FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

FEUP

Integração de serviços sobre plataforma de comunicações multi-uso

Pedro Miguel Mendes de Oliveira

Tese submetida no âmbito do

Mestrado Integrado em Engenharia Electrotécnica e de Computadores

Major de Telecomunicações

Orientador: Professor Manuel Alberto Pereira Ricardo

Fevereiro de 2009

Resumo

Este trabalho incide sobre a integração do sistema IPBrick com uma solução *Customer Relationship Management* (CRM). Surge da necessidade de automatizar os registos das comunicações com os clientes, de modo a ter uma base de dados sempre actualizada com todos os contactos efectuados, permitindo uma maior eficiência dos recursos humanos com a eliminação de inserir esses mesmos contactos de forma manual e uma maior gestão dos seus clientes. A IPBrick é uma solução para gestão de redes, baseada em Linux, com uma administração simples e funcional com acesso via *web*, e possui vários métodos de implementação de comunicações (voz, fax, email, *instant messaging*), através de linhas analógicas , digitais (RDIS) ou puramente IP. Um CRM é um sistema integrado de gestão com foco no cliente, constituído por um conjunto de procedimentos/processos organizados e integrados num modelo de gestão de negócios. O SugarCRM, a solução CRM adoptada para esta integração, é bastante flexível e customizável.

O objectivo é efectuar o registo automático no SugarCRM da comunicação entre a empresa e o seu cliente, independentemente do método de comunicação utilizado da IPBrick. Este registo fica disponível na ficha do cliente no SugarCRM para futura utilização. Realiza-se o registo da comunicação recorrendo a um *Enterprise Service Bus* (ESB). Este barramento de mensagens transporta a mensagem da IPBrick com a informação do método e intervenientes da comunicação, para o SugarCRM. A adopção do barramento oferece uma maior flexibilidade na implementação dos sistemas IPBrick e o SugarCRM, permitindo que ambos os sistemas não necessitem de partilhar a mesma máquina ou a mesma rede local. Obtem-se assim um base de dados sempre actualizada com as comunicações efectuadas e os seus resultados, entre a empresa e o cliente, para futura gestão.

Abstract

This document focuses on integrating the operative system IPBrick with a Customer Relationship Management (CRM) solution. Arises from the need to automate the records of communications with costumes, to have a database always updated with all contacts made, allowing a greater efficiency of the human resources with the elimination of inserting those contacts manually and a greater management of their customers. IPBrick is a Linux-based network management solution with a simple and functional administration interface with access via web, and has several methods of implementing communications (voice, fax, email, instant messaging) via analog lines, digital (ISDN) or pure IP. CRM is an integrated management system focused on customer, consists of a set of procedures/processes organized and integrated into a management model of business. The SugarCRM, the CRM solution adopted for this integration, it's very flexible and customizable.

The aim is to register automatically in SugarCRM the communication between the company and his client, regardless of the method of communication used by IPBrick. This record will be available in the client file in SugarCRM for future use. The record of communication is held by using an Enterprise Service Bus (ESB). This bus carries the message from IPBrick, with the information of the method and intervenient of the communication, to SugarCRM. The adoption of the bus provides increased flexibility in the implementation of both systems, allowing the two systems the option of not sharing the same machine or the same local network. This yields a database always updated with the communications and is results, between the company and the customer for future management.

Agradecimentos

Começo por agradecer à minha família, em especial aos meus pais e irmãs pela sua dedicação, muita paciência e por acreditarem em mim.

À minha namorada pela compreensão, paciência e incondicional apoio que sempre demonstrou.

Um agradecimento muito especial ao meu orientador Prof. Manuel Alberto Pereira Ricardo pela sua disponibilidade e acompanhamento.

A todos os colaboradores da empresa iPortalMais, em especial ao Engenheiro Raúl Oliveira, pela sua disponibilidade e apoio para a realização deste projecto.

Aos meus amigos que de uma forma directa ou indirecta me apoiaram ao longo do trabalho.

A todos muito obrigado.

Pedro Oliveira

"Experiência não é o que acontece a um homem,
é aquilo que um homem faz com o que lhe acontece."

Aldous Huxley

Conteúdo

Lista de Figuras

Lista de Tabelas

Abreviaturas e Símbolos

API	*Application Programming Interface*
CRM	*Customer Relationship Management*
DHCP	*Dynamic Host Configuration Protocol*
DMZ	*DeMilitarized Zone*
DNS	*Dynamic Network Services*
EAI	*Enterprise Application Integration*
EDA	*Event-Driven Architecture*
FAX	*Facsimile*
FEUP	Faculdade de Engenharia da Universidade do Porto
FTP	*File Transfer Protocol*
GUI	*Graphical User Interface*
HTTP	*Hypertext Transfer Protocol*
IETF	*Internet Engineering Task Force*
IIS	*Internet Information Services*
IM	*Instant Messaging*
IMAP	*Internet Message Access Protocol*
JDBC	*Java Database Connectivity*
JMS	*Java Message Service*
LDAP	*Lightweight Directory Access Protocol*
MMS	*Multimedia Messaging Service*
PHP	*Hypertext Preprocessor*
PME	Pequenas e Médias Empresas
POJO	*Plain Old Java Objects*
SIP	*Session Initiation Protocol*
SMS	*Short Message Service*
SMTP	*Simple Mail Transfer Protocol*
SO	Sistema Operativo
SOA	*Service-Oriented Architecture*
SOAP	*Simple Object Access Protocol*
SSL	*Secure Sockets Layer*
TI	Tecnologias da Informação
UCoIP	*Unified Communications Over Internet Protocol*
UMO	*Universal Message Object*
VoIP	*Voice over Internet Protocol*
VPN	*Virtual Private Network*
WSDL	*Web Services Definition Language*
XML	*Extensible Markup Language*
XMPP	*Extensible Messaging and Presence Protocol*

Capítulo 1

Introdução

O presente projecto foi desenvolvido na empresa iPortalMais - Soluções de Engenharia para Internet e Redes, Lda. no âmbito de um protocolo estabelecido entre a empresa e a Faculdade de Engenharia da Universidade do Porto .

Os objectivos propostos consistiram na análise, especificação e desenvolvimento de uma plataforma de integração de módulos autónomos que integram o sistema operativo IPBrick, nomeadamente as soluções *Customer Relationship Management* (CRM) e *Unified Communications Over IP* (UCoIP) adoptadas.

Se, por um lado, a modularização crescente de sistemas de informação empresariais permite garantir a escalibilidade do produto IPBrick às necessidades do cliente, por outro, acarreta o desafio de apresentar uma interface uniforme e transparente para o utilizador, garantindo o fluxo de informação entre as diversas aplicações.

1.1 Contexto

Com um mercado cada vez mais globalizado a sobrevivência das empresas, principalmente das Pequenas e Médias Empresas (PME) torna-se difícil e complexo. A tecnologia muitas vezes tem custos que estas não conseguem suportar já que são mais vocacionadas para as grandes empresas. No entanto não podem deixar de ter recursos tecnológicos que lhes permitam a sobrevivência, sustentabilidade e durabilidade. O custo de mão-de-obra especializada nem sempre é uma possibilidade já que estes geralmente implicam um aumento de custos.

O *software* presente numa empresa, principalmente se especializado, representa uma boa percentagem nos custos. A utilização de *software open source* dá a possibilidade de reduzir estes custos mas geralmente está associada a uma difícil implementação já que a sua configuração nem sempre é fácil.

A IPBrick é desenvolvida sobre a tecnologia *open source* Linux, e possui uma interface de gestão funcional com acesso via *web*, que permite efectuar a sua configuração sem que o seu administrador possua conhecimentos de Linux ou mesmo de redes e serviços. É um sistema UCoIP, isto é, as comunicações implementadas pela IPBrick podem ser baseadas unicamente no endereço do utilizador para todos os tipos de comunicação. Oferece todos os serviços de um servidor de intranet (ex: *Lightweight Directory Access Protocol* (LDAP), correio electrónico (POP/IMAP), *Dynamic Host Configuration Protocol* (DHCP), *Dynamic Network Services* (DNS), de comunicações (ex: servidor *web*, servidor *File Transfer Protocol* (FTP), *Virtual Private Network* (VPN), telefonia Voz sobre IP (VoIP)) e segurança (ex: *firewall*, Anti-vírus, Anti-Spam). Estes são bons exemplos de serviços considerados vitais numa empresa nos dias de hoje conjugado com uma facilidade de configuração e instalação. Possui também um calendário/agenda partilhados e gestor de contactos, IPContactos, de maneira a ser possível gerir todos os projectos de um ponto central.

No departamento comercial de uma empresa ou de um *call-center*, um *Customer Relationship Management* (CRM) é uma mais valia. Entende-se CRM como sendo uma estratégia de negócio voltada ao entendimento e à antecipação das necessidades dos clientes actuais e potenciais de uma empresa. Do ponto de vista tecnológico, CRM envolve capturar dados do cliente ao longo de toda a empresa, consolidar todos os dados capturados interna e externamente numa base de dados central, analisar os dados consolidados, distribuir os resultados dessa análise aos vários pontos de contato com o cliente e usar essa informação ao interagir com o cliente através de qualquer ponto de contacto com a empresa. [1] Uma das actividades da Gestão do Relacionamento com o Cliente (tradução livre para português de CRM) implica registar os contactos por si realizados, de forma centralizada. Os registos não dependem do canal de comunicação que o cliente utilizou (voz, fax, email, *instant messaging*, SMS, MMS, etc) e servem para que se tenham informações úteis e catalogáveis sobre os clientes. Qualquer informação relevante para as tomadas de decisões podem ser registadas e analisadas periodicamente de forma a produzir relatórios de gestão. [2]

Neste contexto o SugarCRM é uma boa solução. É *open source* e de fácil manuseamento. É um produto de CRM corporativo com módulos para gerenciamento de empresas e divisões, contactos, *prospects*, oportunidades, ocorrências, campanhas de *marketing*, projectos, documentos, agenda e histórico. [3] Caracteriza-se por ter uma forte componente em gestão comercial, gestão de *marketing*, gestão de clientes e *reports*. Tudo isto é acompanhado por um grande suporte a nível de processos administrativos presentes na aplicação.

Uma empresa com uma IPBrick consegue centralizar e uniformizar os seus serviços e comunicações. Com o recurso a um CRM esta gere de uma maneira mais eficaz os contactos com os clientes conseguindo manter uma base de dados actualizada. A troca de informação com o cliente é registada de modo manual, ocupando assim tempo e o recurso humano. Surge então a necessidade de automatizar o registo de maneira a aumentar o desempenho. Como todas as comunicações têm um ponto em comum, a IPBrick, a integração aparece de forma natural.

1.2 Objectivos

O objectivo deste trabalho é fazer o registo automático no SugarCRM da comunicação da empresa com o seu cliente, independentemente do método de comunicação utilizado através da IPBrick. Este registo ficará disponível na ficha do cliente no SugarCRM para futura utilização.

Este trabalho terá duas fases. Numa primeira fase é que a instalação e a implementação do SugarCRM terá que ser rápida. O administrador de sistemas não deverá perder tempo a copiar informação dos utilizadores e contactos da IPBrick. Na segunda fase é possibilitar a utilização dos registos dos contactos por outra aplicação. Para que assim seja, a interacção entre a IPBrick e o SugarCRM não deverá ser directa, ou seja, a IPBrick não escreverá directamente na base de dados do SugarCRM. Como os serviços da intranet podem estar em vários servidores "espalhados" pela intranet, terá que se arranjar uma forma de comunicação entre os sistemas.

1.3 Estrutura da Dissertação

Para além da introdução, esta dissertação contém mais 5 capítulos. No capítulo 2 são apresentadas as tecnologias usadas. No capítulo 3 a teoria necessária para o desenvolvimento do projecto é apresentada. No capítulo 4 apresentam-se as especificações. No capítulo 5 mostra-se o sistema implementado e possíveis utilizações. No capítulo 6 são apresentadas as conclusões do trabalho e perspectivados futuros desenvolvimentos.

Capítulo 2

Sistemas Open Source

No presente capítulo pretende-se fazer um pequeno *briefing* sobre a origem das plataformas tecnológicas que irão ser usadas no presente trabalho.

Justifica-se a razão de ser da IPBrick mas antes, apresenta-se o sistema que serve de base à IPBrick , o Linux.

O número de aplicações *Open Source* e a sua utilização torna-se cada vez mais comum. Porquê?

A facilidade com que estas podem ser testadas e modificadas sem pagar nada é uma grande vantagem. Evoluem mais rápido já que existem mais utilizadores e programadores a desenvolver, com a possibilidade de satisfazer necessidades individuais de clientes, sem estes terem que esperar que um programa comercial numa *release* futura tenha a funcionalidade desejada. De um modo geral também têm melhor performance e são mais seguras. Isto porque não há demasiados interesses comerciais que impliquem fazer *upgrades* de *hardware* e *software* para manter as coisas a funcionar e como é um *software* "aberto", é possível saber o que está a executar e não precisa de se preocupar se está a recolher informação privada e a enviar para algum lado.

A definição do *Open Source* foi criada pela Open Source Iniciative (OSI) a partir do texto original da Debian Free Software Guidelines (DFSG) e determina que um programa de código aberto deve garantir:

1. Distribuição livre

   A licença não deve restringir de nenhuma maneira a venda ou distribuição do programa gratuitamente, como componente de outro programa ou não.

2. Código fonte

   O programa deve incluir o seu código fonte e deve permitir a sua distribuição também na forma compilada. Se o programa não for distribuído com o seu código fonte, deve haver

algum meio de se obter o mesmo seja via rede ou com custo apenas de reprodução. O código deve ser legível e inteligível para qualquer programador.

3. Trabalhos Derivados

   A licença deve permitir modificações e trabalhos derivados, e deve permitir que eles sejam distribuídos sobre os mesmos termos da licença original.

4. Integridade do autor do código fonte

   A licença pode restringir o código fonte de ser distribuído em uma forma modificada apenas se a licença permitir a distribuição de arquivos *patch*(de actualização) com o código fonte para o propósito de modificar o programa no momento de sua construção. A licença deve explicitamente permitir a distribuição do programa construído a partir do código fonte modificado. Contudo, a licença pode ainda requerer que programas derivados tenham um nome ou número de versão diferentes do programa original.

5. Não discriminação contra pessoas ou grupos

   A licença não pode ser discriminatória contra qualquer pessoa ou grupo de pessoas.

6. Não discriminação contra áreas de actuação

   A licença não deve restringir qualquer pessoa de usar o programa em um ramo específico de actuação. Por exemplo, ela não deve proibir que o programa seja usado em uma empresa ou de ser usado para pesquisa genética.

7. Distribuição da Licença

   Os direitos associados ao programa devem ser aplicáveis para todos aqueles cujo programa é redistribuído, sem a necessidade da execução de uma licença adicional para estas partes.

8. Licença não específica a um produto

   Os direitos associados ao programa não devem depender que o programa seja parte de uma distribuição específica de programas. Se o programa é extraído desta distribuição e usado ou distribuído dentro dos termos da licença do programa, todas as partes para quem o programa é redistribuído devem ter os mesmos direitos que aqueles que são garantidos em conjunção com a distribuição de programas original.

9. Licença não restrinja outros programas

   A licença não pode colocar restrições em outros programas que são distribuídos juntos com o programa licenciado. Isto é, a licença não pode especificar que todos os programas distribuídos na mesma mídia de armazenamento sejam programas de código aberto.

10. Licença neutra em relação à tecnologia

    Nenhuma cláusula da licença pode estabelecer uma tecnologia individual, estilo ou interface a ser aplicada no programa.

Esta definição foi retirada da Wikipédia. [4]

O sistema Linux é um dos melhores e dos mais famosos exemplos que se pode apresentar. Outro, sem dúvida, é a aplicação *Customer Relationship Management* usada neste trabalho, o SugarCRM.

2.1 Linux

Linux é o termo geralmente usado para designar qualquer sistema operativo que utilize o kernel compatível com o Unix.

Foi originalmente escrito por Linus Torvalds do Departamento de Ciência da Computação da Universidade de Helsínquia, Finlândia, com a ajuda de vários programadores voluntários através da Usenet. Começou o seu desenvolvimento em 1991 como um projecto particular, inspirado pelo seu interesse no Minix, um pequeno sistema UNIX desenvolvido por Andrew S. Tanenbaum. Ele limitou-se a criar, nas suas próprias palavras, "um Minix melhor que o Minix".

No dia 5 de Outubro de 1991 anunciou a primeira versão "oficial"do kernel Linux, versão 0.02 e trabalhou arduamente até 1994, altura em que a versão 1.0 do kernel do Linux foi lançada. A versão actual (estável) é a 2.6 sendo que o desenvolvimento de novas versões não pára, e o seu código fonte está disponível sob licença GPL (*General Public License*) para qualquer pessoa utilizar, estudar, modificar e distribuir de acordo com os termos da licença.

Inicialmente desenvolvido e utilizado por grupos de entusiastas em computadores pessoais, o sistema Linux passou a ter a colaboração de grandes empresas, como a IBM, Sun Microsystems, Hewlett-Packard, Novell e a Canonical. [5]

Figura 2.1: Tux, o pinguim, mascote do Linux

2.1.1 Arquitectura

As funções do kernel (agendamento de processos, gestão de memória, operações de entrada e saída, acesso ao sistema de arquivos) são executadas no espaço do kernel, compiladas estaticamente ou usando módulos carregáveis. Herdou da linhagem UNIX a capacidade de multitarefa, multiprocessamento, uso de encadeamentos no kernel (kernel *threading*), preempção do kernel, suporte a aplicações multi-encadeadas (*multithreaded application support*, por meio de processos leves - LWP), sistema de arquivos baseado em *Virtual File System* (VFS). (BOVET e CESATI, 2005, cap.1)

Todo o acesso ao núcleo por uma aplicação deve ser efectuado por meio de chamadas de sistema. Uma chamada de sistema é uma função que permite a criação de processos, solicitação de memória, operação com arquivos, etc.

Já a comunicação do núcleo com o *hardware* ocorre, basicamente, de dois modos: (i) o núcleo reconhece e gere o *hardware* usando *drivers* de dispositivos; (ii) o *hardware* comunica com o núcleo por meio de interrupções.

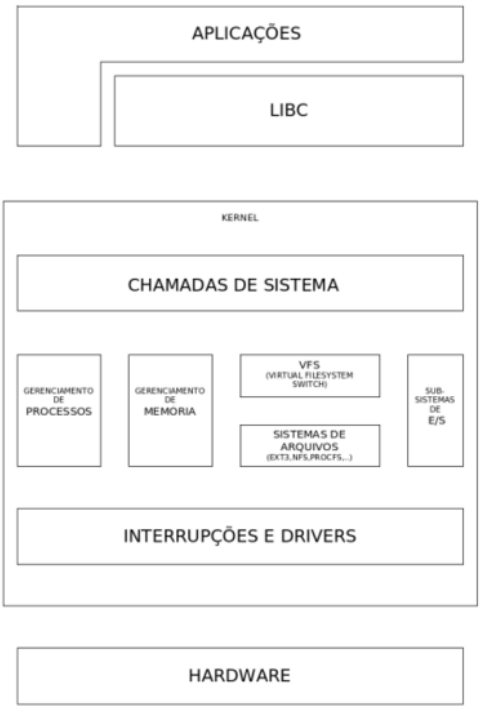

Figura 2.2: Um diagrama simplificado do núcleo Linux, mostrando seus componentes principais e as camadas que o cercam [6]

Os *drivers* de dispositivos e extensões do núcleo correm tipicamente no espaço do kernel, juntamente com o restante do núcleo, com acesso total ao *hardware*. Diferentemente dos núcleos monolíticos tradicionais, os *drivers* de dispositivos podem ser configurados como módulos (LKM

- *loadable kernel modules*), que são bibliotecas compiladas fora do kernel propriamente dito, o que permite que sejam carregados e descarregados enquanto o sistema corre.

Também podem ser interrompidos (*preempted*) sob certas condições. Esta característica foi adicionada para lidar com interrupções de *hardware* correctamente e para melhorar o suporte ao multiprocessamento. [6]

2.1.2 Licenciamento

Inicialmente, Torvalds lançou o Linux sob uma licença que proibia qualquer uso comercial. Isso foi mudado de imediato para a Licença Pública Geral GNU. Essa licença permite a distribuição e mesmo a venda de versões possivelmente modificadas do Linux mas requer que todas as cópias sejam lançadas dentro da mesma licença e acompanhadas do código fonte.

Apesar de alguns dos programadores que contribuem para o kernel permitirem que o seu código seja licenciado com GPL versão 2 ou posterior, grande parte do código (incluído as contribuições de Torvalds) menciona apenas a GPL versão 2. Isto faz com que o *kernel* como um todo esteja sob a versão 2 exclusivamente, não sendo de prever a adopção da nova GPLv3.

2.1.3 Distribuições

Actualmente, um Sistema Operativo Linux ou GNU/Linux completo (uma "Lista de distribuições de Linux ou GNU/Linux") é uma colecção de *software* livre (e por vezes não-livre) criados por indivíduos, grupos e organizações de todo o mundo, incluindo o núcleo Linux. Companhias como a Red Hat, a SuSE, a Mandriva (união da Mandrake com a Conectiva) e a Canonical (desenvolvedora do Ubuntu Linux), bem como projectos de comunidades como o Debian ou o Gentoo, compilam o *software* e fornecem um sistema completo, pronto para instalação e uso.

As distribuições do Linux ou GNU/Linux começaram a receber uma popularidade limitada desde a segunda metade dos anos 90, como uma alternativa livre para os sistemas operativos Microsoft Windows e Mac OS, principalmente por parte de pessoas acostumadas com o Unix na escola e no trabalho. O sistema tornou-se popular no mercado de *Desktops* e servidores, principalmente para a internet e servidores de base de dados.

No decorrer do tempo, várias distribuições surgiram e desapareceram, cada qual com sua característica. Algumas distribuições são maiores outras menores, dependendo do número de aplicações e sua finalidade. Algumas distribuições de tamanhos menores cabem numa disquete com 1,44 MB, outras precisam de vários CDs, existindo até algumas versões em DVD. Todas elas tem o seu público e sua finalidade, as pequenas (que ocupam poucas disquetes) são usadas para recuperação de sistemas danificados ou em monitorização de redes de computadores.

De entre as maiores, distribuídas em CDs, podem-se citar: Slackware, Debian, Suse e Red Hat. O que faz a diferença é como estão organizadas e pré-configuradas as aplicações.

Existem distribuições com ferramentas para configuração que facilitam a administração do sistema. As principais diferenças entre as distribuições estão nos seus sistemas de pacotes, nas estruturas dos directórios e na sua biblioteca básica. Por mais que a estrutura dos directórios

siga o mesmo padrão, o FSSTND (*Filesystem Hierarchy Standard*) é um padrão muito relaxado, principalmente em arquivos onde as configurações são diferentes entre as distribuições. Então normalmente todos seguem o padrão FHS (*File Hierarchy System*), que é o padrão mais novo. Vale a pena realçar, que qualquer aplicação ou *driver* desenvolvido para Linux pode ser compilado em qualquer distribuição que vai funcionar da mesma maneira.

Quanto à biblioteca, é usada a biblioteca `libc`, contendo funções básicas para o sistema operativo Linux. O problema está quando do lançamento de uma nova versão da biblioteca. Algumas das distribuições colocam logo a nova versão, enquanto outras aguardam um pouco. Por isso, alguns programas funcionam numa distribuição e noutras não. Existe um movimento *Linux Standard Base* (LSB) que proporciona uma maior padronização. Auxilia principalmente vendedores de *software* que não liberam para distribuição do código fonte, sem tirar características das distribuições. O sistema de pacotes não é padronizado.

Caixa Mágica, Debian, Fedora, Mandriva, Red Hat, Slackware, SuSE e Ubuntu são algumas das distribuições mais utilizadas actualmente, listadas aqui por ordem alfabética. De entre as distribuições consideradas mais difíceis de gerir (por preferirem assegurar a estabilidade tecnológica em detrimento da interface de utilizador), destacam-se a Debian, Gentoo e Slackware.

O Debian é especialmente conhecido pelo seu sistema de gestão de pacotes, chamado *Advanced Packaging Tool* (APT), que permite: actualizações relativamente fáceis a partir de versões realmente antigas; instalações quase sem esforço de novos pacotes e remoções limpas dos pacotes antigos.

Figura 2.3: Debian logo

Várias distribuições comerciais baseiam-se (ou basearam-se) no Debian, incluindo: Linspire (antigo Lindows), Xandros, Kurumin, Debian-BR-CDD, Libranet e a mais popular de momento, Ubuntu.

O ciclo de desenvolvimento das versões do Debian passa por três fases:

- *Unstable* - instável
- *Testing* - teste
- *Stable* - estável

Uma pequena curiosidade: quando as versões estão na fase *testing* elas são identificadas por *codenomes* tirados das personagens do filme Toy Story. Ao se tornarem *stable* as versões recebem um número de versão (ex: 4.0).

A actual versão *stable* é a "Etch", a *testing* é a "Lenny"(a próxima versão, após a estabilização do "Lenny"será a "Squeeze") e a versão *unstable* terá sempre o nome Sid (também uma personagem do filme Toy Story).

Inicialmente, todos os pacotes são apenas disponibilizados na versão *unstable*, que contém a versão mais recente de cada pacote. No entanto, o novo código é também código não testado, por isso os pacotes são mantidos nesta área de desenvolvimento durante vários dias (a duração exacta depende da urgência do *upload*).

Para um pacote passar da área de desenvolvimento para a versão *testing* – ou seja, o grupo de pacotes que são candidatos a fazer parte da próxima *release* da distribuição Debian – tem de cumprir vários critérios:

- tem de ter estado na área desenvolvimento durante uma duração apropriada de tempo;
- não pode ter nenhum *bug release-critical* a ele associado (*bugs* tão sérios que fazem com que não possam ser libertados);
- tem de ser compilado para todas as arquitecturas planeadas para a *release*;
- não pode depender de versões de nenhum pacote que não cumpra as condições anteriormente definidas;

Desta forma, como é de esperar, um *bug release-critical* num pacote de que vários pacotes dependam, como por exemplo uma *shared library*, pode impedir muitos pacotes de entrarem na área de testes, porque essa biblioteca é considerada deficiente. [7]

2.2 IPBrick - A ideia

A proliferação das vantagens de um sistema *open source* como o Linux suscita bastante interesse a utilizadores particulares mas principalmente a empresas pela possibilidade de baixar os seus custos em *software*. Mas coloque-se este cenário: uma empresa precisa de um servidor para os serviços de rede e/ou comunicações e tem a oportunidade de optar por um sistema *open source*, mas não tem conhecimentos suficientes ou mesmo nenhuns para o configurar. Isto porque ao verificar que a sua instalação e configuração é em grande parte manual, precisa de possuir profundos conhecimentos para todos os serviços e mesmo assim não significa que consiga com sucesso e rapidez suficiente. Então pensa em contratar um administrador de sistemas especializado em Linux mas chega à conclusão que os custos não justificam. Mas e se existir um sistema *open source* fácil e rápido de instalar, pré-configurado e sem a necessidade de ter conhecimentos em Linux para fazer alterações e com uma interface de gestão gráfica?

Foi com este pensamento que surgiu a ideia para a IPBrick.

É um servidor integrado completo e possui uma interface de gestão funcional com acesso via *web*, que permite efectuar a sua configuração sem que o seu administrador possua conhecimentos Linux ou mesmo de redes e serviços. Para além da interface funcional, a IPBrick tem ainda uma

interface avançada, a partir da qual um administrador de redes e sistemas tem acesso directo a todos os seus serviços, tais como *Domain Name System* (DNS), *Dynamic Host Configuration Protocol* (DHCP), *Lightweight Directory Access Protocol* (LDAP), servidor de correio electrónico, servidor de ficheiros, gestor de projectos, *Virtual Private Network* (VPN) *server*, *firewall* entre outros.

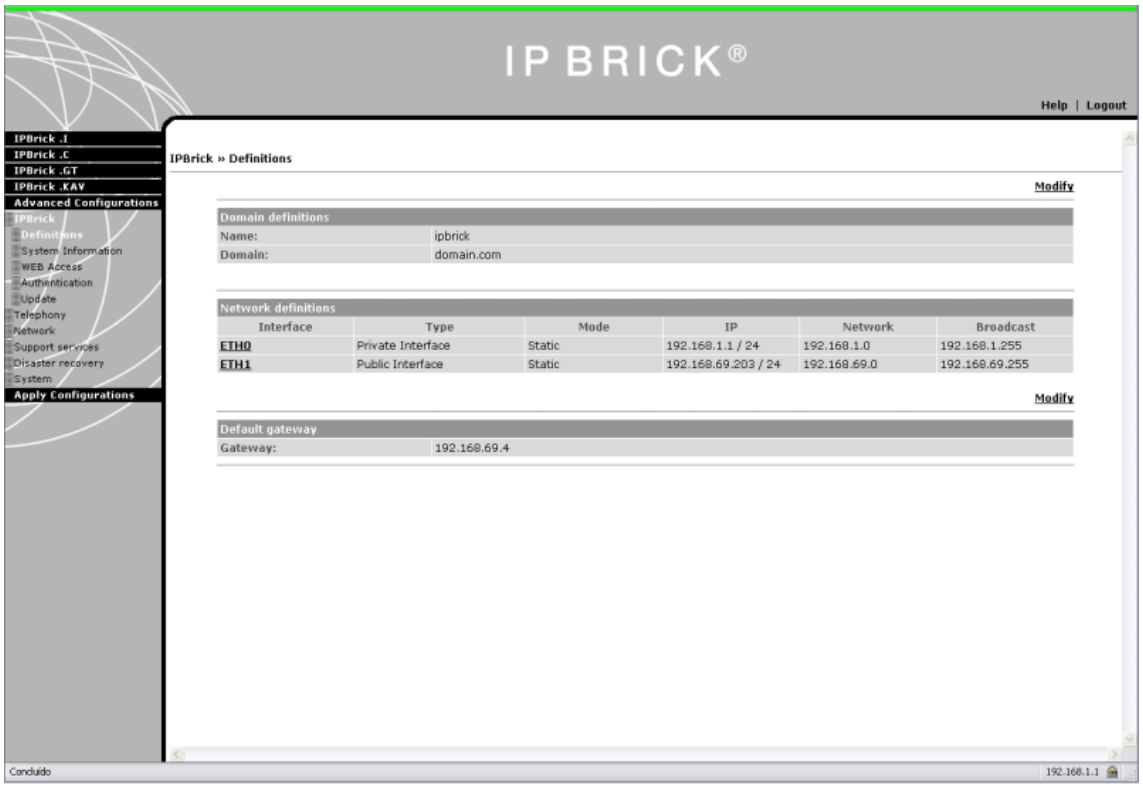

Figura 2.4: Interface inicial de gestão da IPBrick

A IPBrick baseia-se na distribuição Debian. Presentemente na versão 5, esta tem como base a versão mais recente do Debian, a Etch.

E porquê Debian? Porque esta é uma das distribuições mais estáveis, customizáveis, fiáveis e a sua gestão de pacotes facilita a instalação e configuração.

Em fase final de desenvolvimento encontra-se a IPBrick baseada nas distribuições Ubuntu e Red Hat Enterprise. Esta última implementa o sistema IPBrick com uma base dados e servidor de aplicações Oracle com a mesma facilidade e rapidez.

2.3 Customer Relationship Management

Customer Relationship Management (CRM) é uma expressão em inglês que pode ser traduzida para a língua portuguesa como Gestão de Relação com o Cliente. Foi criada para definir toda uma classe de ferramentas que automatizam as funções de contacto com o cliente. Essas ferramentas compreendem sistemas informatizados e fundamentalmente uma mudança de atitude corporativa, que objectiva ajudar as companhias a criar e manter um bom relacionamento com seus clientes

armazenando e inter-relacionando de forma inteligente, informações sobre suas actividades e interacções com a empresa.

No fundo tudo começou, com a necessidade das empresas manterem e tratarem o melhor possível o seu bem mais precioso, o cliente. Tornou-se então necessário fazer um trabalho de verdadeiro comercial de 1 para 1, em que se adapta os objectivos e as palavras às necessidades e ao negócio do cliente. Saber que um administrador gosta, por exemplo, de uma abordagem objectiva e curta, e outro prefere conhecer por completo o que vai comprar. Ou que um deve ser abordado no inicio do dia e não no fim do dia. Tudo isto, resume-se então no seguinte:

Se conhecer o cliente, os seus gostos, as suas necessidades, o que fez, para onde evolui, e o que quer, e tiver isso sempre presente, a percentagem de sucesso no relacionamento com ele sobe exponencialmente. A melhor forma de se conhecer alguém, pessoa ou conceito, é estudá-lo a fundo, e com tempo, a estudar o seu comportamento e as suas reacções, começar a traçar-lhe um perfil. Para que tudo isto aconteça, é preciso tempo, e é preciso história. Dando atenção ao cliente, mostrar esse interesse, registar as suas reacções... e finalmente poderá ser classificado.

Imagine-se as diferenças competitivas entre duas empresas que tratam os seus clientes, uma como qualquer outro, e outra que sabe como o mesmo gosta de ser tratado, quais as suas preferências, e quais foram as suas reacções a factos passados. No fundo fala-se de ganhos de objectividade, tempo, e satisfação de quem pretende um serviço sendo servido de uma forma em que alguém do lado do fornecedor antevê essa necessidade e a apresenta de uma forma pró-activa e pragmática.

O CRM abrange, na generalidade, três grandes áreas:

- Automatização da gestão de *marketing*
- Automatização da gestão comercial, dos canais e da força de vendas
- Gestão dos serviços ao cliente

Os processos e sistemas de gestão de relacionamento com o cliente permitem que se tenha controlo e conhecimento das informações sobre os clientes de maneira integrada, principalmente através do acompanhamento e registo de todas as interacções com o cliente, que podem ser consultadas e comunicadas a diversas partes da empresa que necessitem desta informação para guiar as tomadas de decisões. [2]

Qualquer informação relevante para as tomadas de decisões podem ser registadas, analisadas periodicamente, de forma a produzir relatórios de gestão.

- **CRM Operacional**: visa à criação de canais de relacionamento com o cliente.
- **CRM Analítico**: visa a obter uma visão consistente do cliente, usando os dados recolhidos pelo CRM operacional para obter conhecimento que permita optimizar e gerar negócios.
- **CRM Colaborativo**: foca na obtenção do valor do cliente através de colaboração inteligente, baseada em conhecimento.

Porquê usar CRM? As empresas actualmente reconhecem que para alcançar uma vantagem competitiva sustentável devem tornar-se empresas dirigidas ao cliente. Devem trabalhar e fazer todo o seu esforço para lhe agradar. È o seu bem mais precioso. O seu modo de processar o negócio deve ser desenhado com o cliente em mente, simplificando-lhe a forma de fazer negócio. Tornando-se empresas dirigidas ao cliente, as mesmas, mantêm e crescem de um modo sustentado a sua carteira de clientes e de participações. Quantas empresas no nosso dia a dia, ganham 20% de novos clientes anualmente, e perdem 15% dos clientes passados. É um esforço inglório e que reduz drasticamente o esforço do crescimento, uma vez que o esforço do novo terreno conquistado, é compensado negativamente pela perda de terrenos conquistados no passado.

Para que seja possível dar a um cliente uma experiência que o conduza a reter-se como capital imobilizado da empresa, deve ser aplicada a melhor política em todos os aspectos de gestão de relacionamento. Isto significa, por exemplo, fazer negócios do modo que o cliente quer – a qualquer hora e em qualquer lugar - através de qualquer canal de comunicação.

Significa personalizar toda a interacção e oferecer aos clientes apenas aquilo (produtos e serviços) que vão de encontro às suas necessidades.

Os benefícios de fornecer ao cliente experiências excepcionais e pessoais, são claros. Os clientes satisfeitos:

- Tornam-se fiéis e entram no "capital da empresa";
- Acreditam no seu fornecedor;
- Levantam menos questões;
- Compram produtos e serviços excepcionais;
- Respondem a ofertas *cross-sell* e *up-sell*;
- São menos sensíveis ao preço que os outros;

ou seja, como sabem com o que contam, compram com a confiança, de que quem os serve pensa nos seus interesses.

Reconhecendo os benefícios de ser uma empresa dirigida ao cliente, os empresários começam a focar os seus investimentos de tecnologias de informação, naqueles que apostam na melhoria do relacionamento com o cliente (CRM) – a tecnologia que capacita a empresa a entregar uma experiência superior ao cliente. Enquanto as melhores práticas no processo de contacto com o cliente – vendas, *marketing*, e serviço – são essenciais para o sucesso, é preciso uma experiência extensiva para identificar e documentar as melhores práticas. As aplicações CRM que as empresas seleccionam para apoiar estes processos devem reflectir melhores práticas e por tal, devem basear-se num registo provado de experiências e de sucesso.

A maioria das empresas reconhece que não são dirigidas ao cliente e, por consequência, não operam no seu total potencial para adquirir, reter e aumentar a sua carteira de clientes. Também reconhecem que estes não estão totalmente satisfeitos quando os negócios são feitos, e que ano após ano, ganham novos clientes que apenas compensam a saída de outros.

A maioria das empresas de serviços, aquando de um contacto comercial de um cliente, desconhecem se se trata de um cliente com assuntos pendentes com a empresa, ou se é um cliente que não contacta a empresa há vários meses. Obrigam o cliente a justificações intermináveis que apenas demonstram a pouca atenção e importância que lhe dão. O mais grave é que estes acontecimentos também se passam nas grandes empresas. [8]

2.4 Enterprise Service Bus

Um *Enterprise Service Bus* (ESB) move dados entre múltiplos *endpoints*, seja dentro ou fora de uma empresa. Usa padrões abertos para conectar, transformar, e distribuir documentos de negócios como mensagens de *Extensible Markup Language* (XML), entre aplicações. Permite a monitorização e a gestão de dados do negócio, com o mínimo de impacto em aplicações existentes. Um *Enterprise Service Bus* é a infra-estrutura básica para desenvolver uma *Service-Oriented Architecture* (SOA) e uma *Event-Driven Architecture* (EDA). [9]

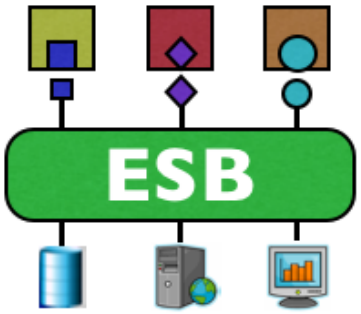

Figura 2.5: *Enterprise Service Bus*

Um ESB geralmente fornece uma abstracção de camadas na implementação de um sistema empresarial de mensagens, que permita integração da arquitectura para explorar o valor das mensagens sem escrever código. Contrariando a clássica integração de aplicações comerciais (*Enterprise Application Integration* - EAI). A base de um ESB é construída da quebra de funções básicas em partes, que são distribuídas onde for preciso.

ESB não implementa uma arquitectura orientada a serviço (SOA), mas fornece as características para que possa ser implementado. Não precisa necessariamente de ser implementado usando *web-services*, mas devem ser baseados em padrões flexíveis, suportando vários meios de transportes. Baseado no EAI melhor que padrões SOA, este tenta remover o acoplamento entre o serviço que é chamado e o meio de transporte.

Em uma arquitectura empresarial fazendo uso de um ESB, uma aplicação irá comunicar via o barramento, que actua como um *message broker* entre aplicações. A principal vantagem desta abordagem é a redução de conexões ponto a ponto necessárias para permitir a comunicação entre aplicações. Isto por sua vez afecta directamente na simplificação das mudanças de sistema. Por reduzir o número de conexões ponto a ponto para uma aplicação específica, o processo de adaptar um sistema às mudanças em um dos seus componentes torna-se mais fácil. [10]

Há cinco aspectos principais que precisam de ser considerados ao selecionar um ESB:

1. Desempenho e escalabilidade
   É essencial que a infra-estrutura utilizada para a integração dentro de uma SOA forneça o desempenho esperado dos sistemas empresariais, enquanto processa alterações a pedido com relativa facilidade. É crítico que um ESB não crie *bottlenecks* ou *chokepoints* na infra-estrutura que poderá restringir a amplitude da sua implementação ou limitar o fluxo de dados que pode transportar para/ou a partir de recursos ligados. Um *Enterprise Service Bus* com uma arquitectura distribuída irá facilitar o processamento dos serviços e da subjacente infra-estrutura de comunicações. Embora a distribuição é um meio eficaz para garantir a produção elevada e escalabilidade, é importante que a implementação física não tenha impacto sobre a lógica e que uma infra-estrutura altamente distribuída seja fácil de gerir, acompanhar e controlar.

2. Segurança, fiabilidade e disponibilidade
   Um ESB deverá prestar qualidade do serviço a nível empresarial, para garantir que o serviço de comunicação é tão seguro e fiável ao ponto de cumprir as necessidades requeridas de uma determinada empresa. Ao fornecer essas garantias, os serviços de desenvolvimento podem depender desses recursos, simplificando dramaticamente o trabalho destes para se concentrarem no desenvolvimento da actividade empresarial lógica, não a integração lógica. Por exemplo, serviços de produção não precisam de gerir a retransmissão de dados se os serviços aos consumidores estão temporariamente indisponíveis; uma infra-estrutura que garante a entrega irá armazenar os dados e transmiti-los quando o serviço de consumidor estiver disponível. A integração de uma infra-estrutura SOA altamente disponível isola aplicações de falhas resultantes de falhas do servidor e de comunicação. Um ESB altamente disponível irá apresentar-se aos serviços não apenas como uma infra-estrutura confiável, mas uma integração cujos serviços estão permanentemente disponíveis, mesmo que um componente na sua infra-estrutura distribuída falhe.

3. Distribuição
   Num SOA, os serviços irão interagir com os serviços espalhados por uma organização e entre organizações. Um ESB fornece facilidades de comunicação que as une com qualidade de serviço configurável e mensagens semânticas, com ambos a terem um comportamento de *queuing* e *publish-and-subscribe*. Isto permite a recolha e distribuição de dados, bem como a notificação de eventos empresariais e correlação das respostas dos serviços subscritos. Num ambiente em que os serviços ultrapassam as fronteiras da empresa ou num em que os serviços de comunicação ultrapassam barreiras geográficas, é crítico que os dados, eventos, e as respostas sejam encaminhadas para o lugar certo no momento certo, sem gestão adicional. Capacidades de distribuição mais avançadas permitem uma extensão flexível do ESB para incorporar e ligar recursos em domínios de segurança adicional, domínios sem qualquer reconfiguração ou interrupção de funcionamento de sistemas.

4. Flexibilidade
   A flexibilidade de uma arquitectura ESB correctamente projectada, permite a uma organização mudar o esquema de rede, as regras, *data mapping* e as relações entre as aplicações com o mínimo de esforço e de quebra de serviço. Aqui o requisito principal é que seja possível mudar dinâmicamente os serviços, processos e esquema sem a interrupção dos sistemas em uso. Ao mesmo tempo, a arquitectura deve permitir total liberdade para definir e mudar a implementação física de modo a ser mais eficaz, processamento escalável sem qualquer impacto para a semântica ou comportamento das integrações. Sofisticada semântica *publish-and-subscribe*, permite o *broadcast* simultâneo de informação ou de notificações de eventos de negócios para um grande número de partes interessadas, e também tornar mais fácil para expandir o sistema sem fazer mudanças para o prestador de serviços. Ao isolar serviços uns dos outros, o ESB cria uma *framework* para uma integração *loosely-coupled*, permitindo assim uma gestão de mudança muito mais fácil num ambiente complexo.

5. Visibilidade e controlo
   Um ESB deve gerir e monitorizar a infra-estrutura, bem como os processos e serviços implementados nele. A primeira tarefa de gestão é a implementação de serviços, que podem ser hospedados em qualquer lugar em que o ESB é executado. Uma vez que uma implementação pode ser altamente distribuída, o ESB deverá tornar mais fácil a implementação e actualização de serviços remotamente a partir de uma localização central. Um ESB é orientado à configuração de maneira a ser dinâmico: os serviços são parametrizados e as suas relações são declaradas, assim eles podem ser alterados sem recompilação e reimplementação. Isto permite a modificação de dados sem ter que parar serviços em execução. Para facilitar o desenvolvimento no mundo real, ambientes distribuídos, o ESB necessita de mensagens de monitorização em tempo real e um fluxo de controlo distribuído para *debug*. Por último, tendo em vista permitir o diagnóstico e gestão de problemas em sistemas distribuídos complexos, recursos administrativos devem fornecer serviços de *logging* e auditoria, bem como a monitorização de falhas, serviços e estado dos processos, e de estatísticas detalhadas sobre o desempenho da infra-estrutura de comunicação do ESB.

Vantagens de um *Enterprise Service Bus* :
A nível das TI:

- Redução no tempo e esforço necessário para integrar novas e aplicações existentes;
- Cria novos processos através da reutilização de aplicações existentes e dados;
- Aumenta a flexibilidade para mudar sistemas complexos de comportamento ao minimizar as dependências não conhecidas entre aplicações, serviços e *middleware* num ambiente distribuído;
- Reduz o *total cost of ownership* e aumenta a flexibilidade para acomodar as necessidades futuras através da utilização de protocolos e interfaces *standard*;

- Fiável entrega das mensagens através dos serviços, mesmo depois de falhas no *software*, *hardware* ou rede;
- Oferece um serviço de alojamento e gestão de infra-estrutura que é altamente distribuído, sendo no entanto gerida centralmente;
- Divulga informação ao longo de todo a empresa, bem como a clientes e parceiros comerciais;
- Pode ser constantemente implementado, acelerando a entrega do serviço aos clientes e reduzir os riscos para grandes e complexos projectos;

A nível empresarial:

- Sistemas empresariais integrados: redução nas despesas operacionais, melhoria do serviço aos clientes, facilita fusões e aquisições, mais fácil decidir e tomar decisões mais exactas baseadas em informação actualizada para a *"single version of the truth"*;
- Redução dos custos de TI e de controlo através da padronização de componentes de negócios reutilizáveis;
- Melhora o tempo de resposta face à mudança de processos de negócios de forma rápida e eficiente;
- A capacidade de responder a condições especiais de negócios à medida que surgirem;

Parte do texto apresentado nesta secção foi retirado e traduzido da página da SonicSoftware. [11]

Capítulo 3

Arquitecturas Tecnológicas Open Source: IPBrick, SugarCRM e Mule

Neste capítulo apresentam-se as tecnologias utilizadas para a realização do trabalho.

3.1 IPBrick

Cada vez mais as empresas estão dependentes da partilha da informação e das comunicações, onde a velocidade e segurança são sempre factores importantes. A administração de redes, a gestão dos seus recursos, impressoras, ficheiros, contas de email, acessos à Internet, segurança contra intrusões e outras ameaças que "espreitam"qualquer sistema que não se encontre isolado do mundo, os vírus e os *worms*, podem tornar a vida de uma empresa num suplício.

Figura 3.1: IPBrick logo

A IPBrick é um sistema UCoIP, traduzido para português como Comunicações Unificadas sobre IP. Isto significa que as comunicações implementadas pela IPBrick podem ser baseadas unicamente no endereço do utilizador para todos os tipos de comunicação: Voz, Email, *Instant Messaging* e *Web*. Isto é alcançado usando unicamente serviços de comunicação da Internet: SIP, SMTP/IMAP, XMPP e HTTP, assente nos serviços de suporte DNS e LDAP devidamente integrados. O conceito de UCoIP implementado pela IPBrick apenas sobre protocolos da Internet Engineering Task Force (IETF) constitui a forma mais aberta de permitir a comunicação unificada entre equipamentos de fabricantes diferentes.

É um sistema para gestão de redes com administração simples e funcional, devido à sua interface gráfica *Graphical User Interface* (GUI - Interface Gráfica do Utilizador). Desta forma, as empresas não dependem mais de peritos informáticos para manter a espinha dorsal do negócio em funcionamento com elevada disponibilidade.

Como o centro de todo o trabalho colaborativo, pode ligar-se através do Microsoft Outlook ou através de um *browser web*. A tecnologia baseada em Linux garante confiança, velocidade e eficiência nas transferências de dados. Com a integração do antivírus Kaspersky minimiza-se o risco de perda da integridade dos dados.

Constituída por um modelo de serviços de Intranet e de comunicações, válido em qualquer tipo de empresa, tornando assim fácil a configuração de todos os serviços sem quaisquer conhecimentos de Linux e/ou redes. As configurações mais específicas podem ser realizadas por técnicos qualificados através da interface avançada.

3.1.1 Arquitectura

Como foi dito no capítulo anterior, a base da IPBrick é a distribuição Debian. Com uma instalação básica do Debian são posteriormente instalados todos os pacotes de *software* e respectivas dependências relativos aos serviços utilizados pela IPBrick. Todos estes serviços irão depois ser configurados, para assim que a instalação da IPBrick esteja finalizada, esta está pronta a ser usada, porque já tem todos os serviços instalados e pré-configurados.

O seu modelo de funcionamento é o seguinte:

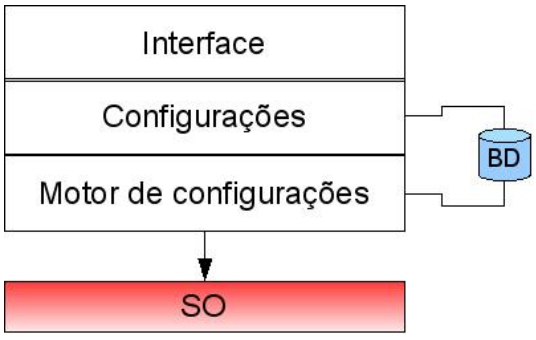

Figura 3.2: Modelo de funcionamento da IPBrick

Quando o administrador efectua alterações na configuração dos serviços através da interface da IPBrick, estas alterações não têm efeito imediato. Poderá fazer as modificações necessárias e em vários serviços de acordo com o pretendido, sem se preocupar que possa estar a interromper o normal funcionamento de um serviço que prejudique os utilizadores. As configurações ficam guardadas e quando o administrador finalizar, confirma-as na interface da IPBrick e o seu "motor" de configurações irá gerar os ficheiros de configuração dos serviços que foram alterados.

É constituída por várias versões de acordo com as necessidades de uma empresa:

Tabela 3.1: Versões IPBrick disponíveis

Modelo	Destina-se a
IPBrick.I	Serviços de Intranet (DNS, DHCP,LDAP...)
IPBrick.C	Serviços de Comunicação (WEB, PROXY, FTP...)
IPBrick.GT	*Gateway* de Comunicações (VoIP, PBX...)
IPBrick.KAV	Segurança

3.1.2 IPBrick.I

A IPBrick.I disponibiliza todos os serviços necessários de um servidor de Intranet. IPBrick.I pode ser o servidor de email, ficheiros, domínio, fax, impressão e de *backup*. Para além do calendário/agenda partilhados e gestor de contactos (sincronizados via MS Outlook ou *browser web*), é possível gerir todos os projectos de um ponto central.

IPBrick.I integra-se facilmente com serviço de directório. Em termos de escalabilidade oferece um conceito *master/slave* baseado em LDAP e MS *Active Directory*.

O servidor de Intranet IPBrick.I pode ser utilizado em três modos diferentes: *master*, *slave* e *Active Directory client*. Consequentemente os utilizadores da IPBrick, grupos e dispositivos (terminais e telefones SIP (*Session Initiation Protocol*)) podem ser definidos na IPBrick ou no Windows AD (*Active Directory*). Na Intranet, é capaz de funcionar num ambiente totalmente IPBrick ou integrada com sistemas Windows.

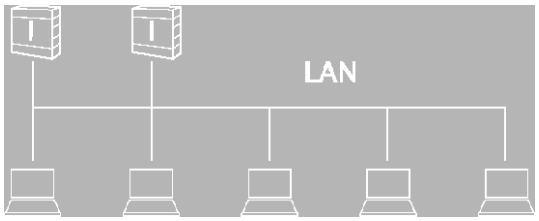

Figura 3.3: IPBrick.I inserida na LAN de uma empresa [12]

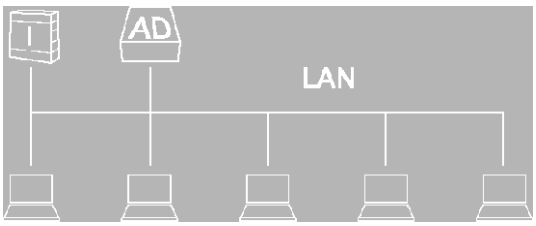

Figura 3.4: IPBrick.I na LAN de uma empresa com AD [12]

A IPBrick.I disponibiliza o seguinte conjunto de serviços, sendo estes apresentados na tabela 3.2.

3.1.3 IPBrick.C

A IPBrick.C garante a segurança na transferência de dados entre Intranet e Internet. Os dados transferidos (*download and upload*) são filtrados, analisados e geridos pela IPBrick.C. Estas operações são executadas pelos principais componentes da IPBrick.C: *Mail-Relay*, Servidor-Proxy, IDS (*Intrusion Detection System*), Antivírus e AntiSpam. A IPBrick.C controla todas as interfaces disponíveis para a Internet. O objectivo da IPBrick.C é gerir as ligações da sua empresa com a

Tabela 3.2: Serviços da IPBrick.I

Serviços
Servidor de áreas de trabalho individuais e de grupo
Servidor LDAP (gestão de máquinas, utilizadores e grupos)
Servidor de Domínio (com *roaming-profiles* e *centralized scripting*)
Servidor de correio electrónico
Servidor de impressoras
Servidor de base de dados
Servidor de DHCP (*Dynamic Host Configuration Protocol*)
Servidor de DNS (*Dynamic Network Services*)
Servidor de imagens de estações de trabalho
Servidor de Groupware (Calendário e Livro de Endereços)
Servidor de *Backup* (ARKEIA)
Gestor de projectos (dotProject)
Serviços Opcionais
Kaspersky Anti-Malware
Correio Electrónico (AntiVírus, AntiSpam)
Servidor de ficheiros (AntiVírus)
Single sign-on (Autenticação Biométrica)

Internet. Os serviços oferecidos são: servidor *web*, servidor FTP (*File Transfer Protocol*), VPN (*Virtual Private Network*) *gateway* e telefonia VoIP (voz sobre IP).

A tecnologia VPN não só permite o fluxo de dados seguro mas também permite a escolha do espaço de trabalho conforme as necessidades. A IPBrick.C gere as ligações permanentes entre diversos pontos através de túneis IPSec (*IP Security*) assim como através de túneis OpenSSL (*Secure Sockets Layer* e *Transport Layer Security*) para ligações de confiança. O correio electrónico assim como o tráfego *web* é continuamente filtrado através de regras configuráveis. Relatórios com estatísticas apresentam a informação essencial em gráficos de interpretação fácil. Através do *web mail* podem ser geridos os e-mails de uma forma segura, mesmo que através de um *browser web* em qualquer lugar e momento.

A IPBrick.C pode ser colocada numa DMZ (*DeMilitarized Zone*) protegida por *firewall*, como um servidor de comunicações protegida pela *firewall*, ou mesmo como um servidor completo de comunicações e sistema de *firewall* integrado. A IPBrick.C pode importar os utilizadores, grupos e dispositivos (estações de trabalho e telefones SIP) de uma IPBrick.I ou de um sistema Windows AD. Isto significa que, instalando a IPBrick.C como servidor de comunicações, não é necessário redefinir as informações do sistema já disponíveis no servidor de Intranet da empresa.

A IPBrick.C disponibiliza os seguintes serviços de empresa para a ligar à Internet, sendo estes representados na tabela 3.3.

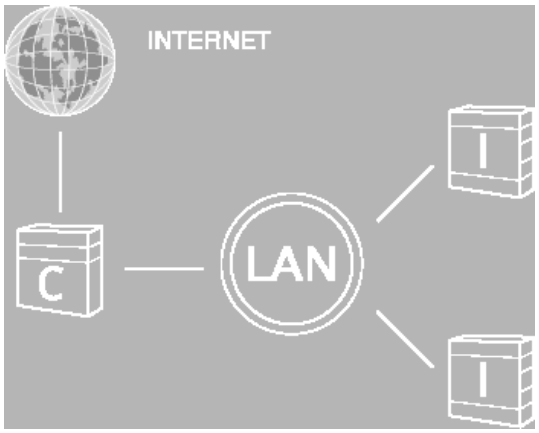

Figura 3.5: IPBrick.C numa DMZ com sistema de *firewall* [12]

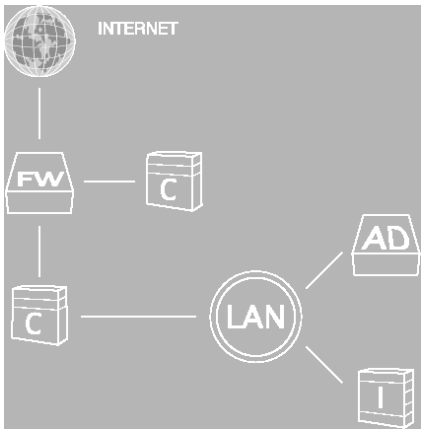

Figura 3.6: IPBrick.C numa DMZ protegida por *firewall* [12]

Tabela 3.3: Serviços da IPBrick.C

Serviços
Relay de correio electrónico
webmail
Servidor *web*
Servidor *Proxy* (HTTP, HTTPS, FTP com estatísticas)
Traffic shaping
Port security (*packet based firewalling*)
IDS (*Intrusion Detection System*)
Servidor de VoIP (RTP bases *routing*)
Integração transparente com PBX (ISDN E1/BRI e linhas analógicas)
Serviços Opcionais
Kaspersky Anti-Malware
Correio Electrónico (AntiVírus, AntiSpam)
Servidor de ficheiros (AntiVírus)
Integração transparente com PBX (ISDN E1/BRI e linhas analógicas)

3.1.4 IPBrick.GT

A IPBrick.GT complementa a solução de *software* IPBrick.IC com *hardware* seleccionado para permitir uma integração optimizada de voz e dados, esta responde às exigências da integração necessária entre a telefonia tradicional, os novos telefones VoIP SIP e fornecedores de SIP.

A IPBrick.GT implementa um *gateway* completo PSTN/VoIP, permitindo a ligação directa ao PBX (RDIS E1/BRI e linhas analógicas), a operadores PSTN, à LAN (*Local Area Network*) da empresa e à Internet. O encaminhamento inteligente permite o fluxo de voz através de VoIP, ou através do tradicional RDIS, ou mesmo linhas analógicas. A IPBrick integra uma funcionalidade de controlo de tráfego que irá dar prioridade todas as ligações de telefonia. O número de chamadas telefónicas activas e simultâneas é apenas limitado pela capacidade da ligação à Internet.

Tabela 3.4: Características da IPBrick.GT

Características
Comunicação de dados VoIP baseada em protocolos standard de Internet (SIP, RTP)
Transferência segura de dados VoIP através de túneis VPN definidos
Negociações Voice Call inteligentes para servidores VoIP estrangeiros, Fornecedores de SIP e redes de telefone tradicionais.
Controlo da ligação entre a linha RDIS/Analógica e o seu PBX
Uma evolução da migração, incorporando a herança dos telefones RDIS/Analógicos e PBX para a tecnologia VoIP
Integração com operadores de VoIP
Características avançadas de QoS com integração de gestão de largura de banda

3.1.5 IPBrick.KAV

A IPBrick.KAV é uma *appliance* de segurança baseada no *software* de base IPBrick.IC com o objectivo de proteger as empresa em quatro vertentes de segurança mais importantes da actualidade: correio electrónico, acessos *web*, segurança de rede e de Intranet.

Impede que as estações de trabalho se liguem directamente à Internet, não permitindo que programas do tipo *trojan* estabeleçam túneis com o exterior, ou abram *backdoors* que permitem o acesso à rede da empresa a partir do exterior, eliminando os problemas causados por *trojans*. Para isso a IPBrick dispõe de um conjunto de serviços funcionando em modo *proxy*.

Foi concebida para, através de um servidor de comunicações instalada numa *appliance* desenhada com elevado nível de integração ao nível do *hardware*, do sistema operativo de rede IPBrick.IC e *software* de segurança da Kaspersky, aumentar a segurança das redes empresariais.

O *software* de segurança da Kaspersky elimina na protecção perimetral assumida pela IPBrick.KAV todos os conteúdos de *malware* (*trojans*, *worms*, *spyware*, *phishing*, vírus, entre outros) que se destinam a infectar os computadores da rede interna.

A IPBrick.KAV apresenta as seguintes características:

É um elemento essencial para a segurança da empresa, mas também oferece funcionalidades para melhorar a operação da sua empresa, na interacção entre a rede interna e a Internet:

- *webmail*: para poder consultar o seu correio electrónico a partir de qualquer lugar de forma segura;
- VPN SSL: para poder aceder com total conforto e segurança aos dados e aplicações da sua empresa desde o exterior;
- VoIP: para telefonar via Internet;
- Servidor *web*: para disponibilizar conteúdos desde a sua rede local para a Internet.

Tabela 3.5: Características da IPBrick.KAV

Características
Segurança *web*
Anti-vírus Kaspersky, remover conteúdos perigosos existente em sites
Black and White lists para controlo dos acessos à Internet impedindo ou limitando o acesso a certos
Segurança de Mail
Anti-vírus e Anti-spam da Kaspersky, limpar conteúdos perigosos enviados por correio electrónico
Denial of service, reduz os ataques que pretendem bloquear o correio electrónico da empresa (*mail bombing*)
Segurança de Rede
Firewall, para impedir acessos não desejados à rede interna a partir do exterior
VPN, aumentar nível de segurança do acesso dos utilizadores aos recursos internos
Detecção de Intrusão, identificar ou alertar os administradores de acessos suspeitos
Segurança de Intranet
Detecção de máquinas infectadas na Intranet

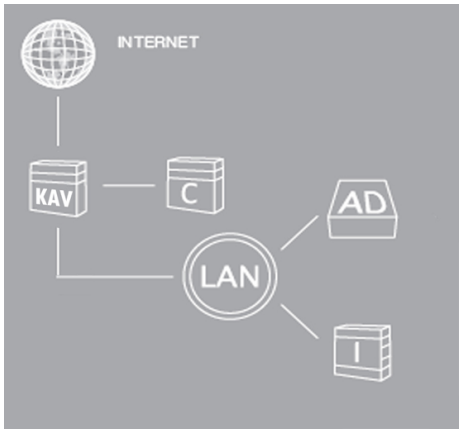

Figura 3.7: IPBrick.KAV - *Appliance* de Segurança [12]

É na IPBrick que se faz a inserção e gestão dos utilizadores bem como dos grupos aos quais eles pertencem. Também é aí que serão criados os registos das entidades (através da aplicação IP Contactos), e respectivos contactos com as quais a instituição interage.

A informação apresentada ao longo desta secção foi recolhida da página oficial do produto [13] e do manual de referência [12].

3.2 SugarCRM

O SugarCRM baseia-se num modelo flexível de *deployment* da solução, totalmente customizável ao utilizador e não o inverso, com liberdade de escolha de módulos a utilizar e assente sobre uma arquitectura *open source* (Plataforma LAMP: Linux, Apache, MySQL, PHP) e compatível com qualquer sistema operativo.

Figura 3.8: SugarCRM Logo

É um produto de CRM corporativo com módulos para gerenciamento de empresas e divisões, contactos, *prospects*, oportunidades, ocorrências, campanhas de *marketing*, projectos, documentos, agenda e histórico.

Caracteriza-se por ter uma forte componente em gestão comercial, gestão de *marketing*, gestão de clientes e *reports*. Tudo isto é acompanhado por um grande suporte a nível de processos administrativos presentes na aplicação. [14]

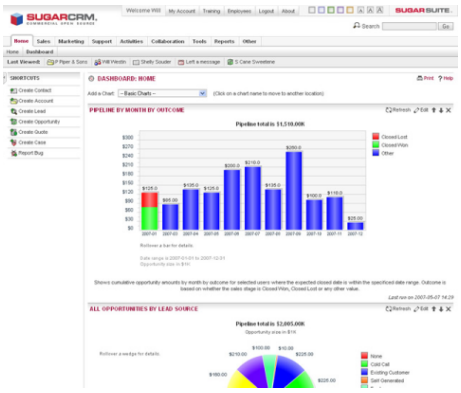

Figura 3.9: SugarCRM *Dashboard* [14]

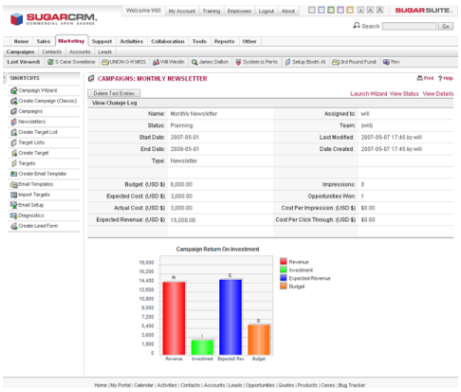

Figura 3.10: SugarCRM *Campaigns* [14]

É desenvolvido em ambiente *open source* e disponível em três versões: Open Source, Professional e Enterprise. A versão Open Source é livre de licenças de uso e possui um número limitado de funcionalidades. As versões Professional e Enterprise são comercializadas e possuem mais funcionalidades.

De seguida serão apresentadas algumas das funcionalidades de cada versão do SugarCRM:

3.2.1 Open Source

As principais características da versão Open Source são:

- Criação e gestão de Contas, com qualquer número de contactos, associado a cada conta;

- Histórico das Actividades (Reuniões, Telefonemas, Assuntos, Notas com arquivos anexados e Emails) realizadas com Contatos, Contas, *Leads*, Oportunidades e Casos;
- Tarefas podem ser atribuídas aos utilizadores, e notificações automáticas via email podem ser opcionalmente enviadas para avisar os utilizadores de novas tarefas;

Automatização da Força de Vendas:

- Visualização resumida de compromissos que estão por vir, melhores Oportunidades, Casos abertos, *Leads*, Tarefas Abertas, Problemas atribuídos, Gráfico de planeamento de vendas, Calendário mensal e facilidade de rápida entrada nos contactos;
- Criação e Acompanhamento de *Leads* de venda e conversão de *Leads* de venda em oportunidades;
- Visualizamento de Painel Gráfico de Planeamento de Oportunidades, fontes de *Lead* & Resultados.

Acompanhamento de Serviço ao Cliente:

- Um sistema de gestão de casos que permite ao utilizador acompanhar os problemas dos clientes e suas resoluções. Permite que cada problema tenha um ciclo de vida de informações para melhorar a satisfação do cliente;
- Cada caso é ligado a Conta relacionada, Contactos, Notas, Arquivos associados, histórico de telefonemas e reuniões.
- Um sistema de detecção de problemas (*bug tracking*) para gerir problemas apontados a diferentes revisões/versões do *software*.

Calendário Corporativo:

- Visualização de Calendário (por Dia, Semana, Mês, ou Ano) de todas as actividades corporativas, com uma lista de Tarefas associadas;
- Calendário compartilhado para visualização de calendários de outros utilizadores para evitar conflitos de programação e agendamento.

Novos Serviços:

- Um Módulo de novas notícias RSS permite seleccionar e gerir as notícias favoritas, e colocá-las na página *My RSS News Feeds*;

Interface Consolidada:

- O Módulo Portal permite que administradores e utilizadores façam *links* de *sites* externos e aplicações da *web* dentro da interface do utilizador do Sugar Suite, permitindo que o Sugar Suite se torne uma interface de informação unificada para os seus utilizadores.

- O Sugar Open Source foi construído baseado em tecnologias de open-source estabelecidas em parâmetros industriais amplamente solidificados, incluindo o desenvolvimento do ambiente em PHP, Base de dados MySQL, servidor web Apache ou IIS, e os sistemas operativos Linux e Windows Server. O sistema suporta ambas plataformas: LAMP (Linux, Apache, MySQL, PHP) e WIMP (Windows, IIS, MySQL, PHP).

3.2.2 Professional

Esta solução destina-se a empresas que queiram mais do que simplesmente registar dados. Fornece uma visibilidade sobre todos os seus projectos em curso, o estado dos mesmos, a prioridade, assim como também sobre os recursos que irão fazer parte do mesmo, a sua disponibilidade ou até a sua carga laboral no momento.

Permite a criação de um *pipeline* de tarefas e a sua respectiva atribuição aos recursos pré-definidos, assim como a duração da mesma, e datas de início e de fecho.

Tem todos os recursos da edição Open Source do Sugar, mais módulos adicionais, extensões e modelos disponíveis apenas na versão profissional, incluindo:

- Gestão de Contactos;
- Gestão de Contas;
- Gestão de *Leads*;
- Gestão de Actividades e Tarefas;
- Gestão de equipas com segurança de dados;
- Anotações;
- Pesquisas;
- Gráficos;
- Internacionalização;
- Importação e exportação de dados do utilizador;
- Anexos;
- Integração com o MS Outlook;
- Calendário;
- Home page editável;
- Relatórios;
- Cotações e Lista de Preços;

3.2.3 Enterprise

Esta é a versão mais completa e robusta.
Fornece todos os recursos da edição Professional do Sugar, mais:

- Suporte para base de dados Oracle 10g e 9i;
- Suporte para cliente *offline* com posterior sincronização dos dados;
- Relatórios através de um editor SQL avançado;
- Portal de serviço ao cliente;

Na seguinte tabela 3.6 apresenta-se um resumo das funcionalidades das várias versões.

Tabela 3.6: Comparativo entre as versões do SugarCRM

Módulos e Funcionalidades	*Open Source*	*Professional*	*Enterprise*
Gestão de Marketing	X	X	X
Gestão de Vendas	X	X	X
Gestão de Serviços	X	X	X
Administração	X	X	X
Multi-idioma	X	X	X
Multi-moeda		X	X
Utilizadores Ilimitados	X	X	X
Workflow		X	X
Principal	X	X	X
Meu Portal	X	X	X
Calendário	X	X	X
Actividades	X	X	X
Contactos	X	X	X
Contas	X	X	X
Leads	X	X	X
Oportunidades	X	X	X
Ocorrências	X	X	X
Bug Tracker	X	X	X
Documentos	X	X	X
Emails	X	X	X
Campanhas	X	X	X
Projetos	X	X	X
RSS (notícias)	X	X	X
Painél	X	X	X
Cotações		X	X
Catálogo de Produtos		X	X
Previsão de Vendas		X	X
Relatórios		X	X
Gestão de Equipas		X	X
Plug in MS Outlook		X	X
Plug in MS Word		X	X
Versão Smart Device		X	X
Suporte Oracle 10g			X
Suporte Oracle 9i			X
Relatórios SQL			X
Cliente Off Line			X

Alguma da informação presente ao longo desta secção foi recolhida da página [1].

3.3 Mule

Mule é um *lightweight Java-based messaging framework* que permite de uma maneira rápida e fácil interligar aplicações de modo a permitir a troca de informação entre elas. Utiliza uma arquitectura SOA, permitindo uma fácil integração de sistemas existentes. Independentemente das diferentes tecnologias que as aplicações utilizam, incluindo JMS, Web Services, JDBC, HTTP, e mais, Mule gere as interacções entre todos eles sem problemas.

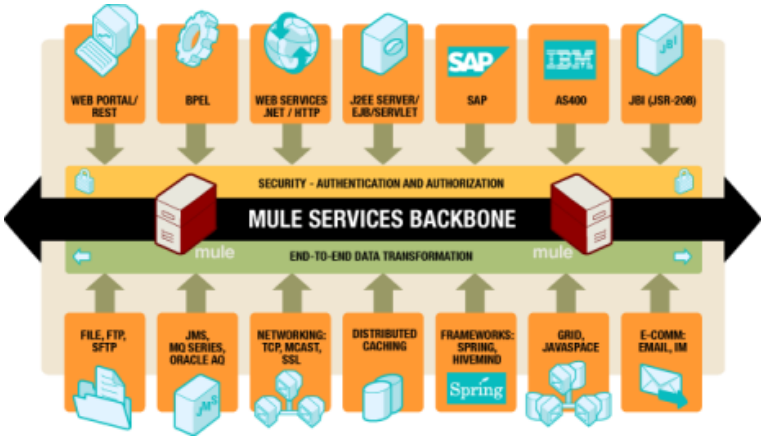

Figura 3.11: Mule ESB

A *framework* do Mule é altamente escalável, permitindo que se começe por interligar poucas aplicações e ir aumentando o número destas ao longo do tempo. Mule gere todas as interacções entre as aplicações e os componentes de modo transparente, independentemente se existem na mesma máquina virtual ou através da Internet, e independentemente do protocolo de transporte utilizado.

Alguns pontos-chave do Mule:

- Os componentes do Mule podem ser do tipo que se desejar. Pode-se facilmente integrar qualquer coisa a partir de um "*Plain Old Java Object*"(POJO) para um componente de outra *framework*.

- Mule e o modelo ESB permitem uma significativa reutilização dos componentes. Ao contrário de outros *frameworks*, Mule permite que se use os componentes existentes sem qualquer alteração. Os componentes não exigem qualquer código específico para ser executado no Mule, e não há nenhuma API exigido. A lógica empresarial é mantida completamente separada das mensagens lógica.

- As mensagens podem ser em qualquer formato desde SOAP a ficheiros binários. Mule não força quaisquer restrições ao desenho da arquitecta, como serviços de mensagens XML ou WSDL.

- Pode-se implementar o Mule numa variedade de topologias, e não apenas ESB. Porque é leve e incorporável, o Mule pode diminuir drasticamente o tempo de entrega e aumenta

a produtividade para projectos de modo a prestar aplicações seguras, escaláveis, que são adaptáveis à mudança e podem escalar para cima ou para baixo, conforme necessário.

3.3.1 Sistema de Mensagens

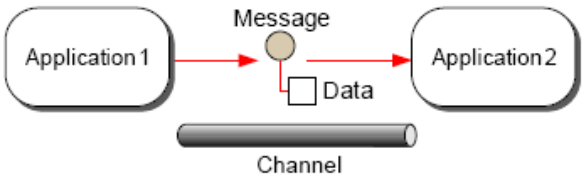

Figura 3.12: Sistema de mensagens do Mule

De um modo simples, quando se conectam aplicações ao Mule, este lê os dados de uma aplicação, transforma-la como necessária para que possa ser lido pela aplicação destino e envia-la. Isto permite integrar todo o tipo de aplicações, mesmo as que não foram construídas para ser integradas.

Uma diferença entre o Mule e um ESB tradicional é que o Mule apenas converte os dados que necessitam. Com um típico ESB, terá que se criar um adaptador para cada aplicação que se interligue com o barramento e converter os dados da aplicação em um único formato de mensagem comum. O desenvolvimento destes adaptadores e o tempo necessário para processar cada mensagem requer muito tempo e esforço. O Mule elimina a necessidade de um formato único de mensagem. A informação é enviada em qualquer canal de comunicação, tais como o HTTP ou JMS, e é traduzido apenas se necessário ao longo do caminho. Por isso, Mule aumenta a performance e reduz o tempo de desenvolvimento relativamente a um ESB tradicional.

3.3.2 Modelo de Programação

O Mule é descrito pelos seus criadores como um *lightweight messaging framework* que gere comunicações entre componentes distribuídos de serviços do cliente, denominado como Universal Messaging Object (UMOs). Um UMO é um simples POJO que recebe e processa mensagens, e que comunica com o resto dos componentes do serviço geridos pelo Mule via *inbound* e *outbound "messaging endpoints"*. Um *outbound endpoint* é onde o Mule despacha a mensagem para outro componente do serviço. UMOs conectam-se a *endpoints* por meio de *routers inbound and outbound*. *Outbound routers* podem incluir filtragem, a fim de canalizar mensagens de *outbound* para um *endpoint* apropriado.

Mensagens *inbound* e *outbound* podem também sofrer uma transformação por parte de um transformador apropriado conectado ao UMO. Um transformador *inbound* fornece mensagens de transformação do formato de transporte para o formato do cliente imediatamente antes do método invocado pelo UMO, enquanto um transformador *outbound* fornece mensagens de transformação do formato do cliente para o formato do transporte e situa-se entre o *router* da mensagem de *outbound* e o *endpoint* da mensagem.

As mensagens podem também ser interceptadas, a fim de fornecer serviços adicionais, tais como *logging*, recolha de estatísticas, etc. A figura 3.13 ilustra o fluxo de dados entre *inbound* e *outbound endpoints* como apresentado no Mule Programmers Guide.

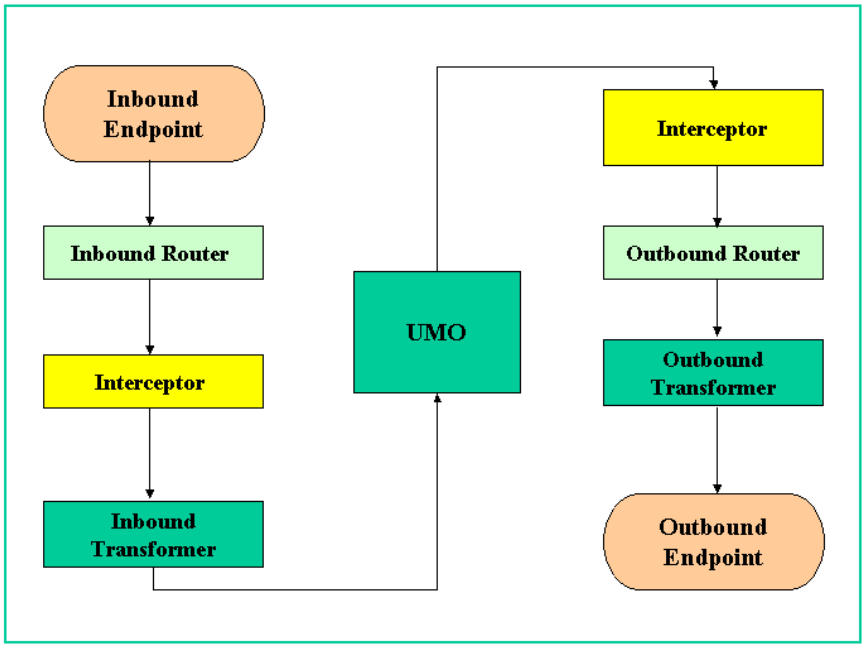

Figura 3.13: Fluxo da mensagem desde o *inbound endpoint* até ao *outbound endpoint* [15]

Aqui fica um pequeno sumário dos conceitos do Mule:

- Universal Messaging Object (UMO)

  Event-driven client software component que recebe e processa a mensagem;

- *Inbound transformer*

  Componente que fornece mensagens de transformação do formato de transporte para o formato do cliente imediatamente antes do método invocado pelo UMO;

- *Outbound transformer*

  Componente que fornece mensagens de transformação do formato do cliente para o formato do transporte e situa-se entre o *router* da mensagem de *outbound* e o *endpoint* da mensagem;

- *Endpoint*

  Segundo o Mule Programmers Guide, este é um "objecto de configuração que define como os dados serão recebidos e transformado pelo Mule. No *endpoint*, é possível configurar o endereço *endpoint*, informação de configuração específica ao transporte, transacções, e filtros";

- *Inbound router*

Componente que controla o fluxo da mensagem na direcção *endpoint-to-UMO*;

- *Outbound router*

  Componente que controla o fluxo da mensagem na direcção *UMO-to-endpoint*; pode incluir vários tipos de filtros para testar se a mensagem pode ser enviado para um dado *endpoint*;

- *Interceptor*

  Fornece serviços adicionais, tais como *logging*, recolha de estatísticas, etc.;

- *Connector*

  Segundo o Mule Architecture Guide, "o conector permite a implementação de ligação ao sistema externo";

Capítulo 4

Especificações para a Interligação do SO IPBrick com o SugarCRM

Para possibilitar a integração dos vários serviços de modo a efectuar o registo, será necessário que os utilizadores e os contactos da IPBrick existam no SugarCRM para fazer a interligação entre eles. Para evitar que o administrador de sistemas tenha que copiar todos os utilizadores e contactos da IPBrick para o SugarCRM, e consequentemente retirar valor à integração, será necessário adicionar à página de administração do SugarCRM uma interface que importe os utilizadores e contactos. Posteriormente faz-se a integração dos serviços.

4.1 Importação dos Utilizadores e Contactos

É no LDAP da IPBrick que se centraliza toda a informação dos utilizadores e dos contactos, como tal, o SugarCRM precisa de ter acesso a este para extrair os dados necessários. Assim sendo, é criado um utilizador de leitura no LDAP.

Na área de administração do SugarCRM serão realizadas ligeiras alterações à interface. Estas alterações implicam o acréscimo de duas páginas *web* para a importação. Será feita uma para os utilizadores e outra para os contactos. No que respeita aos utilizadores, o SugarCRM já tem uma página própria de administração que lhe permite várias acções. Nesta página irá ser criada uma entrada para a página de importação dos utilizadores da IPBrick. A entrada para a página que permite importar os contactos, será criada na página principal de administração.

O funcionamento das páginas será este: quando estas forem solicitadas, é efectuada a leitura do LDAP da IPBrick, compara-se com o que já existe na base de dados do SugarCRM e acrescenta-se os não existentes ou actualizam-se os existentes.

As páginas *web* são desenvolvidas em PHP. Estas contêm funções específicas para o objectivo final da importação. Nas subsecções seguintes são apresentadas as configurações necessárias.

4.1.1 Utilizadores

Na interface de administração do SugarCRM já existe uma página para gerir utilizadores.

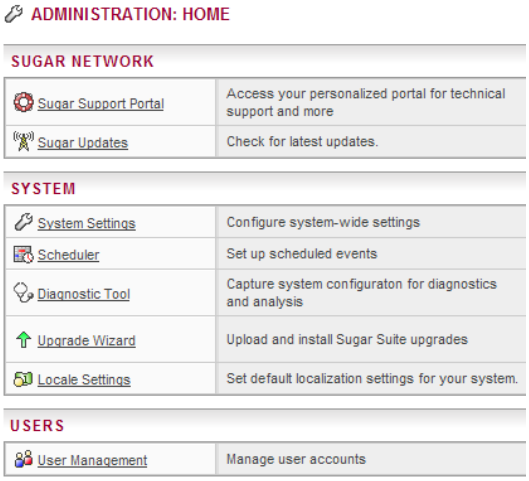

Figura 4.1: Interface de Administração do SugarCRM

Será na última entrada visível da imagem (*User Management*) que se irá criar a entrada para migrar os utilizadores da IPBrick .

De um modo geral, não há necessidade de todos os utilizadores de uma empresa terem acesso ao SugarCRM visto que este é vocacionado para a gestão comercial. Assim sendo, deverá ser dado ao administrador do sistema a opção de quais os utilizadores a que quer dar acesso ao SugarCRM, sejam eles da parte administrativa, comercial ou de outro departamento.

Partindo deste princípio, a interface de migração dos utilizadores deverá listar os utilizadores do SugarCRM e da IPBrick simultaneamente. Isto porque possibilitará ao administrador ver de forma fácil e perceptível quais os que já foram importados para o SugarCRM e os que quer adicionar/remover.

A forma de listar os utilizadores do SugarCRM é efectuando uma *query* à sua base de dados, com o seguinte formato:

$$select\ \ user\_name\ \ from\ \ sugarcrm.users\ \ where\ \ status = \text{"}Active\text{"}\ \ and\ \ deleted = 0$$

onde *user_name* retornará os utilizadores (activos e não-apagados).

Para listar os utilizadores da IPBrick , será necessário fazer um *bind* ao LDAP, filtrar os utilizadores e respectivos dados (*username*, nome e *email*):

$$bind\ \ ldap\ \ server;$$

filter users;
get data;

Adicionalmente, esta interface também deverá fornecer a possibilidade de retirar um utilizador do SugarCRM que tenha sido migrado, para o caso de já não necessitar de ter acesso. No caso de o utilizador ter sido apagado da IPBrick, um *script* de reflexão das alterações na IPBrick (`/opt/apps-scripts.d/script-sugarcrm.php`) irá certificar-se que ele é removido (*deleted = 1*) no SugarCRM.

A existência de um utilizador na tabela *users* indica que o utilizador já foi instanciado no SugarCRM. O seu acesso é ditado pela combinação de:

$$password\ valid\ and\ status = \text{"}Active\text{"}\ and\ deleted = 1$$

Quando o utilizador é migrado, a autenticação no SugarCRM será efectuada no LDAP da IPBrick. Assim o utilizador não necessitará ter mais um *username* e/ou de memorizar mais uma *password*, já que será sempre o que utiliza em todos os serviços centralizados na IPBrick.

4.1.2 Contactos

Ao contrário dos utilizadores, os contactos são todos importados sem dar a opção de escolher ao administrador quais quer importar. Isto porque todos os contactos deverão estar disponíveis para todos os utilizadores.

A nova opção na página de administração do SugarCRM para importar os contactos da IPBrick, será simples já que não fornecerá nenhuma opção extra ao administrador. Conterá um botão que ao premir irá fazer a importação.

O desenvolvimento será semelhante à dos utilizadores. Fazer um *dump* do LDAP da IPBrick e extrai-se a informação necessária. A diferença é que aqui temos as entidades e os contactos, sendo este último associado às entidades, ou seja, uma entidade poderá ter vários contactos. Esta associação terá que ser preservada. As entidades da IPBrick serão *Accounts* no SugarCRM e os contactos de uma entidade serão *Contacts*.
De um modo geral:

bind ldap server;
filter entidades;
filter contactos;
load SugarCRM data;
compare LDAP Entidade entry with SugarCRM Account entries;
insert if not in SugarCRM;
update if in SugarCRM;
compare Account SugarCRM entry with LDAP Entidade entries;
delete if not in LDAP;

Com a informação necessária dos utilizadores e contactos no SugarCRM, podemos passar à integração dos serviços de maneira a registar automaticamente todos os contactos efectuados com o cliente.

4.2 Integração de Serviços

Pretende-se que a integração seja o mais simples e o mais idêntica possível em todos os serviços.

De uma maneira geral cada serviço terá o seguinte processo de integração:

1. Cada serviço tem um *trigger* na IPBrick. Este insere na base de dados uma tabela temporária com os dados associados ao serviço;

2. Existe um processo no sistema que de x em x tempo verifica se existem dados na tabela temporária. Se existir, coloca estes dados no barramento de comunicações (ESB);

3. O ESB recebe esses dados e coloca-os na base de dados do SugarCRM;

Com os serviços de comunicação todos centralizados num ponto, a IPBrick, está implícito que a certa altura ter-se-á todos os dados precisos para o registo no SugarCRM. Cada um já tem o seu registo e tratamento na IPBrick, mas pode conter informação irrelevante para o que nos é pretendido. Será nesta altura, no registo dos dados na IPBrick, que é criado o *trigger*, que irá criar na base de dados uma tabela temporária com os dados por nós seleccionados. Este terá este formato:

$$
\begin{aligned}
&CREATE\ \ TABLE\ \ tabela\_temporaria(dado1, dado2, dado3, dado4, ...);\\
&CREATE\ \ TRIGGER\ \ nome\ \ AFTER\ \ INSERT\ \ ON\ \ tabela\ \ FOR\ \ EACH\ \ ROW;\\
&BEGIN\\
&IF\ \ NEW.disposition\ \ like\ \ '\%'\ \ THEN\ \ INSERT\ \ INTO\ \ tabela\_temporaria\\
&(dado1, dado2, dado3, ...)\ \ VALUES(NEW.dado1, NEW.dado2, NEW.dado3, ...);\\
&END\ \ IF;\\
&END
\end{aligned}
$$

Com os dados recolhidos, é necessário colocá-los no barramento de comunicações.

Para tal, cria-se um *script* que acede à tabela temporária da base de dados e coloca os dados no barramento. Depois de colocados, apaga-os da tabela temporária visto já não serem precisos. Este *script* é desenvolvido em PHP e contém funções especificas para o objectivo. O método utilizado para colocar os dados no Mule será por HTTP POST, através da função de PHP, *cURL*:

$$
\begin{aligned}
&...\\
&dados = @mysql\_fetch\_object(resultado);\\
&ch = curl\_init();
\end{aligned}
$$

$curl\_setopt(ch, CURLOPT\_URL, 'http://servidor:8765/ipb\_servico');$
$curl\_setopt(ch, CURLOPT\_POST, 1);$
$curl\_setopt(ch, CURLOPT\_POSTFIELDS, 'method = ipb\_servico\&parametros');$
$curl\_setopt(ch, CURLOPT\_RETURNTRANSFER, 1);$
$curl\_exec(ch);$
$curl\_close(ch);$
$DELETE \ \ FROM \ \ tabela;$
...

O Mule terá um ficheiro de configuração que "diz"à aplicação que o que entrar em determinado conector, HTTP POST no nosso caso, tem que ser "atendido"por um determinado objecto (uma classe, basicamente).

No conector é preciso separar o HTTP POST para determinar o que se pretende registar (Chamadas telefónicas, FAX, email ou *Instant messaging*) e os parâmetros de cada um. Esta distinção irá instanciar os respectivos objectos para lidar com cada serviço, já que cada registo tem as suas especificidades a nível dos parâmetros, e é tratado com objectos específicos. Para cada um desses métodos, usa-se a API SOAP do SugarCRM para proceder ao registo, de maneira a respeitar as tarefas inerentes ao registo, geridas pelo motor do SugarCRM (disparo de *triggers*, respeito ao *workflow*, notificações, etc.).

Os quatro serviços que se pretende integrar serão especificados com mais pormenor nas secções seguintes, em que cada secção representa um serviço.

O primeiro serviço a ser especificado é o contacto telefónico. Este é o mais frequente e muitas vezes o primeiro passo para angariar um novo cliente.

4.2.1 Chamada Telefónica

Todas as chamadas passam pelo PBX da IPBrick.GT. Quando se realiza ou se recebe uma chamada, já existe neste uma *macro* que pesquisa no LDAP da IPBrick o nome associado ao número. Se o nome existir nos contactos, o número é substituído pelo nome, caso contrário, mantém-se o número.

Para se definir correctamente uma comunicação telefónica com um contacto e ter informação suficiente para se registar, quatro parâmetros são suficientes. São eles: o número de origem, o número de destino, a duração e direcção da chamada.

O número de destino e origem por óbvias razões. A duração para aproveitamento estatístico, por exemplo, e a direcção da chamada para se saber quem ligou a quem, ou seja, se a direcção é *in* significa que o cliente contactou-nos, se for *out* a chamada foi originada por "nós".

O registo para o ESB chamar-se-á *ipb_regcall* e com os parâmetros:

- *snumber* - número de origem (chamador)
- *dnumber* - número de destino

- *duration* - duração da chamada (segundos)
- *direction* - *in*/*out*

Estes quatro parâmetros são os que irão ser colocados pelo *trigger* na tabela temporária da base de dados, usando a informação fornecida pelo PBX da IPBrick.

O código HTTP POST para o registo da chamada telefónica, que o processo irá ter para colocar no ESB é:

```
...
curl_setopt(ch,CURLOPT_URL,'http://servidor:8765/ipb_regcall');
curl_setopt(ch,CURLOPT_POST,1);
curl_setopt(ch,CURLOPT_POSTFIELDS,'method=ipb_regcall&snumber=xxxxxx&dnumber=
xxxxxxxxx&direction=xxxxx&duration=xxxxx');
curl_setopt(ch,CURLOPT_RETURNTRANSFER,1);
...
```

O HTTP POST, após processado e encaminhado dentro do Mule, será transformado numa chamada SOAP que irá instanciar um objecto do tipo *Call* no SugarCRM, com os parâmetros passados e estado *Held* (chamada efectuada – há mais estados: Planned, Cancelled, etc.).

4.2.2 FAX

Este tem algumas semelhanças com o anterior já que também passa pelo PBX da IPBrick. Isto diz-nos que o método de obter o número de origem ou de destino é semelhante. A diferença é que não há necessidade de se registar os dois números de FAX, o de origem e de destino. Isto porque, ao contrário dos telefones, geralmente o número de FAX está associado a uma entidade e não a um contacto. Identificando o sentido do FAX, se é de saída ou de entrada, o número que se regista é o de destino ou o de origem respectivamente.

Estes são os parâmetros que irão ser colocados pelo *trigger* na tabela temporária da base de dados, usando a informação fornecida pelo PBX da IPBrick .

O registo para o ESB chamar-se-á *ipb_regfax* com os parâmetros:

- *snumber* - número do interlocutor (se a direcção for *in* é o número chamador, caso contrário é o destinatário)
- *direction* - (*in*/*out*)

O código do HTTP POST para este registo será:

```
...
curl_setopt(ch,CURLOPT_URL,'http://servidor:8765/ipb_regfax');
curl_setopt(ch,CURLOPT_POST,1);
```

curl_setopt(*ch*,*CURLOPT_POSTFIELDS*,'*method* = *ipb_regfax*&*snumber* = *xxxxxx*&*direction* = *xxxxx*');

curl_setopt(*ch*,*CURLOPT_RETURNTRANSFER*,1);

...

O HTTP POST, após processado e encaminhado dentro do Mule, será transformado numa chamada SOAP que irá instanciar um objecto do tipo *Call* no SugarCRM, com os parâmetros passados, estado *Held*, e *Type* = *Fax*.

4.2.3 Correio Electrónico

Para registar o correio electrónico usam-se quatro parâmetros: o endereço de origem, de destino, assunto e conteúdo. Com estes conseguir-se-á ter toda a informação necessária para um correcto registo.

Estes são os parâmetros que irão ser colocados pelo *trigger* na tabela temporária da base de dados, usando a informação retirado pelo serviço de correio electrónico da IPBrick.

O registo para o ESB chamar-se-á *ipb_regemail* e passará os parâmetros:

- *saddresses* - endereço de origem
- *daddresses* - endereços do destinatário
- *subject* - assunto
- *content* - conteúdo do email

O código HTTP POST do processo para este registo será:

...

curl_setopt(*ch*,*CURLOPT_URL*,'*http*://*servidor*:8765/*ipb_regemail*');

curl_setopt(*ch*,*CURLOPT_POST*,1);

curl_setopt(*ch*,*CURLOPT_POSTFIELDS*,'*method* = *ipb_regemail*&*saddresses* = *xxxxx*@*yyyy*&*daddre*
xxxxxxxx@*yyyy*[,*xxxxxxxx*@*yyyy*]&*subject* = *xxxxx*&*content* = *xxxxx*');

curl_setopt(*ch*,*CURLOPT_RETURNTRANSFER*,1);

...

O HTTP POST, após processado e encaminhado dentro do Mule, será transformado numa chamada SOAP que irá instanciar um objecto do tipo *Email* no SugarCRM, com os eventuais anexos processados (isto implicará passar em *content* o conteúdo em bruto (leia-se: MIME *multipart*) da mensagem) e introduzidos como *Attachments* do *Email*.

4.2.4 *Instant Messaging*

Para este serviço serão utilizados quatro parâmetros: *peers*, *start*, *duration* e *log*. O primeiro serve para registar os intervenientes, o segundo a hora de início da conversa, o terceiro e quarto registam respectivamente a duração e as mensagens da conversa.

O registo terá o nome *ipb_regim* com os parâmetros:

- *peers* - interlocutores (separados por vírgula)
- *start* - hora de início da conversação
- *duration* - duração da conversação (segundos)
- *log* - registo de toda a conversação

O código HTTP POST do processo para este registo será:

...
$curl\_setopt(ch, CURLOPT\_URL, 'http://servidor:8765/ipb\_regim');$
$curl\_setopt(ch, CURLOPT\_POST, 1);$
$curl\_setopt(ch, CURLOPT\_POSTFIELDS, 'method = ipb\_regim\&peers = xxxx; yyyy\&start = xxxxx\&duration = xxxxx\&log = xxxxx');$
$curl\_setopt(ch, CURLOPT\_RETURNTRANSFER, 1);$
...

O HTTP POST, após processado e encaminhado dentro do Mule, será transformado numa chamada SOAP que irá instanciar um objecto do tipo *Meeting* no SugarCRM, com *Type = Instant Message*, e o registo da conversação incluído no campo de descrição ou como *Note/Attachment*.

4.3 Mule

O Mule comporta-se como um *Enterprise Service Bus* , inerente aos *application servers* da actualidade.

De modo a que o Mule saiba o que fazer com as mensagens que lhe são entregue é criado um ficheiro de configuração. Ao inicializar o Mule com este ficheiro, é-lhe dito que o que entrar em determinado conector tem que ser "atendido"por um determinado objecto.

Neste caso, muito embora se utilizem quatro "serviços" (ipbreg_*), apenas se usará um conector HTTP, um transformador (para transformar o pedido HTTP num objecto) e um único objecto de entrada, *HistoryItem*. A mensagem HTTP dá entrada no ESB como *HistoryItem*. Este objecto é a mensagem do ESB propriamente dita que, neste caso em particular, está dotado para se auto-processar. Este objecto é capaz de se auto-validar, mediante o tipo de registo que se pretende (número de parâmetros correcto para o registo em questão, validação dos mesmos, etc.). O Mule

utiliza um objecto central (UMO) para processar a mensagem. Neste caso, o UMO certificar-se-á que o *HistoryItem* é válido e solicita a chamada SOAP específica para o tipo de registo, dos cenários expressos nas secções anteriores (Chamada Telefónica, FAX, Correio Electrónico, *Instant Messaging*).

Os objecto são desenvolvidos em "Plain Old Java Object" (POJO). O transformador *HttpRequestToHistoryItem* é responsável por desmembrar o HTTP POST num objecto *HistoryItem*, e este, por sua vez, irá processar os parâmetros passados: tipo de registo (ipb_regcall, ipb_regfax, ipb_regmail, ipb_regim), snumber (se aplicável), dnumber (se aplicável), etc.

Sendo que o *HistoryItem* é a mensagem circulante, eventualmente será entregue no UMO. Este objecto será responsável pelas validações, controlo de estado do processo, e respectiva chamada SOAP inerente ao tipo de registo que se solicita, conforme se ilustra na imagem seguinte:

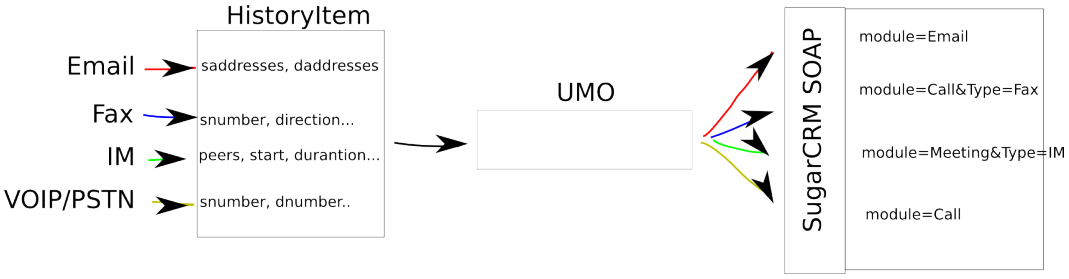

Figura 4.2: Comunicação IPBrick - Mule - SugarCRM

Relembrando também que o UMO apenas processa a mensagem – isto é, ela sai do UMO mantendo o tipo – será necessário um novo transformador para podê-la transformá-la num código de estado da operação. A transformação consiste apenas em transformar as variáveis de estado internas do *HistoryItem* numa *string*, para ser processada pelos *scripts* que invocaram quando se ligaram ao conector.

O resultado será sempre 3 linhas:

- código
- mensagem
- extra

4.4 Instalação

Todo o *software* usado nesta integração necessita de ser instalado na IPBrick. Esta possui na sua interface de gestão um menu que permite inserir *updates*, tanto para actualizações, correcções ou *software*. E sendo a IPBrick um sistema tão *user friendly*, esta integração também terá que ser fácil de instalar.

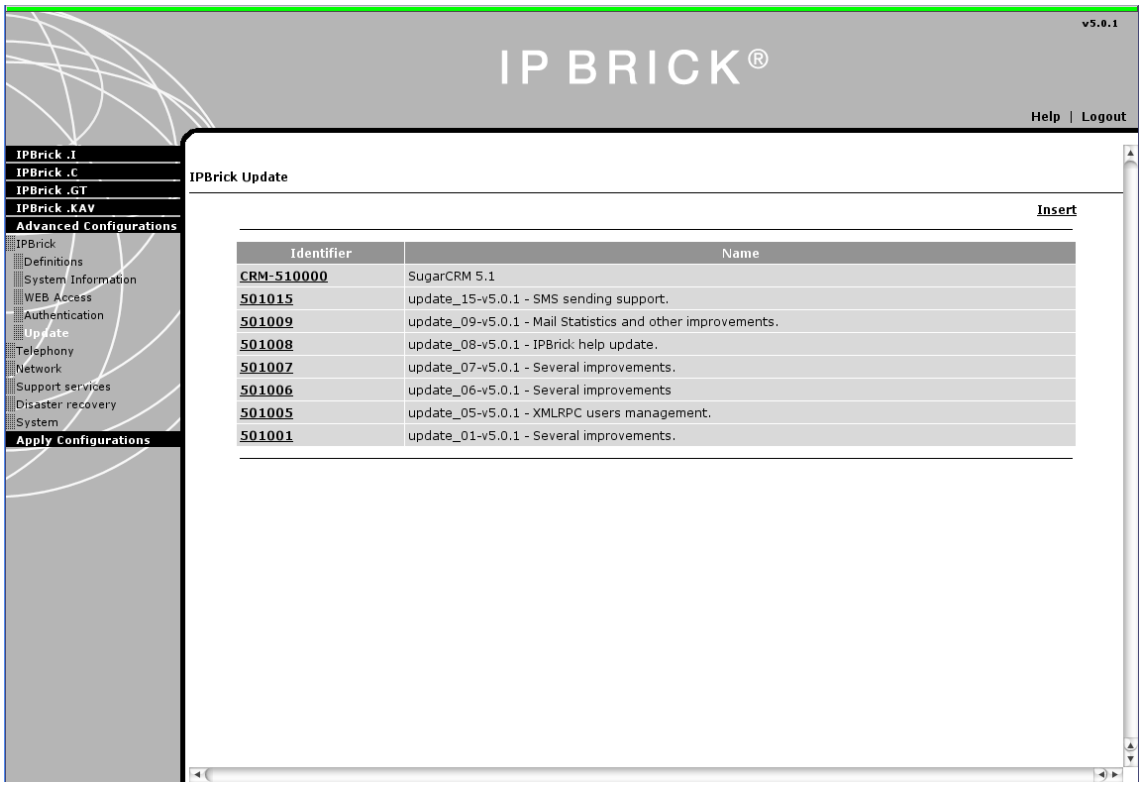

Figura 4.3: Página na IPBrick para inserir *updates*

Serão criados pacotes de instalação para o SugarCRM, Mule e outro para o ficheiro de configuração do Mule e os *scripts*. Porquê esta separação e não apenas um só pacote com todo eles já que a integração depende de todos? A resposta é intuitiva. Se todos estiverem no mesmo pacote, quando se precisar de actualizar apenas um deles, será necessário refazer o pacote com todos. Se forem individuais não há esse trabalho, refaz-se apenas o que necessita.

Estes pacotes têm a estrutura de ficheiros e directórios de um pacote Debian. Os ficheiros de instalação inserem na IPBrick toda a informação necessária para que assim que acabe, o *software* esteja disponível e pré-configurado. No caso do SugarCRM, insere-se no serviço Web Server e DNS de modo a criar o *website* do SugarCRM. Por exemplo, se a IPBrick tiver o dominio *petrus.pt*, o endereço será *sugarcrm.petrus.pt*.

Capítulo 5

Implementação da Interligação do SO IPBrick com o SugarCRM

Neste capítulo apresenta-se o resultado das especificações e de várias topologias de utilização. Apresentam-se possíveis casos onde a implementação da solução será de grande utilidade.

5.1 Topologia

A utilização de um *Enterprise Service Bus* e a facilidade de integração da IPBrick permitem adoptar vários cenários de implementação. É possível cenários em que os sistemas IPBrick e o SugarCRM estão na mesma máquina, por exemplo numa IPBrick.GT, ou em máquinas separadas, o UCoIP numa IPBrick.GT e o SugarCRM numa IPBrick.I e não necessáriamente na mesma rede.

Possivelmente o cenário mais comum é o apresentado na figura 5.1, em que ambos os sistemas partilham a mesma máquina, a IPBrick.GT.

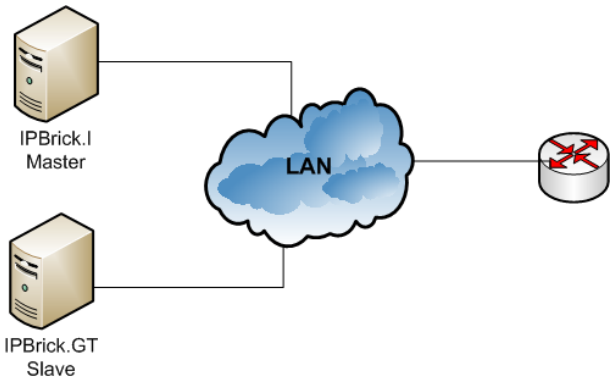

Figura 5.1: Cenário em que a máquina é partilhada pelos dois sistemas

Com todos os sistemas a correr na mesma máquina, implica que esta terá que possuir um *hardware* com mais capacidade para melhor suportar os recursos necessários da utilização dos

sistemas. Num cenário em que estejam em máquinas separadas, há um melhor balanceamento de carga e a comunicação também ocorre sem problemas. 5.2

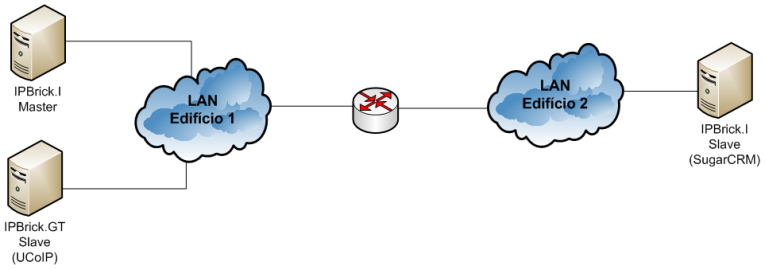

Figura 5.2: Cenário em que os dois sistemas estão separados

Para os administradores de sistemas, a possibilidade de esta implementação permitir várias topologias é uma mais valia. É possível implementar esta integração sem necessidade de alterar uma arquitectura já existente.

5.2 Inicialização

Na interface da IPBrick insere-se os pacotes de instalação do *software*. Após a instalação dos pacotes, poder-se-á aceder de imediato à interface do SugarCRM. Na figura 5.3, mostra-se a página de entrada. O utilizador *administrator* da IPBrick é automaticamente importado na instalação do pacote com as mesmas permissões do administrador do *software* SugarCRM.

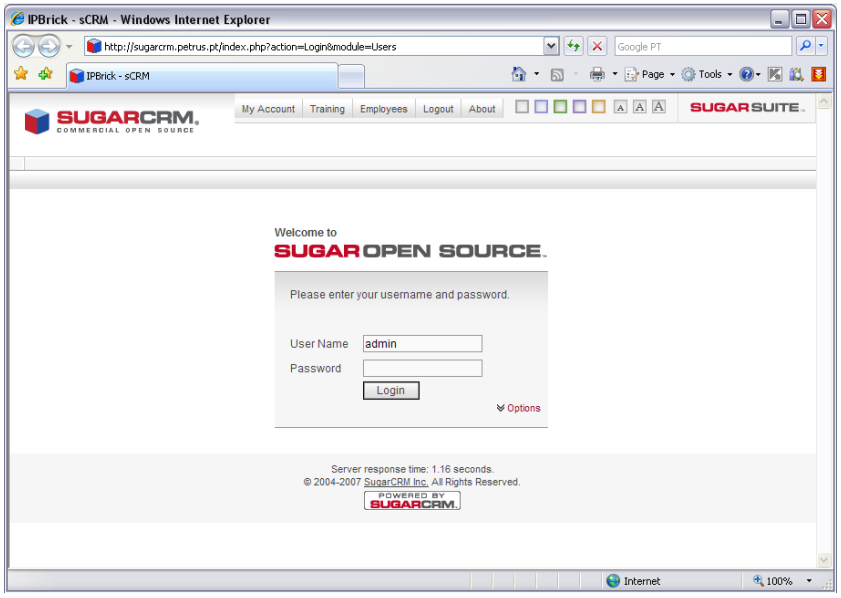

Figura 5.3: Interface de Login do SugarCRM

O passo seguinte será importar os utilizadores da IPBrick e os contactos do IPContactos.

Na página de administração do SugarCRM utiliza-se as páginas criadas para o efeito.

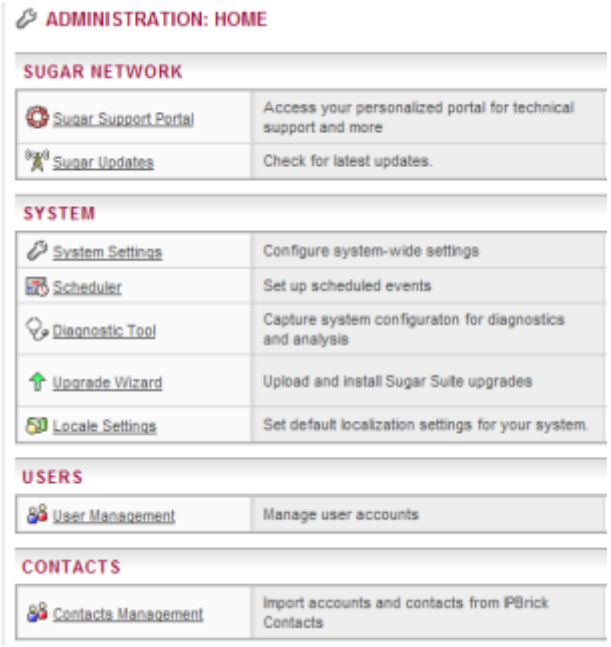

Figura 5.4: Interface de Administração do SugarCRM

5.2.1 Utilizadores

Na página de administração do SugarCRM , no *link USERS* foi adicionado o botão "`Get From LDAP`"para aceder a interface de importação dos utilizadores da IPBrick para o SugarCRM.

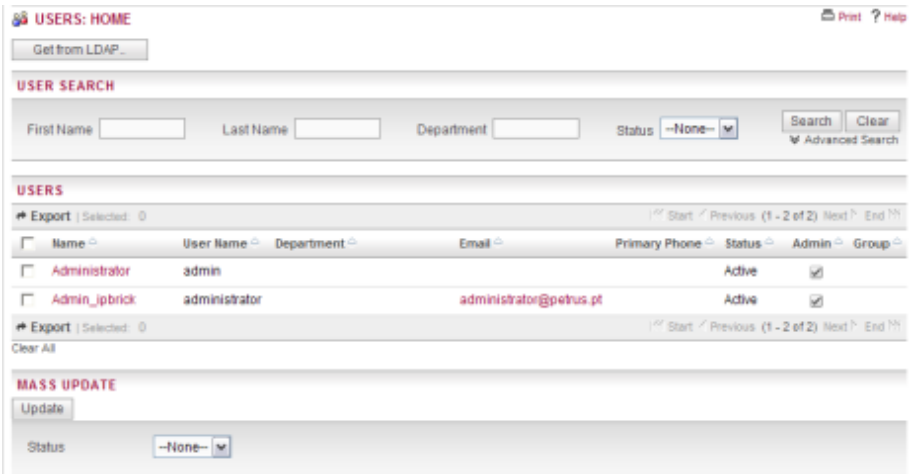

Figura 5.5: Gestão dos Utilizadores do SugarCRM

Ao clicar neste botão, tem-se acesso à página de importação que foi criada para importar os utilizadores da IPBrick :

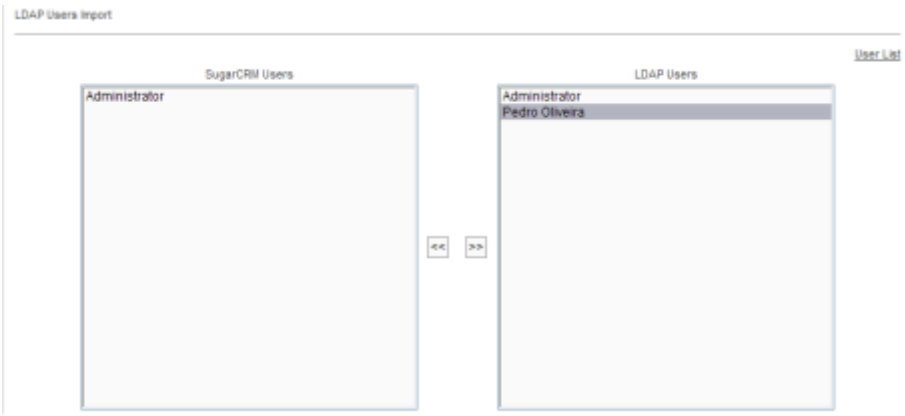

Figura 5.6: Interface de Importação dos Utilizadores

São apresentadas duas *select box*. Na *select box* do lado esquerdo estão os utilizadores da IPBrick que já foram importados e têm acesso ao SugarCRM. Como o utilizador Administrator é importado aquando da instalação, já aparece nesta *select box*. Na do lado direito são listados os utilizadores da IPBrick. Nesta *select box* seleccionam-se os utilizadores que se pretende dar acesso ao SugarCRM e clica-se no botão "«"presente no meio das duas *select box*. Do mesmo modo, para retirar o acesso a algum utilizador, selecciona-se na *select box* do lado esquerdo e clica-se no botão "»".

A partir do momento em que o administrador adiciona um utilizador da IPBrick à *select box* do lado esquerdo, este poderá aceder de imediato ao SugarCRM com o mesmo *username* e *password* que já estão habituados a usar em todos os sistemas que a IPBrick é responsável.

5.2.2 Contactos

Para importar os contactos do IPContactos da IPBrick , a interface apresentada é esta:

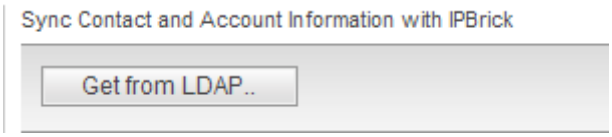

Figura 5.7: Interface de Importação dos Contactos

Ao clicar no botão "`Get from LDAP..`", na primeira importação as Entidades e os respectivos Contactos são criados no SugarCRM. Em importações posteriores, é verificado se alguma Entidade ou Contacto foi alterado ou apagado. Em caso positivo, essa alteração é realizada no SugarCRM.

Estas são as únicas acções que o administrador precisa de efectuar para a integração de serviços. Tudo o resto é transparente, sem necessidade de qualquer intervenção.

5.3 Casos de estudo

Uma empresa que use o SugarCRM e/ou a IPBrick sabe as potencialidades, facilidade de utilização e excelente gestão que ambos os sistemas proporcionam há sua empresa na sua respectiva área.

Imagenemos um cenário de pequena dimensão, uma PME, com ambos os sistemas. A IPBrick como servidor do seu sistema informático e o SugarCRM para uso na parte comercial. Parte do uso normal por parte dos utilizadores no SugarCRM é registar novas oportunidades de negócio. O procedimento normal por parte do comercial será entrar em contacto com o possível cliente, por telefone ou correio electrónico, geralmente estes são os meios principais, e registar esses contactos no SugarCRM. Em resposta a este contacto inicial, geralmente sucede-se novos contactos para troca de informação. Por cada um destes contactos, novos registos no SugarCRM. Em caso de troca de e-mails, será possível organizar toda essa informação por parte do utilizador, mas com mais perda de tempo e falta de produção para a empresa.

Com a implementação desta integração, os contactos telefónicos e troca de e-mails ficarão todos registados automaticamente e associados ao possível cliente sem nenhuma interacção por parte do utilizador. O mesmo se passará caso seja enviado e/ou recebido algum fax. Outra grande vantagem é a organização automática. Não será necessário por parte do utilizador andar a organizar todos os e-mails que trocou com o cliente. Acede ao registo do possível cliente no SugarCRM e verá todos os e-mails trocados com ele. Isto implica mais tempo de trabalho e melhor produção para a empresa.

Outro cenário onde esta integração é de grande utilidade é um *call center*.

Geralmente um *call center* está associado a um grande número de clientes, como exemplos, um operador de telecomunicações ou de um ISP *Internet Service Provider*. Daqui resulta um grande volume de contactos, iniciados pelo cliente no caso de problemas, dúvidas ou informações, ou por parte do operador para dar conhecimento de novos produtos aos clientes. Por estas razões apresentadas e por outras, o número de contactos efectuados são elevados.

Com o uso desta implementação, ao aceder à ficha do cliente no SugarCRM , será visível todos os anteriores contactos com o cliente, como por exemplo, as chamadas telefónicas e troca de e-mails. Sabendo o histórico, poderá ser possível ter uma ideia da razão pela qual o cliente liga, por um novo problema ou até no seguimento dos contactos efectuados para celebrar um novo contrato.

A organização que daqui provém também é notória. Não será necessário perder tempo a arquivar e organizar os contactos dos clientes. Resulta em mais tempo de produção para o empregado e empregador.

Capítulo 6

Conclusões e Trabalho Futuro

Neste projecto apresentaram-se dois sistemas de grande importância e utilidade numa empresa nos dias de hoje.

Numa altura em que sistemas baseados em Linux se tornam mais comuns, mas que de modo geral necessitam de mão-de-obra cara e especializada, mostrou-se a facilidade de utilização e configuração de um sistema tão completo e customizável como a IPBrick. Este possui todos os requisitos que uma empresa necessita como servidor central de toda a sua infra-estrutura e comunicação.

Uma boa gestão e organização resulta num melhor funcionamento global de uma empresa. Com uma completa cobertura de todas as necessidades que uma área comercial e de gestão possa requer, o SugarCRM ajuda a conseguir esses resultados.

6.1 Satisfação dos Objectivos

Os objectivos propostos foram atingidos. A integração dos dois sistemas foi conseguida e ainda mais do que isso, tornou-se a comunicação entre eles independente da localização física com o uso de uma aplicação de transporte de mensagens. É possível para as empresas que usem esta solução tirar mais partido dos sistemas com a vantagem de os utilizadores não ocuparem tempo com o registo dos contactos o que traduz numa melhor utilização dos recursos humanos.

6.2 Trabalho Futuro

Neste momento a integração é unidireccional, isto é, a importação de utilizadores e contactos só é efectuada no sentido IPBrick - SugarCRM. De futuro deverá fazer-se nos dois sentidos, ou seja, importar os utilizadores e contactos da IPBrick para o SugarCRM e possibilitar a criação e importação dos utilizadores e contactos do SugarCRM para a IPBrick. Sendo possível nos dois

sentidos, retira a obrigatoriedade de criar sempre as Entidades e Contactos apenas num dos dois sistemas. Estes são criados no sistema de preferência e em seguida sincronizado com o outro.

Em estudo está a funcionalidade de o utilizador iniciar uma chamada telefónica para o cliente ao visualizar a sua ficha no SugarCRM. Para tal adiciona-se ao perfil de utilizador a possibilidade de inserir a sua extensão telefónica e à ficha do cliente é adicionado um botão para iniciar a chamada. Ao clicar no botão, o SugarCRM irá buscar os números de telefone do utilizador e do contacto e faz o pedido à IPBrick. Esta inicia a chamada entre eles de forma automática. A facilidade de clicar-para-chamar na ficha do cliente, induz o utilizador a introduzir o que ficou apurado nessa chamada. Outros pontos em estudo é a possibilidade de ficheiros guardados no SugarCRM serem enviados por FAX directamente para clientes e que através do serviço IM saber se o cliente/contacto se encontra *online* e abrir janelas de diálogo directamente do SugarCRM.

De modo a facilitar os registos e justificar a bidireccionalidade, poder-se-á desenvolver um *software* adicional que quando entre uma chamada no sistema IPBrick, na estação de trabalho do utilizador, abra a respectiva ficha do cliente no sistema SugarCRM para que se possa aceder rapidamente aos contactos anteriores e tomar notas da razão do novo contacto do cliente. Caso o número que esteja a ligar não exista na base de dados, será aberta uma nova ficha para registar os dados do cliente.

Bibliografia

[1] HighCRM. (2008). Customer relationship management HighCRM - The CRM Company. Available: http://www.highcrm.com.br

[2] Wikipédia. (2008). Customer relationship management Wikipédia a enciclopédia livre. Available: http://pt.wikipedia.org/wiki/Customer_relationship_management

[3] Wikipédia. (2008). SugarCRM Wikipédia a enciclopédia livre. Available: http://pt.wikipedia.org/wiki/SugarCRM

[4] Wikipédia. (2008). Open Source Wikipédia a enciclopédia livre. Available: http://pt.wikipedia.org/wiki/Open_source

[5] Wikipédia. (2008). Linux Wikipédia a enciclopédia livre. Available: http://pt.wikipedia.org/wiki/Linux

[6] Wikipédia. (2008). Linux Wikipédia a enciclopédia livre. Available: http://pt.wikipedia.org/wiki/Linux_(kernel)

[7] Wikipédia. (2008). Debian Wikipédia a enciclopédia livre. Available: http://pt.wikipedia.org/wiki/Debian

[8] Dalera. (2006). Customer relationship management DALERA e-business solutions. Available: http://www.dalera.com/cache/imagens/XPQnHCP8s7887Bu21XzJtmwZKU.pdf

[9] FonsecaNET. (2008). SugarCRM FonsecaNET - Consultoria em Sistema e Informática. Available: http://www.fonsecanet.com.br/?dest=soa

[10] Wikipédia. (2008). Enterprise Service Bus Wikipédia a enciclopédia livre. Available: http://pt.wikipedia.org/wiki/Enterprise_Service_Bus

[11] Progress Software Corporation. (2008). Enterprise Service Bus Progress Software Corporation. Available: http://www.sonicsoftware.com/solutions/service_oriented_architecture/enterprise_service_bus/index.ssp

[12] *IPBrick - Manual de Referência*, ed. 4.3, iPortalMais - Serviços de Internet e Redes, Porto, 2008.

[13] iPortalMais. (2008). IPBrick. iPortalMais - Serviços de Internet e Redes. Available: http://www.ipbrick.com

[14] Log. (2008). SugarCRM Log - Open Source Consulting. Available: http://www.log.pt/solucoes/sugarcrm

[15] Sun. (2008). Mule Java.net - Exploring ESB Patterns with Mule. Available: http://today.java.net/pub/a/today/2007/07/31/exploring-esb-patterns-with-mule.html