware, divisão da empresa Red Hat. Visto ser um servidor aplicacional totalmente desenvolvido em Java, pode ser executado em qualquer Sistema Operativo que suporte Java.

Sendo um dos mais famosos e populares entre os servidores aplicacionais, o JBoss Application Server é construído tendo como base o *JBoss Microcontainer*, um container para gestão de *Plain Old Java Objects* (`POJOs`), incluindo a sua instalação, configuração e ciclo de vida. O *JBoss Microcontainer* vem substituir o *JBoss JMX Microkernel* das versões anteriores deste servidor aplicacional.

São vários os recursos disponibilizados por este servidor aplicacional, designadamente, o JBoss Web[1], Web container[2] baseado na implementação do Apache Tomcat, inclui o

[1] `https://www.jboss.org/community/wiki/JBossApplicationServerOfficialDocumentationPage`
[2] Ver Secção 2.2

Apache Portable Runtime (APR) e promete ter uma escalabilidade e desempenho equiparados ao Apache HTTP Server. A inclusão do JBoss EJB3 nas últimas versões do JBoss Application Server fornece a implementação da última revisão da especificação do Enterprise Java Beans (EJB). Outras tecnologias JEE como JMS, JTA, JNDI, JMX, SOAP, entre outras, estão, também, incluídas neste servidor aplicacional.

Este servidor aplicacional possui ainda suporte para JEE Web Services e *Service Oriented Architecture* (SOA) através do JBossWS, suporte para *Aspect-Oriented Programming* (AOP), que amplia a produtividade do programador e, além disso, possui suporte a clustering e a distributed cache.

Como foi possível verificar, o JBoss Application Server tem tudo o que é preciso para corresponder às exigências de uma aplicação JEE. Torna-se, ainda assim, necessário avaliar se a performance e escalabilidade do mesmo tiveram, efectivamente, um melhoramento significativo nas últimas versões do servidor aplicacional, face aos resultados apresentados na secção 3.1.

É de salientar que a última versão que saiu é totalmente compatível com o Java EE 5, validada pelos testes de conformidade para certificação desta tecnologia, levados a cabo pela Sun[3].

5.2.2 JOnAS

O JOnAS Application Server [OW2b] é um servidor aplicacional open-source, desenvolvido e gerido por um grupo designado *OW2 consortium*.

A última versão deste servidor aplicacional foi desenvolvida tendo como base uma arquitectura OSGi™. Este implementa uma arquitectura orientada a serviços no próprio servidor aplicacional, sendo que uma das grandes vantagens é a possibilidade de configuração dinâmica do servidor, onde os serviços podem ser parados, reconfigurados e iniciados em runtime. Como pode ser observado na Figura 5.2, que representa a arquitectura do JOnAS, os serviços são disponibilizados como OSGi bundles, havendo um isolamento entre eles [OW2a].

Este servidor aplicacional proporciona escalabilidade e disponibilidade através de soluções de clustering e mecanismos de optimização internos. São, ainda, implementados mecanismos de distribuição de carga, replicação e failover, quer ao nível do HTTP, quer ao nível do RMI [OW2a].

Ao contrário das últimas versões do JBoss Application Server que têm um Web container próprio, o JOnAS Application Server pode utilizar o Tomcat ou o Jetty, para esse

[3] http://java.sun.com/javaee/overview/compatibility.jsp

Figura 5.2: Arquitectura do JOnAS 5 [OW2b]

efeito.

A função EJB3 container[4] é desempenhada pelo EasyBeans. Este container possuí, ainda, mecanismos de distribuição de carga e failover.

Para lidar com a maior complexidade do deployment, gestão e monitorização das aplicações, é apresentada uma ferramenta, JASMINe, que promete reduzir o custo dessas tarefas [OW2a].

A JASMINe permite a configuração e a gestão de clusters, garantindo, desta forma, a criação dinâmica dos mesmos, a distribuição da carga pelos servidores que formam o cluster e a replicação. Outra vantagem desta ferramenta é a possibilidade de aceder a informação do sistema como processador e memória em utilização, memory heap da JVM, logs do servidor aplicacional, entre outras.

Ao apresentar uma sofisticada modularidade com a arquitectura OSGi, há uma grande aposta para aumentar a utilização deste servidor aplicacional, sendo que, a última versão do JOnAS Application Server também recebeu a certificação de compatibilidade com Java EE 5.

5.2.3 Apache Geronimo

O Apache Geronimo [Fou] é um servidor aplicacional criado pela Apache Software Foundation compatível com a especificação Java EE 5, que é distribuído sob a licença Apache.

São cinco as distribuições deste servidor aplicacional, sendo que apenas duas delas estão certificadas pela Sun. Como não tem Web container próprio, existe a versão Apache Geronimo com o Tomcat para essa função e o AXIS2 para os Web Services. Pode-se ainda

[4]Ver Secção 2.2

escolher o Apache Geronimo com o Jetty como Web container e o CXF para os Web Services. Para a persistência ambas as distribuições utilizam o OpenJPA. As distribuições estão disponíveis para Windows, Linux e Mac OS.

Nas últimas versões deste servidor aplicacional o suporte a clustering foi melhorado e alargado à distribuição que utiliza o Tomcat como Web container. Uma consola de monitorização permite recolher várias estatísticas e dados sobre a performance de múltiplos servidores, sendo esta informação apresentada ao utilizador.

Na consola de administração, para além da consola de monitorização introduzida acima, é possível encontrar, nas últimas versões, o *Plan Creator*. O *Plan Creator* ajuda a simplificar a criação dos planos de deployment para este servidor aplicacional. Indicando um determinado pacote de uma aplicação Web - os WAR - este utilitário guia o utilizador através de um *Wizard* para, deste modo, poder gerar o ficheiro *geronimo-web.xml* automaticamente.

5.2.4 Oracle WebLogic Application Server

O Oracle WebLogic Server [Cora] é um servidor aplicacional apresentado pela Oracle Corporation. Anteriormente conhecido como BEA WebLogic, quando em 2008 a Oracle adquiriu a empresa BEA, passou a designar-se Oracle WebLogic Server.

Existem quatro versões do Oracle WebLogic Application Server:

- **WebLogic Server Standard Edition**

  Esta versão é totalmente compatível com o Java EE 5 e EJB3, tendo sido certificada pela Sun. Possui, ainda suporte para as mais variadas `APIs` Java EE, incluindo Web Services, transacções, persistência e segurança. Nesta versão está, também, incluído o *Oracle TopLink*, uma solução de persistência Java à semelhança do Hibernate. Esta solução de *Object-Relational Mapping* (`ORM`) tem como objectivo ligar os objectos às base de dados relacionais.

  O WebLogic Server está equipado com a capacidade de *FastSwap* que permite reduzir o tempo do ciclo development-deploy-debug [Corb]. Isso acontece uma vez que é possível actualizar classes java e essas alterações terem efeito imediato na aplicação que está em desenvolvimento, sem ser necessário reiniciar a aplicação ou o próprio servidor.

  De modo a facilitar a gestão da infra-estrutura, através da consola administrativa, é possível fazer vários testes diagnósticos ao sistema. E, de forma a evitar que o servidor fique em baixo devido a erro humano, quando se tentam efectuar alterações

nas configurações do servidor e este não as poder aceitar é possível fazer o roll-back das mesmas.

- **WebLogic Server Enterprise Edition**

  Com as mesmas funcionalidades da Standard Edition, acrescenta-se ainda o suporte a clustering e sua gestão.

  Esta versão usufrui, também, de um sistema de monitorização e gestão da infra-estrutura e componentes JEE mais avançando que os presentes na Standard Edition.

- **WebLogic Suite**

  É a solução mais completa. Para além de conter todas as funcionalidades do WebLogic Server Enterprise Edition, possui distributed in-memory caching melhorando, desta forma, o acesso aos dados. Segundo a Oracle, está equipada com a mais rápida *Java virtual machine* (`JVM`), a Oracle JRockit.

  O Oracle WebLogic Operations Control, presente nesta versão, permite definir políticas que controlam a alocação de recursos de hardware e software de modo a assegurar que a qualidade dos serviços atinge os objectivos definidos. Deste modo, as aplicações podem ser instaladas e os recursos vão sendo ajustados dinamicamente.

- **WebLogic Application Grid**

  É uma grid que pode complementar as infra-estruturas dos servidores aplicacionais possibilitando às aplicações uma maior performance e escalabilidade.

  Esta versão possui in-memory data grid que possibilita que os acessos aos dados mais usados sejam mais rápidos. Encontra-se, também incluída a Oracle JRockit como `JVM`.

  Da mesma forma que a versão WebLogic Suite, a WebLogic Application Grid disponibiliza o Oracle WebLogic Operations Control.

5.2.5 IBM WebSphere Application Server

O WebSphere Application Server [IBM] é uma das soluções apresentadas pela IBM para correr aplicações JEE, sedno certificado pela Sun, como totalmente compatível com JEE 1.2, JEE 1.3, JEE 1.4 e Java EE 5[5].

Uma das grandes vantagens da família WebSphere é a existência de versões para um vasto número de plataformas, nomeadamente:

[5] `http://java.sun.com/javaee/overview/compatibility.jsp`

- Linux
- AIX
- HP-UX
- Z/OS (Mainframes)
- Windows

Este servidor aplicacional está disponível em diferentes versões, tendo cada uma delas características específicas [AHP+09]. De seguida são apresentadas algumas especificações dessas versões.

- WebSphere Application Server - Express

  Esta versão tem como alvo as empresas médias ou os departamentos das grandes empresas. Contém suporte completo para Java EE 5, `EJB` 3.0 e Web services. Tem como limitações, o facto de apenas poder ser instalado num único servidor de 32-bit e para um máximo de 240 *Processor Value Units* (`PVUs`).

- WebSphere Application Server

  Com funcionalidades equivalentes à versão Express (ambiente com um único servidor), está disponível para as versões 32-bit e 64-bit e não tendo a limitação de `PVUs`.

- WebSphere Application Server for Developers

  Características iguais à versão *WebSphere Application Server*, difere apenas no tipo de licença. Neste caso, o uso deste pacote apenas é permitido em ambiente de desenvolvimento.

- WebSphere Application Server Network Deployment

  Nesta versão já se encontram funcionalidades para construir uma infra-estrutura mais complexa. Com as funcionalidades do *WebSphere Application Server* são acrescentadas capacidades de clustering, alta disponibilidade e escalabilidade. Além disso, possui mecanismos avançados de gestão de configurações distribuídas.

  Um serviço de cache dinâmica encontra-se, também, incluído neste pacote, aumentando, desta forma, a performance relativamente aos pacotes anteriores.

- WebSphere Application Server for z/OS

  É a versão deste servidor aplicacional para mainframes. Possuindo as funcionalidades presentes na versão *WebSphere Application Server Network Deployment*, contém, ainda, mecanismos de gestão do Workload e disponibiliza relatórios sobre o consumo de recursos.

Na Figura 5.3 pode-se visualizar um esquema resumo dos recursos de cada uma das versões disponíveis do WebSphere Application Server, descritas anteriormente.

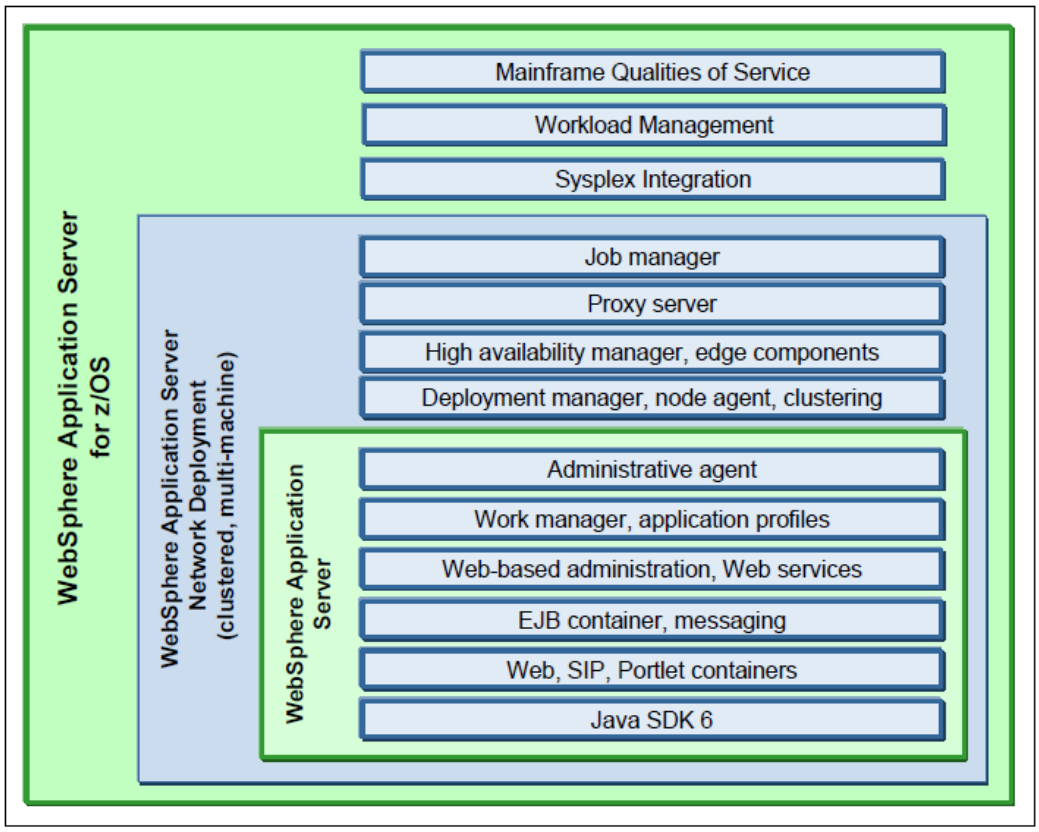


Figura 5.3: Versões do WebSphere Application Server [IBM]

De acordo com a necessidade e o objectivo da aplicação JEE em causa, é importante fazer uma opção ponderada sobre a versão do servidor aplicacional a escolher, uma vez que, cada versão tem um custo e licenciamento diferente.

5.2.6 Sun GlassFish Server

O Sun GlassFish Server [Sunb], previamente denominado Sun Java System Application Server, é um servidor aplicacional apresentado pela Sun Microsystems, Inc, totalmente compatível com o Java EE 5.

Vem equipado com outro projecto da Sun, Metro[6], uma stack de Web Services que permite às aplicações a interoperabilidade com *Windows Communication Foundation* (`WCF`).

Com suporte para clusting, é possível criar e gerir clusters através de uma única consola de administração. É exequível fazer crescer ou diminuir um cluster de forma dinâmica. Vem com um sistema de distribuição de carga integrado, que permite que esta seja distribuída pelos diferentes nodos.

Através da consola de administração é possível monitorizar as aplicações, nomeadamente os recursos que consomem e, caso seja necessário, lançar avisos sobre uma determinada situação. Através dos sistemas de monitorização disponibilizados, é mais fácil perceber o que pode estar a causar um problema de performance.

5.2.7 Comparação entre os Servidores Aplicacionais

Com a diversidade de oferta no que toca a servidores aplicacionais disponíveis, a escolha do mais apropriado pode ser complexa. Se, por um lado, todos disponibilizam as mesmas funções básicas, existem, porém, algumas diferenças, como foi possível constatar na secção anterior. Na Tabela 5.1 estão representadas algumas das características dos servidores aplicacionais apresentados.

Tabela 5.1: Tabela Resumo dos Servidores Aplicacionais apresentados

Servidor Aplicacional	Fabricante	Versão	Clustering	Licença
Jboss	Red Hat	5.1.0	x	LGPL
JOnAS	OW2 Consortium	5.1.0-RC2	x	LGPL
Apache Geronimo	Apache	2.1.4	x	Apache License
WebLogic	Oracle	10.3	x*	Proprietária
WebSphere AS	IBM	7.0.0.0	x*	Proprietária
Glassfish	Sun Microsystems	2.1 / 3 Prelude	x	GPL

* Apenas disponível na versão comercial

Factores como a escalabilidade e disponibilidade, com cada vez maior relevância, são características preponderantes num servidor aplicacional. Portanto, a presença de mecanismos que permitam atingir uma maior escalabilidade e disponibilidade têm de ser tidos em conta.

Pela observação da Tabela 5.1, podemos retirar um outro factor relacionado com o custo e a licença de cada servidor aplicacional. Existem servidores aplicacionais em que

[6] https://metro.dev.java.net/

para ter acesso a determinadas funcionalidades é preciso pagar, como são os casos do WebLogic e do WebSphere AS. Em outros casos, porém, os custos podem estar relacionados com o suporte dado pelas empresas que disponibilizaram o servidor aplicacional, que, apesar de poder ser utilizado de forma totalmente livre, ter uma assistência 24h tem os seus custos.

Outro factor a ter em conta é o desempenho de cada servidor aplicacional. Como foi observado em alguns estudos apresentados na Secção 3.1, nomeadamente em [CMZ02], o desempenho de diferentes servidores aplicacionais quando testados em situações iguais pode ser diferente. De salientar que o desempenho pode, ainda, ser afectado pela arquitectura da aplicação e da forma como esta é instalada e configurada, ideia reforçada em [Liu02] e [MZ05].

Em determinados casos é necessário, também, ter em conta a versão das `APIs` presentes. No entanto, nesta dissertação apenas foram apresentados servidores aplicacionais que obtiveram a certificação da SUN para JAVA EE, daí todos terem as mesmas funcionalidades básicas.

Depois de enunciados vários factores que podem auxiliar na escolha do servidor aplicacional mais adequado, percebe-se a necessidade de um estudo profundo do sistema que se pretende construir, bem como a forma como a instalação e configuração das aplicações vai ser feita para, deste modo, se obter resultados satisfatórios.

5.3 Descritores de Deployment JEE

Como explanado na Secção 2.4, todos os tipos de empacotamento de aplicações JEE incluem ficheiros descritores XML cujos elementos descrevem como instalar e configurar componentes num ambiente específico. Mais concretamente, estes descritores podem conter informações, como o nome da aplicação e a sua descrição, que podem ser utilizados pelo servidor aplicacional para gerar logs com as aplicações nele instaladas [FC05]. Nestes ficheiros podemos encontrar, igualmente, informações dos componentes que fazem parte da aplicação, as partes que se referem a esses mesmos componentes e o tipo de serviços que o container deve fornecer ao componente. Além disso, é possível verificar nos ficheiros descritores um outro tipo de informações que se referem às dependências dos componentes no que toca a recursos que devem estar disponíveis para o seu correcto funcionamento. Estes recursos podem ser, por exemplo, um tipo específico de fonte de dados `JDBC` ou, então, outros componentes necessários para executar uma determinada tarefa [FC05].

Existem dois tipos de descritores, os normalizados (*standard*) e os específicos. Os descritores standard configuram genericamente o componente com comportamentos providos

por todos os servidores aplicacionais.

No entanto, a competitividade do mercado faz com que diferentes servidores aplicacionais possuam recursos e serviços variados, como se pode verificar na secção 5.2. Não existe uma forma de prever quais são esses serviços, sendo muito difícil elaborar uma configuração genérica, uma vez que essa configuração varia de servidor para servidor. Por esta razão existem descritores que são específicos de cada servidor aplicacional. De salientar, ainda, que nesta secção serão apenas abordados os descritores normalizados.

De acordo com a especificação JEE, cada tipo de encapotamento tem um ficheiro descritor específico. Na Tabela 5.2 podemos encontrar o nome e localização dos ficheiros descritores de acordo com o empacotamento utilizado.

Tabela 5.2: Descriptores JEE standard para o Deployment

Pacote JEE	Descritor de deployment
Enterprise Java Beans JAR	META-INF/**ejb-jar.xml**
WAR	WEB-INF/**web.xml**
RAR	META-INF/**ra.xml**
EAR	META-INF/**application.xml**
Enterprise Applicattion Client Archive (JAR)	META-INF/**application-client.xml**

Nas subsecções seguintes são descritos, com mais detalhe, cada um desses ficheiros descritores.

5.3.1 Ficheiro descritor para Enterprise Java Beans JAR

O ficheiro ejb-jar.xml especifica quais são as classes e interfaces de cada `EJB`, indicando se este suporta transacções. No caso de haver suporte para transacções, no descritor deve estar, ainda, a informação de que métodos o container vai gerir as transacções.

Na Figura 5.4, podemos ver um fragmento deste descritor XML.

Para identificar o `EJB` é utilizada a tag <ejb-name>. As tags <ejb-class>, <home> e <remote> informam onde estão a classe que representa o Bean e as interfaces Home e Remote desse componente.

Para indicar se o *Session Bean* é do tipo *Stateful* ou *Stateless*, é utilizada a tag <session-type>.

A tag <transaction-type> configura quem irá gerir o serviço de transacção do componente, podendo ser o próprio Bean ou o *container*. A primeira opção transfere para a implementação do Session Bean a responsabilidade controlar a transacção do componente

```
<ejb-jar version="3.0">
    <enterprise-beans>
        <session>
            <ejb-name>AppTeste</ejb-name>
            <remote>com.business.AppTeste</remote>
            <ejb-class>com.business.AppTesteBean</ejb-class>
            <session-type>stateless</session-type>
            <transaction-type>Container</transaction-type>
        </session>
    </enterprise-beans>
....
    <assembly-descriptor>
        <container-transaction>
            <method>
                <ejb-name>AppTeste</ejb-name>
                <method-name>*</method-name>
            </method>
            <trans-attribute>Required</trans-attribute>
        </container-transaction>
        <security-role>
            <role-name>utilizadores</role-name>
        </security-role>
    </assembly-descriptor>
</ejb-jar>
```

Figura 5.4: Exemplo de um descritor ejb-jar.xml

através do uso da API JTA. Já a opção container define que as transacções do componente serão da responsabilidade do servidor aplicacional.

A tag <assembly-descriptors> configura os serviços providos por todos os servidores aplicacionais, como serviço de nomes, segurança, transacções, persistência, entre outros. Na figura apresentada nesta tag está descrito como será o comportamento das transacções no container.

Existem ainda outras tags[7] que não estão contempladas no exemplo, nomeadamente, as referências a outros Beans (<ejb-ref> ou <ejb-local-ref>).

5.3.2 Ficheiro descritor para WAR

O web.xml especifica todos os servlets e páginas JSP que fazem parte de uma aplicação Web JEE. Contém, ainda, informações sobre referências a EJBs que são utilizados pela aplicação.

Na Figura 5.5 está representado um descritor de uma aplicação Web J2EE.

Neste ficheiro são definidos os mapeamentos dos servlets para um determinado URL (<url-pattern>), e, caso seja necessário, os parâmetros iniciais.

Através da tag <session-config> são definidos os parâmetros de uma sessão. No caso da figura está definido o tempo de espera que uma sessão inactiva fica em memória. Quando esse tempo passar, a sessão é eliminada.

Neste descritor, através da tag <error-page>, pode ser especificado o mapeamento

[7]Schema XML para o descritor EJB - http://java.sun.com/xml/ns/javaee/ejb-jar_3_0.xsd

```
<web-app>
    <display-name>NomeApp</display-name>
    <description>Breve descricao da app</description>
    <servlet>
        <servlet-name>TesteServlet</servlet-name>
        <servlet-class>teste.App.TesteServlet</servlet-class>
    </servlet>
    <servlet>
        <servlet-name>PaginaErroServlet</servlet-name>
        <servlet-class>teste.Error.PaginaErroServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>PaginaErroServlet</servlet-name>
        <url-pattern>/PaginaErroServlet</url-pattern>
    </servlet-mapping>
    <welcome-file-list>
        <welcome-file>paginainicial.html</welcome-file>
    </welcome-file-list>
    <error-page>
        <exception-type>java.lang.ArrayIndexOutOfBoundsException</exception-type>
        <location>/PaginaErroServlet</location>
    </error-page>
    <error-page>
        <error-code>404</error-code>
        <location>/error404.html</location>
    <error-page>
    <session-config>
      <session-timeout>30</session-timeout>
    </session-config>
</web-app>
```

Figura 5.5: Exemplo de um descritor web.xml

entre um determinado código de erro ou uma excepção e uma página ou um servlet específico.

5.3.3 Ficheiro descritor para EAR

O descritor application.xml contém informações sobre os módulos constituintes de um ficheiro EAR.

Neste descritor podemos definir os módulos que fazem parte do pacote EAR através da tag <module>. Dentro dessa tag, escolhemos entre <ebj>, <web> e <java> para descrever o módulo, como pode ser observado na Figura 5.6.

```
<application>
    <display-name>NomeAplicacao</display-name>
    <module>
        <ejb>AppTeste.jar</ejb>
    </module>
    <module>
        <web>
            <web-uri>WebTeste.war</web-uri>
            <context-root>/webteste</context-root>
        </web>
    </module>
    <module>
        <java>clienteTeste.jar</java>
    </module>
</application>
```

Figura 5.6: Exemplo de um descritor application.xml

É, também, possível colocar regras de segurança que são globais à aplicação, descrição

esta feita através da tag <security-role>.

5.3.4 Ficheiro descritor para RAR

No ra.xml estão incluídas informações relacionadas com a configuração JEE. Neste descritor é especificado o resource adapter, sendo que estas informações são utilizadas pelo servidor aplicacional na sua configuração.

```
<connector>
    <display-name>Skeleton Resource Adapter</display-name>
    <vendor-name>Sun Microsystems, Inc.</vendor-name>
    <eis-type>Unknown</eis-type>
    <resourceadapter-version>1.0</resourceadapter-version>
    <resourceadapter>
        <resourceadapter-class>
            com.ra.ResourceAdapterImpl
        </resourceadapter-class>
    </resourceadapter>
    <authentication-mechanism>
        <authentication-mechanism-type>BasicPassword</authentication-mechanism-type>
        <credential-interface>
            javax.resource.spi.security.PasswordCredential
        </credential-interface>
    </authentication-mechanism>
    <reauthentication-support>false</reauthentication-support>
</connector>
```

Figura 5.7: Exemplo de um descritor ra.xml

Como pode ser observado na Figura 5.7, este descritor, numa primeira parte, contém várias informações sobre o recurso externo, nomeadamente, o nome de quem o disponibiliza (<vendor-name>), o tipo de EIS que suporta (<eis-type>) e a sua versão (<resourceadapter-version>).

Além disso, torna-se necessário ter o nome completo da classe que implementa a interface
javax.resource.spi.ResourceAdapter. Esta informação é colocada dentro da tag <resourceadapter>.

Na última secção da figura são especificados os mecanismos de autenticação suportados pelo fornecedor de recursos adaptativos. Neste caso específico apenas o método "BasicPassword"é suportado. Este método é um mecanismo de autenticação baseado em utilizador-password[8].

5.3.5 Ficheiro descritor para Aplicações Cliente

O ficheiro descritor application-client.xml contem a descrição dos módulos `EJB` e outros recursos utilizados pela aplicação cliente.

[8]A informação completa das tags, juntamente como Schema XML que verifica a boa formação do descritor pode ser encontrada em `http://java.sun.com/j2ee/connector/download.html`

Na Figura 5.8 é apresentado um exemplo de um descritor de aplicação cliente.

```
<application-client>
    <display-name>Aplicacao Java</display-name>
    <env-entry>
        <description>Pequena Descricao</description>
        <env-entry-name>TesteMem</env-entry-name>
        <env-entry-type>java.lang.String</env-entry-type>
        <env-entry-value>Mensagem de Teste</env-entry-value>
    </env-entry>
    <ejb-ref>
        <ejb-ref-name>ejb/AppTeste</ejb-ref-name>
        <ejb-ref-type>Session</ejb-ref-type>
        <home>com.business.AppTesteHome</home>
        <remote>com.business.AppTeste</remote>
    </ejb-ref>
    <resource-ref>
        <res-ref-name>AppDB</res-ref-name>
        <res-type>javax.sql.DataSource</res-type>
        <res-auth>Container</res-auth>
    </resource-ref>
</application-client>
```

Figura 5.8: Exemplo de um descritor application-client.xml

A tag <env-entry> declara variáveis de ambiente para a aplicação. Dentro dessa tag é indicado o nome, o tipo e o valor dessa variável.

Como referido anteriormente, as referências a enterprize beans são feitas através da tag <ejb-ref>. Já para referenciar recursos externos utilizados pela aplicação é utilizada a tag <resource-ref>.

5.4 Deployment e gestão de Aplicações JEE

Ao longo da presente dissertação foi perceptível a dificuldade de executar uma instalação e configuração de uma aplicação. Efectivamente, é necessário ter em conta diversos aspectos de forma a garantir que a aplicação funcione convenientemente. Aspectos esses que vão desde a escolha do servidor aplicacional mais apropriado, até à correcta construção dos ficheiros descritores que fazem parte da aplicação.

A presente secção tem como finalidade dar a conhecer estratégias que são utilizadas para gerir e disponibilizar as aplicações JEE.

5.4.1 Ferramentas de Administração

Uma forma de gerir e disponibilizar aplicações passa pela utilização de ferramentas incorporadas nos próprios Servidores Aplicacionais sob a forma de um executável ou através da consola de administração dos mesmos.

Na Figura 5.9 podemos observar parte do painel de administração do servidor aplicacional JOnAS[9] referente ao deployment de módulos WAR.

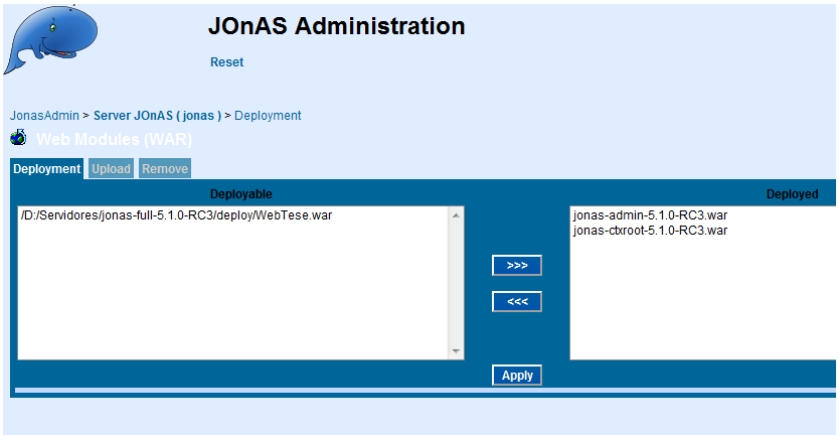


Figura 5.9: Painel de Administração do JOnAS

Através deste painel é possível efectuar toda a gestão dos módulos WAR instalados no servidor, designadamente adicionar, eliminar, activar e desactivar módulos. No caso concreto da figura apresentada, podemos constatar a existência de aplicação que, apesar de estar efectivamente no servidor, **WebTese.war**, encontra-se inactiva, podendo, no entanto, ser activada a qualquer momento.

5.4.2 JSR-88

A norma JSR-88, que originou a especificação de deployment [Suna], define `APIs` standard que permitem a instalação de aplicações JEE nos Servidores Aplicacionais. Através desta é possível estabelecer um contrato entre as ferramentas de instalação e os servidores aplicacionais, permitindo que as ferramentas, independentemente do seu fornecedor, sejam capazes de configurar, instalar e gerir aplicações em qualquer Servidor Aplicacional que tenha concluído com sucesso os testes de certificação da Sun [Exe04].

DeploymentManager

O DeploymentManager é um serviço disponibilizado pelas plataformas JEE que é o centro da interacção entre a ferramenta de instalação e configuração de aplicações e a plataforma JEE.

Este serviço disponibiliza interfaces para configurar, distribuir, iniciar e remover uma aplicação, possibilitando, também, a gestão e monitorização dos módulos instalados [Exe04,

[9]Ver Secção 5.2.2

Suna]. As operações de distribuição, arranque, paragem e desinstalação podem ser aplicadas num único servidor aplicacional JEE ou, então, a um cluster de servidores JEE [Exe04]. Uma ferramenta de deployment tem de obter uma referência para o DeploymentManager através da DeploymentFactory, que funciona como um driver para o servidor JEE.

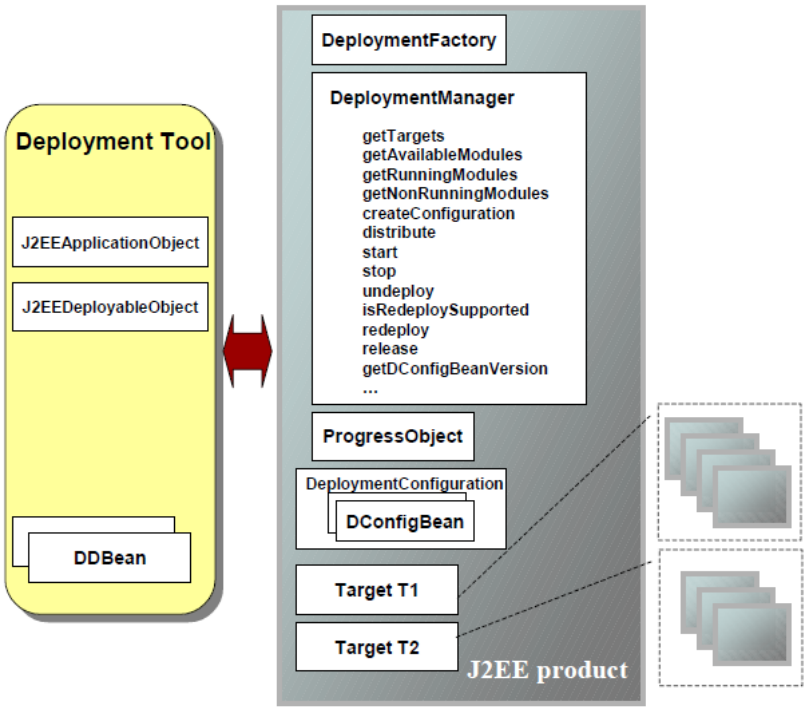


Figura 5.10: API Standard de Deployment

Na Figura 5.10 podes observar um sub-conjunto das `APIs` que devem ser disponibilizadas pelo Servidor aplicacional de forma a de tornar possível a instalação e configuração de aplicações JEE por parte de uma ferramenta externa.

5.4.3 "Hot Deploy"de forma manual

É a capacidade de instalar uma aplicação enquanto o servidor se encontra em execução, bastando, para tal, colocar a aplicação compilada numa pasta específica do servidor aplicacional. Além disso, esta característica permite do mesmo modo remover uma aplicação, sendo apenas necessário remover os ficheiros que constituem a mesma do servidor.

De forma a tornar exequível esta estratégia é necessária uma constante monitorização, por parte dos servidores aplicacionais, de pastas específicas onde são colocadas as aplicações.

De salientar que o Hot Deployment torna a instalação de aplicações JEE bastante simples.

5.4.4 Ant

O Apache Ant, é uma ferramenta escrita em Java utilizada para automatizar tarefas de construção, instalação e configuração de software.

O facto de ser desenvolvida em Java torna-a portável por natureza. A ferramenta utiliza ficheiros escritos em XML (build files) nos quais estão descritas as operações a executar. Cada ficheiro XML possui um projecto, constituído por um conjunto de alvos que por sua vez possuem por um número variável de tarefas a executar [HNG05].

A partir do Apache Ant, é possível automatizar todo o processo de compilação e instalação de aplicações JEE. O Ant compila as fontes das aplicações, gera os EARs, WARs ou JAR, podendo, ainda, transferir os ficheiros para os lugares desejados.

Na Figura 5.11 podemos observar um excerto de um ficheiro XML que tem como objectivo efectuar o deployment da aplicação **myAPP.jar** num servidor aplicacional JOnAS.

```
<property name="jonas.root" value="C:/servers/jonas"/>
<property name="lib.dir" value="C:/projectos"/>

<serverdeploy action="deploy" source="${lib.dir}/myApp.jar">
      <jonas server="MyJOnAS" jonasroot="${jonas.root}">

          <classpath>
              <pathelement path="${jonas.root}/lib/RMI_jonas.jar"/>
              <pathelement path="${jonas.root}/config/"/>
          </classpath>
      </jonas>
</serverdeploy>
```

Figura 5.11: Exemplo de um BuildFile

Apesar de no exemplo apresentado na figura o deployment ser local, este pode ser feito remotamente através do Ant, tirando partido de uma ligação `SSH`, bastando, para isso, colocar a seguinte tarefa no ficheiro XML:

```
<scp file="$lib.dir/myAPP.jar"todir="user@somehost:${jonas.deploydir}"password="password"/>
```

É de referir que, para efectuar a ligação remota o Ant depende de uma biblioteca externa[10], o Jsch, que será abordada em mais pormenor mais tarde.

[10]http://ant.apache.org/manual/OptionalTasks/scp.html

5.4.5 OSGI

O OSGi [Allb] (Open Services Gateway initiative) define um modelo de componentes dinâmicos para ambientes de programação e execução baseados na linguagem Java. Os componentes da arquitectura OSGi são designados por *bundles*.

Um *bundle* OSGi possui a vantagem de poder ser instalado, iniciado, actualizado, ou removido sem obrigar a reiniciar o sistema, sendo que estas acções podem ser executadas local ou remotamente. A comunicação entre *bundles* é efectuada através das suas interfaces.

Hoje em dia, são vários os projectos que utilizam OSGi, nomeadamente alguns servidores aplicacionais que são construídos sobre este modelo, por exemplo, Glassfish V3 ou JOnAS v5.

Capítulo 6

Ferramenta de Apoio

A crescente importância e complexidade das tarefas de construção de infra-estruturas e de deployment de aplicações foi notória ao longo deste trabalho. Cientes da necessidade de as simplificar, foi desenvolvido uma ferramenta com o propósito de auxiliar o utilizador na realização dessas tarefas.

Podemos dividir a ferramenta em vários módulos, como é perceptível na Figura 6.1, cada um com funcionalidades específicas que permitem, por exemplo, o deployment e a gestão de aplicações.

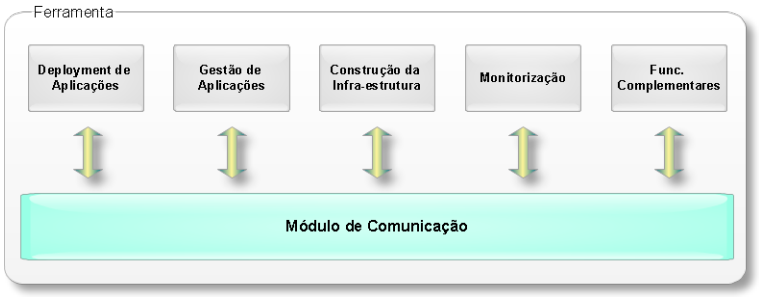


Figura 6.1: Arquitectura da Ferramenta

Dividido em seis secções, este capítulo apresenta uma descrição do protótipo desenvolvido. Na primeira secção é explicado como é feita a comunicação remota entre a ferramenta e os servidores que compõem, ou irão compor a infra-estrutura. A segunda secção aborda a temática da construção da infra-estrutura e as decisões que tiveram de ser tomadas a este nível, mostrando a necessidade de automatizar este processo. A terceira secção tem como objectivo apresentar todo o processo de deployment e gestão de aplicações. A quarta secção visa uma análise profunda dos mecanismos utilizados para a monitorização da infra-estrutura e das aplicações. Na quinta secção apresentamos algumas funcionalidades complementares presentes na ferramenta que visam auxiliar, ainda

mais, o utilizador. Por último é apresentada a área de administração da ferramenta e respectivas funcionalidades.

6.1 Comunicação Remota

Uma parte importante para ser possível a construção da infra-estrutura é a comunicação entre a ferramenta e os servidores que vão fazer parte da infra-estrutura.

Como se pode verificar na Secção 4.6, existem vários protocolos utilizados para a comunicação com máquinas remotas, seja para a troca de ficheiros ou para a execução de comandos. O protocolo escolhido e que vai ser utilizado pela ferramenta é o `SSH`.

Uma das principais vantagens do `SSH` em relação aos outros protocolos abordados é a sua segurança, ao utilizar métodos criptográficos praticamente inquebráveis, se bem configurados, consegue-se manter uma ligação íntegra.

Existem várias implementações do protocolo `SSH` em java, designadamente:

- JTA - é um cliente telnet e `SSH` com suporte para um terminal VT100/ANSI. Esta implementa a versão 1.5 do protocolo `SSH`.

- JCTerm - um emulador de um terminal SSH2 com suporte para sftp e redireccionamento de portas, pode ser utilizado através de SOCKS5 ou proxy HTTP.

- JSch - uma implementação SSH2 com redireccionamento de portas e ambiente gráfico, transferência de ficheiros, entre outras funcionalidades.

- JSSH - biblioteca que implementa a versão 1.5 do protocolo `SSH`.

Como podemos verificar cada implementação tem as suas particularidades, seja a versão do protocolo `SSH` que implementa, seja o tipo de serviços que é possível desenvolver através dela.

Para estabelecer a comunicação entre a ferramenta Web e os servidores que vão fazer parte da infra-estrutura, foi escolhida a biblioteca JSch. Esta escolha assenta na variedade das funcionalidades disponibilizadas e uma vez que implementa a versão 2.0 do protocolo `SSH`. A implementação da versão 2.0 do protocolo `SSH`, em detrimento da versão 1.5, é relevante, na medida em que é mais segura que a 1.5 e mais flexível em termos de redireccionamento remoto [BSB05].

6.1.1 Módulo de Comunicação

Através do Módulo de Comunicação, a ferramenta é capaz estabelecer uma ligação segura a um determinado servidor remoto, que pode ser feita através de um proxy, HTTP ou SOCKS v5, caso seja necessário, e deste modo poder efectuar determinadas tarefas, nomeadamente:

- Executar comandos na máquina remota, e recolher os respectivos resultados;
- Copiar ficheiros da máquina onde está instalada a ferramenta, para a máquina remota e vice-versa;

Estas funcionalidades são de extrema importância para a construção da infra-estrutura, como será abordado nas secções seguintes.

Tendo em conta a eventual existência de situações em que a execução de determinadas tarefas obriguem à execução de vários comandos, que podem variar de acordo com os resultados obtidos, a ferramenta Web disponibiliza uma consola `SSH` que permite ao utilizador estabelecer uma ligação com uma máquina remota, como se estivesse no próprio terminal do seu computador. Desta forma, potencia-se uma maior flexibilidade na execução de tarefas de gestão e instalação de máquinas remotas.

6.2 Construção da Infra-estrutura

Sendo a infra-estrutura de extrema importância para qualquer aplicação Web, um dos objectivos da ferramenta desenvolvida é agilizar o processo de construção de uma infra-estrutura capaz de receber aplicações Web JEE.

Uma Infra-estrutura, como foi possível observar ao longo do Capítulo 4, baseia-se num conjunto de componentes de hardware e software, cuja a organização tem como objectivo garantir as especificações para que foi criada. Esta é normalmente constituída por diferentes camadas:

- Camada de Servidores Web
- Camada de Servidores Aplicacionais
- Camada de Base de Dados

De forma a construir cada uma destas camadas, numa fase inicial, o utilizador precisa de fornecer à plataforma informações importantes sobre os servidores que as vão alojar,

nomeadamente o IP da máquina, os dados necessários para o acesso SSH, informações sobre o proxy, caso seja necessário, para a ligação ao servidor e por fim a directoria onde vão ser instalados todos os pacotes necessários para colocar em funcionamento a parte da infra-estrutura em questão. Na Figura 6.2 está representado esquematicamente o processo de construção da infra-estrutura.

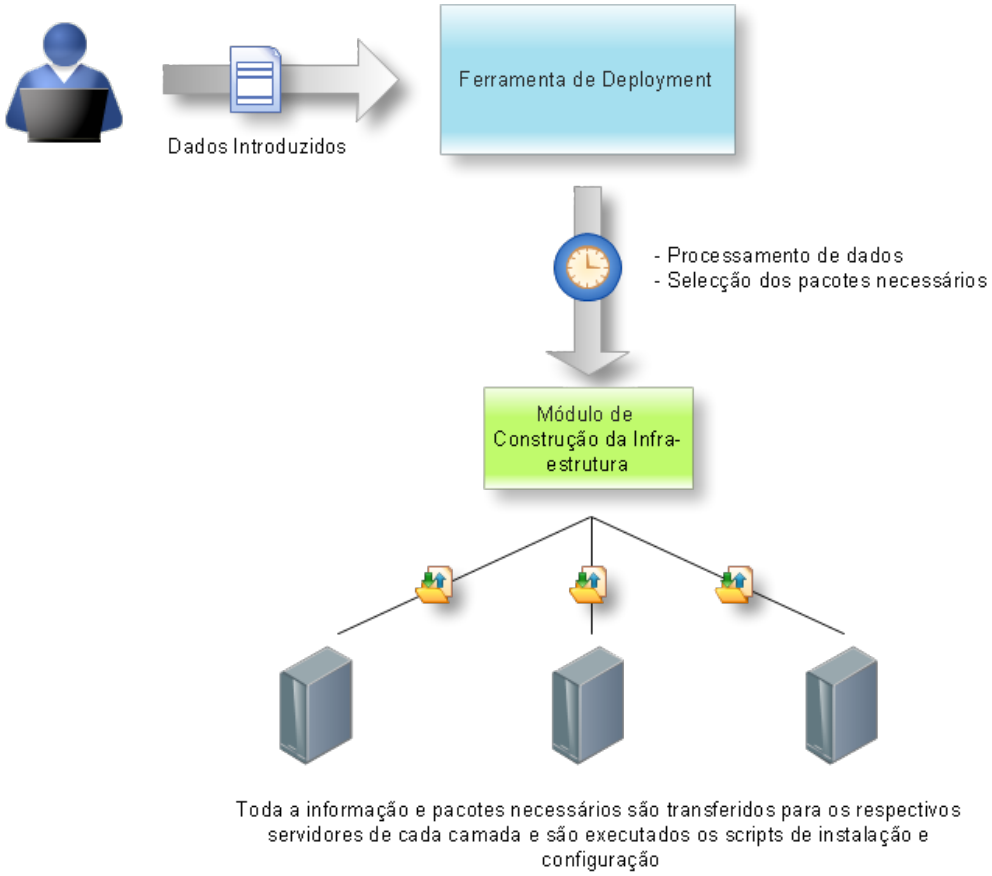


Figura 6.2: Construção da Infra-Estrutura

Podemos observar que, depois dos dados introduzidos pelo utilizador estes vão ser processados, a fim de se poder seleccionar os pacotes de software necessários e os scripts para a sua instalação e configuração. Após este processamento, todos os ficheiros necessários são transferidos para a máquina destino por SCP e são executados os scripts de instalação e configuração automáticos. Esta comunicação é feita através do Módulo de Comunicação da ferramenta, referido anteriormente.

6.2.1 Camada de Servidores Web

A camada de Servidores Web é a primeira camada a ter contacto com o pedidos efectuados pelos utilizadores/clientes, podendo alojar todo o tipo de conteúdo estático, nomeada-

mente, imagens, páginas HTML, entre outros.

Uma das tarefas que esta camada pode realizar é a distribuição dos pedidos recebidos que necessitam de receber algum tipo de processamento. Sem esta camada não seria possível fazer esta distribuição, uma vez que, a maior parte dos servidores aplicacionais não possui mecanismos de distribuição de carga dos pedidos, sendo esta apenas possível para aplicações que correm na máquina dos utilizadores.

De salientar que, apesar de ser importante, pelos factos apontados anteriormente, esta camada não é obrigatória. Os servidores aplicacionais também podem lidar com o tipo de conteúdo com que esta lida, deste modo, é dada total liberdade ao utilizador de escolher como quer a sua infra-estrutura. No entanto, o utilizador é avisado do perigo que envolve a não construção desta camada, uma vez que sem ela perde a capacidade de distribuição dos pedidos.

Configuração para a distribuição dos pedidos

De forma a ser possível a uma distribuição dos pedidos recebidos por parte da camada de servidores Web é necessário efectuar algumas configurações nos mesmos.

O primeiro passo a dar é adicionar ao servidor Web um *plugin*, de nome mod_jk, que irá gerir a comunicação entre o servidor Web e o Web container presente no servidor aplicacional. O mod_jk utiliza o conceito de *worker* para identificar as instâncias dos Web containers que constituem a camada de servidores aplicacionais. Cada uma delas é identificada pelo seu host, porta que utiliza e pelo protocolo utilizado para a troca de mensagens. Na Figura 6.3 é apresentado um pequeno exemplo do ficheiro, gerado pela ferramenta, que contem as informações, referidas acima, das instâncias pelas quais serão distribuídos os pedidos.

Convém frisar que os nomes atribuídos a cada nodo, no caso do ficheiro apresentado, **server1** e **server2**, têm de ser configurados da mesma forma no Web container correspondente.

Para finalizar a configuração, e tendo em conta que os servidores Web podem apresentar conteúdo estático, é necessário, ainda, configurar o mapeamento de URLs que serão redireccionados para os Web containers.

6.2.2 Camada de Servidores Aplicacionais

A camada de servidores aplicacionais é o local onde vão ser instaladas as aplicações JEE. Na Secção 5.2 foram apresentados diversos servidores aplicacionais, cada qual com características próprias, sendo que todos eles disponibilizam as mesmas funcionalidades básicas.

```
# ----------------
# Server2
# ----------------
worker.server2.port=8009
worker.server2.host=192.168.1.4
worker.server2.type=ajp13
worker.server2.lbfactor=1

# Load-balancing behaviour
worker.loadbalancer.type=lb
worker.loadbalancer.balanced_workers=server1,server2

# Status worker for managing load balancer
worker.status.type=status
```

Figura 6.3: Exemplo de ficheiro com as propriedades das Instâncias

Tendo em conta as suas necessidades, o utilizador, aquando da construção desta camada, deve indicar, para além dos dados de acesso à máquina remota que irá fazer parte desta camada, o servidor aplicacional escolhido de entre as várias opções disponibilizadas, nomeadamente JBoss, JonAS ou Apache Geronimo. A partir desta escolha, os scripts sofrem as adaptações necessárias à correcta instalação e configuração de cada nodo. De modo a melhor ilustrar como é feito o processamento dos dados para a configuração de um servidor aplicacional, apresentamos de seguida um pequeno caso de estudo.

Caso de Estudo

Suponhamos que durante a construção da infra-estrutura o utilizador coloca os dados apresentados na Figura 6.4.

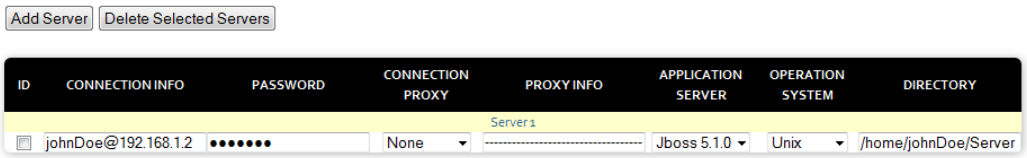


Figura 6.4: Caso de Estudo - Servidor Aplicacional

Pela análise dos dados presentes, sabemos que o utilizador escolheu para a sua infra-estrutura o servidor aplicacional JBoss e que este vai ser instalado e configurado num sistema Unix.

Após o processamento dos dados o comando a ser executado na máquina remota para a inicialização do JBoss será:

```
./run.sh -c all -g InfraTest -u 239.255.100.100 -b 192.168.1.2 -Djboss.messaging.ServerPeerID=1
```

A vermelho estão assinaladas as informações retiradas a partir da informação introduzida pelo utilizador. O nome dado ao cluster é o nome que o utilizador atribuiu à infra-estrutura, neste caso, *InfraTest*, que acompanha a opção **-g** do script de inicialização do servidor, de seguida é introduzido o endereço de host da máquina remota.

A opção **-u** define o endereço multicast utilizado para a comunicação dentro do cluster. Na última parte do comando está a ser atribuído um identificador único a cada nodo, de forma a ser possível a utilização do JBoss Messaging a nível do cluster. Neste caso a ferramenta atribui o número do servidor, que como podemos observer na figura, é o 1.

A questão que se coloca, neste caso em concreto, é o porquê de iniciar o servidor com o modo de cluster activo, uma vez que o utilizador apenas achou necessário ter um servidor aplicacional. Estando cientes da possibilidade de mais tarde ser necessário adicionar mais nós a esta camada da infra-estrutura, ao iniciar desta forma o servidor, permite a adição de novos nós sem ser necessário colocar o sistema em baixo, impedindo, desta forma, algum *downtime* ao mesmo.

De salientar que ao iniciar o servidor com esta configuração levará a um maior consumo de memória por parte deste, uma vez que o número de serviços que são inicializados é maior do que na configuração por defeito. Apesar disso, o custo do *downtime* associado a uma eventual necessidade de aumentar o número de servidores, na maioria dos casos, seria mais prejudicial do que uma eventual pequena perda de desempenho. No entanto, e de forma a ser o utilizador a tomar esta decisão aquando da submissão dos dados para a construção da infra-estrutura, caso a ferramenta detecte que a camada de servidores aplicacional apenas é constituída por um nó, é requerido ao utilizador que escolha a forma como este vai ser inicializado, podendo optar pela configuração por defeito, ou, então, pela configuração com o modo de cluster activo.

Configurações Adicionais

De modo a tirar todo o partido da ferramenta, são necessárias algumas configurações adicionais nos servidores aplicacionais.

Para efectuar uma monitorização remota aos servidores é essencial configurá-los de modo a permitirem acessos `JMX` remotos. Sem esta configuração não seria fácil obter dados como a memória que está a ser consumida ou as classes que estão carregadas no servidor de forma directa[1].

[1] A monitorização da infra-estrutura será aprofundada na Secção 6.4

```
set JONAS_OPTS=%JONAS_OPTS% -Djavax.xml.soap.SOAPFactory=com.sun.xml.messaging.saaj.soap.ver1_1.SOAPFactory1_1Impl
set JONAS_OPTS=%JONAS_OPTS% -Djavax.xml.soap.MetaFactory=com.sun.xml.messaging.saaj.soap.SAAJMetaFactoryImpl
set JONAS_OPTS=%JONAS_OPTS% -Djavax.xml.soap.MessageFactory=com.sun.xml.messaging.saaj.soap.ver1_1.SOAPMessageFactory1_1Impl

set JAVA_OPTS=%JAVA_OPTS% -Dcom.sun.management.jmxremote

Rem -----------------------------------------
Rem Get args
Rem -----------------------------------------

if [%1]==[]   goto no_arg

set ARGS=
```

Figura 6.5: Configuração JonAS

Existem duas formas para fazer os servidores aplicacionais aceitarem acessos `JMX` remotos. Uma das formas consiste em adicionar uma configuração directamente no ficheiro de inicialização do servidor aplicacional, como é possível observar através da Figura 6.5, que ilustra um excerto do ficheiro executável utilizado para a inicialização do servidor aplicacional JOnAS. A outra forma é adicionar ao script gerado pela ferramenta a seguinte opção:

```
-Dcom.sun.management.jmxremote
```

Existe, ainda, a necessidade de adicionar ao repositório de drivers, presente em cada servidor aplicacional, o driver responsável pela ligação ao middleware que pode estar presente na camada de dados, o Sequoia. Sem este, o utilizador não poderia usufruir da solução para atingir uma maior escalabilidade e disponibilidade na camada de dados construída pela ferramenta. De notar que todos os servidores aplicacionais presentes no repositório da ferramenta já se encontram equipados com este driver.

6.2.3 Camada de Dados

A camada de dados é de extrema importância, uma vez que, é o local onde a informação vai ser guardada. Em sistemas onde o volume de dados guardado é grande, esta camada pode ser um dos principais *bottleneck* do sistema, caso não esteja devidamente construída.

De entre todas as soluções para a obtenção de uma maior escalabilidade e disponibilidade da camada de dados estudadas na Secção 4.5 a que nos pareceu mais apropriada foi o Sequoia.

A capacidade do Sequoia de trabalhar com diferentes motores de base de dados, nomeadamente, Microsoft SQL Server, DB2, MySQL, PostgreSQL, Oracle, entre outros, contribui para a construção uma infra-estrutura modular. Desta forma a camada de dados não se encontra dependente de um determinado motor de base de dados e das tecnologias subjacentes a este, como aconteceria em alguns casos apresentados na secção 4.5.

Outra vantagem do Sequoia é a capacidade de distribuir a carga pelos diversos nós, melhorando, assim, o desempenho e a escalabilidade do sistema.

Middleware Sequoia

Como referido, o Sequoia é um middleware que permite a qualquer aplicação Java aceder, de forma transparente, a um cluster de base de dados através de `JDBC`. Desta forma, não é necessário realizar modificações na aplicação ou no motor de base de dados, bastando apenas que estes sejam acedidos pelo Sequoia.

Provido de uma arquitectura flexível, torna possível alcançar a escalabilidade e disponibilidade necessárias para o correcto funcionamento do sistema, implementando o conceito de *Redundant Array of Inexpensive Databases* (`RAIDb`) [CMZ05].

Podemos comparar o `RAIDb` com o conceito de RAID nos discos, sendo que a diferença está na utilização de motores de base de dados em vez dos discos rígidos. Este conceito esconde a complexidade da replicação, disponibilizando ao cliente um acesso único à base de dados. Os pedidos do cliente são direccionados ao controlador `RAIDb`, que, por sua vez, os distribui ao conjunto de sistemas de gestão de bases de dados que fazem parte do sistema. Existem três níveis básicos de `RAIDb` [CMZ05]:

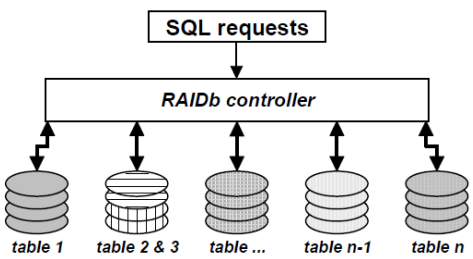
- RAIDb-0;
- RAIDb-1;
- RAIDb-2;

O RAIDb-0 consiste no particionamento das tabelas de uma determinada base de dados pelos vários nodos presentes. Uma tabela em si não pode ser dividida, mas tabelas diferentes podem estar em diferentes nós.

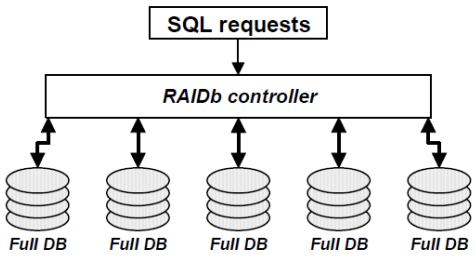
Esta configuração requer no mínimo dois `SGBD` e permite um melhor desempenho do que numa situação em que apenas um `SGBD` estivesse presente. No entanto, como não há duplicação da informação, não existe tolerância a falhas. A Figura 6.6(a) ilustra um exemplo da utilização do RAIDb-0.

A utilização do RAIDb-0 permite que grandes base de dados possam ser distribuídas, e dado não haver informação duplicada, a eficiência conseguida é melhor, quando comparada com os outros níveis de `RAIDb`.

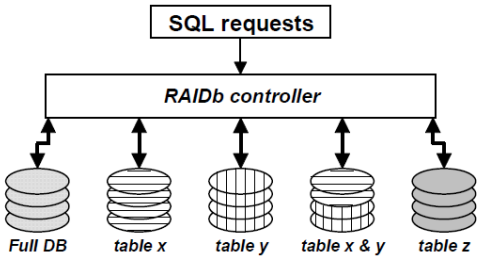
No caso do RAIDb-1, a base de dados é replicada na sua totalidade em todos os nós presentes. Este modelo de configuração é o que apresenta melhor tolerância a falhas, já que o sistema continuará disponível, desde que pelo menos um nó esteja activo. Uma



(a) RAIDb-0



(b) RAIDb-1



(c) RAIDb-2

Figura 6.6: Níveis de RAIDb [CMZ05]

desvantagem desta estratégia é que não há melhoria nos processos de escrita (UPDATE, INSERT, DELETE), uma vez que todos estes tipos de pedidos têm de ser enviados para todos os nós, a fim de se obter cópias coerentes. A Figura 6.6(b) ilustra um exemplo da utilização do RAIDb-1.

Por fim temos o RAIDb-2 que é uma combinação do RAIDb-0 e RAIDb-1. Existe a replicação parcial da base de dados, de forma a diminuir a degradação no processo de replicação de cada tabela, possibilitando, assim, a obtenção de melhores resultados de leitura e escrita. Este modelo requer que cada tabela se encontre disponível em pelo menos 2 nós, e ao contrário do RAIDb-1, não é obrigatório a presença de uma cópia completa da base de dados num dos nós. A Figura 6.6(c) ilustra um exemplo da utilização do RAIDb-2.

Com estes mecanismos presentes no Sequoia podemos atingir o nível de escalabilidade e disponibilidade pretendidos, sendo, todavia, necessário ter sempre presente que tipo aplicação vai estar presente no sistema, uma vez que o seu tipo irá condicionar o mecanismo de `RAIDb` a ser escolhido. A título de exemplo, para uma aplicação onde os dados são críticos, o mecanismo de RAIDb-0 deve ser excluído devido à inexistência de tolerância a falhas.

Configuração do Sequoia

A configuração do Sequoia é de extrema importância, pois é nesta fase que definimos, por exemplo, o tipo de replicação a ser utilizado e os servidores que fazem parte do cluster construído.

A primeira configuração a ser efectuada, é relativa ao controlador que irá fazer a comunicação com a aplicação Web. Neste ficheiro são definidas, para além de outras configurações, o número da porta para onde devem ser enviados os pedidos para, após processamento, serem enviados para cada nó contendo a base de dados.

Na Figura 6.7 podemos observar um pequeno exemplo de um ficheiro de configuração do controlador do Sequoia.

```
<SEQUOIA-CONTROLLER>
<Controller port="25222">
<Report hideSensitiveData="true" generateOnShutdown="true" generateOnFatal="true" enableFileLogging="true" />
<JmxSettings>
<RmiJmxAdaptor/>
</JmxSettings>
</Controller>
</SEQUOIA-CONTROLLER>
```

Figura 6.7: Sequoia - Ficheiro de Configuração do Controlador

No exemplo apresentado, o controlador irá gerar automaticamente um relatório sempre que ocorrer um erro ou ao desligar-se, isto é configurado pela tag <Report>. A administração remota, através de JMX, também foi contemplada neste controlador Todos os ajustes neste ficheiro de configuração do controlador, irão ser aplicadas a todas as base de dados geridas pelo mesmo.

Após a configuração do controlador, é necessário configurar as base de dados virtuais que irão ser utilizadas pelas aplicações.

Na Figura 6.8 podemos observar um pequeno exemplo de um ficheiro de configuração de uma Base de Dados Virtual.

Neste exemplo começamos por atribuir um nome à nossa base de dados virtual e configuramos os utilizadores e seus privilégios. Podemos, ainda, verificar a presença de dois nodos definidos pela tag <DatabaseBackend>, cada um com o SGBD Mysql. O sistema de replicação utilizado é o RAIDb-1, como podemos verificar através da tag <RequestScheduler>.

Constata-se facilmente que a configuração do Sequoia pode ser bastante complexa e morosa, dado aos extensos ficheiros que podem ser criados. A ferramenta apresentada, tenta de alguma forma agilizar todo este processo, gerando estes ficheiros de acordo com os dados introduzidos pelo utilizador.

```
<SEQUOIA>
  <VirtualDatabase name="dbteste">
    <AuthenticationManager>
      <Admin>
        <User username="admin" password="password" />
      </Admin>
      <VirtualUsers>
        <VirtualLogin vLogin="user" vPassword="password" />
      </VirtualUsers>
    </AuthenticationManager>
    <DatabaseBackend name="node1" driver="com.mysql.jdbc.Driver"
    url="jdbc:mysql://192.168.1.60:3306/dbteste" connectiondbtesteStatement="select 1">
      <ConnectionManager vLogin="user" rLogin="dbteste" rPassword="">
        <VariablePoolConnectionManager initPoolSize="10"
        minPoolSize="5" maxPoolSize="50" idleTimeout="30"
        waitTimeout="10" />
      </ConnectionManager>
    </DatabaseBackend>
    <DatabaseBackend name="node2" driver="com.mysql.jdbc.Driver"
    url="jdbc:mysql://192.168.1.61:3306/dbteste" connectiondbtesteStatement="select 1">
      <ConnectionManager vLogin="user" rLogin="dbteste" rPassword="">
        <VariablePoolConnectionManager initPoolSize="10"
        minPoolSize="5" maxPoolSize="50" idleTimeout="30"
        waitTimeout="10" />
      </ConnectionManager>
    </DatabaseBackend>
    <RequestManager>
      <RequestScheduler>
        <RAIDb-1Scheduler level="passThrough" />
      </RequestScheduler>
```

Figura 6.8: Sequoia - Ficheiro de Configuração da Virtual DB

Motores de Base de Dados

Para além da instalação e configuração do Sequoia é necessário configurar cada um dos nodos que vão fazer parte da camada de dados. Após o utilizador ter submetido todos os dados referentes aos nodos que deseja para a camada de dados, estes são sujeitos a uma analise e um tratamento, a fim de estabelecer a ligação com cada um dos servidores que vão alojar os motores de base de dados.

Depois de estabelecida a ligação, são copiados dois ficheiros, o binário do `SGBD` e o script de instalação do mesmo, estando estes de acordo com o sistema operativo indicado pelo utilizador.

Na Figura 6.9 está representado um excerto do script standard para a instalação e configuração de um nodo com PostgreSQL.

Como podemos constatar, uma das tarefas a efectuar, para além da escolha da directoria alvo para a instalação, é a criação de um grupo de utilizadores e de um utilizador. Após a instalação e a configuração do `SGBD`, o script inicializa o mesmo.

```
#!/bin/bash

tar -xjvf $2

cd postgreslq
./configure --prefix=/usr/local/postgres \
--bindir=/usr/bin --sysconfdir=/etc/postgres

gmake
su
gmake install

if [ $1 -eq 1 ]
then
groupadd postgres
adduser postgres
fi
```

Figura 6.9: Script de Instalação do PostgreSQL em Unix

6.2.4 Gestão de Scripts

Ao longo desta secção foi perceptível a importância que os scripts disponibilizados pela ferramenta têm para a correcta instalação e configuração da infra-estrutura.

Tendo em conta as limitações que causaria a existência de apenas um script de configuração standard, é facultada a oportunidade ao utilizador de editar esses mesmos scripts, tornando, desta forma, mais flexível e personalizada a instalação e configuração dos servidores.

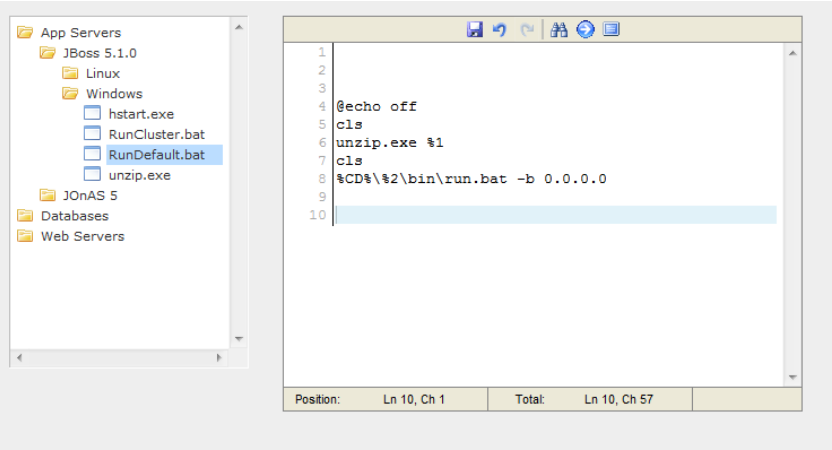


Figura 6.10: Gestão de Scripts

Quando o utilizador decide modificar algum dos scripts são lhe apresentados todos os scripts presentes na ferramenta, divididos por tipos, e, ainda, um pequeno editor para efectuar as devidas alterações, tal como está representado na Figura 6.10.

6.3 Deployment e Gestão de Aplicações

Com o aumento da complexidade e dimensão das aplicações, torna-se evidente que a dificuldade do processo de deployment e a gestão das próprias aplicações aumenta consideravelmente. Deste modo, é indispensável a existência de ferramentas que auxiliem estes processos.

Para além de tentar agilizar, o mais possível, a construção da infra-estrutura, é objectivo da ferramenta simplificar o processo de deployment e disponibilizar mecanismos de controlo e coordenação a fim de melhorar a gestão de aplicações.

6.3.1 Módulo de Deployment

De forma a disponibilizar a aplicação, o utilizador necessita de introduzir uma série de informações, que vão permitir não só a correcta instalação da mesma, mas, também, a sua fácil identificação, por parte deste, aquando da gestão das aplicações.

Em primeiro lugar, o utilizador deve introduzir a designação e versão da aplicação. De seguida, é necessário especificar se a aplicação em causa possui base de dados e, se for esse o caso, o utilizador pode especificar o nome de um ficheiro que irá fazer upload e contém as informações necessárias sobre a base de dados, ou, então, escolher um ficheiro gerado a partir do modelo de dados[2] criado pelo utilizador. De realçar que se a infra-estrutura não possuir base de dados, a ferramenta não permite escolher de nenhum destes métodos, avisando o utilizador da não existência de uma camada de dados na infra-estrutura seleccionada.

Finalmente, o utilizador indica quais os ficheiros que fazem parte da aplicação.

Na Figura 6.11 está ilustrado o formulário que o utilizador deve preencher com as informações acima descritas.

Tendo em conta que o ficheiro gerado a partir do modelo de dados tenta ser o mais genérico possível, é dada a oportunidade ao utilizador de o editar, permitindo torná-lo mais específico, de acordo com o motor de base de dados presentes na infra-estrutura. Na Figura 6.12 está representado o modo de edição de um ficheiro gerado a partir de um modelo de dados.

Numa segunda fase, o utilizador escolhe de que forma a aplicação vai ser distribuída, ou seja, pode especificar se uma determinada parte da aplicação fica num só servidor da infra-estrutura, ou se esta vai ser disponibilizada em todo o cluster.

[2]A ferramenta disponibiliza meios para criar o modelo de dados de uma aplicação. Este tema será abordado mais à frente.

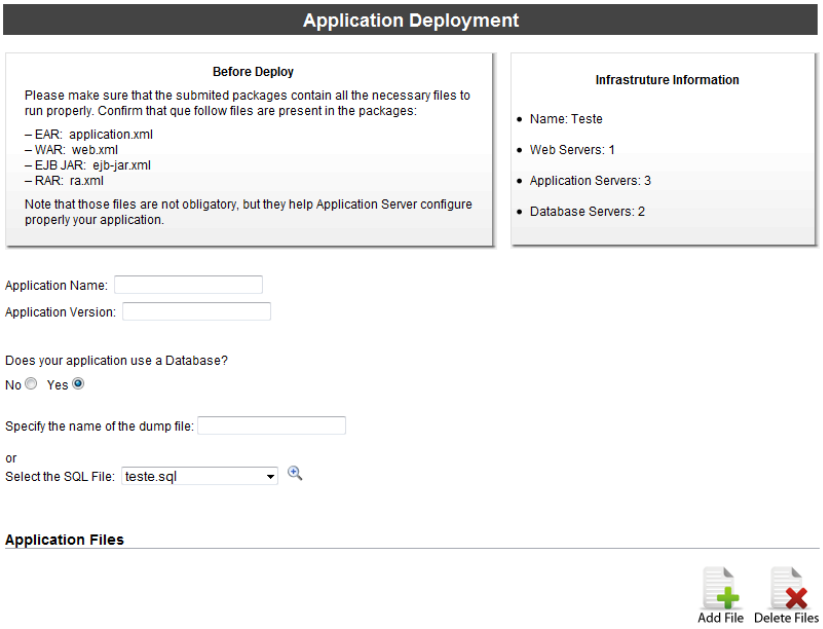


Figura 6.11: Formulário para o deployment de uma aplicação

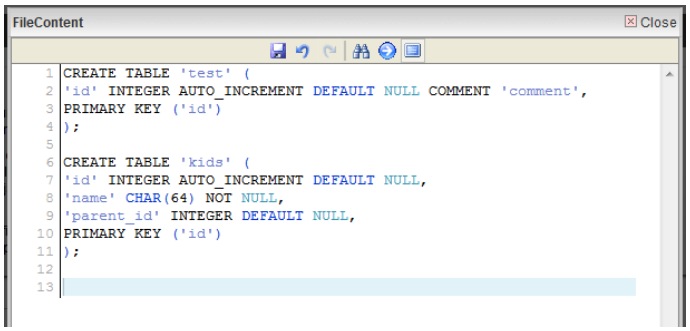


Figura 6.12: Edição do script SQL

Depois de todas as informações serem introduzidas, a ferramenta faz uso do Hot Deployment de forma manual[3] presente nos servidores aplicacionais compatíveis com Java EE. A escolha desta estratégia deve-se, na sua maioria, à independência que esta permite em relação às implementações dos servidores aplicacionais e à segurança que se obtém mediante a utilização de mecanismos de cópia segura de ficheiros (`SCP`).

6.3.2 Módulo de Gestão

O módulo de gestão presente na ferramenta, tem como objectivo garantir ao utilizador o controlo e coordenação das aplicações presentes na infra-estrutura. A partir deste módulo o utilizador pode identificar todas as aplicações presentes na infra-estrutura.

Dado que uma aplicação JEE pode estar repartida pelos vários servidores aplicacionais

[3] Ver Secção 5.4

presentes na infra-estrutura, consideramos pertinente facultar a possibilidade ao utilizador de explorar especificamente um determinado servidor.

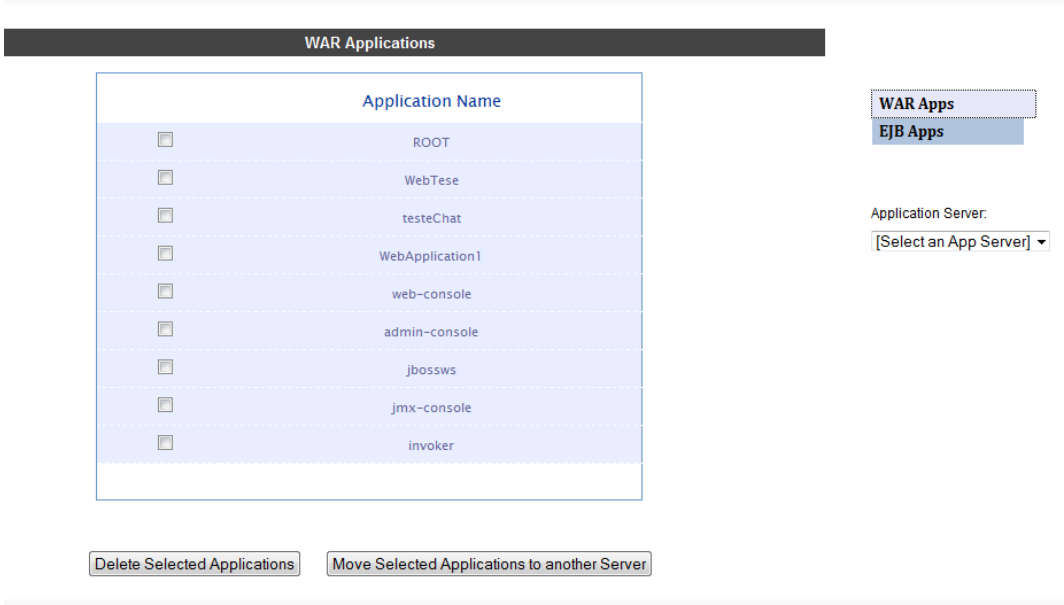


Figura 6.13: Gestão de Aplicações

Como podemos observar, na Figura 6.13, são listados todos os ficheiros, organizados por tipo, que neste caso em concreto são WAR ou `EJB`, presentes no servidor escolhido. É dada ao utilizador a possibilidade de eliminar ou mover ficheiros para outro servidor aplicacional presente na infra-estrutura. Este último caso é extremamente útil quando, através da monitorização do servidor, se chega à conclusão que este está a ser sobrecarregado por pedidos, havendo, portanto, a necessidade de mover alguns ficheiros para outro servidor de modo a equilibrar a carga.

6.4 Monitorização e Gestão da Infra-Estrutura

Como tem sido referido ao longo deste trabalho, a monitorização é um aspecto que tem vindo a ganhar popularidade à medida que as aplicações e as infra-estruturas se tornam mais complexas.

A monitorização de uma qualquer aplicação ou infra-estrutura implica uma observação do seu comportamento e respectiva evolução de modo a possibilitar um melhor controlo da mesma, sendo, para tal, necessário tomar uma decisão quanto ao nível de detalhe pretendido. É possível a monitorização ao nível aplicacional, do sistema operativo ou, até, ao nível de hardware. Neste último caso, a obtenção de resultados depende da existência de dispositivos de recolha de informação, designadamente, sensores de temperatura,

contadores no processador, monitores no disco, entre outros. A partir dos dados obtidos, o controlo de uma aplicação pode passar por várias medidas que podem ir desde a reconfiguração da infra-estrutura ou da própria aplicação. Estes dados podem, ainda, ser armazenados em suporte persistente, o que permite uma análise *a posteriori* da informação, fornecendo indicadores relevantes das zonas críticas ou, ainda, detectar erros de performance em casos específicos. Este estudo pode evitar erros nas soluções futuras, tornando-as mais sólidas.

A dificuldade na construção de um sistema de monitorização advém, sobretudo, da diversidade de elementos que podem estar presentes numa infra-estrutura JEE. De facto, para cada camada são necessários diferentes mecanismos de modo a obter os valores desejados.

Uma das formas mais utilizadas para a monitorização, quer de servidores aplicacionais, quer de aplicações JEE é a especificação *Java Management Extensions* (`JMX`), que passamos a descrever.

6.4.1 Java Management Extensions

A especificação Java Management Extensions define uma arquitectura, APIs e serviços para a gestão e monitorização de aplicações, serviços, entre outros. Esta, foi desenhada com o propósito de responder às necessidades de gestão dos utilizadores da tecnologia Java, mantendo a compatibilidade com standards de gestão já existentes, nomeadamente, o *Simple Network Management Protocol* (`SNMP`) [Per02].

Arquitectura

A arquitectura `JMX` é composta por três níveis, instrumentação, agente e gestão, tendo cada nível um papel distinto na arquitectura [JL02]. Esta encontra-se desenhada de forma a possibilitar a monitorização dos recursos existentes e torná-los visíveis a aplicações externas.

Na Figura 6.14 estão representados os componentes fundamentais desta arquitectura.

A camada de instrumentação descreve os procedimentos e requisitos para se desenvolver recursos com capacidade de gestão de acordo com a norma `JMX`. Estes recursos que podem ser, por exemplo, uma aplicação ou um serviço, e são representados como um ou mais objectos Java denominados Mbeans.

Os MBeans são classes Java não abstractas que implementam uma interface de administração que pode ser estática ou dinâmica [JL02], tendo de seguir os padrões e as interfaces definidas na especificação `JMX`. Cada MBean possui atributos, construtores, op-

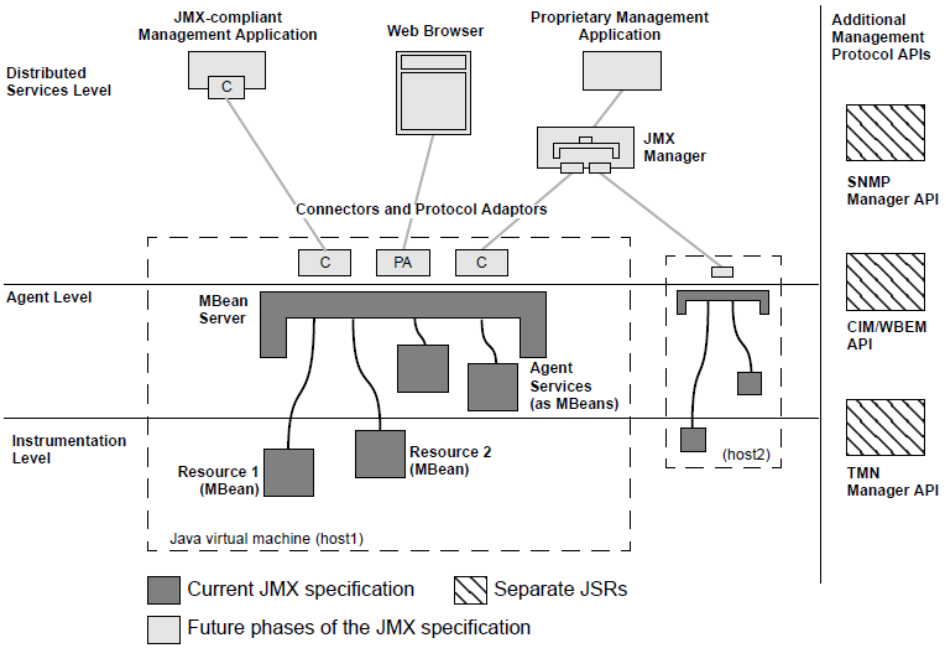


Figura 6.14: Arquitectura JMX [Mic08]

erações e mecanismos de notificação [Per02]. Os atributos contêm o estado interno do MBean, podendo ser unicamente de leitura, de escrita ou englobar ambos. Os construtores são utilizados para criar instâncias do recurso. As operações permitem executar determinadas acções no objecto, podendo receber parâmetros. Os mecanismos de notificação permitem informar as aplicações de gestão ou outros agentes `JMX` sobre alterações no MBean.

É através dos MBeans pré-existentes que obtemos informações sobre a utilização de memória, threads, especificação sobre o servidor aplicacional, entre outras. De salientar, que é possível desenvolver MBeans específicos que permitam expor ou configurar parâmetros, de uma aplicação específica, em tempo de execução.

O nível agente da arquitectura `JMX` é responsável por todos os MBeans, sendo o seu principal componente o servidor de MBeans, como podemos observar na Figura 6.14.

No servidor de MBeans estão registados os MBeans presentes no nível de instrumentação, possuindo cada um destes um identificador próprio. A este compete, ainda, a responsabilidade de prover os serviços necessários à manipulação dos MBeans, podendo ser acedido local ou remotamente.

Uma das responsabilidades deste nível é separar as aplicações de gestão e as instâncias dos MBeans, funcionando como um intermediário para a realização desta comunicação e permitindo a manipulação dos MBeans através de um conjunto de protocolos expostos

por conectores ou adaptadores no nível de distribuição.

Além disso, este nível disponibiliza quatro serviços que têm como finalidade facilitar a gestão dos MBeans:

- Monitorização - um monitor observa o valor de um atributo de um MBean, seja ele um valor numérico ou uma string, e pode notificar as entidades que estão à registadas nesta monitorização.
- *Timer* - este serviço tem como objectivo enviar notificações em intervalos específicos de tempo.
- Carregar dinamicamente MBeans - através do M-Let (applet de gestão) é possível carregar MBeans de qualquer lado, através da rede.
- Serviço de Relações - fornece um mecanismo para associações entre MBeans, formalizando as regras para a descrição e manutenção dessas mesmas relações.

Já o nível de gestão é responsável por tornar os agentes `JMX` disponíveis para o exterior. É constituído pelos adaptadores para os vários protocolos, nomeadamente, HTTP e `SNMP`, como podemos observar na Figura 6.14, e por conectores standard, tornando os agentes JMX acessíveis a aplicações de gestão remota.

Para além dos três níveis da arquitectura `JMX` já apresentados, esta fornece, ainda, um modelo de notificações semelhante ao modelo de eventos presente no Java. A partir do modelo de notificações é possível disparar alertas ou enviar relatórios para a pessoa ou ferramenta responsável pela gestão da aplicação, de acordo com a ocorrência de determinados eventos.

6.4.2 Módulo de Monitorização e Gestão

O objectivo deste módulo é recolher e agrupar os dados necessários à monitorização e gestão de uma infra-estrutura.

Numa primeira etapa é mostrado ao utilizador uma visão global da infra-estrutura seleccionada. Aqui o utilizador pode saber quais os servidores que se encontram em funcionamento e quais, por alguma razão adversa, não respondem dentro do intervalo configurado. Esta visão global da infra-estrutura está representada na Figura 6.15. Para além disso, caso seja necessário, é possível adicionar ou remover elementos da infra-estrutura.

Para além das informações gerais sobre a infra-estrutura podemos obter dados mais concretos de um determinado servidor.

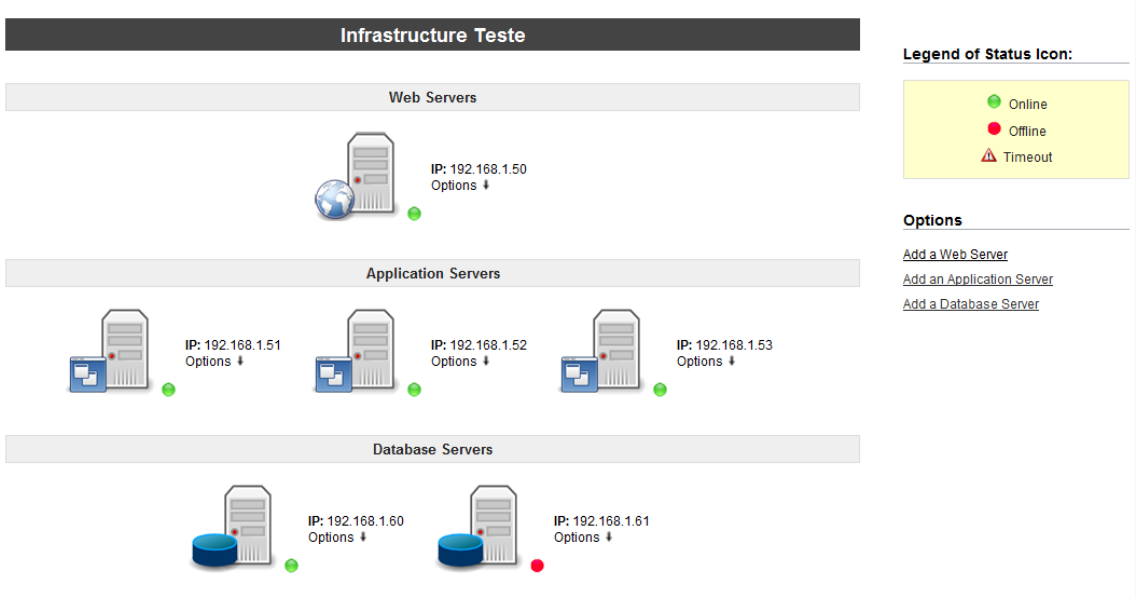


Figura 6.15: Monitorização da Infra-Estrutura

Servidor Aplicacional

De forma a obter os dados necessários para a monitorização de um servidor aplicacional utilizamos a especificação `JMX`, que é bastante utilizada para a gestão e monitorização de servidores aplicacionais e aplicações JEE.

Na Figura 6.16 podemos observar um vasto leque de informações sobre o servidor, concretamente algumas características do Sistema Operativo instalado no servidor, informações sobre a máquina virtual Java, a memória que está a ser consumida, entre outras.

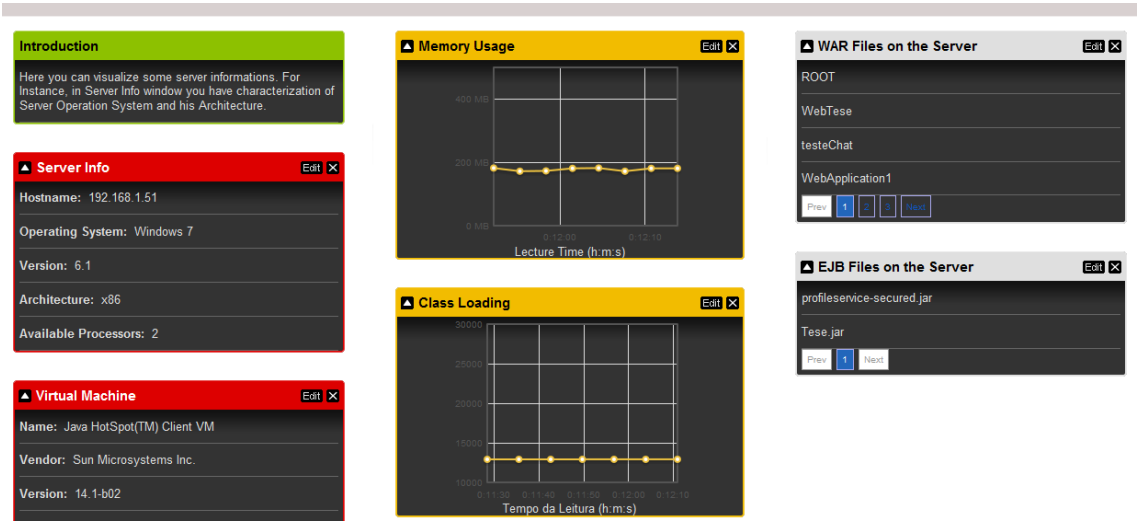


Figura 6.16: Monitorização de um Servidor Aplicacional

Além disso, são listados os ficheiros, organizados por tipo, que se encontram no re-

spectivo servidor.

De modo a tornar mais intuitiva, esta divisão de informação, optámos pela utilização de uma espécie de código de cores, que torna perceptível quais os dados que estão a ser constantemente actualizados, os dados referentes às aplicações e, ainda, dados estáticos referentes aos servidor. Esta codificação é visível na Figura 6.16.

De referir que uma das dificuldades em obter algumas das informações presentes na figura prende-se com o facto de estas serem obtidas por MBeans específicos que variam de acordo com o servidor aplicacional utilizado.

Servidor de Base de Dados

Ao contrário do que acontece com os servidores aplicacionais, onde através da especificação `JMX`, conseguimos obter grande parte da informação pretendida de forma standard, no caso dos motores de base de dados não é bem assim. Para obtermos as mesmas informações temos de executar comandos ou *querys* diferentes, o que torna extremamente complexo o desenvolvimento de um sistema de monitorização que obtenha os mesmos dados com motores de base de dados diferentes.

De modo a tentar contornar esta dificuldade, a quando da introdução de um novo `SGBD` na ferramenta são pedidas algumas informações relativas às querys ou comandos a executar para obter determinados dados, nomeadamente, as base de dados existentes, o tipo de configuração das mesmas, entre outras. De salientar que a quantidade de dados apresentados pode variar de acordo com o `SGBD`.

Na Figura 6.17 estão representadas as informações obtidas do Mysql. Como podemos observar, temos uma listagem das base de dados que estão neste servidor específico e, ainda, listas de vários parâmetros sobre o estado do `SGBD` e sua configuração.

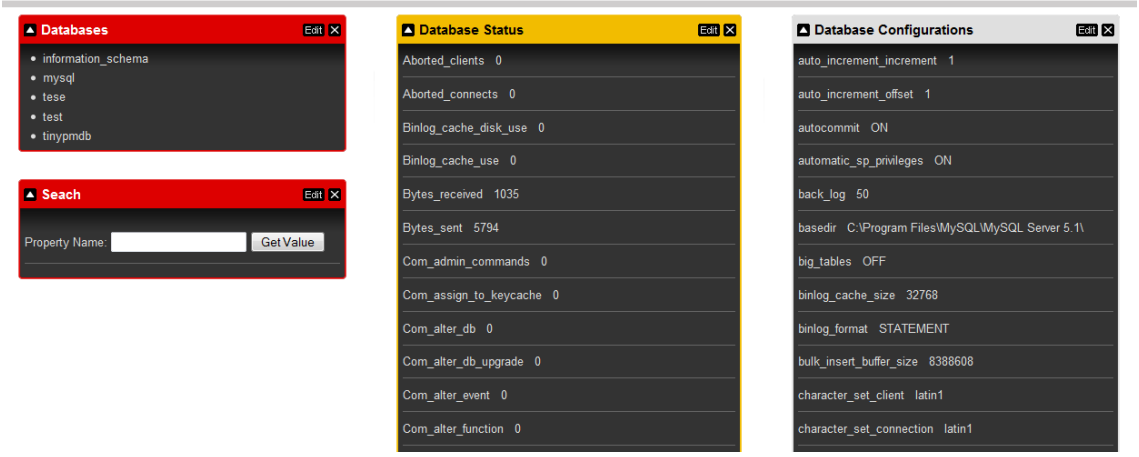


Figura 6.17: Monitorização de um Servidor de Base de Dados

Como é perceptível pela figura quer a listagem referente ao estado do SGBD, quer a referente à sua configuração, são bastantes extensas, não se encontrando, portanto, todos os elementos que as constituem representados na figura. A título informativo, a listagem do estado do SGBD tem cerca de 20 páginas, sendo que cada página contém 15 elementos.

Dado que uma procura por um determinado valor pode ser extremamente moroso, é dada a possibilidade ao utilizador de procurar um parâmetro específico através de um mecanismo de sugestão automática, que vai permitir ajudar na procura, uma vez que ao serem introduzidas as primeiras letras do nome do parâmetro, a ferramenta sugere alguns parâmetros que vão de acordo com aquilo que o utilizador pode estar a procurar. A Figura 6.18 representa um pequeno exemplo desse mecanismo de ajuda.

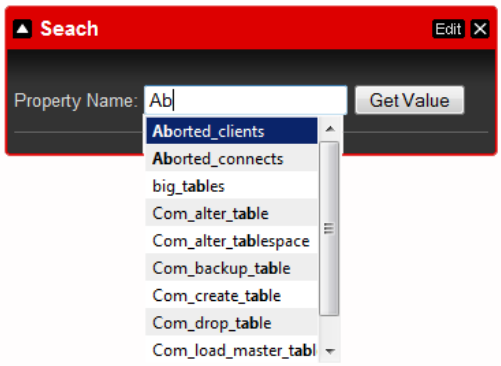


Figura 6.18: Mecanismo de procura de Parâmetros do SGBD

6.5 Funcionalidade Complementares

Tendo em mente que a ferramenta deve auxiliar o mais possível o utilizador, para além das funcionalidades anteriormente mencionadas, consideramos que seria uma mais valia adicionar outras à ferramenta. De seguida, são identificadas e descritas essas mesmas funcionalidades e de que modo podem auxiliar o utilizador a desempenhar tarefas como a gestão da infra-estrutura e das aplicações.

Consola SSH

Sabendo à partida que não é de todo possível colocar na ferramenta todas as tarefas que o utilizador possa vir a precisar, optámos por colocar uma consola SSH embutida na própria ferramenta.

A partir desta consola, o utilizador pode aceder a qualquer um dos servidores que fazem parte das infra-estruturas que construiu, podendo efectuar tarefas que, de outro modo não

seriam possíveis através da ferramenta, como, por exemplo, consultar os ficheiros de log do sistema operativo instalado no servidor escolhido.

De salientar, ainda, que para implementar esta funcionalidade foi utilizado o JCTerm[4], já descrito anteriormente.

Cliente SVN

O aumento da complexidade e da dimensão das aplicações, que temos vindo a abordar ao longo deste trabalho, e a crescente utilização de processos de desenvolvimento iterativos e incrementais trouxe a necessidade de um controlo mais rigoroso de todos os protótipos e versões finais da aplicação. Deste modo, os sistemas de controlo de versões assumem um papel de maior importância.

Um dos sistema de controlo de versões mais utilizados é o *Subversion* (`SVN`). Este é usado para manter o histórico de versões dos ficheiros que fazem parte de uma aplicação, mantendo um historial de todo o trabalho e alterações efectuadas num conjunto de ficheiros pelos membros das equipas de desenvolvimento.

Para implementar o cliente `SVN`, foi utilizada a biblioteca svnKit[5]. O svnKit é uma biblioteca totalmente implementada em Java que permite todas as interacções necessárias para aceder e manipular repositórios `SVN`.

Para efectuar a ligação, o utilizador deve introduzir o endereço do repositório `SVN`, o username e a password a utilizar, que, por defeito, são ambos *anonymous* , e ainda os dados do proxy, caso a ligação o exija. Na Figura 6.19 temos um exemplo de uma ligação a um determinado repositório estabelecida com sucesso.

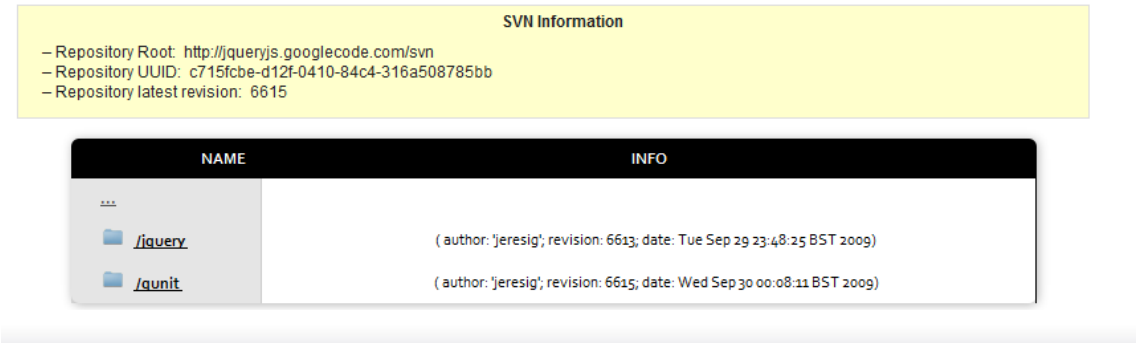
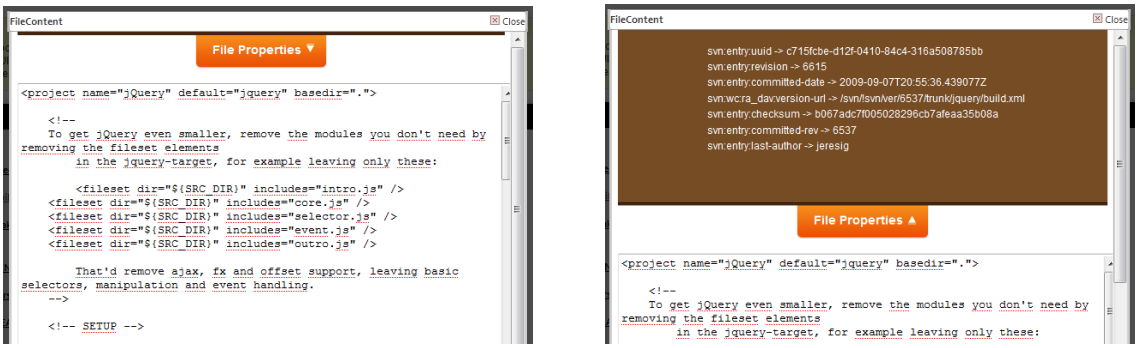


Figura 6.19: Ligação com um repositório SVN

Podemos ter informações das pastas e ficheiros que contém, bem como, quem foi a última pessoa a submeter uma alteração e quando esta ocorreu. No caso dos ficheiros, é

[4] Ver Secção 6.1
[5] http://svnkit.com/

dada a possibilidade de visualizar o seu conteúdo, bem como, mais informação referente ao mesmo, como podemos observar através da Figura 6.20.



(a) Conteúdo do ficheiro (b) Propriedades do ficheiro

Figura 6.20: Visualização do conteúdo de um ficheiro e suas propriedades

A integração de um cliente `SVN` na ferramenta vai permitir dar um maior relevo à fase de deployment de um qualquer processo de desenvolvimento de software.

Modelo de Dados

O desenho de bases de dados e a criação de modelos de dados é uma parte importante do ciclo de vida do desenvolvimento de aplicações.

A modelação dos dados consiste numa série de análises conceptuais e lógicas com o objectivo de encontrar a melhor disposição possível para o armazenamento e manutenção das informações na base de dados. Esta modelação implica a identificação das entidades do domínio, a caracterização dessas entidades através da definição dos seus atributos e a definição dos relacionamentos entre as entidades do caso em estudo.

Tendo em conta a sua importância, consideramos pertinente adicionar à ferramenta mecanismos que permitam não só a criação de Diagramas Entidade-Relacionamento, mas também, a geração, a partir destes, dos scripts SQL necessários para a criação da base de dados.

Com o auxílio da ferramenta open source WWW SQL Designer[6] foi implementado um ambiente que permite ao utilizador criar entidades, definir os seus atributos e, ainda, relacioná-las entre si, permitindo, posteriormente, gerar e guardar os scripts SQL.

A geração do script SQL a partir do diagrama entidade-relacionamento criado está ilustrada na Figura 6.22.

De notar que o utilizador deve indicar qual o `SGBD` que pretende utilizar, uma vez que o SQL gerado pode variar mediante esta escolha. Os `SGBD` para os quais o utilizador pode

[6]http://code.google.com/p/wwwsqldesigner/

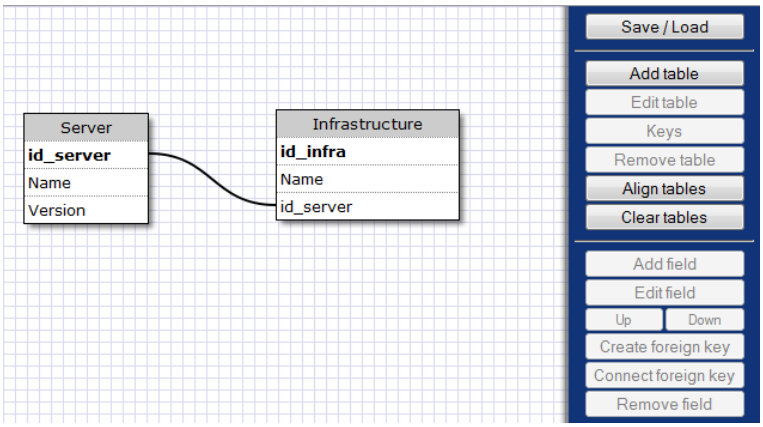


Figura 6.21: Desenho de um Modelo de Dados

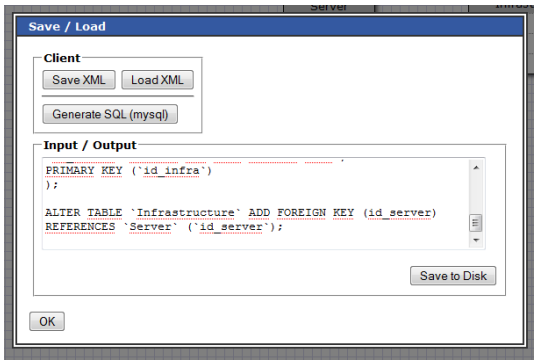


Figura 6.22: Geração dos scripts SQL

gerar o script SQL são:

- Mysql;
- PostgreSQL;
- Microsoft SQL Server;
- SQLite;

No entanto, e sabendo à partida que as diferenças entre os scripts gerados para os diferentes SGBD não são muito acentuadas, não será difícil ao utilizador modificar o script[7], adaptando-o para um `SGBD` que não esteja presente.

A integração de novos `SGBD` pode ser feito através de ficheiros descritores XML que contêm várias informações sobre o mesmo, nomeadamente, os tipos de dados suportados e a forma como deve ser gerado o SQL.

[7] A modificação do script é feita de forma idêntica à situação apresentada na Secção 6.3

6.6 Administração da Ferramenta

Uma ferramenta cujo objectivo principal consiste em auxiliar o utilizador na construção e gestão de uma infra-estrutura e no deployment e gestão de aplicações JEE, necessita de ser modular e adaptável. Esta necessidade advém das constantes actualizações sofridas pelos componentes que fazem parte da infra-estrutura, ou, até, pelo aparecimento de novos servidores mais eficientes. Tendo em conta estes factos, é necessário que o utilizador seja capaz de, por exemplo, adicionar à ferramenta uma nova versão de um determinado servidor aplicacional, a fim de este estar disponível para uma próxima infra-estrutura. A pensar nisso foi elaborado um módulo de administração que será explanado de seguida.

6.6.1 Módulo de Administração

A partir do módulo de Administração são disponibilizadas, ao utilizador, todas as opções de administração da ferramenta. De salientar, que devido à sua importância, apenas os administradores da ferramenta têm acesso a este módulo.

Gestão dos Utilizadores

Na gestão de utilizadores é dada a possibilidade, ao administrador, de consultar os dados referentes aos utilizadores registados na ferramenta, bem como a capacidade de adicionar, editar ou remover utilizadores.

Sendo os utilizadores os principais intervenientes, e sabendo à partida que deve haver uma distinção entre eles, tivemos a preocupação de criar perfis para cada utilizador. Assim na ferramenta podemos distinguir três tipos de utilizadores:

- Administrador - responsável pela gestão da ferramenta, devendo garantir o correcto funcionamento da mesma. Este pode criar, bloquear ou eliminar utilizadores, estando, ainda, encarregue da gestão do repositório dos diferentes tipos de servidores, podendo adicionar novos ou eliminar os existentes. Em síntese, um administrador tem acesso a todas as funcionalidades presentes na ferramenta.

- Gestor de Infra-estruturas - responsável pela gestão, disponibilização e configuração da infra-estrutura que irá receber as aplicações. Possui acesso a todas as funcionalidades da ferramenta, à excepção do módulo de Administração;

- Gestor de Deployment - responsável por planear e efectuar o deployment de uma aplicação, assegurando que esse processo é executado de forma apropriada. Este tem, ainda, acesso ao cliente `SVN` e a todos os mecanismos de monitorização.

Gestão dos Servidores Web

Na gestão dos Servidores Web, para além da possibilidade de consultar os dados referentes a estes, o administrador pode adicionar ou remover elementos. Na introdução de um novo servidor, além das informações normais, como o nome, versão e o próprio executável, é necessário especificar os comandos de arranque deste, e, ainda, os ficheiros que devem ser alterados de forma a possibilitar a distribuição da carga dos pedidos pela camada de servidores aplicacionais, caso esta funcionalidade esteja disponível.

No caso concreto de o novo servidor Web não permitir a distribuição de carga, o utilizador é avisado que a infra-estrutura pode ficar condicionada.

Gestão Servidor Aplicacional

Para a gestão dos servidores aplicacionais é dada a possibilidade ao administrador de consultar as informações referentes a um dado servidor aplicacional, eliminar o seu registo e todos os ficheiros associados e, ainda, adicionar um novo servidor aplicacional.

Ao adicionar um novo servidor aplicacional, o administrador tem de introduzir toda a informação necessária para o correcto funcionamento do mesmo. De entre essa informação destacam-se os comandos necessários para executar o servidor em modo standard e em cluster, o nome dos MBeans necessários para efectuar uma monitorização completa, entre outras. Na Figura 6.23 está representado parte desse formulário.

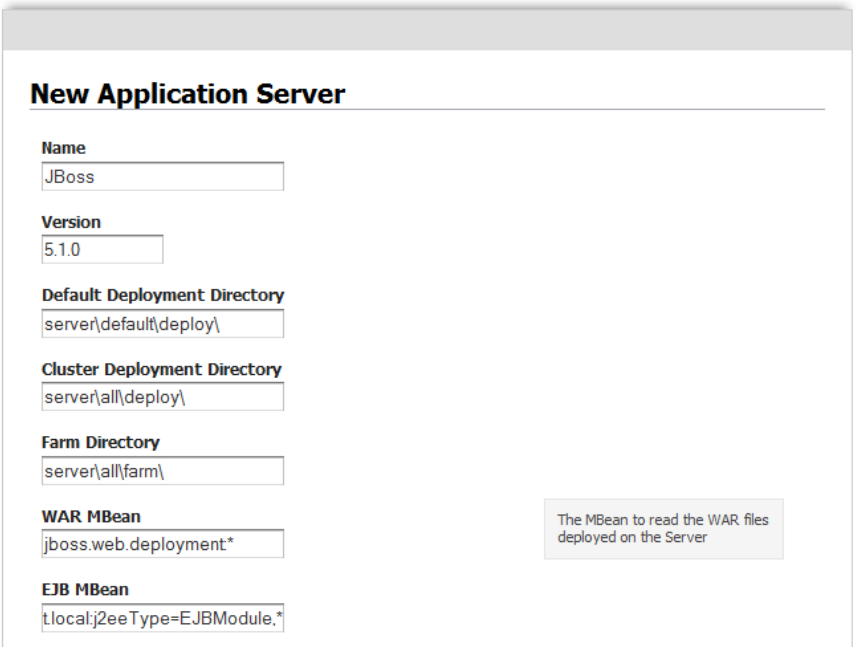


Figura 6.23: Adicionar um novo Servidor Aplicacional

De relevar que o administrador, ao adicionar um novo servidor aplicacional, deve ter em atenção se este recebeu a certificação Java EE atribuída pela Sun. Caso contrário, não são dadas garantias que, quer o deployment de aplicações, quer a monitorização do mesmo, funcionem correctamente.

Gestão dos SGBD

A gestão dos `SGBD` encontra-se, de alguma forma, limitada. Ou seja, caso o novo `SGBD` a ser introduzido não seja suportado pelo Sequoia, middleware utilizado na camada de dados, a utilização do novo `SGBD` numa infra-estrutura vai inviabilizar uma maior escalabilidade e disponibilidade da camada de dados desta, uma vez que só será possível ser constituída por um nó. No entanto, o administrador tem a liberdade de adicionar qualquer SGBD que deseje, devendo assinalar se o mesmo é compatível com o Sequoia.

As funcionalidades de remover o registo e os respectivos ficheiros, bem como a consulta de informação, também estão disponíveis.

Capítulo 7

Testes e Avaliação dos Resultados

Com esta dissertação pretendemos estudar a problemática, por um lado, da construção e gestão de infra-estruturas, e por outro da gestão e deployment de aplicações Web JEE. De modo a auxiliar os utilizadores na execução das referidas tarefas foi desenvolvida a ferramenta apresentada no capítulo anterior.

Ao longo deste capítulo efectuamos alguns testes e contabilizamos o tempo dispendido pelo utilizador na execução de determinadas tarefas, executadas com e sem a ajuda da ferramenta. Desta forma, poderemos avaliar a sua utilidade. Portanto, sem o recurso à ferramenta, o utilizador deve executar o comando **time** antes de cada tarefa, comando este que vai permitir saber o tempo de execução da mesma. Pelo contrário, com a utilização da ferramenta, é necessário efectuar pequenas alterações à mesma, de modo a conhecer o tempo de execução das tarefas. Para isso, é guardada, antes e no final da execução, a data e hora, em milisegundos, e, ao subtrair o tempo final ao inicial, obtemos o valor desejado.

Não tendo como principal objectivo avaliar os resultados de acordo com o tempo de execução de cada tarefa, é espectável que este seja menor com a utilização da ferramenta. No entanto, não é garantido que tal aconteça em todas as situações. De salientar, ainda, que os exemplos utilizados não são representativos do universo de aplicações que se podem encontrar, correspondendo a meros casos de estudo, mas que mimetizam um grande conjunto de aplicações.

7.1 Perfil das máquinas utilizadas

Para a realização das experiências foram utilizadas seis máquinas. As características de todas elas são:

- Processador - Intel Pentium Dual Core E5400;
- Memória - 1GB 800MHz DDR2;
- Disco - 50GB a 7200 rotações;
- Sistema Operativo - Ubuntu 9.04 Server Edition

De relevar, que todas as máquinas utilizadas nas experiências efectuadas encontravam-se, na altura, na mesma rede local. Desta forma, o tempo de transferência de ficheiros é mais baixo, do que seria caso as máquinas se encontrassem em locais e redes distintas. A rede local era constituída por um router e um switch, ambos com velocidades de transferência de 100Mbps.

7.2 Experiência 1 - Construção de uma infra-estrutura

Esta experiência tem por objectivo a construção e configuração correcta uma infra-estrutura JEE. Esta infra-estrutura engloba um servidor Web, dois servidores aplicacionais em cluster, o middleware Sequoia e um Sistema de Gestão de Base de Dados, podendo ser considerada uma infra-estrutura mínima de um portal Web.

Para esta experiência utilizamos os seguintes pacotes de software:

- Apache 1.3.41 + JK-1.2.28
- JBoss 5.1.0
- Sequoia 2.10.10
- Mysql 5.1

Os pacotes, acima indicados, já se devem encontrar na máquina do utilizador.

7.2.1 Sem a ferramenta de apoio

Para construir uma infra-estrutura é necessário efectuar algumas tarefas, que, de acordo com a experiência do utilizador, podem ser mais ou menos demoradas e morosas. Resumidamente, estas tarefas consistem em:

1. Transferir os pacotes de software para os respectivos servidores

2. Instalar e configurar o Servidor Web
3. Instalar e configurar os Servidores Aplicacionais
4. Instalar e configurar o `SGBD`
5. Instalar e configurar o Sequoia

Os tempos de instalação e configuração de cada camada da infra-estrutura podem ser consultados na Tabela 7.1.

	Tempo de Instalação e Configuração (h:min:seg)
Servidor Web	00:12:59
Servidores Aplicacionais	00:12:05
Base de Dados	00:04:15
Sequoia	00:13:51
Tempo Total	0:43:10

Tabela 7.1: Tempos da Exp. 1 - sem a ferramenta

Podemos constatar, através de uma análise à tabela acima apresentada, que a instalação e configuração do servidor Web e do middleware Sequoia foram as tarefas em que o utilizador despendeu mais tempo. De facto, entre as tarefas a executar, estas são as que necessitam uma maior configuração.

O gráfico 7.1 ilustra a percentagem de tempo que o utilizador gastou em cada uma das tarefas.

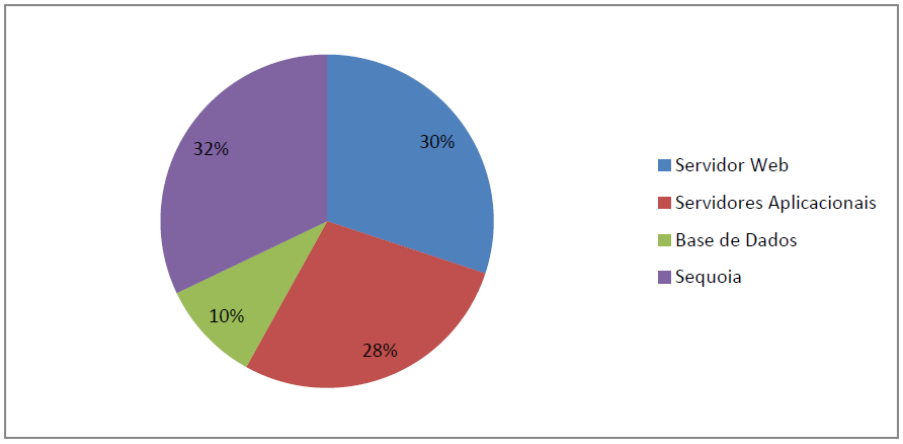


Figura 7.1: Percentagem do tempo gasto em cada tarefa

De salientar que, apesar do tempo de instalação e configuração da camada de Servidores Aplicacionais ocupar 28% do tempo total, este valor obtido resultou da instalação e configuração de dois servidores aplicacionais.

7.2.2 Com a ferramenta de apoio

De modo a percebermos até que ponto a ferramenta apresentada vem apoiar o utilizador na construção de uma infra-estrutura, agilizando todo este processo, a experiência, descrita acima, foi executada, desta feita, com o recurso à ferramenta. Na Figura 7.2 podemos ver parte dos dados introduzidos pelo utilizador.

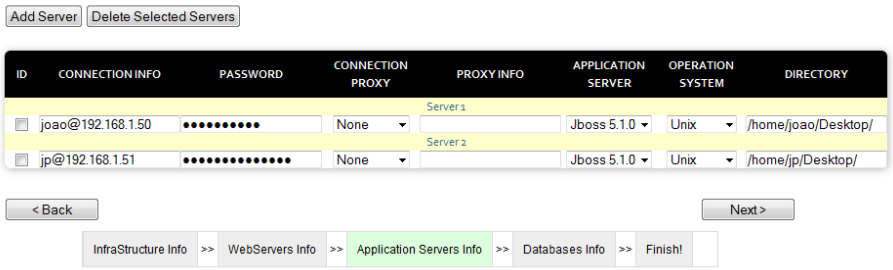


Figura 7.2: Introdução dos dados por parte do utilizador

O tempos de execução podem ser consultados na Tabela 7.2.

	Tempo de Instalação e Configuração (h:min:seg)
Introdução dos dados e seu processamento	0:00:55
Servidor Web	0:02:23
Servidores Aplicacionais	0:04:37
Base de Dados	0:01:05
Sequoia	0:01:20
Tempo Total	0:10:20

Tabela 7.2: Tempos da Exp. 1 - com a ferramenta

Perante os resultados obtidos com a experiência, podemos concluir que, quer utilizando ou não a ferramenta, a instalação e configuração do servidor Web ocupa uma parte considerável do tempo total, facilmente observável na Figura 7.3. Esta situação deve-se principalmente, ao tempo que demora a compilação do mesmo, não sendo possível a optimização deste processo, em qualquer uma das situações.

7.2.3 Avaliação dos Resultados

Nesta experiência pretendíamos perceber se a ferramenta podia, ou não, ajudar o utilizador a agilizar o processo de construção de uma infra-estrutura. Se, sem o recurso à ferramenta o utilizador demorou cerca de 43 minutos a completar a instalação e configuração da infra-estrutura, com o apoio desta, este tempo foi reduzido cerca de 4.3 vezes, para 10 minutos, sendo, notória a diferença de tempos nas duas situações.

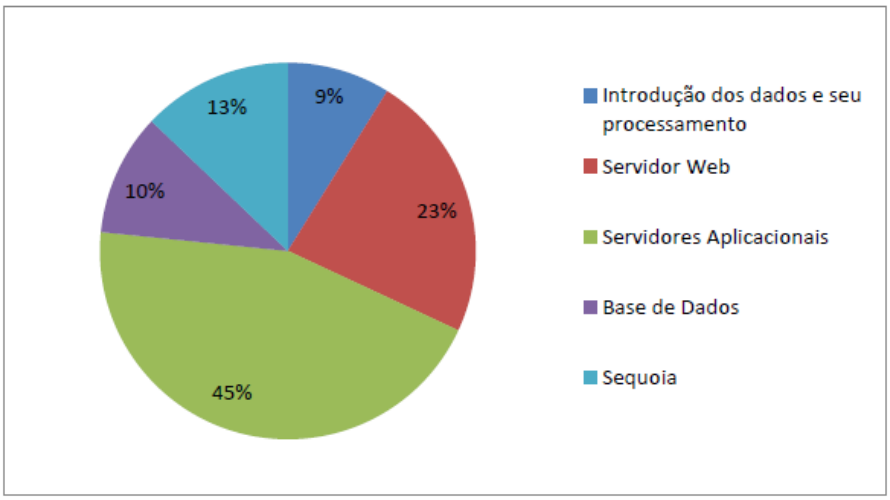


Figura 7.3: Percentagem do tempo gasto em cada tarefa

De referir ainda que, enquanto na primeira situação o utilizador teve os 43 minutos sempre ocupado com as configurações, dos 11 minutos que demorou na segunda apenas foi necessário a interacção do utilizador em cerca de 3 minutos, tempo este utilizado para a introdução dos dados da infra-estrutura.

7.3 Experiência 2 - Deployment de uma Aplicação

Nesta segunda experiência, o objectivo é efectuar o deployment de uma aplicação numa determinada infra-estrutura remota, partindo do principio que nenhuma aplicação estava a ser executada na infra-estrutura. Esta infra-estrutura é constituída por um servidor Web (Apache 1.3.41), um servidor aplicacional (JBoss 5.1.0), um servidor com o middleware Sequoia e um servidor de Base de Dados (Mysql 5.1). A aplicação a ser utilizada no deployment é um sistema de chat.

7.3.1 Sem a ferramenta de apoio

De modo a efectuar o deployment da aplicação de forma correcta, o utilizador deve executar os seguintes passos:

1. Criar a base de dados. Esta tarefa implica uma ligação remota ao servidor onde está instalado o `SGBD` da infra-estrutura;

2. Criar os ficheiros de configuração do Sequoia, carregando-os de seguida. Vamos admitir que o utilizador tem acesso a ficheiros que servem como template, não sendo, desta forma, necessário criá-los de raiz;

3. Transferir os ficheiros da aplicação para o servidor aplicacional;
4. Configurar a ligação à Camada de Dados;
5. Editar os ficheiros de configuração dos URL presentes no servidor Web, de modo a que este efectue o correcto redireccionamento;

A Tabela 7.1 ilustra os tempos de execução das tarefas descritas anteriormente.

	Tempo de Instalação e Configuração (h:min:seg)
Criação da Base de Dados	0:00:56
Configuração do Sequoia	0:07:32
Transferência de ficheiros da aplicação para o servidor aplicacional	0:00:30
Configuração da ligação à Camada de Dados	0:02:37
Edição de ficheiros dos Servidores Web	0:01:35
Tempo Total	0:13:10

Tabela 7.3: Tempos da Exp. 2 - sem a ferramenta

Analisando os resultados obtidos, é perceptível que o utilizador gasta uma considerável parte do seu tempo a efectuar as configurações necessárias, sendo que a transferência de ficheiros da aplicação para o servidor aplicacional foi a tarefa com menor tempo de execução.

O tempo total de execução desta experiência seria consideravelmente maior se não partíssemos do princípio que o utilizador possuía exemplos de ficheiros (templates) das configurações necessárias.

7.3.2 Com a ferramenta de apoio

A mesma experiência, mas desta vez com o apoio da ferramenta gerou os resultados presentes na Tabela 7.4.

	Tempo de Instalação e Configuração (h:min:seg)
Introdução dos dados sobre a aplicação e seu processamento	00:01:20
Criação da Base de Dados	0:00:17
Configuração do Sequoia	0:01:05
Transferência de ficheiros da aplicação para o servidor aplicacional	0:00:43
Configuração da ligação à Camada de Dados	0:00:04
Edição de ficheiros dos Servidores Web	0:00:12
Tempo Total	0:03:41

Tabela 7.4: Tempos da Exp. 2 - com a ferramenta

Pelos resultados obtidos podemos constatar que a tarefa que demorou mais tempo a executar foi a configuração do Sequoia, devido à sua necessidade em carregar a informação referente à nova base de dados. De salientar, ainda, a forma rápida como foram completadas as outras tarefas de configuração.

Outro dado a retirar destes resultados, foi o tempo necessário para a introdução dos dados na ferramenta (1m20s).

7.3.3 Avaliação dos Resultados

Comparando os resultados obtidos nas duas situações, numa lógica estritamente de comparação temporal, é perceptível a diferença entre os tempos totais em cada uma. Efectivamente, com o apoio da ferramenta obtivemos um tempo total de deployment e configuração cerca de 3.5 vezes menor do que na situação em que não é utilizada a ferramenta de apoio. Esta diferença é particularmente visível na configuração do middleware Sequoia e na configuração da ligação de dados.

No entanto, se não contabilizarmos o tempo demorado pelo utilizador a introduzir os dados, esta diferença é ainda mais significativa, sendo o tempo total do deployment e configuração 5 vezes menor com o apoio da ferramenta.

7.4 Avaliação Global

Através dos resultados obtidos pelas experiências realizadas, podemos comprovar a importância de uma ferramenta que permite automatizar todo o processo de construção e configuração de infra-estruturas, bem como o deployment de aplicações. Na Figura 7.4 estão representados os resultados globais das experiências realizadas.

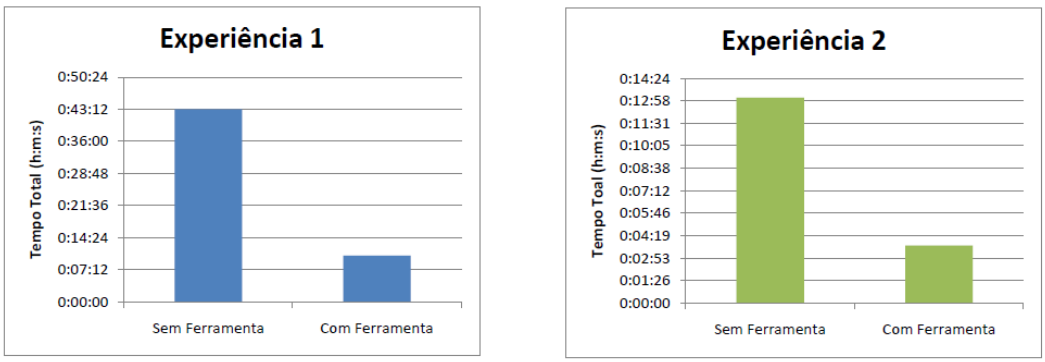


Figura 7.4: Resultados globais das experiências realizadas

São visíveis as diferenças dos tempos de execução das tarefas, designadamente a quando do recurso, ou não, à ferramenta. De facto, enquanto que sem o recurso à ferramenta, o utilizador está o tempo todo ocupado, com o apoio desta, este apenas necessita de introduzir os dados, sendo o restante processo automático.

De salientar, ainda, que as tarefas realizadas, sobretudo na Experiência 1, necessitavam de um aprofundado conhecimento técnico sobre infra-estruturas, que nem todos os utilizadores possuem, dificultando, deste modo, a conclusão das mesmas. Sem os mecanismos de apoio, provavelmente seria necessário haver um utilizador especializado que apoiasse as equipas de desenvolvimento na execução das tarefas apresentadas em ambas as experiências. Com a ferramenta apresentada tenta-se suprir a possível falta de conhecimento sobre infra-estruturas, necessário para uma correcta instalação e configuração das aplicações, tornando, desta forma, transparente todo este processo. Com a automatização do processo, os eventuais erros de deployment devido a problemas nas configurações, causados por erro humano, deixam de existir.

Em suma, pelo exposto e através dos resultados das experiências realizadas, tentamos comprovar e validar a necessidade de uma ferramenta que auxiliasse o utilizador, não só no processo de deployment, mas também, na construção de uma infra-estrutura. Os resultados obtidos são animadores e levam-nos a pensar que a ferramenta auxilia, de facto, os utilizadores. De salientar, mais uma vez, que os exemplos são apenas demonstrativos em relação ao universo de aplicações que se podem encontrar. No entanto, conseguem abranger um subconjunto de aplicações relativamente elevado.

Capítulo 8

Conclusões

Com a passagem das aplicações para a Internet, a garantia de qualidade não se faz apenas a nível dos programas, a infra-estrutura assume, também, um papel de extrema importância. Conscientes deste facto, o objectivo primordial que orientou o desenvolvimento da presente dissertação foi a problemática em torno, por um lado, da construção e gestão de infra-estruturas, e por outro da gestão e deployment de aplicações Web JEE.

Estando as aplicações expostas nessa montra global que é a Internet, é imprescindível que estas assentem numa infra-estrutura capaz de responder às exigências a que vão estar sujeitas. A importância da infra-estrutura e da própria aplicação foi notória ao longo deste trabalho, sendo que o sucesso de uma qualquer plataforma Web depende da boa ligação e do bom funcionamento de ambas.

No capítulo 4 foram apresentados dois casos concretos onde a plataforma Web ficou indisponível, devido à falta de capacidade da sua infra-estrutura em suportar o número de pedidos a que este foi sujeito. O planeamento da infra-estrutura de um sistema assume-se como um processo cada vez mais imprescindível. De facto, apenas com um estudo profundo é possível estimar a carga que deve ser suportada pela infra-estrutura de um sistema. Devido à exposição dos sistemas Web, este estudo, torna-se, portanto, bastante complexo, sendo, em muitos casos, impossível prever a carga de pedidos a que pode ser sujeito. Deste modo, torna-se necessário construir uma infra-estrutura capaz de se adaptar a diferentes fluxos de carga, devendo ser o mais escalável possível.

Conscientes da complexidade inerente à construção e gestão de uma infra-estrutura, um dos objectivos da ferramenta desenvolvida é auxiliar o utilizador nestas tarefas críticas. Na secção 6.2 descrevemos de que forma se desenrola todo o processo de construção da infra-estrutura, desde a escolha dos servidores que irão constituir a mesma, bem como, os pacotes de software a ser instalados e configurados em cada um.

Através dos resultados obtidos na experiência apresentada em 7.2, percebemos a utili-

dade da ferramenta na agilização de todo este processo. No que concerne à construção da infra-estrutura, apenas um utilizador com um profundo conhecimento técnico em infra-estruturas seria capaz de configurar e instalar correctamente esta em tempo útil. Com o auxílio da ferramenta, esta tarefa torna-se praticamente transparente ao utilizador, sendo que este apenas necessita de indicar a constituição da infra-estrutura e os pacotes de software a serem utilizados. De facto, esta transparência vai permitir de alguma forma, não só reduzir os erros humanos que sucedem nas configurações dos sistemas, uma vez que, em muitos casos, estas configurações possuem demasiada meta-informação, mas também, permitir que um maior leque de utilizadores possa efectuar esta tarefa tão importante. Por acréscimo, com o recurso à ferramenta, o tempo que um utilizador demora a efectuar esta tarefa foi reduzido em cerca de quatro vezes.

Para além de auxiliar os utilizadores na construção de uma infra-estrutura, são disponibilizados, ainda, mecanismos de monitorização da mesma. Através dos quais é possível obter várias informações sobre os servidores que fazem parte da infra-estrutura, nomeadamente, a memória que está ser consumida, as aplicações que estão a ser executadas nos servidores aplicacionais, entre outras. Com os dados obtidos, o utilizador é capaz de perceber se algum servidor da infra-estrutura apresenta anomalias ou se é necessário aumentar o número de nodos de alguma camada específica a fim de garantir tempos de resposta satisfatórios.

Reconhecendo a constante evolução e actualização dos vários tipos de servidores que constituem uma infra-estrutura e não querendo tornar, de forma alguma, a ferramenta limitada, é dada a possibilidade ao utilizador de adicionar novas versões dos pacotes de software, que poderão ser instalados nos servidores. No entanto, esta tarefa implica a introdução de uma quantidade substancial de informação, de modo a permitir ao utilizador tirar partido das potencialidades da ferramenta apresentada. Isto deve-se aos diferentes mecanismos, presentes nesses pacotes, para disponibilizar determinados dados.

Outro factor que pode influenciar o comportamento de uma plataforma Web é a maneira como o deployment e a configuração de uma aplicação na infra-estrutura são efectuados. De facto, o aumento da complexidade e dimensão das aplicações tornou a tarefa de deployment inevitavelmente mais delicada.

Tendo em conta que uma aplicação Web pode estar repartida em vários ficheiros, a forma como estes são distribuídos pela infra-estrutura pode influenciar o correcto funcionamento de todo a plataforma Web, como podemos atestar em alguns estudos apresentados na secção 3.1. Esta constatação vem aumentar, ainda mais, a necessidade de um planeamento cuidado do deployment. Com o apoio da ferramenta desenvolvida, o utilizador pode gerir e efectuar o deployment de uma aplicação de uma forma simplificada e transparente.

Este poderá escolher como os ficheiros da aplicação poderão ser distribuídos, entre os vários nodos existentes, tendo, ainda, a capacidade de adaptar essa mesma distribuição, caso pretenda

Com o comportamento dos servidores a ser monitorizado pela ferramenta, o utilizador poderá efectuar um estudo mais detalhado da infra-estrutura, estudo este que permitirá efectuar uma melhor distribuição da aplicação pelos vários nodos, sem que esta afecte outras aplicações que possam já estar em execução.

Através da Experiência 7.3 podemos comprovar que a ferramenta, não só facilita o processo de deployment, mas também o optimiza. De facto, o utilizador apenas necessita de indicar os ficheiros que compõem a aplicação e a forma como estes vão ser distribuídos pela infra-estrutura, sendo as restantes tarefas executadas automaticamente pela ferramenta. No caso da aplicação utilizar uma base de dados, o utilizador deve indicar o nome do ficheiro *dump* que contem as informações sobre a mesma, ou então escolher um modelo de dados criado através dos mecanismos apresentados em 6.5. Além disso, os resultados obtidos demonstram que o deployment com o apoio da ferramenta demora 1/3 do tempo gasto sem esse recurso. No entanto, este resultado é mais expressivo no deployment inicial da aplicação, uma vez que, são necessárias diversas configurações que, na maior parte dos casos, se mantêm inalteradas ao longo do ciclo de vida da aplicação. Portanto, não é de estranhar que a optimização conseguida por parte da ferramenta não seja tão notória em todas as fases do ciclo de vida de uma aplicação.

O considerável aumento da dimensão das aplicações não veio, apenas, agravar a complexidade do processo de deployment da mesma, tendo, também, conduzido a uma mudança nos paradigmas de desenvolvimento das mesmas. Esta mudança levou os sistemas de controlo de versões a assumirem um papel de maior relevo. Sendo o objectivo primeiro da ferramenta auxiliar o mais possível o utilizador a desempenhar as suas tarefas, consideramos importante adicionar à ferramenta mecanismos que permitissem uma comunicação com os sistemas de controlo de versões. No caso concreto deste protótipo, apenas o `SVN` foi contemplado.

Em síntese, com este trabalho pretendemos demonstrar que o sucesso de uma aplicação Web depende fortemente de como é desenvolvida, mas também como e onde é efectuado o seu deployment. Desta forma, torna-se essencial um estudo cuidado da aplicação e da sua finalidade, de modo a projectar uma infra-estrutura capaz de se constituir como uma base sólida do sistema.

Após a construção da infra-estrutura é prioritário considerar a forma como a aplicação vai ser distribuída pela infra-estrutura. Estes são factores de extrema importância que podem ditar o sucesso de uma plataforma Web. Reconhecendo a complexidade destas

tarefas, bem como a exigência de um elevado conhecimento técnico, que nem sempre se encontra disponível, tentámos demonstrar as mais valias de uma ferramenta que auxiliasse o utilizador, não só na execução destas tarefas mas, também, na gestão de toda a infra-estrutura e respectivas aplicações.

8.1 Trabalho Futuro

Existem vertentes que, apesar de enriquecedoras e pertinentes, não foram abordadas uma vez que tornariam o âmbito de estudo demasiado abrangente. Deste modo, nesta secção abordamos determinados aspectos que constituem linhas de investigação que se podem traçar na sequência desta dissertação.

8.1.1 Segurança

A passagem das aplicações e dos dados para Internet não acarretou, apenas, vantagens, tendo originado, também, novas preocupações, sendo a segurança uma delas. Torna-se prioritário equipar todo o sistema (infra-estrutura e aplicações) de mecanismos de segurança que impeçam utilizadores e ferramentas não autorizadas a interagir com esse sistema. Por conseguinte, um dos aspectos fundamentais a introduzir na ferramenta é a temática da segurança associada às infra-estruturas.

Como referido no capítulo 4 a segurança nas infra-estruturas é de extrema importância, sendo que, na maior parte das vezes, configurar os mecanismos que a garantem é uma tarefa bastante complexa. Portanto, seria uma mais valia equipar a ferramenta de mecanismos que auxiliassem o utilizador na configuração da segurança de uma infra-estrutura.

8.1.2 Camada Física de Dados

Dada a necessidade cada vez mais premente em criar uma camada de dados escalável e que esteja sempre disponível, e tendo em consideração que o protótipo desenvolvido apenas consegue configurar o middleware Sequoia com RAIDb-1, disponibilizando, no entanto, recursos ao utilizador para elaborar outra configuração de uma forma não assistida, seria interessante adicionar mecanismos que contemplassem outros métodos de configuração do Sequoia, flexibilizando, assim, a escolha do utilizador.

Tendo presente que a replicação de Base de Dados é uma área extremamente vasta, e sabendo que existe uma variedade de soluções, um outro caminho a seguir seria a introdução de mecanismos que permitissem outras formas de assegurar a disponibilidade e escalabilidade da camada de dados para além do middleware Sequoia.

8.1.3 Virtualização

Nos últimos anos o poder de processamento dos computadores aumentou consideravelmente, levando a que conceitos como a virtualização ganhassem maior importância. A virtualização está relacionada com a capacidade de partilhar os recursos físicos de uma máquina - processador, memória RAM, disco rígido - por múltiplos ambientes de execução designados por máquinas virtuais.

Seria interessante dotar a ferramenta, apresentada neste trabalho, de mecanismo que tirassem partido da virtualização para a construção de uma infra-estrutura, podendo aproveitar da melhor forma os recursos disponíveis em cada máquina.

Bibliografia

[AA04] B. Randell C. Landwehr A. Avizienis, J. Laprie. Basic concepts and taxonomy of dependable and secure computing. *IEEE Trans. Dependable Secur. Comput.*, 1(1):11–33, 2004.

[AHP+09] Arden Agopyan, Hermann Huebler, Tze Puah, Thomas Schulze, David Soler Vilageliu, and Martin Keen. *WebSphere Application Server V7.0: Concepts, Planning, and DesignHermann Huebler.* Redbooks. Vervante, February 2009.

[Alla] OSGi™ Alliance. Open services gateway initiative. `http://www.osgi.org/Main/HomePage`.

[Allb] The OSGi Alliance. Osgi service platform core specification, release 4, version 4.2. `http://www.osgi.org/Download/Release4V42`.

[Ash04] Derek C. Ashmore. *The J2EE Architect's Handbook.* DVT Press, May 2004.

[ATK05] Anatoly Akkerman, Alexander Totok, and Vijay Karamcheti. Infrastructure for automatic dynamic deployment of j2ee applications in distributed environments. pages 17–32. 2005.

[BC08] Kirk Bauer and Nathan Campi. *Automating Linux and Unix System Administration, Second Edition.* Apress, Berkely, CA, USA, 2008.

[BD06] Meriem Belguidoum and Fabien Dagnat. Analysis of deployment dependencies in software components. In *SAC '06: Proceedings of the 2006 ACM symposium on Applied computing*, pages 735–736, New York, NY, USA, 2006. ACM.

[BD07] Meriem Belguidoum and Fabien Dagnat. Dependency management in software component deployment. *Electron. Notes Theor. Comput. Sci.*, 182:17–32, 2007.

[BG03] Paul Brebner and Jeffrey Gosper. How scalable is j2ee technology? *SIGSOFT Softw. Eng. Notes*, 28(3):4–4, 2003.

[BH04] Gunnar Brataas and Peter Hughes. Exploring architectural scalability. In *WOSP '04: Proceedings of the 4th international workshop on Software and performance*, pages 125–129, New York, NY, USA, 2004. ACM.

[BK06] Christian Bauer and Gavin King. *Java Persistence with Hibernate*. Manning Publications Co., Greenwich, CT, USA, 2006.

[BP75] R. Barlow and F. Proschan. *Statistical Theory of Reliability and Life Testing Probability Models*. Holt, Rinehart and Winston Inc., 1975.

[BP03] Antonia Bertolino and Andrea Polini. A framework for component deployment testing. In *ICSE '03: Proceedings of the 25th International Conference on Software Engineering*, pages 221–231, Washington, DC, USA, 2003. IEEE Computer Society.

[BSB05] Daniel J. Barrett, Richard E. Silverman, and Robert G. Byrne. *SSH, the Secure Shell, 2nd Edition*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, May 2005.

[CE00] T. Coupaye and J. Estublier. Foundations of enterprise software deployment. In *CSMR '00: Proceedings of the Conference on Software Maintenance and Reengineering*, page 65, Washington, DC, USA, 2000. IEEE Computer Society.

[Che04] Whei-Jen Chen. *DB2 Integrated Cluster Environment Deployment Guide*. International Technical Support Organization, 2004.

[Chi01] Willy Chiu. Design for scalability - an update. Technical report, IBM High Volume Web Sites, Software Group, April 2001.

[CM03] Emmanuel Cecchet and Julie Marguerite. C-jdbc: Scalability and high availability of the database tier in j2ee environments. In *the 4th ACM/IFIP/USENIX International Middleware Conference (Middleware), Poster session*, Rio de Janeiro, Brazil, June 2003.

[CMZ02] Emmanuel Cecchet, Julie Marguerite, and Willy Zwaenepoel. Performance and scalability of ejb applications. *SIGPLAN Not.*, 37(11):246–261, 2002.

[CMZ05] Emmanuel Cecchet, Julie Marguerite, and Willy Zwaenepoel. RAIDb: Redundant Array of Inexpensive Databases. Technical report, 2005.

[Cora] Oracle Corporation. Oracle weblogic server. `http://www.oracle.com/appserver/appserver_family.html`.

[Corb] Oracle Corporation. Weblogic server standard edition. `http://www.oracle.com/appserver/docs/weblogic-server-se-datasheet.pdf`.

[Dea07] A Dearle. Software deployment, past, present and future. In *International Conference on Software Engineering*, pages 269–284. IEEE Computer Society, 2007.

[DKMy04] Alan Dearle, Graham N. C. Kirby, Andrew McCarthy, and Juan Carlos Diaz y Carballo. A flexible and secure deployment framework for distributed applications. In *Component Deployment*, pages 219–233, 2004.

[DKS] Dr Michael Doherty, Kerstin Kleese, and Shoaib Sufi. Database cluster for e-science. In *Proceedings of the UK e-Science All Hands Meeting 2004*, pages 268–271.

[DMM04] Ada Diaconescu, Adrian Mos, and John Murphy. Automatic performance management in component based software systems. In *ICAC '04: Proceedings of the First International Conference on Autonomic Computing*, pages 214–221, Washington, DC, USA, 2004. IEEE Computer Society.

[EKKP05] Tamar Eilam, Michael H. Kalantar, Alexander V. Konstantinou, and Giovanni Pacifici. Reducing the complexity of application deployment in large data centers. In *Integrated Network Management*, pages 221–234, 2005.

[Exe04] François Exertier. J2ee deployment: The jonas case study. *CoRR*, cs.NI/0411054, 2004.

[FC05] Jim Farley and William Crawford. *Java Enterprise in a Nutshell*. O' Reilly & Associates, Inc., 2005.

[Fer] Paul Ferraro. Ha-jdbc: High-availability jdbc. `http://ha-jdbc.sourceforge.net/`.

[Fir] Firstworks. Sql relay. `http://sqlrelay.sourceforge.net/`.

[Fou] The Apache Software Foundation. Apache geronimo. `http://geronimo.apache.org/`.

[GAG+06] Leonel Aguilar Gayard, Paulo Astério, Castro Guerra, Ana Elisa, Campos Lobo, Cecília Mary, and Fischer Rubira. Automated deployment of component architectures with versioned components, 2006.

[GD02] John Grundy and Guoliang Ding. Automatic validation of deployed j2ee components using aspects. In *ASE '02: Proceedings of the 17th IEEE international conference on Automated software engineering*, page 47, Washington, DC, USA, 2002. IEEE Computer Society.

[Gil06] Nasib S. Gill. Importance of software component characterization for better software reusability. *SIGSOFT Softw. Eng. Notes*, 31(1):1–3, 2006.

[GL03] Joseph D. Gradecki and Nicholas Lesiecki. *Mastering AspectJ: Aspect-Oriented Programming in Java*. John Wiley & Sons, Inc., New York, NY, USA, 2003.

[GLP04] Jiang Guo, Yuehong Liao, and Behzad Parviz. A survey of j2ee application performance management systems. In *ICWS '04: Proceedings of the IEEE International Conference on Web Services*, page 724, Washington, DC, USA, 2004. IEEE Computer Society.

[GLP06] Jiang Guo, Yuehong Liao, and Behzad Parviz. A performance validation tool for j2ee applications. In *ECBS '06: Proceedings of the 13th Annual IEEE International Symposium and Workshop on Engineering of Computer Based Systems*, pages 387–396, Washington, DC, USA, 2006. IEEE Computer Society.

[Gro] Object Management Group. Deployment and configuration of component-based distributed applications specification. `http://www.omg.org/docs/formal/06-04-02.pdf`.

[GTWJ03] Jerry Zayu Gao, Jacob Tsao, Ye Wu, and Taso H.-S. Jacob. *Testing and Quality Assurance for Component-Based Software*. Artech House, Inc., Norwood, MA, USA, 2003.

[Hat] Red Hat. Jopr. `http://www.jboss.org/jopr`.

[HBB+08] Geoffrey Hambrick, Kyle Brown, Roland Barcia, Robert Peterson, and Kulvir S. Bhogal. *Persistence in the Enterprise: A Guide to Persistence Technologies (The developerWorks Series)*. IBM Press, 2008.

[Hey06] Abbas Heydarnoori. Caspian: A qos-aware deployment approach for channel-based component-based applications. Technical report, University of Waterloo, October 2006.

[HHW99] Richard S. Hall, Dennis Heimbigner, and Alexander L. Wolf. A cooperative approach to support software deployment using the software dock. In *ICSE '99: Proceedings of the 21st international conference on Software engineering*, pages 174–183, New York, NY, USA, 1999. ACM.

[Hig01] High-performance web databases, design, development, and deployment, 2001.

[HMA06] Abbas Heydarnoori, Farhad Mavaddat, and Farhad Arbab. Deploying loosely coupled, component-based applications into distributed environments. In *ECBS '06: Proceedings of the 13th Annual IEEE International Symposium and Workshop on Engineering of Computer Based Systems*, pages 93–102, Washington, DC, USA, 2006. IEEE Computer Society.

[HNG05] Steve Holzner, Daniel Nehren, and Ben Galbraith. *Ant: The Definitive Guide*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 2005.

[Hui99] M. Huizing. Component based development. *Component Technology*, 6(2):5–9, January 1999.

[HW04] Jamie Hillman and Ian Warren. An open framework for dynamic reconfiguration. In *ICSE '04: Proceedings of the 26th International Conference on Software Engineering*, pages 594–603, Washington, DC, USA, 2004. IEEE Computer Society.

[HWM+04] Gang Huang, Meng Wang, Liya Ma, Ling Lan, Tiancheng Liu, and Hong Mei. Towards architecture model based deployment for dynamic grid services. In *CEC-EAST '04: Proceedings of the E-Commerce Technology for Dynamic E-Business, IEEE International Conference*, pages 14–21, Washington, DC, USA, 2004. IEEE Computer Society.

[Hyp] Inc Hyperic. Hyperic hq. `http://community.zenoss.org/index.jspa`.

[IBM] IBM. Websphere application server. `http://www-01.ibm.com/software/webservers/appserv/was/`.

[Inca] Continuent Inc. Sequoia. `http://community.continuent.com/community/sequoia`.

[Incb] Oracle Inc. Oracle real application clusters. `http://www.oracle.com/technology/products/database/clustering/index.html`.

[JL02] The JBoss Group Juha Lindfors, Marc Fleury. *JMX: Managing J2EE with Java Management Extensions*. Sams Publishing, 2002.

[JMZ04] Emmanuel Cecchet Julie, Julie Marguerite, and Willy Zwaenepoel. C-jdbc: Flexible database clustering middleware. In *In Proceedings of the USENIX 2004 Annual Technical Conference*, pages 9–18, 2004.

[JN04] Ivar Jacobson and Pan-Wei Ng. *Aspect-Oriented Software Development with Use Cases (Addison-Wesley Object Technology Series)*. Addison-Wesley Professional, 2004.

[Joh02] Rod Johnson. *Expert One-on-One J2EE Design & Development*. Wrox Press Ltd., Birmingham, UK, UK, 2002.

[Jos07] David Josephsen. *Building a Monitoring Infrastructure with Nagios*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2007.

[JPR+07] Alfrânio Correia Jr., José Pereira, Luís Rodrigues, Nuno Carvalho, Ricardo Vilaça, Rui Oliveira, and Susana Guedes. Gorda: An open architecture for database replication. *Network Computing and Applications, IEEE International Symposium on*, 0:287–290, 2007.

[KA00] Bettina Kemme and Gustavo Alonso. Don't be lazy, be consistent: Postgres-r, a new way to implement database replication. In *VLDB '00: Proceedings of the 26th International Conference on Very Large Data Bases*, pages 134–143, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[KB05] Abdelmadjid Ketfi and Noureddine Belkhatir. Model-driven framework for dynamic deployment and reconfiguration of component-based software systems. In *MIS '05: Proceedings of the 2005 symposia on Metainformatics*, page 8, New York, NY, USA, 2005. ACM.

[KDM04] Graham N. C. Kirby, Alan Dearle, and Andrew J. McCarthy. A framework for constraint-based deployment and autonomic management of distributed applications. In *ICAC '04: Proceedings of the First International Conference on Autonomic Computing*, pages 300–301, Washington, DC, USA, 2004. IEEE Computer Society.

[Koz05] Charles M. Kozierok. *The TCP/IP Guide*. TCPIPGUIDE.com, September 2005.

[Kru98] P. Krutchen. Modeling component with the unified modeling language. *International Workshop on Component-Based Software Engineering*, 1998.

[Liu02] J. Liu. Performance and scalability measurement of cots ejb technology. In *SBAC-PAD '02: Proceedings of the 14th Symposium on Computer Architecture and High Performance Computing*, page 212, Washington, DC, USA, 2002. IEEE Computer Society.

[LM07] Jun Li and Keith Moore. A runtime and analysis framework support for unit component testing in distributed systems. In *HICSS '07: Proceedings of the 40th Annual Hawaii International Conference on System Sciences*, page 261c, Washington, DC, USA, 2007. IEEE Computer Society.

[LS06] Yu David Liu and Scott F. Smith. A formal framework for component deployment. In *OOPSLA '06: Proceedings of the 21st annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications*, pages 325–344, New York, NY, USA, 2006. ACM.

[Ltd] Tavant Technologies Pvt. Ltd. Infrared. `http://infrared.sourceforge.net/versions/latest/`.

[Mar02] Floyd Marinescu. *Ejb Design Patterns: Advanced Patterns, Processes, and Idioms with Poster*. John Wiley & Sons, Inc., New York, NY, USA, 2002. Foreword By-Roman, Ed.

[Mar06] F. Mário Martins. *Java 5 e Programação por Objectos*. FCA - Editora de Informática, 2006.

[MB04] Noëlle Merle and Noureddine Belkhatir. Open architecture for building large scale deployment systems. In *Software Engineering Research and Practice*, pages 930–936, 2004.

[MH04] Hong Mei and Gang Huang. Pkuas: An architecture-based reflective component operating platform. *Future Trends of Distributed Computing Systems, IEEE International Workshop*, 0:163–169, 2004.

[Mic] Sun Microsystems. Ecperf specification and kit. `http://java.sun.com/developer/releases/j2ee/ecperf/`.

[Mic08] Sun Microsystems. Java management extensions specification 1.4. `http://java.sun.com/javase/6/docs/technotes/guides/jmx/JMX_1_4_specification.pdf`, 2008.

[MLCS90] Dan C. Marinescu, James E. Lumpp, Jr., Thomas L. Casavant, and Howard Jay Siegel. Models for monitoring and debugging tools for parallel and distributed software. *J. Parallel Distrib. Comput.*, 9(2):171–184, 1990.

[MRMBM04] Marija Mikic-Rakic, Sam Malek, Nels Beckman, and Nenad Medvidovic. A tailorable environment for assessing the quality of deployment architectures in highly distributed settings. In *Component Deployment*, pages 1–17, 2004.

[MS03] Evan Marcus and Hal Stern. *Blueprints for high availability: designing resilient distributed systems, 2nd Edition*. John Wiley & Sons, Inc., New York, NY, USA, 2003.

[Mul02] Craig S. Mullins. *Database Administration: The Complete Guide to Practices and Procedures*. Addison Wesley, 2002.

[MZ05] C.H. Messom and M. Zhou. Application scalability for clustered server systems. In *Proceedings of the Twelfth Electronics New Zealand Conference*, pages 81–86, 2005.

[Nag] Nagios. Nagios. `http://www.nagios.org/`.

[Obe98] Patricia Oberndorf. Cots and open systems. In *SEI Monographs on the Use of Commercial Software in Goverment Systems*, January 1998.

[OW2a] OW2. Jonas 5 - white paper. `http://wiki.jonas.ow2.org/xwiki/bin/download/Main/Documentation/JOnAS5_WP.pdf`.

[OW2b] OW2. Jonas application server. `http://wiki.jonas.ow2.org/xwiki/bin/view/Main/WebHome`.

[PA06] Eleni Patouni and Nancy Alonistioti. A framework for the deployment of self-managing and self-configuring components in autonomic environments. In *WOWMOM '06: Proceedings of the 2006 International Symposium on on World of Wireless, Mobile and Multimedia Networks*, pages 480–484, Washington, DC, USA, 2006. IEEE Computer Society.

[PDC01] Allen Parrish, Brandon Dixon, and David Cordes. A conceptual foundation for component-based software deployment. *Journal of Systems and Software*, 57(3):193–200, 2001.

[Per02] J. Steven Perry. *Java Management Extensions*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 2002.

[RAC+02] Matthew J. Rutherford, Kenneth M. Anderson, Antonio Carzaniga, Dennis Heimbigner, and Alexander L. Wolf. Reconfiguration in the enterprise javabean component model. In *CD '02: Proceedings of the IFIP/ACM Working Conference on Component Deployment*, pages 67–81, London, UK, 2002. Springer-Verlag.

[Red] LLC Red Hat Middleware. Jboss application server. `http://www.jboss.org/`.

[RM01] Marija Rakic and Nenad Medvidovic. Increasing the confidence in off-the-shelf components: a software connector-based approach. *SIGSOFT Softw. Eng. Notes*, 26(3):11–18, 2001.

[RT04] Mikael Ronstrom and Lars Thalmann. Mysql cluster architecture overview. Technical report, MySQL Technical White Paper, April 2004.

[SDH06] Sylvain Sicard, Noel De Palma, and Daniel Hagimont. J2ee server scalability through ejb replication. In *SAC '06: Proceedings of the 2006 ACM symposium on Applied computing*, pages 778–785, New York, NY, USA, 2006. ACM.

[SG96] Mary Shaw and David Garlan. *Software architecture: perspectives on an emerging discipline*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1996.

[SG08] Ritu Sabharwal and Julio Guijarro. Avalanche: managing deployments for enterprise scale grids. In *Compute '08: Proceedings of the 1st Bangalore annual Compute conference*, pages 1–4, 2008.

[SIA] ZABBIX SIA. Zabbix. `http://www.zabbix.org/`.

[SJ02] Nils P. Sudmann and Dag Johansen. Software deployment using mobile agents. In *CD '02: Proceedings of the IFIP/ACM Working Conference on Component Deployment*, pages 97–107, London, UK, 2002. Springer-Verlag.

[SKG08] Arun Sharma, Rajesh Kumar, and P. S. Grover. Estimation of quality for software components: an empirical approach. *SIGSOFT Softw. Eng. Notes*, 33(6):1–10, 2008.

[Sma] SmartFrog. Smart framework for object groups. `http://www.smartfrog.org/`.

[SP09] Luís Soares and José Pereira. A simple approach to shared storage database servers. In *WDDM '09: Proceedings of the Third Workshop on Dependable Distributed Data Management*, pages 21–24, New York, NY, USA, 2009. ACM.

[Spe03] Richard Sperko. *Java Persistence for Relational Databases.* Apress, 2003.

[SSJ02] Inderjeet Singh, Beth Stearns, and Mark Johnson. *Designing enterprise applications with the J2EE platform.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[Ste98] Hans-Peter Steiert. Towards a component-based n-tier c/s-architecture. In *ISAW '98: Proceedings of the third international workshop on Software architecture*, pages 137–140, New York, NY, USA, 1998. ACM.

[Suna] Sun. J2ee deployment specification 1.1. `http://java.sun.com/j2ee/tools/deployment/reference/docs/index.html`.

[Sunb] Inc Sun Microsystems. Sun glassfish enterprise server. `http://www.sun.com/software/products/appsrvr/`.

[Szy02] Clemens Szyperski. *Component Software: Beyond Object-Oriented Programming.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[Uni] Rice University. Rubis: Rice university bidding system. `http://rubis.ow2.org/`.

[Voa98] Jeffrey M. Voas. The challenges of using cots software in component-based development. *Computer*, 31(6):44–45, 1998.

[Wey98] Elaine J. Weyuker. Testing component-based software: A cautionary tale. *IEEE Softw.*, 15(5):54–59, 1998.

[Wie05] Jan Wieck. Slony-i - a replication system for postgresql. Technical report, 2005.

[WWS04] Han Wang, Hao Wang, and Jinmei Shen. Architectural design and implementation of highly available and scalable medical system with ibm websphere middleware. In *CBMS '04: Proceedings of the 17th IEEE Symposium on Computer-Based Medical Systems*, page 174, Washington, DC, USA, 2004. IEEE Computer Society.

[Zen] Inc Zenoss. Zenoss - open source monitoring and systems management. `http://community.zenoss.org/index.jspa`.