

Trabalho Final de Curso, LIG 2010

FreeBSD VPS Manager

Lisboa, 15 de Outubro de 2010

	<p>Elaborado por: Francisco Alves Cabrita - 20060645 Luís Monteiro Castro - 20060658</p>
--	--

Índice

1 - Resumo / Abstract.....	3
2 – Agradecimentos.....	5
3 - Introdução e Objectivos	6
4 - Enquadramento Teórico	8
4.1 - O Problema.....	8
4.2 - Levantamento de Requisitos.....	11
4.3 - Descrição do Sistema	12
4.4 - Análise de Requisitos	16
5 – Tecnologias.....	17
5.1 - Conceitos Básicos de Virtualização	17
5.2 - Métodos de Virtualização	20
5.3 - Análise aos Sistemas de Virtualização	26
5.4 - Sistema de Virtualização escolhido	27
5.5 - Implicações do Sistema de Virtualização Utilizado	31
6 – Objectivos	32
6.1 - Solução Proposta	32
6.2 - Requisitos do Protótipo	33
6.3 - <i>Wireframes</i> Modelo	34
7 - Modelo Conceptual	36
7.1 - Use Case	36
8 – Implementação	37
8.1 – Metodologia de Desenvolvimento.....	37
8.1.1 - Fases do Projecto.....	38
8.2 - Arquitectura do Sistema	39
8.3 - Instalação e Configuração.....	40
9 – Conclusão e Trabalho Futuro	43
10 – Bibliografia	44
11 – Anexos.....	45
11.1 Listagem de Acrónimos e Definições	45

	Elaborado por: Francisco Alves Cabrita - 20060645 Luís Monteiro Castro - 20060658
--	---

1 – RESUMO / ABSTRACT

Este projecto protótipo desenvolve uma solução alternativa para gerir e distribuir local e remotamente FreeBSD Jails - Virtual Private Systems. O sistema é uma aplicação web para administrar ambientes virtuais e seus utilizadores, assim como monitorizá-los enquanto sistemas residentes em anfitriões FreeBSD. Esta componente é responsável por permitir administradores de sistemas configurar e agendar a construção de servidores virtuais nos seus nós físicos. O seu principal propósito é disponibilizar uma ferramenta que facilite e, em parte, automatize a criação de servidores virtuais. Com esta ferramenta obtemos ainda uma visão global, holística de toda a infra-estrutura o que permite controlar a parte como um todo, de forma simples. Fica em aberto no futuro, o desenvolvimento e integração de um módulo capaz de enviar e receber mensagens XMPP com instruções aos agentes instalados em cada nó na rede, com a finalidade de expandir este sistema, criando uma nuvem de sistemas virtuais. Ambos os componentes vão comunicar com sub-sistemas que abstraem a gestão das Jails recorrendo a ferramentas disponíveis no sistemas operativo. Outros sistemas dependentes são o motor de bases de dados e o servidor XMPP em si.

Palavras chave: FreeBSD, Jails, Virtualização, Cloud, Gestão e Distribuição de Sistemas.

This prototype project develops an alternative solution to manage and deploy locally and remote FreeBSD Jails, Virtual Private Systems.

The system is a backoffice web interface to administrate virtual environments it's users, as well as monitorize them while residents of FreeBSD nodes.

This component is responsible for allowing administrators to configure and schedule the building of virtual servers on their physical nodes.

It's main propose is to provide systems administrators a birds eye and tools to expand their infrastructure without losing global control.

In the future this component may be able to send and receive its instructions messaging using XMPP agents installed on every main node in

	Elaborado por: Francisco Alves Cabrita - 20060645 Luís Monteiro Castro - 20060658
--	---

the network, with the purpose of expand this system, building a cloud of virtual systems.

Both components will communicate with another layer of sub-systems; this systems will abstract the jail management using systems tools. Other systems dependent might be a Data Base Engine and/or a XMPP Server itself.

Keywords: FreeBSD, Jails, Virtualization, Cloud, Systems Management and Deployment.

	Elaborado por: Francisco Alves Cabrita - 20060645 Luís Monteiro Castro - 20060658
--	---

2 - AGRADECIMENTOS

Há imensas frases fantásticas e imortais com as quais gostaria de acabar o meu parágrafo de agradecimento mas seriam demais as frases e de menos os dias. Aos familiares e amigos, muito obrigado.

"Viva como se fosse morrer amanhã.

Aprenda como se fosse viver para sempre."

Mahatma Gandhi

Francisco Alves Cabrita

Agradeço naturalmente a todos aqueles que de alguma forma me ajudaram a alcançar um objectivo individual e pessoal de longa data. Sei que sou uma pessoa diferente e isso reflecte-se na minha vida pessoal e profissional, se mudei, foi também graças às experiências e conhecimentos que vivi e adquiri, naquela que foi e é a "minha" universidade. Escrevo-o com o orgulho de alguém que sabe o esforço e dedicação que foram necessários para conseguir concluir com proveito esta etapa académica.

Obrigado a todos.

Luís Monteiro Castro

	Elaborado por: Francisco Alves Cabrita - 20060645 Luís Monteiro Castro - 20060658
--	---

3 - INTRODUÇÃO e OBJECTIVOS

Até à bem poucos anos atrás as organizações sempre que se sentia necessidade de expandir a uma infra-estrutura, adquiria-se mais “ferro”, termo comum usado para expressar a aquisição de mais hardware para agregar à rede e partilhar as funções pelos demais servidores.

Para um servidor de HTTP em sub-carga, a engenharia adquiria mais hardware e posteriormente replicava toda a camada aplicacional para este novo servidor, por fim, pela utilização de mecanismos de *load balancing*, a carga era distribuída também para este novo elemento da rede. Isto sem dúvida é uma solução muitíssimo viável ainda nos dias de hoje para solucionar a maior parte dos problemas relativos à falta de capacidade do sistema, não obstante de ser a mais dispendiosa em detrimento de outras alternativas também eficazes, quer na óptica da engenharia, quer na óptica financeira, como veremos adiante.

Vive-se numa era onde os computadores, quer de secretária quer servidores, são cada vez mais poderosos a nível de processamento e memórias, sejam elas voláteis ou não. Os processadores evoluíram a um ponto em que é possível que cada computador execute sobre si mais do que um sistema operativo, quase sem impacto para o sistema operativo nativo; sistema anfitrião que iniciou e sustenta os restantes sistemas operativos, no mesmo computador, físico.

Este tipo de tecnologia tem a designação de Virtualização de Sistemas, uma vez que há vários sistemas de virtualização como veremos adiante. Com esta tecnologia é possível aos sistemas operativos abstrair-se do hardware, falando com uma API que serve de *gateway* entre os sistemas operativos e o hardware.

Há no mercado soluções que facilitam a gestão destes sistemas virtuais, isto para os fabricantes de maior renome, no entanto outras soluções de virtualização mais simples e também não tão conhecidas carecem destes interfaces gráficos que mais não são do que facilitadores à gestão corrente, do dia-a-dia.

	Elaborado por: Francisco Alves Cabrita - 20060645 Luís Monteiro Castro - 20060658
--	---

O presente trabalho propõe a criação de um protótipo de um *front-end* web capaz de dar aos administradores de sistemas uma ferramenta para configurar, criar e distribuir estes sistemas virtuais pela sua rede, permitindo-os assim quer manter um controlo apertado sobre os sistemas existentes, quer delegar alguma capacidade administrativa a clientes destes sistemas virtuais, que neste contexto podem ser outros administradores de sistemas de um outro departamento de uma dada empresa ou outros.

Este projecto foi levado a cabo com duas constantes sempre bem presentes. A eficácia do sistema desenvolvido e a simplicidade quer do seu interface web quer dos seus mecanismos de funcionamento. Para isso, o levantamento de requisitos foi confrontado com diversos cenários e no final reduzido ao que os intervenientes deste projecto pensam ser o extremamente necessário para concretizar a tarefa proposta, sempre seguindo a mesma linha de desenvolvimento, cumprir o objectivo planeado, desenvolvendo uma ferramenta robusta.

No decorrer do trabalho são trazidos vários conceitos à luz do dia. São aqui explicados em detalhe todos os elementos que constituem esta problemática tais como:

- Tecnologia de virtualização utilizada para gerir os sistemas virtuais;
- Sistema operativo em causa, uma vez que tem particular interesse devido à tecnologia de virtualização/enclausuramento utilizada só estar presente numa variante de UNIX, as FreeBSD Jails;
- Componente aplicacional web desenvolvida em Ruby para gerir e replicar estes sistemas virtuais via web interface.

No trabalho também são apresentados ao de leve alguns dos demais sistemas de virtualização mais utilizados nos dias de hoje, tais como VMware, XEN, KVM, a fim de se explicar as razões pelas quais a escolha recaiu sobre as FreeBSD Jails.

Com este interface o administrador de sistemas é capaz de fazer uma administração e monitorização centralizada de todos os servidores virtuais, de uma forma rápida e livre de erros.

	Elaborado por: Francisco Alves Cabrita - 20060645 Luís Monteiro Castro - 20060658
--	---

4 - ENQUADRAMENTO TEÓRICO

4.1 - O Problema

A criação de servidores virtuais é uma tarefa que pode ser conseguido pela implementação de algo muito simples até algo bastante complexo, tudo depende em parte não só da tecnologia utilizada para levantar estes serviços virtuais mas também da arquitectura da nossa solução. Podemos ter cenários mistos onde a gestão de servidores virtuais é considerado caótico mesmo em ambientes simples. Contudo, qualquer rede deste tipo é susceptível a erros humanos e à má gestão por parte dos administradores de sistema.

Uma rede física começa sempre por ser implementada como uma rede clara, normalmente esta é milimetricamente idealizada e trazida para o papel, por fim esta é implementada no *datacenter* para cumprir com uma dada função, disponibilizando serviços.

É do senso comum que um dado computador, seja ele de secretária ou mesmo servidor, independentemente do sistema operativo utilizado, tem de ser mantido de forma correcta, persistente e com algum controlo na periodicidade em que estas actualizações e limpezas são efectuadas. A incorrecta execução destas tarefas ou mesmo a falta delas pode trazer consequências, no pior dos casos algo desastrosas aos sistemas, ou então deixar os sistemas operativos num estado em que é impossível ou muito arriscado proceder à actualização de *software* ou mesmo bibliotecas residentes no sistema.

É também sabido pela experiência que no decorrer do tempo de vida do servidor, este é por vezes alvo de más práticas de administração no que diz respeito à utilização dos mesmos servidores físicos para outros fins; testes ou aplicação de remendos é o mais usual.

A título de exemplo tomamos um *webserver* Apache com módulos PHP que comunica com um servidor de Base de Dados MySQL. Não é correcto que este MySQL *daemon* seja disponibilizado no mesmo servidor físico, não só pelo sub-aproveitamento do sistema físico que deve estar optimizado para grandes volumes de dados escrita/leitura nos discos.

	Elaborado por: Francisco Alves Cabrita - 20060645 Luís Monteiro Castro - 20060658
--	---

Mediante a arquitectura podemos distribuir o SGDB num *cluster Master-Master* ou outro cenário *Master-Slaves*, etc; mas também pelo factor que mais chama aqui à atenção neste trabalho, os contextos virtuais.

Este conceito de contextos virtuais parte da importância de agregar software e serviços por contexto, quer estejam eles numa fase de "*Development, Integration, Staging ou Production*". É uma boa prática isolar um *webserver* num contexto deste tipo, de *webserver*s, deixando de fora e também isolado numa zona específica, os restantes serviços como por exemplo, MySQL, *Reverse Proxies*, DNS, etc. Este ponto será discutido em detalhe nos capítulos seguintes, abordando as muitas vantagens desta metodologia, arquitectura.

De volta ao tópico, no decorrer do ciclo de vida de um servidor é comum que por via das circunstâncias, do cliente, do negócio ou mesmo do CEO que ordena que se levante o serviço de CRM seja em que servidor for, é reconhecido que um servidor que começou por ser um simples *Application Server*, é hoje também utilizado como NTP *Server*, DNS-Cache porque "dá jeito" ou "tem mesmo de ser" e um Squid em modo *Reverse Proxy* para resolver "aquela" falha de segurança de *SQL-Injection* que o CRM tem; isto claro a título de exemplo. É claro que este cenário é um pouco exagerado mas muitos *datacenters* sofrem silenciosamente com problemas semelhantes a este, degradando a performance global dos serviços e tornando-os mutuamente inseguros, vulneráveis.

Existem hoje bastantes métodos de virtualização assim como ferramentas para facilitar, automatizar e simplificar a administração destes sistemas, sejam elas ferramentas *web based* ou nativas, sejam elas orientadas a Unices; produtos Microsoft ou ainda híbridos.

Recorrendo a produtos existentes no mercado, como por exemplo VMware, Xen, KVM entre outros, sejam software livre ou comercial, pode-se implementar um sistema de virtualização em Linux, capaz de albergar sistemas virtuais convidados *Guests* Linux ou Microsoft Windows entre outros. De notar que para as tecnologias acima referidas, todos os *Guests* em causa são compreendidos por instalações completas dos sistemas operativos, o que quer dizer é que para cada sistema virtual é feita uma instalação automática ou não, de todo um sistema operativo, *Kernel* e *Userland*, obtendo-se assim como que um sistema total, dentro de um outro, como demonstra a seguinte figura.

	Elaborado por: Francisco Alves Cabrita - 20060645 Luís Monteiro Castro - 20060658
--	---

Ilustração 1: Infra-estrutura Virtual

No diagrama seguinte é mostrado um exemplo de como está dividido um sistema de virtualização simples, ou seja, sem descrever que tipo de virtualização se trata; *Hardware Virtualization*, *Full Virtualization*, *Paravirtualization*, etc. No ponto 5 é tomado este ponto em mais detalhe a fim de se anotar as diferenças entre os mesmos.

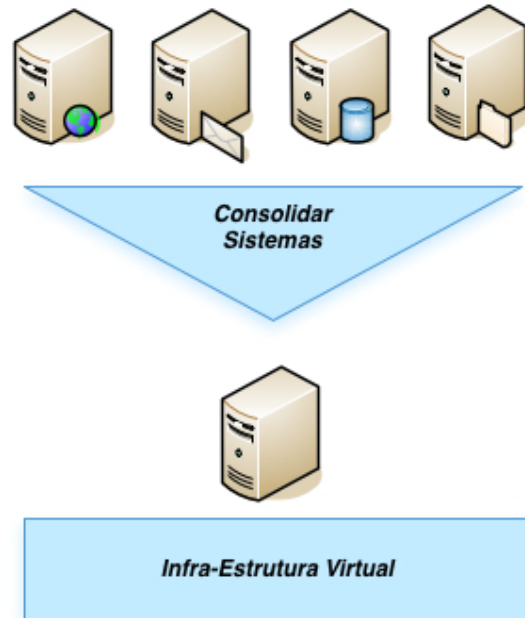
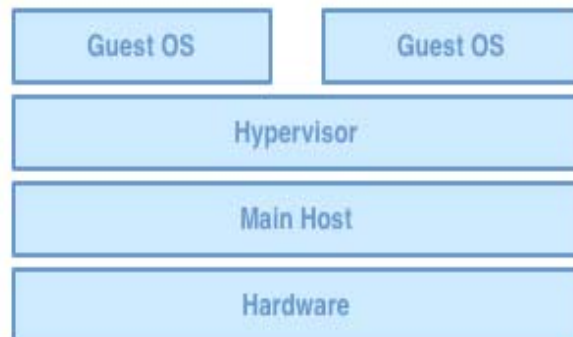


Ilustração 2: Arquitectura Base

Na ilustração 2 verifica-se que para o mesmo hardware há um sistema operativo base, Main Host, que pode ser Linux, UNIX ou Microsoft Windows, este pode disponibilizar uma camada designada de hypervisor que é a API usada para comunicar e fazer a demais gestão de recursos entre o sistema anfitrião e os guests subjacentes. Quando se diz pode disponibilizar, isto quer dizer que, está directamente relacionado com a escolha do sistema de virtualização, pois o Hypervisor pode ou não estar presente.

Designa-se por *Anfitrião*, *Main Host* ou *Dom* ao sistema base que recebe os servidores virtuais.

Designa-se por *Guest*, *VM (Virtual Machine)* ao sistema virtual, instalado “dentro” do Anfitrião. Em alguns sistemas também se dá o nome de “*Virtual Nodes*”



Os sistemas de gestão de servidores virtuais são ferramentas que actuam nos três *layers* mas que falam muito em particular com o *Main host* e com o *Hypervisor*.

Talvez por ser pouco conhecido o mundo da virtualização em FreeBSD mais em particular a tecnologia de enclausuramento de serviços em *Userlands*, *Jails*; foi visto como uma oportunidade desenvolver esta solução para este caso em particular. Até ao momento ainda não foi disponibilizada no mundo open source ou no mercado a nível comercial, nenhuma solução de gestão para este caso.

De uma forma muito pragmática, o problema é de simples identificação: é necessário desenvolver uma solução para gerir servidores virtuais em FreeBSD, ferramenta essa que permita por via de um interface web, a fácil e rápida distribuição destes sistemas.

4.2 - Levantamento dos Requisitos

- Autenticação Básica;
Necessidade de um acesso básico de autenticação, de forma a limitar o acesso à área de gestão das máquinas virtuais.
- Listar servidores virtuais;
Permitir listar as diferentes máquinas virtuais, sob gestão do interface, bem como os detalhes dos respectivos ambientes.
- Criar servidores virtuais;
Permitir a fácil, simples e rápida criação de novos ambientes virtuais, numa lógica quase de "*one click install*".
- Eliminar servidores virtuais.
Acesso rápido a "destruição" das respectivas máquinas virtuais.

	Elaborado por: Francisco Alves Cabrita - 20060645 Luís Monteiro Castro - 20060658
--	---

4.3 - Descrição do Sistema

A solução proposta é composta por um sistema operativo UNIX *based*, no caso o FreeBSD, com tecnologia que permite a virtualização de sistemas através de *Jails*.

Nos *layers* adjacentes, temos na vertente aplicacional o interface web que dá "corpo" à solução desenvolvido em Sinatra uma *micro-framework* web escrita em Ruby. Como *middleware* de ligação entre o interface e o sistema operativo que nos permite a gestão das diversas máquinas virtuais utilizamos o Ezjail, como o nome indica é uma forma simples e fácil de coordenar e gerir as máquinas virtuais existentes no sistema anfitrião.

Ruby, a linguagem de programação

O Ruby é uma linguagem de muito alto nível. O seu criador Yukihiro "matz" Matsumoto, uniu partes das suas linguagens favoritas (Perl, Smalltalk, Eiffel, Ada, e Lisp) para formar uma nova linguagem que equilibra a programação funcional com a programação imperativa.

Matz diz com frequência que está a "tentar tornar o Ruby natural, não simples", de uma forma que reflecta a vida. Sobre isto, acrescenta:

"O Ruby é simples na aparência mas muito complexo no interior, tal como o corpo humano."

Inicialmente, Matz estudou outras linguagens em busca de encontrar uma sintaxe ideal. Recordando a sua busca, disse:

"Eu queria uma linguagem interpretada que fosse mais poderosa que o Perl e mais orientada aos objectos do que o Python³."

	Elaborado por: Francisco Alves Cabrita - 20060645 Luís Monteiro Castro - 20060658
--	---

Em Ruby, tudo é um objecto. Cada parcela de informação e código podem receber as suas próprias propriedades e acções. A Programação orientada aos objectos denomina as propriedades como *variáveis de instância* e as acções como *métodos*. A aproximação pura, da orientação aos objectos do Ruby, é geralmente demonstrada pelo seguinte excerto de código que aplica uma acção a um número.

```
5.times { print "Nós *amamos* o Ruby -- é fantástica!" }
```

Em muitas linguagens, números e outros tipos primitivos não são objectos. O Ruby segue a influência da linguagem Smalltalk em atribuir métodos e variáveis de instância a todos os seus tipos. Esta abordagem facilita a utilização do Ruby, uma vez que as regras que se aplicam aos objectos aplicam-se a tudo em Ruby.

O Ruby é visto como uma linguagem flexível, uma vez que permite aos seus utilizadores alterar partes da Linguagem. Partes essenciais do Ruby podem ser removidas ou redefinidas à vontade. Partes existentes podem ser acrescentadas. O Ruby tenta não restringir o programador.

Por exemplo, a adição é realizada com o operador mais (+). Mas, se preferir utilizar a palavra escrita *plus*, poderia adicionar esse método à classe nativa do Ruby Numeric.

O Ruby é rico em outras características, entre as quais se destacam as seguintes:

- Capacidade de tratamento de excepções, tal como o Java ou Python, por forma a facilitar o tratamento de erros.
- Um verdadeiro *mark-and-sweep garbage collector* para todos os objectos Ruby. Não é necessário manter contadores de referência em bibliotecas de extensão (*extension libraries*). Tal como Matz diz, "Isto é melhor para a sua saúde."
- Escrever extensões C em Ruby é mais fácil do que em Perl ou Python, com uma API refinada para chamar Ruby desde o código C. Isto inclui chamadas para embeber Ruby em software externo por forma a ser utilizado como uma linguagem interpretada dentro do software. Uma interface SWIG também se encontra disponível.

	Elaborado por: Francisco Alves Cabrita - 20060645 Luís Monteiro Castro - 20060658
--	---

- O Ruby pode carregar bibliotecas de extensão (*extension libraries*) dinamicamente se um Sistema Operativo o permitir.
- O Ruby tem um sistema de *threading* independente do Sistema Operativo. Portanto, para todas as plataformas nas quais o Ruby corre, temos *multithreading*, independentemente de o Sistema Operativo o suportar ou não.
- O Ruby é altamente portátil: é desenvolvido principalmente em ambiente GNU/Linux, mas trabalha em muitos tipos de ambientes UNIX, Mac OS X, Windows 95/98/Me/NT/2000/XP, DOS, BeOS, OS/2, etc.

Sinatra, a framework de desenvolvimento

Sinatra é um *Domain-Specific-Language* que permite criar aplicações web em Ruby.

Tem um conjunto de funcionalidades minimalista, deixando o programador usar as ferramentas que melhor se adequam aos requisitos da aplicação.

O *framework* Sinatra não assume muito sobre a aplicação que está a ser desenvolvida, à parte de:

- Vai ser escrita em Ruby;
- Vai ter URL's.

Em Sinatra pode escrever pequenas aplicações *ad hoc* como *middleware* ou interfaces de API's ou mesmo aplicações maduras.

Pela utilização das *Rubygems* (módulos Ruby que implementam funcionalidades) ou outras bibliotecas em Ruby, podemos expandir a aplicação desenvolvida em Sinatra.

	Elaborado por: Francisco Alves Cabrita - 20060645 Luís Monteiro Castro - 20060658
--	---

Sinatra não é um típico MVC (*Model Viewer Controller*) como o muito conhecido *framework* Ruby on Rails. Em Sinatra os pedidos são efectuados contra um conjunto de rotas definidos pelos respectivos URLs que fazem disparar procedimentos internos no código da aplicação. Isto coloca no programador a responsabilidade de escrever código limpo, correctamente organizado, podendo criar e separar os controladores da vista e inclusive, caso se verifique, dos modelos.

EzJail, o *middleware*

Ezjail, consiste em tornar o mais simples possível a criação e gestão das Jails, utilizando o mínimo de recursos do sistema. Todas as referências ao termo Jail, são referentes a ambientes virtuais em FreeBSD, com o seu próprio *hostname*, endereço IP e a sua *Jail root*.

O página de manual do comando jail, mostra a forma de criar jails, no entanto no caso de se pretender diversas jails, *Jail Directory Tree* rapidamente utiliza uma grande parte dos recursos físicos do disco (espaço). *Ezjail* evita este problema de alocação de espaço, utilizando o funcionalidade do FreeBSD *nullfs*. A maioria da base do sistema (/bin, /boot, /sbin, /lib, /libexec, /rescue, /usr/{bin, include, lib, libexec, ports, sbin, share, src}, apenas existe numa das cópias no *host system* e está a ser “montada” como *read-only* em todas as restantes *jails* através de *nullfs*. Desta forma as *jails*, são extremamente pequenas, podendo ocupar apenas cerca de 2Mb cada uma e apenas consistem em *softlinks* para a *main jail mount point* e para directorias não partilhadas como é o caso do (/etc, /usr/local, etc.)

O *Ezjail*, oferece uma série de vantagens, nomeadamente:

- Permite poupar espaço em disco, estrutura de dados (*inodes*), e até memória, já que o sistema apenas necessita de guardar uma cópia dos binários de sistema para todas as *Jails*.
- Permite a actualização de todas as *jails* através de um único directório.
- Intrusos nas *jails* estão limitados e não conseguem instalar *standard rootkits*, já que o sistema base está “montado” em modo *read-only*.

	Elaborado por: Francisco Alves Cabrita - 20060645 Luís Monteiro Castro - 20060658
--	---

- O *ezjail* está inteiramente escrito em sh (*shell script*), não existe qualquer necessidade de instalar outras linguagens de script no *host-system*.
- Como o sistema base é facultado através de *softlinks*, os utilizadores das *jails* podem optar por não utilizar o *mounted world*.
- Menor complexidade é sinónimo de maior segurança.

Ezjail, consiste em dois scripts: *ezjail-admin* e *ezjail.sh*, o primeiro *script* é usado para criar, alterar e apagar *jails*, o segundo para iniciar, parar e reiniciar as respectivas *jails*.

4.4 - Análise de Requisitos

Tendo como base o pensamento da *Layered Architecture*, optou-se por procurar e utilizar software que permite de uma forma simples alcançar o objectivo proposto, dessa análise concluiu-se que haveria um maior numero de vantagens na utilização de *frameworks* de desenvolvimento e de um *middleware* para a ligação entre a aplicação a o sistema operativo.

A *framework Sinatra*, permite efectuar um desenvolvimento mais rápido, mais cuidado e sem tropeçar em erros básicos na programação de uma aplicação web. A utilização de uma linguagem de muito alto nível facilita a escrita da aplicação e o avanço rápido no *debug* e resolução de problemas.

O *middleware ezjail*, permite em termos de desenvolvimento uma total abstracção da mensagem/comandos de sistema operativo para a criação e gestão das *Jails*. A utilização destas aplicações para desenvolvimento da solução proposta tem também a vantagem de ambas serem *task-oriented* ou seja, sem grandes capacidades de realizar outro tipo de tarefas e como tal menos susceptíveis a falhas.

	Elaborado por: Francisco Alves Cabrita - 20060645 Luís Monteiro Castro - 20060658
--	---

5 - TECNOLOGIAS

5.1 - Conceitos básicos de virtualização

A virtualização é uma tecnologia de software desenvolvida com o intuito de retirar um maior proveito do hardware existente, já que antes, uma máquina servia apenas para executar um único sistema operativo, fazendo com que a maioria das máquinas estivesse sub-aproveitada. A virtualização permite executar diversas máquinas virtuais sobre a mesma máquina física, partilhando recursos dessa máquina entre os vários ambientes (máquinas virtuais), conseguindo assim, executar diferentes sistemas operativos e aplicações sobre a mesma máquina física, garantindo uma maior eficiência e aproveitamento do hardware.

O termo “virtualização” é utilizado actualmente de diversas formas e em momentos tão distintos do nosso dia-a-dia, sobretudo no que diz respeito à infra-estrutura de recursos computacionais. Ao dizer que tal suporte ou arquivo está virtualizado, imagina-se que esteja alocado em algum ponto do ciberespaço, no sentido de que existe mas é intangível. É importante notar que a virtualização se trata de uma tecnologia acessível a todos e não apenas às empresas com maiores orçamentos nas TI's. Embora com a mesma finalidade, o conceito de “virtualização” encontra muitas definições na computação.

A virtualização está em mudança, em quase todos os aspectos, na forma e método, de como gerir os sistemas, *storages*, redes, segurança, sistemas operativos e aplicações, trazendo inúmeros benefícios e vantagens aos departamentos de TI das empresas. Em suma, podemos destacar as seguintes vantagens:

- Gestão centralizada;
- Instalações simplificadas e homogéneas;
- Simplicidade na execução de *backups e restore*;
- Maior segurança;

	Elaborado por: Francisco Alves Cabrita - 20060645 Luís Monteiro Castro - 20060658
--	---

- Simplicidade no suporte e manutenção;
- Acesso controlado a dados sensíveis e à propriedade intelectual mantendo-os seguros dentro do *datacenter* da empresa;
- Independência de hardware;
- Maior disponibilidade de novos desktops;
- Migração de sistemas/serviços para novo hardware de forma simples e directa;
- Maior facilidade na distribuição e recuperação de sistemas/serviços;
- Compatibilidade total com as aplicações.

Os servidores virtuais proporcionam um ambiente que disponibiliza uma máquina virtual em regime de exclusividade. Cada servidor virtual consiste numa área protegida e privada que funciona como se de um servidor independente se tratasse.

A virtualização permite que múltiplos clientes/recursos partilhem os elevados custos de hardware e conectividade, eliminando ao mesmo tempo a necessidade de preocupações com a manutenção, sem sacrificar o desempenho e liberdade de executar as aplicações que pretenda no servidor.

O segredo da virtualização está na forma como uma máquina física é partilhada: apesar de vários clientes/recursos partilharem o mesmo hardware, eles não partilham o mesmo software. Todos os servidores têm a sua estrutura completa de directórios e um conjunto de aplicações dedicadas (Exemplo: servidor web, servidor de base de dados, etc.), permitindo que, no mesmo servidor físico, ao reiniciar um dos servidores virtuais, os restantes não são afectados.

Adicionalmente, apesar do hardware ser partilhado, a tecnologia de virtualização gere a partilha de recursos, processos, memória, conectividade, etc. de forma a que nenhum dos servidores virtuais do mesmo servidor físico seja afectado. É garantida uma forma de evitar que um único servidor virtual abuse dos recursos partilhados e, em último caso, que comprometa as restantes aplicações de funcionarem com base num desempenho consistente.

	Elaborado por: Francisco Alves Cabrita - 20060645 Luís Monteiro Castro - 20060658
--	---

É também possível definir servidores privados virtuais que tenham recursos de memória e processadores (neste caso cores - núcleos - inteiros) que estão totalmente dedicados ao seu usufruto, dando assim garantias totalmente semelhantes a um servidor dedicado. Para igualar as características de fiabilidade, segurança e flexibilidade de um servidor virtual é necessário um investimento bastante superior num servidor dedicado.

A virtualização, se os dados estiverem num *storage* partilhada, também permite um conjunto de funcionalidades adicionais muito interessantes, como por exemplo a migração muito rápida, quase instantânea de um servidor virtual que está alojado num determinado servidor físico para um outro servidor físico. Isto é extremamente útil, por exemplo, quando existe problemas de hardware num dos servidor, ou então para aumentar os níveis de serviço, adicionando mais um servidor à *pool* de recursos virtuais.

Devido ao facto de um servidor virtual se assemelhar e comportar como um servidor dedicado, as possibilidades são quase ilimitadas podendo ser usado para disponibilizar o alojamento Web de um único site, ou hospedar um complexo sistema de comércio electrónico ou base de dados.

Algumas ideias chave sobre os servidores virtuais:

- Disponibilizam o seu próprio ambiente de trabalho. Permitem que cada servidor opere independentemente dos restantes servidores virtuais alojados no mesmo servidor físico;
- Cada servidor virtual corre as suas aplicações de forma independente;
- Com um servidor virtual, o cliente dispõe de permissões administrativas sobre a sua máquina virtual, podendo inclusivamente proceder ao reboot do seu servidor, sendo que este tipo de operações é realizado sem que os outros clientes com servidores virtuais alojados no mesmo servidor físico sejam afectados;
- Um servidor virtual actua de forma igual a um servidor dedicado e pode ser configurado pelo cliente da forma que este entender dado que cada servidor virtual é uma área privada e protegida que opera como um servidor independente;

	Elaborado por: Francisco Alves Cabrita - 20060645 Luís Monteiro Castro - 20060658
--	---

- Com um servidor virtual poderá ter o seu próprio servidor web, servidor de base de dados, hospedar múltiplos domínios, etc.;
- Todos os servidores virtuais possuem a sua estrutura de directorias e um conjunto de várias aplicações pré instaladas, possuindo os seus próprios *port numbers*, endereços IP, regras de *routing*, etc.;
- Um servidor virtual privado pode ser acedido como se de um servidor dedicado se tratasse, através do seu próprio endereço IP, podendo ter diferentes características ou recursos disponíveis, de acordo com as necessidades de cada cliente;
- Soluções baseadas em servidores virtuais partilham os elevados custos de hardware, conectividade, manutenção, etc..

5.2 - Métodos de virtualização

Abaixo ficam enunciados e descritas algumas características dos principais sistemas de virtualização mais conhecidos, evoluídos e utilizados desde ambientes corporativos passando por universidades, centros de investigação e claro, pequenas e médias empresas.

Xen

O *Xen hypervisor* é uma camada de software que é executada directamente na máquina em substituição do sistema operativo, permitindo desta forma a execução em paralelo de diversos "guest" sistemas operativos sobre o mesmo hardware de forma concorrente.

Suporta processadores com a arquitectura x86, x86_64, Itanium, Power PC, and ARM, permite ao *hypervisor* Xen, a execução de uma vasta variedade de dispositivos e actualmente suporta Linux, NetBSD, FreeBSD, Solaris, Windows, e outros sistemas operativos comuns.

A comunidade Xen.org mantém e desenvolve o *hypervisor* Xen como uma solução "free" sob a licença *GNU General Public License*.

	Elaborado por: Francisco Alves Cabrita - 20060645 Luís Monteiro Castro - 20060658
--	---

Vmware

Propriedade da empresa VMware Inc., trata-se de um software de virtualização semelhante aos restantes. Permite executar diversos sistemas operativos em paralelo. Apresenta uma série de funcionalidades que acabam por tornar esta ferramenta, como a número um, no que diz respeito a virtualização. Nomeadamente, *Hypervisor, VMFile System, Virtual SMP, Update Manager, Virtual Center Agent, Consolidated backup, HA, VMotion, Storage VMotion, Distributed Resource Scheduler*.

KVM - Kernel-based Virtual Machine

É um motor de virtualização, relativamente recente e simples, no entanto bastante capaz, já que conseguiu aceder directamente ao Kernel do Linux, dando a capacidade de virtualização ao Kernel nativo do Linux. Como a virtualização do KVM é baseada em hardware, não é necessário qualquer alteração sobre os sistemas operativos "convidados", pode suportar qualquer plataforma baseada em Linux, desde que seja instalado sobre um processador suportado.

A implementação do KVM foi levada a cabo como se de um módulo do kernel se tratasse, o desenvolvimento do hypervisor de KVM, acabou por ser mais simples, já que ao contrário dos restantes motores de virtualização, não precisaram de desenvolver um "novo" kernel de sistema operativo de base.

Integrando as capacidades do hypervisor como um módulo de kernel de Linux, permitiu melhorar a gestão e a performance dos sistemas virtualizados. Estas terão sido os principais motivos que levaram os programadores a incluírem o KVM no kernel de Linux.

	Elaborado por: Francisco Alves Cabrita - 20060645 Luís Monteiro Castro - 20060658
--	---

VirtualBox

Este software de virtualização foi criado pela empresa Innotek, tendo sido adquirido e atualizado pela Oracle em 2009.

O VirtualBox visa criar ambientes para a instalação de sistemas distintos. Permite a instalação e utilização de um sistema operativo dentro de outro dando suporte real a *software* de outros sistemas, porém compartilham fisicamente o mesmo hardware.

Com o intuito de tornar mais fácil o controlo de várias *interfaces* de uma só vez, este software tem um desenho modular com interfaces de programação interna bem definidas e um desenho cliente/servidor. Possui ainda um *Software Development Kit* completo com o propósito de não existir um corte na fonte aquando da introdução de uma nova interface.

O armazenamento das definições de configuração de máquinas virtuais é efectuado utilizando o ficheiros em formato XML, permitindo a transferência para outros computadores.

Para os Sistemas Operativos Microsoft Windows e Linux, o VirtualBox possui um software especial com o intento de melhorar o desempenho e a integração entre a máquina hospedeira/anfitrião e a máquina virtual.

Com o objectivo de facilitar a troca de dados entre anfitrião e os servidores virtuais, este software consente a declaração de directórios como "directórios partilhadas" sendo possível acedê-los dentro das instâncias virtuais.

OpenVZ

Solução também open source de virtualização, *container-based* para Linux. O conceito é parecido a todos os outros, passa pela criação independente e isolada de *containers* seguros, onde pode ser instalado qualquer outro sistema *Linux based*. Com gestão independente, cada *container* é executado como se de um servidor físico único se tratasse, sem conflitos entre si.

Mais uma solução *free* sobre o licenciamento GNU GPL.

	Elaborado por: Francisco Alves Cabrita - 20060645 Luís Monteiro Castro - 20060658
--	---

Solaris Zones

A tecnologia de partição do Solaris Zones é usada na virtualização de Sistemas Operativos, oferece um ambiente isolado e seguro, denominado regiões (*zones*), produzindo-se assim um ambiente de execução de aplicações em que os processos estão isolados do resto do sistema, comportando-se como se estivesse a executar o seu próprio Sistema Operativo, evitando conflitos com outros executados em zonas distintas, podendo ser facilmente destruída e recriada, caso o resultado não seja o desejado. Cada zona tem a sua própria identidade e fica separada da subcamada do *hardware*. Este software permite a transferência de recursos, incluindo CPU, memória física, que não foram utilizados para outros recipiente, conforme necessário. Todas as zonas podem ser criadas ou iniciadas manualmente, programaticamente ou automaticamente quando o sistema for iniciado, existindo ainda a possibilidade de desligar uma zona dentro de outra ou da zona global, bem como de fazer atualizações em todas ou em uma específica.

Existem zonas de dois tipos: zona global é a instância principal do Solaris; zona não-global sendo esta um ambiente virtual criada para hospedar aplicações.

	Elaborado por: Francisco Alves Cabrita - 20060645 Luís Monteiro Castro - 20060658
--	---

A tabela abaixo resume as características de regiões globais e não-globais.

Tipo de região	Característica
Global	<ul style="list-style-type: none"> • O sistema atribui o ID 0; • Fornece uma instância única do kernel do Solaris que é inicializável e executada no sistema; • Contém uma instalação completa dos pacotes de software de sistema do Solaris; • Pode conter pacotes de software adicionais ou software, diretórios e arquivos adicionais ou outros dados não instalados através de pacotes; • Oferece uma base de dados de produto completo e consistente que contém informações sobre todos os componentes de software instalados na região global; • Armazena informações de configuração específicas somente da região global, como o nome do nó da região global e a tabela do sistema de arquivos; • É a única região que reconhece todos os dispositivos e todos os sistemas de arquivos; • É a única região com conhecimento da existência e da configuração da região não global; • É a única região a partir da qual uma região não global pode ser configurada, instalada, gerida ou desinstalada.
Não global	<ul style="list-style-type: none"> • O sistema atribui um ID de região quando a região é inicializada; • Compartilha operação no kernel do Solaris inicializado a partir da região global; • Contém instalado um subconjunto dos pacotes de software completos do <i>Solaris</i>; • Contém pacotes de software do Solaris compartilhados a partir da região global; • Pode conter pacotes de software adicionais instalados não compartilhados a partir da região global; • Pode conter software, directórios e arquivos adicionais, e outros dados criados na região não global que não são instalados através de pacotes ou compartilhados a partir da região global; • Tem uma base de dados do produto completo e

	Elaborado por: Francisco Alves Cabrita - 20060645 Luís Monteiro Castro - 20060658
--	---

	<p>consistente que contém informações sobre todos os componentes de software instalados na região, presentes na região não global ou somente leitura compartilhados a partir da região global;</p> <ul style="list-style-type: none"> • Não reconhece a existência de quaisquer outras regiões; • Não pode instalar, gerir ou desinstalar outras regiões, inclusive ela mesma; • Tem informações de configuração específicas somente dessa região não global, como o nome do host da região não global e a tabela do sistema de arquivos; • Pode ter sua própria configuração de fuso horário
--	---

No que concerne à memória esta é gerida pela zona global, existindo uma optimização dos recursos de memória, uma vez que os executáveis usados e bibliotecas podem ser carregados uma só vez e partilhados entre as várias zonas existentes no sistema.

O Solaris Zone suporta o uso de múltiplos tipos de armazenamento como *Direct Attached Storage*, *Network Attached Storage*, *Storage Area Network* e múltiplos tipos de sistema de arquivos.

Existem dois modelos de *layout* do arquivo do sistema operativo para uma zona não global:

Whole root: é uma cópia completa do sistema operativo no disco.

Sparse root: é uma zona que não inclui a sua cópia privada dos arquivos de sistema operativo e demais bibliotecas.

As configurações de rede são efectuadas a partir da zona global aquando da criação de cada zona, podendo ser alteradas só a partir desta.

As comunicações entre zonas estão apenas disponíveis através das interfaces virtuais de rede.

	<p>Elaborado por: Francisco Alves Cabrita - 20060645 Luís Monteiro Castro - 20060658</p>
--	--

Estes dois últimos sistemas de virtualização, mais correctamente, sistemas de enclausuramento de serviços, são o parente mais próximo das FreeBSD Jails. Todas estas tecnologias recriam *userlands POSIX* de si próprias. Não é possível de todo, com estas tecnologias, virtualizar outros sistemas operativos que não os derivados do anfitrião; ou seja; Jails em FreeBSD são FreeBSD, *containers* em Linux, são Linux e por fim, Zonas criadas em Solaris apenas podem conter instâncias de Solaris.

Para os mais puristas, estas tecnologias não fazem parte do mundo da virtualização de sistemas por não poderem ser a base de um qualquer sistema operativo, no entanto, na óptica aplicacional, os serviços instalados nestes contentores/zonas virtuais, são eles também servidores virtuais e em conjunto formulam uma solução virtual.

5.3 - Análise aos sistemas de virtualização

	Vmware	Jails	XEN	virtualbox
Tipo	Nível OS	<i>Container</i>	Paravirtualização/HVM	<i>Full Virtualization</i>
Desempenho	Alto	Nativo	Alto	Baixo
Isolamento	Alto	Alto	Alto	Alto
Multiplos OS	Sim	Não	Sim	Sim
Virtual Networking	Sim	Sim	Sim	Sim
Clustering	Não	Não	Sim	Não
Finalidade	<i>Hosting, Datacenter</i>	<i>Hosting, Datacenter</i>	<i>Hosting, Datacenter</i>	Pessoal
Interface gráfico	Sim	Não	Não, na versão opensource	Sim

	Elaborado por: Francisco Alves Cabrita - 20060645 Luís Monteiro Castro - 20060658
--	---

5.4 - O Sistema de Virtualização Utilizado

FreeBSD Jails

O sistema operativo FreeBSD, UNIX based, proporciona um mecanismo de limitação de processos denominado *Jails*, criado para aumentar a segurança do sistema UNIX, permitindo múltiplos utilizadores e um administrador (root) privilegiado em cada partição, embora restrinja as actividades do último. O administrador do servidor FreeBSD pode particionar a máquina em prisões separadas, e fornecer acesso a cada uma delas forma autónoma, como se fosse um sistema separado, nunca perdendo o controlo sobre todo o ambiente.

As FreeBSD *Jails* são caracterizadas por elementos:

- sub-estrutura do directório - será o ponto de partida onde está instalada a *Jail*, funcionando como os ramos de uma árvore, todos independentes mas dependentes de um único tronco (directório);
- *Hostname* - o nome do servidor virtual que será utilizada dentro de cada *Jail*;
- Endereço IP - cada *Jail* criada irá ter o seu próprio IP que não poderá ser alterado de forma alguma e é normalmente um endereço de *alias* para uma interface de rede existente.
- Um comando - o nome do caminho de um executável para ser executado dentro da prisão. Este é relativo ao directório raiz do ambiente virtual.

	Elaborado por: Francisco Alves Cabrita - 20060645 Luís Monteiro Castro - 20060658
--	---

Alguns administradores de *jails* podem dividi-las em dois tipos: "*complete*" *jails* , que se assemelham a um sistema FreeBSD real, e "*service*" *jails*, dedicado a uma aplicação ou serviço, possivelmente executado com privilégios. Esta é apenas uma divisão conceptual não sendo o processo de construção de uma *Jail* afectado.

Os processos iniciados dentro de uma Jail, são exclusivos daquela Jails, sendo os processos filhos também lá aprisionados.

No momento em que cada Jail é criada, são passados alguns argumentos que constroem o seu funcionamento, um destes argumentos é o *path* da *root jail*; ou seja, qual é o caminho sobre o qual todos os seus serviços vão ser montados e executados. Todos os processos iniciados nesta Jail, não podem manipular ficheiros fora desta zona. A integridade e a confidencialidade dos ficheiros que estão fora da Jail ficam garantidas.

Os processos confinados a uma Jail, mesmo com privilégios de administrador, têm diversas restrições; não podendo:

- reconfigurar o núcleo do sistema;
- carregar ou remover módulos do núcleo;
- mudar configurações de rede;
- montar ou desmontar sistemas de ficheiros/volumes;
- alterar as configurações do núcleo em tempo real;
- redefinir políticas de segurança;
- criar novos dispositivos físicos;
- aceder a recursos que não pertençam ao seu próprio domínio.

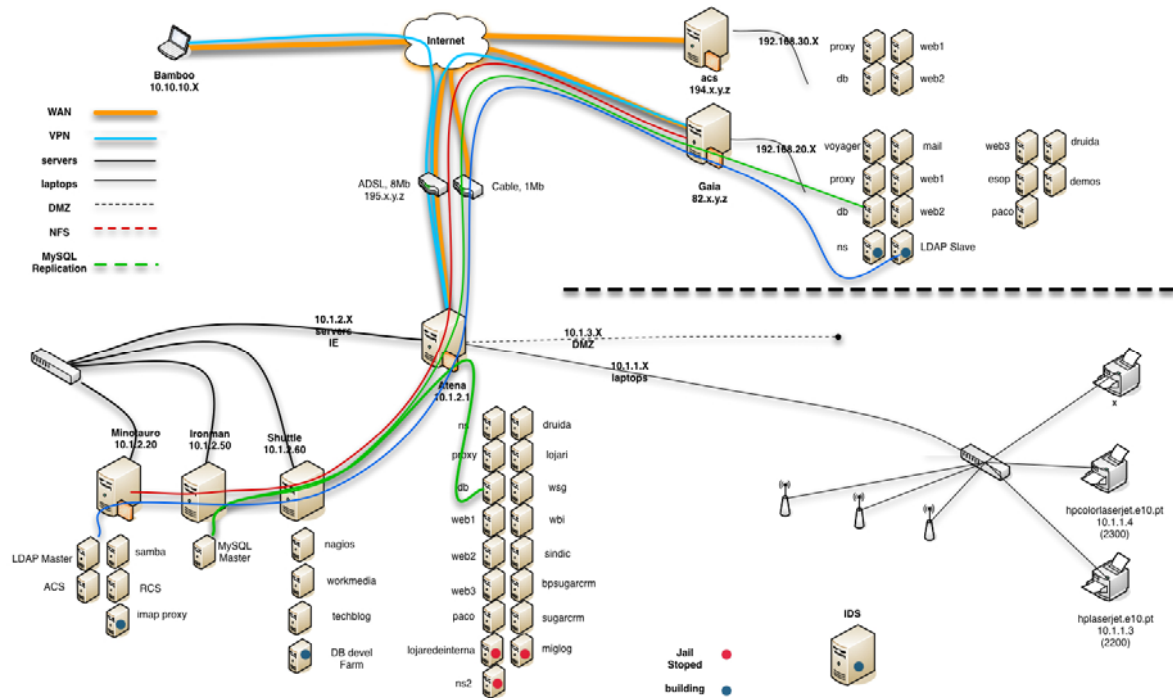
	Elaborado por: Francisco Alves Cabrita - 20060645 Luís Monteiro Castro - 20060658
--	---

Este sistema de virtualização, enclausura parte do sistema convidado; podendo os restantes recursos ser usados directamente; no entanto exige que todos os demais sistemas convidados executem o mesmo núcleo, o que trás algumas vantagens quer a nível de segurança como na facilidade para uma gestão centralizada e homogénea de todos os servidores virtuais. A título de exemplo; aquando a descoberta de uma falha de segurança ao nível do núcleo do sistema operativo, podemos proceder rapidamente à actualização de apenas um sistema operativo, de apenas um núcleo que é o do sistema anfitrião. Todos os sistemas virtuais herdam este núcleo, como tal passam automaticamente a actuar com as respectivas actualizações. De notar que o mesmo não acontece se uma dada falha, actualização ou outra configuração tiver ao nível *Userland*, ou seja, ou nível aplicacional. Uma vez mais, a título de exemplo vamos supor que é uma falha de segurança no serviço OpenSSH; neste momento vamos ter de proceder rapidamente à actualização deste software não só em todas as instâncias virtuais mas também no próprio anfitrião, caso se verifique claro está, que este serviço está presente em cada um dos sistemas.

Uma outra mais valia da utilização deste tipo de enclausuramento, é a possibilidade de reutilizar aplicações e configurações de sistema. Porque está-se presente de um sistema bastante homogéneo, pode-se construir (compilar) aplicações de forma centralizada e partilhar apenas os binários dessas aplicações para cada uma das máquinas virtuais; além de que é possível mesmo em alguns casos, partilhar uma única fonte do binário por via de mecanismos de *nullfs*; ou seja; partilhar *mountpoint* do sistema anfitrião em modo leitura para dentro de várias instâncias, servidores virtuais. Com este método é possível compilar um Perl standard e partilhar esta linguagem de *scripting* muito utilizada em tarefas de administração de sistemas. A gestão dos módulos desta (CPAN) pode ser feita quer no sistema anfitrião, quer salvo raras excepções, no servidor virtual. A vantagem é sábia; há uma única preocupação com a gestão das versões e esta é feita no anfitrião por um administrador de sistemas de confiança.

	Elaborado por: Francisco Alves Cabrita - 20060645 Luís Monteiro Castro - 20060658
--	---

Exemplo de um cenário real onde as FreeBSD jails são utilizadas em toda a infra-estrutura tecnologica de uma pme portuguesa a operar em Lisboa.



A rede é segmentada. Vlan para Servidores, Desktops instalados na rede local, onde estão agregados os colaboradores, DMZ onde ficam os roaming users que chegam por vpn e por fim uma outra DMZ fora das instalações onde são mantidos alguns protótipos para demonstração ao cliente; assim como servidores de email ou réplicas slave de servidores de base de dados MySQL, OpenLDAP outros.

De notar que no diagrama, todos os pequenos icons, servidores são de facto FreeBSD Jails.

5.5 - Implicações do sistema de virtualização utilizado

A escolha da solução Jails da FreeBSD, implica que todos os “sub-sistemas” ou seja, todas as máquinas virtuais criadas, sejam sobre a mesma plataforma e por consequência que as aplicações em si executadas, sejam obviamente desenhadas para o sistema em uso, ou compatível com sistemas POSIX. Caso se verifique uma dependência anormal de uma dada aplicação com o mundo do Linux, pode-se invocar um módulo de compatibilidade binária com Linux, isto dentro de FreeBSD, dentro de uma Jail.

Hoje já é possível alocar mais ou menos capacidade de processamento a uma dada Jail. Esta é uma nova funcionalidade que durante muito tempo foi um dos calcanhares de Aquiles das FreeBSD Jails. Tanto as Solaris Zones como OpenVZ já tinham resolvido esta questão há muito, no entanto hoje isto está ultrapassado e podemos restringir tempo de utilização de CPU numa dada Jail em detrimento de uma outra que por razões várias, necessita de mais processamento, recursos. De notar que o total de recursos existente é igual ao existente no anfitrião de forma nativa. Não podemos requisitar mais do que o existente.

Para o efeito do desenvolvimento deste protótipo foi escolhido a última versão do sistema operativo FreeBSD 8.1-RELEASE por razões tais como, o suporte destas configurações de *cpuset* entre outras.

	Elaborado por: Francisco Alves Cabrita - 20060645 Luís Monteiro Castro - 20060658
--	---

6 - OBJECTIVOS

6.1 - Solução proposta

Actualmente, a forma mais rápida para se criar uma FreeBSD Jail é recorrendo a uma linha de comandos e executar o seguinte conjunto de comandos, isto numa demonstração simplificada:

```
$ifconfig NOME_INTERFACE inet alias IP_DA_JAIL netmask
0xffffffff
$ mkdir /usr/jails ; mkdir /usr/jails/NOME_DA_JAIL
$ cd /usr/src
$ make -j4 world DESTDIR=/usr/jails/NOME_DA_JAIL
$ cd etc
$ make distribution DESTDIR=/usr/jails/NOME_DA_JAIL
$ cd /usr/jails/NOME_DA_JAIL ; mkdir tmp/BUILD
$ touch etc/fstab ; echo "hostname=\"NOME\" " >> etc/rc.conf
$ echo "WRKDIRPREFIX=/tmp/BUILD" >> etc/make.conf
$ echo "sshd_enable=YES" >> etc/rc.conf
$ echo "ListenAddress IP_DA_JAIL" >> etc/ssh/sshd_config
```

Não é uma tarefa complexa no entanto pode ser algo repetitiva e sujeita a erros humanos. É também pouco amigável ao utilizador e abre ainda portas a situações críticas que devem ser ocultadas de administradores menos experientes, protegendo assim a integridade de um dado servidor em particular e de toda uma infra-estrutura em geral.

A solução proposta recai sobre o desenvolvimento de um protótipo web para a gestão de servidores virtuais, “*user-friendly*” para uma rápida, simples administração e eficiente gestão de infra-estruturas de ambientes virtuais.

A aplicação web assenta sobre um *web server* e passa instruções a uma camada de *middleware* sobre o que executar no sistema operativo. É este elemento de middleware que de facto executa directamente no sistema operativo FreeBSD as instruções construídas pela *interface web*.

A componente a desenvolver de raiz é o módulo da aplicação web, todos os restantes módulos serão integrados com este.

	Elaborado por: Francisco Alves Cabrita - 20060645 Luís Monteiro Castro - 20060658
--	---

Para o acesso ao interface web é necessário proceder à autenticação, só depois de correctamente identificados é que é possível ao administrador efectuar operações tais como listar, criar ou eliminar servidores virtuais.

Este cenário é certamente mais agradável, intuitivo e seguro usar.

A forma modular como as componentes estão interligadas, lança caminhos para que se possa integrar outros *middlewares* ou mesmo desenvolver funcionalidades adicionais para colmatar requisitos que sem ser críticos são interessantes de ter hoje em dia, tais como a migração, agendamento de criação de sistemas virtuais automaticamente, definição de bandwidth ou utilização de cpu/ram por cada jail ou grupo de jails. O protótipo pertence ser algo modular e extensões adicionais são de fácil integração, bastando apenas carregar o novo módulo no arranque da aplicação e actualizar o interface web, formulários etc caso seja necessário.

Será visto adiante alguns *wire-frames* do protótipo proposto.

6.2 - Requisitos do Protótipo

Requisitos Nucleares

- Capacidade de um utilizador/administrador efectuar autenticação;
- Operações de CRUD sobre os sistemas virtuais;
 - ☐ *Create* - Criar jails;
 - ☐ *Read* - Ler jail status;
 - ☐ *Update* - Actualizar configurações de uma dada jail;
 - ☐ *Delete* - Apagar jail;
- *Start*;
- *Stop*.

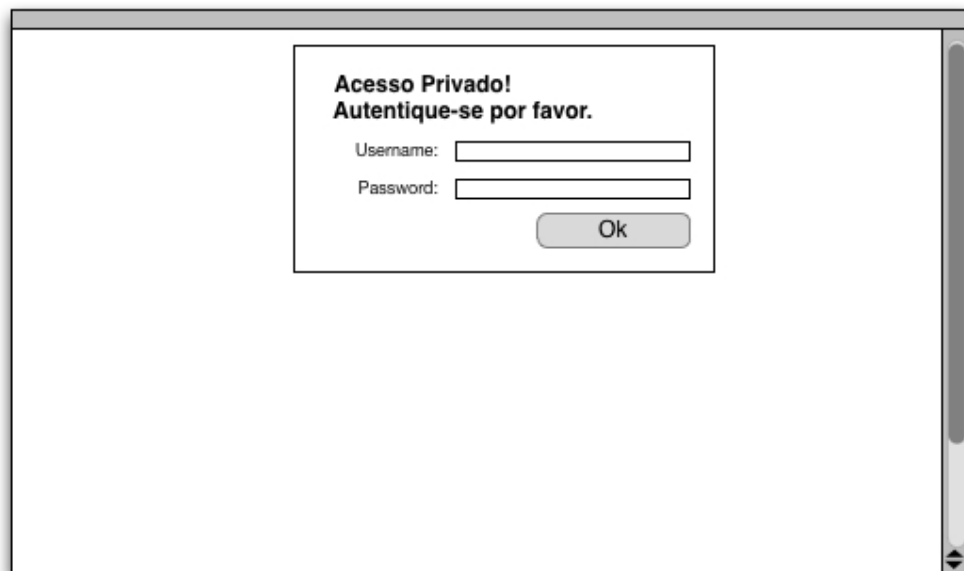
Requisitos para o *Interface Web*

- Fácil utilização;
- Seguro;
- Rápido e que não fique locked aquando a criação de uma jail;
- *Birds Eye* com Listagem servidores.

	Elaborado por: Francisco Alves Cabrita - 20060645 Luís Monteiro Castro - 20060658
--	---

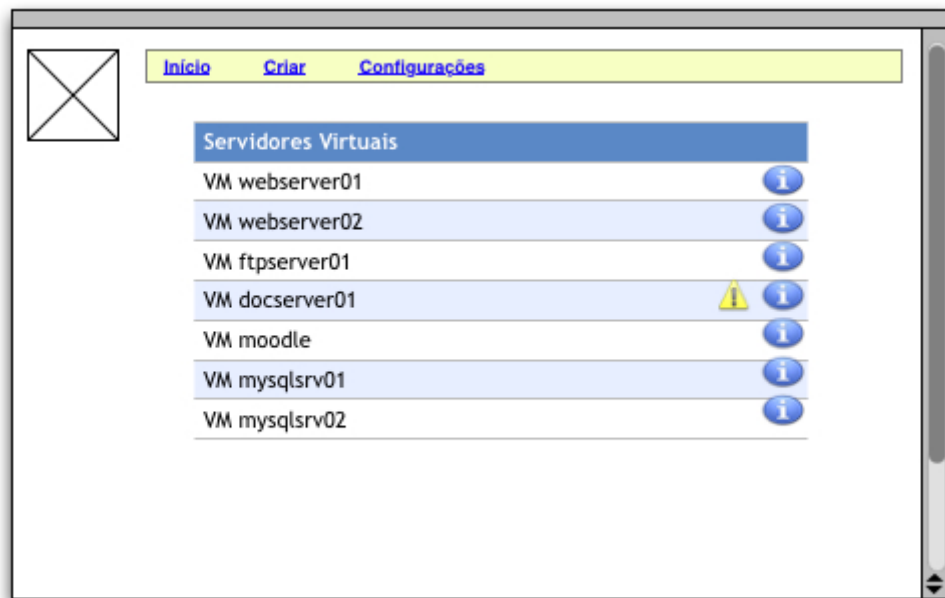
6.3 - Wireframes modelo

Ecrã de autenticação



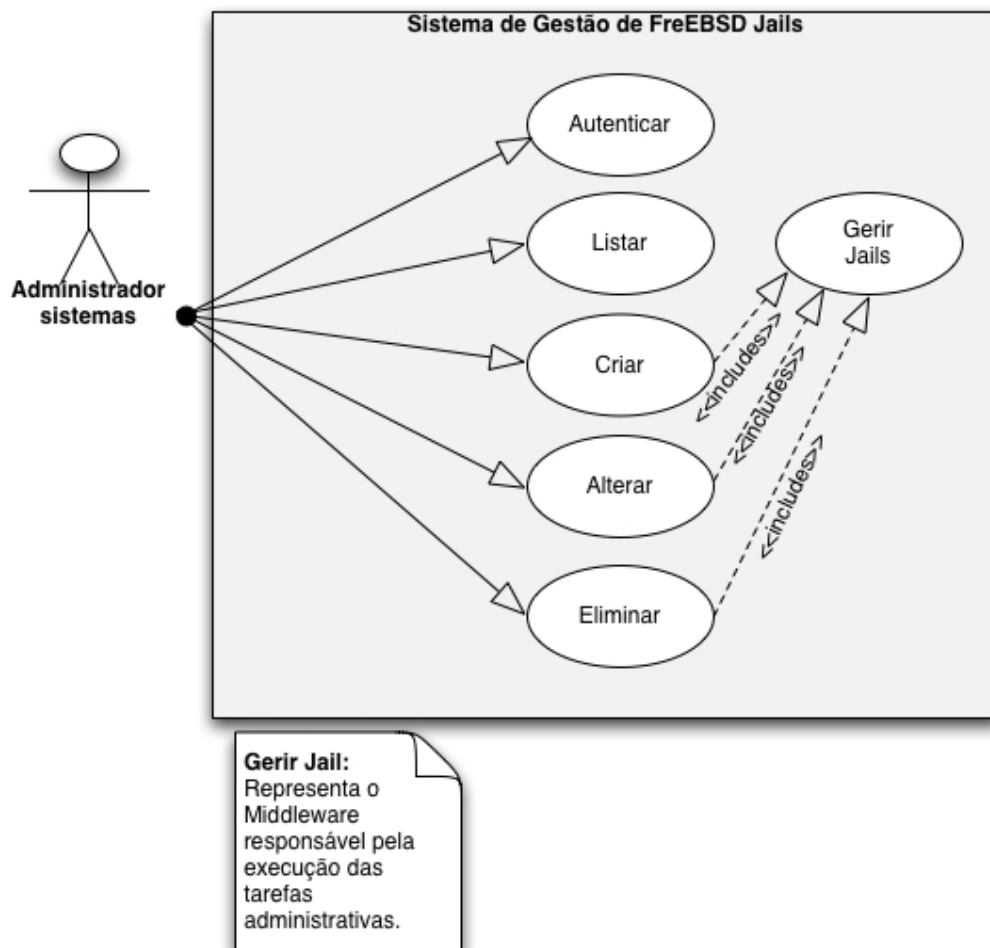
A wireframe of an authentication screen. It features a central dialog box with a title bar. Inside the dialog, the text "Acesso Privado!" is followed by "Autentique-se por favor." Below this, there are two input fields: "Username:" and "Password:". To the right of the "Password:" field is a button labeled "Ok". The dialog box is set against a larger window background that includes a vertical scrollbar on the right side.

Ecrã de Inicial



7 - MODELO CONCEPTUAL

7.1 - Use Case



No caso de uso, os verbos representam as acções que o utilizador, "Administrador de Sistemas" pode executar. Todas estas acções são comunicadas ao sistema operativo por via do *middleware* aqui representado pela modulo "Gerir Jails". Estas operações só podem ser efectuada após a válida autenticação do administrador.

8 - IMPLEMENTAÇÃO

A implementação do protótipo foi levada a cabo de forma bastante segmentada. Tratando-se de uma aplicação web desenvolvida por não programadores, todo o processo foi constituído por pequenas iterações de desenvolvimento.

As componentes base são o sistema operativo e o ambiente de desenvolvimento. Depois disto foram implementados mecanismos de controlo de versões, “*Git*” no caso. Foi também instalado um wiki e um *ticket system* “*Redmine*” para efectuar a documentação do projecto e registar problemas detectados na fase de desenvolvimento a serem resolvidos.

8.1 - Metodologia de desenvolvimento

No projecto foram seguidas algumas *guide lines* do que é o movimento de desenvolvimento ágil. Com isto, foi possível restringir e focar no objectivo proposto.

- Com pequenas iterações, garantiram-se entregas de softwares funcionais;
- Cooperação constante entre pessoas as várias pessoas que integram o projecto;
- Projectos surgem através de indivíduos motivados, entre os quais existe relação de confiança;
- Simplicidade;
- Rápida adaptação às mudanças;
- Indivíduos e iterações mais do que processos e ferramentas;

	Elaborado por: Francisco Alves Cabrita - 20060645 Luís Monteiro Castro - 20060658
--	---

8.1.1 - Fases do Projecto

A metodologia de gestão de projecto desenvolvida e utilizada pela equipa, nasceu da necessidade de ter uma metodologia de gestão adaptada às necessidades e constrangimentos quer do projecto em causa quer da equipa. O recurso a tecnologias open source faz mover também alguns dos elementos da gestão de projecto como a disponibilidade de software de terceiros e a resolução de problemas inerentes a estes ambientes.

- Proposta
- *Setup* inicial do Projecto
- Planeamento
 - ☐ do documento
 - ☐ do desenvolvimento
- Implementação
- Fecho do Projecto

Fase 1 - Proposta

Esta fase permite reduzir o erro da proposta quer a nível de âmbito quer a nível de custos em tempo e recursos.

A mais correcta elaboração da proposta mais focado e orientado ao objectivo final se está.

Fase 2 - Setup inicial do Projecto

Esta fase é de preparação e arranque do projecto. Aqui são recolhidos todos os inputs para que se inicie a instalação de todos os elementos necessários quer a nível de elaboração de documentação, mecanismos de controlo e revisão. Contactos, prazos e demais informações para o bom funcionamento do projecto são revistos.

Fase 3 – Planeamento

Delimitar o âmbito do projecto e fazer as alterações necessárias para que seja dada uma resposta o mais congruente possível. Aqui, é sabido precisamente qual é o caminho a percorrer e qual o objectivo final.

	Elaborado por: Francisco Alves Cabrita - 20060645 Luís Monteiro Castro - 20060658
--	---

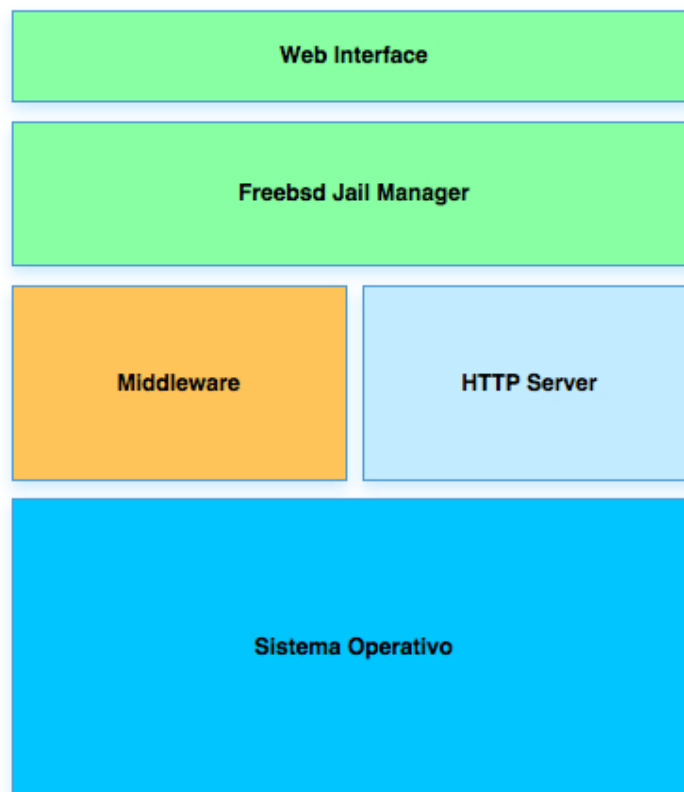
Fase 4 - Implementação

Suporte à implementação do projecto e desenvolvimento. Alimentado pelas fases anteriores e com a aplicação meticulosa das tarefas, os riscos são calculados e minimizados. Os interesses são protegidos.

Fase 5 - Fecho

O projecto é testado e verificado se tudo está como é esperado, é feita a transição final dos elementos descritivos da solução e colocado em produção caso seja o caso.

8.2 - Arquitectura do Sistema



	Elaborado por: Francisco Alves Cabrita - 20060645 Luís Monteiro Castro - 20060658
--	---

8.3 - Instalação e Configuração

Para a instalação do Sistema Operativo FreeBSD, os requisitos são apenas:

- Conhecimentos básicos de UNIX;
- Conhecimentos básicos de Redes IPv4.

O *layout* do sistema de ficheiros deve contemplar uma área para alojar os sistemas virtuais, o qual deve estar separado das demais partições.

A nível de *networking* não há até ao momento detalhes que possam por em causa a implementação da solução.

No capítulo seguinte é feita a instalação dos componentes que interagem directamente com a solução desenvolvida.

Ruby, instalação base

Como administrador do sistema executar:

```
% pkg_add -r ruby
```

Instalação de bibliotecas base:

```
% pkg_add -r ruby-gems
```

RVM, Ruby Version Manager

```
% gem install rvm
```


O RVM permite que o utilizador possa instalar e desenvolver em versões distintas de Ruby. É uma excelente escolha para que se possa distinguir zonas de desenvolvimento, pré-produção e produção. É visto no passo seguinte como se instala no ambiente o RVM e a sua utilização prática.

Uma outra grande vantagem é que com RVM podemos isolar a instalação do Ruby do sistema operativo das demais instalações que integram os desenvolvimentos aplicativos. A versão instalada em primeiro lugar fica a ser gerida pelo sistema de gestão de pacotes do sistema operativo "*Ports Systems*" no que diz respeito às outras versões, estas são geridas pelo RVM em si. Qualquer problema numa das versões instaladas a posteriori não deve afectar o normal funcionamento do Ruby instalado pelos *ports*.

Neste passo, com um utilizador do sistema que não o administrador, o qual deve ter a Bash como shell pré-definida. Assim procede-se a:

Instalar no ambiente do utilizador o RVM:

Cada RVM fica agregado à *home directory* de cada utilizador.

```
$ rvm-install
```

Carregar os *paths* de ambiente para podermos executar o comando *rvm* sem complicações:

```
$ rehash
```

A partir deste momento procedemos à instalação da versão do Ruby que ficará embebida e "anexa" ao projecto.

```
$ rvm install 1.9.2
```

```
$ rvm use 1.9.2
```

Para verificar que versões ficaram instaladas, executar:

```
$ rvm list
```

Instalação dos plugins:

```
$ gem install sinatra thin warden bundler couchrest pony
```

EzJail

Instalação

Uma vez mais, como administrador do sistema, executar:

```
% pkg_add -r ezjail
```

Configuração

```
% echo "ezjail_enable=\"YES\"" >> /etc/rc.conf
% cd /usr/local/etc
% cp ezjail.conf.sample ezjail.conf
```

Construir a primeira Jail

```
% echo "ifconfig_lo0_alias=\"inet 127.0.0.2 netmask 0xffffffff\"" >>
/etc/rc.conf
% ezjail-admin create -f jailprojectofinal
jailprojectofinal.exemplo.pt 127.0.0.2
/usr/jails/jailprojectofinal/COPYRIGHT
/usr/jails/jailprojectofinal/basejail
/usr/jails/jailprojectofinal/bin
...
Note: Shell scripts installed, flavourizing on jail first startup.
Warning: Some services already seem to be listening on all IP,
(including 127.0.0.2) This may cause some confusion, here they are:
root  sshd  1010  4      tcp4  *:22  *.*
root  syslogd  816    udp4  *:514 *.*
[root@jailprojectofinal ~]# jls
```

Inicializar a primeira Jail

```
% /usr/local/etc/rc.d/ezjail.sh start
ezjailConfiguring jails:.
Starting Jails: jailprojectofinal.exemplo.pt
% jls
  JID  IP Address  Hostname      Path
  ---  -
    1   127.0.0.2   jailprojectofinal /usr/jail/jailprojectofinal
```

Aceder à jail por SSH

```
% ssh admin@127.0.0.2
```

9 – CONCLUSÃO E TRABALHO FUTURO

Este trabalho foi elaborado como uma prova de conceito para demonstrar que gerir servidores virtuais é um assunto importante, que deve ser fruto de uma administração cuidada e pensada. O protótipo foi desenvolvido recorrendo a tecnologias menos comuns e em alguns casos menos avançadas mas que por questões várias, são ou podem ser as mais viáveis mediante um contexto particular.

Neste trabalho foi mantida uma relação muito próxima entre a gestão dos sistemas de informação e as camadas mais técnicas da engenharia de sistemas. É importante saber alocar recursos físicos ou virtuais no tempo, como se de projectos se tratassem para que a sua gestão possa ser correctamente contabilizada, mantida e acompanhada.

Há um mundo de sistemas operativos, de sabores e distribuições; assim como há um mundo de linguagens de programação e frameworks que disponibilizam camadas de abstracção de inúmeros níveis de complexidade e de integração. Neste trabalho foi possível tocar vários instrumentos para além daqueles sujeitos a ensino nas disciplinas do curso de Informática de Gestão; isto prova em parte a capacidade cultivada durante o curso para aquando a exposição de um problema, ser levada a cabo a sua análise, exploração, integração de temas e resolução do mesmo.

Fica revisto no protótipo um exemplo de uma aplicação que pode sofrer sucessivas alterações, módulos adicionais, extendendo a sua capacidade para várias utilizações futuras; pois tratando-se também de software livre, fica desde já o convite à reutilização do mesmo por parte de futuros alunos da nossa Universidade que pretendam utilizar o mesmo em laboratórios experimentais.

	Elaborado por: Francisco Alves Cabrita - 20060645 Luís Monteiro Castro - 20060658
--	---

10 – BIBLIOGRAFIA

Golden, Bernard; *Virtualization for Dummies*; Wiley Publishing, Inc; 2008
Membrey, Peter; Verhoeven, Tim; Angenendt, Ralph; *The Definitive Guide to CentOS*; Apress; 2010

S. Crawford, Luke; Takemura, Chris. *The Book of Xen: A Practical Guide for the System Administrator*; No Starch Press, 2008

W. Lucas; *Absolute FreeBSD: The Complete Guide to FreeBSD, 2nd Edition*; No Starch Press, 2008

Cabrita, Francisco. "Jails: Como encarcerar Userlands em FreeBSD"

<http://sufixo.com/articles/jails.pdf>

Cabrita, Francisco, "Rápida Introdução às FreeBSD Jails"

http://sufixo.com/articles/jails_barcamp06.pdf

Ezjail, Jail administration

<http://erdgeist.org/arts/software/ezjail/>

FreeBSD Handbook, Jails chapter

<http://www.freebsd.org/doc/handbook/jails.html>

FreeBSD Handbook, Jail Sub-System

<http://www.freebsd.org/doc/en/books/arch-handbook/jail.html>

FreeBSD, Wiki

<http://pt.wikipedia.org/wiki/FreeBSD>

Kernel Based Virtual Machine

http://www.linux-kvm.org/page/Main_Page

Linux Journal, KVM Virtualization

<http://www.linuxjournal.com/article/9764>

Ruby, A Programmer best friend

<http://www.ruby-lang.org/pt/>

Sinatra, Create web applications on ruby

<http://www.sinatrarb.com/>

Solaris Zones Introduction

http://docs.sun.com/app/docs/doc/820-2978/zones.intro-1?l=pt_BR&a=view

What is KVM?

<http://okvm.sourceforge.net/virtalkvm.html>

	Elaborado por: Francisco Alves Cabrita - 20060645 Luís Monteiro Castro - 20060658
--	---

11- ANEXOS

11.1 - Listagem de Acrónimos e Definições

API - Application Programming Interface
Bash - Bourne-again shell
BSD - Berkley Software Distribution
CEO - Chief Executive Officer
Cloud Computing - Conceito de computação na nuvem refere-se à utilização de servidores e computadores interligados pela internet.
CPU - Central Processos Unit
CRM - Customer Relationship Management
Daemon - Aplicação que tipicamente é executada em background
DMZ - Delimitarized Zone
DNS - Domain Name System
Git - Global Information Tracker
GPL – General Public License
GUI - Graphic User Interface
HTTP - Hypertext Transfer Protocol
Jail - Recurso de criação de múltiplos ambientes virtuais nativo do FreeBSD
Middleware - designação genérica utilizada para referir aos sistemas de software que se executam entre as aplicações e os sistemas operativos.
MVC - Model Viewer Controler
NTP - Network Time Protocol
Perl - Linguagem de programação
PHP - Hypertext PreProcessor
POSIX - Portable Operating System Interface for Unix
Python - Linguagem de programação Object-Oriented
Ruby - Linguagem de programação interpretada
RVM - Ruby Version Manager
SGDB - Sistema Gestão de Base de Dados
SQL - Structured Query Language
Squid - Servidor de proxy
SWIG - Simplified Wrapper and Interface Generator
UNIX - Sistema Operativo multi-tarefa criado por Ken Thompson
Vlan - Rede local virtual
VPS - Virtual Private Server
Webserver - Aplicação que entrega dados, como páginas através de http na Internet.

	Elaborado por: Francisco Alves Cabrita - 20060645 Luís Monteiro Castro - 20060658
--	---

Wiki - Termo utilizado para identificar um tipo específico de coleção de documentos.

Wireframes - Guia visual básico usado para sugerir a estrutura de um site e relacionamentos entre páginas

WUI - Web User Interface

XMPP - Extensible Messaging and Presence Protocol

	Elaborado por: Francisco Alves Cabrita - 20060645 Luís Monteiro Castro - 20060658
--	---