Faculdade de Engenharia da Universidade do Porto

FEUP

Frontend Web 2.0 para Gestão de RADIUS

Vítor Hugo Leite Antunes

Dissertação realizada no âmbito do
Mestrado Integrado em Engenharia Electrotécnica e de Computadores
Major Telecomunicações

Orientador: Prof. Dr. Manuel Alberto Pereira Ricardo
Co-orientador: Mestre António Pedro Freitas Fortuna dos Santos
Co-orientador: Mestre Gustavo João Alves Marques Carneiro

Julho de 2009

U.PORTO

FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

MIEEC - MESTRADO INTEGRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES 2008/2009

A Dissertação intitulada

"FRONTEND WEB 2.0 PARA GESTÃO DE RADIUS"

foi aprovada em provas realizadas em 16/Julho/2009

o júri

Presidente Professor Doutor João Francisco Cordeiro de Oliveira Barros
Professor Associado do Departamento de Engenharia Electrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

Professor Doutor Francisco Manuel Marques Fontes
Professor Auxiliar Convidado do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro;

Professor Doutor Manuel Alberto Pereira Ricardo
Professor Associado do Departamento de Engenharia Electrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

O autor declara que a presente dissertação (ou relatório de projecto) é da sua exclusiva autoria e foi escrita sem qualquer apoio externo não explicitamente autorizado. Os resultados, ideias, parágrafos, ou outros extractos tomados de ou inspirados em trabalhos de outros autores, e demais referências bibliográficas usadas, são correctamente citados.

Vitor Hugo Leite Antunes

Autor - VÍTOR HUGO LEITE ANTUNES

Faculdade de Engenharia da Universidade do Porto

Resumo

Cada vez mais a Internet se torna num dos bens essenciais da sociedade e, uma vez que não vivemos num mundo ideal em que o acesso a todos os recursos seria livre, o recurso a mecanismos de autenticação e autorização é crucial para o negócio de vários tipos de empresas. Um dos protocolos que é responsável por estes mecanismos é o RADIUS (*Remote Access Dial In User Service*) que, embora tenha sido criado antes do conceito AAA (*Authentication Autorization Accounting*), apresenta um funcionamento bastante similar a este. Este é um protocolo que pode ser usado em vários tipos de controlo de acesso tais como acesso a um Access Point, controlo de acesso a uma VPN (*Virtual Private Network*) ou acesso a um fornecedor de Internet.

No entanto, a configuração de aplicações que implementam o protocolo de RADIUS não é trivial. Surgiu assim a necessidade de criação de aplicações que automatizassem o processo de configuração de utilizadores e respectivas regras de acesso.

O projecto descrito neste documento apresenta como principal objectivo a elaboração de uma aplicação Web 2.0 para gestão de RADIUS numa empresa que se adapte a qualquer dispositivo e a qualquer *browser Web* e que permita a gestão do RADIUS sem requerer um conhecimento vasto do protocolo.

Com a realização do projecto verificou-se que uma implementação cuidada da interface gráfica facilita a adaptação aos mais diversos tipos de terminais. O recurso a um framework que apresenta uma arquitectura MVC (*Model-View-Controller*) permitiu um desenvolvimento mais escalável e mais fácil de aplicações *Web*.

Foram também realizados testes de forma a avaliar a qualidade da aplicação desenvolvida. Os resultados destes testes permitiram concluir que todos os requisitos mínimos da aplicação foram atingidos assim como a adaptabilidade e compatibilidade pretendidas.

Conclui-se também que o recurso a uma aplicação deste tipo permite uma configuração bastante mais simples dos utilizadores do RADIUS assim como das suas regras de acesso. No entanto, ainda não é possível uma total configuração a partir de uma aplicação uma vez que existem certos módulos da configuração que continuam a exigir a sua configuração manual.

Abstract

The constant rise of the use of the Internet leads it to the one of the most essential goods of society. Since we don't live in an ideal world where the access to all things would be free, the use of authentication and authorization mechanisms is crucial to the proper functioning of various types of business enterprises. One of the protocols that is responsible for those mechanisms is RADIS (Remote Access Dial in User Service) that, although it was created before the concept AAA (Authentication Authorization Accounting), has a functioning similar to that. RADIUS is a protocol that can be used in several types of access control such as access to an Access Point, controlling access to a VPN (Virtual Private Network) or access to an Internet provider.

However, the configuration of applications that implement RADIUS protocol is not trivial. Thus raise the need to create applications that automate the process of setting up users and their access rules.

The project described in this document presents as the main objective the development of a Web 2.0 application for managing RADIUS in an enterprise that can be adapted to any device and to any Web browser and allow to manage RADIUS without complete knowing of the RADIUS protocol.

With the realization of the project it was found that a careful implementation of the graphical interface helps the application to adapt to several types of terminals. Using a framework that provides MVC (Model-View-Controller) architecture allows the implementation of more scalable, easier and faster Web applications.

Some tests to the application were also made to evaluate the quality of the developed application. The results of these tests indicated that all minimum features were achieved as well the adaptability and compatibility required.

It is also concluded that the use of such an application allows a much simpler configuration of the RADIUS users and their access rules. However, it is not possible a full configuration from an application like this since there are certain configuration modules that continue to require a manual setup.

Agradecimentos

Apesar do trabalho solitário a que um aluno está destinado num projecto deste tipo existem sempre inúmeras pessoas que nos fornecem o seu contributo tanto ao nível intelectual como ao nível psicológico. Nesta secção pretendo agradecer a essas pessoas sem as quais esta dissertação não seria possível.

Em primeiro lugar gostaria de agradecer a todos os meus orientadores, Professor Doutor Manuel Alberto Pereira Ricardo, Mestre António Pedro Freitas Fortuna dos Santos e Mestre Gustavo João Alves Marques Carneiro, pela ajuda e conhecimentos passados ao longo de todo este projecto.

Gostaria também de agradecer à minha família pelo seu apoio.

Para concluir, gostaria de agradecer à minha namorada por toda o seu apoio, tolerância, carinho e compreensão.

Índice

Lista de figuras

Lista de tabelas

Abreviaturas

AAA	*Authentication, Authorization and Accounting*
ACL	*Access Control List*
ACS	*Access Control Server*
AJAX	*Asynchronous JavaScript and XML*
AP	*Access Point*
API	*Application Programming Interface*
CMS	*Content Management System*
CPU	*Central Processing Unit*
CRUD	*Create, Read, Update and Delete*
CSS	*Cascading Style Sheets*
EAP	*Extensible Authentication Protocol*
HTTP	*Hypertext Transfer Protocol*
IAS	*Internet Authentication Service*
IP	*Internet Protocol*
ISP	*Internet Service Provider*
JavaEE	*Java Enterprise Edition*
JSON	*JavaScript Object Notation*
MAC	*Medium Access Control*
MVC	*Mode-View-Controller*
NAS	*Network Access Server*
O-R	*Object-Relational*
ORM	*Object-Relational Mapping*
OSI	*Open Systems Interconnection*
PC	*Personal Computer*
PDA	*Personal Digital Assistant*
PHP	*Hypertext Preprocessor*
QoS	*Quality of Service*
RADIUS	*Remote Authentication Dial In User Service*
RIA	*Rich Internet Application*
SQL	*Structured Query Language*

TCP	*Transmission Control Protocol*
TKIP	*Temporal Key Integrity Protocol*
TLS	*Transport Layer Security*
UDP	*User Datagram Protocol*
VPN	*Virtual Private Network*
WLAN	*Wireless Local Area Network*
WPA	*Wi-Fi Protected Access*
XHTML	*eXtensible Hypertext Markup Language*
XML	*eXtensible Markup Language*
XSS	*Cross-site scripting*
XSTL	*Xtensible Stylesheet Language Transformations*

Capítulo 1

Introdução

Cada vez mais a Internet se apresenta como um dos bens essenciais da sociedade. Este aumento da dependência da Internet leva a que haja um aumento das entidades que usam o acesso à Internet para o negócio. De modo a que essas empresas possam permitir, ou negar, o acesso dos seus clientes existe a necessidade de centralização de um mecanismo de autenticação, que obrigue o cliente a autenticar-se perante a entidade fornecedora do serviço, e de autorização, que verifique os serviços a que cada cliente está autorizado a aceder.

Num cenário de uma rede privada de uma empresa a existência de um mecanismo de autenticação é também muito importante na medida em que obriga cada utilizador a autenticar-se, protegendo os utilizadores e projectos da empresa de possíveis ataques à rede.

Para a maior parte dos utilizadores esta autenticação é transparente. Estes apenas necessitam de colocar as suas credenciais e, passado algum tempo, estão autenticados. No entanto, após a introdução das credenciais existe um conjunto de processos e protocolos que visam provar que o utilizador é quem diz ser e verificar quais os recursos a que ele está permitido a aceder. Um dos protocolos que é responsável por estes processos é o RADIUS (*Remote Access Dial In User Service*) que, embora tenha sido criado antes do conceito de AAA (*Authentication Autorization Accounting*), apresenta um funcionamento bastante similar a este.

O conceito AAA, detalhado em [2], descreve os processos de autenticação, autorização e contabilização. A autenticação verifica os dados enviados por parte de um determinado utilizador provando que está registado no sistema e se é, ou não, quem este diz ser. A autorização verifica quais os recursos a que o utilizador está autorizado a aceder. A contabilização destina-se à recolha de informação acerca da utilização de recursos de um sistema pelos seus utilizadores, por exemplo, contabilização do tráfego trocado, tempo da sessão, etc.

O protocolo RADIUS, descrito em [3], surgiu quando, em meados de 1992, a *Livingston* (hoje parte da *Lucent Technologies*) começou a desenvolver um protocolo de AAA que deveria viabilizar o acesso remoto a um serviço de rede. O RADIUS surgiu com os principais objectivos

de ser um protocolo seguro, fiável, escalável e expansível. Este é um protocolo que pode ser usado em vários tipos de controlo de acesso tais como acesso a um *Access Point* (AP), controlo de acesso a uma VPN (*Virtual Private Network*) ou acesso a um fornecedor de Internet. Isto é, desde que haja a necessidade de centralização dos processos de autenticação, autorização e contabilização, o RADIUS pode ser utilizado. Actualmente é mesmo fundamental para a grande maioria de fornecedores de Internet.

No entanto, a configuração de aplicações que implementam o protocolo de RADIUS não é trivial implicando que terá de ser feita, obrigatoriamente, por alguém que entenda o protocolo e que tenha acesso ao servidor onde se encontra a aplicação. Com isto é visível a necessidade de aplicações que automatizem o processo de configuração e que sejam acessíveis por qualquer utilizador que esteja permitido a aceder e modificar a configuração do servidor.

Este projecto surgiu com o objectivo de colmatar essa necessidade. Assim, o projecto descrito neste documento apresenta como principal objectivo a elaboração de uma aplicação *Web* 2.0 para gestão de RADIUS. Esta aplicação terá de possuir as melhores práticas de desenvolvimento para uma maior interacção e facilidade por parte do utilizador. Pretende-se também que a visualização da aplicação seja adaptável a qualquer tipo de terminal fornecendo, aos utilizadores responsáveis pela gestão do RADIUS, a possibilidade de aceder à aplicação em qualquer local e em qualquer altura (desde que possua acesso à Internet).

1.1 - Descrição do Problema

Actualmente, o RADIUS é indispensável para fornecedores de acesso à Internet e empresas que possuam redes privadas na medida em que permite um controlo de acessos centralizado baseado no conceito de autenticação, autorização e contabilização. No caso dos fornecedores de acesso à Internet é indispensável o uso do RADIUS uma vez que é imperativo garantir que os clientes se autentiquem a partir do local para o qual contrataram o serviço e verificar quais os serviços que este contratou.

No caso das empresas é importante gerir os utilizadores que estão autorizados a aceder à sua rede privada de modo a garantir a integridade dos seus projectos e segurança dos seus utilizadores (trabalhadores).

No entanto, a configuração das aplicações que implementam o RADIUS não é trivial implicando que os gestores da aplicação teriam de ser altamente especializados em RADIUS e teriam a seu cargo a gestão de milhares de utilizadores. Surge por isso a necessidade de elaborar ferramentas que automatizem a gestão do RADIUS de modo a que uma pessoa que perceba minimamente o protocolo esteja habilitada a gerir utilizadores do sistema, gerir as respectivas regras de acesso a cada recurso e ainda monitorizar os dados relativos à contabilização da utilização de cada recurso.

Sem uma aplicação de gestão de RADIUS, uma empresa que pretendesse ter mais que uma pessoa responsável pela gestão remota do seu serviço teria de fornecer o acesso ao servidor a cada responsável comprometendo a segurança e integridade do servidor. Assim, a aplicação deverá apresentar a possibilidade de poder ser gerida remotamente e de poderem ser delegadas partes da configuração e gestão a diferentes intervenientes. Isto permitiria ainda a distribuição da gestão dos utilizadores e suas regras de acesso dos diversos utilizadores por mais pessoas minimizando a carga por cada responsável do sistema.

Os responsáveis por cada recurso deverão poder controlar todos os aspectos relacionados

com o mesmo. O registo de todos os acessos e a elaboração da respectiva estatística são dois requisitos que deverão ser obedecidos de modo a que cada gestor possa monitorizar os acessos dos utilizadores aos recursos a que estão permitidos. As sessões, o tempo de cada sessão e o tráfego transferido é alguma da informação que deverá ser mostrada na estatística de utilização de cada recurso.

Uma vez que uma pessoa, responsável por um determinado recurso, não se encontra sempre no mesmo local geográfico torna-se imperativo que uma aplicação deste tipo permita ser acessível a partir de qualquer local a qualquer momento. Tendo também em consideração que um gestor de um determinado recurso pode não se fazer acompanhar sempre de um dispositivo de alta capacidade e velocidade de processamento, como um portátil, a aplicação deverá então permitir a sua adaptação automática a dispositivos com ecrã de baixa resolução e o seu funcionamento não deverá requerer grande capacidade de processamento. Isto permitiria que qualquer gestor, desde que possua consigo um dispositivo como um PDA (*Personal Digital Assistant*) ou *Smartphone* com acesso à Internet possa gerir e monitorizar os recursos de que está responsável.

De seguida serão identificados os objectivos do trabalho em função do problema descrito nesta secção.

1.2 - Objectivos

Este trabalho apresenta como principal objectivo o desenvolvimento de uma aplicação *Web* 2.0 para gestão de servidores de RADIUS empresariais cujas principais características sejam a simplicidade de processos e automatização de tarefas que dizem respeito à configuração de utilizadores e regras de acesso.

Pretende-se que a implementação da aplicação confira ao utilizador facilidade de interacção e utilização, apresentando uma interface atractiva e interactiva e que possa ser acedida a partir de qualquer local e através de diferentes tipos de dispositivos (tanto por computadores *desktop* e portáteis como por PDA, *Android*, *iPhone*, etc). Para isto, deverá ter-se em consideração as diferentes capacidades de processamento e resolução de cada dispositivo de modo a que o tempo de resposta em dispositivos com menor capacidade de processamento não seja muito elevado, e de modo a que a aparência respeite o tamanho e resolução do ecrã do dispositivo. Isto é, a aplicação terá de ser desenvolvida num ponto óptimo de funcionamento que coincidirá num ponto de equilíbrio entre a flexibilidade e o consumo de largura de banda.

A aplicação deverá suportar vários tipos de perfis, cada qual com diferentes permissões para desempenhar partes da configuração de modo a permitir funcionalidades mais restritas aos gestores com permissões superiores. Por exemplo, enquanto que o criador de um determinado recurso poderá controlar todos os aspectos relativos a este, um outro utilizador designado pelo criador como administrador do recurso poderá apenas controlar os utilizadores que estão permitidos a aceder ao recurso. Com a permissão básica, um utilizador não poderá desempenhar qualquer parte da configuração do recurso a não ser a observação dos dados do mesmo.

Pretende-se que a aplicação efectue o registo de todos os acessos aos recursos e que efectue as suas estatísticas mostrando, entre outros, o número de sessões, o tempo de cada sessão e o tráfego transferido. Os acessos e modificações realizados na interface deverão ser também guardados de modo a funcionarem como registo para o administrador do sistema.

Após a fase de implementação deverão ser realizados testes de modo a avaliar a qualidade da aplicação desenvolvida. Estes testes deverão permitir a avaliação dos processos realizados a partir da aplicação num servidor RADIUS. Para isto deverá ser implementado um cenário que se baseará num servidor RADIUS ligado a um AP configurado para usar o servidor RADIUS para autenticar os utilizadores que se pretendam conectar a este AP. Deverão ser também efectuados testes que permitam a avaliação e aprovação da adaptabilidade da aplicação a diversos tipos de dispositivos. Para isto a aplicação deverá ser acedida a partir de emuladores de dispositivos como *Android*, PDA e *iPhone*.

De seguida é descrita a organização do documento, fornecendo a descrição do conteúdo dos vários capítulos do restante deste documento.

1.3 - Organização do Documento

O remanescente deste documento apresenta a seguinte estrutura: no Capítulo 2, Estado da Arte, é feita uma análise do estado da arte das diferentes tecnologias usadas no desenvolvimento do trabalho. Será explicado o funcionamento do RADIUS, quais as aplicações que implementam este protocolo e as características da *Web* 2.0 e como esta é importante na evolução dos negócios baseados na Internet. Serão também realizadas análises comparativas de alguns mecanismos e aplicações já existentes de modo a permitir uma maior facilidade na implementação; No Capitulo 3, Análise e Implementação, é feita uma descrição dos vários requisitos identificados para a aplicação assim como a descrição das várias fases de implementação identificadas; no Capítulo 4, Testes e Análise de Resultados, é feita a descrição dos testes a realizar e a análise desses resultados, tendo por objectivo avaliar qualitativamente a aplicação desenvolvida; no Capítulo 5, Conclusão e Trabalho Futuro, é feita a recolha das principais conclusões retiradas de todos os capítulos anteriores e é feita a descrição do trabalho a realizar no futuro para a continuação deste projecto.

Capítulo 2

Estado da Arte

Neste capítulo irá ser feita uma análise do estado da arte das várias tecnologias necessárias para o desenvolvimento do trabalho pretendido. De início, irá ser dada uma descrição do RADIUS assim como uma explicação do seu funcionamento. É crucial perceber o funcionamento do RADIUS de modo a perceber, posteriormente, as alterações efectuadas nos dados do RADIUS para integração da aplicação. Posteriormente, irão ser também identificadas algumas das limitações do RADIUS. Serão ainda descritas algumas das aplicações RADIUS disponíveis no mercado identificando as suas principais características. Destas aplicações a que será descrita com maior pormenor será o freeRADIUS que é a aplicação que será utilizada para o desenvolvimento da aplicação.

Será descrita também a evolução das técnicas de desenvolvimento *Web*, às quais se deu o nome de *Web* 2.0. Irá ser demonstrado um estudo que demonstra a importância que a *Web* 2.0 teve na evolução dos negócios baseados na Internet.

Outros estudos demonstrados neste documento têm como objectivo a avaliação das várias linguagens de programação *Web* existentes de modo a permitir a escolha daquela que será mais adequada ao desenvolvimento do projecto e, posteriormente, permitir a escolha da *framework* para desenvolvimento Web que mais se adequa a este trabalho.

Nesta secção será ainda introduzido o paradigma MVC (*Model-View-Controller*), o qual separa a lógica da aplicação da apresentação da informação, permitindo o desenvolvimento de uma aplicação facilmente adaptável a diferentes tipos de terminais. De seguida, irá ser feita uma análise comparativa da performance de *frameworks* de desenvolvimento *Web* com vista a seleccionar o que mais se adequa ao trabalho. Esta análise é levada a cabo tendo em conta alguns critérios de avaliação derivados dos requisitos da aplicação a desenvolver.

Existem já outras implementações de aplicações *Web* para gestão de RADIUS. No entanto todas apresentam limitações inerentes à sua utilização. Tendo em conta os requisitos funcionais da aplicação identificados também neste capítulo, irão ser descritos os trabalhos identificados apresentando as vantagens e desvantagens de cada um.

2.1 - Remote Authentication Dial In User Service – RADIUS

O RADIUS é um protocolo de controlo de acessos que autentica e autoriza o acesso de utilizadores a redes. Este protocolo usa um cenário de cliente-servidor em que o cliente, designado de NAS (*Network Access Server*) envia pedidos de autenticação dos utilizadores e o servidor responde. As respostas do servidor poderão ser em forma de desafios, de aceitação ou rejeição do utilizador. O RADIUS é muito utilizado por fornecedores de acesso à Internet, bem como por organizações que pretendam um sistema de autenticação centralizado para autenticar o acesso à rede e aos seus recursos. O RADIUS apresenta ainda a vantagem de poder juntar informação referente à troca de dados e sessões de cada utilizador, a que se chama de contabilização.

Embora, segundo [10], o RADIUS tenha sido criado antes do modelo AAA (Autenticação, Autorização, Contabilização) o seu funcionamento é similar aos protocolos baseados nesta norma. O modelo AAA é utilizado para gerir todas as transacções do início ao fim da cada sessão de cada utilizador e descreve os processos de autenticação, autorização e contabilização. Numa rede que utilize o protocolo RADIUS, um utilizador que pretenda utilizar um determinado recurso necessita de se autenticar de forma a permitir a utilização desse recurso. Após autenticação são determinados quais os serviços a que o utilizador autenticado está autorizado e é contabilizado (*accounting*) e gravado o acesso deste no servidor RADIUS de modo a haver um maior controlo das actividades.

2.1.1 - Autenticação

Este processo tem como objectivo verificar as credenciais que uma determinada pessoa, ou máquina, que pretenda aceder a uma rede e permitir, ou não, o acesso a esta. O principal aspecto da autenticação é permitir a criação de uma relação de confiança entre duas entidades, assumindo que ambas as entidades são válidas.

O processo mais comum para autenticação, o qual é familiar para a maior parte das pessoas, é o método da combinação de nome de utilizador com uma palavra passe. O uso de certificados digitais como parte da infra-estrutura de chave pública, que se está a tornar num dos métodos preferidos de autenticação, é outro método de autenticação e que fornece uma maior segurança, uma vez que a distribuição das palavras passe pode ameaçar a segurança de uma rede.

2.1.2 - Autorização

O processo de autorização tem como finalidade a verificação de quais os recursos a que um determinado utilizador se encontra permitido a aceder. Com base no resultado da autenticação, o processo de autorização envolve uma série de regras para determinar qual a natureza do serviço ao qual o acesso é permitido ao utilizador. O utilizador pode aceder a serviços do tipo de atribuição de endereço, rotas, serviços de QoS (*Quality of Service*), gestão de largura de banda, entre outros. Com a finalização dos processos de autenticação e autorização, o utilizador está habilitado a utilizar os recursos da rede a que foi permitido.

2.1.3 - Contabilização

Contabilização é outro processo da norma AAA e que tem como funcionalidade guardar informação dos recursos acedidos por cada utilizador. Entre a informação guardada pode estar a quantidade de tempo que o utilizador utilizou o recurso, a quantidade de dados que foram trocados com o utilizador, o número de sessões do utilizador, entre outras. A contabilização pode ser usada para controlo de autorização, facturação por parte dos fornecedores de Internet, análises de ameaças, utilização de recursos, entre outros.

2.2 - Algumas Características do RADIUS

Em [3] é referido que o RADIUS é um protocolo baseado em UDP (*User Datagram Protocol*), ou seja, não mantém uma ligação activa. O uso de UDP em detrimento do TCP (*Transmission Control Protocol*) deve-se às semelhanças entre o RADIUS e o UDP, entre elas, o requisito que pedidos rejeitados para um servidor primário de autenticação sejam redireccionados para um servidor secundário e, para isto, uma cópia do pedido original deve estar presente acima da camada de transporte do modelo OSI. Segundo [10], existem dois pontos de vista que justificam a não utilização do TCP. Por um lado, o RADIUS não requer detecção de dados perdidos e o utilizador está disposto a esperar vários segundos para que o processo de autenticação termine. Assim, o mecanismo de retransmissão do TCP e a mensagem de *Acknowledgment* não são necessários. De outro lado, o utilizador não pretende esperar vários minutos para que a autenticação termine e, por isso, de acordo com [10], a entrega garantida dos dados por TCP dois minutos depois é desnecessária. A desvantagem de usar UDP é que a criação e gestão dos tempos de retransmissão ficam a cargo das aplicações envolvidas.

[10] esclarece ainda que o RADIUS usa um modelo de segurança *hop-by-hop*, o qual consiste na transmissão dos dados nó a nó usando *store and foward*, e que os servidores RADIUS são servidores que não guardam o estado (*stateless*), isto é, tratam cada pedido como independentes de todos os outros pedidos anteriores.

2.3 - Limitações do RADIUS

Existem várias limitações na utilização do RADIUS:

1) a segurança de uma rede está dependente da configuração usada. Por exemplo, numa configuração que possua vários servidores a funcionar como proxy, todos os nós devem processar e reencaminhar os dados no pedido. Devido ao modelo *hop-by-hop* do RADIUS, isto significa que dados, como certificados e palavras passe, estarão disponíveis em todos os nós o que ameaça a segurança da rede.

2) o facto de os servidores RADIUS processarem cada pedido independentemente e, por isso, não relacionarem os pedidos do mesmo cliente não associando um estado a este complica as soluções de gestão de recursos e sessões dado que não existe uma sessão, isto é, uma conexão activa entre o cliente e o servidor.

3) é descrito em [10] que a experiência demonstrou que, em grandes sistemas, o recurso a um único servidor RADIUS pode sofrer perda de performance e perda de dados devido à falta de mecanismos de controlo de congestionamento.

2.4 - Descrição do Funcionamento do RADIUS

O protocolo RADIUS baseia-se no modelo cliente/servidor, tendo de um lado o cliente ou o NAS (*Network Access Server*) que, embora seja uma *gateway* que controla o acesso à rede, possui uma componente que funciona como cliente, e do outro o servidor. O utilizador, o NAS e o servidor trocam mensagens entre si quando o utilizador se pretende autenticar para utilizar um determinado recurso da rede.

Uma mensagem RADIUS consiste num pacote UDP com um cabeçalho RADIUS contendo o tipo de mensagem e podendo ainda ter, ou não, alguns atributos associados à mensagem. Cada atributo RADIUS especifica uma parte de informação sobre a tentativa de ligação. Por exemplo, existem atributos RADIUS para o nome de utilizador, palavra passe do utilizador, tipo de serviço pedido pelo utilizador e o endereço IP do cliente de acesso. Os atributos RADIUS são utilizados para transmitir informações entre clientes RADIUS, *proxies* RADIUS e servidores de RADIUS.

O formato dos pacotes enviados por este protocolo é o seguinte:

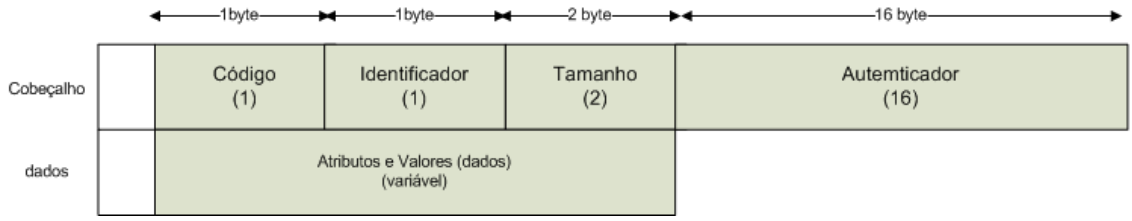

Figura 1 – Formato dos pacotes RADIUS

De seguida são explicados os campos do cabeçalho RADIUS:

- Código: este campo usa um octeto e indica de que tipo de pacote se trata. Os códigos que podem ser utilizados são os seguintes:
  1 Access-Request
  2 Access-Accept
  3 Access-Reject
  4 Accounting-Request
  5 Accounting-Response
  11 Access-Challenge
  12 Status-Server
  13 Status-Client
  255 Reserved

- Identificador: Tal como o anterior, usa também um octeto, e é usado para fazer a ligação entre mensagens referentes ao mesmo pedido. Os servidores RADIUS podem usar este campo, juntamente com o endereço e porta de origem e destino para determinar pedidos repetidos.

- Tamanho: Este campo usa dois octetos e indica o tamanho da mensagem RADIUS. O valor deste campo é calculado somando os tamanhos dos campos de código, identificador, autenticador e campos de atributos.
- Autenticador: Este campo usa até 16 octetos e é onde a integridade da mensagem é verificada. Existem dois tipos de valores de autenticadores: os valores para os pedidos e os valores para as respostas. Autenticadores de pedidos são usados com os pacotes *Authentication-Request* e *Accounting-Request*. Nos pacotes de pedidos os autenticadores usam 16 octetos e são gerados de forma completamente aleatória para evitar possíveis ataques. O autenticador de respostas é enviado com os pacotes de *Access-Accept*, *Access-Reject*, e *Access-Challenge*. Neste caso o valor do autenticador é calculado usando uma *hash* MD5 gerada a partir dos valores do campo do código, identificador, tamanho, autenticador de pedido e ainda com os dados presentes na região de *payload*.

Para autenticação existem quatro tipos de pacotes: *Access-Request*; *Access-Accept*; *Access-Reject* e *Access-Challenge*.

O *Access-Request* é usado pelos clientes (por exemplo APs) para pedir acesso de um determinado utilizador a um determinado recurso. O cliente envia o pedido ao servidor RADIUS com uma lista dos serviços pedidos. O campo dos dados do *Access-Request* inclui o atributo de *username* para identificar o utilizador que pretende aceder aos recursos. É ainda necessário incluir o endereço IP (*Internet Protocol*) do cliente de onde envia o pedido assim como o atributo de palavra-passe com o respectivo valor.

Os pacotes de *Access-Accept* são enviados do servidor RADIUS para o cliente para indicar a este que o pedido do utilizador foi aceite. Este pacote pode ainda definir alguns parâmetros para a inicialização da sessão como um endereço IP, uma mensagem de texto definida anteriormente, entre outros.

O *Access-Reject* é enviado do servidor para o cliente e indica que o pedido do utilizador foi rejeitado. Esta rejeição pode ser provocada por falta de privilégios, politicas ou outro critério.

Os pacotes de *Access-Challenge* são também enviados do servidor para o cliente e são usados quando o servidor recebe informação conflituosa, requer mais informação ou ainda quando pretende diminuir a probabilidade de autenticação fraudulenta.

O servidor RADIUS pode também ser configurado em proxy. Neste caso o servidor irá funcionar como cliente que redirecciona os pedidos de acesso para um outro servidor, ou seja, passa a ser responsável pela troca de mensagens entre o NAS e o servidor remoto.

2.5 - Estabelecimento de uma Sessão

Quando o utilizador pretende usufruir de um determinado recurso da rede este necessita de se conectar ao NAS que, por sua vez, irá trocar mensagens com o servidor RADIUS para o estabelecimento da sessão. Quando um cliente é configurado para utilizar RADIUS qualquer utilizador que pretenda usufruir de um serviço deste NAS necessita de apresentar a este com as suas credenciais de autenticação. Com vista à autenticação, o NAS envia ao servidor

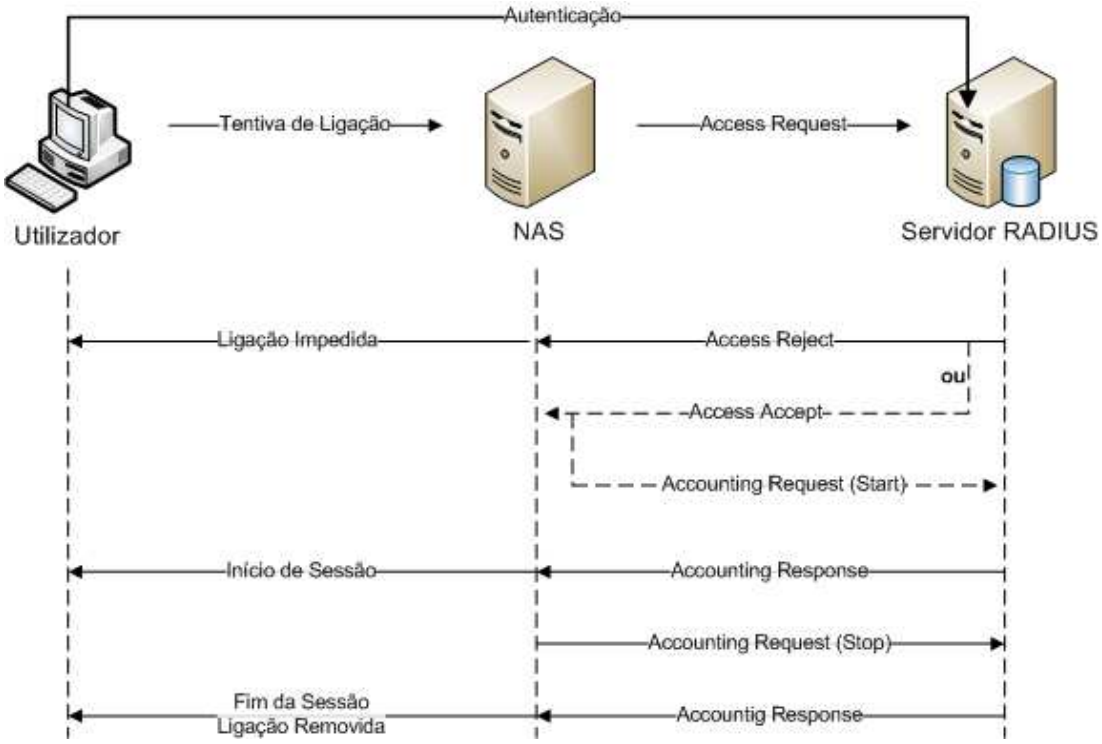

Figura 2 – Troca de mensagens RADIUS

RADIUS um *Access-Request* ao servidor, contendo atributos como o nome de utilizador e palavra passe cifrada. Se não for obtida qualquer resposta por parte do servidor num tempo limite previamente estabelecido, o pedido de acesso é reenviado. O NAS pode ainda optar pelo envio do pedido de acesso para outros servidores alternativos em caso de falha de conexão com o servidor primário.

Quando recebe o *Access-Request*, o servidor RADIUS tenta autenticar o utilizador consultando a sua base de dados. Após autenticação, o servidor consulta novamente a sua base de dados para verificar se o utilizador que pretende o acesso possui as permissões necessárias para obter o acesso ao serviço.

Caso alguma das condições para o acesso do utilizador ao serviço não seja satisfeita, o servidor RADIUS envia para o NAS um pacote de *Access-Reject*, rejeitando o pedido de acesso feito pelo utilizador. Ao receber esta mensagem, o NAS pode, ou não, enviar um aviso ao utilizador acusando a recepção de um *Access-Reject* por parte do servidor RADIUS.

Caso todas as condições sejam satisfeitas, o servidor RADIUS pode pedir ainda pedir uma confirmação enviando um desafio, a mensagem de *Access-Challenge*, para o NAS. Esta mensagem serve para aumentar o grau de segurança. O desafio pode ser enviado em forma de questão a que apenas o utilizador saiba responder e, neste caso, o NAS teria de enviar o desafio ao utilizador pedindo resposta, ou então um código que apenas determinados clientes, recorrendo a mecanismos específicos, consigam responder, garantindo assim a identidade do utilizador. A resposta ao desafio feito é enviada pelo NAS para o servidor numa mensagem em tudo idêntica ao *Access-Request* original mas que, no campo destinado à senha do utilizador, é colocada a resposta do desafio cifrada e uma indicação de que o *Access-Request* é uma resposta ao desafio imposto pelo servidor. O servidor pode responder a este novo *Access-Request* com um *Access-Accept* aceitando a autenticação do utilizador, com um *Access-Reject* rejeitando a autenticação do utilizador, ou ainda enviando a mensagem de

Access-Challenge com um novo desafio.

Por fim, e caso todas as condições sejam satisfeitas o servidor envia um pacote de *Access-Accept* com uma lista de valores de configuração para o utilizador. Estes valores incluem o tipo de serviço e todos os valores necessários para a utilização do serviço desejado por parte do utilizador.

2.6 - Soluções RADIUS Existentes

Existem no mercado muitas soluções para servidores RADIUS. Segundo [1] o freeRADIUS é o servidor RADIUS mais utilizado para sistemas Linux. Este é responsável pela autenticação de pelo menos um terço dos utilizadores na Internet. Os restantes utilizadores encontram-se divididos entre os restantes servidores, destacando-se entre eles o Cisco ACS (*Access Control Server*) e o Microsoft IAS (*Internet Authentication Service)*.

2.6.1 - freeRADIUS

O freeRADIUS, descrito e disponível em [16], é uma implementação de RADIUS modular, de alta performance e rica em opções e funcionalidades. Esta inclui servidor, cliente, bibliotecas de desenvolvimento e muitas outras utilidades. Pode ser instalada em sistemas Linux e Mac. Devido a estas características e tendo em conta o facto de ser uma aplicação *open source* esta foi a implementação RADIUS utilizada para o desenvolvimento do trabalho.

Cada cliente deverá ser correctamente configurado de modo a comunicar com o servidor instalado. Os clientes podem ser computadores de rede, pontos de acesso ou uma máquina com software apropriado de modo a poder implementar um cliente (p.ex. radiusclient).

Entre outras características do freeRADIUS salienta-se o suporte à limitação do número máximo de acessos simultâneos, capacidade de inserir mais do que um valor por omissão e capacidade de funcionar como um servidor *proxy*.

O freeRADIUS ao ser instalado numa máquina disponibiliza as seguintes ferramentas:

- radclient - emula um cliente RADIUS, enviando pacotes para o servidor imprimindo a resposta.
- radlast - mostra as ultimas sessões de utilizadores.
- radtest - frontend para o radclient, utilizado para testar o servidor.
- radwho - mostra os utilizadores ligados.
- radrelay - reenvia dados de *accounting* para um outro servidor RADIUS.
- radwatch - não é utilizado, é instalado apenas por razões históricas.
- radzap - efectua a limpeza da base de dados de sessões activas.

O freeRADIUS possui ainda uma básica aplicação para a sua gestão designada de dialupadmin.

O freeRADIUS fornece permite a adição de utilizadores e regras de acesso tanto em ficheiros de texto como numa base de dados. Os tipos de bases de dados compativeis são: MySQL, PostgreSQL e Oracle. Quando integrado com a base de dados, o freeRADIUS apresenta as seguintes tabelas:

- radusergroup - associa um utilizador a um grupo de utilizadores;
- radcheck - guarda o registo de cada utilizador com os respectivos atributos associados;

- radgroupcheck – associa atributos a um determinado grupo de utilizadores;
- radreply – contém lista de atributos enviados ao utilizador;
- radgroupreply – guarda os atributos que são devolvidos a todos os utilizadores de um grupo;
- radpostauth – guarda informações acerca das respostas enviadas para os utilizadores;
- radacct - onde se encontra toda a informação de contabilização;
- nas – onde se encontra informação dos clientes/NAS. Depois de adicionar, editar ou apagar dados desta tabela é necessário reiniciar o servidor RADIUS para actualizar os dados referentes aos clientes.

De seguida é apresentado o modelo de dados do freeRADIUS. Neste não se encontram as tabelas radpostauth e radacct que não apresentam grande relevância neste ponto uma vez que servem apenas para registo das respostas enviadas e da informação de contabilização, respectivamente.

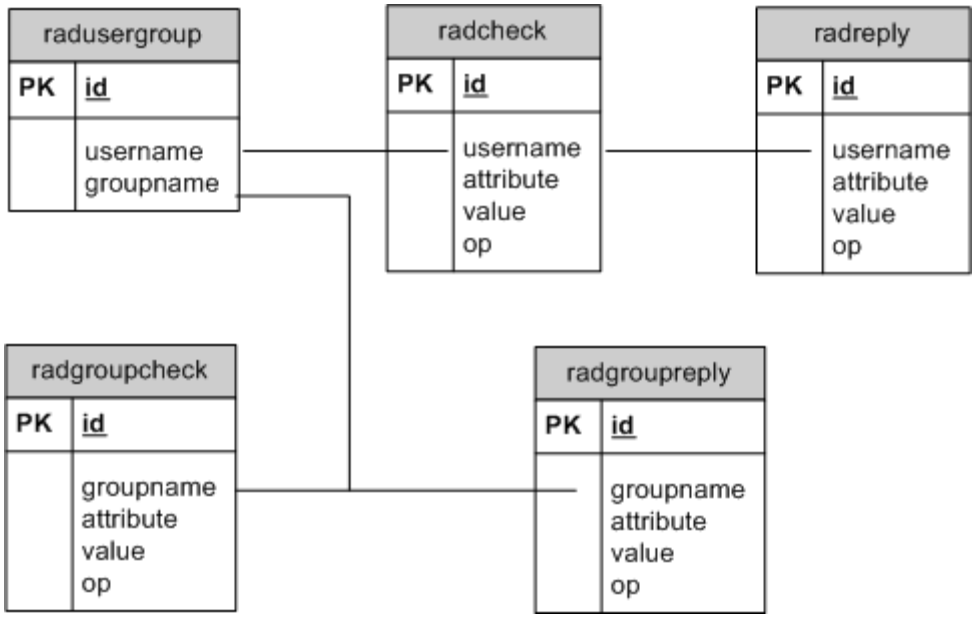

Figura 3 – Modelo de dados do freeRADIUS

Na base de dados irão estar presentes utilizadores com um respectivo *username*. Estes utilizadores irão estar associados a atributos através da tabela radcheck, como já foi referido atrás. Estão presentes também na base de dados os *groupname*. Os utilizadores estão associados aos respectivos grupos através da tabela radusergroup. Por sua vez, cada grupo terá também associado a atributos, o que quer dizer que todos os seus membros também estarão associados a esses atributos. Esta associação é feita pela tabela radgroupcheck. As tabelas de radreply e radgroupreply estão presentes para guardar listas de atributos a enviar para utilizadores e membros dos grupos, respectivamente.

2.6.2 - Cisco Access Control Server

O Cisco *Secure Access Control Server* (ACS), descrito em [17], é uma politica de controlo de acessos que proporciona um ambiente centralizado de controlo de autenticação,

autorização e contabilização de acesso de utilizadores a recursos de rede. Este suporta multiplos cenários simultaneamente. Estes cenários incluem:

- Dispositivo de administração: autentica administradores e comandos de autorização;
- Acesso Remoto: funciona com VPN e outros dispositivos de acesso a redes remotas para reforçar as políticas de acesso;
- Wireless: Autentica e autoriza utilizadores e hosts wireless e reforça as políticas de segurança específicas de wireless;
- Controlo de admissão na rede: comunica com servidores de postura e auditoria para reforçar as políticas de controlo de admissão.

Esta implementação do RADIUS é para o sistema operativo Windows.

2.6.3 - Microsoft Internet Authentication Service

O Microsoft *Internet Authentication Service* (IAS), descrito em [18], é a implementação da Microsoft de um servidor RADIUS. Tal como é especificado pelo RADIUS o IAS oferece serviços de autenticação, autorização e contabilização centralizados para diferentes tipos de acessos de rede, incluindo *wireless* e VPNs. Funcionando como *proxy*, o IAS reencaminha as mensagens de autenticação e autorização para outros servidores RADIUS.

Como a anterior, também esta é uma implementação do RADIUS para Windows.

2.7 - Web 2.0

Segundo [9], em 2004, quando a O'Reilly Media, John Battele e a CMP anunciaram a conferência *Web* 2.0 não tinham ideia que estariam a dar nome à próxima grande evolução da tecnologia computacional. A ideia original da *Web* 2.0 era muito mais simples. Essas companhias viram a *Web* 2.0 não como uma nova versão da *Web* mas sim como um desenvolvimento das técnicas de desenvolvimento *Web* que teriam como objectivo aproveitar mais potencialidades da *Web*.

Em [9] é definido que a *Web* 2.0 levou também a uma evolução nos modelos de negócios. A *Web* 2.0 veio aumentar também a performance da rede uma vez que os utilizadores *online* já não são limitados pelas coisas que podem encontrar, ver, ou fazer *download* mas sim pela quantidade de acções que podemos fazer, interagir, combinar, enviar, alterar e personalizar. Este estudo mostra que na *Web* 2.0 a gestão de combinações de efeitos de rede on-line é fundamental para o sucesso competitivo.

A O'Reilly Media entende que ainda há muito para aprender acerca da *Web* 2.0. A revolução da *Web* 2.0 continua todos os dias e não é de espantar que responsáveis por desenvolvimento *Web* descubram, em cada dia que passa, novas formas de utilizar os princípios da *Web* 2.0.

2.8 - Performance de Linguagens

Antes da escolha do framework teve de ser escolhida a linguagem que melhor se

adequaria aos requisitos da aplicação que irá ser desenvolvida. Irão ser analisadas as características de cada linguagem baseadas em estudos realizados e, posteriormente, irá ser escolhida a linguagem a utilizar no desenvolvimento da aplicação. No entanto, é de salientar que alguns dos estudos encrontrados já possuem alguns anos o que significa que, hoje em dia, os resultados verificados nos estudos podem já não se verificar.

O estudo presente em [4] destinou-se ao estudo das infra-estruturas fornecidas pelas tecnologias de PHP, Ruby e Java. O estudo foi efectuado combinando as infra-estruturas com componentes populares e avaliando depois os prós e contras de cada combinação. No estudo foi implementada a mesma aplicação *Web* nos três ambientes e foram feitos testes de performance sobre cada um deles. Este estudo apresenta como desvantagens de O-R *Mapping* (do Ruby e JavaEE) a falta de controlo directo sobre o acesso a bases de dados, não ser optimizado para bases de dados específicas e ainda um inadequado *tuning* quando preciso. Como benefícios apresenta o modelo de programação de orientação a objectos, a facilidade de manutenção, habilita automaticamente o *caching* de dados e a habilitação de optimizações. Através de testes de performance apresentados no estudo concluiu-se também que os problemas de performance estão principalmente no lado do cliente. Os clientes de Java experimentaram grande uso do CPU (*Central Processing Unit*) e grandes chamadas ao sistema.

Um outro estudo, descrito em [5], mostra a comparação de performance entre PHP (*Hypertext Preprocessor*) e Java. Este comprova que PHP consome menos CPU, obtendo melhor performance que os *servlets* de Java. A isto acresce o facto de o PHP ser executado no mesmo processo do servidor *Web*. Embora seja uma vantagem em termos de execução isto implica que os *scripts* PHP estejam na mesma máquina que o servidor, o que não acontece com Java *servlets* que permite mover o *servlet* para uma máquina dedicada, melhorando a performance. Este estudo conclui ainda que programar scripts PHP é muito mais simples do que programar Java *servlets*.

Em [6] é feito um estudo acerca do impacto das tecnologias Perl, PHP e Java num servidor *Web*. Este estudo mostra que, em geral, as tecnologias de servidor Java superam as de servidores PHP e Perl no que toca à geração de conteúdo dinâmico. No entanto, mostra também que a performance abaixo da sobrecarga é bastante imprevisível, e que as tecnologias de geração de conteúdo dinâmico como PHP e Perl são bastante robustas abaixo da sobrecarga.

De facto, JavaEE (*Java Enterprise Edition*) é mais pesado ao nível de processamento do lado do servidor e bem mais complicado ao nível de implementação quando comparado com PHP. Outras documentações revelam também que o Python e o Perl exigem mais recursos de processamento, não sendo também linguagens adequadas para a implementação.

A escolha recaiu assim sobre PHP uma vez que o trabalho tem como requisito uma ligação a uma base de dados de um servidor RADIUS e PHP é a melhor linguagem para bases de dados específicas e controlo sobre o acesso à base de dados, dado não ter O-R (*Object-Relational*) Mapping implementado de origem, o que não acontece com o JavaEE e Ruby.

2.9 - AJAX

O AJAX é um grupo de técnicas de desenvolvimento Web usadas para aumentar a interactividade com o cliente e criar RIAs (*Rich Internet Applications*). Usando AJAX, as aplicações Web podem enviar, sem ter de recarregar novamente a página inteira, dados do

servidor para o cliente. AJAX também pode ser usado para o servidor recolher dados, de um modo invisível, da aplicação *Web*. Os dados são recebidos pelo servidor usando o objecto XMLHttp ou através do uso de *scripts* remotos em *browsers* que não o suportem.

Em [7] são descritas as tecnologias incorporadas em AJAX e qual o funcionamento deste. O AJAX inclui:

- XHTML (*eXtensible Hypertext Markup Language*) e CSS (*Cascading Style Sheets*) para a apresentação da página;
- Apresentações e interacção dinâmicas usando o modelo de objecto de documento;
- Troca e manipulação de dados usando XML (*eXtensible Markup Language*) e XSLT (*eXtensible Stylesheet Language Transformations*);
- Recepção de dados usando XMLHttpRequest;
- Utilização de Javascript para integrar os anteriores.

Em [8] foi realizado um estudo acerca da performance do AJAX em que mostra que a utilização desta linguagem leva a uma menor ocupação da largura de banda em mais de 61%. Isto pode ser explicado com o facto do AJAX permitir partilhar o processamento, que antes seria inteiramente feito pelo servidor, com o cliente. De facto, uma das principais vantagens associadas à utilização de AJAX é a mais baixa utilização de largura de banda, o que no caso da solução desenvolvida neste trabalho é benéfico uma vez que se pretende que a largura de banda utilizada seja mínima devido à variedade de capacidade de processamento entre dispositivos. Associada ao consumo de largura de banda está associado o tempo de resposta da página que, com AJAX, será inferior do que com uma aplicação normal. Uma outra vantagem muito importante para este trabalho é que RIAs que utilizam AJAX podem ser acedidas usando qualquer *browser Web*, e qualquer sistema operativo.

2.10 - Modelo MVC

O Paradigma *Model-View-Controller* (MVC) foi desenvolvido com base no modelo entrada-processamento-saída de um sistema, com o objectivo de separar o sistema em 3 partes. O Controlador lida com a entrada, controlando tanto a interface com o teclado e o rato, como a interface entre o Modelo e as Vistas associadas. O Modelo contém a lógica da aplicação e acede aos dados persistentes. A Vista é responsável pela exibição da saída criada pelo Modelo.

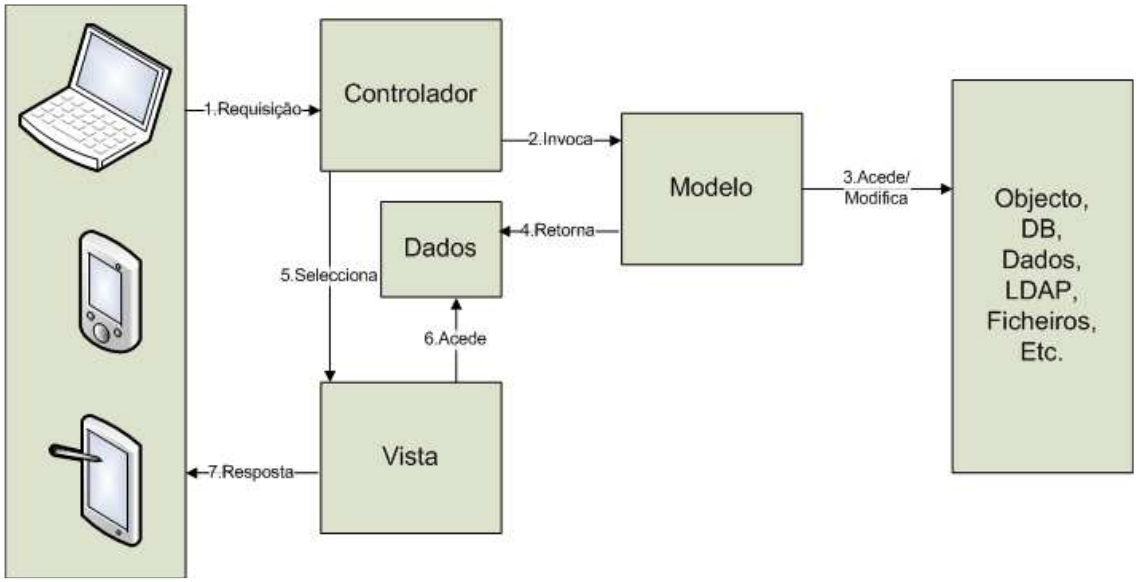

Figura 4 – Diagrama do paradigma Model-View-Controller

A intenção original do uso do MVC era a de facilitar o desenvolvimento de aplicações com interfaces gráficas, mas também tem sido aplicado com sucesso no desenvolvimento cliente-servidor e sistemas *Web*. Controladores são responsáveis pelo nó do controlo do sistema e navegação ao Modelo apropriado e componente da Vista. Estas responsabilidades são geralmente divididas entre um (ou alguns) *Front Controller* e muitos *Page Controllers*. O *Front Controller* aceita todas as entradas do utilizador, define variáveis do sistema, verifica segurança e invoca o *Page Controller* apropriado. Mais de um *Front Controller* é utilizado em aplicações distribuídas ou aplicações segmentadas como, por exemplo, uma rede segura e outra sem segurança. Cada *Page Controller* está relacionado a um Modelo e a uma componente *View Page Controllers* tem normalmente pouco código, simplesmente invocando o Modelo apropriado ou então a componente *Vista*. Um *Page Controller* não deve conter nenhuma lógica da aplicação. Um *Modelo* contém a lógica da aplicação para uma página, incluindo acesso a dados por dados persistentes. O resultado de um Modelo é disponibilizado para uma Vista através de um dispositivo intermediário. Como um Modelo pode ser invocado por diferentes *Page Controllers*, os resultados podem ser representados por diferentes Vistas. Uma Vista está relacionada a um único *Page Controller*.

2.11 - Frameworks de Desenvolvimento Web

Framework é um conjunto de mecanismos usados para resolver um problema específico. *Framework* de software compreende um conjunto de classes, que unidas e implementadas, são usadas para auxiliar o desenvolvimento de uma funcionalidade genérica.

Assiste-se, nos últimos anos, à tendência de se oferecer *software* pela Internet, vendidos não como pacotes para se instalar localmente, mas como serviços. Para que a interactividade fosse comparável a aplicações *desktop*, foi necessário a conjugação de várias tecnologias (como, por exemplo, o AJAX) que tornassem a experiência do utilizador mais rica, com interfaces rápidas e muito fáceis de usar. Constatou-se então que quanto mais simples e modular a programação, melhor é o resultado final. Assim torna-se mais fácil tirar ou acrescentar uma funcionalidade ou reutilizar uma parte do seu software com outro *software*.

De modo a facilitar a implementação da aplicação foi escolhida uma *framework* que se adaptasse ao tipo de trabalho desejado. A escolha deste seguiu vários critérios definidos analisando a descrição do problema e objectivos do trabalho.

2.12 - Critérios de Avaliação das Frameworks

De modo a auxiliar a escolha entre as várias *frameworks* existentes foram definidos critérios de avaliação de acordo com os objectivos do trabalho.

Uma vez que se pretende que o *frontend* implementado seja suportado por diferentes terminais será um requisito que a *framework* suporte o desenvolvimento de Vistas para vários tipos de plataformas. Uma vez que os diferentes terminais apresentam diferentes capacidades de processamento, uma *framework* que implemente uma gestão de largura de banda será preferível relativamente a uma que não o faça. Associado ao problema do acesso através de diferentes terminais está também o acesso através de diferentes *browsers,* logo a *framework* desejada terá de suportar os vários tipos de *browsers Web*.

É também importante averiguar se a f*ramework* possui linguagens de *client-side* ou apenas de *server-side*. Linguagens de *client-side*, como Javascript, aumentam o processamento do lado do cliente, o que neste trabalho poderá não ser desejável uma vez que aplicações desenvolvidas com linguagens *client-side* poderão não ser acessíveis em dispositivos com baixa capacidade de processamento.

Outro aspecto importante na selecção da f*ramework* é a complexidade do desenvolvimento. É dada preferência a *frameworks* com uma maior facilidade e flexibilidade de implementação tanto em termos de desenvolvimento das interfaces como outras funcionalidades. São também analisadas os mecanismos de interligação com a base de dados, de preferência que abstraiam o acesso à mesma; as várias linguagens suportadas pelas diferentes *frameworks*.

Assim, os critérios de avaliação dividem-se em:

- Compatibilidade - com diferentes *browsers* a usar;
- Adaptabilidade - a diferentes terminais;
- *Client-Side* - linguagens que correm do lado do cliente;
- Facilidade/Flexibilidade - de desenvolvimento;
- Mecanismo de conexão com bases de dados - que abstrai a base de dados;
- Arquitectura
  - API (*Application Programming Interface)* Simples
  - API MVC
  - CMS (*Content Management System)*
- Linguagens suportadas;

De acordo com os critérios identificados foi efectuado um levantamento das *frameworks* e a sua adequabilidade aos objectivos do projecto. Estas *frameworks* são descritas de seguida encontrando-se, posteriormente, uma tabela comparativa das *frameworks* e justificação da escolha tomada.

2.13 - Lista de Frameworks de desenvolvimento *Web*

Embora haja inúmeras frameworks, apenas foram descritas as cinco que mais se adequam ao desenvolvimento da aplicação desejada.

2.13.1 - CakePHP

Figura 5 – Logotipo do CakePHP

O CakePHP, disponível em [11], é uma *framework* de desenvolvimento *Web open-source* e escrita em PHP. Torna mais simples a interligação entre a aplicação e a base de dados e usa o paradigma MVC. Entre outras funcionalidades, o CakePHP é compatível com PHP4 e PHP5, possui um modelo CRUD (*create, read, update and delete*) integrado para facilitar o funcionamento a base de dados, possui um mecanismo de *caching de Views* flexível, ajuda em *Views* com AJAX e formulários HTML (*HyperText Markup Language)* e possui uma lista de controlo de acessos ACL (*Access Control List*) para segurança. Este contém também *templates* rápidos e flexíveis de modo a ser possível desenvolver rapidamente aplicações.

2.13.2 - CodeIgniter

Figura 6 – Logotipo do CodeIgniter

Framework open-source, descrito e disponível em [12], que obedece ao paradigma MVC e que é usado para o desenvolvimento rápido de *Web* sites dinâmicos com PHP. Oferece um mecanismo ORM (*Object-relational mapping*) com *Scaffolding*, que permite a geração de código para as diversas operações da interligação com a base de dados, e que suporta múltiplos tipos de bases de dados. Possui também gestão de sessão, filtro XSS (*Cross-site scripting*) que previne *SQL (Structured Query Language) injections* e possui ainda mecanismo de *caching*.

2.13.3 - Prado

Figura 7 – Logotipo do Prado

Prado, disponível em [13], é uma *framework* para desenvolvimento *Web* com orientação a objectos e alta reutilização de código. Possui uma arquitectura MVC modular, configurável e adaptável. Apresenta uma segurança elevada com a protecção contra *cross-site script* e protecção de cookies. *Cross-site scripting* é uma vulnerabilidade das aplicações Web que permite a injecção de código por parte de utilizadores mal intencionados e protecção de cookies. Possui funcionalidades de *caching*, autenticação e autorização, personalização de erros e excepções, entre outras.

2.13.4 - Symfony

Figura 8 – Logotipo do Symfony

O Symfony, descrito em [14], é uma *framework* completa (*full-stack*) e *open-source*. Esta fornece uma arquitectura MVC, componentes e ferramentas de modo a possibilitar o desenvolvimento de aplicações complexas Web de um modo rápido. Esta oferece também escabilidade e flexibilidade no desenvolvimento das aplicações. Tal como as anteriores também o Symfony possui um mecanismo de *Caching* que permite a gestão da largura de banda.

2.13.5 - Zend Framework

Figura 9 – Logotipo do Zend Framework

A Zend Framework, disponível em [15], é baseada na simplicidade, nas melhores práticas da orientação a objectos e possui uma base de código ágil e extensível. Esta *framework* foca-se na construção de aplicações *Web* 2.0 mais seguras, de confiança e modernas. É uma *framework open-source* e baseada na arquitectura MVC. Esta *framework* proporciona uma simplicidade extrema e um aumento de produtividade por parte do programador. Suporta AJAX através de JSON (*JavaScript Object Notation*), possui formatos de dados e fácil acesso a estes, serviços Web e uma biblioteca PHP5 orientada a objectos de grande qualidade. Tal como os anteriores, possui um mecanismo de *caching*.

2.14 - Quadro Comparativo das Frameworks

De seguida é apresentada uma tabela comparativa das *frameworks* que mais se adequam ao trabalho e será apresentada uma análise a esta com vista a determinar qual destas *frameworks* se adequa melhor ao desenvolvimento do projecto. Nesta tabela apenas são apresentados as cinco *frameworks* que mais se destacam entre as restantes uma vez que, como já foi visto atrás, a lista de *frameworks* é muito extensa e seria complicado realizar uma análise profunda a todas elas.

Frameworks	Client-Side	Facilidade/ Flexibilidade	Conexão à BD	Linguagens	Performance
cakePHP	JavaScript	★★★	★★★	PHP4;PHP5; AJAX	★★
CodeIgniter	N/A	★★★★★	★★★★★	PHP4;PHP5	★★★★★
Prado	JavaScript	★★	★★★	PHP5; AJAX	★★
Symfony	JavaScript	★★★	★★★	PHP4; AJAX	★★
Zend Framework	N/A	★★★	★★★	PHP5	★★★★

Tabela 1 – Tabela Comparativa das Frameworks

Em relação ao período de aprendizagem o CakePHP tem regras restritas quanto aos nomes das tabelas da base de dados, onde os ficheiros devem ser colocados, nomes de métodos e nomes de classes. O Symfony guarda a sua configuração no formato .yml e muita da interacção com a aplicação dá-se através de uma consola (criar tabelas na base de dados, modelos de dados e outros ficheiros). Tanto o CakePHP como o Symfony possuem mecanismos de ORM que contêm regras e convenções restritas que devem ser respeitadas para o bom funcionamento da aplicação. Quanto ao Zend e CodeIgniter são mais flexíveis quanto ao uso de modelos e como são usados. Em termos de flexibilidade também o Zend e CodeIgniter são mais flexíveis quando comparados com as restantes *frameworks*.

Em relação à performance o CodeIgniter é o que apresenta maior velocidade devido à sua simplicidade apresentando o dobro da performance do Zend, enquanto que as restantes se encontram abaixo destas duas.

Até este ponto a escolha seria claramente entre o CodeIgniter ou o Zend. No entanto, o facto de estas duas *frameworks* não possuírem suporte a AJAX poderia ser um ponto que os deixaria fora das opções finais uma vez que limitaria o trabalho a desenvolver no ponto em que se pretende uma aplicação com grande interactividade para com o utilizador, pelo menos nos terminais que apresentem maior capacidade de processamento. No entanto, apesar de estas duas *frameworks* não possuírem suporte a AJAX de origem, existem bibliotecas adicionais para cada uma das *frameworks* que adicionam o suporte do AJAX (no caso do Zend é utilizada a biblioteca JSON e no caso do CodeIgniter é utilizada a biblioteca *AJAX for CodeIgniter*).

Um outro ponto importante para a escolha da framework é a documentação disponível para cada *Framework*. Neste ponto o Symfony destaca-se dos restantes.

Em relação à compatibilidade e adaptabilidade não foram encontrados dados e, por isso, não foram inseridos na tabela os campos referentes a estes requisitos. No entanto, no que toca à adaptabilidade esta pode ser construída dependendo da implementação desenvolvida.

No que toca à compatibilidade os problemas podem surgir aquando da execução de códigos JavaScript, sendo que estes terão de ser desenvolvidos tendo em conta os diversos browsers existentes.

Tendo em conta os diversos pontos discutidos e apresentados na tabela a escolha recaiu sobre o CodeIgniter devido à sua boa flexibilidade e facilidade, suporte das linguagens necessárias para o desenvolvimento do projecto e principalmente devido ao facto de apresentar a melhor performance.

2.15 - Requisitos funcionais

De forma a poder-se avaliar os trabalhos idênticos já implementados serão identificados e descritos, de seguida, os requisitos funcionais pretendidos para a aplicação.

2.15.1 - Autenticação de Gestores

Deve ser solicitado ao gestor um conjunto de nome de utilizador e palavra passe de modo a autenticá-lo no sistema e verificar quais as suas permissões.

2.15.2 – Registar/Editar Gestores

Deverá ser possível o registo de gestores no sistema por parte do administrador do sistema. Este registo deverá apresentar como campos o nome do utilizador, palavra passe pretendida e ainda o endereço de correio electrónico de modo a notificar o novo utilizador do registo e permitir a este a confirmação do registo. Os gestores terão ainda a possibilidade de editar os seus dados ou, no caso dos administradores, os dados e nível de acesso de todos os outros gestores.

2.15.3 – Registar/Editar Utilizadores

Os gestores, dependendo das suas permissões, deverão ser capazes de criar, editar e apagar utilizadores do servidor RADIUS.

2.15.4 – Gestão de Grupos

A aplicação deverá permitir aos gestores criar, editar e apagar grupos. Estes grupos poderão ser grupos de utilizadores do RADIUS ou grupos de gestores. Na edição do grupo deverá ser possível adicionar ou retirar elementos do grupo ou ainda editar os dados do grupo.

2.15.5 – Gestão de Recursos e Tipos de Recursos

Os gestores ao acederem à aplicação deverá ser-lhes possível adicionar, editar e ainda remover recursos da base de dados ou ainda tipos de recursos.

2.15.6 – Regras de Acesso a Recursos

Um dos pontos principais da aplicação é a automatização da criação de regras de acesso a recursos. Aqui deverá ser possível aos gestores, dependendo das suas permissões, adicionar,

editar ou remover regras de acesso por parte dos utilizadores a recursos existentes no servidor RADIUS.

2.15.7 – Atribuir Gestão de Recursos a Gestores

A aplicação deverá permitir aos administradores delegar a gestão de recursos por vários gestores, ou seja, atribuir a gestores ou grupos de gestores a responsabilidade de um ou mais recursos.

2.15.8 – Consulta de Registos

Deverá ser possível aos gestores a consulta dos registos de acesso. A aplicação deverá disponibilizar também uma consulta a registos de modificações efectuadas.

2.15.9 – Realização de Estatística dos Acessos

A aplicação deverá ser também responsável pela criação de estatísticas referentes aos acessos por parte dos utilizadores.

2.16 - Trabalhos Relacionados

Nesta secção são descritas as aplicações já existentes para gestão de servidores de RADIUS. Irá ser dada uma breve descrição de cada um deles assinalando as suas vantagens e desvantagens.

2.16.1 - Dialupadmin

O dialupadmin, disponível em [19], é um frontend para gestão de RADIUS simples mas pobre e que é incluido no FreeRADIUS. Como funcionalidades o dialupadmin permite criar novos utilizadores, verificar estado do servidor, administração de configurações de RADIUS e informação pessoal, entre outras. Ao contrário do que se pretende no trabalho, este é um *frontend* com baixa interacção por parte do cliente e não adaptável a diferentes tipos de terminais devido à sua interface pouco flexível.

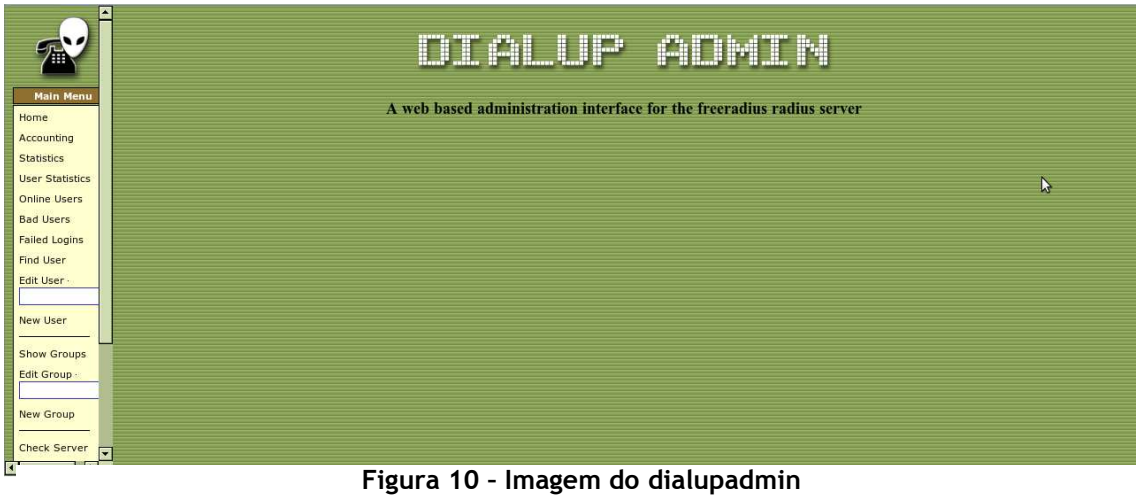

Figura 10 – Imagem do dialupadmin

2.16.2 – daloRADIUS

O daloRADIUS, disponível em [20], é uma aplicação *Web* de gestão de RADIUS. Tem como funcionalidades gestão de utilizador, relatórios gráficos e estatísticas de contabilização. É escrita em PHP e JavaScript e utiliza uma camada de abstracção da base de dado. O facto de utilizar JavaScript faz com que esta aplicação apresente alguns problemas de compatibilidade a diferentes *browsers Web* e aumente a necessidade de maior processamento do lado do cliente.

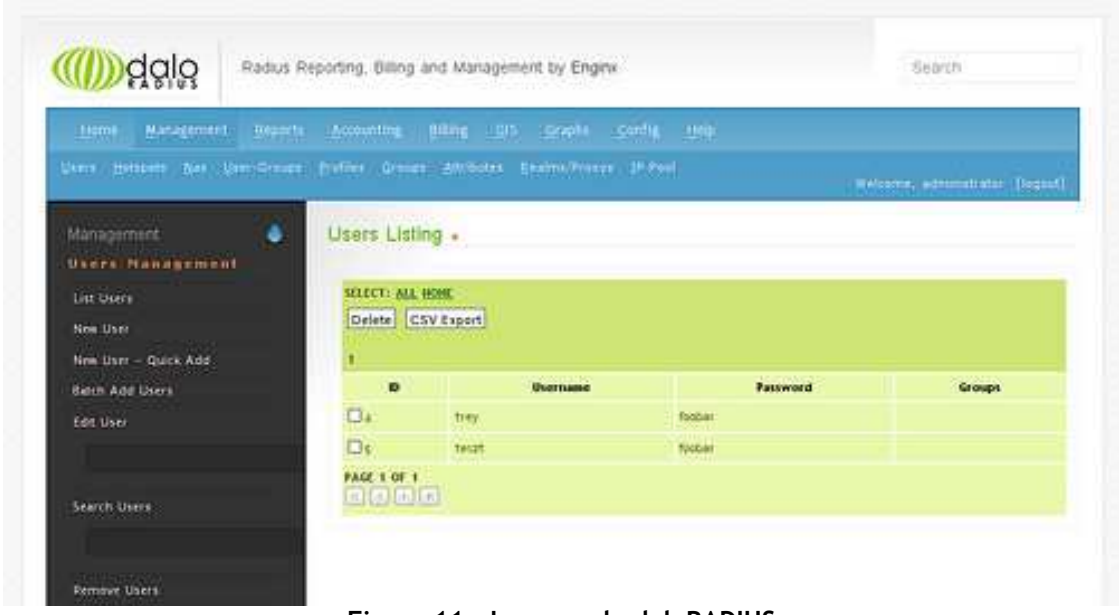

Figura 11 – Imagem do daloRADIUS

2.16.3 - ezRADIUS

Outra aplicação *Web* para gestão de RADIUS é o ezRADIUS, disponível em [21]. Este é uma aplicação simples, fácil e expansível. Esta destaca-se das outras pela sua configuração simples e mecanismo de *login* simples. No entanto, é limitada em termos das funcionalidades pretendidas e níveis de utilizador (inexistentes). Além disto apresenta também uma fraca interactividade e a adaptabilidade é inexistente.

Figura 12 – Imagem do ezRADIUS

2.16.4 - phpMyPrepaid

O PhpMyPrepaid, disponível em [22], é uma ferramenta que fornece a possibilidade de criação de contas RADIUS usufruindo das vantagens das aplicações *Web*. Esta aplicação apresenta ainda funcionalidades de gestão de utilizadores, gestão de múltiplas localizações, ferramentas de contabilização, estatísticas, entre outras. Tal como as ferramentas anteriormente descritas, esta não apresenta a interactividade com o utilizador desejada e não possui adaptabilidade a diferentes tipos de terminais.

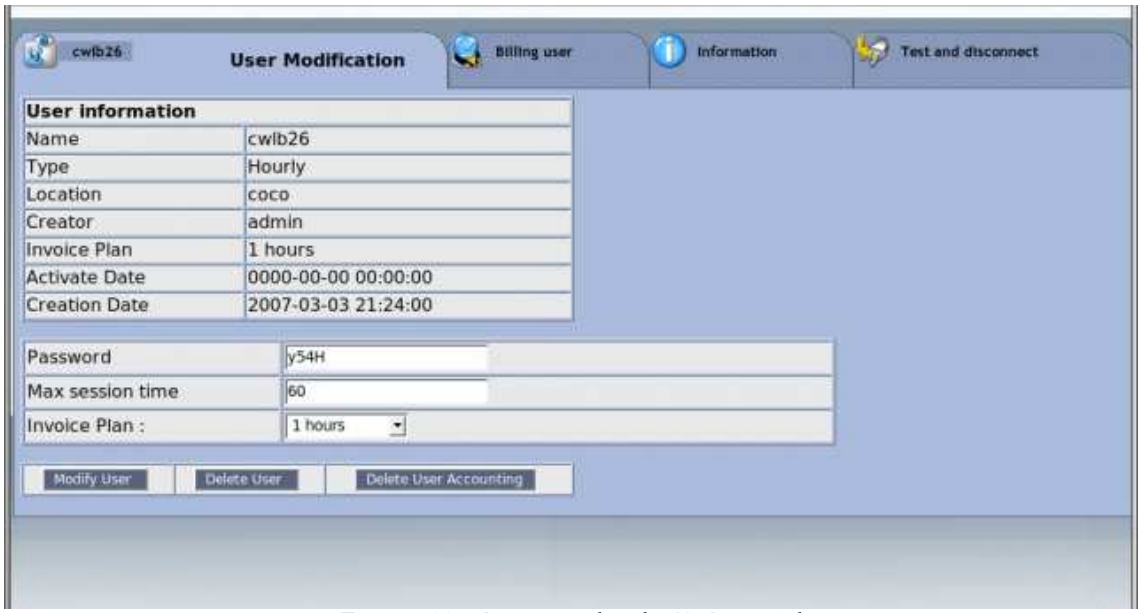

Figura 13 – Imagem do phpMyPrepaid

2.16.5 - phpRADmin

O phpRADmin, disponível em [23], é uma aplicação *Web* escrita em PHP para gestão de freeRADIUS. Esta aplicação permite a gestão de utilizadores, criação de grupos, armazenar informação da actividades dos utilizadores, fazer consultas avançadas na base de dados, verificar o estado do servidor, gestão de facturas, entre outros. Possui ainda geração de estatísticas e perfis de utilizadores. Está limitada em termos de interactividade e a sua adaptabilidade é também inexistente.

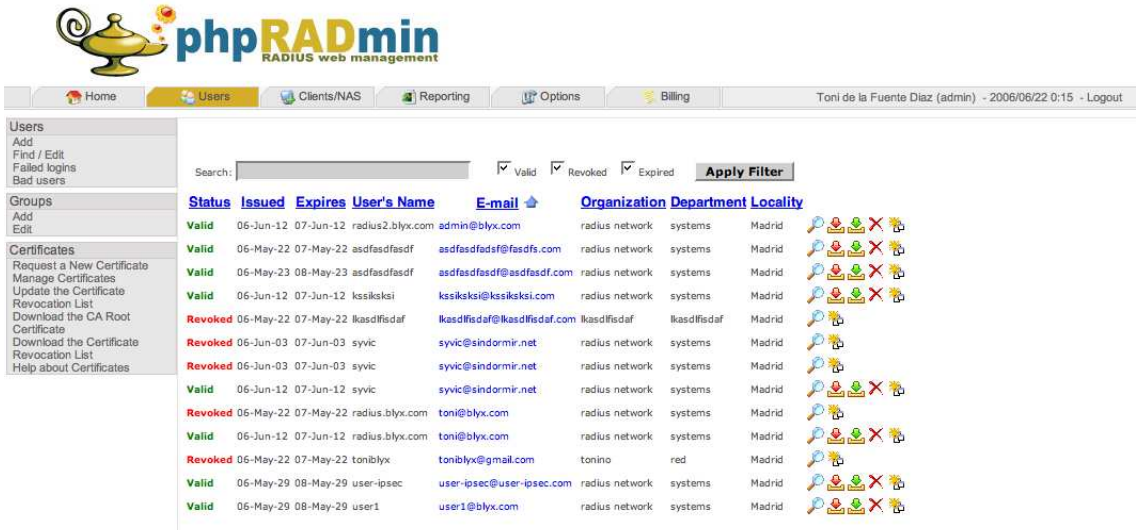

Figura 14 – Imagem do phpRADmin

2.17 - Comparação das Aplicações RADIUS Existentes

De seguida é apresentada uma tabela comparativa dos *frontends* existentes com alguns dos requisitos da aplicação a desenvolver.

Tabela 2- Comparação das Aplicações RADIUS Existentes

Aplicações	Funcionalidades	Facilidade	Interactividade	Adaptabilidade
dialupadmin	★	★★★★	★★	★★
daloRADIUS	★★★★	★★★	★★	★
ezRADIUS	★★	★★★★	★	★★
phpMyPrepaid	★★★	★★★	★★	★★
phpRADmin	★★★	★★★	★★	★★

Observando a tabela verifica-se que existem *frontends* com várias funcionalidades mais complexas e outros que se baseiam apenas nas funcionalidades mais básicas. Neste aspecto o phpRADmin e o daloRADIUS são as que apresentam mais funcionalidades. No entanto as que apresentam uma maior simplicidade apresentam por outro lado, como seria de esperar, uma maior facilidade de utilização. No caso da interactividade, o uso de JavaScript por parte do daloRADIUS torna-a mais interactiva. Todos os restantes apresentam uma interactividade reduzida com o utilizador. Todos os *frontends* falham no ponto da adaptabilidade, mostrando que nenhum deles consegue estar disponível para ser apresentado em qualquer tipo de terminal.

Da observação da tabela verifica-se, principalmente, a notória falta de *frontends* que forneçam a compatibilidade e adaptabilidade pretendida para a aplicação a implementar.

2.18 - Conclusão

Neste capítulo fez-se uma descrição do protocolo RADIUS, o qual apresenta um funcionamento com modelo cliente/servidor tendo, de um lado, o NAS como cliente e do outro o servidor RADIUS. O utilizador, o NAS e o servidor trocam mensagens entre si quando o utilizador se pretende autenticar para utilizar um determinado servidor de rede. Foram vistos também os vários tipos de mensagem que podem ser trocados e os tipos de serviços a que o utilizador pode ter acesso.

Foi verificado também que o RADIUS apresenta algumas limitações na sua utilização. O facto de os servidores RADIUS não guardarem o estado complica as soluções de gestão de recursos e sessões. Foram também identificados problemas de escabilidade, de performance e perda de dados quando o RADIUS é utilizado em grandes sistemas devido à falta de mecanismo de controlo de congestionamento.

Foi realizada uma análise sobre as várias implementações de RADIUS existentes e verificou-se que, segundo [1], o FreeRADIUS é o servidor RADIUS mais utilizado para sistemas Linux, sendo responsável pela autenticação de pelo menos um terço dos utilizadores na Internet. Os restantes utilizadores encontram-se divididos entre os restantes servidores, destacando-se entre eles o Cisco ACS e o Microsoft IAS.

Posteriormente foi estudada a evolução das técnicas de desenvolvimento *Web*, à qual se

designou de *Web* 2.0. A *Web* 2.0 não é uma nova versão da Web mas sim um desenvolvimento das técnicas de desenvolvimento *Web* que teriam como objectivo aproveitar mais potencialidades da *Web*. Através de um estudo apresentado em [9] verificou-se que esta evolução tornou vários negócios mais rentáveis.

Realizando uma análise a alguns estudos que tinham como finalidade a avaliação da performance das linguagens de programação Web verificou-se que a linguagem mais adequada à realização do projecto seria PHP uma vez que o trabalho tem como base uma ligação à uma base de dados de um servidor RADIUS e PHP é a melhor linguagem para bases de dados especificas e controlo sobre o acesso à base de dados, dado não ter O-R *Mapping* implementado de origem. No entanto, é de salientar que alguns dos estudos foram realizados há alguns anos e, por isso, o que foi retirado destes estudos pode, actualmente, não se verificar.

Neste capítulo introduziu-se ainda o funcionamento do modelo MVC que tem por objectivo a separação da lógica da aplicação da respectiva apresentação, permitindo desenvolver uma aplicação adaptável a vários tipos de terminais. A sua intenção é a de facilitar o desenvolvimento de aplicações com interfaces gráficas. Tem sido também utilizado com sucesso no desenvolvimento de modelos cliente-servidor e aplicações *Web*.

Verificou-se também que *framework* é um conjunto de conceitos usados para resolver um problema específico. Um *framework* compreende um conjunto de classes que unidas e implementadas, são usadas para auxiliar o desenvolvimento de uma funcionalidade genérica.

Foi efectuada ainda uma recolha dos *frameworks* que melhor se adequariam à implementação da aplicação. Foram descritas as suas vantagens e desvantagens e, de seguida foi efectuada a comparação entre estes *frameworks* tendo em conta alguns critérios de avaliação identificados através da análise aos objectivos do problema. Desta comparação concluiu-se que o CodeIgniter se apresentou como o framework mais adequado ao desenvolvimento da aplicação devido à sua boa flexibilidade e facilidade, suporte das linguagens necessárias para o desenvolvimento do projecto e ainda devido à sua muito boa performance.

Efectuou-se também uma análise comparativa acerca das várias aplicações para gestão de RADIUS já existentes, concluindo-se que nenhum deles obedece a todos os requisitos pretendidos para o desenvolvimento da aplicação pretendida neste trabalho evidenciando-se principalmente a falta de interactividade da aplicação com o utilizador e a inexistência de adaptabilidade desta aos diferentes tipos de terminais.

Capítulo 3

Análise e Implementação

Neste capítulo serão analisados e descritos os vários requisitos pretendidos para a aplicação. Este capítulo apresenta ainda a especificação da solução a desenvolver.

De início, será feita uma análise à descrição do problema feita previamente identificando todos os casos de uso pretendidos. Esta identificação e descrição dos casos de uso permite a identificação dos requisitos funcionais do sistema e das alterações que são necessárias à base de dados original do freeRADIUS.

Posteriormente irá ser descrito o novo modelo de dados especificando e explicando as diferenças para o modelo original do freeRADIUS. Irá ser descrita também a arquitectura do CodeIgniter, a *framework* selecionada para o desenvolvimento do trabalho, adicionando a esta as camadas que irão ser adicionadas com a implementação do trabalho. Estas modificações irão ser justificadas tendo em conta as alterações que se devem fazer ao nível da base de dados e tendo em conta o nível da aplicação a implementar. Depois de descritas as alterações efectuadas na base de dados original do freeRADIUS é descrita todo o modelo final da base de dados implementada.

Por fim, irá ser descrita a interface gráfica desenvolvida e respectivas vantagens desta.

3.1 - Casos de Uso

Como foi visto anteriormente, o RADIUS apresenta uma configuração bastante complexa no que toca à gestão de utilizadores e regras de acesso. É por isso necessário que a aplicação automatize estas tarefas. A aplicação deverá possuir a opção de gestão de utilizadores e também de gestão dos recursos associados a cada utilizador, ou seja, os recursos a que cada utilizador está autorizado, ou não, a aceder. Dentro da gestão de utilizadores deverá ser permitida a criação de grupos de utilizadores. Isto torna possível que os recursos possam ser atribuídos a um determinado grupo evitando a criação de regras de acesso para múltiplos utilizadores. E também necessário que a aplicação torne possível a criação de novos recursos, e novos tipos de recursos, que possam ser atribuídos a utilizadores ou grupos de utilizadores. Os acessos a recursos por parte dos utilizadores deverão ficar registados na base de dados de

modo a poderem ser realizadas estatísticas desses acessos.

A aplicação irá ter também outro tipo de utilizadores que são os utilizadores da aplicação. Daqui para a frente, estes utilizadores serão denominados de gestores de modo a não criar confusão com utilizadores do servidor de RADIUS. Para estes deverão ser disponibilizados diferentes tipos de perfis de acesso, ou seja, diferentes níveis de gestor, cada um deles com diferentes permissões associadas. Para os gestores a aplicação deverá possuir também as funcionalidades de criar e gerir grupos de modo a ser possível a atribuição de gestão de um determinado recurso a vários gestores (evitando fazê-lo individualmente). Os acessos e modificações realizadas por estes deverão ser também registados na base de dados.

Assim, os casos de uso resumem-se em:

- Login/registo de gestores;
- Pesquisa de utilizadores e gestores;
- Lista de confiança de gestores;
- Novos utilizadores;
- Gestão de utilizadores;
- Gestão de grupos;
- Gestão de membros dos grupos;
- Pesquisa de grupos;
- Gestão de clientes (ou NAS);
- Gestão de recursos;
- Atribuição de recursos a utilizadores e grupos;
- Estatísticas;
- Editar utilizadores;
- Contactar utilizadores.

3.2 - Arquitectura

A Figura 15 representa a arquitectura do sistema desenvolvido tendo identificados, a cor-de-laranja, os módulos implementados.

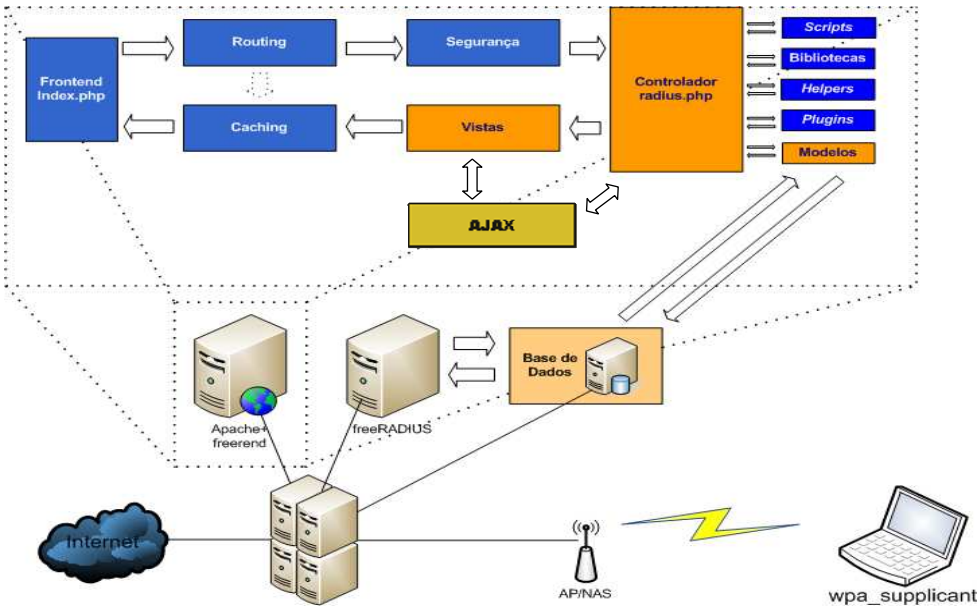

Figura 15 - Arquitectura da aplicação

O index.php, na abertura da aplicação, inicializa o processamento do CodeIgniter. O *Routing* processa o pedido HTTP para decidir o que fazer. No caso de existir um ficheiro em *cache*, este é enviado directamente para o *browser*. No caso de não existir, o pedido é enviado para o controlador da aplicação. No entanto, antes de chegar ao controlador, os pedidos são todos filtrados por razões de segurança. O controlador da aplicação processa então o pedido carregando o modelo, bibliotecas, *plugins* e outros recursos que sejam necessários para a resposta do pedido. O controlador irá fazer também a comunicação com a base de dados do freeRADIUS através do mecanismo ORM. Esta base de dados terá as alterações à base de dados original do freeRADIUS já implementadas.

Por forma a completar o pedido, a vista correspondente é enviada para o *browser* do cliente para ser observada. Se o sistema de *caching* estiver activo, a vista é guardada para evitar carregar o servidor com os mesmos pedidos e de forma a aumentar a velocidade de processamento de páginas do lado do cliente quando este acede a páginas já observadas.

3.3 - Alterações na Base de Dados do freeRADIUS

De modo a poder ser feita a implementação dos requisitos identificados é imperativo efectuar alterações na base de dados original do freeRADIUS. A figura 16 apresenta o diagrama da base de dados original integrada com a base de dados da aplicação. Nesta faltam duas das tabelas originais do freeRADIUS que apresentam funções de registo e, por isso, são de baixa relevância.

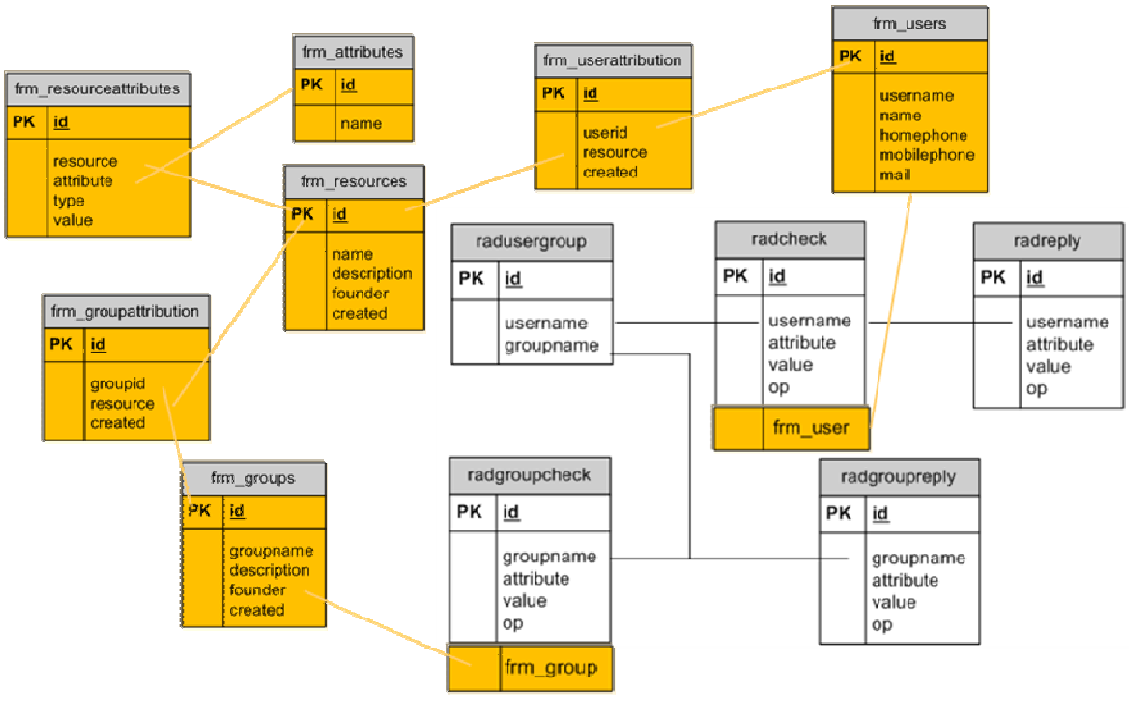

Figura 16 – Modelo de dados da aplicação

Para permitir a existência do conceito de gestores é necessário acrescentar uma tabela na base de dados que permita a inserção deste tipo de utilizadores. Juntamente com esta, e de modo a permitir a criação de grupos de gestores, será necessária a criação de uma outra tabela de grupos, e uma relação entre a tabela de grupos e a de gestores. A delegação de

gestores pelos vários recursos está dependente de uma relação entre a tabela de gestores e a tabela dos recursos que terá de ser inserida através de uma chave estrangeira. Deverá ser também inserida uma tabela que relacione os gestores com a data e registando as alterações efectuadas por cada um.

Assim, a tabela frm_users é a tabela onde são inseridos os dados dos gestores. De modo a integrar esta tabela no modelo do freeRADIUS foi inserida uma chave estrangeira na tabela radcheck a indicar qual a linha da tabela frm_users que corresponde a cada utilizador. Esta tabela apresenta como campos o nome do gestor, o seu contacto de telefone, o seu contacto de e-mail, a data em que foi criado o gestor, contagem de *logins*, o nível de permissão do utilizador e uma *hash* que é criada aquando do registo do gestor para posterior confirmação.

A tabela frm_groups especifica os grupos de utilizadores e gestores. Esta permite adicionar informação como descrição do grupo, responsável pelo grupo e a data de criação. É integrada com o modelo do freeRADIUS adicionando uma chave estrangeira à tabela radgroupcheck indicando qual a linha de dados correspondente a cada grupo.

A associação entre grupos e utilizadores continua a ser feita pela tabela original do freeRADIUS, radusergroup. No entanto foram adicionados a esta três campos adicionais que representam o identificador do grupo a que se atribui o utilizador para simplificar as pesquisas, a permissão do utilizador no grupo e ainda a data de atribuição do utilizador ao grupo.

Outra tabela adicionada foi a tabela de recursos, de nome frm_resources. Nesta é definido o conceito de recurso/serviço e a implementação desta simplifica a atribuição de um utilizador a um recurso e ainda a atribuição da gestão de um determinado recurso a um gestor. Na tabela são guardados o nome do recurso, a descrição do mesmo, a sua data de criação e o responsável pelo recurso.

No novo modelo de dados existem ainda duas tabelas que funcionam apenas para guardar atributos e níveis de permissão e nas quais a inserção de dados é proibida através da introdução de uma regra em SQL. Estas são a tabela frm_permissionslevel e a tabela frm_attributes. A primeira define cinco níveis de permissão: *owner*, que possui todas as permissões; *master*, que não pode apagar ou editar grupos e recursos que não os seus; *admin*, que não pode apagar ou editar qualquer informação que não seja a sua; *user,* que apenas pode observar as informações e verificar os utilizadores dos grupos; *basic*, que apenas pode utilizar os recursos; e é usada para atribuir permissões a gestores na aplicação, permissão aos membros de cada grupo e permissão de cada gestor junto de cada recurso. Na segunda encontram-se guardados todos os atributos possíveis do freeRADIUS e permite a adição desses atributos a cada recurso. A atribuição dos atributos a cada recurso é guardada na tabela frm_resourceattributes. Esta tabela além de fazer uma relação entre cada recurso, os seus atributos e os respectivos valores, define ainda de que tipo de atributo se trata, se é um atributo ao qual o pedido deve obedecer ou se é um atributo que será enviado como resposta no *Access-Accept*.

Para a delegação de recursos foram criadas outras duas tabelas, frm_userattribution e frm_groupattribution. A tabela frm_userattribution destina-se à atribuição de recursos a utilizadores e à delegação desses recursos a um determinado gestor. Ao ser adicionado um recurso a um utilizador, todos os atributos aos quais os pedidos de autenticação terão de obedecer são colocados na tabela radcheck com o nome do utilizador no campo destinado para ele. Para a atribuição de recursos a grupos existe a tabela frm_groupattribution. Ao ser adicionado um recurso a um determinado grupo, os atributos que representem atributos para

o *Access-Request* são colocados na tabela radgroupcheck. Os atributos que representam atributos a serem enviados juntamente com o *Access-Accept* são colocados na tabela radgroupreply.

Um outro conceito adicionado pela aplicação é o de lista de utilizadores confiáveis. Esta lista permite que um determinado gestor tenha vários gestores da sua confiança que podem gerir os recursos que estão a cargo desse gestor. A tabela criada para este fim tem por nome frm_trustedusers e apresenta como campos o identificador do próprio gestor, o identificador do gestor de sua confiança e a data em que foi adicionado.

Por fim, foi ainda acrescentada uma tabela com o nome frm_log que regista as operações efectuadas na aplicação tais como registo de utilizadores, registo de grupos, edição de grupos, entre outros.

3.4 - Interface

Nesta secção irá ser dada uma explicação acerca da interface e dos passos dados na sua implementação.

De acordo com os objectivos definidos, a aplicação deverá ser compatível com os vários tipos de *browser*, apresentar uma interface adaptável a todo o tipo de terminais e sem requerer muita capacidade de processamento do lado do cliente de modo a não prejudicar a visualização em alguns terminais. Estes problemas são resolvidos em grande parte na implementação da interface.

Assim, a interface desenvolvida é apresentada na Figura 17.

Figura 17 - Interface Gráfica

Esta interface usa um sistema de três colunas, definido em [24], com tamanhos definidos por percentagens que resolve, ao mesmo tempo, o problema da adaptabilidade e da compatibilidade uma vez que toda ela é implementada com recurso apenas a CSS. A interface permite ainda inserir tópicos de ajuda ao utilizador em cada página usando a terceira coluna.

Esta interface apresenta uma imagem limpa contendo dois menus de forma a simplificar a navegação na aplicação. O menu horizontal apresenta as opções principais tocando ao menu vertical a apresentação dos atalhos de cada funcionalidade inerente a cada opção.

3.5 - Diagrama de Páginas

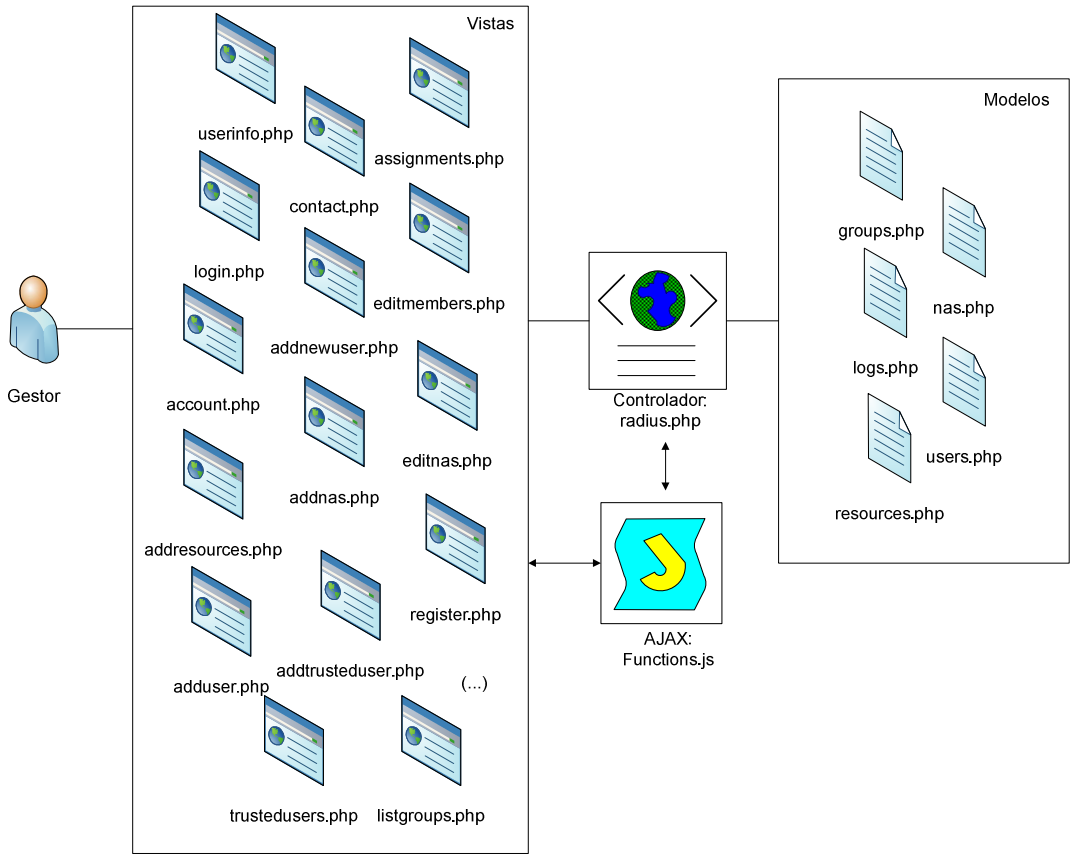

Figura 18 - Diagrama de páginas da aplicação

Na Figura 18 é mostrado o diagrama de páginas da aplicação. No centro encontra-se o principal módulo da aplicação, que é o controlador. Este é um conjunto de inúmeras funções que correspondem à lógica das diversas páginas e funcionalidades da aplicação. Quando o controlador é invocado faz a interligação com os modelos necessários que, por sua vez, efectuam consultas e actualização de dados na base de dados da aplicação. Quando é efectuada toda a lógica da função correspondente é invocada a vista que apresenta o resultado da função invocada. Em cada visualização da aplicação existem também algumas funcionalidades que não requerem a actualização inteira da página. Estas funcionalidades são implementadas recorrendo a um módulo AJAX que efectua a interligação entre as vistas e o controlador e efectua troca de dados entre estes módulos, assíncronamente.

3.6 - Conclusão

Este capítulo descreveu a especificação da solução a desenvolver. Foram identificados os casos de uso pretendidos para a aplicação. É de salientar que a aplicação deverá automatizar todas as funções do RADIUS de modo a exigir a mínima configuração possível.

Neste capítulo foi ainda descrito o novo modelo da base de dados especificando as alterações efectuadas na base de dados original do freeRADIUS e todas as novas tabelas implementadas. Além disto foi ainda descrita a arquitectura do funcionamento da aplicação especificando a integração entre os diversos módulos do *framework* e a sua integração com a base de dados.

Foi ainda demonstrada a interface gráfica desenvolvida. Contatou-se que a interface gráfica desenvolvida contorna desde logo os problemas de compatibilidade e de adaptabilidade devido ao facto de ser toda ela implementada recorrendo apenas a CSS e de os seus tamanhos serem definidos através de percentagens. O sistema de três colunas permite ainda a inserção de tópicos de ajuda em cada página. Esta apresenta ainda um aspecto limpo e uma navegação simples.

Por fim, foi demonstrado o diagrama de páginas contendo os diversos módulos da aplicação. Essencialmente a aplicação baseia-se nos três módulos definidos no modelo MVC: controlador, modelos e vistas. Adicionalmente existe ainda um módulo AJAX que efectua a troca assíncrona de dados entre as vistas e o controlador.

Capítulo 4

Testes e Resultados

Neste capítulo serão apresentados os testes realizados à aplicação, resultados obtidos e discussão dos mesmos. Inicialmente será descrito o cenário de testes e os respectivos passos de configuração necessários. Após a descrição do cenário de testes funcionais serão descritos os resultados obtidos através do uso da aplicação, mais concretamente os resultados da utilização das diversas funcionalidades assim como os resultados da compatibilidade com diversos *browsers Web* e da adaptação a diversos tipos de terminais. Por fim, irão ser discutidos estes resultados de forma a verificar se os objectivos a que a aplicação teria de obedecer são verificados.

4.1 - Cenário de Testes

De modo a permitir a realização de testes a fim de validar todos os objectivos identificados para a aplicação foi implementado e configurado um cenário de testes. Este cenário é ilustrado na seguinte figura:

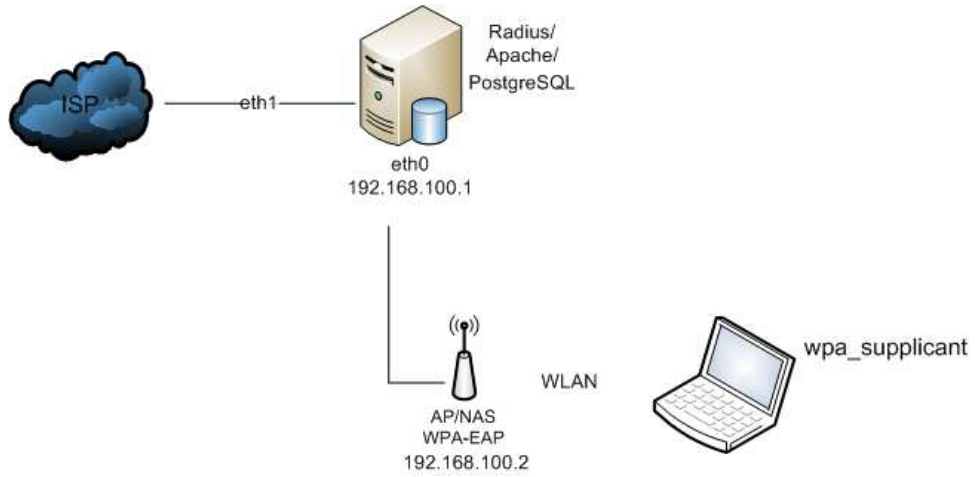

Figura 19 – Cenário de Testes

Foi configurada uma máquina, que utiliza como sistema operativo o Ubuntu 9.04, a funcionar como servidor de RADIUS, instalando nela o freeRADIUS. Esta máquina foi ligada à Internet através da interface eth1 e, através da interface eth0, fará a ligação com o *Access Point* (AP). O AP foi configurado de modo a consular o servidor RADIUS para validar os acessos a este. A máquina foi também configurada como servidor *Web,* onde foi alojada a aplicação implementada. No cenário existe um terminal portátil que, representa um utilizador do sistema e que, recorrendo à aplicação wpa_supplicant, irá efectuar pedidos de autenticação ao AP.

4.2 - Configuração

Após se efectuar as ligações físicas de acordo com o disposto na figura, procedeu-se às configurações do servidor, AP e wpa_supplicant. Começou-se por configurar os endereços de cada interface de acordo com a figura. Foi instalado no servidor, o Apache, para alojar a aplicação desenvolvida. Neste documento não se encontra descritos os passos de configuração das interfaces e instalação e configuração do Apache, em virtude de esta ser trivial e poder ser facilmente encontrada na *Web*. Foi também instalado e configurado um servidor RADIUS, o freeRADIUS.

4.2.1-Instalação e Configuração do freeRADIUS

Existem vários modos de instalar o freeRADIUS. Devido a problemas de licenças a instalação não deverá ser feita utilizando os repositórios do Linux uma vez que estes não possuem suporte OpenSSL, não suportando autenticações via métodos baseados em TLS (*Transport Layer Security*). Assim, o método aconselhado é efectuar *download* da versão mais actual directamente do *Website* do freeRADIUS e compilar na máquina. Isto permite a instalação do freeRADIUS na máquina. De seguida, foi necessário configurar os ficheiros do freeRADIUS para permitir os diferentes tipos de autenticações.

Primeiro alterou-se o tipo de EAP (*Extensible Authentication Protocol*) que se encontra por omissão no ficheiro “eap.conf”. A linha default_eap_type = md5 foi alterada para default_eap_type = peap.

Neste ficheiro foi necessário ainda descomentar outras linhas que são importantes para o uso dos mecanismos baseados em TLS. No módulo ttls e peap foram também alteradas as seguintes linhas:

copy_request_to_tunnel = no
use_tunneled_reply = no

para:

copy_request_to_tunnel = yes
use_tunneled_reply = yes

Esta alteração faz com que os dados enviados no pedido do cliente sejam copiados para o túnel. Isto é importante em passos posteriores, por exemplo, para restringir o acesso a um único AP. Tipicamente, neste ponto de configuração do freeRADIUS, configura-se os

utilizadores e clientes no freeRADIUS. No entanto dado que se vai utilizar uma base de dados PostgreSQL com o freeRADIUS estas tarefas não foram necessárias. Assim, deve-se começar por definir que o freeRADIUS deve rejeitar, por omissão, utilizadores que não apresentem um "Auth-Type" definido. Para tal alterou-se o ficheiro "users" adicionando a seguinte linha:

DEFAULT Auth-Type := "Reject"

De seguida foi configurada a ligação à base de dados. Para tal foi instalado o servidor de base de dados PostgreSQL e a base de dados utilizada pelo freeRADIUS. Para isto seguiu-se os seguintes passos:

> sudo apt-get install postgresql postgresql-client postgresql-contrib

O passo anterior instalou o servidor e cliente postgreSQL e algumas utilidades extra. De modo a ser mais simples a observação dos dados e executar *querys* instalou-se também uma aplicação para gestão das bases de dados executando o seguinte comando:

> sudo apt-get install pgadmin3

Neste ponto foi necessário apagar e alterar a palavra passe para a conta 'postgres' no servidor PostgreSQL. Para tal foram executados os seguintes comandos:

> sudo su postgres -c psql template1
template1=# ALTER USER postgres WITH PASSWORD 'password';
template1=# \q

Foi necessário fazer o mesmo para o utilizador 'postgres' do Unix:

> sudo passwd -d postgres
> sudo su postgres -c passwd

Neste ponto foi utilizada a mesma palavra passe usada anteriormente. De seguida, para habilitar a gestão e autenticação com o pgAdmin executou-se o seguinte comando:

> sudo su postgres -c psql < /usr/share/postgresql/8.3/contrib/adminpack.sql

O passo seguinte consistiu em configurar o servidor postgreSQL definindo quais os locais de onde pode receber pedidos de ligação (a partir de qualquer endereço) e habilitando a encriptação da palavra passe. Para isto, editou-se o ficheiro de configuração do postgreSQL, "postgresql.conf". Aqui alterou-se a seguinte linha:

#listen_addresses = 'localhost'

para:

listen_addresses = '*'

No mesmo ficheiro descomentou-se também a seguinte linha:

password_encryption = on

Após estas alterações deve-se reiniciar o servidor de forma a assumir as alterações de configuração efectuadas:

> sudo /etc/init.d/postgresql-8.3 restart

De seguida foi instalada a base de dados usada pelo freeRADIUS. Para isto executaram-se os seguintes comandos:

> su - postgres
> createuser radius --no-superuser --no-createdb --no-createrole -P

O passo seguinte consistiu em configurar uma palavra passe segura para acesso à base de dados.

> createdb radius --owner=radius
> exit
> cd /usr/share/doc/packages/freeradius/doc/examples/
> psql -U radius radius < postgresql.sql

Neste ponto configurou-se o freeRADIUS de forma a utilizar a base de dados. Para isto foi necessário efectuar mais alterações nos ficheiros de configuração do freeRADIUS.

O ficheiro responsável pela ligação à base de dados é o "sql.conf". A linha database = "mysql" foi alterada para database = "postgresql". O seguinte conjunto de linhas indica as credenciais de acesso à base de dados. Foram alteradas em conformidade com os dados inseridos na criação da base de dados.

server = "localhost"
login = "radius"
password = "radpass"

Neste ficheiro descomentou-se ainda as seguintes linhas:
#read_groups = yes
#readclients = yes
#nas_table = "nas"

Estas linhas permitem ao freeRADIUS a leitura de grupos e clientes na base de dados. De modo a permitir a definir os mecanismos que acedem à base de dados foi também necessário alterar o ficheiro "default" presente na pasta "sites-enabled" do freeradius. Nos módulos "authorize", "accounting", "session" e "post-auth" descomentou-se a linha "#sql". No módulo "authorize" foi também descomentada a linha "#checkval" que define o uso desse módulo, o qual foi necessário posteriormente. Algumas linhas também foram descomentadas

no ficheiro “inner-tunnel” o qual é responsável pelos túneis dos métodos baseados em TLS. Nos módulos “authorize”, “accounting”, “session” e “post-auth” foram descomentadas, tal como no passo anterior, as linhas “#sql”. Foi também descomentada a linha “#checkval” no módulo “authorize”.

Foi reiniciado o servidor RADIUS por forma a este assumir as alterações de configuração realizadas. Para isto executou-se o seguinte comando:

> sudo /etc/init.d/freeradius restart

O freeRADIUS pode ainda ser executado pode forma a mostrar as trocas de dados à medida que ocorrem. Para isto dever-se-á executar os seguintes comandos:

> sudo /etc/init.d/freeradius stop
> sudo freeradius -X

Concluídos estes passos, a configuração do servidor RADIUS foi finalizada.

Neste ponto foi executado o *script* implementado para alteração da base de dados. Por forma a aceitar pedidos do AP foi adicionado o mesmo à lista de clientes autorizados do freeRADIUS. Para tal foi utilizada a aplicação desenvolvida. Esta operação foi realizada acedendo à função “Resources” e ao atalho “Add NAS” da aplicação. Após adicionar o cliente foi reiniciado o freeradius uma vez que este apenas lê os clientes quando é iniciado.

4.2.2- Configuração do Access Point

Antes de configurar o AP para consultar o servidor foi atribuído a este um endereço fixo de acordo com o que é mostrado na Figura 18. Por forma a que o AP consulte o servidor RADIUS com a finalidade de validar as autenticações configurou-se a sua segurança *Wireless* com o WPA-EAP e TKIP como tipo de cifra. Configurou-se o servidor RADIUS com o endereço 192.168.100.1 e com o segredo que foi escolhido quando foi adicionado o cliente.

4.2.3- Configuração do Terminal

Foi instalado e configurado o wpa_supplicant no terminal por forma a permitir a sua autenticação no AP. Para tal, foi criado um ficheiro de configuração de nome wpa-eap.conf com os seguintes dados:

```
ctrl_interface=/var/run/wpa_supplicant

network={
    ssid="nome da ligação"
    key_mgmt=WPA-EAP
    eap=PEAP
    pairwise=TKIP
    group=TKIP
    identity="admin"
    password="admin"
}
```

Para correr o wpa_supplicant usando o ficheiro de configuração criado executou-se o seguinte comando:

> wpa_supplicant -i ath0 -c wpa-eap.conf -d

4.3 - Resultados Obtidos

Esta secção apresenta e discute os resultados obtidos utilizando a aplicação desenvolvida no cenário de teste. Em primeiro lugar, foram validados os requisitos funcionais identificados anteriormente. Para tal foi utilizada a aplicação para testar todos as funcionalidades desta e, efectuando tentativas de ligação e verificação dos dados da base de dados foram validadas essas funcionalidades. De seguida foram validadas a adaptabilidade a diferentes tipos de terminais e a compatibilidade a diferentes *browsers*. Para validar a adaptabilidade foram utilizados emuladores de terminais móveis tais como *Android, iPhone*, entre outros, para abrir e utilizar a aplicação desenvolvida. Para verificar a compatibilidade, a aplicação foi testada com os principais *browsers web* existentes.

4.3.1 - Funcionalidades

Os resultados da validação das várias funcionalidades existentes são apresentados de seguida.

4.3.1.1 –Login/Registo

Ao entrar na aplicação o gestor terá de se autenticar de modo a obter o acesso às funcionalidades pretendidas. Esta autenticação deverá ser feita usando um conjunto nome de utilizador e palavra passe, tal como se observa na Figura 20:

Figura 20 – Demonstração do Login

Após inserir os dados, a aplicação verifica na base de dados se existe um nome de utilizador igual ao digitado e se a palavra passe está correcta. Caso não exista o utilizador na base de dados ou a palavra passe não esteja correcta a aplicação devolve a mesma página com o respectivo erro. Caso as credenciais estejam correctas a aplicação direcciona o utilizador para a página principal onde indica as várias opções das quais o utilizador dispõe.

No caso de um gestor não se encontrar registado no sistema, este pode-se registar usando

a aplicação. Esta opção foi implementada com o intuito de automatizar os processos de registo de gestores. Ao registar-se fica com as permissões mais básicas ficando a cargo do administrador do sistema ou gestores com permissões mais elevadas a alteração das permissões deste.

Figura 21 – Demonstração do Registo

Tal como é possível observar na Figura 21, os dados requisitados para registo do gestor é nome de utilizador, palavra passe desejada e endereço electrónico. A aplicação não permite a duplicação de nomes de utilizadores ou endereços de e-mail. Após o registo a aplicação envia um e-mail de modo a pedir ao utilizador a confirmação da sua conta. Para isto é enviado no *mail* um *link* contendo uma *hash* calculada de forma a ser única ao qual o utilizador terá de aceder de modo a validar a sua conta.

Com a submissão do registo o utilizador fica também registado na base de dados no RADIUS, no entanto fica, por defeito, proibido a aceder a qualquer dos recursos do RADIUS. De modo a validar esta funcionalidade, testou-se a autenticação a partir do terminal portátil junto do AP e verificou-se que o AP recebe um *Access-Reject* do servidor RADIUS não fornecendo por isso acesso ao terminal.

Figura 22 – Demonstração da Página Principal

Ao autenticar-se junto da aplicação com o nome de utilizador criado por defeito ("admin") verificou-se, tal como se observa na Figura 22, que a aplicação redirecciona para a página principal.

As seguintes demonstrações das funcionalidades serão feitas usando este utilizador.

4.3.1.2 -Pesquisa de Utilizadores

Uma das funcionalidades relativas aos utilizadores é a pesquisa de utilizadores. Esta permite encontrar os utilizadores registados no sistema de modo a adicioná-los à sua lista de confiança, adicionar a um determinado grupo e, se tiver nível de permissão suficiente, alterar as suas permissões e até apagá-los do sistema. Na figura seguinte pode observar-se a funcionalidade de pesquisa de utilizadores.

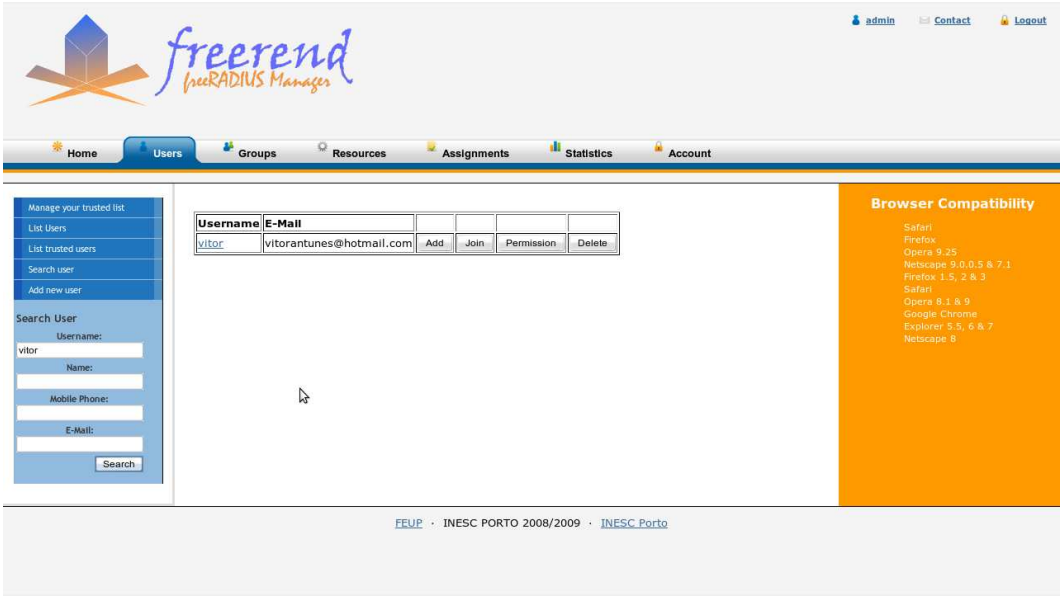

Figura 23 - Demonstração da pesquisa de utilizadores

Como se pode observar na Figura 23 não é obrigatório a inserção de todos os campos da pesquisa. Não é também necessário completar os campos que se pretende pesquisar, isto é, mesmo que se saiba apenas parte do nome de utilizador, nome ou endereço electrónico pode-se pesquisar pelo que se sabe que o sistema fornece os resultados que apresentam essa parte.

Nos resultados da pesquisa é ainda possível aceder aos dados de cada utilizador pressionando em cima do respectivo nome de utilizador.

4.3.1.3 -Lista de Confiança

Esta funcionalidade tem por objectivo permitir a atribuição de recursos a gestores presentes na lista de utilizadores de confiança por forma a permitir que estes estejam também responsáveis por esses recursos.

De modo a adicionar um utilizador à lista de confiança de um determinado gestor é necessário pesquisar o utilizador pretendido e, de seguida, pedir para adicionar à sua lista de confiança. Feito isto, o outro utilizador tem o pedido na sua lista de pedidos podendo este aceitar ou rejeitar o pedido. O pedido fica também na lista do utilizador que enviou o pedido para o caso de o pretender cancelar. Após aceitar o utilizador fica na sua lista de confiança,

tal como se pode observar na Figura 24.

Figura 24 – Demonstração dos estados da lista de confiança

Na figura observa-se os vários estados dos pedidos para a lista de confiança. Verifica-se um utilizador já adicionado à lista, um pedido por parte do gestor autenticado para um outro, e um pedido de outro utilizador para o gestor autenticado.

4.3.1.4 –Adicionar Novos Utilizadores

Se o gestor apresentar as permissões suficientes pode também adicionar novos utilizadores do RADIUS e da aplicação. Esta funcionalidade é idêntica ao registo mas nesta opção o utilizador fica automaticamente com um nível de permissão superior. Após o registo os utilizadores ficam pendentes até confirmação do endereço de endereço electrónico. Tal como no registo, este utilizador fica também registado na base de dados do RADIUS.

4.3.1.5 –Adicionar Novos Grupos

Um gestor pode também adicionar novos grupos apenas tendo de apresentar o nome e a descrição do grupo. Este grupo fica automaticamente registado no RADIUS mas sem qualquer recurso associado. Na Figura 25 pode-se ver a fase de registo de um grupo.

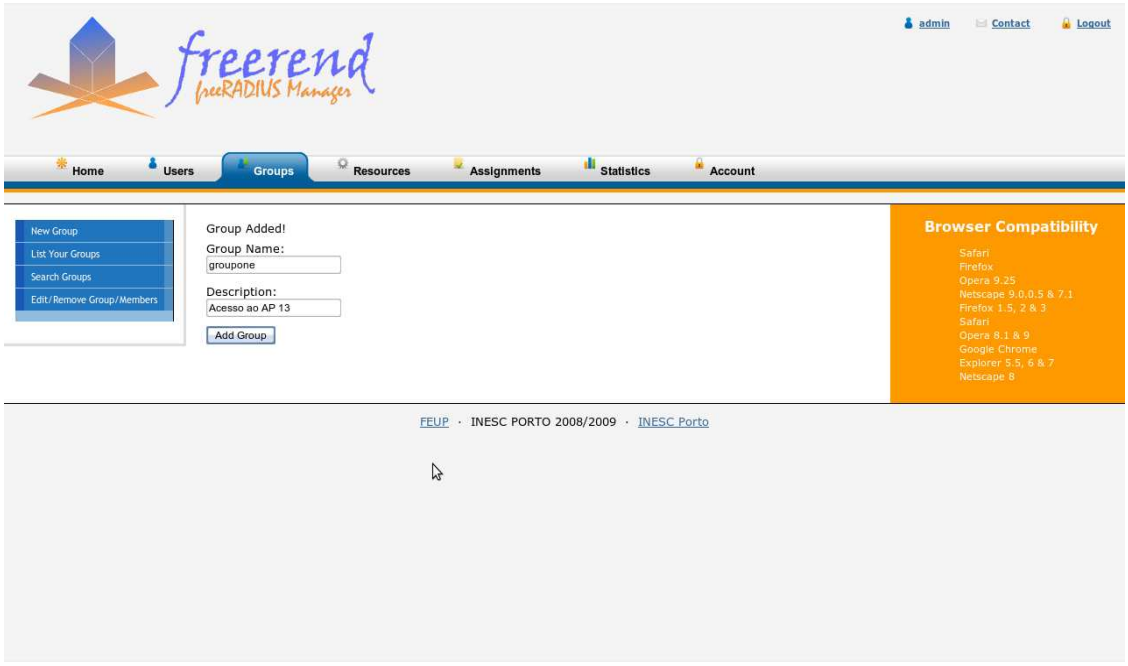

Figura 25 – Demonstração de criação de grupo

4.3.1.6 –Editar/Remover Grupos

Se for o criador do grupo, ou se este tiver dado as permissões suficientes, um gestor pode editar ou ainda remover um grupo do sistema. Os grupos do RADIUS são também automaticamente actualizados quando editados. Esta funcionalidade é apresentada na Figura 26.

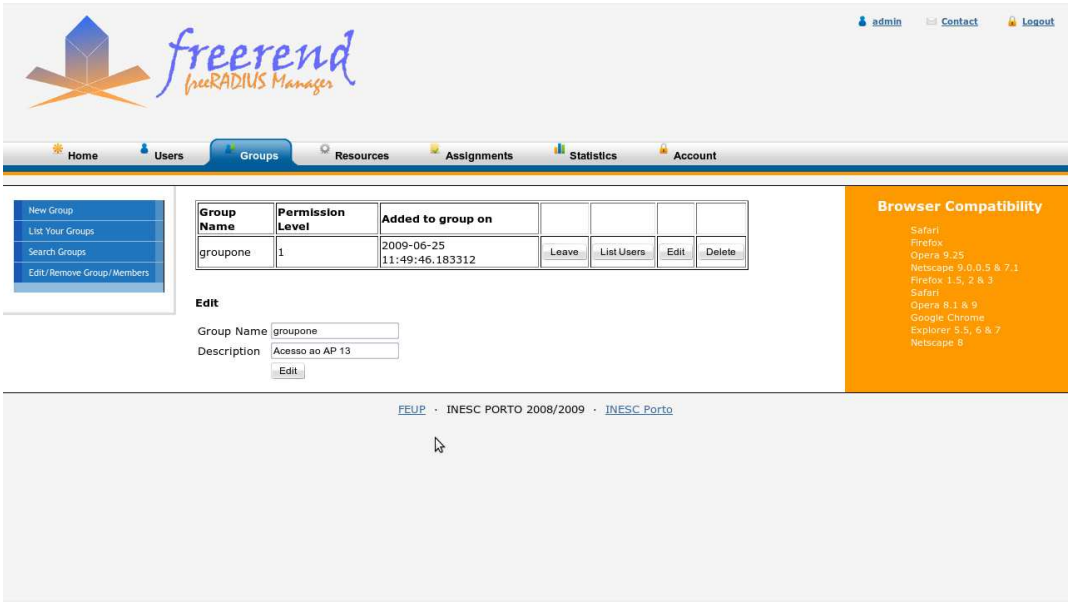

Figura 26 – Demonstração da edição dos dados de um grupo

4.3.1.7 –Listar/Editar/Remover Membros de Grupos

Dentro dos grupos, um gestor pode ainda listar os utilizadores de um determinado grupo, desde que possua as permissões suficientes, sair do grupo e editar a permissão de outros utilizadores. Todas as alterações são efectuadas também nos dados do freeRADIUS. A Figura 27 mostra esta funcionalidade.

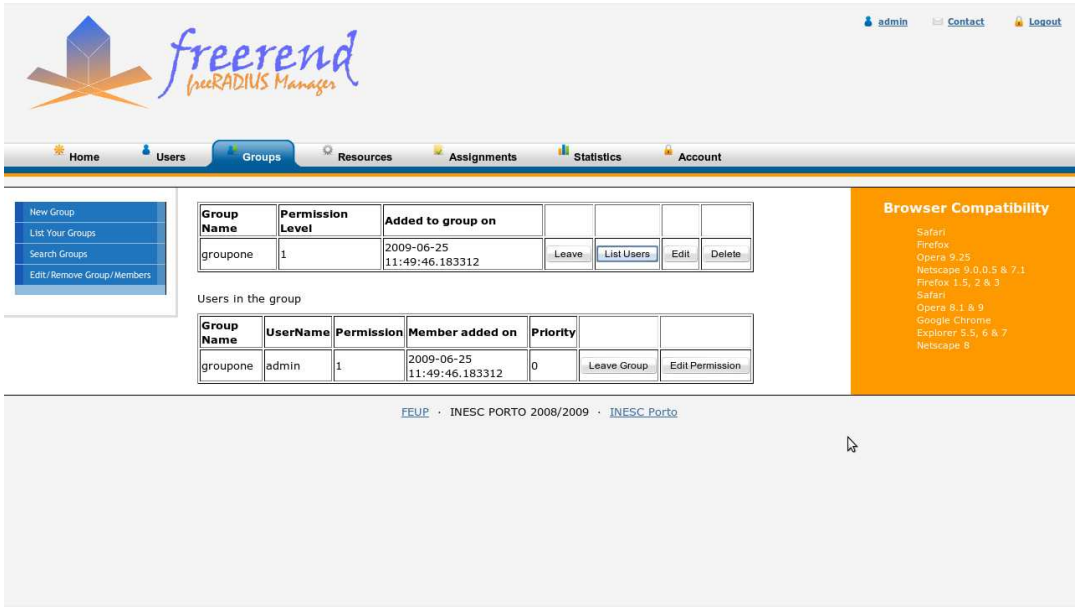

Figura 27 – Listar/Editar/Remover membros de grupos

4.3.1.8 –Pesquisa de Grupos

Um gestor pode ainda pesquisar grupos através do nome, descrição e/ou criador do grupo. Tal como na pesquisa de utilizadores basta colocar parte dos campos para a aplicação devolver os resultados que se assemelham aos dados pesquisados. Usando a pesquisa de grupos um utilizador pode-se, posteriormente, propor a entrar no grupo, podendo os gestores do grupo permiti-lo, ou não. Esta funcionalidade é apresentada na Figura 28.

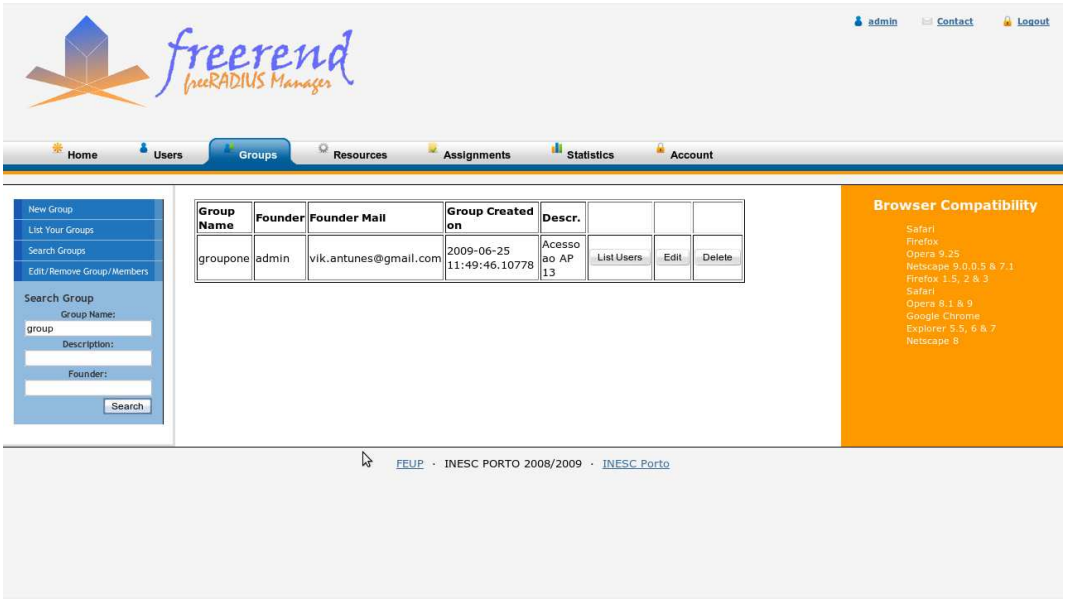

Figura 28 – Demonstração da pesquisa de grupos

4.3.1.9 –Adicionar/Editar/Remover NAS (ou clientes)

Com esta funcionalidade, que apenas se encontra disponível para os gestores que apresentem permissões de administração, um gestor pode adicionar clientes de RADIUS como, por exemplo, um AP. Os campos mais importantes nesta funcionalidade são o nome do AP que representa, normalmente, o seu endereço IP, o seu *Short Name* e o seu segredo, o qual permite o acesso do cliente ao servidor RADIUS. Os clientes adicionados apenas são detectados pelo RADIUS após se reiniciar o servidor uma vez que apenas são lidos no início da aplicação. Esta funcionalidade é apresentada na Figura 29.

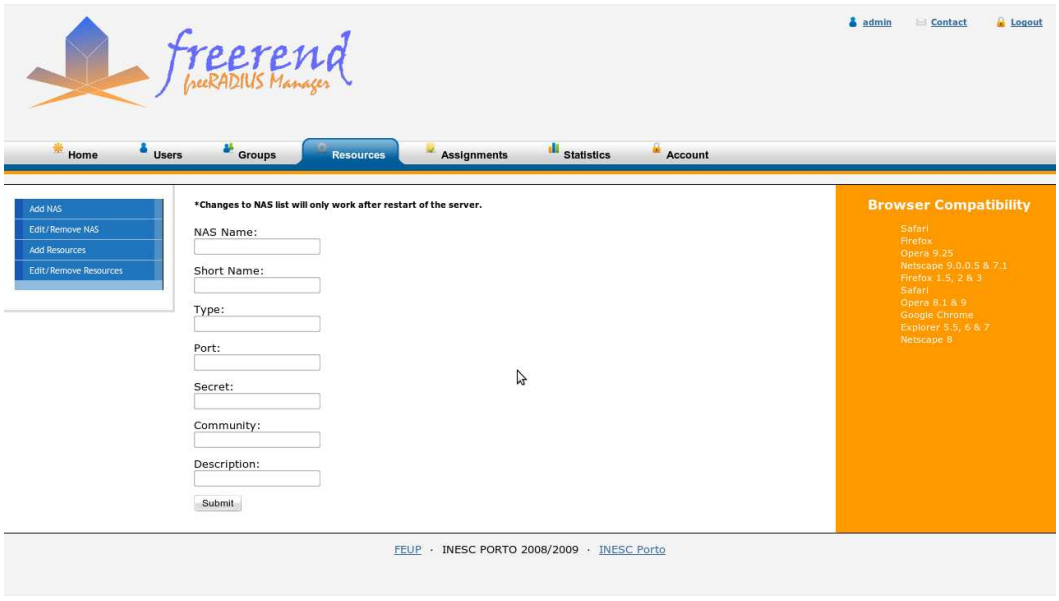

Figura 29 – Demonstração da criação de um cliente

Um gestor que possua as permissões suficientes pode ainda editar os dados de um

determinado cliente ou mesmo remover esse cliente. Esta funcionalidade é apresentada na Figura 30.

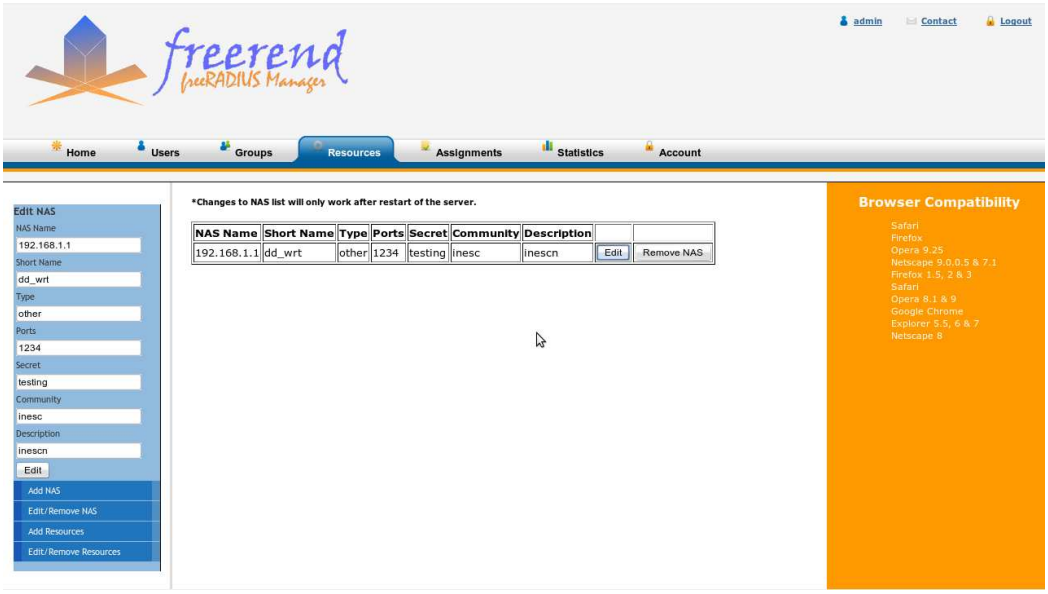

Figura 30 – Demonstração da edição de um cliente

De modo a testar esta funcionalidade removeu-se quaisquer clientes existentes na aplicação. De seguida, reiniciou-se o servidor RADIUS e testou-se a autenticação a partir do terminal utilizando um utilizador existente. Consultando a informação dada pelo freeRADIUS verificou-se que os pedidos enviados pelo AP eram ignorados pelo freeRADIUS uma vez que este não se encontrava registado na base de dados. De seguida, voltou-se a adicionar o AP e reiniciou-se o freeRADIUS. Voltou-se a testar a autenticação do cliente e verificou-se desta vez que o servidor recebe e processa os pedidos, o que significa que permitiu, com sucesso, o registo do AP.

4.3.1.10 –Adicionar Recursos

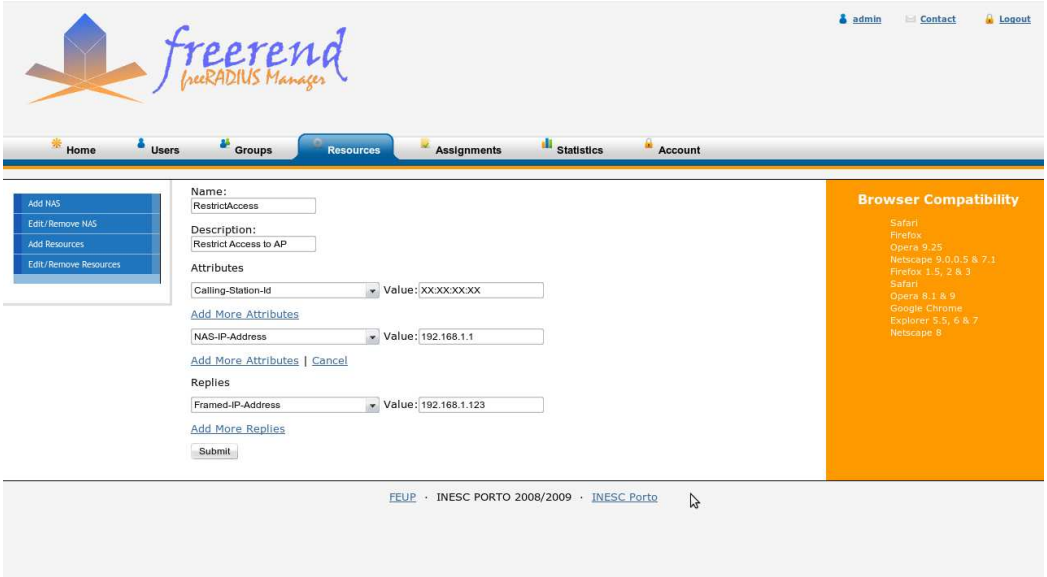

Figura 31 – Demonstração da criação de um recurso

Esta funcionalidade permite a associação do conceito de recursos com os atributos do

RADIUS. Um atributo é visto como sendo, por exemplo, um acesso a um AP. Ao adicionar um novo recurso um gestor pode associar dois tipos de atributos: atributos que deverão ser verificados para permitir um determinado utilizador aceder a esse recurso; e atributos que serão enviados como resposta após a autenticação no servidor.

Esta funcionalidade é apresentada na Figura 31.

Ao criar um recurso o gestor escreve o nome e descrição do recurso, os atributos para verificação do acesso e ainda os atributos para envio com a permissão de acesso. Estes dados apenas são guardados nos dados da aplicação, isto é, neste momento não grava quaisquer dados nos dados do freeRADIUS não ficando associado a qualquer utilizador ou a qualquer grupo. Por forma a um determinado recurso fique associado a um determinado utilizador, ou grupo, este terá de ser atribuído manualmente na opção “Assignments”.

4.3.1.11 –Editar/Remover Recursos

Nesta opção é possível editar os dados assim como os atributos associado a um determinado recurso. Mesmo que este esteja associado a utilizadores ou grupos, os dados alterados são actualizados tanto nos dados da aplicação como também nos dados lidos pelo freeRADIUS. Esta funcionalidade é apresentada na Figura 32.

Além de ser possível editar os dados dos recursos, também é possível apagá-los. No caso de estes estarem atribuídos a utilizadores ou grupos na altura em que se vai apagar todas as atribuições são apagadas, retirando aos utilizadores ou grupos o acesso ao recurso apagado.

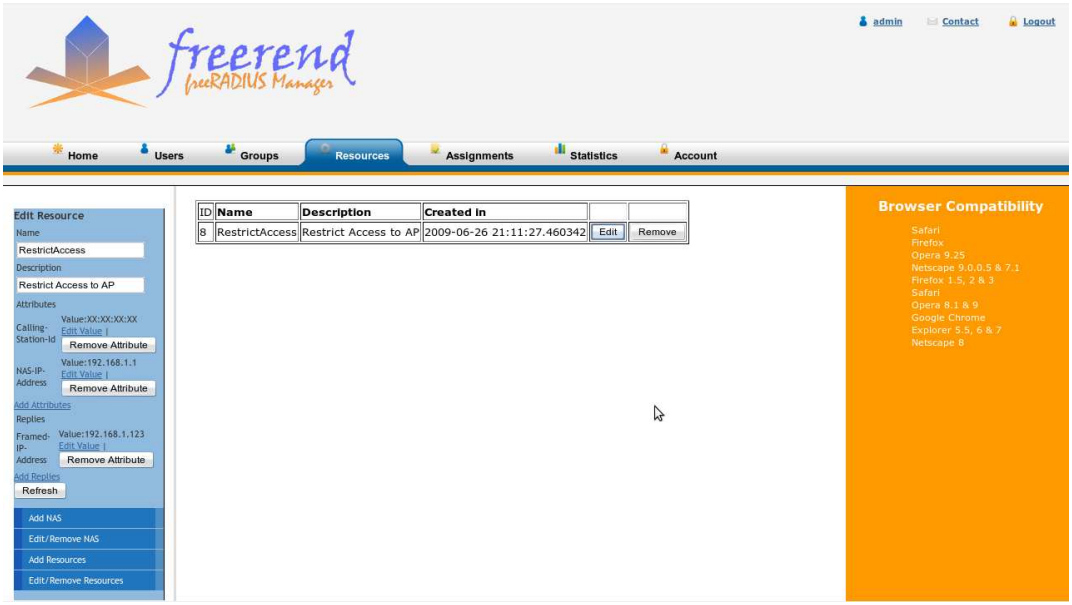

Figura 32 – Demonstração da edição de um recurso

4.3.1.12 – Atribuir Recursos a Utilizadores

Esta opção permite a atribuição de recursos criados anteriormente a utilizadores. Um gestor possui três opções: atribuir o recurso a si mesmo, atribuir o recurso a um gestor presente na sua lista de confiança ou atribuir o acesso a um utilizador do RADIUS. Esta funcionalidade é apresentada na Figura 33.

Figura 33 – Demonstração da atribuição de recursos a utilizadores

Ao efectuar a atribuição é realizada a relação entre os dados do RADIUS e os atributos, inserindo na tabela radcheck os atributos que deverão ser verificados aquando da autenticação por parte do utilizador e na tabela radreply os atributos que deverão ser enviados juntamente com a mensagem de *Access-Accept* por parte do servidor.

Para testar esta funcionalidade foi adicionado um novo recurso e, de seguida, foi atribuído o mesmo a um utilizador registado. O recurso terá de possuir, obrigatoriamente, o atributo "Auth-Type" definido, uma vez que todos os utilizadores que não possuam um "Auth-Type" definido são automaticamente rejeitados.

O recurso foi criado com o objectivo de restringir o acesso do utilizador apenas aquele AP. Neste ponto é que se torna importante o módulo "checkval" do freeRADIUS visto na parte de configuração. Por defeito, este módulo já possui definido o método de verificação do "Calling-Station-ID" que representa o endereço MAC do AP que enviou o pedido. Assim, o recurso foi criado com dois atributos: "Auth-Type" apresentado como valor "EAP" e "Calling-Station-ID" com o endereço MAC (*Medium Access Control*) do AP como respectivo valor. Em ambos os atributos foi usado o operador "==" uma vez que se tratam de atributos que têm de ser obedecidos aquando da autenticação. Testou-se a autenticação no terminal portátil verificou-se que o utilizador foi aceite. De modo a verificar que o utilizador está mesmo restringido ao AP que apresenta o endereço MAC registado no campo foi modificado o campo do "Calling-Station-ID" do recurso alterando apenas um dos algarismos. Com isto, foi tentada novamente a autenticação sendo que desta vez foi rejeitada. Isto aconteceu uma vez que o AP junto do qual o utilizador se pretende autenticar apresenta um endereço MAC diferente daquele que está presente no campo do "Calling-Station-ID".

4.3.1.13 –Atribuir Recursos a Grupos

Nesta funcionalidade o gestor pode atribuir um determinado recurso a um grupo pelo qual esteja responsável. Esta funcionalidade é apresentada na Figura 34.

Figura 34 - Demonstração da atribuição de um recurso a grupos

Ao efectuar a atribuição a um determinado grupo, os atributos que deverão ser identificados no momento da autenticação dos membros do grupo são inseridos na tabela radgroupcheck e os atributos que deverão ser enviados com a validação da autenticação são inseridos na tabela radgroupreply. Para testar esta funcionalidade foi criado um grupo por um dos utilizadores registados. Foi utilizado o mesmo recurso criado no ponto anterior mas, desta vez, foi atribuído ao grupo criado. Testando a autenticação verifica-se que a validação não foi bem sucedida. Isto aconteceu porque o valor do "Calling-Station-ID" foi modificado no ponto anterior. Voltando a modificar para o valor original, que equivale ao endereço MAC do AP e testando novamente a autenticação verificou-se que desta vez a autenticação foi bem sucedida.

4.3.1.14 –Remover Atribuições

Assim como é possível adicionar atribuições a utilizadores e grupos é também possível a remoção destas. Um gestor pode remover as suas atribuições, as atribuições dos gestores presentes na sua lista de confiança e ainda, se possuir as permissões suficientes, remover as atribuições a um determinado grupo.

4.3.1.15 –Estatísticas

Nesta opção, os gestores com permissões suficientes podem observar não só os dados referentes ao uso dos recursos do sistema assim como todos os dados referentes às alterações dos utilizadores, criação de grupos, acessos, entre outros. Esta funcionalidade é apresentada na Figura 35.

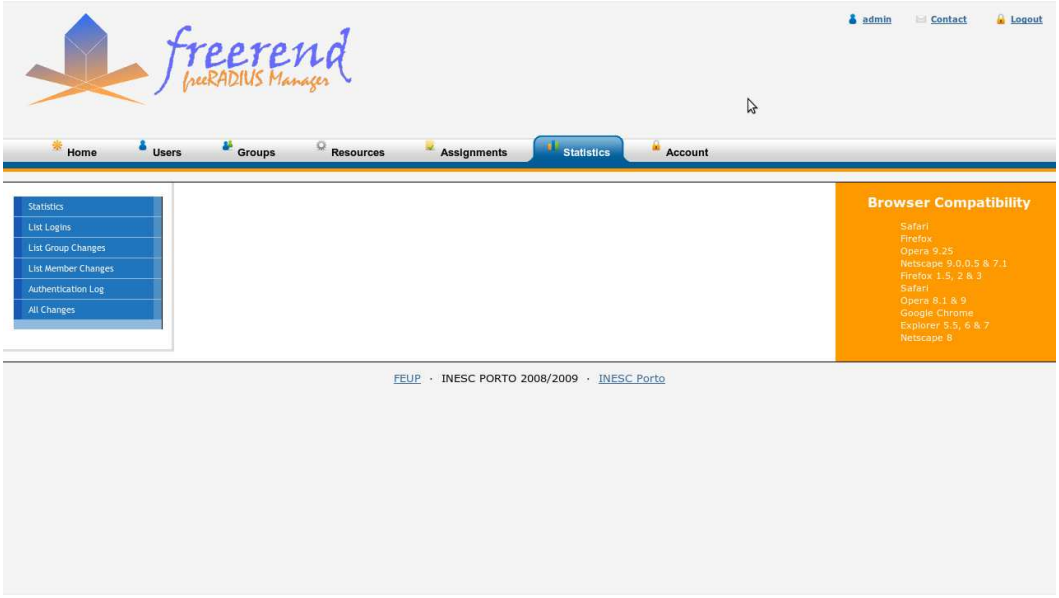

Figura 35 – Demonstração da página das estatísticas

4.3.1.16 –Editar Dados do Utilizador

Cada utilizador tem ainda a possibilidade de editar os seus próprios dados. Pode editar o seu nome de utilizador, nome e contactos. Todas as alterações são guardadas não só nos dados da aplicação assim como nos dados do freeRADIUS. Esta funcionalidade é apresentada na Figura 36.

Figura 36 – Demonstração da edição dos dados de utilizador

4.3.1.17 –Contactar utilizadores

Um gestor pode contactar todos os outros gestores utilizando a página de contacto e introduzindo o nome de utilizador e o texto que pretende enviar. Caso não saiba o nome de utilizador exacto, o gestor pode utilizar a função de pesquisa que se encontra nessa página e, de seguida, usar a opção de enviar mensagem, que actualiza o campo do destinatário. Esta funcionalidade é apresentada na Figura 37. Esta funcionalidade serve, por exemplo, para justificar o pedido para juntar à lista de confiança.

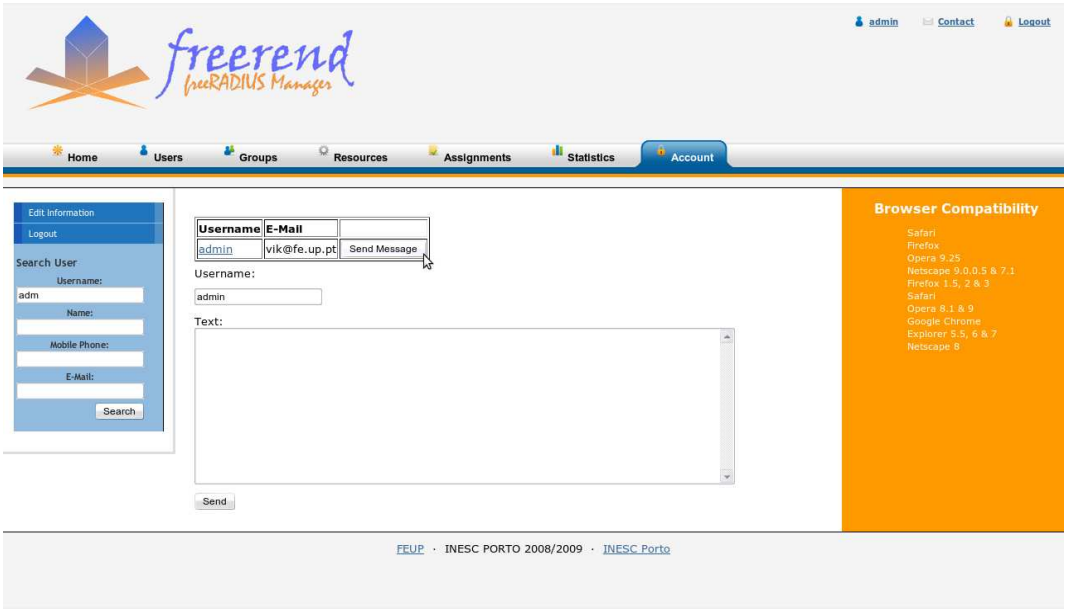

Figura 37 – Demonstração da página de contacto

4.3.2 –Compatibilidade

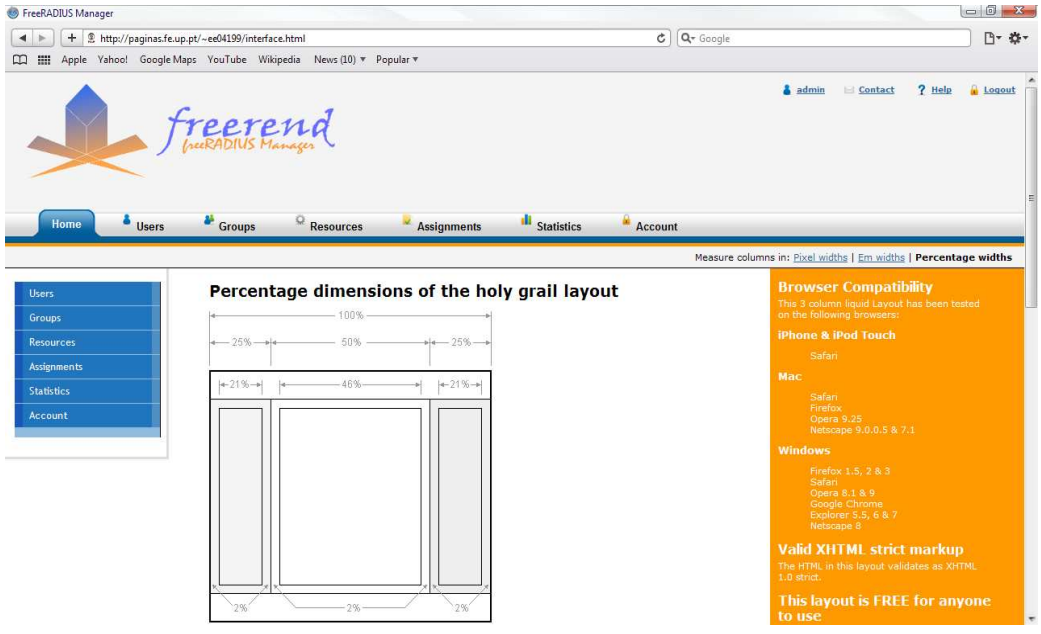

Figura 38 – Teste da aplicação com Safari

Nesta secção irá ser validade a compatibilidade da aplicação com os diferentes *browsers* existentes. A aplicação será testada nos seguintes *browsers*: Safari, Firefox, Opera, Netscape, Google Chrome e Windows Internet Explorer.

Na Figura 38 pode-se ver a aplicação aberta com o Safari. Daqui verifica-se que não apresenta qualquer problema em termos de visualização.

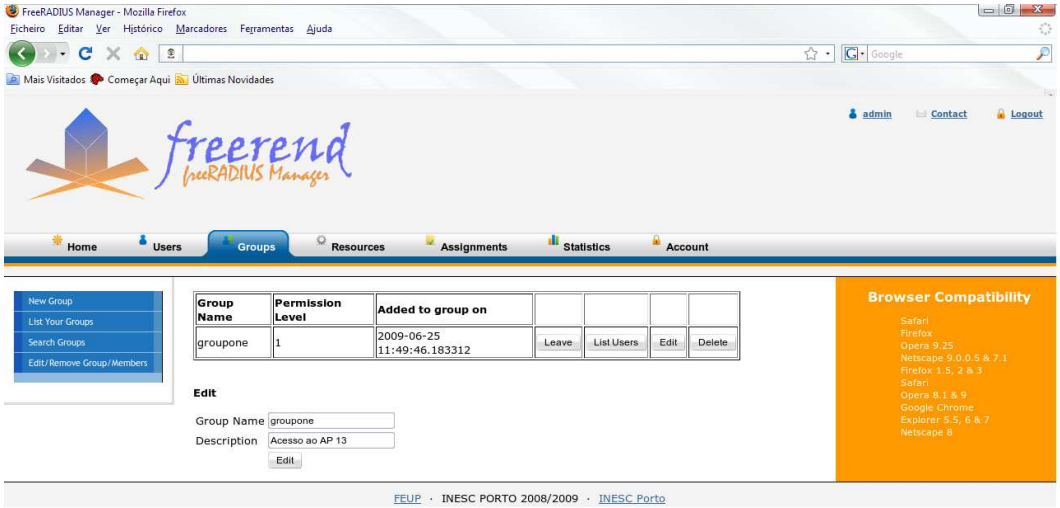

Figura 39 - Teste da aplicação com Firefox

Na Figura 39 é apresentado o resultado da visualização com o Firefox.

Através da visualização e do teste da aplicação utilizando o Firefox verifica-se também não existe qualquer incompatibilidade da aplicação com este *browser*.

De seguida é utilizado o Opera de forma a testar a compatibilidade com este *browser*.

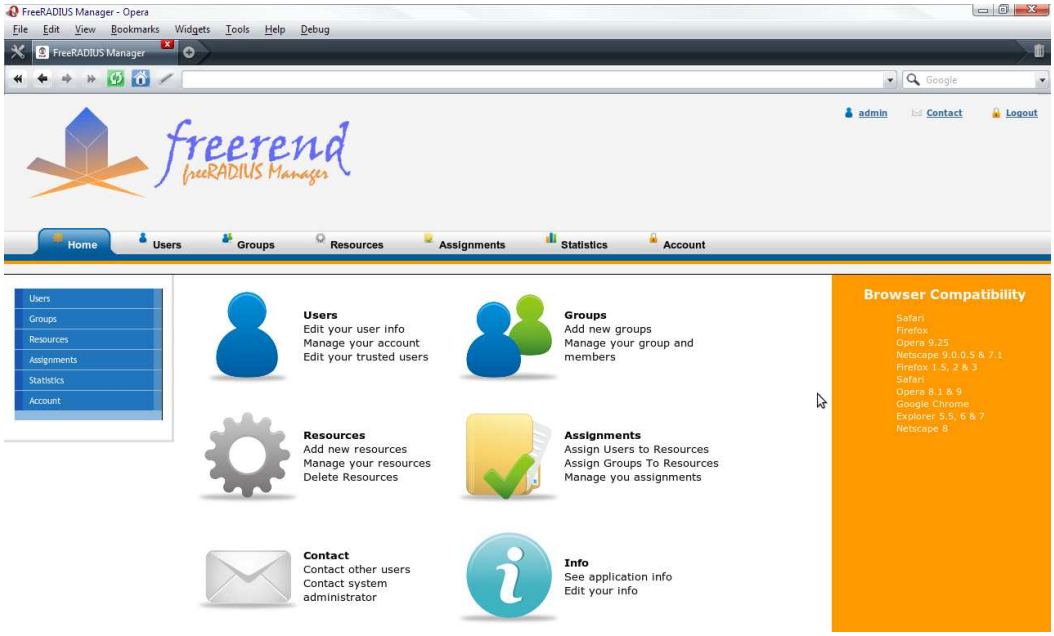

Figura 40 - Teste da aplicação com Opera

Através da utilização da aplicação usando o Opera verifica-se que também não existe qualquer tipo de incompatibilidade com este *browser*.

De seguida é verificada a compatibilidade da aplicação usando o Netscape.

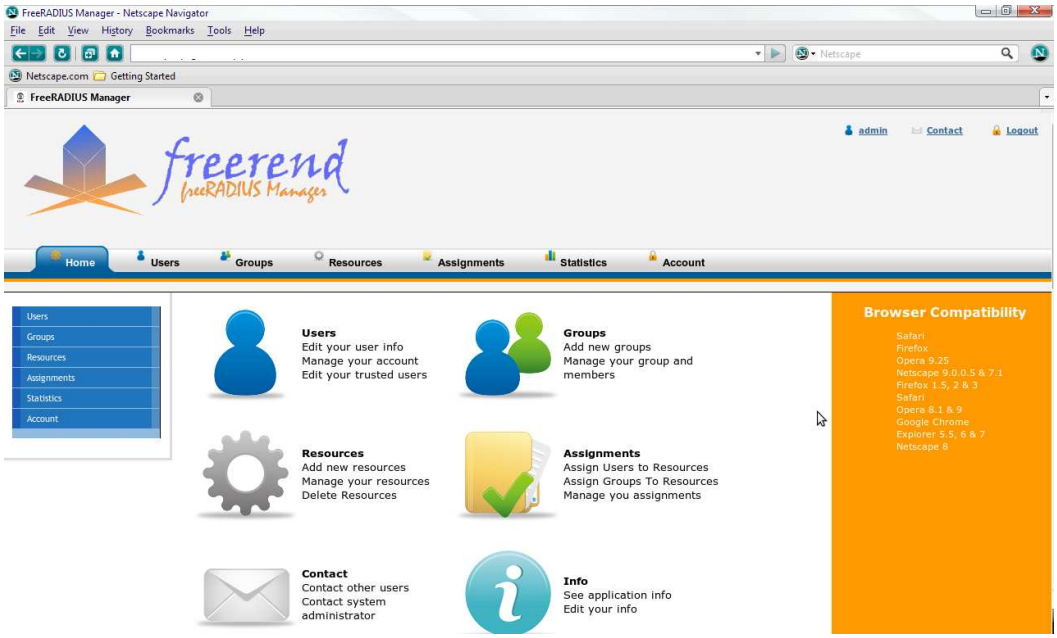

Figura 41 – Teste da aplicação usando o Netscape

Utilizando a aplicação usando o Netscape verificou-se que esta não apresenta nenhuma incompatibilidade com este *browser*. As funcionalidades resultam sem qualquer problema.

De seguida, a aplicação é testada usando o *browser* Google Chrome.

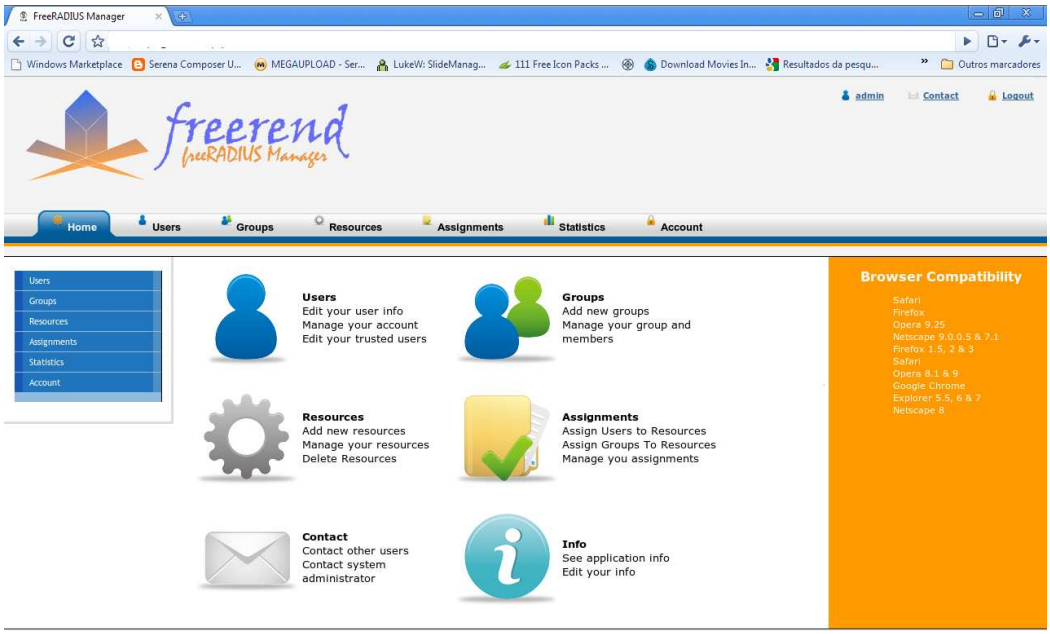

Figura 42 – Teste da aplicação usando o Google Chrome

Correndo a aplicação com o browser Google Chrome verifica-se que também não existe qualquer problema de incompatibilidade no decorrer das funcionalidades.

Por fim, irá ser testada a aplicação utilizando o *browser* da Windows, o Internet Explorer.

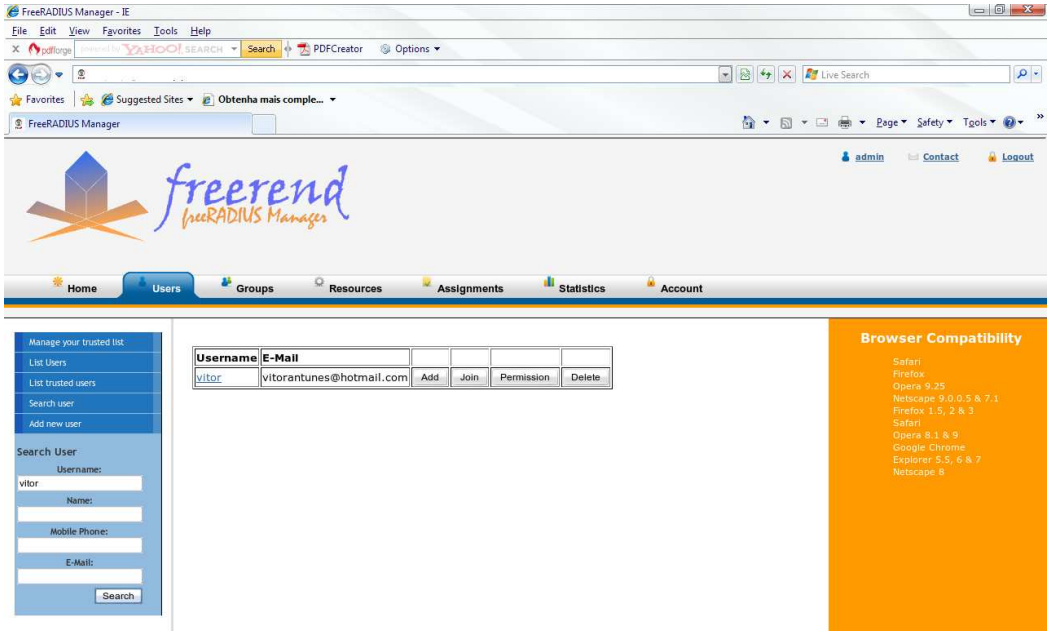

Figura 43 – Teste da aplicação usando o Internet Explorer

Verifica-se que, tal como nos restantes *browsers*, utilizando o Internet Explorer a aplicação não apresenta qualquer tipo de incompatibilidade.

Embora não tenham sido testados todos os *browsers Web* existentes, podemos concluir que a aplicação não apresenta problemas de compatibilidade, tal como era pretendido. A principal causa disto foi a utilização de um design totalmente feito com CSS, um sistema de três colunas com tamanhos definidos através de percentagens e uma biblioteca de AJAX funcional com todos os *browsers* testados.

4.3.3 –Adaptabilidade

Nesta secção irão ser demonstrados os resultados da adaptabilidade da aplicação em diferentes terminais. O primeiro dispositivo em que foi testada a aplicação foi num portátil com um ecrã de nove polegadas. Neste dispositivo a aplicação não apresentou qualquer problema de adaptabilidade. Posteriormente a aplicação foi testada num emulador de *iPhone*. A Figura 44 apresenta a visualização apresentada pelo emulador de iPhone.

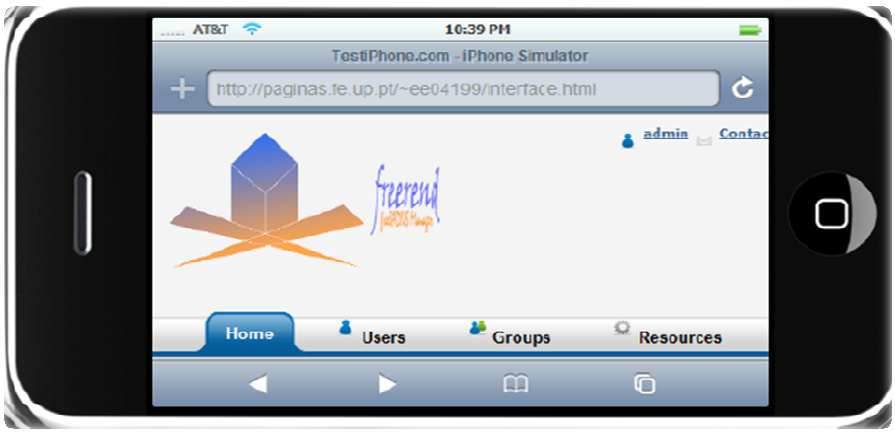

Figura 44 – Teste da aplicação usando um emulador *online* de *iPhone*

Embora não apresente toda a página pode-se verificar que esta é aberta sem problemas. A navegação pela página pode ser feita usando o *touch screen* do *iPhone*. Uma vez que o emulador em que foi testado era um pouco limitado, a aplicação foi também testada num *iPhone* real e foi possível observar que esta se adaptou automaticamente ao tamanho do ecrã. Daqui pode-se concluir que o simulador não apresenta as funcionalidades de um *iPhone* real uma vez que não adapta automaticamente o tamanho da página nem admite a adaptação do *zoom* da página. Pode-se também concluir que, em situações reais, a aplicação apresenta uma adaptação completa ao ecrã de um *iPhone*.

De seguida foi testada a aplicação num emulador de Windows Mobile. Neste caso a apresentação no emulador não foi a ideal uma vez que o *browser* presente no Windows Mobile era uma versão anterior não abrindo correctamente a página. No entanto, não pode ser verificado se o resultado obtido no emulador é o mesmo que nas plataformas reais. A falta de adaptação neste caso pode, eventualmente, ser explicada com a versão (bastante desactualizada) do Internet Explorer presente no emulador.

Por fim, a aplicação foi também testada num emulador de *Android*. A Figura 45 apresenta a aplicação aberta com o emulador de *Android*.

Figura 45 – Teste da aplicação usando um emulador de Android

Da imagem pode-se verificar que a aplicação se adapta completamente ao ecrã do *Android* não apresentando qualquer problema de compatibilidade neste caso.

Com estes resultados conclui-se que a aplicação RADIUS adapta-se a vários tipos de terminais com diferentes resoluções de ecrã. Isto deve-se em grande parte à construção da interface ter sido feita totalmente com CSS assim como o tamanho das colunas ter sido definido por percentagens, o que facilita a adaptação a diferentes resoluções e adaptação no caso de efectuar *zoom* nos casos do *iPhone* e *Android*.

4.4 - Limitações

Uma das grandes limitações encontrada no decorrer do trabalho foi a utilização do módulo "checkval" no freeRADIUS. Isto faz com que parte da configuração através da aplicação esteja limitada uma vez que, se quisermos, por exemplo, restringir um determinado utilizador pelo endereço IP do cliente que efectua o pedido, "NAS-IP-Address", tem de se adicionar esta opção ao módulo "checkval" no caso de não existir (por defeito apenas existe o atributo "Calling-Station-ID" no módulo "checkval"). Sem efectuar isto as autenticações que usem o atributo "NAS-IP-Address" não serão aceites.

Outra dificuldade encontrada foi a opção de rejeitar, por defeito, um utilizador. De modo a permitir esta opção foi necessário indicar ao freeRADIUS que todos os utilizadores que não apresentassem um "Auth-Type" registado seriam rejeitados.

4.5 - Discussão dos Resultados

Os resultados dos testes às funcionalidades demonstram que estas se encontram a realizar o pretendido e estas apresentam os requisitos mínimos identificados para a aplicação.

Em relação à compatibilidade verificou-se que nenhum dos *browsers* apresenta qualquer problema de compatibilidade. Isto significa que qualquer utilizador que queira aceder à aplicação não tenha qualquer obrigatoriedade da utilização de um determinado *browser*. No entanto está garantida a compatibilidade apenas nas versões dos browsers mais actuais uma vez que, embora não tenham sido testadas, existem versões anteriores que não suportam AJAX e que, por isto, a aplicação não seria, provavelmente, compatível nestes casos.

Analisando os resultados obtidos através dos testes da aplicação com diferentes terminais verifica-se que apenas apresentou problemas de adaptabilidade no caso do emulador de Windows Mobile. No entanto, não pode ser comprovado se este emulador representa os resultados obtidos numa plataforma Windows Mobile real.

Embora tenha havido problemas na apresentação com o emulador de Windows Mobile podemos afirmar tendo em conta os resultados das restantes plataformas, as quais são mais actuais que o Windows Mobile, que a aplicação apresenta adaptação aos mais actuais tipos de terminais portáteis.

4.6 - Conclusão

Neste capítulo foram apresentados os resultados dos testes efectuados utilizando a aplicação implementada. Inicialmente foi dada a descrição do cenário de testes e sua configuração. Este cenário consiste num servidor RADIUS ligado directamente, através de uma das interfaces de rede, a um AP configurado com o tipo de segurança WPA-EAP a qual representa o uso de um servidor RADIUS. No cenário existe ainda um terminal portátil que

efectua os pedidos de autenticação ao AP.

Dos resultados dos testes efectuados às funcionalidades da aplicação verificou-se que todas elas realizam o pretendido e que as funcionalidades apresentadas obedecem aos requisitos mínimos da aplicação. Além disto verificou-se que todas as funcionalidades minimizam a configuração necessária ao RADIUS. No entanto, existem módulos que têm de ser sempre configurados através de ficheiros de configuração. Um exemplo disto é o módulo "checkval". Este módulo define os valores verificados aquando do pedido e os pedidos de acesso a recursos que exijam a verificação de atributos que não estejam presentes no módulo "checkval" são rejeitados. Isto foi uma das limitações encontradas na utilização do freeRADIUS no desenvolvimento da aplicação.

Foram também realizados testes de adaptabilidade utilizando emuladores dos diversos tipos de terminais móveis. No caso do *iPhone* e do *Android* a aplicação apresentou-se sem qualquer problema de adaptabilidade. No caso do emulador de Windows Mobile a apresentação da aplicação não foi a correcta. No entanto não foi possível confirmar se a visualização no emulador seria a mesma que numa plataforma Windows Mobile real. A falta de adaptação neste caso pode, eventualmente, ser explicada com a versão (bastante desactualizada) do Internet Explorer presente no emulador.

Embora tenha havido incompatibilidade na apresentação utilizando o emulador do Windows Mobile pode-se afirmar, recorrendo aos resultados nas restantes plataformas, que a aplicação apresenta adaptabilidade aos mais actuais tipos de terminais móveis.

Capítulo 5

Conclusão e Trabalho Futuro

5.1 - Conclusão

A Internet apresenta, hoje em dia, elevada importância na sociedade. No entanto existem problemas de segurança relacionados com a autenticação de utilizadores. Um dos protocolos que resolve o problema não só da autenticação mas também da autorização e contabilização dos utilizadores e seus dados é o RADIUS (*Remote Access Dialin User Service*) que, embora tenha sido criado antes do conceito AAA (*Authentication Autorization Accounting*), apresenta um funcionamento bastante similar aos protocolos que se baseiam neste.

O conceito AAA descreve os processos de autenticação, autorização e contabilização. A autenticação verifica os dados enviados por parte de um determinado utilizador provando que está registado no sistema e se é, ou não, quem este diz ser. A autorização verifica quais os recursos a que o utilizador está autorizado a aceder. A contabilização destina-se à recolha de informação acerca da utilização de recursos de um sistema pelos seus utilizadores, por exemplo, contabilização do tráfego trocado, tempo da sessão, etc.

O protocolo RADIUS surgiu quando, em meados de 1992, a *Livingston* (hoje parte da Lucent Technologies) começou a desenvolver um protocolo que centralizasse os processos de autenticação, autorização e contabilização e que deveria viabilizar o acesso remoto a um serviço de rede. O RADIUS surgiu com os principais objectivos de ser um protocolo seguro, fiável, escalável e expansível. Neste documento verificou-se que o protocolo RADIUS apresenta um funcionamento com um modelo cliente/servidor tendo, tendo de um lado o cliente ou o NAS (*Network Access Server*) que, embora seja uma *gateway* que controla o acesso à rede, possui uma componente que funciona como cliente, e do outro o servidor. O utilizador, o NAS e o servidor trocam mensagens entre si quando o utilizador se pretende autenticar para utilizar um determinado recurso da rede.

Actualmente, o RADIUS é indispensável para os fornecedores de acesso à Internet e empresas que possuam redes privadas na medida em que permite um controlo de acessos centralizado baseado no conceito de autenticação, autorização e contabilização. No entanto, a configuração das aplicações que implementam o RADIUS não é trivial implicando que os gestores da aplicação teriam de ser altamente especializados em RADIUS e teriam a seu cargo

a gestão de milhares de utilizadores. Daqui surgiu a motivação de desenvolver uma ferramenta que automatize a gestão do RADIUS de modo a que uma pessoa que não tenha de perceber todos os detalhes do protocolo e possa mesmo assim gerir utilizadores do sistema, gerir as respectivas regras de acesso a cada recurso e ainda monitorizar os dados relativos à contabilização da utilização de cada recurso. Este trabalho tinha como principal objectivo o desenvolvimento de uma aplicação Web2.0 para gestão de RADIUS cujas principais características seriam a simplicidade de processos e automatização de tarefas que dizem respeito à configuração de utilizadores e regras de acesso.

Pretendia-se ainda que a implementação da aplicação conferisse ao utilizador facilidade de interacção e utilização apresentando uma interface atractiva e interactiva e que possa ser acedida a partir de qualquer local e através de diferentes tipos de dispositivos (tanto por *Personal Computers* e portáteis como por PDA, *Android, iPhone*, etc). Para isto, teve-se em consideração as diferentes capacidades de processamento e resolução de cada dispositivo de modo a que o tempo de resposta em dispositivos com menor capacidade de processamento não fosse muito elevado e de modo a que a aparência respeite o tamanho e resolução do ecrã do dispositivo.

Através de uma pesquisa e análise comparativa dos vários tipos de aplicação para gestão de RADIUS já existentes foi possível concluir que nenhuma destas obedece a todos os requisitos identificados para a aplicação pretendida evidenciando-se principalmente a falta de interactividade da aplicação com o utilizador e a inexistência de adaptabilidade desta aos diferentes tipos de terminais.

A revolução nas técnicas de desenvolvimento *Web*, designada por *Web* 2.0, permitiu uma evolução nos negócios tornando-os mais rentáveis. Realizando uma análise dos estudos realizados anteriormente acerca da performance das linguagens de programação *Web* verificou-se que a linguagem mais adequada à realização do projecto seria PHP uma vez que o trabalho tem como base uma ligação à uma base de dados de um servidor RADIUS e PHP é a melhor linguagem para bases de dados especificas e controlo sobre o acesso à base de dados, dado não ter O-R *Mapping* implementado de origem. No entanto, é importante referir que alguns estudos são antigos (da ordem de alguns anos) e, por isso, os resultados retirados dos estudos podem não se verificar actualmente.

O recurso ao modelo *Model-View-Controller* e a utilização de um *framework*, que é um conjunto de conceitos usados para resolver um problema específico, permite um desenvolvimento mais escalável e mais fácil de aplicações *Web*.

Após terem sido recolhidos as cinco *frameworks* PHP que mais se adequariam à implementação da aplicação verificou-se que o CodeIgniter se apresentou como a *framework* mais adequado ao desenvolvimento da aplicação devido à sua boa flexibilidade e facilidade, suporte das linguagens necessárias para o desenvolvimento do projecto e principalmente devido à sua boa performance.

Com a implementação da interface gráfica baseada constatou-se que uma interface que apresente a visualização realizada apenas recorrendo a CSS e com os tamanhos das colunas, tabelas e restantes elementos definidos através de percentagens, permite a adaptação a diferentes terminais e a compatibilidade a diferentes *browsers Web*. A interface desenvolvida apresenta um grafismo com aspecto limpo e permite a apresentação de tópicos de ajuda em cada página com a utilização de uma coluna do lado direito.

Após a implementação foram realizados testes de forma a verificar se a aplicação obedecia a todos os requisitos definidos e se satisfazia todos os objectivos delineados. Usando

um cenário de teste que consistia num servidor RADIUS ligado directamente, através de uma das interfaces de rede, a um AP configurado com o tipo de segurança WPA-EAP, a qual representa o uso de um servidor RADIUS, e num terminal portátil que efectua os pedidos de autenticação ao AP foram realizados testes a todas as funcionalidades assim como testes à compatibilidade e adaptabilidade da aplicação.

Dos resultados dos testes efectuados às funcionalidades da aplicação constatou-se que todas elas realizam o pretendido e que as funcionalidades apresentadas obedecem aos requisitos mínimos da aplicação. Além disto observou-se que todas as funcionalidades minimizam a configuração necessária ao RADIUS. No entanto, existem limitações inerentes à configuração de regras de acesso do freeRADIUS usando a aplicação. Um exemplo disto é o módulo "checkval" do freeRADIUS. Este módulo define os valores verificados aquando do pedido e os pedidos de acesso a recursos que exijam a verificação de atributos que não estejam presentes no módulo "checkval" são rejeitados. Isto foi uma das limitações encontradas na utilização do freeRADIUS no desenvolvimento da aplicação.

Foram também realizados testes de adaptabilidade utilizando emuladores dos diversos tipos de terminais móveis. No caso do *iPhone* e do *Android* a aplicação apresentou-se sem qualquer problema de adaptabilidade. No caso do emulador de Windows Mobile a apresentação da aplicação não foi a correcta. No entanto não se pode confirmar se a visualização no emulador seria a mesma que numa plataforma Windows Mobile real. A falta de adaptação neste caso pode, eventualmente, ser explicada com a versão (bastante desactualizada) do Internet Explorer presente no emulador.

Embora tenha havido incompatibilidade na apresentação utilizando o emulador do Windows Mobile pode-se afirmar, recorrendo aos resultados nas restantes plataformas, que a aplicação apresenta adaptabilidade aos mais actuais tipos de terminais móveis.

Daqui conclui-se que a aplicação desenvolvida obedece a todos os requisitos definidos e, tal como era pretendido, não apresenta qualquer problema de compatibilidade com diferentes *browsers* nem qualquer problema de adaptabilidade com diferentes terminais.

5.2 - Trabalho Futuro

Como trabalho futuro, e uma vez que neste projecto não era um dos objectivos, deverá ser implementada os mecanismos de segurança de forma a garantir uma gestão segura do servidor RADIUS evitando possiveis ataques.

Poderá ser desenvolvido um sistema de gráficos em tempo real por forma a que um gestor possa ver, em cada instante, as sessões activas assim como o tráfego trocado no decorrer dessas sessões. Neste sistema de gráficos pode ser também desenvolvidas opções para verificar gráficos de todos os restantes dados interessantes para monitorização do sistema.

Podem ser também adicionadas à aplicação funções que diminuam ainda mais a configuração através de ficheiros de configuração.

A aplicação foi desenvolvida de forma a possuir uma única interface para os diversos dispositivos. No entanto, pode ser implementada uma interface mais rica para os terminais que possuam maior capacidade de processamento e outra com menores requisitos de capacidade de processamento para quando fosse acedida através de terminais móveis e outros de baixa capacidade de processamento. As técnicas de desenvolvimento de interfaces estão constantemente em evolução e, por isso, o *design* da aplicação pode sempre ser actualizado de forma a tornar mais intuitivo para o utilizador.

Referências

[1] Bruno Oliveira. “Arquitecturas e serviços IP para redes comunitárias de utilizadores”, 2008.

[2] Lucent Technologies, C. de Laat, e outros. s, “Rfc2903 - generic AAA architecture Regras de escrita e gramática”, Agosto de 2000.

[3] Livingston, C.Rigney, S. Willens, e outros. “Rfc 2865 - remote authentication dial in user service (RADIUS)”, Junho, 2000.

[4] Akara Sucharitakul (Sun Microsystems Inc.) Shanti Subramanyam (Sun Microsystems Inc.). “Cadillac or nascar? A non-religious investigation of modern web technologies”, 2008.

[5] Anupam Chanda, Emmanuel Cecchet e outros. Performance comparison of Middleware architectures for generating dynamic web content, 2003.

[6] Martin Arlitt, Lance Titchkosky e Carey Williamson. “A performance comparison of dynamic web technologies”, 2003.

[7] Jesse James Garrett. “Ajax: A new approach to web applications”, 2005. Disponível em http://www.adaptivepath.com/ideas/essays/archives/000385.php.

[8] Christopher L Merrill. “Performance impacts of AJAX development”, 2006.

[9] “Web 2.0 A Stratagy Guide”, Abril 2008.

[10] O'Reilly & Associates. “RADIUS”, Outubro, 2002.

[11] CakePHP: the rapid development php framework, disponível em: http://cakephp.org/

[12] CodeIgniter: Open source PHP web application framework, disponível em http://codeigniter.com/

[13] Prado: PHP framework, disponível em http://www.xisc.com/

[14] Symfony: Web PHP framework, disponível em http://www.symfony-project.org/

[15] Zend Framework, disponível em http://zendframework.com/

[16] FreeRADIUS: The world’s most popular RADIUS Server. Disponível em http://freeradius.org/

[17] Cisco Secure Access Control Server for Windows, disponível em: http://www.cisco.com/en/US/products/sw/secursw/ps2086/

[18] Microsoft Internet Authentication Service. Disponível em: http://technet.microsoft.com/en-us/network/bb643123.aspx

[19] dialupadmin: Dialup Admin Administration interface. Disponível em: http://freeradius.org/dialupadmin.html

[20] daloRADIUS: RADIUS Management, Reporting, Accounting, Graphs, and much more. Disponível em: http://daloradius.com/

[21] ezRADIUS. Disponível em: http://ezradius.sourceforge.net/

[22] phpMyPrepaid. Disponível em: http://sourceforge.net/projects/phpmyprepaid/

[23] phpRADmin: RADIUS web management. Disponível em: http://sourceforge.net/projects/phpmyprepaid/

[24] The Perfect 3 Column Liquid Layout: No CSS hacks. SEO friendly. iPhone compatible. Disponível em http://matthewjamestaylor.com/blog/perfect-3-column.htm.