Faculdade de Engenharia da Universidade do Porto

API de Serviços Web e Extensão para Software de Webanalytics/Adserving

Luís Miguel Silva

VERSÃO FINAL

Relatório de Projecto realizado no âmbito do
Mestrado Integrado em Engenharia Electrotécnica e de Computadores
Major Telecomunicações

Orientador: Prof. Dr. João Correia Lopes
Responsável da Empresa: Eng. Rui Ribeiro

Julho 2009

U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

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

A Dissertação intitulada

"API DE SERVIÇOS WEB E EXTENSÃO PARA SOFTWARE DE WEBANALYTICS/ADSERVING"

foi aprovada em provas realizadas em 16/Julho/2009

o júri

Presidente Professor Doutor Francisco José de Oliveira Restivo
Professor Associado do Departamento de Engenharia Informática da Faculdade de Engenharia da Universidade do Porto

Professor Doutor Orlando Manuel Oliveira Belo
Professor Associado do Departamento de Informática da Escola de Engenharia da Universidade do Minho

Professor Doutor João António Correia Lopes
Professor Auxiliar do Departamento de Engenharia Informática 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.

Autor LUÍS MIGUEL NOGUEIRA GOMES DA SILVA

Faculdade de Engenharia da Universidade do Porto

Resumo

Neste documento são abordados dois tópicos principais: Webservices e aplicações de Webanalytics e Adserving.

O desenvolvimento de raiz de uma infra-estrutura de Webservices obrigou ao estudo de Frameworks que pudessem ser usadas nessa implementação, à pesquisa dos protocolos mais utilizados e vantajosos para a sua criação e a uma grande pesquisa do estado da arte em sistemas similares. Foram, assim, procurados comparativos de desempenho, escalabilidade, segurança, popularidade entre outros parâmetros, para que o desenvolvimento seguisse a linha dum sistema actual e de alto desempenho. Uma vez definida a estrutura do sistema foi iniciada a sua implementação, tendo sido realizados testes preliminares ao seu bom funcionamento e tendo-se verificado se os resultados correspondiam aos definidos inicialmente. Para aumentar a velocidade de resposta dos webservices, foram estudados e implementados mecanismos de pré-compilação do código, que diminuíram em 3 vezes o tempo de acesso a um serviço. Foram também pesquisados e testados sistemas de fila de espera, para verificar a possibilidade de controlar o fluxo de acessos e de criar serviços assíncronos.

Numa segunda vertente do projecto foram investigadas e comparadas as várias aplicações de Analytics e Adserving existentes no mercado, sobre a qual se pudesse criar um Plug-in que consumisse os serviços disponibilizados pela empresa e que se tornasse num *proof of concept*. Pretendia-se que essa aplicação fosse largamente utilizada na Internet e que tivesse uma boa base de desenvolvimento na qual se pudesse integrar o Plug-In.

Seguem-se, assim, no restante deste documento, a análise em maior profundidade destes temas e as conclusões mais importantes para futuras implementações de sistemas semelhantes.

Página em branco

Abstract

This document addresses two main topics: Webservices and Webanalytics and Adserving applications.

The development of a Webservices infrastructure implied the research of Frameworks that could be used in its implementation as well as the study of the most used protocols in this field and in similar systems. With the intent of creating a modern and high performance system many benchmarks and comparisons were taken into account, like performance, security, scalability and popularity. Once the state of the arte was finished the initial phase of development was started, some code was written and some preliminary tests were performed in order to analyze if it was performing as expected. As the response of the Webservices was not fast enough they were integrated with a pre-compiler for PHP which increased in about 3 times the throughput. In order to test the capability of flow control and asynchronous services some queues were also tested and verified for compatibility.

The second topic consisted in researching and comparing several Analytics and Adserving platforms that had the compatibility with Plug-Ins. That way, a proof of concept could be created that would access the developed Webservices and show reports of information to the user. This software needed to be widely used in the Internet and needed to have a solid base for development.

Throughout the rest of this document these subjects are explained in more detail and some conclusions are taken to help future implementations of similar systems.

Página em branco

Agradecimentos

Ao homem que sabiamente proferiu esta tão verdadeira frase:

"You will forever walk the earth with your eyes turned skyward, for there you have been, and there you will always long to return."

Leonardo Da Vinci

Página em branco

Índice

Lista de Figuras

Lista de Tabelas

Abreviaturas e Símbolos

Lista de abreviaturas (ordenadas por ordem alfabética)

API	*Aplication Programming Interface*
CBR	*Content Based Routing*
PPA	*Pay Per Action*
PPC	*Pay Per Click*
PPM	*Pay Per Impression*
SOA	*Service Oriented Architecture*
WWW	*World Wide Web*

Glossário

Advertiser – Ver Anunciante

API – Conjunto de rotinas estabelecidas para utilização numa determinada aplicação

Anunciante – Pessoa ou entidade que pretende publicar um anúncio ou campanha

Backbone – Ligações centrais de um sistema de computadores com elevado desempenho

Bookmark – Marcar uma página nos favoritos do browser

Browser – Aplicação para navegar na Web (ex. Internet Explorer)

Cache – Memória física ou software com o intuito de guardar dados temporariamente.

CBR – Roteamento das mensagens consoante o seu conteúdo

Click Fraud – Simular cliques do rato num anúncio

ESB - Aplicação que agrega várias tecnologias duma empresa num só barramento para simplificar a sua utilização e manutenção

Expositor – Pessoa ou entidade que exibe numa página Web pelo menos um anúncio

Framework – Conjunto de bibliotecas que disponibilizam funcionalidades para programação

Hosted – Alojável num servidor próprio do utilizador

Namespace – Abstracção que fornece uma desambiguação a itens com o mesmo nome

Link - Hiperligação

Load Balancing – Gerir a carga atribuída a vários computadores, distribuindo-a

Open Source – Aplicações de uso livre

Pipeline – Optimização que se aproxima da paralelização da execução de um serviço

Plugin – Extensão para aplicação informática

PPA/PPC/PPM – Modelos de negócio da publicidade Online

Publisher – Ver Expositor

Script – Um programa que corre num servidor (ou sistema operativo)

SOA – Arquitectura de disponibilização de serviços na Internet

Stored Procedure – Subrotina utilizada em bases de dados relacionais

Throttling – Controlar a largura de banda atribuída a um processo

WWW – Rede de páginas e conteúdos na Internet

Capítulo 1

Introdução

Desde o advento da Internet [1] que a sua utilização não pára de crescer em todo o mundo. Como tal, muitas das infra-estruturas que todos usamos no dia-a-dia, começaram a ser portadas para a rede global, como por exemplo os correios, que têm o correio electrónico como equivalente virtual e as bibliotecas e fóruns, que deram origem a grupos de discussão, Web-fóruns e sistemas com milhares de páginas de informação centralizada. Seguindo a tendência, também a publicidade que é corrente ver-se nas televisões, rádios e painéis por todo o mundo se expandiu para a *World Wide Web*, tendo gerado mais de 23 biliões de dólares americanos só no ano de 2008 [2].

Estes valores extremamente elevados de facturação atraem a atenção de empresas e indivíduos mal intencionados, que tentam a todo o custo criar mecanismos e esquemas fraudulentos com o intuito de receberem algum desse dinheiro. Com essa situação em mente, começam a emergir algumas empresas que disponibilizam os seus serviços para tentar minimizar este fenómeno crescente de fraude. É neste panorama que surge a Auditmark, a empresa proponente do tema desta dissertação. Os seus objectivos são claros: fornecer os seus serviços para auditar a publicidade dos seus clientes, disponibilizando relatórios pormenorizados sobre a actividade fraudulenta nas várias campanhas que este detenha. Para os atingir, é necessário desenvolver e manter vários módulos de recolha, análise e exibição de dados.

No restante deste documento serão detalhados os objectivos específicos a esta dissertação – implementar *Webservices*, criar uma extensão para uma aplicação de gestão de anúncios, entre outros – explicadas as pesquisas das mais recentes técnicas e *software* – aceleradores de PHP, gestores de anúncios, frameworks de desenvolvimento de *Webservices* – e explicitada toda a implementação e testes efectuados.

1.1 Enquadramento

Este projecto insere-se num ambiente de vários temas e projectos que decorrem em paralelo na empresa Auditmark, visando combater a fraude na publicidade na Internet. As suas actividades envolvem investigação, desenvolvimento e implementação de novas técnicas de identificação de utilizadores, *browsers*, detecção de fraude na publicidade, entre outros. Como tal, também é necessário desenvolver plataformas que permitam aos seus clientes ter acesso aos resultados do processamento realizado. Com isto em emente, surgiu o tema para esta dissertação, criar serviços de acesso ao *Backbone* e exibir resultados em aplicações de gestão já utilizadas pelo cliente e disponíveis na WEB, sobre as quais se podem construir extensões.

1.2 Motivação

O envolvimento numa empresa ambiciosa da área das novas tecnologias e Internet, com projectos inovadores de investigação e desenvolvimento, é sempre muito aliciante para qualquer engenheiro. Este trabalho revela-se uma óptima oportunidade de aprofundar conhecimentos técnicos e de entender o fluxo de verbas envolvidos na *World Wide Web*. Permite ainda desenvolver aplicações que fazem a diferença e que têm um forte impacto na vida económica das empresas que lá transaccionam. Além disso, o resultado das pesquisas e efectuados e conclusões tiradas será certamente útil a outros que pretendem prosseguir na mesma linha de trabalhos, pelo que é um óptimo motivo para realizar um trabalho sério e com boas bases de conhecimento.

1.3 Objectivos

O projecto, "API de Serviços web e Extensão para Software de *Webanalytics/Adserving*", consiste no desenvolvimento de software em duas vertentes distintas: a 1ª vertente centra-se no desenvolvimento de uma API de *Webservices*, implementando de raíz uma *SOA – Service Oriented Architecture*, implementando alguns serviços e fornecendo uma base robusta e facilmente escalável. Esta plataforma visa disponibilizar uma forma de acesso rápida e segura a vários serviços desenvolvidos pela empresa. Cria-se, assim, um canal de comunicação que obedece a um standard e fornece uma interface comum que pode ser utilizada pelas mais variadas aplicações. A 2ª vertente assenta na criação de uma extensão para uma aplicação de WebAnaltics/AdServer, cuja escolha dependerá de uma análise criteriosa das várias soluções existentes no mercado. O objectivo deste *plugin* é o de servir como ferramenta de demonstração das funcionalidades desenvolvidas pela empresa. Será usada a API de *webservices* para comunicar com os servidores da Auditmark e recolher informação já processada, que será então exibida ao utilizador, p.ex. relatórios de análise de *Click Fraud*.

Assim, no futuro e tendo como base estas duas implementações, será possível a reutilização dos serviços criados para a integração com outras aplicações, bem como expandir as duas plataformas sob uma interface largamente utilizada na Internet.

1.4 Estrutura

Este primeiro capítulo introduz o trabalho desenvolvido neste projecto intitulado "API de Serviços Web e Extensão para Software de *Webanalytics*/*Adserving*", bem como o contexto em que este se insere e quais objectivos que se pretendem com ele atingir. De seguida, são apresentados alguns conceitos úteis para enquadrar este projecto e é dada uma solução ao problema inicial proposto. Espera-se que no final da sua leitura, seja possível entender os motivos que levaram ao estudo efectuado no Capítulo 3 e à implementação documentada no Capítulo 4. No Capítulo 5 estão incluídas as verificações ao funcionamento dos sistemas e testes de desempenho, para se comprovar que se cumpriram os objectivos aqui descritos. Para terminar este documento, no Capítulo 6 tiram-se as conclusões sobre todo o trabalho, apresentam-se propostas para uma futura continuação do projecto, encerrando-se esta dissertação.

Página em branco

Capítulo 2

Publicidade Online e Solução a Implementar

Este capítulo está dividido em dois temas: A Publicidade Online, onde se explica e analisa o seu funcionamento e se clarificam alguns conceitos e a solução idealizada para atingir os objectivos que são propostos para este projecto.

2.1 Publicidade Online

Neste subcapítulo é explicado o enquadramento geral da publicidade online, para que se consiga entender melhor as questões que originaram o desenvolvimento da empresa em que se insere este projecto e, indirectamente, este projecto.

2.1.1 Conceitos

Alguns conceitos como *Publisher*, *Advertiser,* Agência de publicidade e *Click Fraud* necessitam de ser clarificados para um bom entendimento do funcionamento de meio de publicação de anúncios. A Figura 1 e as respectivas definições apresentadas a seguir são essenciais para uma boa compreensão do modelo de negócio:

- ***Publisher* ou Anunciante**: Empresa, sítio, pessoa ou outra entidade que disponibiliza o conteúdo publicitário aos utilizadores da Internet, servindo de suporte para a exibição dos anúncios. Deve também fornecer, para além de espaço e tempo de publicação, algumas estatísticas e análises de tráfego e visualização, de forma a poder assegurar a qualidade de serviço contratada.

- ***Advertiser* ou Expositor***:* Empresa, sítio, pessoa ou outra entidade que pretende ver o seu anúncio colocado na Internet para visualização. Esta parte interessada paga o serviço de colocação e espera em retorno garantias da qualidade de serviço acordadas.

- **Agência de Publicidade***:* Entidade intermediária que pode ou não existir, responsável pela gestão de vários clientes - *Advertisers*, recebendo destes o pagamento dos anúncios e respectivas comissões e colocando-os em *Publishers* para exibição. São responsáveis por assegurar a qualidade de serviço contratada e pela resolução de eventuais problemas que surjam no negócio.

- ***Click Fraud:*** Consiste na utilização de *scripts* ou redes de utilizadores contratados, com o objectivo de simular cliques de utilizadores distintos num determinado anúncio. Esta actividade é ilegal, pois em certos modos de pagamento (ver página 6 - **PPC**) isso implica que o *Advertiser* tenha que pagar por cliques que estão a ser simulados, não havendo qualquer utilizador interessado na publicidade.

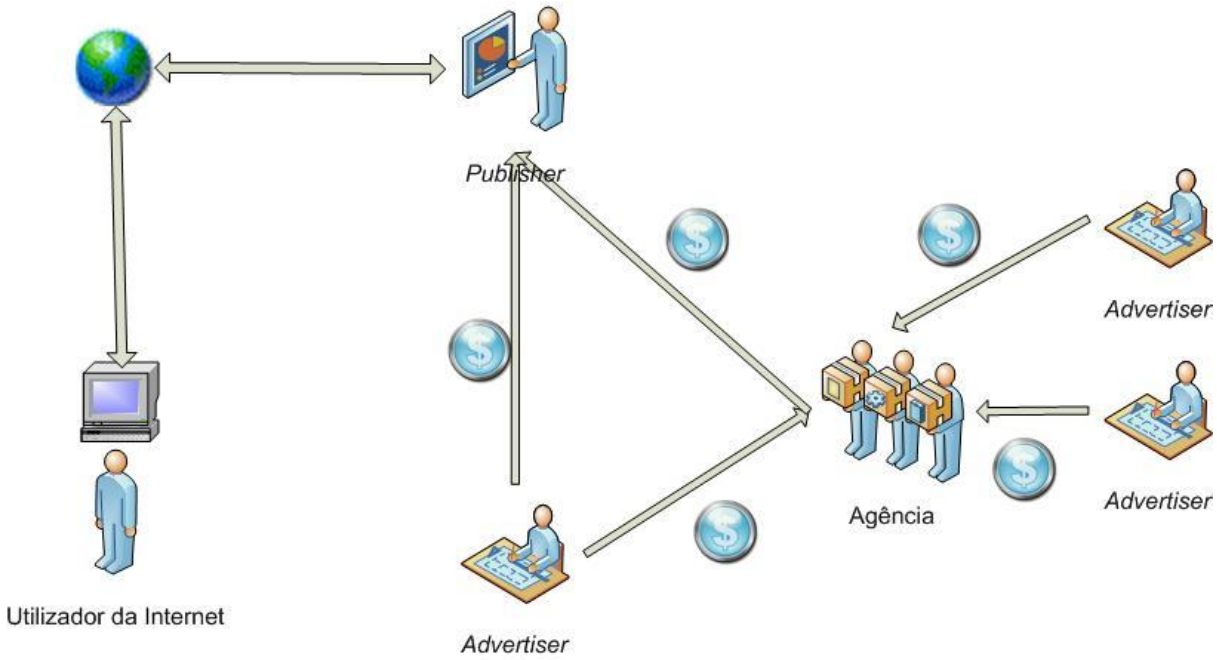

FIGURA 1 - MODELO DA PUBLICIDADE ONLINE

2.1.2 Modelos de Negócio

Uma vez identificadas as partes intervenientes no processo, avança-se para uma explicação da forma como estas interagem e de como funciona o mercado da publicidade online.

No inicio do processo considera-se uma pessoa/entidade que pretende ver um anúncio seu, ou até mesmo uma campanha inteira de publicidade, exibido na Internet - o anunciante ou Advertiser. Para isso, necessita de contactar com os donos de várias páginas Web para que estes publiquem os seus anúncios – os expositores ou *publishers* - de modo a ser visualizado pelo máximo número possível de pessoas. Pode ainda haver um intermediário neste processo, uma agência de publicidade, responsável por gerir uma variedade de locais na Web onde os anúncios serão colocados, recolhendo dados especí-

ficos para enviar ao seu cliente, o anunciante, para efeitos de taxação do serviço e disponibilização de informação sobre o sucesso das suas campanhas, conforme exibido na Figura 1.

No entanto, muitos anunciantes não têm sequer na sua estrutura empresarial, departamentos destinados à publicidade comum/publicidade online, e ao tratamento da informação que lhes é enviada pelos diversos *Publishers*. Assim, preferem contratar os serviços de agências de publicidade, às quais entregam uma carteira que pretendem ver gerida e publicada. Sendo esta a actividade que exercem, dispõem de um conjunto de ferramentas, pessoal e contratos com *Publishers* que rentabilizam muito mais os conteúdos dos seus anunciantes. Serão as agências, as responsáveis por entregar os anúncios e controlar a qualidade de serviço, reportar regularmente ao cliente o ponto da situação e também mediar a resolução de eventuais conflitos. Existem ainda algumas, como o caso do Google que, para além de tudo isto, possuem ainda a sua própria rede de publicação, combinando o papel de agência e expositor.

Depois de celebrados os contratos, os anúncios ficam publicados na Web, onde os utilizadores os podem ver e com os quais podem interagir, dando-se início ao processo de facturação, que segue o esquematizado na Figura 1. As formas de pagamento podem ser baseadas em várias modalidades, sendo de seguida apresentadas as mais usuais:

PPM – *Pay Per Impression*: o anunciante paga pela exibição de um determinado anúncio num determinado sítio(s). O montante depende apenas do número de visualizações da página onde este está inserido, independentemente do número de *cliques* e do rendimento que este lhe possa trazer.

PPC – *Pay Per Click*: ao contrário do PPM, neste caso não é a colocação da publicidade que é paga, mas sim os cliques que os utilizadores fazem em cada anúncio. No entanto, não é necessário que desse clique surja qualquer outro tipo de acção para haver pagamento, o facto do utilizador se mostrar interessado e clicar, basta.

Este modelo é dos mais utilizados nesta área já que todas as entidades, exceptuando o anunciante, ficam a ganhar. É este facto que potencia o crescimento da *click fraud*, uma vez que deixa de ser do interesse das grandes companhias desenvolver aplicações e metodologias para o combater, que têm avultados lucros com as actividades ilegais de terceiros.

PPA – *Pay Per Action*: neste tipo de contrato, apenas é pago o anúncio se daí resultar alguma transacção para a parte interessada, o anunciante. O simples facto de alguém poder visualizar o anúncio ou até clicar e seguir a ligação não é suficiente. O pagamento efectua-se por cada transacção efectuada com sucesso, que pode ser identificada através de um registo, venda, etc.

2.2 Solução a Implementar

Tendo em vista cumprir os objectivos apresentados anteriormente, desenhou-se uma estrutura que servisse de guia ao desenvolvimento. A Figura 2 mostra esse esboço.

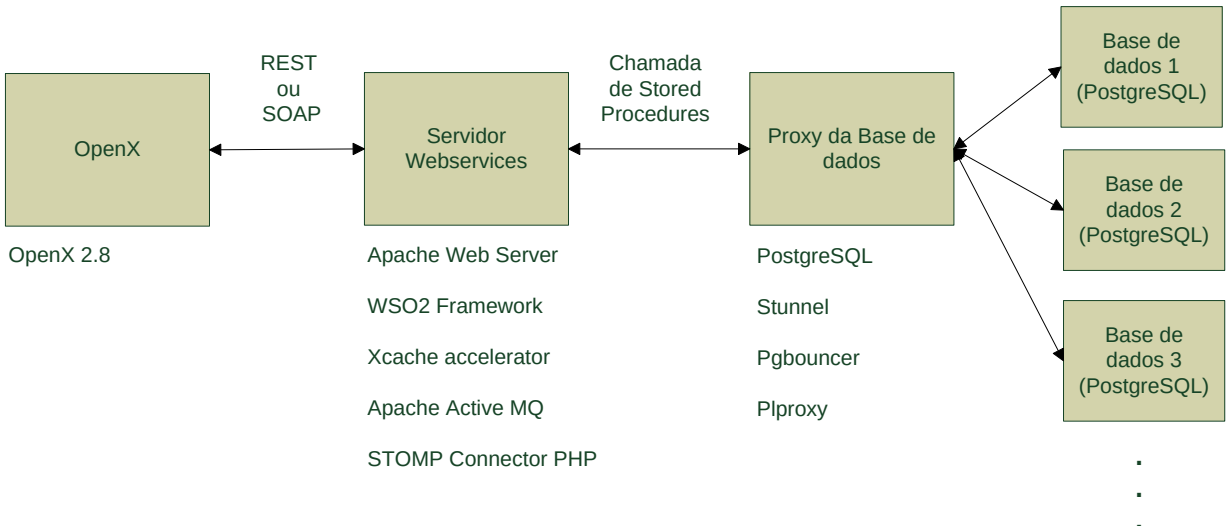

FIGURA 2 – ESTRUTURA A IMPLEMENTAR

São necessárias três componentes para implementar o sistema: a extensão para a aplicação de *Analytics*/*Adserver*, um sistema de disponibilização de serviços Web e uma base de dados.

Uma vez que paralelamente a este projecto, decorreu um outro que visou desenvolver um website público e um portal de acesso exclusivo a clientes da empresa que permite fazer a gestão das contas e relatórios de um cliente [15], houve algum reaproveitamento de infra-estruturas. Foi utilizada a plataforma de bases de dados já existentes que assenta numa configuração distribuída de PostgreSQL, com uma das instalações a servir como um *Proxy* de acesso. Em todas elas era utilizada uma estratégia de *stored procedures*, programadas em Pl/SQL ou plproxy, para aumentar a segurança.

Essas funções são então usadas pelo servidor de *Webservices*, cujo objectivo é criar um filtro entre o exterior e o interior da empresa, e que fornecem uma abstracção aos utilizadores que apenas têm de se preocupar em consumir um serviço. Desta forma, a estrutura interna do *backbone* não é divulgada e pode ser completamente alterada sem que os clientes necessitem de modificar o seu código, sendo apenas necessário garantir que os serviços se mantenham.

Na outra extremidade existe a aplicação de *Analytics*/*Adserving* que servirá de exemplo às potencialidades do sistema. A extensão a incluir no programa irá consumir alguns serviços de recolha de relatórios e exibi-los ao utilizador, demonstrando a sua utilização e eficácia para o transporte de dados.

Capítulo 3

Webservices e Aplicações de *Analytics*

As aplicações de gestão de campanhas publicitárias – *Analytics*/*Adserving*, são um dos pontos sobre o qual se irá desenvolver este projecto, nomeadamente com a construção de uma extensão de acesso aos *Webservices* da AuditMark, o outro ponto de desenvolvimento deste projecto.

Assim, neste capítulo são efectuadas análises e comparativos sobre as várias opções existentes no mercado e as suas características.

3.1 *Analytics* e *Adservers*

Este subcapítulo exibe alguns comparativos entre vários programas de gestão e análise de campanhas publicitárias.

3.1.1 Comparativos

Este tipo de ferramentas pode ser disponibilizado sob a forma de uma aplicação instalável ou alojável (*hosted*). A primeira caracteriza-se por ser possível de instalar num servidor mantido pelo utilizador, podendo alterar as configurações e o seu desempenho. Isto traz os benefícios de tornar o sistema independente de terceiros e completamente configurável, reduzindo também os custos de licenças de operação do software. No entanto, obriga a que se tenha um servidor na Internet e a suportar os custos a ele associados.

A versão *hosted*, por sua vez, não tem quaisquer custos de manutenção e não necessita que o utilizador tenha quaisquer conhecimentos na área de redes de computadores, base de dados, servidores, etc. Possui as mesmas funcionalidades que a versão distribuível, mas limita as configurações e a adição de extensões, estando dependentes dos gestores do sistema. Além disso, tem o custo de licenças de utilização, que associa um utilizador a uma dada conta de acesso ao programa.

Uma vez definido qual o tipo de disponibilização que se pretende, podem-se analisar as várias soluções existentes no mercado. Tanto as ferramentas de *Analytics* como de *Adserving* utilizam *scripts*,

p.ex. Javascript, que são inseridos nas páginas a monitorizar e que recolhe os vários parâmetros de interesse (endereço IP, browser, número de cliques num anúncio, etc.) enviando-os para o programa.

No caso das ferramentas de *analytics*, o grande objectivo é tratar toda essa informação e exibi-la aos seus utilizadores na forma de vários tipos de relatórios e grafismos de actividades, como o tempo dispendido em cada página, se a visita foi directa ou redireccionada de outro sítio, quantas e quais as páginas que viu durante a sua estadia, etc.

As ferramentas de *AdServer* vão um pouco mais longe, uma vez que é normal incluírem também esse tipo de informação, mas destinam-se sobretudo a gerir a publicidade online, podendo estes sistemas ser considerados locais – contêm os anúncios de um determinado *Publisher*, ou remotos - pertencendo a uma outra entidade que contém a publicidade de vários *Publisher*s. De modo a escolher qual o software que melhor se adapta à implementação a que nos propomos, criaram-se duas tabelas comparativas, uma exclusivamente para *Analytics* - Tabela 1 e outra para *AdServers* - Tabela 2.

TABELA 1 - SOFTWARE DE *WEBANALYTICS*

	Open Source	API	Importação de dados	Exportação de dados	Disponibiliza-ção
Google analytics	Não	SIM (Beta)	Não	CSV, texto, XML	Hosted
iwebtrack	NÃO	SIM	N/A	XML, Word, Excel	Hosted
Yahoo analytics	NÃO	NÃO	NÃO	CSV, Excel	Hosted
Nedstat	NÃO	SIM	NÃO	Excel	Hosted
Clicktracks	NÃO	SIM	NÃO	Excel, PDF	Hosted Instalável
Webtrends analytics	SIM	SIM	NÃO	PDF, CSV, Excel	Hosted Instalável

TABELA 2 - SOFTWARE *ADSERVER*

	OpenSource	API	Importação de dados	Exportação de dados	Disponibilização
Atlas Solutions	NÃO	Disponível (limiatado)	Não (em desen-volvimento)	Excel, Página Web, XML	Hosted
Helios IQ	NÃO	Disponível (limitado)	NÃO	Excel, CSV and XML	Instalável
DART	NÃO	Disponível (limitado)	SIM	SIM	Instalável
Adbutler	NÃO	Indisponível	NÃO	CSV	Hosted
Zedo	NÃO	Disponível (limitado)	NÃO	CSV, HTML, PDF	Instalável
OpenX	SIM	Disponível	SIM	Excel	Hosted Instalável

Foram considerados para parâmetros de comparação os mais proeminentes e de maior impacto para uma boa escolha, como o facto de serem Open Source, possuírem uma API que permita aceder a determinadas funcionalidades (especialmente útil caso não se tenha acesso ao código fonte), ser possível importar e exportar dados do programa (bastante crítico devido a ser essa a principal utilização que se iria ter do programa) e a forma como é disponibilizado aos utilizadores. Desta forma, consegue-se basear a escolha do programa mais indicado para o desenvolvimento em causa.

Começa-se assim com a mais pertinente das questões, se se deve optar por *Adserver* ou *Webanalytics*. Este ponto relativiza-se um pouco, já que sendo o objectivo o de atingir o máximo número de pessoas e entidades possível, poder-se-ia pensar que uma análise directa entre os vários programas trouxesse mais vantagens. No entanto, não nos podemos esquecer de que se pretende fornecer um serviço de auditoria complexo e em constante desenvolvimento e não um simples serviço isolado. Como tal, é requerida uma plataforma que forneça um suporte forte para a recolha de variados tipos de dados.

Vejamos o caso dos serviços de *Webanalytics*. Embora algumas das soluções indicadas sejam muito populares e de boa qualidade, como analisadas em [19] e [20], têm um objectivo diferente do pretendido para o trabalho a desenvolver. Isto porque este tipo de software permite recolher variada informação sobre um determinado sítio na internet mas limita-se a ser um observador passivo de toda a publicidade, *cliques*, acessos, etc. que lá ocorrem. Embora útil, este tipo de recurso mostra-se algo limitado para o tipo de análise a que nos propomos.

Comparativamente, um serviço de *AdServer* tal como o próprio nome indica, disponibiliza e gere variados tipos de publicidade. Assim, disponibilizam várias ferramentas de controlo e publicação dos anúncios. Desde o controlo de várias zonas de publicação diferentes, à execução de análises mais orientados para a gestão dos seus produtos.

Como tal, o modelo de negócio adoptado pelos utilizadores de *AdServer* e aquele que se pretende seguir neste projecto convergem, sendo muito mais propício de sucesso e aceitação se o desenvolvimento for adoptado para um dos vários programas existentes. Decidiu-se, então, concentrar toda a atenção na análise comparativa dos programas listados na Tabela 2.

Atlas Solutions

Como pode ser visto na Tabela 2, o software disponibilizado pela Atlas Solutions exibe alguns pontos fortes, como sendo a disponibilização de uma API e a possibilidade de exportação de dados. Qualquer um destes pontos é crucial para uma boa integração de programas criados por terceiros, que obviamente requerem dados internos do *AdSever*. Ainda assim, é um sistema proprietário, o que acarreta alguns pontos negativos, já que é necessário adquirir licenças.

Verifica-se imediatamente que é necessária a compra de uma conta para aceder ao serviço, o que implica custos acrescidos no desenvolvimento da aplicação, o que se faria notar no tipo de disponibilização da integração final do nosso serviço. Além disso, a API é fornecida juntamente com o programa, estando assim o acesso a desenvolvimentos futuros bastante limitado, logo à partida.

Uma terceira questão que inviabiliza o uso deste software é a inexistência de importação de dados. Embora esteja em desenvolvimento, não permite que se executem os planos definidos e que se cumpra a janela temporal planificada.

Por último, verifica-se que é uma solução que é disponibilizada no formato *Hosted*, ou seja, implica constante acesso e dependência dos servidores da Atlas Solutions.

Conclui-se assim que embora seja uma aplicação robusta e com bastantes funcionalidades, não se adequa ao trabalho que se pretende implementar, tem custos e é mantido por uma companhia que demora um pouco a reagir às tendências de mercado, conforme comentado no artigo [3].

HeliosIQ

Tal como a aplicação anterior, o HeliosIQ disponibilizado pela ADTECH, fornece uma API e a possibilidade de exportação de dados, nomeadamente XML e Excel. É igualmente necessário adquirir uma licença de utilização do software, que neste caso é disponibilizado no formato instalável. Ou seja, cada cliente é responsável pela instalação e manutenção dos servidores e do respectivo software.

A API só é fornecida aos clientes da empresa, pelo que seria necessário obter uma conta apenas para o desenvolvimento das extensões, o que é uma situação longe da ideal. Também negativo é o facto de não ser possível realizar importação de dados para a aplicação, estando dependentes de futuras actualizações da API que o possam vir a fornecer.

DART

DART é um programa criado pela DoubleClick, com várias versões, incluindo DART for Advertisers, DART for Publishers e DART Enterprise. Todas estas versões se baseiam na mesma plataforma de Adserving, e fornecem um excelente leque de funcionalidades. É disponibilizada uma API que permite exportar e importar dados e é distribuída no formato instalável. Todas estas vantagens tornam o sistema no melhor AdServer, segundo a conceituada entidade Clickz [5].

Ainda assim, sendo software proprietário, o acesso às funcionalidades implica a compra de uma licença de uso, o que se torna um ponto extremamente negativo face aos nossos propósitos, que em nada necessitam do uso da aplicação, mas sim da API que esta fornece. Como tal, implica um custo de desenvolvimento que não é amortizável.

Em complemento, a dificuldade em obter qualquer informação específica sobre o sistema, como funcionalidades da API, tipos de dados que Importa/Exporta, bem como a não existência de qualquer software desenvolvido por terceiros, levam à conclusão de que provavelmente não será a melhor escolha para a implementação.

AdButler

Um serviço alojado de AdServing, destinado ao mercado *mainstream*, apenas com funcionalidades básicas, sem API e sem importação de dados. Embora seja possível retirar informação da aplicação, de pouco uso face às restantes características bastante limitadoras. Sendo também pouco mencionado em artigos da especialidade, conclui-se ser uma aposta pouco atractiva para ser a escolha final.

Zedo

Uma proposta interessante, criada pela Zedo, o Zedo AdServer possui algumas características necessárias para os objectivos propostos, nomeadamente uma API, a exportação de dados em formatos como PDF, HTML ou CSV. É também fornecido no formato instalável que, como já foi dito, permite ao utilizador usar os seus servidores e fazer a sua própria gestão do programa.

Mais uma vez, não existe a possibilidade de importar dados, o que limita as funcionalidades implementáveis. Ainda assim, a globalidade atingida pela empresa com escritórios espalhados pelo mundo e o crescimento a que se têm proposto mostra um bom perfil de apoio ao cliente. No entanto, o facto de não permitir importar dados e de não permitir que tal seja implementado (p.ex. através da API ou como aconteceria se fosse *Open Source*) não faz deste software a melhor escolha deste leque.

OpenX

Por último, analisa-se o software OpenX, que apresenta um conjunto de funcionalidades como mais nenhum foi capaz de fornecer. De salientar que é uma aplicação de software livre, pelo que não tem qualquer custo associado. Este facto não tem só vantagens em termos de custo por ser gratuito, mas também de funcionalidades, uma vez que estas podem ser implementadas sob a forma de *Plug-Ins*.

É também disponibilizada uma API que permite o acesso a dados internos por via do protocolo XML-RPC, tanto para leitura como para escrita. A exportação de algumas informações em formato Excel só é possível para os relatórios padrão que vêm com a aplicação.

Outra vantagem deste programa é a possibilidade de se escolher entre utilizar uma versão instalável ou alojada, consoante as necessidades do utilizador final.

3.2 WebServices

Este subcapítulo aborda o tema dos serviços WEB, analisando e comparando as tecnologias existentes.

3.2.1 Conceitos

A disponibilização de determinadas funcionalidades criadas por uma empresa, entidade ou outro, de forma segura, prática e rápida é um problema que está sempre na mente dos programadores. Como tal foi um ponto que mereceu atenção redobrada neste projecto e ao qual é dedicado este subcapítulo.

Os objectivos especificavam a utilização de uma estrutura de *WebServices*, ou seja, que permitisse aos utilizadores aceder a conteúdos e efectuar as mais variadas operações pela internet, num formato de troca de mensagens padronizado. O conceito em si é bastante simples, permitindo a um cliente produzir/consumir um conjunto de dados automaticamente ou com interacção humana, que lhe permite actualizar bases de dados, recolher relatórios de actividades, ou realizar qualquer outra acção previamente definida no serviço que subscreve. A grande utilidade deste tipo de arquitectura é o facto de os clientes poderem recolher/inserir dados guardados em base de dados sem que a esta tenham acesso

directo, e de se poder efectuar uma filtragem dos conteúdos a que cada um tem acesso com base nas credenciais de autenticação.

Sobre essa base pode ser montada toda uma estrutura mais complexa que aplique *load-balancing* no acesso, registos próprios de listagem de serviços, e outras funcionalidades implementadas por exemplo em *ESB*'s – *Enterprise Service Bus*, sendo por isso fortemente escalável.

Existem vários protocolos para a troca de mensagens entre os intervenientes, podendo até serem usados vários conforme a preferência do utilizador e as implementações disponibilizadas pelo servidor. Como tal, são analisadas de seguida as duas tecnologias mais recentes e populares utilizadas neste tipo de implementação.

3.2.2 Tecnologias

Existem duas grandes tecnologias de fornecimentos de *WebSevices* na actualidade: REST e SOAP. Ambas são largamente utilizadas na indústria, tendo abordagens diferentes na sua construção e utilização. Verifica-se que não existe grande consenso sobre qual é a melhor alternativa, variando muito com as necessidades específicas de cada implementação. A Figura 3 pretende ilustrar as principais características de cada um dos protocolos, onde se pode ver que as duas tecnologias são vistas mais como complementares do que concorrentes. SOAP é orientada a comunicações com requisitos de segurança mais elevados e com mais funcionalidades embebidas, enquanto REST é uma arquitectura mais prática, rápida e mais simples de manter. Estas características devem-se ao facto de optarem por uma abordagem diferente na sua fundação, sendo que REST foi sobretudo fruto dos trabalhos da tese de Doutoramento de Roy Fielding, envolvido também na especificação do protocolo HTTP. Assim, toda a estrutura assenta na utilização do HTTP para fazer troca de mensagens XML, ao invés do habitual código HTML de uma página WEB. Desta forma, aproveita-se toda uma estrutura já criada e largamente aceite que já disponibiliza funcionalidades como cache e autenticação, e que não necessita de qualquer *Framework* ou biblioteca no cliente para consumo dos serviços.

Pelo contrário, SOAP precedeu por alguns anos a arquitectura REST, e desenvolveu-se a partir do conceito de *RPC – Remote Procedure Calls*. Como tal, torna-se mais lento e com menos funcionalidades nativas ao protocolo, tendo depois sido criadas várias normas que vieram adicionar várias características necessárias à sua utilização, mas que contribuíram para um aumento da complexidade. Normas essas, conhecidas como WS-*, que introduzem segurança acrescida, envio de anexos, entre outros.

Outro dos principais pontos de diferença face a arquitecturas REST é a existência de WSDL – *Web Service Description Language*, que não é mais do que um ficheiro XML que descreve o funcionamento do serviço: parâmetros de entrada e saída, URI de acesso, etc. havendo a necessidade de se utilizarem bibliotecas específicas para se consumirem os serviços disponibilizados nesta topologia.

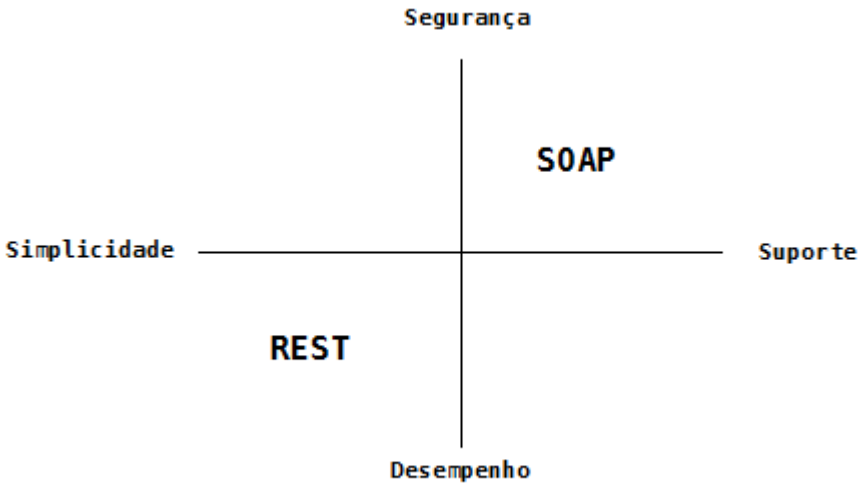

FIGURA 3 - CARACTERÍSTICAS DE SOAP E REST

3.2.3 Popularidade

Actualmente a utilização de REST/SOAP é bastante comum, existindo várias empresas que disponibilizam variados serviços sobre estas tecnologias. Casos como a Yahoo e Chiara_PEAR_Server optaram por utilizar serviços REST, enquanto que empresas como TVCabo, Sapo utilizam SOAP. Mas mais interessante do que utilizar uma ou outra arquitectura é usar ambas, permitindo ao utilizador escolher qual a que mais se adequa à sua estrutura. Assim, entidade como Amazon, Ebay, Flickr e hi5 fornecem ambos os tipos de acesso aos seus clientes.

Devido à grande diferença de abordagem do problema de disponibilização de WebServices levada a cabo pelas duas tecnologias, a concorrência não é directa e a escolha está mais dependente das necessidades impostas à partida. Assim, entidades que forneciam este tipo de acessos antes da arquitectura RESTful emergir, têm a tendência a optar pelo estilo RPC do qual o SOAP deriva, bem como aquelas que exigem elevado grau de segurança e maior controlo e especificação dos dados de entrada/saída.

No entanto, com o aparecimento de REST e da utilização mais eficiente da Internet com Web 2.0, p.ex. AJAX, verificou-se que a perspectiva de WebServices foi ligeiramente alargada e utilizada para várias outras situações que não as habituais até então, tais como a colocação de publicidade online. Assim, as novas implementações de serviços na Web começam a optar pela estrutura REST, mais adequada às necessidades de performance e simplicidade das aplicações actuais, enquanto as que já são utilizadas à vários anos ou que necessitam de segurança mais elevada e algumas funcionalidades específicas optam pela utilização SOAP.

3.2.4 Desempenho

REST:

- (+) Pedidos e respostas podem ser de dimensões bastante reduzidas em termos de bytes enviados.

SOAP:

- (-) Necessidade de incluir cabeçalhos SOAP que aumentam o *overhead* das mensagens a transmitir, tornando o protocolo 3 a 5 vezes mais lento, comparativamente com REST.

Nas Figura 4 e Figura 5, retiradas de [16], são apresentadas análises de desempenho de efectuadas num servidor ColdFusion que, para efeitos de teste de performance, disponibiliza serviços por SOAP e REST.

Conforme é visível, a utilização de REST resulta num aumento da performance, conseguindo processar cinco vezes mais pedidos do que o mesmo acedido por SOAP. Isto resulta do pedido do ficheiro de WSDL que é feito em cada acesso SOAP e dos cabeçalhos deste protocolo que aumentam o tráfego trocado no acesso.

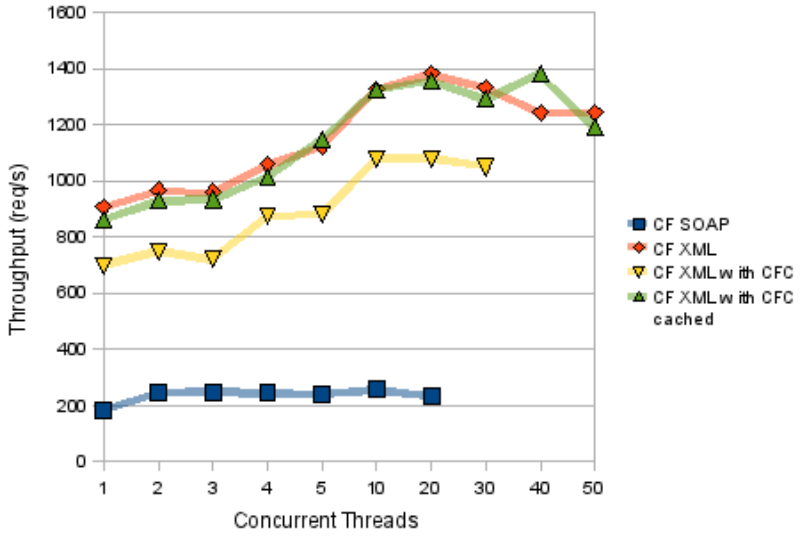

FIGURA 4 - BENCHMARK SOAP VS REST (XML)

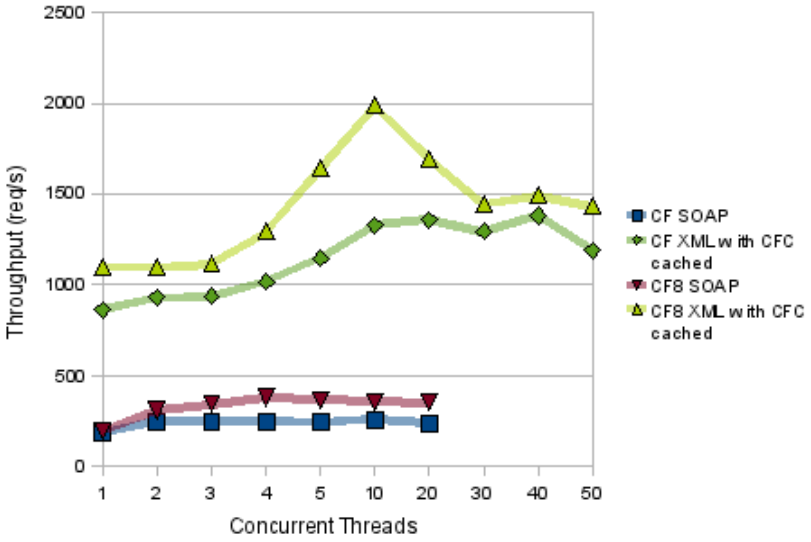

FIGURA 5 – BENCHMARK COM DUAS VERSÕES DIFERENTES DO SERVIDOR COLDFUSION

3.2.5 Acessibilidade

REST:

- (+) Uma vez que os pedidos se baseiam em comando do protocolo HTTP (GET, POST, PUT, DELETE), a utilização dos serviços é bem identificada como conteúdo HTTP. Isso reduz a hipótese de clientes estarem impedidos de aceder aos serviços em redes com elevadas restrições de tráfego (Bancos, Empresas de I&D, Farmacêuticas, Bases Militares, etc.).

- (+) O cliente apenas necessita de fazer pedidos HTTP e interpretar a resposta, que pode ser no formato XML ou até mesmo texto, diminuindo o custo de implementação por parte do utilizador dos serviços.

SOAP:

- (+) O facto de correr sobre http, usando consequentemente a porta 80, significa que todas as firewall que permitam acesso à internet também permitem aceder aos serviços.

- (+) O conceito de procedimentos remotos é muito mais simples de entender para o programador, que pode assim criar código invocando métodos de uma forma similar ao de muitas linguagens de programação.

3.2.6 Caching

REST:

- (+) Por definição, o protocolo REST utiliza as capacidades do protocolo HTTP, disponibilizando, consequentemente, as capacidades de *cache, hiperligação e bookmark*.

SOAP:

- (+/-) Embora não contenha suporte para tais funcionalidades, existem vários ESB que fornecem essas e mais funcionalidades, pelo que o uso deste protocolo não traz qualquer desvantagem neste aspecto.

3.2.7 Segurança

REST:

- (+) Uma vez que os pedidos são identificados por URIs e enviados por http/https, a sua filtragem no acesso à rede do servidor é simples de fazer, não exigindo mais do que ACL's – Access Control Lists.

- (+) Existe a garantia de que se for dada apenas permissão de recolha de dados (apenas permitir o comando GET no controlo de acesso) esta é inviolável. Isto porque esse comando é, por definição, apenas de recolha de dados.

- (-) O facto de a segurança ser apenas implementada por túneis SSL/TLS, permite que esta possa ser comprometida nos terminais intermédios (p.ex. proxies) da ligação, tal como ilustrado na Figura 653, retirada de [17].

SOAP:

- (+) Existem estritas regras de segurança para o uso deste protocolo, WS-Security, WS-Trust, entre outras, que fornecem elevados padrões de segurança ao protocolo.

- (++) A segurança é orientada à mensagem e não à ligação, pelo que não depende de túneis SSL/TLS que acabam em proxies e que podem comprometer a segurança, tal como ilustrado na Figura 6.

- (-) O pedido do serviço é feito por POST, pelo que a identificação na firewall do conteúdo do pedido é complexa e consumidora de recursos.

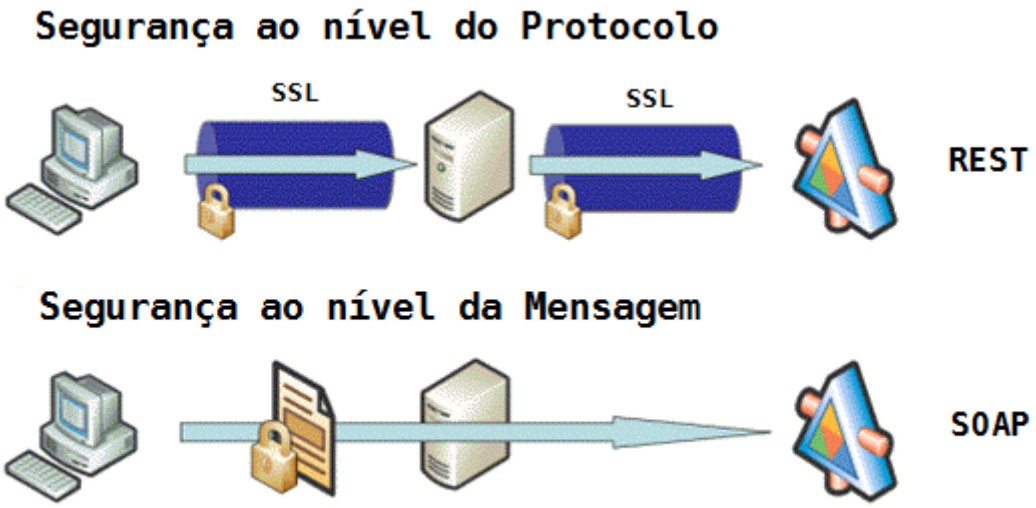

FIGURA 6 – TIPOS DE SEGURANÇA NOS PROTOCOLOS REST E SOAP

3.2.8 Escalabilidade

Devido ao crescente interesse em *WebServices* existe uma grande variedade de ferramentas e *Framework* com excelente suporte e funcionalidades. Tanto REST como SOAP são responsáveis pela quase totalidade dos serviços prestados através da Internet, pelo que qualquer serviço criado usando esse conceito está na vanguarda da tecnologia.

Já em termos da capacidade de desenvolvimento, verifica-se que ambas têm estruturas muito diferentes, mas capazes de implementar um elevado número de serviços. Para SOAP, o desenvolvimento implica escrever e publicar os ficheiros WSDL bem como o serviço, o que implica algum *overhead* na produção, manutenção e desenvolvimento.

Em termos de REST, os URI's podem ser apenas lógicos, pelo que não existe a necessidade de ter ficheiros descritores do serviço bem como os serviços a fornecer, removendo complexidade ao processo. No entanto, com um URI por serviço implica tem de existir uma boa organização de endereços, caso contrário à medida que o número aumenta a gestão de versões e de disponibilizações pode-se tornar demasiado complexa.

Ambas as tecnologias permitem o crescimento e fornecimento de cada vez mais funcionalidades e serviços, mas envolvem sempre uma boa gestão e planeamento de forma a manter uma qualidade de serviço elevada e compatibilidade com as aplicações cliente que os irão consumir.

3.2.9 Frameworks

Nesta secção irão ser listadas e comparadas algumas *Framework* para implementação de *Web-Services* SOAP, REST e SOAP + REST. As Figura 7, Figura 8 e Figura 9 apresentam alguns *benchmarks* efectuados a algumas dessas ferramentas.

SOAP :

BSOAP

Versão escrita em C++ e criada sobretudo para investigação, de modo a reduzir as perdas de performance na serialização e desserialização na conversão de binário para ASCII e vice-versa. Não possui, contudo, todas as funcionalidades de XML e outras funcionalidades SOAP. Pode ser obtida em [21].

GSOAP

Esta *Framework* está escrita completamente em C/C++, e disponibiliza todas as funcionalidades de SOAP e XML, bem como auto-geração de WSDL e XSD a partir do código. É utilizado por empresas como a Siemens e a Boeing, conferindo-lhe alguma credibilidade. Pode ser obtido em [23] e consultado o seguinte artigo: "*The gSOAP Toolkit for Web Services and Peer-To-Peer Computing Networks"* [22].

WS/XSUL2

Uma API em JAVA com bastantes funcionalidades, que implementa WSDL, XSD, TLS, WS-Security, WS-Addressing, assincronismo entre várias outras. Apresenta-se como uma das melhores opções para implementações SOAP. Pode ser obtido em [24].

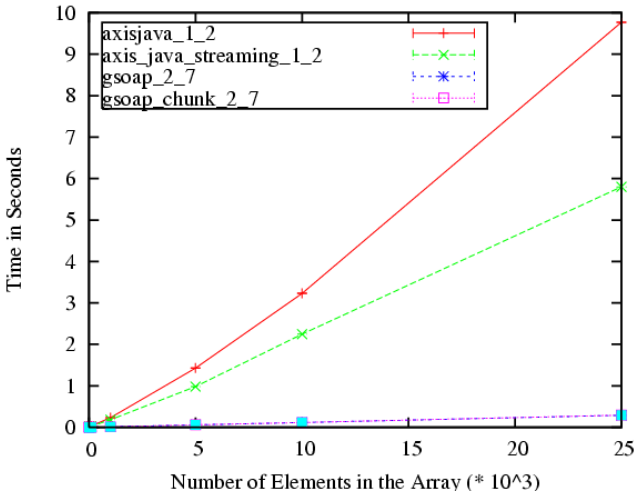

FIGURA 7 - AXIS/JAVA VS GSOAP

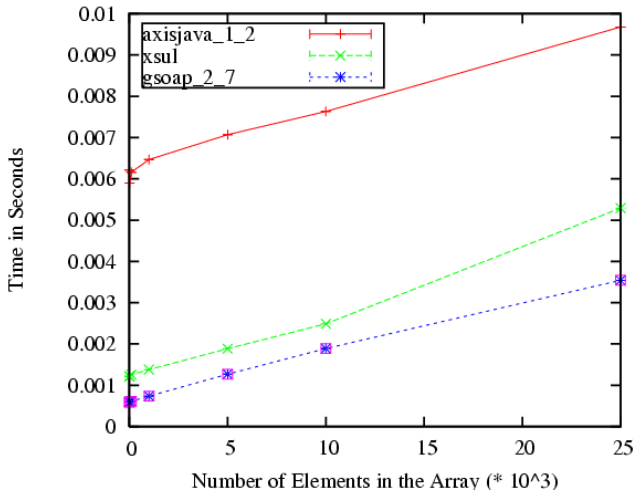

FIGURA 8 - AXIS/JAVA VS XSUL VS GSOAP

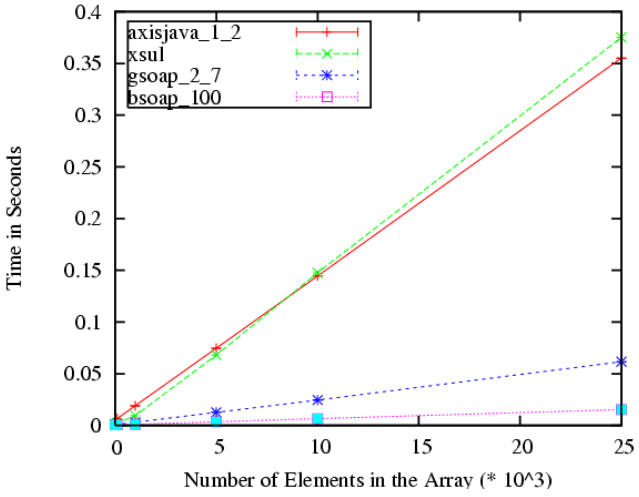

FIGURA 9 - AXIS/JAVA VS XSUL VS GSOAP VS BSOAP

REST:

Nesta secção, todas as *Framework* apenas suportam o protocolo REST. Deve-se salientar que REST, sendo mais uma arquitectura do que um protocolo, não necessita necessariamente de uma Framework para a sua implementação. No entanto, estas são uma ajuda preciosa.

Restlet

A API mais utilizada para criação de serviços REST, escrita em JAVA. Já implementa segurança vi a SSL, especificações WADL, representações JSON, autenticação http e a resolução JAX-RS 1.0. Pode ser obtido em [25].

Simpleweb

Não é propriamente uma Framework para criar serviços REST, mas sim um servidor com um conector que se pode utilizar conjuntamente com Restlet, com uma performance superior ao Jetty e Tomcat. Pode ser obtido em [26].

SOAP + REST:

WSO2

Baseada numa das mais influentes e conhecidas *Framework* existentes, o AXIS2 desenvolvido pela fundação APACHE, a WSO2 estendeu as suas funcionalidades criando várias *Framework* em diferentes linguagens. Isto porque enquanto o AXIS2 vem em duas versões: C e JAVA, a WSO2 disponibiliza versões em PHP, Python, Perl, Ruby, Jython, C++, C e JAVA, bem como ESB e outras soluções.

Desta forma, permite que seja implementado o protocolo SOAP e, através de associações de URI a métodos, fornece também acessos via REST.

Nas Figura 10, Figura 12 e Figura 13 (retiradas de um teste de performance efectuado pelos desenvolvedores de software da WSO2, tendo em vista a comparação entre o AXIS2 baseado em C e em JAVA), fazem-se comparativos entre a versão com base em bibliotecas JAVA e em C.

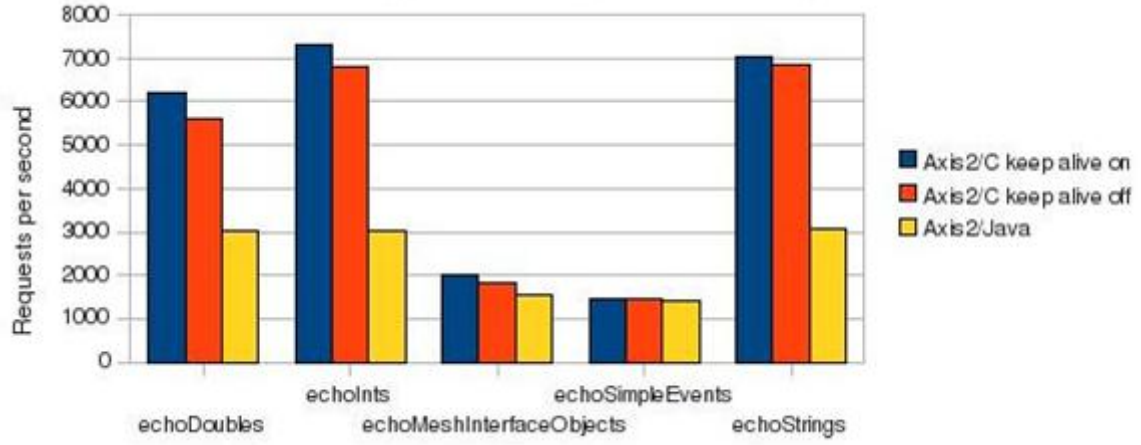

FIGURA 10 – NÚMERO MÁXIMO DE RESPOSTAS PARA TRANSFERÊNCIA DE XML COM 100 ELEMENTOS

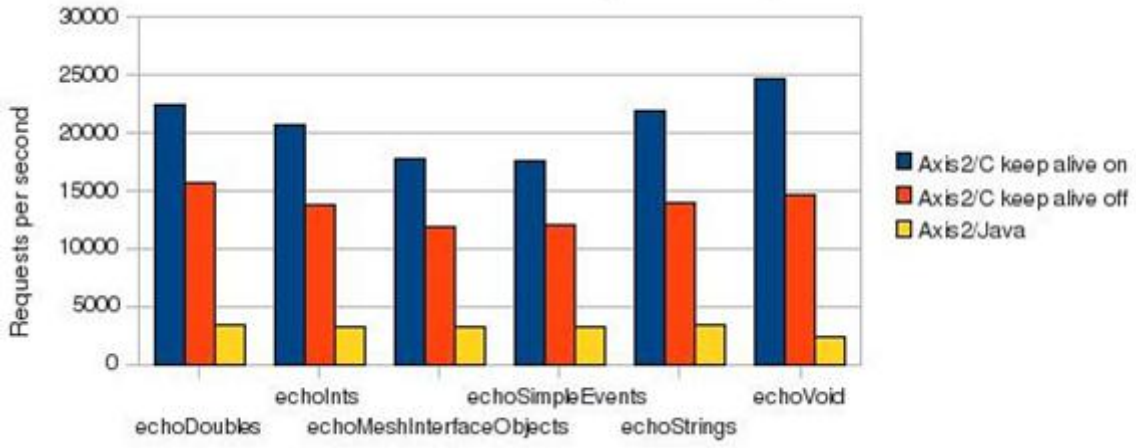

FIGURA 11 – NÚMERO MÁXIMO DE RESPOSTAS PARA TRANSFERÊNCIA DE XML COM 1 ELEMENTO

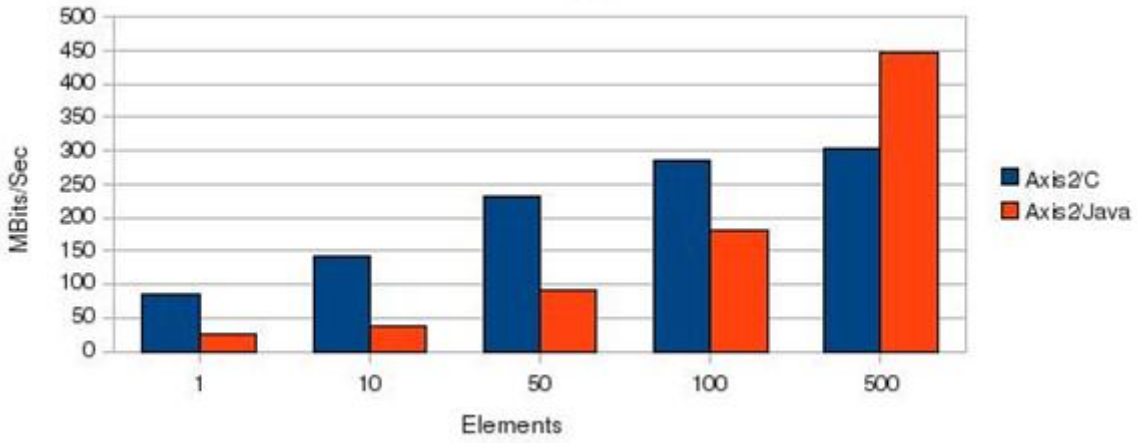

FIGURA 12 – VELOCIDADE MÁXIMA DE TRANSFERÊNCIA

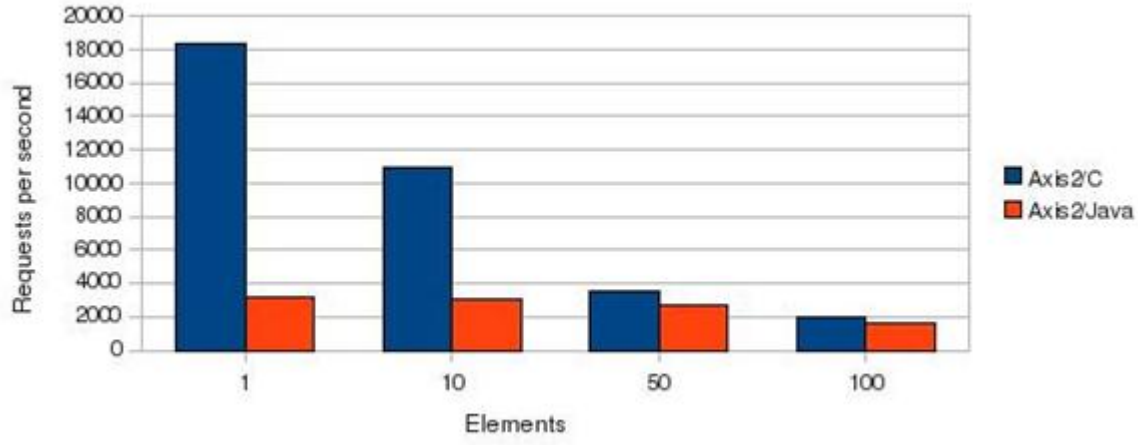

FIGURA 13 – NÚMERO MÁXIMO DE RESPOSTAS PARA XML COM DIFERENTES NÚMEROS DE ELEMENTOS

Conclui-se, então, que as versões baseadas em código C são mais eficientes do que as versões em JAVA, sendo muito mais rápidas nas respostas a pedidos. Assim, e como a WSO2 Framework/PHP corre sobre AXIS2/C, torna-se uma das melhores escolhas em termos de desempenho.

Além disso, contém todas as habituais funcionalidades, como várias políticas WS-*, MTOM, interoperabilidade com .NET e J2EE, auto-geração de código PHP a partir de WSDL, entre muitas outras. Pode ser obtida em [27].

Página em branco

Capítulo 4

Neste capítulo serão abordadas as escolhas e técnicas utilizadas para se implementarem os *Webservices* e a extensão para o OpenX, incluindo extras que se consideraram importantes para uma utilização óptima do sistema, como o caso dos aceleradores de PHP e cache do OpenX.

Implementação

4.1 Escolha de *Analytics/Adserver*

Após a análise das soluções mais populares encontradas no panorama da publicidade online, é necessário escolher aquela que servirá de base para o desenvolvimento da extensão e realização de testes da implementação.

Como já foi dito anteriormente, a utilização de software *WebAnalytics* foi descartada devido aos objectivos a que esse tipo de aplicações se propõe e que são desajustadas ao projecto. Assim, a escolha final recairia entre um dos AdServers incluídos na análise. A escolha foi bastante óbvia e directa e recaiu sobre o OpenX. Isto porque fornece não só uma API e as opções nativas de exportação de dados, bem como plataformas *hosted* e instaláveis, mas também por ser *Open Source*. Este facto traz grandes vantagens, uma vez que permite desenvolver código de forma a criar novas funcionalidades, não estando assim dependentes exclusivamente de funções da API. Assim, será também possível criar uma ferramenta de importação de dados que se adeqúe às necessidades futuras.

4.2 Escolha de *WebService*

Durante a pesquisa e escolha da melhor forma de implementar os WebServices, verificou-se que não existe uma solução óptima quando ao modo de o fazer. Consoante a necessidade de maior segurança/maior eficiência, maior simplicidade de implementação/maior tipagem e funcionalidades, assim deve ser feita a escolha. Além disso, descobriu-se que existem várias *Framework* disponíveis com características muito diferentes em termos de

performance e funcionalidades, bem como uma grande quantidade de linguagens de programação. Assim, optou-se por escolher a Framework da WSO2 em PHP, por mostrar uma elevada performance e robustez (já que é baseada em AXIS2/C), à qual se poderá associar um ESB para permitir usar sistemas como *caching* e *throttling*, e por permitir que tanto se implementem serviços em REST como SOAP. Com isto pretende-se disponibilizar uma base o mais alargada possível, permitindo aos desenvolvedores criar apenas um serviço que poderá ser consumido por duas formas diferentes, bem como deixar ao critério dos utilizadores qual a forma mais conveniente para os consumirem (REST ou SOAP).

4.3 Cache

A utilização de cache foi um ponto estratégico considerado praticamente desde o início do desenvolvimento. Os primeiros testes preliminares indicavam um exagerado tempo de acesso aos serviços disponibilizados, tanto maior quanto maior a quantidade de dados e complexidade do serviço, o que se traduzia numa baixa usabilidade do sistema. Tendo isso em consideração, analisaram-se possíveis soluções para o problema e verificou-se que os acessos às bases de dados estavam já optimizados e muito rápidos e que a ligações cliente-servidor eram dependentes de factores que não são possíveis de controlar. Ainda assim, não era aí que residia o grande atraso no fornecimento das respostas pedidas, mas sim no processamento dos pedidos no Apache. Uma vez que o código PHP que executa os pedidos se mantém igual por largos períodos de tempo, a pré compilação deste revelou-se uma excelente solução para este problema. Isto porque uma vez que PHP é uma linguagem interpretada, todo o processo de *parsing* e criação dos *op-codes* é apenas efectuado uma vez, acelerando o processo de execução do código nos acessos seguintes.

Após se ter verificado que a utilização de sistemas de cache para PHP aumentava a rapidez de acesso, ponderou-se também a utilização dum sistema de cache no OpenX que reduzisse o número de acessos ao servidor, resultando em menor tráfego Web tanto para o cliente como para o fornecedor, bem como maior velocidade na entrega de dados já recolhidos anteriormente.

De seguida são analisadas com mais pormenor as soluções adoptadas e a sua implementação.

4.3.1 *Cache* para PHP

Como foi indicado anteriormente, a cache para PHP era necessária para uma boa utilização dos recursos e para garantir uma qualidade de serviço elevada. Assim, pesquisaram-se algumas alternativas possíveis para o efeito tendo-se verificado que existiam já várias soluções disponíveis para implementação. Estas baseiam-se em extensões para PHP que podem ser instaladas em servidores como Apache ou Ligthppd e que pré-compilam o códi-

go para aumentar a velocidade de execução. Desta forma, garante-se uma integração perfeita com os servidores suportados por essas aplicações, evita-se o desenvolvimento de código específico para este propósito e fornecem-se acessos mais rápidos. Estes programas, também denominados de aceleradores de PHP, são comparados na Tabela 3 onde são apresentados os mais populares e as suas características gerais, e na Tabela 4 onde os mesmos são testados na execução de várias páginas, ambas retiradas de [28].

Conforme se pode verificar, a aplicação XCache é a que consegue melhores resultados no total dos testes, é compatível com Windows e Linux e com todas a versões de PHP a partir da 4.0, possui um interface Web para activação/desactivação e gestão das aplicações e ainda é de utilização livre com licença BSD. Assim, optou-se por utilizá-la para analisar a capacidade de resposta do serviço com um serviço de caching, ressalvando que a qualquer momento se pode alterar, uma vez que o serviço não está dependente de nenhuma aplicação de caching, para o seu correcto funcionamento.

TABELA 3 - SOFTWARE DE CACHING PHP – INFORMAÇÃO GERAL

Produto	PHP - Sem acelera-	Zend Platform	APC	XCache	eAccele-rator	ionCube PHP
Versão Testada	5.1.2	2.2.2	3.0.12p2	1.0.2	0.9.5	6.5
SO Suporta-	-	Windows Unix	Windows Unix	Windows Unix	Windows Unix	Windows Unix
Licencia-mento	-	Comercial	PHP	BSD	GPL	Comercial
PHP 4.x/5.0/5.	-	S/S/S/N	S/S/S/S	S/S/S/S	S/S/S/S	S/S/S/N
Tempo total	765.47s	597.48s	398.55s	392.8s	467.83s	808.33s
Ganho Total	-	128.12%	192.06%	194.88%	163.62%	94.70%
GUI	-	S	S	S	N	N

TABELA 4 - SOFTWARE DE *CACHING* PHP - DESEMPENHO

Teste	PHP - Sem acelerador	Zend Platform	APC	XCache	eAccelerator	ionCube
Hello World (1000)	7.80s	8.70s	7.66s	7.89s	5.78s	7.16s
Ganho	-	89.59%	101.84%	98.81%	134.86%	108.95%
MediaWiki Index (100)	44.67	32.59	21.86	21.78	23.36	47.31
Ganho	-	137.06%	204.36%	205.09%	191.24%	94.42%
MediaWiki Index (1000)	459.27	332.13	221.50	207.58	229.42	468.19
Ganho	-	138.28%	207.34%	221.25%	200.18%	98.09%
phpMyAdmin Index (100)	22.81	20.58	13.53	14.16	16.91	25.38
Ganho	-	110.86%	168.59%	161.15%	134.94%	89.90%
phpMyAdmin Index (1000)	230.92	203.48	134.00	141.39	192.36	260.30
Ganho	-	113.48%	172.33%	163.32%	120.05%	88.71%

4.3.2 *Cache* no OpenX

Tal como efectuado no servidor, ponderou-se a utilização de um sistema de cache na extensão criada para o OpenX, já que iria diminuir o número de transacções cliente-servidor, ao mesmo tempo, gerando assim menos tráfego e aumentado a velocidade de resposta do programa. Para tal necessitava-se de uma forma de guardar os dados que fosse o menos intrusiva possível no lado da aplicação do cliente, mas que se verificasse prática e

rápida. Existiam várias hipóteses, entre as quais a utilização da base de dados do OpenX, das variáveis de sessão, de ferramentas próprias de caching ou a escrita em ficheiros.

Uma vez listadas as possibilidades rapidamente se eliminaram algumas metodologias. Uma delas foi a utilização de ferramentas de caching, já que obriga à instalação de extensões de PHP e não nos é possível garantir que o cliente terá acesso a tais configurações. Como tal, assumir esse pressuposto para a implementação do sistema seria um erro e limitaria o número de utilizadores que tirariam proveito desta funcionalidade.

Uma segunda opção posta de parte foi o recurso a variáveis de sessão. Isto porque a quantidade de dados a guardar em cache pode ser elevada, reduzindo a performance do browser devido à grande quantidade de dados guardados em memória que são constantemente serializados. Além disso, os dados seriam perdidos com o encerrar do *browser*, o que impediria manter em cache dados cujos prazos de expiração fossem mais longos. Assim, restava a utilização da base de dados ou ficheiros e, após alguma pesquisa donde se ressalva o artigo [9], verificou-se que é bastante comum a utilização de ficheiros guardados no directório da aplicação (neste caso o OpenX e ficheiros XML) para esse propósito. Além de ser prático e de manter os dados guardados até expirar o tempo definido, evita que tenha de se aceder à base de dados do cliente para guardar dados de terceiros, anulando qualquer intrusão no resto do sistema.

Uma vez escolhido o ficheiro de *cache* como a melhor opção para se efectuar *caching* dos resultados da pesquisa, definiu-se que cada linha do ficheiro iria representar uma nova entrada de dados com o seguinte formato:

cacheID|expiryTime|CachedData

cacheID - Identificador da entrada de cache; permite distinguir cada uma das linhas para que se possam recolher os dados pretendidos.

expiryTime - Tempo Unix + número de segundos que se pretende manter os dados em cache; Este campo indica se os dados devem ser acedidos ou eliminados, consoante o tempo actual seja inferior ou superior a este parâmetro, respectivamente.

CachedData - Dados serializados com a função serialize do PHP contendo a resposta dos serviços Web. Desta forma consegue-se condensar todos os dados em apenas uma linha, sendo a acção reversível.

Antes de se efectuar o pedido do serviço aos servidores da Auditmark, é verificado se existe em cache alguma entrada com o mesmo cacheID. Se existir e se o período de validade ainda se verificar, os dados são recolhidos do ficheiro e exibidos, sem qualquer tráfego Web. Se não existir essa entrada, se esta estiver expirada, ou se o utilizador forçar um pedido dos dados ao servidor fazendo um *bypass* à cache (utilizando um opção específica na interface de utilizador), então são recolhidos novos dados do servidor e o tempo de expiração e os respectivos dados são actualizados.

Sempre que o utilizador aceder à página de sincronização da extensão da Auditmark, o ficheiro de cache é alvo de manutenção, eliminando-se todas as entradas expiradas. Além disso, existe uma opção na área de configuração do utilizador que permite limpar toda a cache (eliminar o ficheiro de cache).

4.4 Filas de Espera

Com o elevado número de pedidos que o servidor poderá vir a receber, surgiu a necessidade de saber se seria possível utilizar algum controlo de fluxo e reorientação de tráfego com base numa fila de espera. Embora ainda não existam infra-estruturas na empresa que possam tirar partido dessa situação, optou-se por verificar e demonstrar que é possível utilizar este conceito sem alterações de fundo no sistema. Para isso fez-se uma pequena pesquisa sobre filas de espera que fossem compatíveis com PHP e que permitissem uma boa integração com outras linguagens de programação, de forma a integrar-se facilmente com as aplicações já existentes e implementadas em Java, bem como outras que possam vir a ser desenvolvidas em C, Python, Ruby, etc.

Assim, verificou-se que apenas um produto correspondia ao que se procurava, o Apache ActiveMQ [29]. Das grandes vantagens que traz, o facto de ser desenvolvido pela Apache garante a qualidade e suporte que se pretende, é facilmente integrável com o Apache Webserver que é o servidor utilizado para disponibilizar os Webservices e é compatível com os módulos STOMP. Estes módulos não são mais do que conectores, que devem ser acoplados ao ActiveMQ e ao programa com o qual necessitemos de estabelecer uma ligação, efectuando uma abstracção entre as duas aplicações e permitindo utilizar qualquer das linguagens que estes conectores suportam para comunicar entre si (Ruby, PHP, Java, C++, Python, etc.). Estes podem ser obtidos em [30].

4.5 Versionamento

Uma vez que os serviços são constantemente actualizados, eliminados e criados, surgiu a necessidade de ter algum tipo de controlo de versão.

As modificações aos serviços podem ser compatíveis com versões anteriores, como por exemplo, se forem apenas adicionados novos métodos ao serviço. No entanto, pode haver a necessidade de alterar funcionalidades já implementadas, como a alteração de tipos de variáveis ou parâmetros de entrada e saída. Neste caso, os utilizadores que o consumiam devem poder ter um período de transição em que a versão anterior do código se mantém publicada, enquanto os novos acessos podem já ser feitos numa versão mais actual.

Foram pesquisados as formas mais populares de implementar o controlo de versões, donde se destacaram os artigos sobre as melhores práticas [6] e estratégias [7] de implementação de controlos de versão.

Verificou-se que existem quatro grandes conceitos usualmente utilizados para controlo de versões: a utilização de *namespaces* diferentes para cada versão do WSDL; o controlo dos registos UDDI de forma a disponibilizar todas as versões ao utilizador; a utilização de ESB com *pipelines/proxies/content based routing [8]* e a gestão de directório e de nomes de ficheiros/url.

O controlo de versão utilizado deve ser compatível tanto com SOAP como com REST. Por esse facto, qualquer que seja a opção escolhida deve sempre ter esse ponto em atenção. Na Tabela 5 apresentam-se alguns prós e contras de cada um dos tipos.

TABELA 5 - TÉCNICAS DE VERSIONAMENTO

	PROS	CONTRAS
Namespaces	Gestão baseada em URI; Integrada no WSDL não necessita de software extra ou de terceiros;	Útil apenas para o cliente verificar se está a consumir a versão correcta; Por si só não resolve o problema de disponibilização de várias versões do serviço;
Nome dos ficheiros + Gestão de directórios		Obriga a uma gestão manual e com especial atenção dos nomes de directórios e de ficheiros utilizados;
UDDI	Standard habitual para pesquisa de serviços;	Acesso do público em geral à informação dos serviços; Necessidade de actualização dos dados no repositório a cada alteração de serviços;
ESB	Disponibiliza muitas funcionalidades de gestão;	Implica software extra não utilizado nesta fase, que se traduz num overhead desnecessário; Impacto desnecessário na performance;

Uma vez que se começa agora o desenvolvimento da estrutura de *webservices*, e que não existe necessidade de utilização do vasto leque de funcionalidade de um ESB, evita-se essa escolha e poupa-se no *overhead* de manutenção, custo de performance e complexidade de implementação. Também o facto de não haver interesse em que os serviços sejam do domínio público, mas apenas fornecidos a clientes da empresa autenticados com as suas credenciais de acesso, e com o intuito de simplificar a manutenção desta estrutura de serviços a utilização de UDDI é deixada de parte.

Assim, opta-se nesta primeira fase por desenvolver os serviços com um controlo de versões mais rudimentar, mas muito mais prático e eficiente, como é a gestão de directórios e ficheiros.

4.6 *Web Design*

Tal com indicado nos objectivos deste projecto, foi necessário projectar e desenvolver uma extensão para o OpenX que fizesse uma demonstração das capacidades de utilização de webservices, e que facilitasse o acesso às funcionalidades desenvolvidas pela Auditmark aos utilizadores dessa aplicação. Utilizaram-se as técnicas mais recentes de *webdesign*, não só para tornar a utilização mais agradável e intuitiva, mas também para demonstrar que se pretende estar na vanguarda do desenvolvimento.

Toda a estrutura foi criada utilizando sempre uma arquitectura com AJAX, permitindo exibir dinamicamente as páginas sem recarregar todo o conteúdo do browser, bem como executar acções pedias pelo utilizador sem mudar de página. Para se obterem estas funcionalidades optou-se por utilizar a *Framework* JQuery, para a qual existem variados *plugins* com enormes potencialidades para desenvolvimento Web, e a qual estava a ser utilizada no desenvolvimento da plataforma WEB da empresa [15]. Isto permite aumentar o aproveitamento de código entre as duas aplicações e facilita a sua manutenção.

Foi também adoptada uma plataforma de exibição de gráficos no formato flash, que aumenta a apelatividade do sistema e que fornece uma análise visual dos relatórios escolhidos pelo cliente. Esta tecnologia é também utilizada nas restantes aplicações da empresa, pelo que permite uma maior compatibilidade de código e uniformização das interfaces de utilização.

4.7 Detecção de Cliques

Para que a Auditmark possa produzir análises sobre a qualidade do cliques que são efectuados num determinado anúncio, é necessário incluir o seu mecanismo de detecção juntamente com o do OpenX. Isto, porque, os clientes do OpenX gerem nessa aplicação as suas campanhas e é aconselhável que o sistema lhes seja o mais transparente possível, não tendo os utilizadores de criar novas rotinas apenas por terem aderido ao serviço prestado por outra empresa - Figura 14.

O OpenX já fornece um sistema que detecta cliques, embora bastante básico, já que não identifica a sua natureza nem faz qualquer tipo de validação. Para isso, inclui no código HTML do anúncio que irá ser exposto um pequeno javascript, que utiliza técnicas de AJAX para povoar a página com mais informação. De cada vez que for accionada a acção

mouseclick, é novamente executado um pedido utilizando AJAX, que vai actualizar a base de dados do OpenX com mais um clique.

Aproveitando esta estrutura, foi alterada a resposta já criada aos pedidos AJAX de forma a conter também o código javascript da Auditmark, conseguindo-se assim incorporar o mecanismo de validação da Auditmark no OpenX, sem que o utilizador sequer se aperceba de alterações no funcionamento e sem ter que alterar os seus hábitos de publicação e gestão de anúncios.

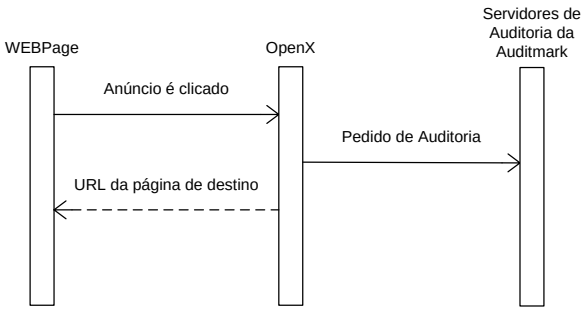

FIGURA 14 - DETECÇÃO E AUDITORIA DE CLIQUES

4.8 Autenticação

Sendo os Webservices um portal de acesso a informação confidencial dos clientes, foi necessário criar um sistema de autenticação que os protegesse. Como estão a ser disponibilizados dados através de dois protocolos diferentes - REST e SOAP – este mecanismo não podia ser específico a apenas um deles. Tal como descrito no artigo da Yahoo! [12], é necessário criar um conjunto de credenciais independentes dos restante tipos de acesso que o cliente possa ter. O principal objectivo é o de conseguir filtrar as operações a que este pode ter acesso bem como aumentar a segurança, impedindo que se consigam executar outras acções com as mesmas credenciais, como aceder ao portal de gestão online da conta de cliente.

Foi, então, criado um mecanismo idêntico ao já utilizado noutros sistemas como o "Amazon Webservices" [13], usando uma comum chave de acesso e palavra passe, mas com a particularidade de usar um mecanismo de encriptação HMAC, tal como descrito nos "Amazon Webservices" [14].

Para isso, usam-se os seguintes parâmetros:

1. *Client ID – (inteiro serializável)*
2. *Timestamp - (ISO-8601)*
3. *Acesskey - (20 caracteres)*
4. *Secretkey - (40 caracteres)*

A *Client ID* é um número inteiro, único e sequencial, usado para identificar o cliente na base de dados da empresa. Como cada cliente pode ter vários programas de acesso aos *webservices* são usadas *acesskeys* para os distinguir, em que cada uma possui permissões, restrições, campanhas, relatórios, etc. independentes, estando todas associadas sempre a um mesmo *Client ID*. Para fornecer alguma segurança contra *replay attacks* é também usado um *timestamp,* que utiliza o formato ISO-8601 e que permite saber a data/hora em que o pedido foi feito. Todos estes parâmetros são usados como entrada do mecanismo de cifra utilizado, o HMAC-MD5, conforme ilustra a Figura 15.

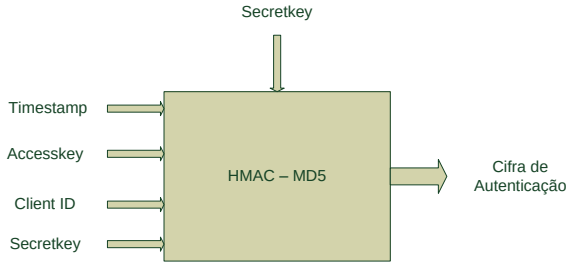

FIGURA 15 - MECANISMO DE CIFRAGEM HMAC-MD5

Os parâmetros 1 a 3 da lista anterior são utilizados como entrada na cifra e enviados em claro na mensagem, uma vez que o servidor tem de ser capaz de os identificar antes de fazer qualquer processamento. Já a chave secreta nunca é enviada em qualquer comunicação, sendo apenas do conhecimento do cliente e do servidor. Assim, com o conhecimento da *Client ID* e da *Accesskey* recebidas, o servidor pode recolher da sua base de dados a Secretkey correspondente e aplicar novamente a cifra. Se a cifra recebida e a realizada no servidor forem iguais, então continua-se com a resposta ao serviço pedido, caso contrário é negado o acesso - Figura 16.

Para impedir que um atacante conhecesse todos os parâmetros de entrada e saída da cifra e realizasse ataques de *known-plain-text*, foi também incluído como parâmetro de entrada a própria *secretkey*, baixando as probabilidades de um qualquer ataque poder ser bem sucedido.

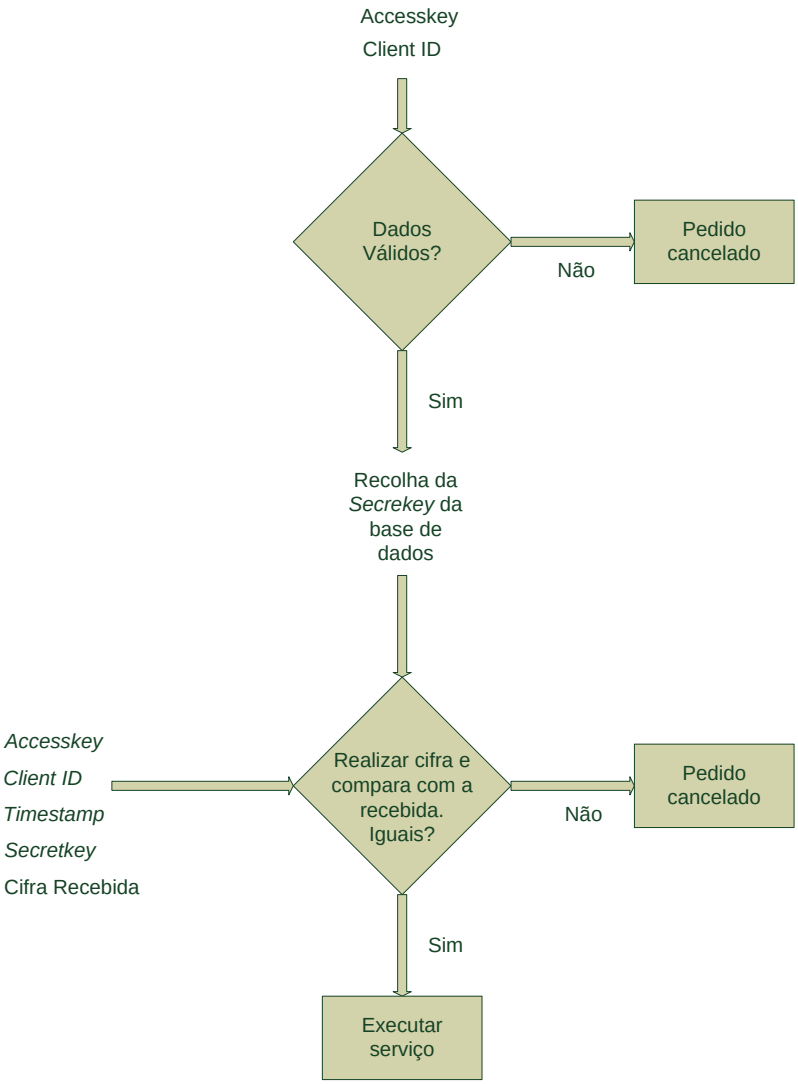

FIGURA 16 - AUTENTICAÇÃO

Página em branco

Capítulo 5

Exploração e Avaliação

Como em qualquer outro sistema, a disponibilização de aplicações na Internet deve ser cuidadosamente testada. É do interesse de pessoas mal intencionados tentar descobrir vulnerabilidades no código e aceder a conteúdo confidencial, eliminar dados, impedir o acesso aos sistemas com ataques de *denial of service*, entre muitos outros, o que pode tornar o sistema perigoso e denegrir a imagem das empresas que os criam.

Neste capítulo, exploram-se os conteúdos da interface gráfica da extensão criada para o OpenX e mostram-se os resultado de testes efectuados à segurança e ao desempenho das aplicações criadas, usando software de teste específico.

5.1 Exploração

Na extensão criada para o OpenX foram utilizadas as mais recentes tecnologias de WEB Design , tal como descrito no ponto 4.6. Foram então desenvolvidas as páginas de acesso aos variados conteúdos: “Home”, “Analysis”, “Settings”, “Export”, “Synchronize”, “Help”, conforme mostram as Figura 17 a Figura 22, respectivamente

Na página “Home” pretende-se apenas fornecer ao cliente uma perspectiva rápida do que está a acontecer com a sua publicidade. Assim, é exibido um somatório do total de cliques e do número de cliques inválidos, referente ao dia actual (no fuso horário definido nas configurações) e com uma escala horária. Desta forma é possível fazer uma análise fidedigna e expedita da situação e evolução das campanhas publicitárias que disponibiliza, podendo tomar-se as medidas necessárias para corrigir qualquer situação de fraude fora dos parâmetros habituais.

Para facilitar essa visualização, os dados são exibidos sob a forma gráfica e tabular, conforme ilustra a Figura 17, permitindo também a sua exportação para um ficheiro CSV, que pode depois ser tratado pelo cliente. Assim, não só é possível guardar a informação obtida diariamente, como também processá-la noutras aplicações.

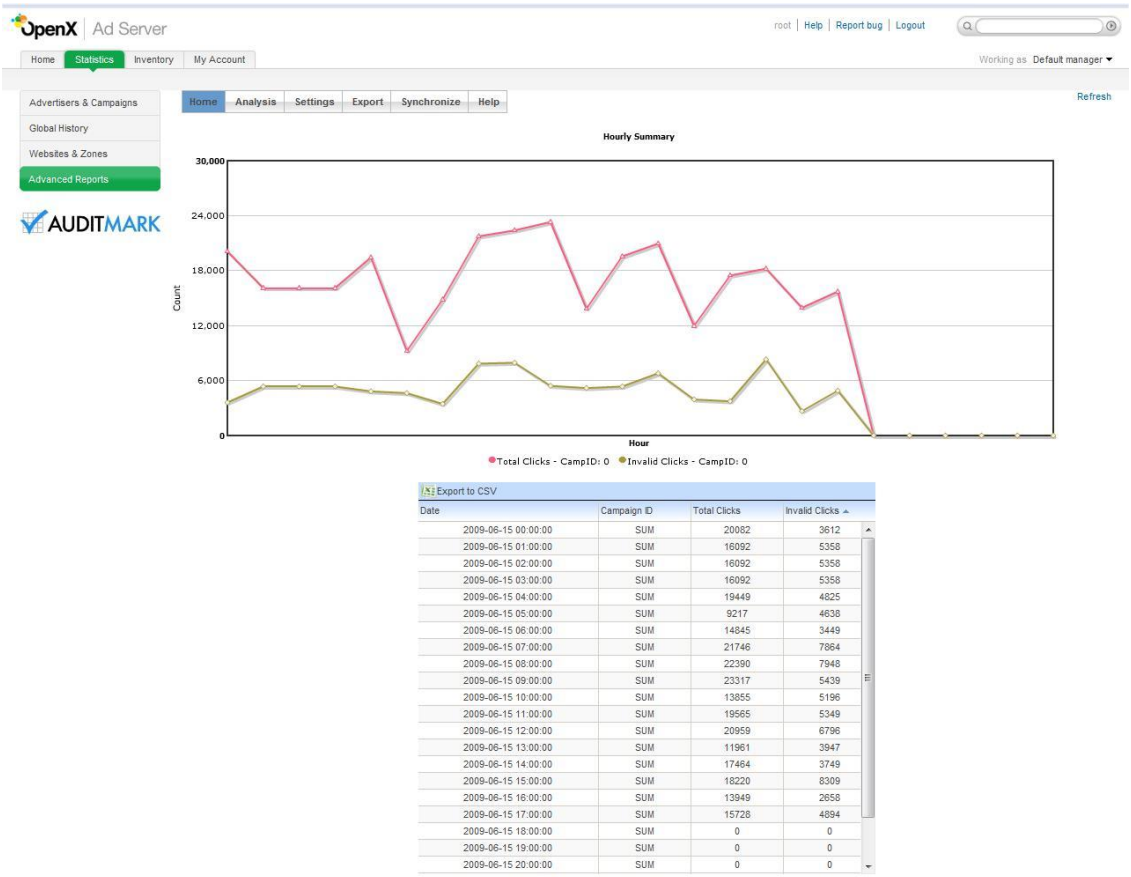

FIGURA 17 – PÁGINA "*HOME*"

Se o cliente pretender fazer pesquisas mais específicas, deve aceder à página "Analysis", ilustrada na Figura 18. Esta é composta por vários menus de selecção das opções pretendidas, bem com pela área de exibição dos dados processados. No menu superior são listadas todas as campanhas que o cliente possui, sendo dada a opção de escolher quais as que pretende analisar. No menu apresentado do lado esquerdo do painel é possível escolher as datas de início e fim da pesquisa, o fuso horário pretendido, o modo de apresentação (discretizada por campanha ou somatório das campanhas), e a escala temporal a usar para recolher e exibir dados (hora, dia, semana, mês, ano).

Após ter feito as escolhas, o utilizador tem a possibilidade de escolher efectuar a pesquisa normalmente ou com *bypass* à cache. Em condições normais, a consulta da cache é o método ideal de pesquisa, já que verifica se existe na cache alguma pesquisa com os mesmo parâmetros e, caso isso se verifique, exibe-os. Isto não só aumenta a rapidez de processamento mas também reduz o tráfego Web, uma vez que não é necessária qualquer comunicação com os servidores da Auditmark. No entanto, caso se pretenda forçar a uma actualização

dos dados pode-se optar por efectuar o *bypass* à cache, que realiza uma nova pesquisa nos servidores da Auditmark.

Também nesta página é possível visualizar os resultados sob a forma gráfica e tabular e efectuar a exportação para um ficheiro CSV.

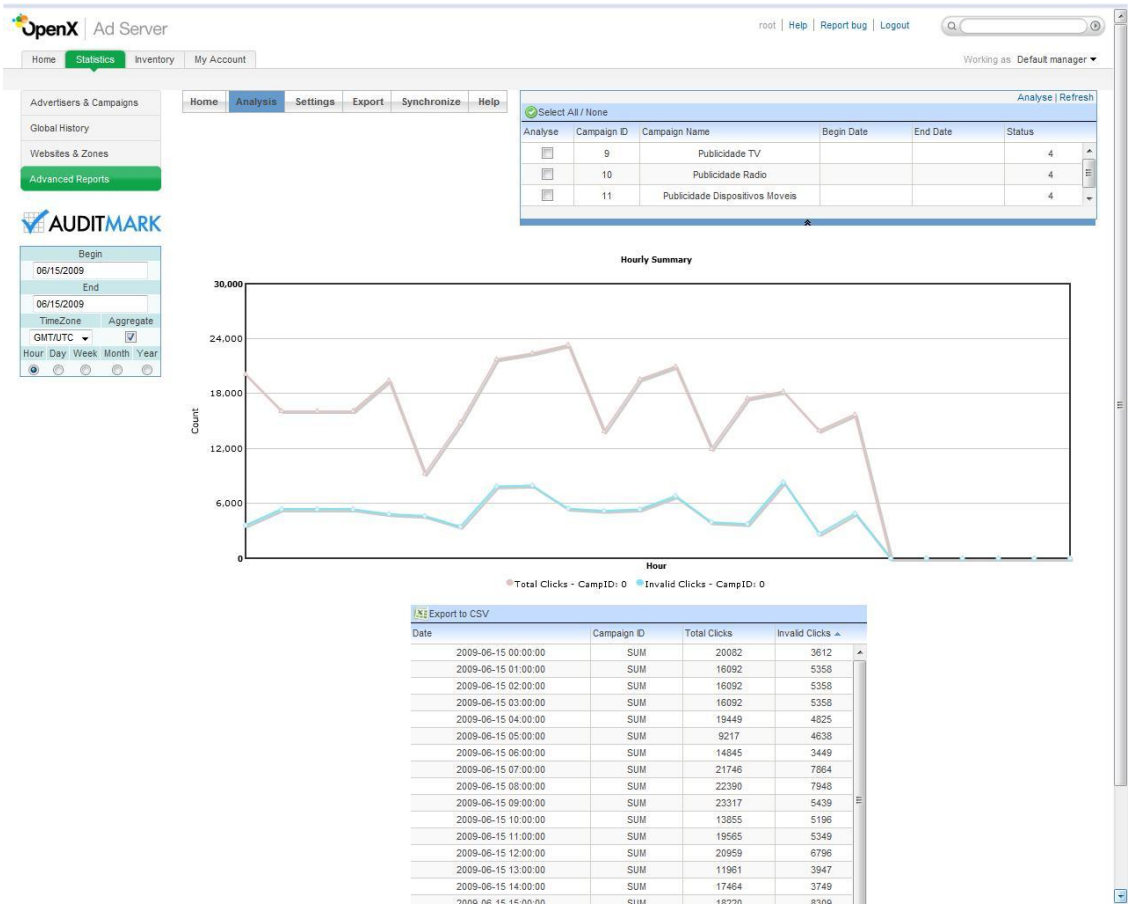

FIGURA 18 – PÁGINA "*ANALYSIS*"

Para que estas acções sejam possíveis é necessário que certos parâmetros sejam configurados. Com esse intuito, foi criada a secção de "Settings" conforme se vê na Figura 19. Esta página é composta por uma série de campos, dos quais apenas alguns são editáveis pelo utilizador, sendo os restantes preenchidos automaticamente durante a primeira execução da extensão desenvolvida para o OpenX. Assim evita-se que alguns campos críticos sejam alterados pelo utilizador inadvertidamente, mas permite-se a consulta para facilitar a resolução de algum problema que possa ocorrer.

Campos editáveis:

- *Auditmark Client ID* – Um identificador único do cliente perante as bases de dados da Auditmark. Fornecido ao cliente pela Auditmark, aquando do registo no site da empresa.
- *Accesskey* – Chave de acesso (*Username*) que identifica o cliente como utilizador dos *Webservices*
- *Secretkey* – Palavra passe de acesso aos Webservices.
- *Timezone* – Definição do fuso horário que deve ser utilizado como padrão para todas as operações.

Campos não editáveis:

- *Cache Expiry Time* – Tempo em segundos para manter os dados em cache. Por omissão é colocado o valor de 3600s (1hora).
- *Provider Key* – Chave que identifica a aplicação que acede aos serviços, uma vez que cada cliente pode ter várias aplicações diferentes a gerir diferentes conjuntos de campanhas.
- Os restantes campos são parâmetros de acesso à base de dados do OpenX. Estes campos são preenchidos automaticamente na primeira execução através da pesquisa no ficheiro de configuração do OpenX dos dados pretendidos, utilizando um *script* em PHP criado especificamente para esse efeito.

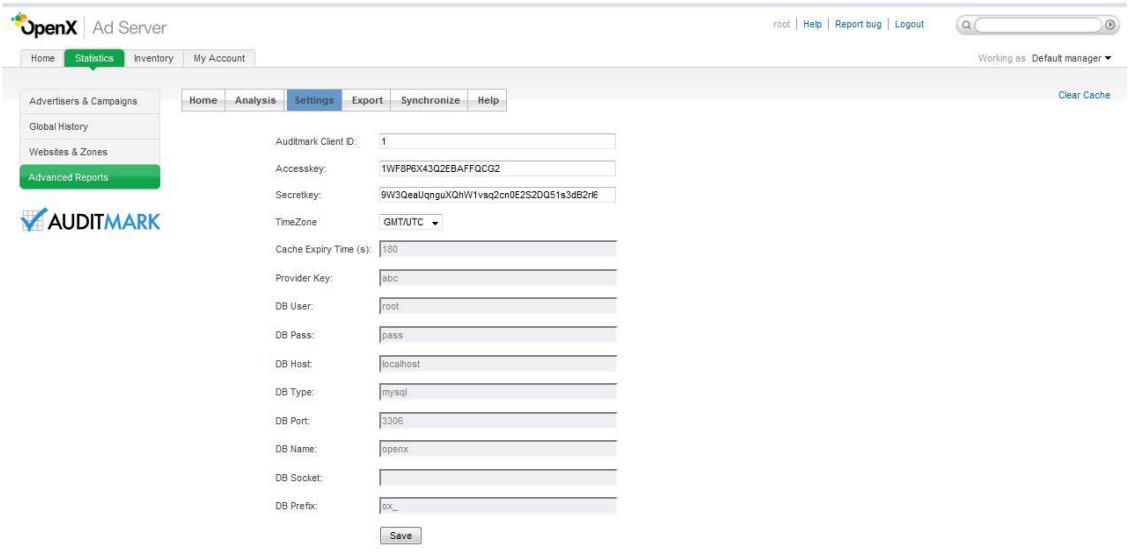

FIGURA 19 – PÁGINA "*SETTINGS*"

Todas estas configurações são guardadas num ficheiro XML, para facilitar a leitura e escrita das mesmas, bem como aumentar a sua escalabilidade. Assim, embora o utilizador esteja impedido de alterar algumas configurações a partir do página da extensão do OpenX, estas podem sempre ser alteradas directamente neste ficheiro. Com isto garante-te que estas não são alteradas inadvertidamente, mas que caso seja necessário realizar alguma operação mais delicada (p.ex. alterar a base de dados) a alteração dos parâmetros é tão simples quanto abrir o XML e alterar os parâmetros directamente.

Estas configurações são essenciais para que a opção exportação (explicada mais à frente neste subcapítulo) possa funcionar correctamente. Esta opção pode ser acedida através da página "*Export*", como mostra a Figura 20, e permite exportar todas as campanhas, anúncios e todos os outros parâmetros relacionados com campanhas publicitárias para um ficheiro CSV. Actualmente já é possível utilizar este ficheiro para importar dados para o servidor da Auditmark através do site da empresa. Além disso, a grande vantagem está em permitir que estes dados possam ser carregados noutras aplicações, simplesmente criando um *parser* que interprete esta informação, facilitando a migração entre aplicações.

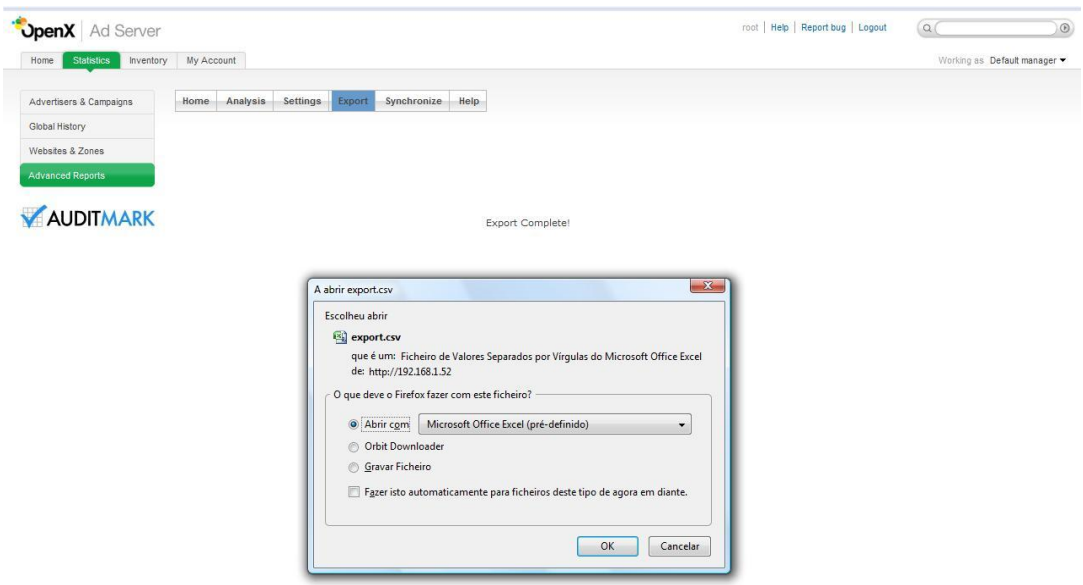

FIGURA 20 - PÁGINA "*EXPORT*"

Embora esta opção seja bastante útil, não é a forma ideal de manter o OpenX e o sistema da Auditmark sincronizados. Isto porque obrigaria a um controlo manual sobre o que já existe, o que deve ser actualizado e o que deve ser apagado. Assim, decidiu-se disponibilizar uma ferramenta de sincronização automática dos dados entre o OpenX e os servidores da Auditmark. A Figura 21 é um exemplo da página de Sincronização.

O método de funcionamento baseia-se num pedido inicial da listagem de campanhas existentes nos servidores que é depois comparada com as campanhas existentes no OpenX, sendo a cada uma atribuído um dos seguintes estados:

- Insert – A campanha ainda não existe nos servidores da Auditmark, pelo que será inserida.
- Update – A campanha já existe nos servidores da Auditmark, pelo que os seus dados serão actualizados/reescritos.
- Delete – A campanha já não existe no OpenX mas ainda está presente nos servidores da Auditmark, pelo que será apagada.

Estes dados são então exibidos ao utilizador, que tem a possibilidade de seleccionar individualmente quais as campanhas que pretende sincronizar, o que lhe permite, p.ex., apagar apenas campanhas obsoletas mas não actualizar nem inserir dados novos nos servidores. Uma vez terminada a inspecção e selecção das acções a tomar, basta prosseguir com a sincronização e esta é automaticamente processada.

FIGURA 21 – PÁGINA "*SYNCHRONIZE*"

Embora todas as funcionalidades sejam bastante intuitivas, foi também criada uma página de ajuda com alguns conteúdos sobre a utilização desta extensão. Desta forma, é possível adicionar respostas às questões mais pertinentes que possam surgir na utilização desta aplicação.

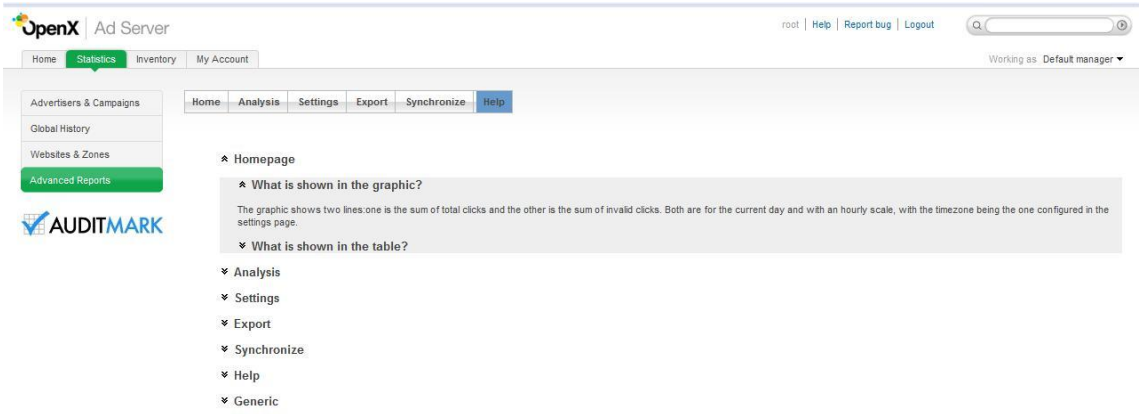

FIGURA 22 - PÁGINA "*HELP*"

5.2 Avaliação de Desempenho

Após se terem criado os serviços de acesso, tornou-se necessário testá-los e verificar a sua rapidez de resposta aos vários pedidos. Para isso, foi utilizada a ferramenta Apache Benchmark (AB) que é fornecida juntamente com o Apache Web Server. Esta executa um determinado número de pedido de um URL, sequencialmente ou em paralelo, medindo os tempos de acesso mínimo, máximo e médio, bem como o número de pedidos por segundo que se conseguem enviar.

Para se tirar partido dessas funcionalidades preparou-se um contexto de testes com as seguintes características:

- **Máquina Cliente:** MS Windows Vista Business SP1 32 bit, 1.5GB Ram, Intel T5500
- **Máquina Servidor de WebServices:** Ubuntu 8.10 Intrepid 32 bit (Virtual Box 2.0.4), 512MB Ram, Intel T5500
- **Máquina de base de dados:** Ubuntu 8.04 Hardy 64 bit, PostgreSQL 8.3
- **Pedidos de Serviços**:
    1. 1000 pedidos sequenciais sem acelerador de PHP (cache)
    2. 1000 pedidos sequenciais com acelerador de PHP (cache)
    3. 100 pedidos paralelos sem acelerador de PHP (cache)
    4. 100 pedidos paralelos com acelerador de PHP (cache)

Devido a limitações de hardware no PC que corria os *Webservices* em máquina virtual, o número de pedidos em paralelo teve de ser reduzido, para 100 pedidos, sem que isso tenha afectado a viabilidade do teste. Os resultados obtidos são exibidos na Tabela 6 e na Tabela 7.

TABELA 6 - PERFORMANCE NO ACESSO AOS WEBSERVICES

	Sequencial sem cache (1.)	Sequencial com cache (2.)	(2.)/(1.)	Paralelo sem cache (3.)	Paralelo com cache (4.)	(4.)/(3.)
Min. (ms)	83	42	**50,6%**	1232	601	**48,8%**
Med. (ms)	206	60	**29,1%**	28652	13255	**46,3%**
Max. (ms)	1002	632	**63,1%**	48750	29276	**60,1%**
Pedidos p/ Segundo	4.84	16.49	**340,7%**	2.04	3.40	**166,7%**

TABELA 7 - UTILIZAÇÃO DE RECURSOS

	Processador	Memória
Em Repouso	0.2%	72.3%
Resposta a pedidos	2.1%	72.4%

Conforme se pode constatar observando a Tabela 6, existe uma grande variação nos tempos de acesso com e sem acelerador de PHP, conseguindo-se diminuir em cerca de três vezes e meia o tempo médio de cada pedido sequencial e em duas vezes em pedidos simultâneos. Mas mais importante que isso é a redução significativa no tempo máximo, o que significa que mesmo no pior cenário de utilização o tempo de espera é bastante reduzido.

Também interessante é a quantidade de pedidos por segundo que é possível realizar, aumentado perto de 3.5 vezes para pedidos sequenciais e 1,5 para pedido simultâneos. Desta forma aumenta-se a disponibilidade do serviço para os clientes, sem necessidade de investimentos em mais servidores.

Analisando a utilização de recursos exibida na Tabela 7, também se conclui que não existe um impacto significativo na globalidade do sistema, que apenas aumentou em 0.1% na ocupação da memória e em cerca de 2% na utilização do processador.

5.3 Avaliação de Segurança

A questão da segurança deve ser igualmente posta à prova, uma vez que se pretende garantir o mínimo de risco para os clientes que utilizam o sistema. Com isso em mente, esco-

lheu-se uma plataforma que fizesse vários tipos de análise a aplicações Web: Paros [11]. Esta é capaz de realizar testes de injecção (*sql injection*, *cross site scripting*, etc.), obtenção de informação privilegiada, listagem de directório, entre muitos outros. A Tabela 8, Tabela 9 e Tabela 10 apresentam o resultado obtido numa execução do teste de segurança.

TABELA 8 - RESUMO DE AMEAÇAS

Nível de Risco	Número de Alertas
Alto	0
Médio	1
Baixo	2

TABELA 9 - AMEAÇAS DE NÍVEL BAIXO

Baixo	Descrição	Solução
#1	Ficheiros genéricos, *backups*, obsoletos e temporários presentes no mesmo directório. Estes ficheiros, se contiverem material sensível, podem ser exibidos no *browser*, pois o servidor pode não os processar devidamente.	Remover esses ficheiros da pasta desnecessários da pasta.
#2	Endereços Privados detectados nos campos de resposta http.	Implementação num servidor com IP público.

TABELA 10 - AMEAÇAS DE NÍVEL MÉDIO

Médio	Descrição	Solução	Referência
#1	É possível obter a listagem de directório. Pode revelar dados sensíveis em fiheiros contidos no directório.	Desactivar a listagem de directório.	• Em IIS, desligar listagem de directório. • Em Apache, usar a directiva 'Options -Indexes' para a desactivar. 52[10]

Conforme indica a Tabela 8, que não é mais do que um resumo do total de ameaças encontradas, é possível verificar que houve apenas 3 problemas encontrados, 2 de nível baixo e 1 de nível médio. Todos eles são resultantes do sistema se encontrar em sistemas de desenvolvimento, daí serem encontrados ficheiro temporário e de *backup* e endereços privados. A listagem de directório não tinha sido desactivada por se estar a correr o servidor Apache com as configurações padrão, uma vez que o objectivo é a implementação em qualquer servidor Web. Assim, as afinações do servidor são deixadas a cargo do responsável pelo sistema, consoante as suas necessidades específicas e das restantes aplicações instaladas.

É possível constatar que não existe qualquer erro relacionado com vulnerabilidades de acesso a bases de dados, erros de funcionalidades ou qualquer outro problema crítico no sistema. É também garantida a compatibilidade total com os *browsers* mais populares em utilização, como o Firefox 3, o Internet Explorer 7 e 8.

Capítulo 6

Conclusões e Trabalho Futuro

Tendo começado esta dissertação pela análise da publicidade online e seu modelo de negócio, verificou-se que esta indústria movimenta avultadas quantias de dinheiro e que existe interesse de várias partes em tentar defraudar o sistema. Esse modelo de negócio permite que se simulem cliques nos anúncios, retirando lucro por elevadas visualizações de publicidade que na verdade não passam de artifícios. Cria-se, portanto, lugar para o desenvolvimento de empresas e aplicações que disponibilizem técnicas de detecção dessas actividades e permitam às vítimas tomar as acções que entenderem. Nesse enquadramento, surgiu a motivação para este projecto, que forneceu a ligação entre o cliente e a empresa que fornece esse serviço. Foram, assim, investigadas as mais recentes tecnologias no âmbito dos *Webservices, Webdesign* e mecanismos para aumento de desempenho. A implementação destas ferramentas permitiu tirar algumas conclusões sobre o real funcionamento das várias componentes, isoladamente e em conjunto.

Numa primeira fase, foram criados os vários serviços que viriam a fornecer acesso ao *backbone* da empresa, foram configurados os servidores e foi instalados o acelerador de PHP. Isto permitiu verificar a capacidade de resposta desta ferramenta e confirmar a sua viabilidade para utilização regular. Os tempos de resposta que rondavam em média os 200ms sem utilização de cache conseguiram ser reduzidos para cerca de ¼, aproximadamente 60ms, com a cache providenciada pelo acelerador de PHP. Consegue-se, com a utilização deste sistema, um ganho significativo nos tempos de pedido-resposta de um determinado serviço, com a vantagem de ser independente da restante implementação, ou seja, pode ser desligado ou substituído por outra versão, sem que isso altere o normal funcionamento dos *webservices* (para além do tempo de resposta).

Verificou-se, também, que a utilização de *webservices* para fornecer os dados aos clientes é bastante segura, sobretudo se aliados a mecanismos de acesso às base de dados como as *stored procedures*. Isto, porque, conforme confirmado pelos testes de segurança, reduz o risco de ataques por *sql-injection* e evita que se tenha de fornecer aos utilizadores dados de acesso às BD's. Cria-se uma abstracção que torna a recolha da informação completamente

transparente para quem os consome, podendo toda a estrutura interna ser alterada desde que os parâmetros de acesso aos serviços de mantenham.

Uma vez terminada esta componente, criou-se a extensão para o software de *adserving* OpenX, que pretende demonstrar a utilização da componente abordada no parágrafo anterior, bem como servir de *showcase* para as funcionalidades e relatórios que a empresa disponibiliza.

A escolha da aplicação a usar mostrou-se acertada, tendo em conta que era *OpenSource* e fornecia alguma flexibilidade em termos de programação, bem como um bom suporte técnico online. Por outro lado, a API disponibilizada era algo reduzida, o que obrigou a que se acedesse directamente à base de dados do programa para recolher alguma da informação necessária para exportação. O restante desenvolvimento correu como previsto, tendo-se decidido adicionar uma estrutura de *caching*, após verificados os benefícios que isso tinha trazido no servidor. Mas, uma vez que não podemos forçar o cliente a instalar na sua máquina uma extensão para esse efeito, optou-se por seguir uma estratégia ligeiramente diferente. Criou-se um ficheiro de texto no qual eram guardados os dados serializados, o qual era consultado a cada novo pedido. Desta forma conseguiram-se reduções para cerca de 30% do tempo de exibição dos dados e uma redução de 10kb de tráfego Web, o que multiplicado por inúmeras visualizações e clientes se traduz em valores não desprezáveis.

A utilização da tecnologia AJAX, associada a gráficos Flash e tabelas Javascript, possibilitaram criar páginas rápidas e dinâmicas, que são apelativas e extremamente intuitivas para utilização, para além de estarem na vanguarda da tecnologia no *webdesign*. A integração com a aplicação OpenX mostrou-se viável, embora requeira algumas pequenas alterações do código fonte original. Isto, devido ao facto da ferramenta de *plugins* no OpenX ainda não estar suficientemente madura e bem documentada para se poder efectuar uma instalação directa através de um só ficheiro. Também o consumo dos serviços decorreu conforme esperado, embora com um número de dados disponíveis para recolha algo reduzido, mas completamente funcional, eficiente e seguro.

Conclui-se, assim, que os sistemas de sincronização automática, de exportação de dados, de ajuda, as *Framework* de gráficos e tabelas usadas para criar as páginas de exibição de relatórios, os serviços criados e toda a estrutura de optimização e de segurança implementados formam um conjunto de funcionalidades extremamente práticas e profissionais, que cumprem e ultrapassam todos os requisitos definidos no início do projecto.

No entanto, existem ainda muitas formas de expandir o sistema, adicionando mais serviços, disponibilizando acessos com autenticação e encriptação por chave pública/privada, fornecendo assincronismo nos acessos, utilizando filas de espera e implementado mecanismos de *load-balancing*, desenvolvendo extensões para outras aplicações, melhorando as funcionalidades e aspecto das interfaces, entre muitas outras possibilidades. Uma vez que tudo foi pen-

sado para ser escalável, é possível implementar uma grande variedade de mecanismos para continuar o projecto e assim continuar a inovação numa área em grande crescimento.

Referências

[1] LEINER, Barry M., CERF Vinton G., et al. 2003. “A Brief History of the Internet”. The Internet Society (ISOC)

[2] PricewaterhouseCoopers LLP, 2009. “IAB Internet AdvertisingRevenue Report – 2008”. The Interactive Advertising Bureau

[3] STERNE, Jim. 2007. “2007 Web-Analytics Shootout Final Report – Part1”. Stone Temple Consulting

[4] STERNE, Jim. 2007. “2007 Web-Analytics Shootout Final Report – Part2”. Stone Temple Consulting

[5] ACUFF, Courtney; ATCHINSON, Shane; BEARDSELL, Christine; et al. “ClickZ Marketing Excellence Awards”. ClickZ

[6] BROWN, Kyle, ELLIS, Michael. “Best practices for Web services versioning”. IBM Technical Library

[7] PELTZ, Chris, SUBBARAU, Anjali Anagol. “Design Strategies for Web Services Versioning”. SOA World Magazine

[8] LOUIS, Adrien, DUTOO, Marc. “Choosing between Routing and Orchestration in an ESB”. InfoQ

[9] GERVASIO, Alejandro. “Caching Result Sets in PHP: Cost-efficient PHP acceleration”. Dev-shed

[10] Apache Core Features. Disponível em: http://httpd.apache.org/docs/mod/core.html#options. Último acesso em: 23/06/2009

[11] Paros Software. Disponível em: http://www.parosproxy.org/. Último acesso em: 23/06/2009

[12] DEAN, Drew. “Authentication for web services”. Yahoo!, Inc

[13] Amazon Webservices. “SOAP without WS-Security”. Disponível em: http://docs.amazonwebservices.com/AWSSimpleQueueService/2007-05-01/SQSDeveloperGuide/NotUsingWSSecurity.html. Último acesso em: 25/06/2009

[14] Amazon Webservices. “HMAC Signatures”. Disponível em: http://docs.amazonwebservices.com/AWSSimpleQueueService/2007-05-01/SQSDeveloperGuide/SummaryOfAuthentication.html Último acesso em 25/06/2009

[15] SANTOS, Pedro. FEUP. “Sistema Integrado de Gestão de Plataforma de Auditoria WEB”. Relatório de Projecto submetido em 29/06/2009

[16] LYNCH, Mark. “CFMX – SOAP vs REST benchmarks”. Lynch Consulting. Disponível em: http://www.lynchconsulting.com.au/blog/index.cfm/2008/3/13/CFMX--SOAP-vs-REST-benchmarks. Último acesso em 27/06/2009

[17] Microsoft. “Why Web Service Enhancements”. Disponível em: http://msdn.microsoft.com/en-us/library/ms996935.aspx. Último acesso em: 27/06/2009

[18] GOVINDARAJU, Madhusudhan, HEAD, Michael, SLOMINSKI, Aleksander. “SOAP BENCH-MARKS”. Disponível em: http://grid.cs.binghamton.edu/projects/soap_bench/. Último acesso em: 27/06/2009

[19] STERNE, Jim. “2007 Web-Analytics Shootout Final Report – Part1”. Disponível em: http://www.stonetemple.com/articles/analytics-report-august-2007.shtml. Último acesso em 02/02/2009

[20] STERNE, Jim. “2007 Web-Analytics Shootout Final Report – Part2”. Disponível em: http://www.stonetemple.com/articles/analytics-report-august-2007-part2.shtml. Último acesso em: 02/02/2009

[21]LEWIS, Mike, Abu-Ghazaleh, Nayef. “bsoap”. Disponível em: http://www.cs.binghamton.edu/~nayef/bsoap/. Último Acesso em: 27/06/2009

[22] ROBERT, A. van Engelen, KYLE, Gallivan. “*The gSOAP Toolkit for Web Services and Peer-To-Peer Computing Networks”* páginas 128-135, 21-24 Maio, 2002, Berlim, Alemanha

[23]“gSOAP Toolkit”. Disponível em: http://www.cs.fsu.edu/~engelen/soap.html. Último acesso: 27/06/2009

[24]“WS/XSUL2”. Disponível em: http://www.extreme.indiana.edu/xgws/xsul/. Último acesso em: 27/06/2009

[25]“Restlet”. Disponível em: http://www.restlet.org/. Último acesso em: 27/06/2009

[26]“Simpleweb”. Disponível em: http://www.restlet.org/. Último acesso em: 27/06/2009

[27] “WSO2”. Disponível em: http://wso2.com/. Último acesso em: 27/06/2009

[28]“PHP Bytecode cacher review”. Disponível em: http://itst.net/wp-content/uploads/2006/10/PHP%20Bytecode%20Cacher%20Review.html. Último acesso em: 27/06/2009

[29] The Apache Software Foundation. “Active MQ”. Disponível em: http://activemq.apache.org/. Último acesso em: 27/06/2009

[30] Codehaus. “STOMP”. Disponível em: http://stomp.codehaus.org/Home. Último acesso em: 27/06/2009