Universidade de Aveiro
2012
Departamento de Electrónica, Telecomunicações e Informática

Pedro Miguel Gonçalves Ferreira

DADOS ABERTOS @ UA

Universidade de Aveiro
2012
Departamento de Electrónica, Telecomunicações e Informática

Pedro Miguel Gonçalves Ferreira

DADOS ABERTOS @ UA

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica do Prof. Doutor Cláudio Teixeira, Professor do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro e do Prof. Doutor Diogo Gomes, Professor do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

Dedico este trabalho aos meus pais, Carlos e Filomena, à minha irmã, Ana Rosa, e à Raquel, pelo incansável apoio.

o júri

presidente

Doutor Osvaldo Manuel da Rocha Pacheco
Professor Auxiliar da Universidade de Aveiro

Doutor Alfredo Miguel Melo Matos
Investigador Sénior na Caixa Mágica S.A.

Doutor Cláudio Jorge Vieira Teixeira
Professor Auxiliar Convidado da Universidade de Aveiro

Doutor Diogo Nuno Pereira Gomes
Professor Auxiliar Convidado da Universidade de Aveiro

agradecimentos

Em primeiro lugar gostaria de agradecer aos meus orientadores, o Professor Doutor Cláudio Teixeira e Professor Doutor Diogo Gomes, por todos os conselhos e partilha de conhecimentos que me transmitiram durante a realização deste trabalho.

Agradeço ao Francisco Gouveia pela sua disponibilidade, sugestões e colaboração, indispensáveis para ultrapassar algumas dificuldades encontradas.

Agradeço aos meus pais pela possibilidade de realização deste trabalho. O seu incentivo e apoio incondicional são a base do meu sucesso académico e pessoal.

Por fim deixo o meu agradecimento à minha namorada Raquel, por toda a paciência e compreensão.

palavras-chave

Open data, Serviços web, SOAP, REST, Portal web, Web services security, Identidade, OAuth

resumo

Hoje em dia vivemos numa sociedade de informação. Qualquer pessoa ou entidade pode aceder, através de inúmeros meios de comunicação, a fontes de informação. Com a evolução tecnológica, a Internet transformou-se num dos principais meios de divulgação de conhecimento, permitindo o acesso a inúmeros recursos. Acompanhando este progresso surgiram novos termos e conceitos referentes à divulgação de informação, sendo o *open data* um dos principais.

O conceito *open data* aplica-se a dados disponibilizados publicamente por organizações e instituições. Este movimento está cada vez mais presente em instituições governamentais, que publicam os seus conjuntos de dados e bases de dados em portais web. Os benefícios do cruzamento de dados e implementação de *mashups*, assegurando a criação de novo conhecimento, são bastante evidentes, tanto a nível social como a nível económico.

A divulgação de dados através da Internet deverá obedecer a normas e padrões da indústria. Os serviços web assumem-se neste contexto como um método importante de transmissão de dados e informação devido, sobretudo, à sua vasta utilização por diversas aplicações e entidades.

Paralelamente à divulgação de informação pública, existem fontes de informação cujo seu conteúdo é privado. Neste contexto o protocolo OAuth surge como uma solução tecnológica que permite a divulgação de informação privada após a devida autorização pelo detentor dos conteúdos.

O objectivo desta dissertação passa por aplicar os conceitos acima referidos na Universidade de Aveiro. Numa primeira fase foi realizada uma triagem e categorização dos serviços web já existentes, assim como das fontes de informação com potencialidade para o desenvolvimento de um serviço. Após a implementação dos serviços identificados, foi desenvolvido um portal web agregador dos múltiplos serviços da Universidade de Aveiro, PT Inovação e SAPO.

Por fim foi implementado um servidor OAuth com o intuito de providenciar um mecanismo de autorização para o acesso a serviços web cuja informação é considerada sensível ou privada.

keywords

Open data, Webservices, SOAP, REST, Web portal, Web services security, Identity, OAuth

abstract

Nowadays we live in an information society. Every person or entity is able to access, through endless communication methods, to information sources. As technology evolved, Internet has become one of the most important ways of spreading knowledge, allowing access to several resources. As this progress continued, some new concepts about information sharing started to arise, being open data one of the most important.

The open data concept refers to data made publicly available by organizations and institutions. This concept is gaining importance on governmental parties, which publish their datasets and databases in web portals. Crossing data and implementation of *mashups*, providing new knowledge, comes with great social and economic benefits.

Spreading data through the Internet must follow industry standards and protocols. Webservices can be faced, in this context, as an important method of data and information transmission, essentially because of its high utilization.

Simultaneously there are some information sources that carry sensitive and private data. In this context, OAuth protocol stands as a technologic solution to share private contents.

The objective of this thesis is to implement the concepts described above in The University of Aveiro. On a first approach it was gathered information about existing webservices as well as potential information sources to create new webservices. After the implementation of the identified webservices, a web portal was developed to aggregate webservices from the University of Aveiro, PT Inovação and SAPO.

Finally an OAuth server was developed in order to provide an authorization mechanism to access private data webservices.

Índice

Lista de Figuras

Lista de Tabelas

Lista de Acrónimos

AJAX	Asynchronous JavaScript and XML
AMA	Agência para a Modernização Administrativa
API	Application Programming Interface
APIn	Academic Playground & Innovation
Atom	Atom Syndication Format
CC	Creative Commons
CC0	CC Zero
ccREL	Creative Commons Rights Expression Language
CORS	Cross-Origin Resource Sharing
CRUD	Create, Retrieve, Update and Delete
CSS	Cascading Style Sheets
CSV	Comma Separated Values
DETI	Departamento de Electrónica, Telecomunicações e Informática
DoS	Denial of Service
FAQ	Frequently Asked Questions
HMAC-SHA1	Hash-based Message Authentication Code with Secure Hash Algorithm 1
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
IdP	Identity Provider
IETF	Internet Engineering Task Force
ISBN	International Standard Book Number
ISSN	International Standard Serial Number
IT	Instituto de Telecomunicações
JAX-WS	Java API for XML Web Services
JS	JavaScript
JSON	JavaScript Object Notation
JSONP	JSON with Padding
JWT	JSON Web Token
MD5	Message-Digest Algorithm
MIME	Multipurpose Internet Mail Extensions
OASIS	Advancing Open Standards for the Information Society

OAuth	Open Authorization
ODC	Open Data Commons
ODC-By	Open Data Commons Attribution Licence
ODC-ODbL	Open Data Commons Open Database Licence
OGL	Open Government Licence
PACO	Portal Académico Online
PAP	Policy Administration Point
PDDL	Public Domain Dedication Licence
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PIP	Policy Information Point
PTin	PT Inovação
RDF	Resource Description Framework
REST	Representational State Transfer
RFC	Request For Comments
RFID	Radio-Frequency Identification
RP	Relying Party
RSS	Really Simple Syndication
SAC	Serviços Académicos
SAML	Security Assertion Markup Language
SAPO	Servidor de Apontadores Portugueses Online
SASUA	Serviços de Acção Social da Universidade de Aveiro
SGML	Standard Generalized Markup Language
SIC	Sistemas de Informação Computorizados
SMTP	Simple Mail Transfer Protocol
SOAP	Simple Object Access Protocol
SOP	Same-Origin Policy
SP	Service Provider
SSL	Secure Sockets Layer
SSO	Single Sign-On
STIC	Serviços de Tecnologias de Informação e Comunicação
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UA	Universidade de Aveiro

UDDI	Universal Description Discovery and Integration
UE	União Europeia
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UTF-8	Unicode Transformation Format-8
W3C	World Wide Web Consortium
WAYF	Where are you from?
WS-Security	Web Services Security
WSD	Web Service Description
WSDL	Web Service Definition Language
WSS	Web Services Security
WSS4J	Web Services Security for Java
WWW	World Wide Web
WYSIWYG	What You See Is What You Get
XACML	eXtensible Access Control Markup Language
XML	eXtensible Markup Language
XML-RPC	eXtensible Markup Language-Remote Procedure Calling
XMPP	eXtensible Messaging and Presence Protocol

1 Introdução

Este capítulo começa por enquadrar o trabalho realizado. De seguida é apresentada a motivação para a sua realização e enumerados os objectivos propostos. Por fim é resumida a estrutura da dissertação.

1.1 Contextualização

A informação é parte integrante dos alicerces da sociedade do século XXI. Vivemos numa época em que a informação nos chega das mais variadas formas e formatos: rádio, televisão, jornais e, sobretudo, através da Internet. No entanto, o que de facto é a informação? O termo informação é aplicado a um conjunto de dados que sofreram um processo de transformação. Esses dados foram organizados, manipulados e seleccionados com o objectivo de produzir um conjunto de premissas úteis e de valor acrescentado. Essas premissas devem conter certas características que influenciam o seu valor, das quais se podem destacar a sua exactidão, relevância, totalidade e fidedignidade [Stair 2012]. O valor da informação para uma instituição é crucial, tanto a nível económico como competitivo. É imperativo que uma dada entidade disponha de informação correcta e actual por forma a que as suas decisões estratégicas sejam o mais acertadas possível.

Com o avançar da tecnologia, os computadores vieram revolucionar o modo como a informação é gerada, distribuída e consumida. Os Sistemas de Informação Computorizados (SIC) são parte integrante das empresas de hoje em dia. Estes sistemas permitem obter dados de diversas fontes, processá-los e gerar um conjunto de informação relevante para a área de negócio da instituição.

Numa instituição de dimensões consideráveis, tal como uma Universidade, que está dividida em diversos departamentos e serviços, é comum haver um certo isolamento. Cada serviço tem as suas próprias fontes de dados, internas e externas, sendo a informação processada e consumida internamente e em virtude das necessidades. Este método proporciona que diversas fontes de informação departamentais estejam circunscritas, impedindo a partilha de informação com potenciais entidades, incluindo entidades da própria instituição.

Recentemente surgiu uma iniciativa denominada *Open Data*, introduzida pela Administração Obama nos Estado Unidos da América. Esta iniciativa tem como principal premissa a de que os dados recolhidos pelo governo são de interesse público e devem ser disponibilizados de um modo faseado [Obama 2010], permitindo assim que qualquer cidadão tenha acesso a esses dados.

Da mesma forma que a iniciativa *open data* é aplicada a um Estado, esta também pode ser aplicada a qualquer instituição de cariz público. No entanto, surgem certas questões com bastante relevância:

- De que forma devem ser organizadas, estruturadas e publicitadas as fontes de informação?
- Que dados e informação são, de facto, considerados públicos?

- A informação pessoal deve ser disponibilizada? Se sim, de que forma e através de que métodos?
- Que mecanismos de segurança devem estar envolvidos na partilha de dados e informação?

A segurança e o controlo de acesso a fontes de dados e informação é uma problemática de extrema importância, principalmente quando esses dados são de natureza pessoal. Devido a este facto, muitas empresas colocam a decisão e controlo de partilha desses dados nas mãos do utilizador.

Numa sociedade de informação como a que presenciamos actualmente, os métodos de manuseamento de dados e, principalmente, de informação não devem ser tratados levianamente. É necessário criar mecanismos e técnicas de partilha de informação sensível de modo a salvaguardar a privacidade dos intervenientes. Este paradigma acentua-se cada vez mais com a evolução tecnológica, dificultando a distinção da informação pública da privada.

1.2 Motivação

Na Universidade de Aveiro (UA) é possível identificar facilmente um conjunto de fontes de informação de carácter público: as senhas dos Serviços Académicos (SAC), as ementas das cantinas e restaurantes universitários, diversas *feeds* de notícias (Jornal Online) e de informações académicas e institucionais (Guia de Acesso), entre outras. Estas fontes de informação estão divididas por serviços e/ou departamentos, não havendo um ponto comum entre as mesmas.

Cada uma das fontes de informação identificadas anteriormente serve apenas, e só, o propósito para as quais foram desenvolvidas. Por exemplo, a informação das senhas SAC apenas está disponível através do ecrã presente nos SAC ou, mais recentemente, através do Portal Académico Online (PACO - `http://paco.ua.pt/`). Não está construída nem definida nenhuma *Application Programming Interface* (API) que permita a terceiros desenvolver aplicações que usufruam desta fonte de informação. O mesmo acontece para as ementas das cantinas e restaurantes universitários. É, portanto, facilmente detectável uma lacuna referente à partilha de informação de um modo transparente e padrão, tanto a nível interno da própria instituição como para entidades exteriores.

No âmbito da necessidade identificada anteriormente, surge a oportunidade de aperfeiçoar os métodos de distribuição de informação existentes na UA. A criação destes novos *endpoints* proporcionará uma melhor organização e estruturação das fontes de informação da UA. A estruturação de algumas fontes de informação numa instituição com as proporções da UA exige um estudo cuidado, tanto das tecnologias envolvidas, como da arquitectura a ser implementada ou das licenças sob as quais os dados podem ser divulgados.

No entanto, nem todas as fontes de informação presentes na UA são de carácter público. Existe informação de natureza pessoal e institucional que deve ser disponibilizada de uma forma controlada e autorizada. É, portanto, pertinente estudar a melhor arquitectura a ser implementada de modo a que a informação pessoal possa ser divulgada a terceiros, salvaguardando sempre a livre vontade do proprietário em autorizar e, posteriormente, cancelar a divulgação dos seus dados pessoais.

Com a criação de novos *endpoints* de serviços web há igualmente oportunidade para a criação de um portal web agregador de todos os *endpoints*, públicos e privados, referentes à UA.

1.3 Objectivos

Com esta dissertação pretende-se identificar e caracterizar as diferentes fontes de informação da UA e, caso seja necessário, construir APIs ou serviços web de acesso à informação. Estas fontes de informação devem ser devidamente categorizadas em função do tipo de informação que representam.

De modo a disponibilizar os *endpoints* já existentes e os que irão ser criados, será também necessário a implementação de um portal web agregador. Devido ao enquadramento desta dissertação no projecto de investigação *Academic Playground & Innovation* (APIn), o portal web contará não só com serviços da UA, como também da PT Inovação (PTin) e do Servidor de Apontadores Portugueses Online (SAPO). Devido à vertente académica do projecto APIn é bastante importante a disponibilização de uma *wiki* para cada serviço que contenha todas as informações necessárias para o consumo do serviço em causa: documentação, exemplos, descrição concisa das possíveis mensagens de erro assim como referências aos métodos de autenticação e/ou autorização, caso sejam aplicáveis.

Para além da disponibilização dos serviços no portal, serão disponibilizados servidores virtuais para a criação de projectos, um local para a divulgação de aplicações desenvolvidas no âmbito do APIn e uma página com *Frequently Asked Questions* (FAQ) para a ajuda nas diversas interacções com o portal.

Como já foi referido anteriormente, as fontes de informação presentes na UA estão repartidas quanto ao seu carácter: público ou privado. Esta condição implica colocar um nível de autorização e autenticação de forma a que o acesso à informação privada seja o mais controlado possível. A solução encontrada passa pela implementação do protocolo *Open Authorization* (OAuth) como *middleware* no acesso a recursos considerados sensíveis [RFC 5849]. Existem algumas variáveis inerentes ao protocolo OAuth que necessitam de ser controladas. Por exemplo, deve ser dada a possibilidade de um utilizador revogar autorizações concedidas a uma aplicação externa para o acesso aos seus dados pessoais e institucionais. Assim como deve ser possível a um administrador gerar chaves de acesso para uma nova aplicação que necessite de utilizar o protocolo OAuth. Ou suspender a utilização deste serviço a uma aplicação com comportamentos abusivos. Surge, então, a necessidade de implementação de um portal web que permita a gestão do protocolo OAuth e de todos os parâmetros inerentes.

1.4 Organização da Dissertação

No sentido de estruturar esta dissertação num documento coerente, foram redigidos seis capítulos, sendo o primeiro dedicado à contextualização da problemática, motivação e objectivos do trabalho desenvolvido. O segundo capítulo é dedicado ao estudo do estado da arte, onde será discutida uma abordagem às tecnologias e arquitecturas utilizadas na elaboração desta dissertação. Irá ser explorado o conceito *Open Data* e o impulsionar do mesmo para que instituições de cariz público disponibilizem as suas fontes

de informação. Veremos de que forma as instituições, tanto nacionais como internacionais, começam a ceder os seus dados e sob que condições. Posteriormente, será feita uma análise das arquitecturas utilizadas para a disponibilização de informação através de serviços web: a arquitectura *Representational State Transfer* (REST) e a arquitectura *Simple Object Access Protocol* (SOAP). São igualmente analisadas algumas arquitecturas e conceitos no que respeita ao desenvolvimento de portais web, assim como o modelo Web 2.0. São também apresentadas soluções de autenticação e autorização web. Será também feito um breve estudo da disseminação do protocolo OAuth em aplicações web, *desktop* e móveis, assim como uma análise do impacto deste na forma como a identidade do utilizador é gerida e partilhada por diferentes domínios.

O terceiro capítulo é iniciado com uma caracterização dos serviços existentes actualmente na UA, assim como a identificação das potencias fontes de informação para as quais ainda não está disponibilizada uma API. Depois de identificadas as fontes de informação que carecem de um modelo programático de acesso computorizado, será exposto o método de implementação e criação de novos serviços. Em alguns serviços, nomeadamente nas senhas dos SAC, foi necessário uma prévia implementação de um *middleware*, devido à forma como a aplicação que gere as senhas nos SAC disponibiliza os dados. Será também analisada a forma como o serviço das ementas da UA está implementado e quais os mecanismos de *cache* que são utilizados com o objectivo de evitar invocações desnecessárias ao serviço. Um outro serviço implementado denomina-se de *File Bucket*. Este serviço permite o armazenamento e partilha de ficheiros de pequenas dimensões, garantindo que o detentor do ficheiro é uma pessoa com conta na UA. Por fim serão discutidas algumas fontes de informação que poderão originar novos serviços.

Devido à confidencialidade de algumas informações disponibilizadas por certos serviços, é necessária a adopção de mecanismos de autenticação e autorização. No quarto capítulo são exploradas estas duas componentes. A solução implementada para o acesso a recursos privados foi o protocolo OAuth. Neste capítulo é realizado um estudo sobre este protocolo, a sua arquitectura e o modo utilizado para agilizar o processo de acesso à informação pessoal e institucional do utilizador. De forma a controlar todas as variáveis envolvidas num servidor OAuth, foi desenvolvido um portal para a gestão deste protocolo, tanto a nível administrativo como pessoal. Posteriormente, é detalhada a implementação do padrão *Web Services Security* (WSS). No âmbito do projecto *myPersonas*, da PTin, foi necessário a aplicação de uma camada de segurança nos serviços web. O WSS é uma extensão do protocolo SOAP que permite a transacção, de uma forma segura e confidencial, de mensagens assinadas através deste protocolo [IBM 2002]. Será estudada a forma como o WSS pode ser implementado com recurso à linguagem de programação JAVA.

No quinto capítulo é abordado o projecto APIn**.** São descritas todas a tecnologias utilizadas no desenvolvimento do portal APIn, assim como a melhor forma de interligar os diversos recursos necessários à elaboração de um portal agregador de serviços web. Para a criação deste portal, e devido à natureza sensível sobre o acesso a alguns serviços, foi necessária a implementação de um sistema de autenticação. O sistema de autenticação escolhido foi o *Identity Provider* (IdP) da UA.

No sexto e último capítulo desta dissertação são discutidas as conclusões e o possível trabalho a ser

desenvolvido no âmbito deste projecto. É realizado um balanço do trabalho implementado e o impacto que teve numa melhor estruturação e disseminação das fontes de informação na UA. Posteriormente são expostas algumas directrizes para uma eventual continuação do trabalho desenvolvido, tanto a nível de identificação de novos serviços da UA, como a nível de desenvolvimento do portal APIn.

2 Estado da Arte

Neste capítulo é realizada uma introdução e análise de alguns conceitos relacionados com a temática do *open data*. São estudadas as diferentes licenças existentes e como estas podem suportar legalmente a livre distribuição de dados e conjuntos de dados (*datasets*). Embora instituições definam os seus dados como públicos e de livre acesso, são necessárias ferramentas para a distribuição, através de padrões abertos e bem conhecidos, dos dados. Assim como o acesso a dados públicos é realizado de uma forma livre, sem que seja necessário nenhum controlo específico, o acesso a dados privados deverá ser realizado sob uma vigilância mais rigorosa. Para o acesso a um recurso privado, o detentor deste deverá sempre controlar a partilha, retirando permissões de acesso se achar pertinente. Este tipo de controlo é definido através do protocolo OAuth.

De modo a que um sistema tenha a capacidade de garantir o acesso a dados por parte de uma entidade, esta necessita de estar informada sobre a entidade que pretende aceder aos recursos. Estas informações podem ser partilhadas entre várias entidades na Internet, recorrendo a sistemas de *Single Sign-On* (SSO).

2.1 Open Data

Introduzido nos finais da década de 2000, o conceito *open data* generaliza a acessibilidade da informação que pode ser pública e facilmente alcançável por qualquer pessoa [Hoxha 2011]. Esta secção é reservada ao estudo da importância da abertura de dados para utilização pública. Com a disponibilização de dados de uma forma aberta, é prudente ter associada uma licença que defina concretamente os termos de utilização dos mesmos, assim como as regras que devem ser cumpridas na reutilização destes. Tendo em conta que os dados são disponibilizados digitalmente, é necessário especificar o formato em que estes podem ser consumidos. Com a disseminação do conceito *open data* e dos seus benefícios (principalmente económicos), vários governos, administrações locais e entidades públicas começaram a disponibilizar, através da Internet, os *datasets* ou bases de dados dos seus serviços.

2.1.1 O que é Open Data

O tema principal desta dissertação é *open data*. Mas o que é, na realidade, *open data*? De um ponto de vista técnico, o *open data* "refere-se a *datasets* que podem ser reutilizados sem quaisquer restrições por parte de licenças ou patentes, dados esses que devem ser bem estruturados, facilmente acessíveis e reutilizáveis por instituições, cientistas ou pela comunidade web" [Hoxha 2011]. Segundo o *Open Definition*, o termo *open data* aplica-se a "dados que podem ser usados, reutilizados e redistribuídos livremente por qualquer pessoa, estando sujeitos, no máximo, à identificação da fonte e conservação dos direitos de autor" [Dietrich 2010]. Esta definição pode ser detalhada, caracterizando especificamente os conceitos que devem ser inerentes aos dados distribuídos como *open data*. Um desses conceitos é o **acesso e**

disponibilidade dos dados. Estes devem ser disponibilizados como um todo e, preferencialmente, através da Internet sem custos associados. Os dados também devem ser disponibilizados de uma forma conveniente e facilmente personalizável, contribuindo para outro conceito do *open data*: a possibilidade de **reutilização e redistribuição** da informação. Este conceito determina que os dados devem ser acessíveis sob termos que permitam a reutilização e redistribuição dos mesmos, assim como o cruzamento com dados de fontes distintas. A licença de reutilização e redistribuição dos dados não deve requerer quaisquer taxas ou custos. Com a abertura dos dados publicamente, a **participação universal** é outro dos conceitos inerentes ao *open data*: todos devem ter a possibilidade de utilizar, reutilizar e redistribuir. Não deve existir discriminação contra pessoas, grupos ou ramos de actividade. Por exemplo, restrições de "não-comercialização" que poderiam impedir o uso dos dados para fins comerciais, ou restrições contra determinadas áreas de aplicação (educação, saúde, banca, etc).

As características acima descritas são fundamentais para tornar viável o cruzamento de dados, conferindo interoperabilidade ao sistema em causa. A interoperabilidade de um sistema é definida como a capacidade de trocar, reutilizar e interpretar dados por diferentes arquitecturas, através de padrões abertos [Kaplan 2005]. A capacidade de poder agregar componentes diferentes é essencial para a construção de sistemas complexos e, sobretudo, escaláveis. A interoperabilidade de um sistema ou organização permite que as suas fontes de dados possam ser cruzadas através de *mashups*, permitindo assim a transformação de dados em informação de valor acrescentado.

O conceito *open data* tem sido bastante impulsionado por parte dos governos de diversos países, tendo inclusive sido bastante apoiado pela União Europeia (UE). Este facto deve-se essencialmente a várias oportunidades que a abertura de dados pode vir a proporcionar [Comission 2011]. Do ponto de vista económico, as **oportunidade de negócio** surgem como um factor determinante para a implementação de *mashups* inovadoras que originem novos serviços e/ou produtos informativos. Como consequência, surge a possibilidade de criação de novos postos de trabalho e riqueza. Do ponto de vista informativo, a abertura de dados governamentais proporciona uma **maior transparência** na administração pública, impulsionando a visibilidade de informação que antes era privada. Ao permitir um melhor conhecimento por parte de pessoas e empresas sobre políticas aplicadas, financiamento público e despesas, a partilha deste tipo de informação governamental poderá incitar à criação de novas empresas e investimentos com menos riscos associados. Do ponto de vista governamental, o *open data* incentiva a **melhores decisões políticas baseadas em dados reais e eficiência administrativa**. A disponibilização destes dados poderá resultar em melhores serviços públicos para o cidadão, aplicados directamente a focos de necessidades.

O factor económico é de extrema importância para a disponibilização de dados. Um estudo recente indica que o mercado total para o sector informativo na UE foi de 28 mil milhões de Euros em 2008. O mesmo estudo também realça que os ganhos económicos, directos e indirectos, da reutilização de informação pública na UE será de 140 milhões de Euros anuais [Vickery 2011]. Embora os dados disponibilizados de forma pública e aberta não representem, em primeira instância, qualquer valor, o cruzamento destes poderá gerar informação com valor económico.

Surge, no entanto, a necessidade de definir formatos e licenças de partilha de dados. Quais os melhores métodos para uma partilha padronizada da informação? E que tipo de regulamentações jurídicas são aplicadas a estes dados? Nas próximas secções serão discutidas algumas tecnologias de partilha e licenças a aplicar a conjuntos de dados e bases de dados.

2.1.2 Licenças

Embora a disponibilização de dados seja impulsionadora de uma maior transparência, interoperabilidade e desenvolvimento académico, social e económico, a não especificação de termos de utilização dos dados poderá ter um efeito contra-produtivo. A forma como diferentes jurisdições tratam a utilização de dados em diferentes contextos é, por vezes, de difícil compreensão. Este aspecto poderá dificultar a partilha de dados entre organizações, sendo assim criada uma barreira legal à utilização e redistribuição de dados [Ball 2011]. Desta forma é essencial aplicar uma licença para o uso e redistribuição dos dados.

A licença a ser aplicada deverá ser *standard*: além de melhorar a eficiência organizacional e proporcionar a poupança de custos, o uso de licenças padrão permite aumentar a interoperabilidade dos dados facultados. A utilização de licenças bem conhecidas é igualmente benéfica para as entidades consumidoras dos dados, devido a uma melhor compreensão das regras que são aplicadas [Korn 2011].

2.1.2.1 *Creative Commons*

A *Creative Commons* (CC) é uma organização sem fins lucrativos fundada em 2001. O objectivo desta organização é o de produzir licenças *standard* simples, mas robustas, que permitam aplicar direitos de autor aos trabalhos de uma entidade [Korn 2011, Ball 2011].

Figura 2.1: Camadas das licenças CC. Fonte: [Commons 2003]

As licenças da CC estão divididas em três camadas [Commons 2003]: o **código legal**, que proporciona uma camada formal e técnica da licença, o ***Commons Deed***, que é definida como a camada não formal da licença, responsável por sumarizar os termos e condições mais importantes através de uma lingua-

gem leiga e, na última camada, está presente a **Linguagem Máquina**, que sumariza os pontos cruciais da licença, sendo estes disponibilizados num formato que permita a sistemas de software, motores de pesquisa e outros tipos de tecnologias a sua compreensão.

Estão disponíveis seis licenças CC. Todas as licenças tem por base uma única licença: *attribuition*. Isto significa que as propriedades da licença *attribuition* são herdadas pelas restantes cinco licenças.

Attribution `CC BY` Esta licença permite que terceiros distribuam, copiem, exponham e alterem o trabalho desde que o autor seja devidamente referenciado. Também prevê a utilização do trabalho para fins comerciais. O autor deve especificar a forma como deve ser referenciado. Esta licença é comum nas restantes licenças do CC.

Attribution-ShareAlike `CC BY-SA` Todos os trabalhos baseados no trabalho que detém esta licença devem manter a licença. Ou seja, qualquer trabalho que tenha por base um trabalho com esta licença deverá referenciar o seu autor original, e assim sucessivamente.

Attribution-NoDerivs `CC BY-ND` Licença que permite a redistribuição, comercial ou não comercial, desde que não sejam feitas alterações no trabalho base.

Attribution-NonCommercial `CC BY-NC` Esta licença determina que os trabalhos derivados deste apenas devem ser utilizados para fins não comerciais.

Attribution-NonCommercial-ShareAlike `CC BY-NC-SA` Licença que determina que os trabalhos resultantes apenas devem ser utilizados para fins não comerciais. Além disso, esta licença deve ser mantida nos trabalhos resultantes.

Attribution-NonCommercial-NoDerivs `CC BY-NC-ND` À semelhança da licença `CC BY-NC`, esta licença especifica que os trabalhos derivados devem ser usados apenas para fins não comerciais. Também não devem ser feitas alterações no trabalho original.

Licença	Nomenclatura	Logótipo
Attribution	`CC BY`	
Attribution-ShareAlike	`CC BY-SA`	
Attribution-NoDerivs	`CC BY-ND`	
Atributivo-NonCommercial	`CC BY-NC`	
Attribution-NonCommercial-ShareAlike	`CC BY-NC-SA`	
Attribution-NonCommercial-NoDerivs	`CC BY-NC-ND`	

Tabela 2.1: Lista de licenças CC

Embora as licenças CC sejam bastante usadas hoje em dia, estas não são indicadas para o licenciamento de dados, *datasets* e bases de dados. Uma base de dados representa um ou vários *datasets*,

muitas das vezes provenientes de fontes ou autores distintos. Como já foi analisado anteriormente, todas as licenças da CC baseiam-se na licença de atribuição: trabalhos derivados devem conter referência ao autor original. Visto que um conjunto de dados (ou base de dados) pode ter inúmeros autores e contribuidores, surge um problema denominado *attribution stacking*: todos os autores devem ser referenciados [Vollmer 2011].

2.1.2.2 CC Zero

A licença *CC Zero* (CC0) é uma ferramenta desenvolvida pela CC com o objectivo de facilitar a disponibilização de conteúdos, dados, *datasets* e bases de dados no domínio público. Esta licença permite que o detentor dos dados mantenha todos os direitos de autor, assim como o direito de ser identificado como o respectivo autor. O detentor destes direitos mantém todos os direitos sobre a base de dados e deverá ser identificado como o seu criador. Quando tal não é possível, a licença CC0 dispõe de mecanismos para que o detentor dos direitos sobre os dados tenha a possibilidade de permitir o uso destes para qualquer fim, por qualquer entidade e sem ser aplicado nenhum condicionalismo ou taxa [Korn 2011].

Esta licença prevê colmatar o problema *attribution stacking* identificado nas licenças *Creative Commons* (2.1.2.1), tornando-se ideal para o universo *open data*.

Os dados, conjuntos de dados ou bases de dados que apliquem esta licença podem ser identificados através da figura 2.2.

Figura 2.2: *Creative Commons CC0 Mark*

2.1.2.3 Representação de Licenças CC

As licenças CC estão divididas em três níveis de representação: código legal, *common deads* e linguagem máquina. O último nível tem o objectivo de permitir a interpretação da licença por parte de aplicações de *software*. Para tal, a CC desenvolveu uma linguagem própria para a descrição das suas licenças: a *Creative Commons Rights Expression Language* (ccREL). Descrita pela primeira vez em Março de 2008, esta linguagem é baseada no formato *Resource Description Framework* (RDF), devido sobretudo à interoperabilidade e universalidade associados a este formato.

O RDF é uma *framework* utilizada para descrever entidades na web. Este formato assenta sobre uma estrutura de descrição atómica denominada de "*triples*". Cada "*triple*" consiste num sujeito, uma propriedade e um valor para a propriedade do sujeito. Por exemplo, consideremos que se pretende aplicar a licença CC BY ao *website* `http://api.web.ua.pt/`.

Figura 2.3: Exemplo de um *triple* da *framework* RDF

O "*triple*" resultante está representado na figura 2.3. Ao *website* `http://api.web.ua.pt/` (sujeito) é aplicada a licença CC BY (valor) através da propriedade *vocab#licence*. Esta estrutura de ligações forma um grafo, onde os vértices representam uma ligação nomeada entre dois recursos, representados pelos nós do grafo. Ou seja, um modelo RDF é uma colecção de "*triples*" que podem ser visualizados através de grafos.

A conversão do grafo da figura 2.3 para um modo textual é realizado através da linguagem de marcação *eXtensible Markup Language* (XML). Por exemplo, o "*triple*" representado anteriormente pode ser descrito da seguinte forma:

```
1 <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
2        xmlns:xhtml="http://www.w3.org/1999/xhtml/vocab#">
3   <rdf:Description rdf:about="http://api.web.ua.pt/">
4     <xhtml:license rdf:resource="http://creativecommons.org/licenses/by/3.0/" />
5   </rdf:Description>
6 </rdf:RDF>
```

Como é possível reparar, todos os identificadores são representados por *Uniform Resource Locator* (URLs). Isto permite obter uma descrição detalhada sobre a propriedade ao aceder ao URL, possibilitando também a formulação de propriedades por qualquer entidade. Neste seguimento, a CC desenvolveu duas propriedades para representar as suas licenças: a propriedade ***work***, que permite descrever aspectos específicos de um trabalho, incluindo sob que licenças este é distribuído, e a propriedade ***licence***, que permite descrever os aspectos da licença.

Nas propriedades ***work***, a CC recomenda a utilização de pelo menos quatro propriedades:

- `dc:title`: representa o título do documento (`dc` é a abreviatura para o vocabulário *Dublin Core*, disponível no endereço `http://purl.org/dc/elements/1.1/`).
- `cc:attributionName`: descreve o nome a citar quando o trabalho é redistribuído ou modificado (`cc` é a abreviatura para o endereço `http://creativecommons.org/ns#`).
- `cc:attributionURL`: o endereço de ligação a ser citado.
- `dc:type`: define o tipo do recurso a licenciar.

As propriedades ***licence*** podem ser divididas em seis categorias: a `cc:permits`, que define as permissões a aplicar ao trabalho, para além das que são aplicadas por defeito pela licença, a `cc:prohibits`, que permite revogar permissões, a `cc:requires`, que permite requerer algumas acções por parte do utilizador quando aplica as permissões definidas em `cc:permits`, a `cc:jurisdiction`, que associa a licença a uma jurisdição particular, a `cc:deprecatedOn`, que indica a data na qual a licença deixou de ser válida, e a `cc:legalCode`, que referencia o texto legal correspondente à licença. A utilização destas propriedades implica a criação de novas licenças por parte de quem as aplica, mas sempre baseadas nas licenças CC [Abelson 2008].

2.1.2.4 Open Data Commons

O projecto *Open Data Commons* (ODC), financiado pela fundação *Open Knowledge*, foi iniciado em Março de 2008 e tem como objectivo providenciar soluções legais para *open data*. Ao contrário das licenças disponibilizadas pelo CC, que podem ser aplicadas a qualquer trabalho, as licenças do ODC estão vocacionadas especialmente para dados, *datasets* e bases de dados. O ODC dispõe de três licenças:

ODC Attribution Licence (ODC-By) Esta licença, compatível com a licença `CC BY`, prevê a livre cópia, distribuição, modificação, transformação e produção de trabalhos baseados numa base de dados ou conjunto de dados. Requer igualmente que o autor seja devidamente identificado. Esta licença deverá ser mantida para qualquer trabalho derivado.

ODC Open Database Licence (ODC-ODbL) Semelhante à licença anterior (ODC-By), esta licença prevê igualmente que qualquer trabalho derivado a herde. No caso de a base de dados ser redistribuída, é necessário manter uma versão da mesma sem ser aplicada qualquer restrição. Esta licença aplica-se apenas a dados, *datasets* e bases de dados e é comparável à licença `CC BY-SA`.

Public Domain Dedication Licence (PDDL) Esta licença mantém todas as propriedades definidas pela ODC-By. Além disso, prevê que a base de dados a que a licença é aplicada passe para o domínio público, mantendo os autores todos os direitos reservados. Esta licença é comparável à licença CC Zero (2.1.2.2).

Licença	Ideal para *open data*	Utilização Comercial	Modificação Recurso	Conservação Licença	Atribuição
CC:					
`BY`	✗	✓	✓	✗	✓
`BY-SA`	✗	✓	✓	✓	✓
`BY-NC`	✗	✗	✓	✗	✓
`BY-ND`	✗	✓	✗	✗	✓
`BY-NC-SA`	✗	✗	✓	✓	✓
`BY-NC-ND`	✗	✗	✗	✗	✓
`CC0`	✓	✓	✓	✗	✓
ODC:					
`By`	✓	✓	✓	✓	✓
`ODbL`	✓	✓	✓	✓	✓
`PDDL`	✓	✓	✗	✗	✓

Tabela 2.2: Comparação das diversas licenças

Na tabela 2.2 é exposta uma comparação das licenças descritas anteriormente. Excepto a licença CC0, todas as restantes licenças da CC não são indicadas para o licenciamento de *datasets* e/ou bases de dados, devido ao problema de *attribution stacking*. As licenças CC caracterizam-se por permitirem quase todo o tipo de combinações possíveis, desde a utilização para fins comerciais, modificação do recurso às quais se aplicam e conservação da licença em produtos derivados. Estas licenças estão bastante divulgadas no mundo digital e são aplicadas sobretudo a ficheiros audiovisuais, artigos e publicações.

As licenças do projecto ODC foram desenvolvidas especificamente para o licenciamento de dados, *datasets* ou bases de dados. Bastantes comuns entre si, estas licenças caracterizam-se pela permissão de utilização comercial dos recursos às quais são aplicadas. A licença PDDL destaca-se das restantes pela proibição de alteração das fontes do trabalho e por não exigir a manutenção da licença nos produtos derivados.

Em todas as licenças referidas, tanto provenientes da CC ou do ODC, é exigida a identificação do autor das fontes em que os produtos se baseiam.

Para além das licenças estudadas anteriormente, alguns governos desenvolveram as suas próprias clausulas a aplicar a dados *open data*. Por exemplo, os dados disponibilizados pelo governo do Reino Unido estão sob a licença *Open Government Licence* (OGL). À semelhança das licenças ODC e CC0, a OGL aplicasse a dados, *datasets*, bases de dados e também a conteúdos. Permite a utilização dos recursos para uso comercial e deve ser aplicada a *datasets* relativamente pequenos, de modo a limitar o *attibution stacking* [Korn 2011]. Os pormenores sobre esta licença podem ser consultados no endereço `http://www.nationalarchives.gov.uk/doc/open-government-licence/`.

2.1.3 Open Data em Portugal

2.1.3.1 Dados.gov

Em Junho de 2011 foi lançado em Portugal o portal Dados.gov (`http://www.dados.gov.pt/`). O portal Dados.gov tem como objectivo a divulgação de dados partilhados por instituições da Administração Pública, assim como reunir uma série de aplicações que utilizam esses dados. De entre os fornecedores de dados presentes no portal, é possível destacar-se a Agência para a Modernização Administrativa (AMA), a Comissão Nacional de Eleições e a Fundação para a Computação Científica Nacional, que contam com o maior número de dados disponibilizados. Este portal está sob a alçada da AMA.

2.1.3.2 Open Data Lx

O Open Data Lx (`http://www.lisboaparticipa.pt/pages/smartlx.php`) é uma iniciativa promovida pela Câmara Municipal de Lisboa e pela AMA, com o objectivo de disponibilizar colecções de dados sobre a cidade de Lisboa. À semelhança do portal Dados.gov, é solicitada a utilização por parte dos cidadãos dos *datasets* disponibilizados, assim como a implementação de aplicações *mashup*.

2.1.3.3 Academic Playground & Innovation

O portal APIn (`http://api.web.ua.pt/`) resultou de um projecto de investigação financiando pela PTin e SAPO, em parceria com a UA. Este portal visa reunir um conjunto de serviços web de carácter público de três fornecedores: PTin, SAPO e UA. Contando com 75 serviços, este portal público incentiva ao desenvolvimento de aplicações com base nos serviços publicitados. Devido a informações sensíveis disponibilizados por alguns dos serviços, estes são considerados privados e apenas estão disponíveis para os alunos e funcionários da UA. À semelhança dos portais Dados.gov (2.1.3.1) e Open Data Lx (2.1.3.2), também estão disponíveis algumas aplicações que utilizam serviços públicos.

2.1.4 Open Data na Europa

Os portais *open data* europeus começaram a surgir no início de 2010, com a lançamento do portal **data.gov.uk** (`http://data.gov.uk/`) pelo governo do Reino Unido. Neste portal estão publicados cerca de 5400 *datasets* originários de departamentos governamentais, sectores públicos e autoridades locais. Nos temas mais populares encontram-se a saúde, transparência e finanças. Este portal reúne dados de Inglaterra, Irlanda do Norte, Escócia e País de Gales. Os dados disponibilizados neste *website* estão protegidos pela licença OGL.

Pela mesma data surgiu o **London Datastore** (`http://data.london.gov.uk/`). Dedicado a conjuntos de dados da cidade de Londres, este portal disponibiliza inúmeros *datasets* e incentiva os cidadão a desenvolverem aplicações *mashup*. Em Abril de 2010 foi lançado o portal *open data* Austríaco. À semelhança dos portais já descritos, o **Data.gv.at** (`http://www.data.gv.at/`) conta com inúmeros *datasets*, acessíveis em vários formatos. Este portal apenas está disponível na língua Alemã. A partir desta data começaram a surgir portais *open data* nacionais, como o **Data.overheid.nl** (`http://data.overheid.nl/`) da Holanda, o **Dati.gov.it** (`http://www.dati.gov.it/`) de Itália, o **Data.gov.be** (`http://data.gov.be/`) da Bélgica e, mais recentemente, o **Data.gouv.fr** (`http://www.data.gouv.fr/`) de França, como também portais *open data* de cidades europeias, como o **Helsinki Region Infoshare** (`http://www.hri.fi/en/`), o **Open Data - Manchester City** (`http://www.manchester.gov.uk/opendata`) ou o **Paris Data** (`http://opendata.paris.fr/`).

Todos estes portais partilham o mesmo conceito: divulgação do seus *datasets* e bases de dados e incentivo à implementação de aplicações que cruzem estes dados. Citando o portal *London Datastore*, "a disponibilização dos dados é apenas metade da batalha. Queremos encorajar Londrinos talentosos a transformarem linhas de texto e números em aplicações, *websites* ou produtos móveis de utilidade pública".

2.1.5 Open Data no Mundo

Um pouco por todo o mundo foram surgindo portais governamentais de *open data*. Os grandes impulsionadores deste conceito, os Estados Unidos da América, foram os pioneiros no lançamento de um portal *open data*: o **Data.gov** (`www.data.gov`). Neste portal é possível encontrar centenas de milhares (mais de

445 mil) *datasets,* que variam entre a educação, saúde, energia e segurança. Lançado em Maio de 2009, este portal conta com mais de 1500 publicações governamentais ou desenvolvidas por cidadãos.

Após o lançamento deste portal, diversas cidades dos EUA começaram a lançar os seus próprios *website* de *open data*: o **San Francisco Data** (`https://data.sfgov.org/`, o **Washington Data Catalog** (`http://data.dc.gov/`), o **City of Chicago Data Portal** (`https://data.cityofchicago.org/`) ou o **NYC Open Data** (`https://nycopendata.socrata.com/`), da cidade de Nova Iorque.

Paralelamente forma surgindo portais *open data* por outros países. Em Novembro de 2009 foi lançado o portal **Data.govt.nz** (`http://www.data.govt.nz/`) da Nova Zelândia. De entre os *datasets* mais consultados, destacam-se as categorias geográfica, ambiental e desportiva. Em Maio de 2011, foi a vez do Canadá lançar o portal **Data.gc.ca** (`http://www.data.gc.ca/`). Contando com cerca de 13 mil *datasets*, este *website* reúne informações de diversos departamentos e agências estatais.

Na continente Asiático também foram surgindo alguns portais *open data*. Em Junho de 2011 foi lançado o portal **Data.gov.sg** (`http://data.gov.sg/`) do governo de Singapura, que disponibiliza mais de 5000 *datasets* provenientes de 50 ministérios e agências. Este portal tem como objectivos o acesso conveniente a dados governamentais por parte dos cidadãos, a criação de valor através do desenvolvimento de aplicações e facilitar a investigação baseada nos dados fornecidos. Na Arábia Saudita foi lançado o portal **SAUDI Open Data** (`http://saudi.gov.sa/`) em Setembro de 2011. Contando com inúmeros *datasets*, estes estão divididos por várias categorias, como condições atmosféricas, serviços sociais ou indústria. Embora este *website* disponibilize bastantes informações, maior parte destas são relativas a anos anteriores a 2010.

Em África também existem iniciativas *open data*. Por exemplo, o governo queniano lançou a 8 de Julho de 2011 o portal **Kenya Open Data** (`https://opendata.go.ke/`). Considerado o primeiro portal *open data* de uma país em vias de desenvolvimento, e o segundo do continente Africano, os primeiros *datasets* publicados foram referentes ao *census* de 2009 e exportações regionais e nacionais. O primeiro portal *open data* do continente Africano foi o **Data.gov.ma** (`http://data.gov.ma/`), do governo de Marrocos. Este portal conta com inúmeros *datasets*, principalmente sobre educação e finanças. Todos os dados contidos neste portal são disponibilizados ao abrigo da licença ODbL.

Além de países, também algumas instituições lançaram os seus portais *open data*. Em Março de 2010 o Google lançou o **Public Data Explorer** (`http://www.google.com/publicdata/directory`). Este portal permite explorar e visualizar *datasets* de interesse público, assim como o desenvolvimento de gráficos *mashup* de várias fontes de dados. Posteriormente, em Abril de 2010, o Banco Mundial inaugurou o portal **The World Bank Data** (`http://data.worldbank.org/`). Neste portal é possível aceder a dados do Banco Mundial, que se encontram categorizados por país, tópico e indicadores. Em Junho de 2010, a UE lançou o portal **PublicData.EU** (`http://publicdata.eu/`). Este portal partiu da iniciativa de agregar num único local *datasets* europeus que se encontram divididos por diversos *websites*.

Na figura 2.4 estão representadas as datas de lançamento dos portais *open data* descritos anteriormente. É possível constatar que a maioria deste *websites* foram lançados em 2011, o que prova a recente

divulgação do conceito *open data*. À medida que alguns regimes são substituídos e mentalidades alteradas, há lugar para a divulgação de dados de interesse público que podem permitir melhorar sistemas e processos governamentais ou institucionais.

2.1.6 Cronologia do Lançamento de Portais *Open Data*

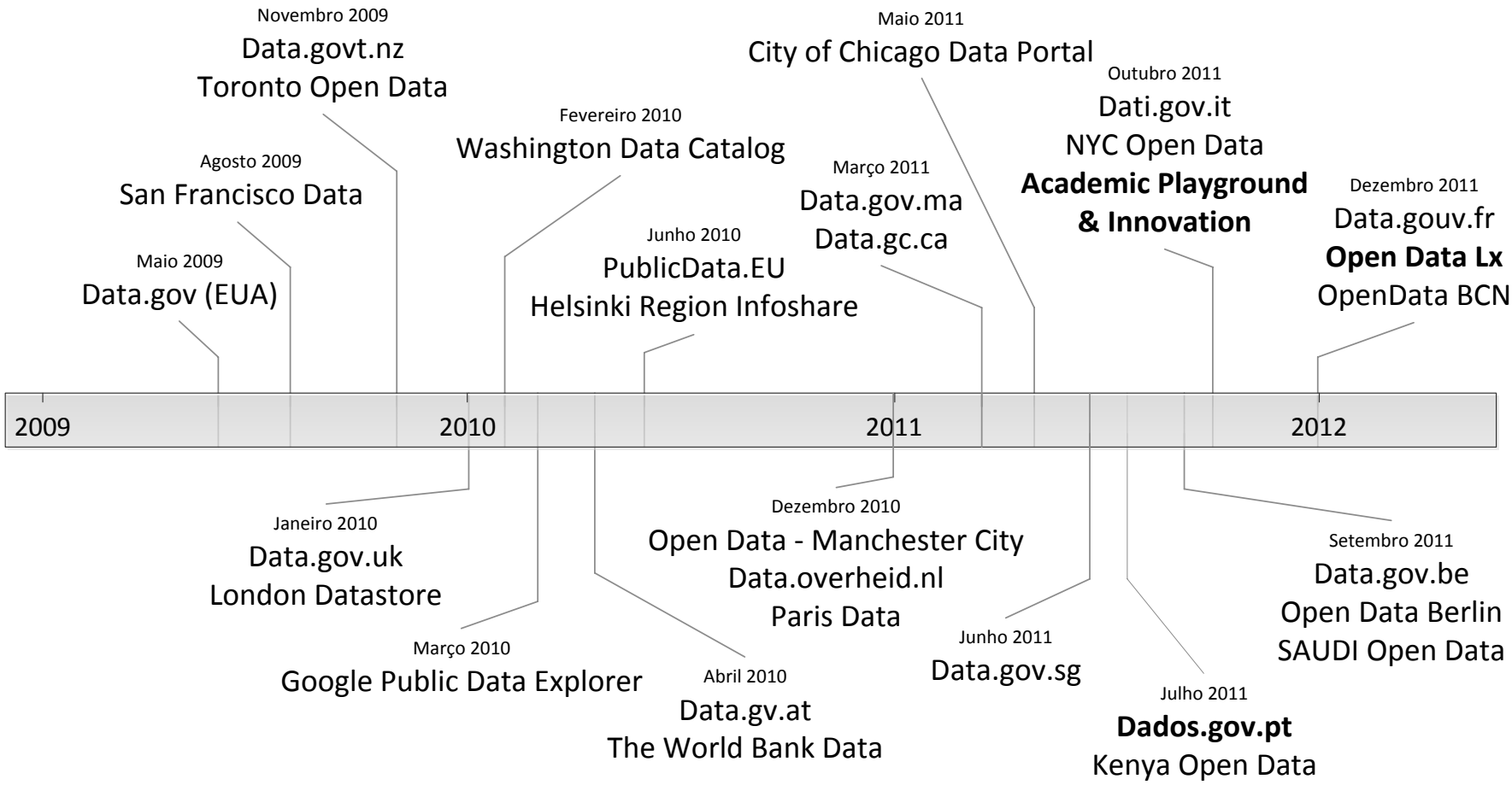


Figura 2.4: Cronologia do lançamento de portais *open data*

Nota: as datas de lançamento dos portais foram obtidas nos próprios portais ou respectivos *blogs*.

2.1.7 Representação de Dados

Neste secção é tratada a problemática da disseminação de dados. Com a disponibilização de dados e *datasets* através da Internet, é necessária a utilização de formatos padrão e livres para a sua transmissão. Hoje em dia o formato mais utilizado é o XML, seguido de perto do *JavaScript Object Notation* (JSON). Devido ao problema de acesso a dados presentes em domínios diferentes através de uma página web, surgiu a necessidade de implementação de um formato *JavaScript* (JS), para a representação de dados JSON, denominado de *JSON with Padding* (JSONP). Também será estudado o formato *Comma Separated Values* (CSV), utilizado para a transmissão de grandes quantidades de informação.

2.1.7.1 eXtensible Markup Language

A XML é uma linguagem de marcação desenvolvida com o intuito de transportar dados. Derivada da metalinguagem *Standard Generalized Markup Language* (SGML), a primeira versão (1.0) do XML foi especificado pelo World Wide Web Consortium (W3C) em 1996 e surgiu devido à complexidade do SGML. O consórcio W3C é uma organização internacional com o intuito de desenhar e desenvolver standards web, protocolos e guias que permitam que o World Wide Web (WWW) seja um canal de comunicação seguro, tanto para providenciar serviços como para criar conteúdos.

O XML é um formato aberto e foi desenvolvido com o intuito de ser uma linguagem simples, concisa, legível e de fácil utilização através da Internet [Bray 2008].

A informação que é estruturada através de um ficheiro XML pode conter vários tipos de dados, tais como texto e imagens. A estruturação da informação é essencial num ficheiro XML, pois especifica o contexto dos dados presentes e o seu papel no conjunto. É também importante especificar a codificação dos caracteres presentes neste tipo de ficheiros [RFC 3023].

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <filmes>
3    <filme>
4      <titulo>The Avengers</titulo>
5      <ano>2012</ano>
6      <duracao metrica="min">142</duracao>
7      <resumo>Nick Fury of S.H.I.E.L.D. brings together a team of super humans to form The
           Avengers to help save the Earth from Loki and his army.</resumo>
8      <actores>
9        <actor>Robert Downey Jr.</actor>
10       <actor>Scarlett Johansson</actor>
11       <actor>Jeremy Renner</actor>
12       <actor>Tom Hiddleston</actor>
13     </actores>
14     <categorias>
15       <accao />
16       <aventura />
17     </categorias>
18   </filme>
19 </filmes>
```

No exemplo anterior é possível identificar várias características inerentes a um ficheiro XML:

- **Codificação**: Na linha (01), o atributo `encoding=”utf-8”` especifica que o documento está codificado no formato *Unicode Transformation Format-8* (UTF-8).

- **Elementos**: Os elementos são cadeias de caracteres delimitadas pelos caracteres menor (<) e maior (>), e permitem identificar a natureza dos dados que contêm. Os elementos podem ser, ou não, vazios. Por exemplo, nas linhas (15) e (16) existem dois elementos vazios. Os elementos também são muitas vezes apelidados de *tags*.
- **Atributos**: Os atributos são pares nome-valor declarados após o nome do elemento. Normalmente são usados para indicar informação adicional ao elemento. Por exemplo, na linha (06), o elemento `duracao` contem o valor indicativo da duração do filme. O atributo `metrica` permite especificar as unidades da duração do filme (neste caso em minutos).
- **Estruturas de dados**: No exemplo anterior é possível identificar algumas estruturas de dados que permitem contextualizar a informação a ser transmitida. Por exemplo, a estrutura `filme` contém seis elementos filho que, no seu todo, representam um filme: `titulo`, `ano`, `duracao`, `resumo`, `actores` e `categorias`. A mesma analogia pode ser aplicada às estruturas de dados `actores` e `categorias`.

A elasticidade na representação de informação através de ficheiros XML leva a que muitas outras tecnologias web utilizem este meio para o transporte de informação. Por exemplo, o XML é utilizado para a representação de *webfeeds*, tanto através do formato *Atom Syndication Format* (Atom) como também do *Really Simple Syndication*[1] (RSS). É igualmente bastante utilizado em serviços web, tanto na arquitectura SOAP como na arquitectura REST. O protocolo de comunicações *eXtensible Messaging and Presence Protocol* (XMPP), bastante utilizado para serviços de *chat* (*GTalk*, *Facebook Chat*, etc), utiliza o XML como formato das mensagens.

O XML também é utilizado no protocolo *XML-Remote Procedure Calling* (XML-RPC). O XML-RPC define um formato XML para mensagens que são transmitidas entre clientes e servidores, através do protocolo *HyperText Transfer Protocol* (HTTP) [RFC 3529].

Vantagens do XML

- Flexibilidade, acessibilidade e portabilidade.
- O formato e linguagem de um ficheiro XML é definido pelo utilizador, proporcionando uma fácil adaptação e legibilidade.
- Possibilidade de representar relações complexas, como estruturas em árvore e herança.
- Suporte de *Unicode*, permitindo transmitir quase todos os caracteres em diversas codificações.
- Suporta *schemas* e *namespaces*.

Desvantagens do XML

- Necessidade de existir uma aplicação *middleware* que processe o ficheiro XML.

[1] Esta nomenclatura aplica-se à versão 2.0.

- Susceptibilidade a redundância.
- Linguagem muito expressiva: maior *overhead*.
- Inexistência de tipos de dados (*integer*, *boolean*, etc).

2.1.7.2 JavaScript Object Notation

À semelhança do XML, o formato JSON é baseado em texto e foi desenhado com o intuito de transportar e representar dados. Definido como um subconjunto da linguagem de programação JS, o JSON permite tanto uma leitura como escrita bastante simples. Embora tenha sido oferecida alguma resistência à adopção deste formato para a transmissão de dados, o JSON começa a rivalizar com o XML para a transmissão de informação, devido principalmente ao processamento e utilização mais simples e eficaz por parte de sistemas computacionais [Severance 2012]. O JSON assenta sobre duas estruturas de dados: uma colecção de pares nome-valor e uma lista ordenada de valores, que podem ser representadas através das seguintes formas [JSON 2006, RFC 4627]:

Objecto

Um objecto é um conjunto não ordenado nome-valor, que começa com o chaveta à esquerda (`{`) e termina cm chaveta à direita (`}`). Cada nome é procedido de dois pontos (`:`) e do valor. Cada par nome-valor é separado por vírgula (`,`).

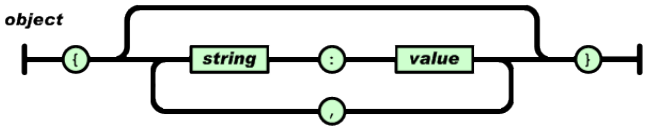


Figura 2.5: Objecto em JSON. Fonte: [JSON 2006]

Array

Um *array* representa uma colecção ordenada de valores. É delimitado por parêntesis recto à esquerda (`[`) e parêntesis recto à direita (`]`). Os valores de um *array* são separados por vírgula (`,`).

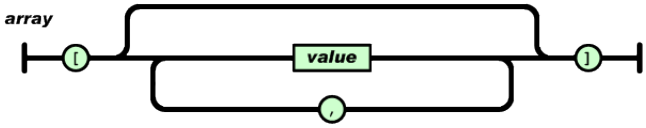


Figura 2.6: *Array* em JSON. Fonte: [JSON 2006]

Valor

Um valor pode representar uma *string* (delimitada por aspas (`""`)), um número, um booleano (`true` e `false`), um objecto, um *array* ou um objecto nulo (`null`).

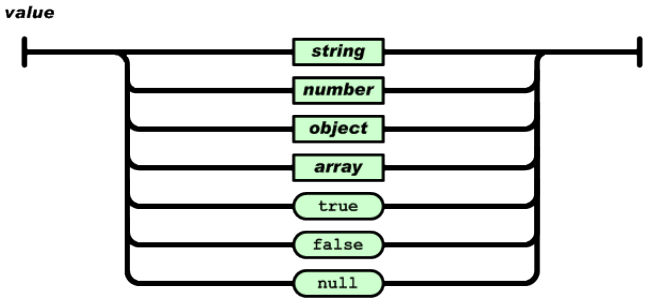


Figura 2.7: Valor em JSON. Fonte: [JSON 2006]

String

Uma *string* é uma sequência de zero ou mais caracteres *Unicode*, delimitadas por aspas (“”). A codificação por defeito de um ficheiro JSON é UTF-8.

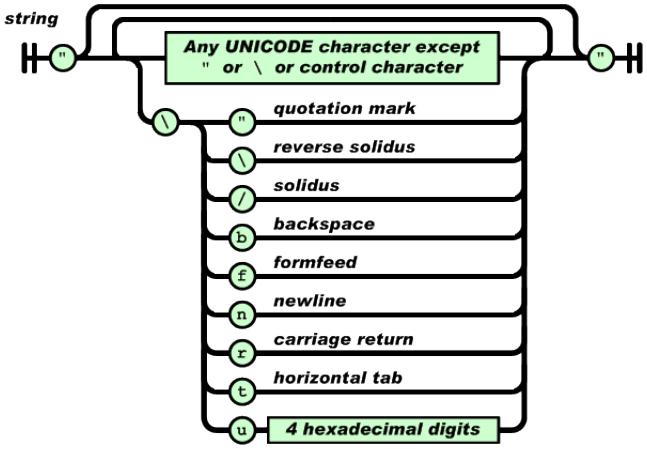


Figura 2.8: *String* em JSON. Fonte: [JSON 2006]

Número

Um número é representado em JSON da mesma forma que é representado nas linguagens de programação C e JAVA. No entanto, os formatos octal e hexadecimal não são utilizados.

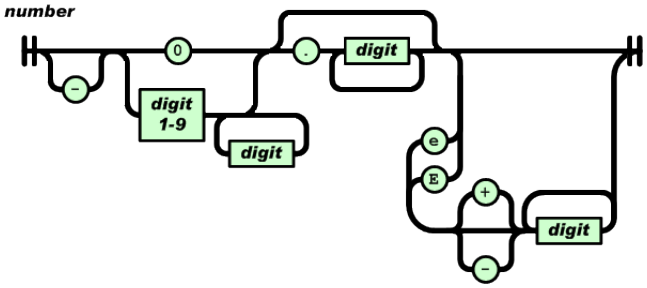


Figura 2.9: Número em JSON. Fonte: [JSON 2006]

As estruturas de dados descritas anteriormente podem ser facilmente identificadas no exemplo seguinte:

```
1  {
2    "filmes": {
3      "filme": {
4        "titulo": "The Avengers",
5        "ano": 2012,
6        "duracao": {
7          "metrica": "min",
8          "value": 142
9        },
10       "resumo": "Nick Fury of S.H.I.E.L.D. brings together a team of super humans to form The
             Avengers to help save the Earth from Loki and his army.",
11       "actores": {
12         "actor": [
13           "Robert Downey Jr.",
14           "Scarlett Johansson",
15           "Jeremy Renner",
16           "Tom Hiddleston"
17         ]
18       },
19       "categorias": {
20         "accao": null,
21         "aventura": null
22       }
23     }
24   }
25 }
```

- **Objecto**: Representado, por exemplo, nas linhas (03) a (23). O nome do objecto é "filme" e o valor é um conjunto de objectos.
- ***Array***: Na linha (12) é definido um *array*. Conta com 3 elementos (*strings*), devidamente separados por vírgula.

Vantagens do JSON

- Linguagem pouco expressiva: menor *overhead*.
- Parte integrante do JS, permitindo uma maior flexibilidade e tratamento dos dados.
- É possível importar dados de origens diferentes através de JSONP.

Desvantagens do JSON

- Falta de suporte a *namespaces*.
- Bastante sensível a erros de sintaxe.

2.1.7.3 JavaScript Object Notation with Padding

Com a introdução de portais web 2.0 (ver Web 2.0 (2.6)), o processamento e renderização das páginas web é efectuado no lado do cliente. Este característica permite a implementação de *mashups*, com a obtenção de dados a ser realizada directamente a partir do *browser*. Através da tecnologia *Asynchronous JavaScript and XML* (AJAX), é possível que um cliente (*browser*) efectue pedidos HTTP assincronamente.

Este tipo de comunicações é conseguido através da classe `XMLHttpRequest` da linguagem JS. Contudo, uma restrição denominada *Same-Origin Policy* (SOP) previne a obtenção de dados provenientes de servidores externos ao do *website* a partir do qual o pedido é gerado. O JSONP é utilizado precisamente para contornar esta restrição [Richardson 2007, Chatti 2011].

O JSONP é um complemento ao JSON que providencia um método para aceder a dados alojados num servidor de diferente domínio. Esta tecnologia é, no fundo, uma forma de representar dados no formato JSON como argumento de uma função JS. Por exemplo, considere-se o seguinte objecto JSON:

```
1 { biblioteca: "filmes", itens: 20 }
```

A representação do objecto em JSONP com a função de *callback* `mostrar_itens` seria:

```
1 mostrar_itens({ biblioteca: "filmes", itens: 20 })
```

Se considerar-mos que a nossa página web contém uma função JS de protótipo `mostrar_itens(dados)`, será possível aceder aos dados disponibilizados através de JSONP:

```
1 function mostrar_itens(dados) {
2   document.write("A biblioteca " + dados.biblioteca + " contém " + dados.itens);
3 }
```

Desta forma é possível carregar dinamicamente dados provenientes de um serviço em JSONP e processá-los na página web previamente carregada.

Same-origin Policy

Hoje em dia há vários métodos de segurança inerentes a um *browser* que evitam diversos tipos de ataques na web. Um desses métodos de segurança é denominado *Same-Origin Policy*. O SOP não permite que um objecto web alojado num *website* tenha acesso a outro objecto web alojado noutro *website* distinto. Este mecanismo é conseguido no *browser* através da verificação dos domínios, portas e protocolos dos dois objectos: se não coincidirem a comunicação não é estabelecida [Chatti 2011].

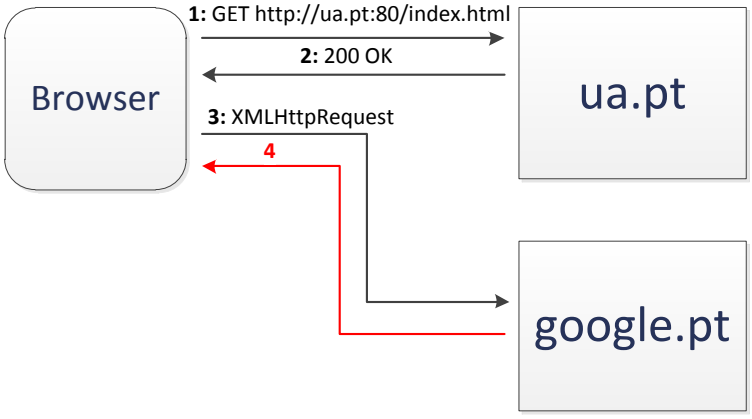


Figura 2.10: Exemplo de *Same-Origin Policy*

Na figura 2.10 é possível verificar as interacções inerentes a um pedido AJAX e a aplicação do SOP:

1. O *browser* do utilizador carrega a página `index.html`. A comunicação é feita através do protocolo HTTP na porta 80, a um servidor que tem por domínio “ua.pt”.

2. A página pedida no passo anterior existe e é carregada pelo *browser*.

3. Um *script* JS presente na página carregada no passo anterior tenta aceder a dados alojados num servidor de domínio “google.pt”. Este pedido é efectuado através AJAX.

4. Como os domínios são distintos, a comunicação não obedece à política SOP e, por isso, não é estabelecida. As comunicações AJAX realizadas através do JS da página carregada no passo 2 ficam circunscritas a recursos presentes num servidor com domínio “ua.pt”, realizadas através da porta 80 e do protocolo HTTP.

Embora seja necessário, o SOP apresenta-se como uma barreira à interoperabilidade de dados provenientes de servidores com domínios distintos na implementação de aplicações web. Uma das formas de contornar esta restrição é através de JSONP.

A *HyperText Markup Language* (HTML) é uma linguagem de marcação que permite construir páginas web. Semelhante ao XML, o HTML conta com diversos elementos que permitem definir a acção que o *browser* deve aplicar sobre os seus conteúdos. Um dos seus elementos é o `<scritp>`.

O elemento `<script>` permite definir *scripts* que são executados no *browser* do utilizador [Raggett 1999]. Este elemento é uma excepção ao SOP: pedidos originários da *tag* `<script>` para um domínio diferente são autorizados. Desta forma é possível obter dados provenientes de um domínio distinto evitando as restrições impostas pela política SOP.

Questões de Segurança no JSONP

Embora o JSONP seja um bom método para contornar as limitações imposta pelo SOP, é necessário ter em conta algumas questões de segurança. A utilização de JSONP requer a injecção de um elemento `<script>` na página HTML. Caso ocorra um erro quando os dados são retornados (função de *callback* errada ou sintaxe incorrecta no JSON), este erro não será visível, impossibilitando o tratamento de uma eventual excepção. Um outro problema associado à utilização do JSONP prende-se com a confiança que existe nos recursos pedidos. O JSONP apenas deve ser considerado quando todos os recursos, de domínios distintos, são controlados pela mesma entidade. Visto que um serviço JSONP retorna uma chamada a uma função, com o argumento dessa função a resposta JSON, a aplicação web fica vulnerável a uma série de ataques. Por isso é essencial que os serviços JSONP provenham de fonte fidedigna [Özses 2009].

Alternativa ao JSONP

O JSONP poderá ser a solução em casos onde estão disponíveis serviços web que exportem dados neste formato. No entanto muitos dos serviços web existentes só exportam dados no formato XML, impedindo a criação de *mashups* web que necessitem de carregar recursos no lado do cliente provenientes de outros domínios. Uma solução, tanto para esta problemática como para as questões de segurança do JSONP, denomina-se *Cross-Origin Resource Sharing* (CORS).

O CORS é um mecanismo que permite a realização de pedidos a domínios distintos no lado do cliente (*browser*). Tal é conseguido através do campo `Origin` no cabeçalho HTTP, que deve ser definido no pedido, e do campo `Access-Control-Allow-Origin` no cabeçalho HTTP da resposta [van Kesteren 2012]. No exemplo da figura 2.10, as alterações seriam as seguintes:

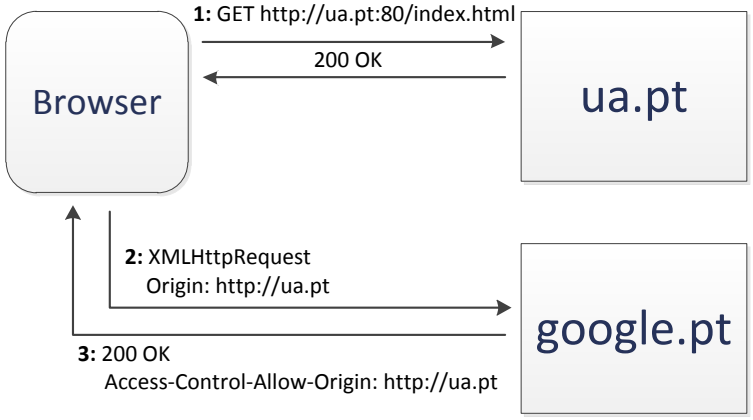


Figura 2.11: Exemplo de Cross-Origin Resource Sharing

1. É realizado um pedido HTTP para carregar a página `index.html`, alojada no domínio "ua.pt", sendo recebida a resposta (página HTML com JS).

2. É necessário invocar um serviço web XML, via AJAX, presente num servidor de domínio "google.pt". Para tal é necessário definir o campo `Origin` do cabeçalho HTTP do pedido. Neste campo deve ser especificado o domínio de origem do pedido [RFC 6454], que neste caso seria "http://ua.pt".

3. Caso o pedido seja aceite por parte do servidor alojado no domínio "google.pt", a resposta deverá incluir o campo `Access-Control-Allow-Origin` no cabeçalho HTTP da resposta, contendo o valor do campo `Origin` definido no passo anterior.

Este mecanismo de partilha de recursos inter-domínios está disponível nas últimas versões dos principais *browsers*.

Com a introdução do HTML5 e o respectivo suporte pela maioria dos *browsers*, surge outra alternativa ao JSONP denominada de *Cross-Document Messaging* (ou *Web Messaging*) [Hickson 2012]. Este mecanismo permite que documentos provenientes de diferentes domínios comuniquem entre si, através de uma API em JS. Utilizando a função `postMessage()`, é possível enviar uma mensagem de texto para um destinatário identificado pelo seu URL. Para receber uma mensagem, é necessário definir um *Event Listener* responsável por executar uma função sempre que um novo evento é despoletado. Quando uma mensagem é recebida, é definido um objecto que contém vários atributos, de entre os quais se podem destacar o `data`, que contém o valor da mensagem, e o `origin`, que contém informações sobre o remetente da mensagem (esquema, *hostname* e porto). Este atributo permite definir alguma segurança em interacções inter-domínios: qualquer mensagem proveniente de um domínio não confiável deve ser descartada.

2.1.7.4 Comma Separated Values

O formato CSV, tal como os formatos eXtensible Markup Language (2.1.7.1) e JavaScript Object Notation (2.1.7.2), tem como principal propósito a representação e troca de informação entre aplicações. Embora este formato seja bastante utilizado por várias aplicações, nunca foi formalmente documentado. Devido à falta de documentação oficial sobre este formato, várias instituições optaram por implementar sintaxe própria para representar dados através de ficheiros CSV [RFC 4180]. No entanto, está documentado no RFC 4180 o formato que é comum às várias implementações:

- Cada registo esta localizado numa linha individual, delimitada por um quebra de linha (`CRLF`), com excepção ao último registo.
- É possível definir um cabeçalho.
- Cada registo pode conter vários campos, que devem ser separados por vírgula (`,`).
- Cada campo que contiver vírgulas (`,`), quebras de linha (`CRLF`) ou aspas (`"`) deve ser delimitado por aspas.
- Caso exista um campo que contenha aspas, estas devem ser precedidas de outras aspas.

Exemplo de um documento no formato CSV:

```
1 título,ano,duracao,resumo
2 "The Avengers",2012,142,"Nick Fury of S.H.I.E.L.D. brings..."
3 "The Amazing Spider-Man",2012,139,"Peter Parker finds a clue that might..."
4 "Dark Shadows",2012,127,"An imprisoned vampire, ""Barnabas Collins"", is set..."
```

O CSV é um formato baseado em texto que não requer qualquer tipo de codificação.

2.2 Single Sign-On

O SSO é um método de autenticação num ambiente federativo, que permite que um utilizador apenas necessite de introduzir as suas credencias uma única vez. A partir desse momento, o utilizador poderá aceder a qualquer outro recurso sem necessitar de uma nova autenticação [Teixeira 2009]. Este mecanismo reduz significativamente a complexidade de um sistema de informação, permitindo um método de autenticação único nos diverso *websites* de uma organização. Em termos de segurança, este método pode ser visto de dois pontos distintos. Por um lado, uma aplicação web não necessita de implementar um sistema de autenticação próprio, que muitas vezes são susceptíveis a vulnerabilidades. Por outro lado, e do ponto de vista do utilizador, se as suas credenciais forem comprometidas, qualquer pessoa tem acesso a todos os serviços do utilizador que implementem este método de autenticação.

2.2.1 Security Assertion Markup Language

Desenvolvido pela *Advancing Open Standards for the Information Society* (OASIS) em Março de 2005, o standard *Security Assertion Markup Language* (SAML) é baseado em XML e tem como objectivo a troca

de dados de autenticação, privilégios e atributos do utilizador entre serviços [OASIS 2005]. O SAML está dividido em quatro componentes [Madsen 2005, Teixeira 2009]:

- **Alegações**: uma alegação é um conjunto de informações sobre um utilizador que são fornecidos de uma entidade para outra. Existem três tipos de alegações:
  - **Autenticação**: O utilizador foi autenticado de uma certa forma e numa determinada data. Este tipo de alegação é geralmente emitida por uma autoridade SAML denominada de fornecedor de entidades. A sua responsabilidade é a de autenticar utilizadores e manter registos dos dados relacionados com a sua sessão, enquanto esta for válida.
  - **Atributo**: Informação específica de um determinado utilizador. Esta informação é representada sob a forma nome-valor.
  - **Decisão de autorização**: Informação sobre a decisão de permitir, ou não, o acesso a um determinado recurso por parte do utilizador.
- **Protocolos**: o SAML define uma série de mensagens protocolares do tipo pedido/resposta que permitem o fornecedor de serviços a:
  - Requisitar a uma autoridade SAML uma, ou mais, alegações;
  - Pedir a autenticação de um utilizador a um provedor de identidades e obter as alegações correspondentes;
  - Cancelar que a utilização de uma identidade seja terminada;
  - Etc...
- **Vinculações**: as vinculações têm como objectivo o mapeamento entre mensagens protocolares e normas de comunicação entre sistemas.
- **Perfis**: este tipo de alegações definem um conjunto de restrições e/ou extensões para o suporte da utilização de SAML numa determinada aplicação.

Para além dos blocos descritos atrás, existem mais dois blocos que, embora não sejam de utilização obrigatória, poderão ser de alguma utilidade [Ragouzis 2008]. O bloco **metadados** que permite definir uma forma de expor e partilhar a configuração entre entidades SAML, e o bloco **contexto de autenticação** que permite definir o tipo de autenticação a que um utilizador está sujeito. Por exemplo, um fornecedor de serviços poderá necessitar de informação detalhada referente ao tipo e nível de autenticação a ser aplicada a um utilizador. No caso de a autenticação efectuada posteriormente não cumprir os níveis de segurança exigidos, o fornecedor de serviços poderá pedir ao utilizador que se autentique através de um método mais seguro.

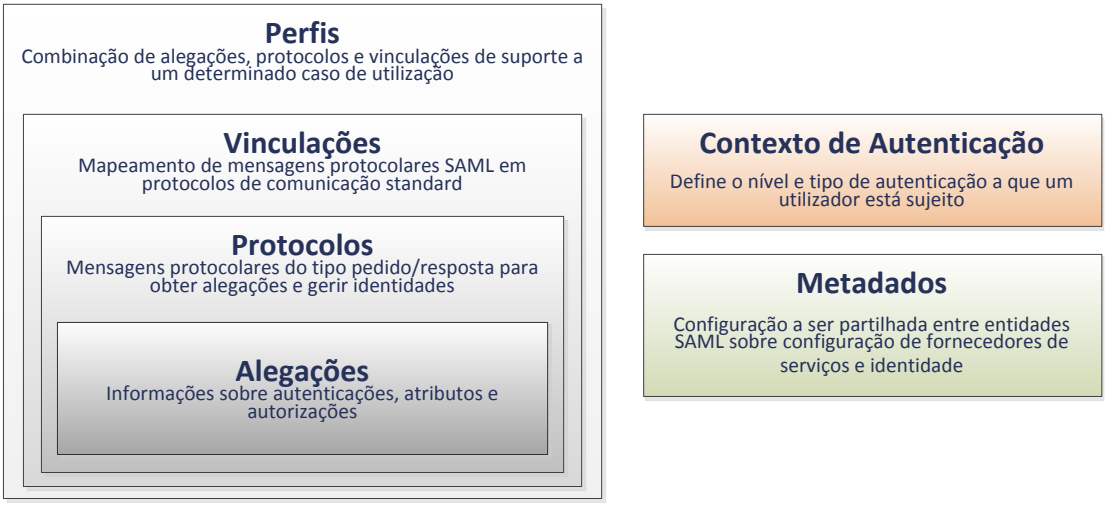


Figura 2.12: Relação entre os diferentes blocos SAML. Fonte: [Ragouzis 2008]

Com a introdução da versão 2.0 do SAML, foram adicionadas uma série de novas características, das quais se podem destacar os pseudónimos, gestão de identidades e de sessões, metadados, encriptação, perfis de atributos, suporte a dispositivos móveis, mecanismos de políticas de privacidade e um método de pesquisa de fornecedores de identidades.

2.2.2 Shibboleth

O projecto Shibboleth é uma iniciativa do consórcio *Internet2*. Baseado em SAML, este sistema de código livre permite a gestão de identidades assim como providenciar um mecanismo SSO de autenticação federativa [Internet2 2005]. Desenvolvido para se ajustar às necessidades de instituições de ensino superior, investigação e respectivos parceiros, o *Shibboleth* utiliza activamente SAML na troca de mensagens.

Tal como o SAML, o *Shibboleth* esta dividido em vários componentes. Estes componentes incluem um IdP, um *Service Provider* (SP) e um componente opcional denominado "*Where are you from?*" (WAYF) [Scavo 2005].

O **IdP** é responsável por manter as credenciais e atributos de utilizadores. Quando recebe um pedido, o IdP verifica a validade das credenciais introduzidas ou entrega os atributos do utilizador ao SP. O IdP está dividido em três sub-componentes: uma autoridade de autenticação, responsável por emitir alegações de autenticação aos outros componentes, um serviço de SSO, que inicia o processo de autenticação no IdP e uma autoridade de atributos, que autentica e autoriza todos os pedidos que recebe. Apenas entidades autorizadas (*relay parties*) têm permissão para efectuar pedidos ao IdP.

O componente **SP** tem como função gerir os recursos protegidos, assegurando que apenas são partilhados com entidades autorizadas.

O **WAYF**, componente de carácter opcional, é operado separadamente do IdP e do SP. Este serviço pode ser usado pelo SP para determinar o IdP de preferência do utilizador.

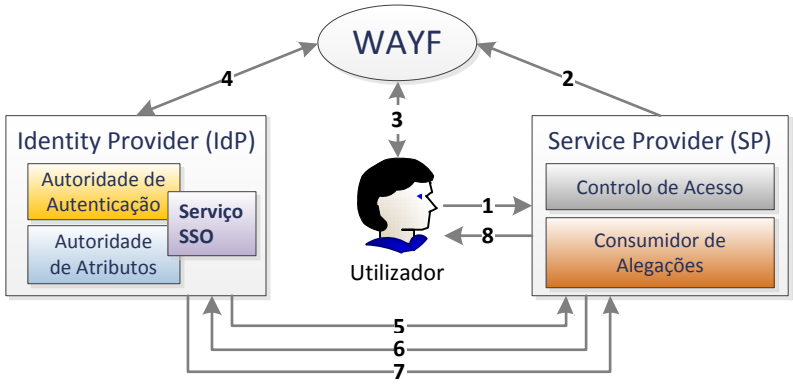


Figura 2.13: Funcionamento do *Shibboleth*

Na figura 2.13 está esquematizada uma transacção através de *Shibboleth* com o utilizador (*browser*), no passo **1**, a contactar o SP para obter um recurso protegido. No passo **2**, o utilizador é redireccionado para o WAYF federativo, onde são listados os IdPs suportados. Após o utilizador seleccionar o IdP da sua instituição, é realizada a autenticação (passo **4**). É da responsabilidade do IdP o método de autenticação do utilizador. Após autenticação com sucesso, é gerada no IdP uma sessão para o utilizador que inclui alegações de autenticação SAML, sendo de seguida enviada, no passo **5**, para o SP. Após receber as alegações, o SP requer informações sobre os atributos do utilizador (passo **6**). Fica a cargo do IdP seleccionar os atributos a que o SP tem acesso, enviando-os, no passo **7**, ao SP. Por fim, no passo **8**, o SP pode decidir se o utilizador tem permissões de acesso ao recurso pretendido, com base nas alegações recebidas pelo IdP [Ying 2010].

2.2.3 OpenID

O *OpenID* é um protocolo SSO que permite a autenticação em diversos *websites* através de um *Uniform Resource Identifier* (URI). Desenvolvido em 2005 pela comunidade *open source*, este protocolo caracteriza-se pelo controlo que é concedido ao utilizador da sua própria identidade: é o utilizador que decide a informação a ser partilhada com uma aplicação ou *website*. Para obter uma identidade *OpenID* o utilizador necessita de se registar num IdP. Após concluir o registo, será atribuído um URI (identificador) ao utilizador que representará a sua identidade. No processo de autenticação através deste protocolo estão definidos dois componentes: o **Relying Party** (RP), que representa o *website* no qual o utilizador tenta autenticar-se, e o **IdP**, que representa o servidor *OpenID* detentor das credenciais do utilizador e responsável por validar a identidade do URI ao RP.

No protocolo *OpenID* estão definidos dois modos de funcionamento: o modo **Dumb** e o modo **Smart**. A grande diferença entre os dois deve-se, no modo *Smart,* à manutenção da informação sobre o estado da ligação e das chaves partilhadas [Rehman 2007].

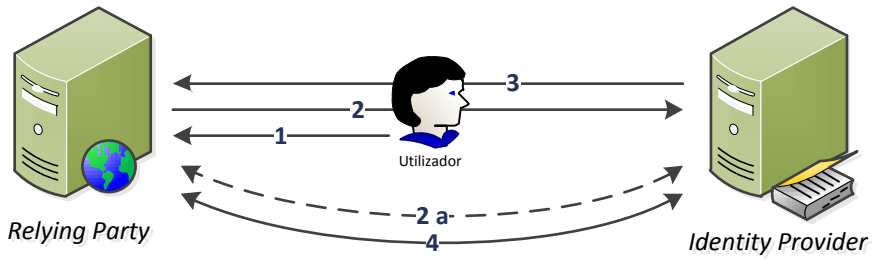


Figura 2.14: Autenticação através de *OpenID*

Uma autenticação, em modo *Dumb*, caracteriza-se pela troca de mensagens entre o IdP e o RP, tendo como *man-in-the-middle* o utilizador. Na figura 2.14 estão presentes os passos necessários à autenticação no RP:

1. O utilizador acede ao *website* onde pretende realizar a autenticação (RP). Neste é apresentado ao utilizador um formulário onde colocar o seu identificador (URI), assim como um botão de *login*.

2. Após o utilizador carregar no botão de *login*, o RP identifica o URL do IdP através do protocolo *Yadis* e reencaminha o *browser* do utilizador através de um HTTP GET para o IdP, de modo a obter as alegações necessárias. Opcionalmente, o RP poderá estabelecer uma ligação com o IdP para a troca de chaves no caso de uma eventual comunicação futura (2a).

3. Caso o utilizador ainda não tenha uma sessão iniciada no IdP, é necessário a introdução das suas credenciais (utilizador e palavra-chave). Se a autenticação foi bem sucedida no IdP, o *browser* do utilizador é reencaminhado para o RP através de um HTTP GET. Juntamente com este redireccionamento estão anexadas informações sobre as alegações do utilizador e respectiva assinatura.

4. Caso a autenticação seja bem sucedida, o RP irá estabelecer uma ligação directa com o IdP, preferencialmente sobre o protocolo *Secure Sockets Layer* (SSL). Esta comunicação tem como objectivo obter uma cópia da informação de autenticação obtida no passo 3 pelo RP. É realizada uma comparação entre a informação recebida pelo *browser* do utilizador e a recebida pelo IdP de modo a evitar ataques de personificação. Caso ambas as informações coincidam, o utilizador é autenticado no RP.

Este tipo de autenticação apenas é realizada em duas situações: numa primeira comunicação entre o RP e o IdP ou caso o RP não mantenha as chaves partilhadas numa ligação prévia. No modo *Smart*, a ligação 4 da figura 2.14 não é realizada. O RP já dispõe das chaves partilhadas (no passo 2a) necessárias para validar a informação de autenticação redireccionada pelo *browser* do utilizador.

2.2.4 Microsoft Account

O *Microsoft Account* é um sistema centralizado de autenticação de utilizadores desenvolvido pela *Microsoft*. Previamente conhecido como *Microsoft .NET Passport* ou *Windows Live ID*, este sistema caracteriza-se por permitir que um utilizador aceda a diversos *websites* que suportem esta tecnologia,

necessitando apenas de se autenticar uma vez. Para que um *website* tenha a possibilidade de usufruir deste método de autenticação é necessário pertencer à rede do *Microsoft Account*, sendo partilhada uma chave criptográfica entre ambas as entidades. Esta chave tem como objectivo a encriptação/desencriptação de *tickets* gerados pelo *Microsoft Account*. Após autenticação, este serviço poderá delegar o acesso a informações pessoais do utilizador, mas sempre após autorização [Oppliger 2003]. Caso a autenticação tenha sido bem sucedida, é partilhado um identificador único com o *website* em causa. Na figura 2.15 estão presentes os passos necessários para autenticação através deste serviço.

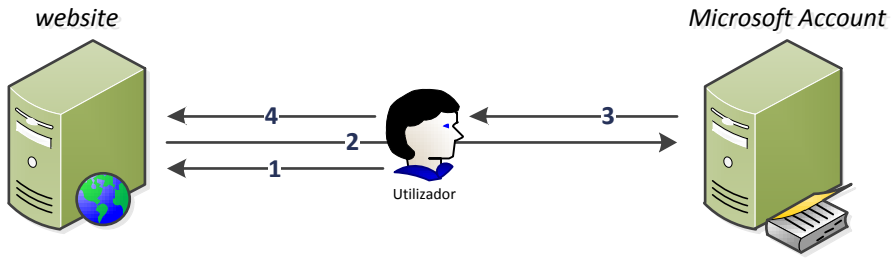


Figura 2.15: Autenticação através de *Microsoft Account*

1. O utilizador acede a um *website* que permite autenticação através do serviço *Microsoft Account*.

2. Após carregar no botão para se autenticar, o utilizador é reencaminhado para uma página de autenticação, onde deverá introduzir as suas credenciais (nome de utilizador e palavra-chave). Caso a autenticação seja bem sucedida, são gerados dois *tickets*, contendo ambos um identificador único e a data de autenticação.

3. Ambos os *tickets* são guardados no *browser* do utilizador. Um deles é guardado com o domínio do *Microsoft Account* e encriptado com a sua chave privada.

4. O outro *ticket* é guardado com o domínio do *website*, e encriptado com a chave pública. Como o serviço de autenticação não tem acesso ao domínio do *website*, este *cookie* é passado como argumento do redireccionamento HTTP e guardado no *browser* do cliente pelo próprio *website*.

No caso do utilizador desejar autenticar-se num *website* distinto através do *Microsoft Account*, o processo é semelhante ao da figura 2.15. No entanto, e caso a sessão ainda esteja activa, o utilizador não necessita de introduzir as suas credenciais (passo 2) [Teixeira 2009].

2.2.5 BrowserID

O *BrowserID* é um protótipo de um sistema descentralizado de identificação desenvolvido pela *Mozilla* [Hilaiel 2011]. O conceito deste mecanismo passa por o utilizador provar que é o detentor de um endereço de *email* através de um mecanismo seguro. Este sistema é um dos primeiros passos para autenticação sem ser necessário recorrer a nomes de utilizador e palavras-chave, sendo o *browser*, a peça chave para a implementação dos mecanismos associados. No caso do *browser* não ter suporte para este

mecanismo, devem ser utilizadas bibliotecas JS que, juntamente com HTML5, providenciam as funcionalidades necessárias. O *BrowserID* recorre a criptografia assimétrica e assinaturas digitais para permitir que *browsers* gerem alegações assinadas sobre a identidade de um utilizador. Para a comunicação entre documentos de diferentes domínios, este sistema utiliza *Cross-Document Messaging*, permitindo a utilização imediata sem necessidade de alterar os *browsers* actuais.

A autenticação através de *BrowserID* está dividida em dois processos: o **Aprovisionamento de Certificado** e a **Geração de Alegações**.

Aprovisionamento de Certificado

Neste procedimento é verificado se o utilizador é detentor do endereço de *email* especificado e é gerado um certificado assinado. Para tal, é necessário fazer a verificação através de um serviço que providencie contas de *email*, tal como o *Gmail* ou o *Yahoo! mail*.

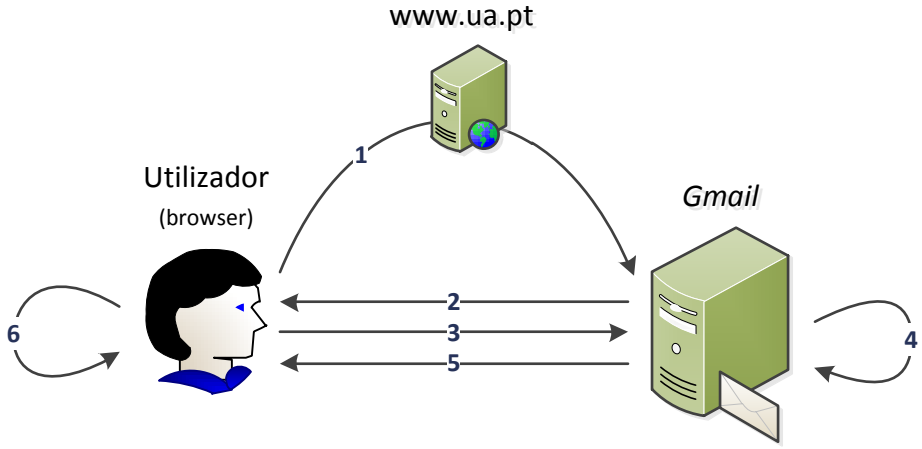


Figura 2.16: Aprovisionamento de certificado no *BrowserID*

No exemplo da figura 2.16, o serviço de email é o *Gmail*. Está subentendido que o *browser* do utilizador e o *Gmail* suportam o *BrowserID*. No caso do *Gmail* não suportar o *BrowserID*, é realizado um *fallback* para o domínio deste sistema (`browserid.org`), ficando este responsável por verificar manualmente o endereço de *email*.

1. O utilizador acede ao *website* `www.ua.pt` (RP), no qual pretende autenticar-se. Após introduzir o endereço de *email* no RP, o utilizador é reencaminhado para o serviço de *email*, onde deve proceder à sua autenticação.

2. Após autenticar-se, é invocada a função JS `navigator.id.genKeyPair()`. Esta função do lado do cliente implementada pelo *browser* permite gerar um par de chaves (pública e privada). Quando o processo estiver concluído, a chave pública é devolvida ao *Gmail*. O par de chaves é salvaguardado na *cache* do *browser* durante o resto da sessão.

3. Neste passo, uma rotina JS encarrega-se de enviar a chave pública para o servidor do *Gmail* através de SSL.

4. O *Gmail* assina o *email* do utilizador, a chave pública enviada no passo 3 e um intervalo de validade, gerando um objecto *JSON Web Token* (JWT).

5. De seguida o objecto JWT (certificado) gerado no passo anterior é enviado ao utilizador, através da invocação da função JS `navigator.id.registerVerifiedEmail()`.

6. Juntamente com a chave privada salvaguardada no passo 2, o *browser* move o certificado para o *BrowserID keyring*. Neste ponto o utilizador tem as ferramentas necessárias à geração de alegações que comprovem a sua identidade.

Geração de Alegações

Este processo permite ao *browser* do utilizador gerar alegações que comprovem a posse do endereço de *email* por parte do utilizador. Este processo é executado depois de o *browser* já possuir o certificado obtido no processo aprovisionamento de certificado.

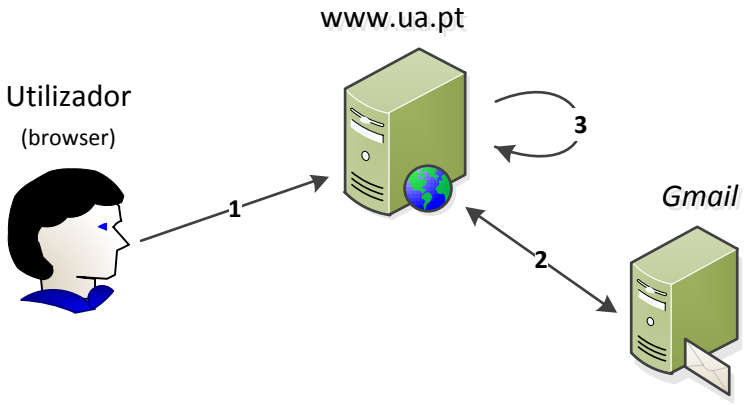


Figura 2.17: Geração de alegações no *BrowserID*

Na figura 2.17 estão representados os passos para gerar uma alegação:

1. O RP (`www.ua.pt`) transmite, de forma segura, a alegação do cliente para o seu servidor. No lado do servidor são verificadas as validades temporais, tanto do certificado como da alegação. O *hostname* do email contido na alegação é extraído, de modo a que o RP possa contactar o serviço de email (neste caso o *Gmail*).

2. O RP contacta o *Gmail* e é trocada a chave pública, de modo a verificar a validade do certificado.

3. A assinatura do certificado e da alegação são verificadas. A correcta validação destes dois elementos permite assegurar ao RP que a chave pública do utilizador é válida e que este é o detentor do endereço *email* em causa.

No caso de o serviço de *email* não suportar o *BrowserID*, o certificado incluirá um campo onde pode ser definido o domínio da autoridade de validação.

Devido à autenticação do utilizador ser efectuada através de chaves criptográficas, este método torna-se muito mais seguro que a utilização de credenciais. Este método dispensa a utilização directa de

credenciais, relegando este processo para o serviço de email. Caso este seja comprometido, também a autenticação através do *BrowserID* fica comprometida [Halpin 2012].

2.3 Autorização

Na secção anterior forma estudadas diversas tecnologias de SSO e gestão de identidades. Estas tecnologias permitem a uma entidade identificar-se perante um *website* ou até mesmo um serviço. Embora sejam partilhadas informações sobre o utilizador, não são especificadas quaisquer tipos de autorizações sobre os recursos que o utilizador detém nesse serviço. Surge então a necessidade da implementação de um método seguro para a concessão de autorizações sobre o acesso a recursos pertencentes a uma entidade.

2.3.1 eXtensible Access Control Markup Language

Desenvolvida pela OASIS em 2003 [OASIS 2010], a *eXtensible Access Control Markup Language* (XACML) define uma linguagem de controlo de acessos e autorizações. Num cenário típico de controlo de acesso e autorização estão envolvidas três entidades: um sujeito, um recurso e uma acção. O sujeito é o elemento que efectua um pedido de acção sobre um determinado recurso. A XACML tem como objectivo a implementação de uma linguagem padrão para a descrição de autorizações de acesso a recursos protegidos. Para a utilização desta linguagem, é necessário a existência de diversos actores:

- **Policy Enforcement Point** (PEP): entidade responsável pela controlo de acessos. Submete pedidos recebidos e aplica as autorizações recebidas.
- **Policy Decision Point** (PDP): entidade que avalia políticas a serem aplicadas e emite decisões de autorização.
- **Policy Information Point** (PIP): entidade fonte de atributos (sujeito, recurso, acção, etc).
- **Policy Administration Point** (PAP): entidade responsável pela gestão de políticas.

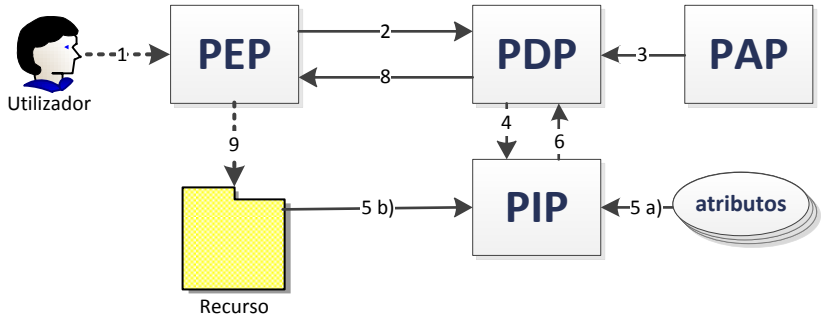


Figura 2.18: *Workflow* de uma autorização através de XACML

Na figura 2.18 estão representadas as mensagens trocadas entre os vários actores num pedido de acesso a um recurso. É realizado um pedido de acção sobre um determinado recurso (ficheiro, base

de dados, serviço web, etc) por parte de um utilizador. Este pedido é direccionado ao PEP (1), que por sua vez produz uma mensagem no formato XACML. Esta mensagem é encaminhada para o PDP (3). Opcionalmente o PEP poderá adicionar atributos que sejam relevantes para a decisão final. O PDP recebe do PAP as políticas referentes ao recurso que o utilizador pretende aceder, no formato XACML (3). De modo a poder formular uma decisão sobre o pedido recebido, o PDP necessita de outras informações para além das políticas recebidas do PAP. Para tal envia um pedido de atributos ao PIP (4). O PIP reúne vários atributos (5 a)): sobre o utilizador, recurso (5 b)) e acção, reencaminhando-os de seguida para o PDP (6). Com base em todos os elementos descritos anteriormente, o PDP formula a decisão sobre o pedido recebido. Esta decisão é enviada o PEP com o formato XACML (8). Caso a decisão de acesso ao recurso seja positiva, o PEP concede permissões de acesso ao utilizador (9). Embora o PEP conceda acesso ao recurso em casa, este poderá requerer o cumprimento de obrigações por parte do utilizador antes do acesso ser autorizado.

A XACML define uma linguagem que permite definir políticas. Esta linguagem esta dividida em três componentes: **PolicySet**, **Policy** e **Rule**.

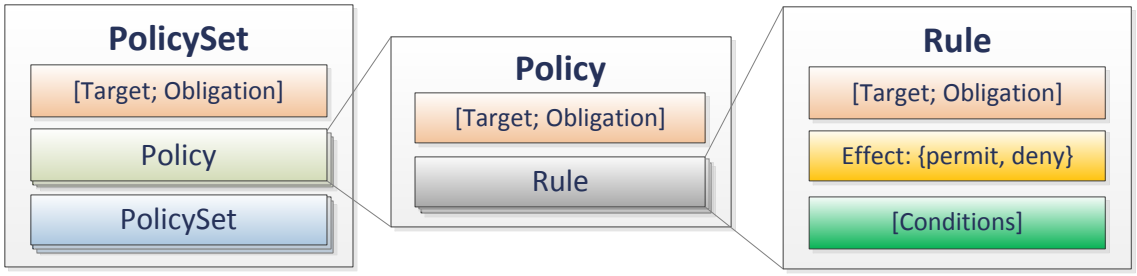


Figura 2.19: Componentes da linguagem de políticas

Cada elemento de uma política deve conter os elementos *Target* e *Obligation*. O elemento *Target* permite determinar se a política é relevante para o pedido em causa, resultando na sua eventual aplicação. O elemento *Obligation* permite definir as obrigações que o PEP deve aplicar antes de o utilizador poder aceder a um recurso. Quando uma política é avaliada, a autorização é enviada para o PEP juntamente com as respectivas obrigações, sendo este responsável pela sua aplicação.

As *Rules* (regras) definem os detalhes a aplicar através de uma política. Neste elemento existem dois campos que determinam se o a resposta ao pedido: o *Effect*, que especifica o resultado da regra (se é permitido ou negado) e o *Conditions*, que determina as condições sob as quais a regra deve ser aplicada.

A linguagem XACML é hoje em dia bastante utilizada por diversas aplicações. A grande vantagem desta linguagem é a sua flexibilidade e expressividade, nomeadamente em relação a decisões dependentes de contexto externo [Alm 2010].

2.3.2 Open Authorization Protocol

O protocolo OAuth começou como um esforço comunitário entre várias companhias de modo a resolver o problema de gestão de identidades entre provedores de serviços. A primeira versão do OAuth, 1.0, foi

proposta a 4 de Dezembro de 2007. Em Maio de 2009, o *Internet Engineering Task Force* (IETF) criou o *OAuth Working Group* [Leiba 2012].

Especificado no *Request For Comments* (RFC) 5849, o protocolo OAuth é definido como "um método para que clientes acedam a um recurso num servidor, em nome do dono do recurso. Também providencia um processo para que utilizadores autorizem o acesso aos seus recursos num servidor por parte de terceiros, sem a necessidade de partilhar as suas credenciais".

No modelo tradicional cliente-servidor, o utilizador usa as suas credenciais (normalmente utilizador e palavra-passe) para se autenticar e ter acesso a recursos presentes no servidor. No modelo OAuth, o cliente requer acesso a recursos controlados pelo dono destes, mas que estão alojados num servidor. Além disso, o OAuth permite que o servidor verifique se o cliente tem autorização para aceder ao recurso, como também identificar o cliente que executa o pedido [RFC 5849].

O modelo sobre o qual assenta o OAuth é distinto do modelo XACML. No XACML são definidas políticas de acesso a recursos por uma autoridade. Essas políticas, em conjuntos com outros factores, é que determinam se a entidade em causa tem acesso ao recurso e sob que condições. No caso do OAuth, a autorização de acesso a um recurso por parte de uma entidade é completamente dependente do seu detentor. Não são definidas quaisquer tipo de políticas ou obrigações: o acesso ao recurso é autorizado ou negado. Além disso, o protocolo OAuth implementa um mecanismos de concessão de autorizações baseada em *tokens*. Isto permite que o utilizador apenas necessite de conceder autorizações uma vez para que a entidade que deseja aceder ao recurso obtenha uma janela temporal de autorização.

O protocolo OAuth assenta sobre HTTP para a troca de mensagens.

2.3.2.1 Modo de Funcionamento

Para explicar o funcionamento do protocolo OAuth, é necessário especificar algumas terminologias utilizadas:

- **Cliente**: serviço que pede autorização para aceder a um recurso.
- **Utilizador**: entidade proprietária da informação a que o cliente quer aceder.
- **Servidor do Recurso**: serviço que providencia acesso à informação pretendida.
- **Servidor de Autenticação**: serviço que verifica as credenciais e autentica o utilizador.
- **Servidor de Autorização**: serviço que verifica a se a autorização de acesso aos recursos é disponibilizada pelo utilizador.

Consideremos o seguinte exemplo: todos os alunos da UA têm um horário escolar. Esta informação é pessoal e privada. Um aluno (utilizador) deseja importar o seu horário para outro serviço, por exemplo, o *Google Calendar* (cliente), através de OAuth:

1. O utilizador acede, através de um *browser*, ao *website* do *Google Calendar* e carrega no botão "Importar horário da UA".

2. O *browser* do utilizador é redireccionado para o servidor de autorização. Este servidor contem um mecanismo de autenticação, como por exemplo um IdP, responsável pela autenticação do aluno.

3. Após a autenticação ter sido bem sucedida, o *browser* do utilizador é redireccionado para o servidor de autorização. Neste passo, é apresentado ao utilizador a intenção de o cliente (*Google Calendar*) aceder ao seu horário escolar.

4. Caso o utilizador autorize, é devolvido um código de autorização ao cliente. Este código de autorização consiste num *token* temporário que permite verificar que o utilizador autorizou o acesso à informação. O *browser* do utilizador é redireccionado para o *Google Calendar*.

5. Este próximo passo não necessita da interacção do utilizador: o cliente necessita de trocar o *token* temporário que obteve no passo anterior por um *Access Token*. O *Access Token* consiste num par de chaves que, juntamente com outros *tokens*, permitem o acesso ao recurso por parte do cliente. O *Access Token* deve expirar ao fim de algum tempo.

Após cumpridos os passos definidos anteriormente, a aplicação cliente (*Google Calendar*) já tem informação relativa ao horário escolar do aluno.

Do ponto de vista do cliente, esta operação é bastante fácil e transparente: aceder ao serviço consumidor, autenticar e autorizar o acesso aos dados e estes aparecem no serviço consumidor. Do lado do serviço consumidor este processo não é tão trivial.

O processo de consumo de recursos através de OAuth traduz-se numa troca de mensagens entre o cliente, utilizador e um servidor de autorização, através do protocolo HTTP ou HTTPS. Normalmente o mecanismo de autorização é parte integrante do servidor OAuth. O cliente necessita de ser reconhecido pelo servidor OAuth como uma entidade confiável. Para tal, são atribuídas duas *tokens* ao cliente: uma `consumer_key` e um `consumer_secret`. A função destas duas chaves é a de identificar o cliente e permitir assinar os pedidos efectuados.

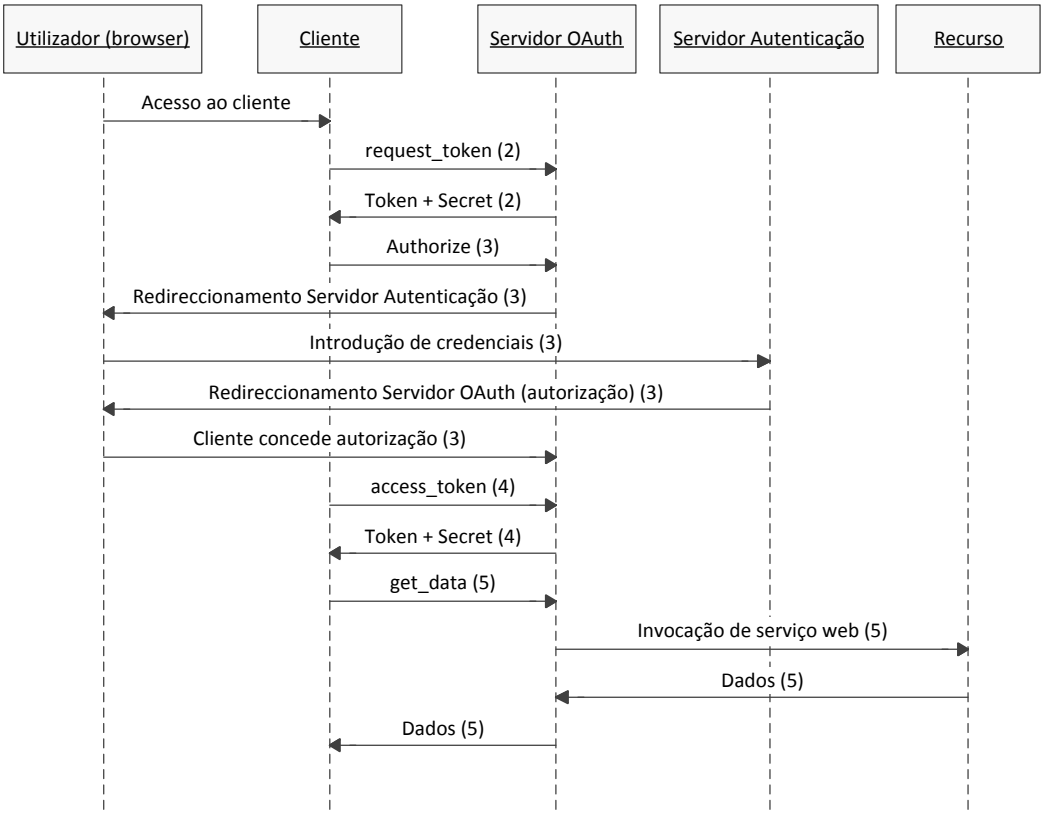


Figura 2.20: Diagrama de sequência do processo OAuth

Para que seja correctamente efectuada uma troca de informação através de OAuth, devem ser executados os seguintes passos [RFC 5849][2]:

1. O cliente deve efectuar um registo no servidor OAuth. Neste registo deve ser especificado o nome do cliente, uma breve descrição. Ao cliente são fornecidos dois *tokens*: uma `consumer_key` e um `consumer_secret`.

2. A primeira comunicação entre cliente e servidor serve para obter uma autorização temporária. Esta autorização é concedida sob o formato de um *token*, denominado "Request Token". O cliente necessita de enviar um pedido HTTP, através do método GET ou POST, para o servidor OAuth. Neste método devem ser especificados vários argumentos:

   - `oauth_consumer_key`: o *token* `consumer_key` atribuído ao cliente no passo 1.
   - `oauth_signature_method`: o método para assinar o pedido. O OAuth permite três métodos de assinatura: *Hash-based Message Authentication Code with Secure Hash Algorithm 1* (HMAC-SHA1), RSA-SHA1 e PLAINTEXT. Não é obrigatório o uso de nenhum método específico e cada implementação do servidor OAuth poderá adoptar o método de assinatura mais conveniente.

[2] Versão 1.0a

- `oauth_signature`: assinatura gerada através do algoritmo definido em `oauth_signature_method`. A construção desta assinatura é realizada através da concatenação dos seguintes elementos:
  - O método HTTP em letras maiúsculas (GET ou POST), seguido de um "i" comercial (`&`);
  - O URI base do pedido, depois de codificado, seguido de um "`&`". Por exemplo, o URI `http://identity.ua.pt/oauth/request_token` passa a `http%3A%2F%2Fidentity.ua.pt%2Foauth%2Frequest_token`;
  - Os parâmetros do pedido, normalizados e codificados.
- `oauth_timestamp`: o *timestamp* do pedido. O *timestamp* é um número inteiro definido pelos segundos decorridos desde as 0 horas do dia 1 de Janeiro de 1970.
- `oauth_nonce`: o *nonce* deverá ser uma string aleatória e única. Este parâmetro previne ataques de repetição, visto que o servidor OAuth guarda as *nonces* utilizadas e rejeita pedidos com este parâmetro repetido.
- `oauth_callback`: URI absoluto, codificado, para o qual o cliente é redireccionado no fim do processo de autorização[3].
- `oauth_version` (opcional): a versão OAuth a ser utilizada (1.0, 1.0a ou 2.0).

Exemplo de um pedido por parte do cliente:

```
GET oauth/request_token HTTP/1.1
Host: identity.ua.pt
Params:
  oauth_consumer_key=_b0e742818dfe3e9d3cb287e131f24d219bebcd4a7e&
  oauth_nonce=VCNJvGr6l7cMfO&
  oauth_signature=LbqX7zdMZbaIWq7xBoScjBjXWBM%3D&
  oauth_signature_method=HMAC-SHA1&
  oauth_timestamp=1336572381&
  oauth_version=1.0a
```

Resposta do servidor OAuth:

```
HTTP/1.1 200 OK
Content-Type: application/x-www-form-urlencoded

oauth_token=_cf2abf72caf253c642dbbe7a9f9a8053315b1c0f77&oauth_token_secret=
    _4aff822ce118cd448198fc699bbe31aa2f4233a61e&oauth_callback_confirmed=true
```

A resposta inclui dois novos *tokens* temporários: *oauth_token* e *oauth_token_secret*. Normalmente estes *tokens* tem uma validade bastante reduzida.

3. Depois de obter as credenciais temporárias, o próximo passo é a autenticação e autorização por parte do utilizador. O processo é em tudo semelhante ao passo anterior: deve ser feito um pedido ao servidor OAuth por parte do cliente para que o utilizador autorize o acesso ao recurso. Neste passo apenas é necessário incluir um parâmetro no pedido: `oauth_token`, obtido no passo anterior:

```
GET oauth/authorize HTTP/1.1
Host: identity.ua.pt
Params:
  oauth_token=_cf2abf72caf253c642dbbe7a9f9a8053315b1c0f77
```

[3] Embora esteja especificado no RFC que este parâmetro é obrigatório, alguns servidores OAuth definem um URI de *callback* estático aquando do registo do cliente. Este procedimento evita que o utilizador seja redireccionado para uma página maliciosa.

Caso seja necessário, é pedido ao utilizador que se autentique no servidor OAuth. Após a autenticação, o utilizador é questionado sobre conceder autorização ao cliente para aceder ao recurso protegido. Após esta ser concedida, o utilizador é redireccionado para o URI de *callback* especificado anteriormente. No redireccionamento são incluídos dois parâmetros na resposta:

- `oauth_token`: parâmetro que foi adicionado no pedido;
- `oauth_verifier`: código de verificação disponibilizado pelo servidor (introduzido na versão 1.0a).

Exemplo de resposta:

```
GET app/?oauth_verifier=_dd5a3d91f995ff6b0b1d991bda99ddb18005f98527&oauth_token=
    _cf2abf72caf253c642dbbe7a9f9a8053315b1c0f77
Host: cliente.ua.pt
```

A solução descrita anteriormente pressupõe que a aplicação cliente é uma aplicação web. No caso de aplicação cliente ser uma aplicação móvel ou *desktop*, e caso esta aplicação não possua as mesmas capacidades de um *browser*, deverá ser indicado ao utilizador o valor do parâmetro `oauth_verifier` para que seja este a introduzir na aplicação cliente. A solução OAuth do *Twitter* é um caso onde esta técnica é aplicada [Twitter 2011].

4. Após a obtenção de credenciais temporárias no passo 1 e da autorização concedida pelo cliente no passo 2, é necessário trocar os *tokens* temporários do passo 1 por *tokens* de acesso ao recurso. É efectuado novamente um pedido por parte do cliente com todos os argumentos especificados no passo 2 (excepto o `oauth_callback`), sendo também necessário acrescentar os parâmetros `oauth_token` e `oauth_verifier`. Exemplo:

```
GET oauth/get_data HTTP/1.1
Host: identity.ua.pt
Params:
  oauth_consumer_key=_b0e742818dfe3e9d3cb287e131f24d219bebcd4a7e&
  oauth_nonce=w97DAhMSZXeSDv&
  oauth_signature=i2JvM3SWnqy%2B7RY%2FNnObGhpCexQ%3D&
  oauth_signature_method=HMAC-SHA1&
  oauth_timestamp=1336581272&
  oauth_token=_caefc2d3ca2a6c20597860a95fc74770a3b966a587&
  oauth_version=1.0a
```

Resposta do servidor OAuth:

```
HTTP/1.1 200 OK
Content-Type: application/x-www-form-urlencoded

oauth_token=_caefc2d3ca2a6c20597860a95fc74770a3b966a587&oauth_token_secret=
    _34aed2c269a2ebe83070da7c61526f7b2d97d08f0d
```

Na resposta são obtidos dois novos *tokens*: o `oauth_token` e o `oauth_token_secret`. Estes *tokens* permitem aceder aos recursos protegidos do utilizador por parte do cliente. A validade atribuída a estes *tokens* varia de servidor para servidor. Em alguns casos poderá nunca expirar.

5. Após todo o processo de autenticação, autorização e obtenção de *tokens* de acesso, o cliente pode aceder ao recurso protegido do utilizador. O acesso ao recurso é feito à semelhança da obtenção dos *request tokens* e *access tokens*. Devem ser incluídos os parâmetros `oauth_consumer_key`, `oauth_token` (*access token* obtido no passo 4), `oauth_signature_method`, `oauth_signature`, `oauth_timestamp`, `oauth_nonce` e `oauth_version` (opcional). Exemplo:

```
GET oauth/get_data HTTP/1.1
Host: identity.ua.pt
Params:
  oauth_consumer_key=_b0e742818dfe3e9d3cb287e131f24d219bebcd4a7e&
  oauth_nonce=w97DAhMSZXeSDv&
  oauth_signature=i2JvM3SWnqy%2B7RY%2FNnObGhpCexQ%3D&
  oauth_signature_method=HMAC-SHA1&
  oauth_timestamp=1336581272&
  oauth_token=_caefc2d3ca2a6c20597860a95fc74770a3b966a587&
  oauth_version=1.0a
```

A resposta a esta invocação por parte do cliente irá conter o recurso que o cliente deseja aceder.

Após concluído este processo, o cliente tem acesso ao recurso autorizado pelo utilizador por um período de tempo equivalente à validade do *access token*. Nesse período não é necessário repetir todo o processo atrás descrito: basta repetir o passo 5. No entanto, a autorização concedida pelo utilizador ao cliente não deverá ser definitiva. O servidor OAuth deverá implementar mecanismos de revogação de *tokens*. Consideremos que um recurso é partilhado por diversos clientes através de OAuth. Devido a vários motivos, tais como a modificação da informação contida no recurso ou confidencialidade da mesma, o utilizador poderá ter interesse em que o recurso não seja novamente acedido por terceiros, justificando assim a necessidade de revogação de autorizações concedidas.

Um outro ponto que está para além da versão 1.0a do protocolo OAuth consiste no acesso a vários recursos. Nesta versão é assumido que o utilizador concede autorização para aceder a qualquer recurso partilhado através de OAuth. Esta generalização de recursos limita bastante a utilização do OAuth, afectando a extensibilidade e escalabilidade do protocolo.

Na primeira versão do protocolo OAuth, a 1.0, foi detectada uma falha de segurança que permitia a um utilizador mal intencionado aceder aos dados de outro utilizador. Este ataque, denominado de ataque de fixação de sessão, consiste em ludibriar um utilizador a autorizar um `oauth_token` que foi atribuído a outro utilizador (atacante). Após o utilizador efectuar a autorização de acesso a recursos, o atacante tenta aceder ao recurso utilizando o mesmo *token* [Hammer 2009]. Esta situação foi resolvida na versão 1.0a do protocolo OAuth, com a introdução do parâmetro `oauth_verifier` no redireccionamento do utilizador, após este autorizar o acesso aos recursos.

2.3.2.2 OAuth 2.0

De modo a colmatar algumas das falhas detectadas na versão 1.0a do OAuth, está neste momento em desenvolvimento a versão 2.0 deste protocolo. Ainda em versão *draft* no IETF, algumas empresas, como o Facebook, Google, e Microsoft, já começaram a implementar esta nova verão. Um dos requisitos da versão 2.0 é a utilização de SSL para todas as comunicações que envolvam a geração de um *token*. Deste modo é possível diminuir consideravelmente a complexidade do sistema, visto que deixa de ser necessário gerar assinaturas para os pedidos. Outra alteração prende-se com o número de *tokens* necessários às transacções. Na versão 1.0a era necessário um par de *tokens* para gerar um pedido. Um dos *tokens* (descrito na secção Modo de Funcionamento (2.3.2.1) com o prefixo "_secret") era fundamental para gerar a assinatura. Nesta nova versão, em que já não são utilizadas assinaturas, passa a ser necessário apenas a utilização de um *token*. Uma outra alteração, esta ao nível dos recursos, permite que

o utilizador (dono dos recursos) especifique a que recursos o cliente pode ter acesso. Esta característica permite que o sistema sejam muito mais escalável em relação à versão 1.0, permitindo utilizar o OAuth para o acesso a diversos serviços por clientes distintos [draft-ietf-oauth-v2-26].

2.4 Service Oriented Architecture

2.5 Serviços Web

O W3C define serviço web como "um meio de obter interoperabilidade através de uma rede computacional entre diferentes aplicações de software, independentemente da sua plataforma ou *framework*" [Booth 2004]. A invocação de um serviço é definido pelo W3C através da seguinte figura:

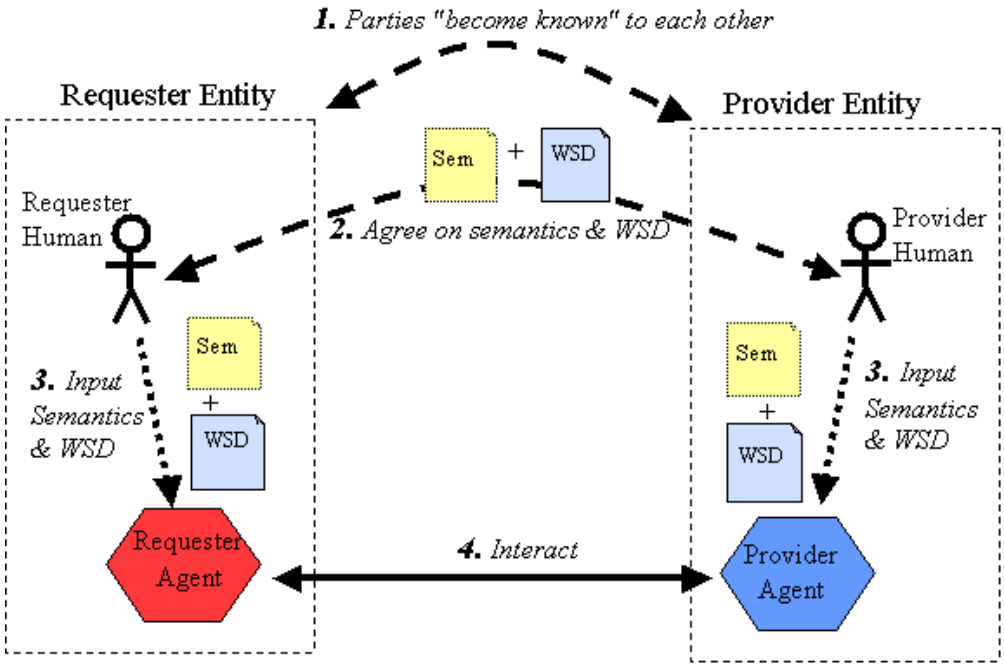


Figura 2.21: Processo de invocação de um serviço web. Fonte: [Booth 2004]

O processo descrito na figura 2.21 é composto por 4 passos genéricos:

1. As entidades fornecedor e consumidor tornam-se conhecidas uma da outra (ou pelo menos uma destas tem conhecimento sobre a outra).
2. É efectuado um acordo entre as entidades sobre a descrição e semântica do serviço que irá gerir a interacção entre ambos os agentes.
3. São desenvolvidas em ambas as entidades as implementações acordadas no passo 2.
4. Os agentes fornecedor e consumidor trocam mensagens que concretizam a interacção entre as partes.

No diagrama da figura 2.21 também é possível encontrar elementos cruciais à transacção de informação. Um **Agente** refere-se a software que envia e recebe mensagens, representando um fornecedor ou consumidor durante a interacção. O ***Web Service Description* (WSD)** deve conter informação suficiente sobre o formato da mensagem, tipos de dados, protocolos, localização do serviço e os mecanismos de interacção que devem ser esperados quando o serviço é invocado. A **semântica** refere-se à partilha do conhecimento dos conceitos geridos pelo serviço e os seus comportamentos. Tanto o WSD como a semântica devem criar um consenso entre fornecedor e consumidor. A **mensagem** é o modo de comunicação entre agentes. O seu formato é definido no WSD e é constituído por um cabeçalho, que contém metadados, e por um corpo, que contém a mensagem em si.

Mais especificamente, um serviço web pode ser descrito como uma aplicação de software identificada por um URI, cujas interfaces sejam capazes de serem definidas, descritas e descobertas como artefactos XML. Um serviço web deve suportar interacções directas com outros agentes através de mensagens XML, trocadas através de protocolos web [Austin 2002].

Nas secções seguintes serão analisados as arquitecturas mais comuns para a implementação de serviços web: a *Simple Object Access Protocol* (SOAP) e a *Representational State Transfer* (REST).

2.5.1 Simple Object Access Protocol

O SOAP é um protocolo de comunicações para a troca de informação num ambiente descentralizado e distribuído [Box 2000]. Criado em 1998, este protocolo utiliza XML como linguagem para a representação das mensagens. As mensagens trocadas através deste protocolo são encapsuladas numa estrutura XML denominada de envelope, que também pode conter metadados sobre a mensagem.

Figura 2.22: Estrutura de uma mensagem SOAP

O cabeçalho SOAP permite a inserção de metadados específicos da aplicação, tais como autenticação, definições da transacção, informações de pagamento, etc. Este cabeçalho é opcional. Caso esteja definido, o cabeçalho SOAP pode ser lido e modificado a cada iteração da mensagem SOAP até chegar

ao destino.

Exemplo de uma mensagem SOAP [Box 2000]:

```
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
  <SOAP-ENV:Header>
     <t:Transaction
      xmlns:t="some-URI"
      xsi:type="xsd:int" mustUnderstand="1">
        5
     </t:Transaction>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
     <m:GetLastTradePriceResponse
      xmlns:m="Some-URI">
        <Price>34.5</Price>
     </m:GetLastTradePriceResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Normalmente o protocolo SOAP é usado em conjunto com o *Web Service Definition Language* (WSDL). O WSDL é uma linguagem em formato XML utilizada para descrever serviços web e as suas interfaces. Além do WSDL, também é utilizada a *framework Universal Description Discovery and Integration* (UDDI), que tem como objectivo descrever e permitir a descoberta de serviços web.

Vantagens

- Protocolo heterogéneo: é independente da plataforma, sistema operativo e linguagem de programação.
- É baseado em XML, permitindo uma fácil leitura.
- Não é depende do protocolo de transporte.
- Interoperabilidade.

Desvantagens

- Tamanho elevado dos pacotes SOAP, aumentando o *overhead*.
- Segurança. Os pacotes SOAP são transmitidos em texto. Será necessário transmitir os pacotes através de uma linha segura (com SSL ou *Transport Layer Security* (TLS)). Também pode ser implementado WSS, que será documentado mais à frente nesta dissertação.

2.5.2 Representational State Transfer

A arquitectura *Representational State Transfer* (REST), descrita pela primeira vez por Roy Fielding em 2000, resume-se a um conjunto de princípios arquitecturais utilizados para o desenvolvimento de serviços web RESTful. [Fielding 2000]. Estes princípios podem ser resumidos em cinco pontos: a utilização de

um **protocolo cliente-servidor que trate cada pedido como uma transacção independente**, uma **interface uniforme entre componentes**, a implementação de ***cache***, a **divisão do sistema por camadas** e uma **sintaxe universal para o endereçamento**.

As aplicações web são particionadas em dois componentes, cliente e servidor, onde a componente servidor providencia o serviço à componente cliente. O caso mais flagrante deste sistema pode ser representado por um *browser*, como cliente, e um servidor Apache (HTTP), como servidor, comunicando através de um protocolo *stateless*, como o HTTP. Um protocolo *stateless* pode ser definido por um método de comunicação onde as mensagens são independentes. Isto é, o processamento de uma mensagem *X* não depende da mensagem *X-1* ou *X-2*. Esta característica permite a escalabilidade do servidor, visto que pode libertar recursos no final de cada pedido.

Fielding considera a utilização de uma interface uniforme entre componentes como sendo fundamental na arquitectura REST. Ao generalizar as interfaces, a arquitectura do sistema é simplificada e a visibilidade das interacções é melhorada. No entanto, esta generalização poderá diminuir a eficiência do sistema, devido à aplicação não poder transmitir a informação num formato que lhe seja nativo.

Por forma a aumentar a eficiência da rede, deverá ser aplicada a utilização de *cache*. Este princípio sugere que na resposta a um pedido deverá ser definido, explícita ou implicitamente, se está sujeito à aplicação de *cache*. Quando é carregada uma página web num *browser*, poderá ser definido através do cabeçalho HTTP da resposta se a página web esta sujeita a *cache*. Caso esteja, e se a página voltar a ser acedida pelo *browser*, é carregada uma cópia guardada localmente [Huston 1999]. Ao ser adicionada *cache* poderão ser eliminadas por completo algumas interacções, melhorando a eficiência, escalabilidade e latência de acesso a recursos pressentida pelo utilizador.

Um dos princípios arquitecturais do REST é a divisão do sistema por camadas. Ao restringir a visibilidade de um sistema a uma única camada, é possível delimitar a complexidade do sistema assim como promover a independência de cada uma das camadas. Este tipo de divisão também torna o sistema mais robusto e resistente a erros.

Na arquitectura REST, qualquer tipo de informação disponibilizada por uma interface poderá ser representada como um recurso. Estes recursos devem ser identificados por um identificador (endereço) único e global denominado de URI. Um URI é definido como uma sequência de caracteres que identificam um recurso físico ou abstracto [RFC 3986]. Em conjunto com um método (verbo) HTTP (GET, POST, PUT ou DELETE), o cliente consegue obter uma representação do recurso acedido ou manipular o mesmo. Os serviços web REST devem implementar um interface *Create, Retrieve, Update and Delete* (CRUD) para aceder ao recurso.

Método HTTP	CRUD	Descrição
`POST`	*Create* (criar)	Cria um novo recurso.
`GET`	*Retrieve* (obter)	Obtém a representação de um recurso
`PUT`	*Update* (actualizar)	Actualiza um recurso
`DELETE`	*Delete* (apagar)	Apaga um recurso

Tabela 2.3: Correspondência entre CRUD e os métodos HTTP

Normalmente um serviço REST é implementado sobre o protocolo HTTP ou *HTTP Secure* (HTTPS). Através do campo `Content-type` do protocolo HTTP é possível definir o formato do recurso a ser devolvido através de *Multipurpose Internet Mail Extensions* (MIME). Definidos inicialmente para o protocolo *Simple Mail Transfer Protocol* (SMTP) [RFC 2045], os tipos MIME foram expandidos para outros protocolos, sendo um deles o HTTP. Alguns tipos MIME:

- Texto simples: `text/plain`
- XML: `text/xml`
- CSV: `text/csv`
- HTML: `text/html`
- JSON: `application/json`
- JSONP: `application/javascript`
- *Webfeeds*:
  - Atom: `application/atom+xml`
  - RSS: `application/rss+xml`
- SOAP: `application/soap+xml`

Exemplo da invocação de um serviço REST:

```
1 GET http://services.web.ua.pt/sas/ementas HTTP/1.1
2 Host: services.web.ua.pt
```

O serviço presente no URI `http://services.web.ua.pt/sas/ementas` é invocado através do método HTTP GET (linha 1).

Resposta da serviço REST:

```
1  HTTP/1.1 200 OK
2  Cache-Control: max-age=82900, must-revalidate
3  Content-Length: 3018
4  Content-Type: text/xml
5  Etag: "6d7d80ded4d6a0ffb759f12d90f75cc9"
6
7  <?xml version="1.0" encoding="UTF-8"?>
8  <result request="/sas/ementas" request_timestamp="1336435100">
9    <menus zone="santiago" type="day">
10     <menu canteen="Refeitório de Santiago" meal="Almoço" date="Tue, 08 May 2012 00:58:20
           +0100" weekday="Tuesday" weekdayNr="2" disabled="0">
```

```
11      <items>
12        <item name="Sopa">Creme de espinafres</item>
13        <item name="Prato normal carne"/>
14        <item name="Prato normal peixe">Filetes de pescada com arroz de legumes</item>
15        <item name="Prato dieta">Pescada grelhada com arroz e couve lombarda</item>
16        <item name="Prato vegetariano"/>
17        <item name="Prato opção">Frango de churrasco com batata frita</item>
18        <item name="Salada">Buffet de saladas</item>
19        <item name="Diversos">Pão de mistura</item>
20        <item name="Sobremesa">Fruta da época</item>
21        <item name="Bebida">Água mineral</item>
22      </items>
23    </menu>
24    (...)
25  </menus>
26 </result>
```

No cabeçalho HTTP da resposta do serviço web, podemos verificar que o pedido ocorreu sem erros (código 200 - OK). Além disso é definida uma *cache* no *browser* de cerca de 23 horas (linha (02)). O conteúdo da resposta está no formato XML (linha (04)) e tem 3018 bytes de tamanho (linha (03)). Das linhas (07) à (26) podemos ver um excerto da resposta do serviço.

Vantagens

- A implementação de um serviço REST é bastante simples.
- Possibilidade de usar *cache* através do protocolo HTTP.
- Comunicações *stateless*.
- *Overhead* mínimo: o conteúdo da resposta de um serviço REST não contém cabeçalhos de controlo.

Desvantagens

- O único mecanismo de segurança passa pela utilização de SSL ou TLS.
- O URI de invocação de um serviço REST através do método HTTP GET está limitado a cerca de 4000 caracteres [Connolly 2002].
- Nem todos os standards dos serviços web são implementados pela arquitectura REST.

2.6 Web 2.0

Hoje em dia existe um novo conceito no desenvolvimento de páginas web denominado de Web 2.0. O termo Web 2.0 surgiu oficialmente em 2004, quando Dale Dougherty, vice-presidente da empresa literária O'Reilly Media Inc., discutia uma conferência sobre web. A grande questão passava por saber que tecnologias permitem apelidar um *website* como pertencente à Web 2.0 [O'Reilly 2005]. Embora não exista um consenso sobre a definição concreta deste termo [Zervaas 2008], o Web 2.0 pode ser especificado por um conjunto características associadas a este tipo de *websites*.

A **utilização de *standards*** é uma das características da Web 2.0. Na produção de um *website* deverão ser utilizadas linguagens de programação *standard*, tais como HTML ou *Cascading Style Sheets* (CSS). Tal permite o suporte por parte de diversos *browsers* e a correcta visualização dos conteúdos do website.

A **interacção com o utilizador** é um dos pontos mais importantes. Nos primórdios do WWW, um *website* estava dividido por várias páginas distintas. Para mudar de página, ou efectuar uma acção (adicionar, editar ou apagar um registo), era necessário carregar essa página com os parâmetros necessários. Com o aparecimento do AJAX este tipo de acções deixam de ser obrigatórias e passam a ser opcionais. Depois de carregada a página inicial, é possível carregar conteúdos ou executar acções assincronamente, melhorando significativamente a interacção com o utilizador.

No passado, um utilizador que consultasse notícias através da Internet era obrigado a aceder a cada *website* individualmente. Na Web 2.0 tal já não é necessário. Com a **partilha de dados através de *webfeeds* e serviços web** é possível difundir notícias através de outro meios para além do HTTP. Estas tecnologias permitem agregar estes conteúdos num único website.

As redes sociais são um dos conceitos mais importantes da Web 2.0 [Zervaas 2008]. Tal como as redes sociais, os *blogs* e fóruns promovem a comunicação entre pessoas e uma aproximação através da Internet. A **incorporação nas redes sociais** é um ponto fundamental para a partilha e divulgação, tanto do *website*, como dos seus conteúdos.

3 Serviços Web

Este capítulo é dedicado aos serviços web da UA. Numa primeira fase foi proposta a identificação e caracterização dos serviços web existentes. Após alguns contactos com elementos do Serviços de Tecnologias de Informação e Comunicação (STIC) da UA, foi possível identificar alguns serviços, de entre os quais se podem destacar o da biblioteca da UA, o do Code.UA assim como diversos *webfeeds*.

Após esta caracterização e identificação dos serviços, foi necessário proceder à avaliação de potenciais fontes de informação para as quais fosse relevante a criação de novos serviços. Uma das primeiras fontes de informação com bastante potencial para a implementação de uma API foram as senhas dos SAC.

Os SAC da UA estão presentes no edifício da reitoria desta instituição. Através de um pequeno dispensador de senhas, é possível retirar uma senha de uma dada categoria. Perto dos balcões de atendimento existe um ecrã com informação sobre a senha a ser atendida, para cada uma das categorias. Esta informação tem bastante potencial para poder ser difundida através de um serviço web, permitindo a qualquer pessoa saber quando se deve dirigir aos balcões dos SAC, sem ter a necessidade de lá ficar à espera.

Um outro potencial serviço que foi identificado refere-se à informação sobre as ementas das cantinas e restaurantes universitários. Através de uma página dos Serviços de Acção Social da Universidade de Aveiro (SASUA), é possível obter informações sobre todos os menus disponíveis, tanto para o próprio dia como para a restante semana. Após alguns contactos com os Serviços de Informação do SASUA, verificou-se que a informação disponível na página é estática. Isto é, a informação disponível não é guardada em nenhuma base de dados, sendo a única hipótese de a obter através de uma técnica denominada *web scraping*.

Além dos serviços descritos anteriormente, que são baseados em fontes de informação já existentes, foi implementado um novo serviço denominado de *File Bucket*. Este serviço permite a um utilizador da UA armazenar ficheiros de pequenas dimensões num servidor remoto, ficando acessível publicamente. Este serviço também serviu de apoio ao portal APIn.

Para além dos potenciais serviços identificados anteriormente, cuja informação poderá ser considerada pública, existem também uma série de serviços de carácter privado. Após autenticação no PACO, o aluno tem acesso a diversas informações de cariz pessoal: os dados pessoais disponibilizados aquando da matrícula, o seu horário escolar, as turmas a que pertence assim como os sumários das aulas. Estes dados, embora sejam privados, são interessantes para o desenvolvimento de serviços. Para tal, será necessário implementar um mecanismo de acesso a estes serviços, controlável pelo detentor da informação, e que pressuponha uma prévia autorização de acesso. Este mecanismo é detalhado no capítulo Autenticação e Autorização (4).

3.1 Identificação de Serviços Web na UA

Nesta secção será feita uma descrição dos serviços web existentes na UA.

3.1.1 Biblioteca da UA

Um dos serviços já existentes refere-se às bibliotecas e mediateca da UA. Este serviço REST permite executar várias pesquisas, assim como apresentar informações detalhadas sobre os volumes pesquisados [Libris 2011]. O serviço está dividido em três operações: *Find*, *Present* e *Finddoc*.

A operação *Find* permite iniciar uma pesquisa numa base das bibliotecas da UA. Com vista a uma pesquisa mais eficiente, as bibliotecas/mediateca da UA foram divididas em bases: a biblioteca principal que está presente no *campus* de Santiago, a biblioteca do Instituto Superior de Contabilidade e Administração de Aveiro, a Escola Superior de Saúde, Núcleo de Documentação do Complexo Pedagógico da UA, a Mediateca, a Escola Superior de Tec. e Gestão de Agueda, o CRC/ESAN - Programa Aveiro Norte e o Centro de Documentação Europeia. Para além destas bases físicas, existem bases lógicas, que permitem a pesquisa por tipo de documento (monografias, partituras, documentos electrónicos, etc), assim como por fundos especiais (provas académicas da UA, trabalhos de docentes, fundo La-Roche, etc). Para uma pesquisa global em todas as categorias descritas anteriormente, estão disponíveis três bases: catálogo geral, bibliografia recomendada e autoridades. A invocação da operação *Find* permite obter um identificador da pesquisa, referenciado pelo elemento `set_number` da resposta XML, necessário para a invocação da operação *Present*. Ou seja, a operação *Find* não devolve uma listagem das publicações, mas uma referência para a próxima operação e o número de registos existentes.

É através da operação *Present* que são listadas informações detalhadas sobre as publicações. Como já foi referido, um dos parâmetros necessários para a invocação desta operação é o `set_number`. Cada publicação é identificada pelo valor do elemento `doc_number` da resposta XML. Este valor é necessário para a invocação da operação *Finddoc*.

Para obter os detalhes de uma publicação específica, é necessário invocar a operação *Finddoc*. Um dos parâmetros necessários nesta operação é o `doc_number`, que identifica a publicação a ser listada. Não existe qualquer diferença nos detalhes das publicações obtidos por esta operação e pela operação *Present*.

As interfaces destas três operações, os seus parâmetros e exemplos de invocações e respectivas respostas podem ser consultados no anexo Biblioteca da UA (A.1).

3.1.2 Code.UA

O Code.UA (`http://code.ua.pt/`) é uma plataforma web que permite gerir projectos e repositórios da UA. Após a criação de um novo projecto, é possível fazer o controlo do ciclo de vida do mesmo, assim como gerir inúmeros factores inerentes a este, dos quais se podem destacar a equipa do projecto, a

calendarização e repositório do projecto, definir uma *wiki* de ajuda e reportar erros. Esta plataforma assenta sobre a *framework* Redmine (`http://www.redmine.org/`). O Redmine dispõe de uma API REST que permite expor algumas operações [Lang 2010]. Embora estas operações estejam implementadas com uma interface CRUD, algumas funções estão desactivadas, nomeadamente a remoção de elementos. Do conjunto das operações disponíveis podem salientar-se a obtenção de ocorrências e notícias de um projecto. Uma análise mais detalhada das interfaces e exemplos das operações disponíveis estão descritos no anexo Code.UA (A.2).

3.1.3 Guia de Acesso Online

Através página principal do *website* da UA (`http://www.ua.pt/`), é possível encontrar diversa informação institucional, nomeadamente sobre os cursos e unidades curriculares. Essa informação não está introduzida de um modo estático no *website*: é proveniente de um serviço web REST de domínio público, que expõe diversas operações [Pereira 2011a]. Sendo apenas de leitura, estas operações permitem obter inúmeras informações, das quais se podem destacar os cursos disponíveis na UA e os respectivos planos curriculares, uma listagem dos departamentos e os vários graus académicos disponíveis. Quase todos as operações deste serviço permitem o retorno da informação em português e inglês. A descrição das interfaces e exemplos de invocações das operações deste serviço web podem ser encontradas no anexo Guia de Acesso (A.3).

3.1.4 Jornal Online UA

Na UA está disponível um serviço de informação online denominado UA Online. O UA Online (`http://uaonline.ua.pt/`) é constituído por diversas fontes de informação que podem ser agrupadas em duas categorias principais: o jornal e a agenda.

Além da divulgação de noticias online no endereço referido anteriormente, as notícias são também difundidas utilizando uma tecnologia web denominada *webfeeds*. Os *webfeeds* são ficheiros XML onde são sumariadas as notícias ou eventos relativos a uma determinada organização. No UA Online o formato utilizado para o XML é o RSS 2.0. Este método de divulgação é amplamente utilizado, principalmente por *websites* informativos, *blogs* e redes sociais [Hammersley 2005]. Torna-se assim possível obter remotamente actualizações de um dado *website* sem aceder directamente a este. Este serviço REST, à semelhança dos anteriores, expõe diversas operações das quais se podem destacar a obtenção das últimas notícias, dos eventos e *banners* institucionais. Permite igualmente uma pesquisa de notícias. As interfaces e parâmetros destas operações estão disponíveis no anexo Jornal Online UA (A.4).

3.1.5 PACO

Um utilizador (aluno ou docente) que aceda ao PACO tem acesso a várias informações, desde os seus dados pessoais ao horário escolar, turmas e respectivos sumários das turmas. Devido à utilização destes dados por outras aplicações, como o portal my.ua (`http://my.ua.pt/`) [Teixeira 2009], foi implementado

um serviço web. Contrariamente aos serviços web discutidos anteriormente, este serviço foi implementado recorrendo à arquitectura SOAP. A descrição das operações deste serviço web é feita através de um ficheiro WSDL, acessível através do endereço `https://paco.ua.pt/contactua/contactua.asmx?wsdl`. No anexo PACO (A.5) estão descritas algumas operações presentes no WSDL com relevância para esta dissertação, assim como exemplos de invocações das mesmas.

Devido à natureza privada da informação disponibilizada através deste serviço, o acesso às suas operações é conseguido por meio do protocolo OAuth.

3.2 Novos Serviços Web na UA

Os serviços descritos nesta secção foram implementados num servidor Linux, acessível através do endereço `http://services.web.ua.pt/`. Tendo em conta as vantagens dos serviços REST em relação aos serviços SOAP (ver Representational State Transfer (2.5.2)), optou-se por implementar estes serviços web na arquitectura REST.

Tendo em conta algumas características necessárias para a implementação de serviços REST, tais como a utilização de um protocolo cliente-servidor *stateless* e a divisão do sistema por camadas, optou-se pelo desenvolvimento dos serviços web na linguagem de programação PHP. O protocolo cliente-servidor utilizado foi o HTTP, suportado através de uma instância do servidor web Apache. De modo a facilitar a implementação dos novos serviços web, foi utilizada um software denominado Tonic. O Tonic (`http://peej.github.com/tonic/`) é uma *framework* desenvolvida em PHP que permite a implementação de serviços web REST. Esta *framework* conta com algumas características importantes para o desenvolvimento de serviços web RESTful, tais como a implementação de métodos distintos para cada verbo HTTP e a possibilidade de adicionar *cache* às invocações dos serviços.

Todos os serviços desenvolvidos para a UA mantêm uma cópia dos dados recolhidos em bases de dados MySQL.

3.2.1 Ementas

Os SASUA dispõem de uma página web (`http://www2.sas.ua.pt/site/temp/alim_ementas_V2.asp`) onde é possível consultar as ementas das cantinas e restaurantes universitários. Além de obter as ementas para o próprio dia, também é possível obter as ementas para a restante semana. Devido a esta informação ser de interesse académico, optou-se por criar uma API para aceder programaticamente a esta informação.

Como já foi referido no sumário deste capítulo, a informação é introduzida estaticamente no *website*. Isto significa que a única forma de obter esta informação é através de uma técnica denominada de *web scraping*. Segundo Malik e Rizvi, o *web scraping* "é um técnica para a extracção de dados de uma página HTML, utilizando aplicações específicas para manipulação e conversão da página web para outro formato" [Malik 2011]. Embora esta solução não seja de todo a ideal, a falta de uma infraestrutura informática para

a publicação das ementas por parte dos Serviços Técnicos do SASUA não permite adopção de outra técnica para a obtenção destes dados.

3.2.1.1 Serviço Ementas

URL `http://services.web.ua.pt/sas/ementas`

Método HTTP GET

Formato da Resposta XML / JSON

Argumentos:

- `place`: Indica a cantina/restaurante para o qual irá ser devolvida a ementa. Valores possíveis:
  - **`santiago`**: Cantina de Santiago (campus de Aveiro).
  - `estga`: Cantina da Escola Superior de Tecnologia e Gestão de Águeda.
  - `esan`: Cantina da Escola Superior Aveiro Norte.
  - `rest`: Restaurante Universitário (campus de Aveiro).
  - `all`: Todos os anteriores.
- `date`: Indica a data da ementa. Valores possíveis:
  - **`day`**: Devolve a ementa para o dia actual.
  - `week`: Devolve a ementa para a semana (Segunda-feira a Domingo).
  - `YYYY-MM-DD`: Devolve a ementa para um dia especifico, anterior à data de invocação do serviço.
- `format`: Define o formato da resposta. Valores possíveis:
  - **`xml`**: A resposta é devolvida em XML.
  - `json`: A resposta é devolvida em JSON.
  - `jsonp`: A resposta é devolvida em JSONP.
- `cb`: Caso o formato da resposta seja em JSONP, é também necessário definir a função de *callback*. Este parâmetro tem o intuito de indicar o nome da função.
- `help`: Permite obter uma breve descrição do serviço assim como uma ligação para a *wiki* do serviço no portal APIn (consultar Operação *help* (3.2.4.1)).
- `dump`: Através deste parâmetro é possível obter um ficheiro CSV com detalhes de todas as ementas guardadas (consultar Operação *dump* (3.2.4.2)).
- `licence`: Esta operação permite obter a licença aplicada a este serviço, no formato RDF (consultar Operação *licence* (3.2.4.3)).

Exemplo `http://services.web.ua.pt/sas/ementas?date=day&place=rest`

Resposta Consultar Ementas SASUA (A.6).

Nota: O argumento `format` não necessita de ser explicitamente declarado. Através do campo `Accept` do cabeçalho do pedido HTTP é possível definir o formato da resposta [RFC 2616]:

- `text/xml`: A resposta é devolvida em XML.
- `application/json`: A resposta é devolvida em JSON.
- `application/javascript`: A resposta é devolvida em JSONP. É necessário definir o nome da função através do argumento `cb`.

3.2.1.2 Análise ao *website* das Ementas dos SASUA

Após uma cuidada análise ao *website* das Ementas dos SASUA, verificou-se que todas as segundas-feiras o *website* é actualizado com as ementas para a semana. No entanto, houve situações em que as ementas do próprio dia são alteradas antes da abertura das cantinas, que acontece às 12h. Assim sendo, e para manter a base de dados deste serviço o mais actualizada possível, optou-se por criar um *script* que é executado todos os dias às 12h. O propósito deste *script* é o de guardar na base de dados deste serviço as ementas de todas as cantinas e restaurantes disponibilizadas através do *website*. Este *script* invoca o serviço das ementas (`http://services.web.ua.pt/sas/ementas?place=all&date=day`) por forma a obter as ementas do próprio dia para todas as cantinas. De seguida faz um *parse* à resposta em XML e introduz os dados na base de dados as ementas.

O *website* com as ementas para o próprio dia é actualizado às 0h de cada dia.

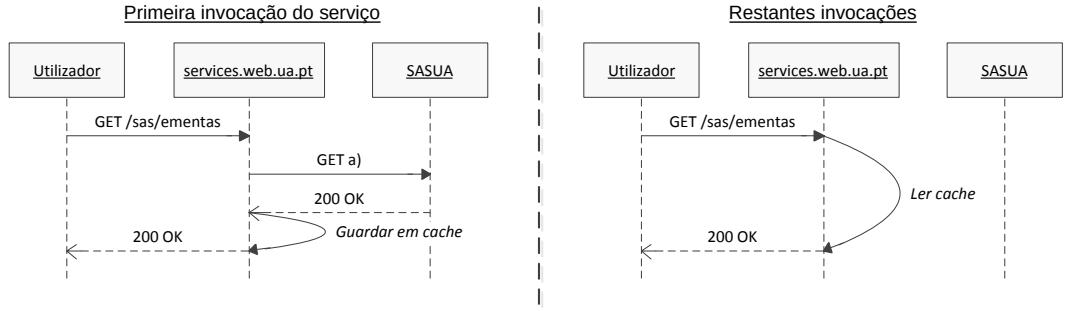
3.2.1.3 *Cache*

A invocação deste serviço web envolve a execução de três passos distintos: a chamada ao serviço presente no servidor services.web.ua.pt, o *web scraping* da página das ementas por parte do servidor services.web.ua.pt e a respectiva devolução dos dados nos dois passos anteriores (figura 3.1).

a) `http://www2.sas.ua.pt/site/temp/alim_ementas_V2.asp`

Figura 3.1: Comunicações envolvidas na invocação do serviço ementas

Como já foi referenciado anteriormente, a actualização do *website* das ementas é realizada diariamente à meia noite, conferindo aos dados presentes na página uma longevidade de 24 horas. Ou seja, quando este serviço é invocado pela primeira vez, existe a necessidade de obter a informação pretendida através da página dos SASUA. Para as restantes invocações, executadas no mesmo dia, deixa de existir a necessidade identificada anteriormente no caso de ser mantida uma cópia dos dados obtidos primordialmente. Em [Kambayashi 2001], os autores definem *web proxy cache* como um "armazenamento intermediário de objectos web, implementado num servidor *proxy*, e partilhado por um grupo de clientes". Esta definição encaixa-se no problema identificado anteriormente: armazenar temporariamente um objecto web num servidor intermediário (*proxy*), que pode ser partilhado por vários clientes (entidades que invocam o serviço web). Através deste mecanismo, e caso não ocorra nenhum erro, a página web das ementas é acedida, no máximo, uma vez por dia para cada pedido distinto. Os ficheiros de temporários de *cache* são removidos diariamente, assegurando uma utilização mínima de espaço no disco do servidor. A informação sobre as ementas é salvaguardada diariamente numa base de dados, ficando disponível para consulta através deste serviço web.



a) http://www2.sas.ua.pt/site/temp/alim_ementas_V2.asp

Figura 3.2: Diagramas de sequência da invocação do serviço ementas

Equivalentemente, um método de *cache* semelhante pode ser definido no lado do cliente. Roy Fielding define *cache* no lado do cliente como um dos princípios mais importantes da arquitectura REST, permitindo eliminar transacções de dados desnecessárias. Deste modo é melhorada a eficiência e escalabilidade do sistema, diminuindo a percepção da latência no acesso ao serviço por parte do cliente [Fielding 2000]. No RFC do protocolo HTTP/1.1 [RFC 2616], a *cache* no lado do cliente é definida com "o objectivo de eliminar a necessidade de enviar pedidos em muitos casos, assim como eliminar a necessidade de enviar respostas completas noutros casos". Enquanto que o primeiro objectivo permite reduzir o número de pedidos/respostas para muitas operações, o segundo diminui consideravelmente a largura de banda necessária para a resposta.

Para o primeiro caso, em que não existe uma nova comunicação com o servidor de origem do recurso, estão definidos dois métodos. Um desses métodos passa por definir um tempo em que o recurso expira. Através do campo `Expires` da resposta HTTP por parte do servidor, é possível definir a data para a qual o recurso é considerado válido. Esta data deverá ser especificada no formato definido pelo RFC 1123. Por exemplo, uma resposta HTTP que contenha a entrada `Expires: Sat, 30 Jul 2012 19:00:00 GMT` especifica ao *browser* que o recurso é válido até às 19 horas do dia 30 de Julho de 2012. Este campo também permite definir recursos que nunca expiram ao retornar uma data com um ano a mais em relação à data de acesso. Um outro método, introduzido na versão 1.1 do protocolo HTTP, passa por definir o campo `Cache-Control` da resposta à invocação do recurso. É através deste campo que são especificadas as directivas a serem respeitadas por todos os mecanismos de *cache* durante uma transacção. De entre estas directivas, existe uma denominada de `max-age`, que permite definir, em segundos, a validade do recurso. Em conjunto com a directiva `must-revalidate`, este mecanismo de *cache* permite assegurar que, passado o tempo definido através de `max-age`, o recurso é obrigatoriamente actualizado. Quando definido, o campo `Cache-Control` tem precedência sobre o campo `Expires`.

Para o segundo caso, em que o servidor é contactado mas o recurso ainda está actualizado no cliente, o mecanismo de *cache* é ligeiramente diferente. Quando o cliente invoca o serviço pela primeira vez, é definido o campo HTTP `ETag` da resposta. Este campo permite definir um valor para a entidade em causa, que servirá para a comparação com outras entidades do mesmo recurso. Em conjunto com o campo `ETag` é usado um outro campo HTTP, o `If-None-Match`. A função deste campo, definido no

cabeçalho de um pedido HTTP, é a de executar pedidos condicionais. O valor deste campo deverá conter todos os valores das entidades do recurso acedido, definidos pelo `ETag`. Cabe ao servidor de origem do recurso avaliar os valores presentes no `If-None-Match` e, caso coincidam com a entidade requerida, despoletar os mecanismos de indicação da validade do recurso presente no cliente.

No serviço das ementas foram aplicados dois mecanismos de *cache*: um com a intenção de eliminar pedidos desnecessários e outro para evitar o envio de respostas completas em algumas invocações. Quando um utilizador acede ao serviço, são executados três passos: verificação se o pedido já foi executado posteriormente, execução do serviço e atribuição de mecanismos de *cache*. No primeiro passo é comparado o campo `If-None-Match` do pedido com a *hash* do URI do recurso, gerada pelo algoritmo *Message-Digest Algorithm* (MD5). É através desta comparação que é identificada a eventual situação em que o recurso obtido posteriormente pelo cliente ainda é válido. A partir deste ponto, a execução do serviço é bifurcada. No caso da comparação ser positiva, é atribuído o código `304 Not Modified` à resposta, que indica ao *browser* do cliente que o recurso não foi modificado. Caso a comparação seja negativa, o recurso é invocado no lado do servidor (os dados são lidos da *cache* ou é executado *scrapping* à página web dos SASUA, como foi analisado), e é atribuído o código HTTP `200 OK` à resposta do serviço. Em ambas as situações, o campo `Cache-Control` é actualizado para a diferença entre o instante da invocação e a meia noite do dia seguinte ($T_{max-age} = T_{0_{X+1}} - T_{k_X}$, em que x representa o dia do pedido e k representa o tempo do pedido). Por fim o a resposta à invocação do serviço é devolvida ao cliente.

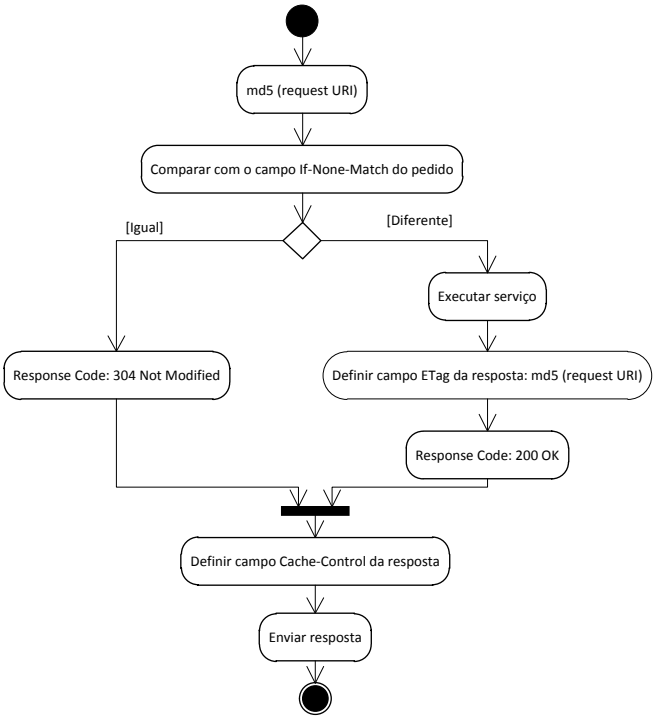


Figura 3.3: Diagrama de actividade da *cache* no serviço ementas

Os mecanismos de *cache* aplicados ao serviço das ementas, e descritos anteriormente, podem ser consultados através do diagrama de actividade da figura 3.3.

De modo a comprovar a eficácia dos mecanismos de *cache*, tanto no lado do servidor (*web proxy cache*) como no lado do cliente (através do *browser*), foram efectuados alguns testes de invocação do serviço das ementas. No primeiro teste forma desactivados todos os mecanismos de *cache*. Ou seja, o ciclo de invocação do serviço web é igual ao especificado na figura 3.1: o cliente invoca o recurso `http://services.web.ua.pt/sas/ementas`. De seguida, o servidor services.web.ua.pt invoca uma rotina de *web scraping* da página web das ementas, convertendo os dados para o formato XML. Por fim o XML é devolvido ao cliente. No segundo teste, é activado a *cache* no lado do servidor. Ou seja, após a primeira invocação do serviço das ementas, o serviço não volta a contactar o *website* das ementas do SASUA.

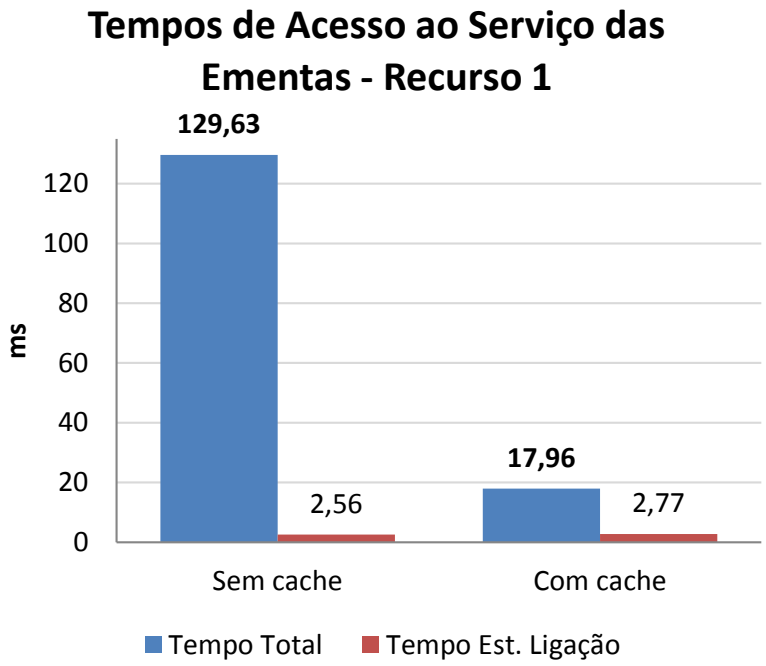


Figura 3.4: Tempos de acesso ao serviço das ementas, recurso 1

No gráfico presente na figura 3.4[1] é possível constatar a diferença entre a utilização, ou não, de *cache* no servidor. O tempo total representa a soma do tempo de resolução do *hostname* (na ordem dos 0,03 ms), o tempo de estabelecer a ligação e o tempo de resposta do serviço. Em ambos os casos, o tempo de resolução do *hostname* e o tempo de estabelecer a ligação são iguais, não influenciando o resultado final. No caso em que não foi utilizada *cache*, representada no lado esquerdo, o tempo médio de acesso ao recurso situa-se, aproximadamente, nos 130 milissegundos. Do lado direito podemos observar os resultados do teste utilizando *cache* no lado do servidor. O tempo médio de acesso ao recurso é de aproximadamente 18 milissegundos, conferindo uma eficácia na ordem dos 721% em relação ao caso anterior. É de salientar que no tempo total de acesso ao recurso com *cache*, o primeiro valor da amostra situa-se nos valores do tempo total de acesso sem *cache*. Como já foi referido, no primeiro acesso ao recurso é sempre contactado o *website* das ementas.

Para a confirmação dos resultados anteriores, foi efectuado outro teste com os mesmos padrões dos testes anteriores, sendo apenas trocado o recurso a ser acedido (`http://services.web.ua.pt/sas/e`

[1] Os tempos apresentados na figura 3.4 e na figura 3.5 representam a média aritmética de 100 amostras.

`mentas?place=all`). Enquanto que a resposta do primeiro recurso contém 2,9KB de tamanho, neste recurso o corpo da resposta contém 4,8KB.

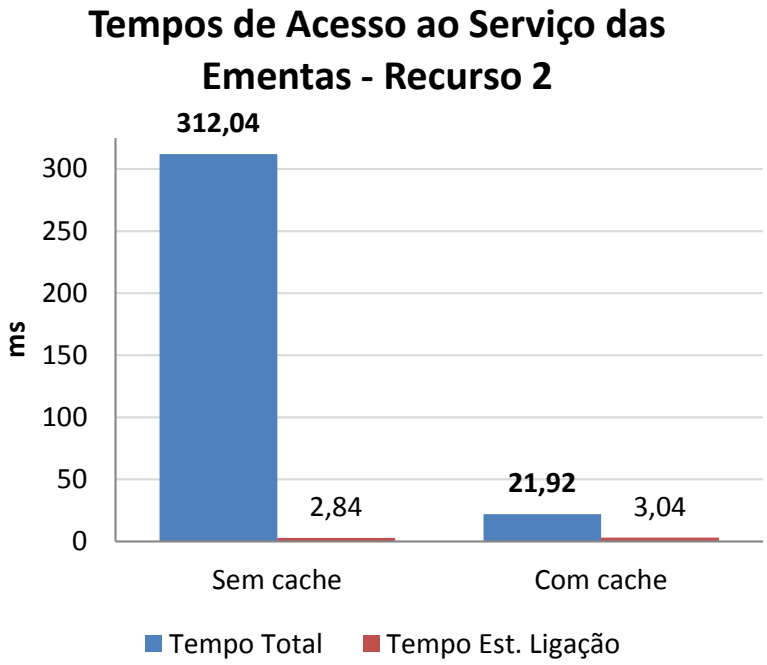


Figura 3.5: Tempos de acesso ao serviço das ementas, recurso 2

Neste segundo caso de estudo, a diferença no tempo total de acesso ao recurso com e sem *cache* situa-se na ordem dos 1423%. Em comparação com o caso de estudo da figura 3.4, o tempo total de acesso ao recurso (diferença de 1,9KB) aumentou para mais do dobro. No entanto, no caso de acesso com *cache*, a diferença foi de apenas 18%. No caso da figura 3.5, os tempos de resolução do *hostname* e de estabelecer a ligação também não afectam os resultados. A grande diferença entre este teste e o anterior deve-se, entre outros factores, ao tamanho da página obtida e, sobretudo, à necessidade de fazer *parse* a uma estrutura com quase o dobro do tamanho.

Tal como previsto, a implementação de uma *web proxy cache* no servidor reduz significativamente o tempo de acesso ao recurso. Além disso é evitada a comunicação com um servidor exterior, o que elimina a hipótese de propagação de erros, caso o servidor não responda em tempo útil.

3.2.2 Senhas SAC

A UA dispõe de um local de atendimento aos alunos na reitoria, denominado de Serviços Académicos (SAC). O atendimento nos SAC está dividido em várias categorias: alunos da UA, alunos de Erasmus, estágios internacionais, atendimento prioritário, etc. Para um aluno ser atendido, dirige-se a uma pequena máquina dispensadora de senhas e escolhe a categoria a que pertence. Este mecanismo está ligado a um computador que faz a gestão das senhas pedidas, dos balcões de atendimento e mostra o estado das senhas num ecrã.

O programa de gestão instalado nesse computador tem um componente de software que permite obter

informação das senhas através de uma *stream* no formato XML. Esse software, à escuta no porto 1500, permite estabelecer uma ligação através do protocolo *Transmission Control Protocol* (TCP) para o envio e recepção de *strings*. Por cada mensagem XML enviada, este software fica à escuta de uma *string* "ack". Se essa *string* não for recebida dentro de um prazo estabelecido, a ligação TCP é terminada [Figueira 2011]. Após a análise das mensagens XML devolvidas por este software, detectaram-se os seguintes padrões:

- A primeira mensagem recebida após a ligação é uma mensagem de configuração dos postos de atendimento, sem qualquer interesse para o desenvolvimento do serviço.
- A segunda mensagem recebida contém uma lista das categorias de senhas que já foram atendidas no dia, assim como o seu estado (activa ou suspensa). Também contém um mapeamento dos postos de atendimento.
- A cada hora é enviada uma mensagem idêntica à anterior, com uma actualização do estado das categorias.
- Sempre que um posto de atendimento chama uma nova senha, é enviada uma nova mensagem. Esta mensagem contém várias informações, tais como a categoria da senha (letra) (linha (05)), a sua descrição (linha (04)), o número de senhas em espera (linha (06)), o tempo médio de espera (linha (07)), o tempo médio de atendimento (linha (08)), o numero da senha (linha (11)) e o posto de atendimento (linha (13)). O número real do posto de atendimento é mapeado na segunda mensagem descrita anteriormente.

```
1  <MIOMessage>
2    <GEFListaChamada>
3      <chamada>
4        <nome>Lic. (1 ciclo), Mestrado (2 ciclo)</nome>
5        <letra>A</letra>
6        <nue>1</nue>
7        <tme>00:04:50</tme>
8        <tma>00:01:53</tma>
9        <descricao />
10       <numero>03</numero>
11       <numerosenha>41</numerosenha>
12       <servicoid>73</servicoid>
13       <postoid>117</postoid>
14     </chamada>
15   </GEFListaChamada>
16 </MIOMessage>
```

Por forma a comunicar com o software das senhas SAC, foi criado um *daemon* em JAVA. Um *daemon* é um software que se caracteriza por ser executado em segundo plano num sistema informático. O objectivo deste *daemon* é o de estabelecer uma ligação ao software descrito anteriormente, processar as mensagens recebidas e invocar um serviço REST.

3.2.2.1 *Daemon*: Primeira Versão

Numa primeira versão do *daemon*, este servia apenas como *proxy* para as mensagens recebidas. Como o acesso ao software das senhas SAC está bloqueado por endereço IP, os STIC deram permissão

à máquina de *staging* do projecto APIn para estabelecer uma ligação TCP com o software das senhas. Todo o processamento das mensagens é feito no lado do serviço web.

Figura 3.6: Diagrama da primeira versão do *daemon*

A implementação especificada na figura 3.6 revela vários problemas:

1. Problemas de ligação. Na necessidade de o *daemon* passar para outro servidor, numa rede distinta, poderiam surgir problemas na comunicação entre este e o serviço web através do porto 1500, principalmente devido ao uso de *firewalls*.
2. Sobrecarga do serviço web.

3.2.2.2 *Daemon*: Segunda Versão

Devido aos problemas detectados na primeira versão do *daemon*, optou-se por uma abordagem um pouco diferente. Nesta segunda versão, o *daemon* fica responsável pelo processamento das mensagens XML recebidas. Após o processamento, chama o serviço web (REST) com o método HTTP PUT.

Figura 3.7: Diagrama da segunda versão do *daemon*

Esta versão é a que está a funcionar actualmente. Tal como previsto, o *daemon* foi realojado no computador do PACO. Como nesta versão a comunicação entre o *daemon* e o serviço web é feito através do protocolo HTTP (porto 80), a comunicação deixou de ser um problema. O sobrecarga de processamento passou para o *daemon*. Sempre que o serviço web recebe uma invocação através do método HTTP PUT, os dados são colocados numa base de dados.

Este *daemon* corre como serviço na máquina dos SAC. É iniciado automaticamente quando o computador é ligado. Também produz um *log* para despistagem de eventuais erros, caso seja necessário.

3.2.2.3 Serviço Senhas SAC

URL `http://services.web.ua.pt/sac/senhas`

Método HTTP GET

Formato da Resposta XML / JSON

Argumentos:

- `letter`: Indica a letra da categoria das senhas. Valores possíveis: “`a`”, “`b`”, “`c`”, etc.
- `date`: Indica a data das senhas. Esta deverá ser representada no formato `YYYY-MM-DD`.

- `count`: Indica o número de senhas que devem ser mostradas.
- `format`: Define o formato da resposta. Valores possíveis:
  - **`xml`**: A resposta é devolvida em XML.
  - `json`: A resposta é devolvida em JSON.
  - `jsonp`: A resposta é devolvida em JSONP.
- `cb`: Caso o formato da resposta seja em JSONP, é também necessário definir a função de *callback*. Este parâmetro tem o intuito de indicar o nome da função.
- `help`: Permite obter uma breve descrição do serviço assim como uma ligação para a *wiki* do serviço no portal APIn (consultar Operação *help* (3.2.4.1)).
- `dump`: Através deste parâmetro é possível obter um ficheiro CSV com detalhes de todas as ementas guardadas (consultar Operação *dump* (3.2.4.2)).
- `licence`: Esta operação permite obter a licença aplicada a este serviço, no formato RDF (consultar Operação *licence* (3.2.4.3)).

Exemplo `http://services.web.ua.pt/sac/senhas?date=2012-01-27&count=2`

Resposta Consultar Senhas SAC (A.7).

Nota: O argumento `format` não necessita de ser explicitamente declarado. Através do campo `Accept` do cabeçalho do pedido HTTP é possível definir o formato da resposta [RFC 2616]:

- `text/xml`: A resposta é devolvida em XML.
- `application/json`: A resposta é devolvida em JSON.
- `application/javascript`: A resposta é devolvida em JSONP. É necessário definir o nome da função através do argumento `cb`.

Caso não sejam adicionados nenhuns parâmetros à invocação do serviço atrás descrito, é retornada a última senha gerada para cada categoria. Ao contrário do serviço das ementas dos SASUA, cuja periodicidade de actualização da informação é diária, neste serviço não existe uma definição temporal concreta para a validade do recurso, impossibilitando a aplicação de *cache*.

3.2.3 *File Bucket*

O serviço *File Bucket* foi concebido com o intuito de permitir o armazenamento e acesso a ficheiros de pequenas dimensões. A UA já dispõe de um serviço de armazenamento pessoal, denominado ARCA. No entanto esta solução não permite a partilha de ficheiros nem expõe uma interface para acesso programático. Surge assim a necessidade de implementação de um serviço com as características descritas no meio académico. No serviço *File Bucket* estão disponíveis duas operações: o armazenamento do ficheiro no servidor e a obtenção de um ficheiro.

Um dos requisitos necessários para o armazenamento de um novo ficheiro neste serviço é a identifi-

cação da entidade que efectua o pedido. Esta restrição permite não só limitar o *upload* de ficheiros a pessoas com conta na UA, como também identificar o proprietário do ficheiro. Para tal, o utilizador necessita de efectuar o armazenamento do ficheiro através do protocolo OAuth. Após ser feito o *upload* do ficheiro, é devolvido ao utilizador um URI público de acesso ao recurso que foi armazenado. Este acesso é livre, ou seja, não é necessário proceder qualquer tipo de autenticação para obter o ficheiro.

3.2.3.1 Primeira Versão

Numa primeira versão deste serviço, o servidor OAuth estava implementado no mesmo servidor onde estava o serviço, ambos com o mesmo domínio (services.web.ua.pt). Como foi estudado anteriormente, no passo 2 da secção Modo de Funcionamento (2.3.2.1) (do OAuth), a assinatura que é necessária incluir nos pedidos OAuth tem por base, entre outros factores, o URI base do pedido. Ou seja, qualquer pedido OAuth que fosse realizado para o servidor services.web.ua.pt seria considerado válido, independentemente do *endpoint* a que se destinasse. Isto é, uma invocação de um serviço web presente no servidor services.web.ua.pt que contivesse parâmetros OAuth válidos poderia ser utilizado para certificar a validade do pedido. No caso do *File Bucket*, é realizado um pedido OAuth directamente para o serviço (e não para o servidor OAuth) através do método HTTP POST, contendo todos os parâmetros necessários para que esta invocação seja válida. Adicionalmente, o ficheiro destinado a ser armazenado é também enviado para o servidor através do mesmo pedido, mas sem fazer parte da assinatura OAuth.

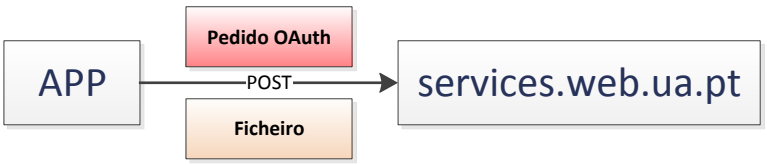


Figura 3.8: *Upload* de um ficheiro no serviço *File Bucket*, primeira versão

A principal vantagem deste método é o *upload* do ficheiro directamente para o servidor de destino. O envio do ficheiro para o serviço é realizado através do tipo MIME `multipart/form-data`. Definido no RFC 2388 [RFC 2388], o tipo `multipart/form-data` é dividido numa série de partes distintas. Cada parte contém vários cabeçalhos HTTP, tais como `Content-Disposition`, que permite definir o ficheiro a ser submetido, e o `Content-Type`, que permite definir o tipo MIME do ficheiro. Esta técnica também permite a submissão de vários ficheiros num único pedido. Com a utilização da linguagem de programação PHP na implementação deste serviço, é possível obter várias características do ficheiro armazenado. Na variável global do PHP `$_FILE`, que representa um *array*, são colocadas diversas definições do ficheiro transmitido, das quais se podem destacar o seu nome, tamanho e o tipo MIME associado. Estes dados representam toda a informação necessária para a correcta visualização do ficheiro através de um *browser*. Outro aspecto a considerar nesta implementação é a utilização do protocolo OAuth para acesso a um recurso de um modo invertido. Numa aproximação tradicional ao consumo de serviços através deste protocolo, o pedido é realizado por um cliente ao servidor OAuth, ficando à responsabilidade do último a invocação do serviço web pretendido. A resposta segue o percurso inverso até ao cliente. Na proposta apresentada anteriormente, o serviço é contactado directamente pelo cliente, ficando à responsabilidade do serviço a validação das credenciais OAuth.

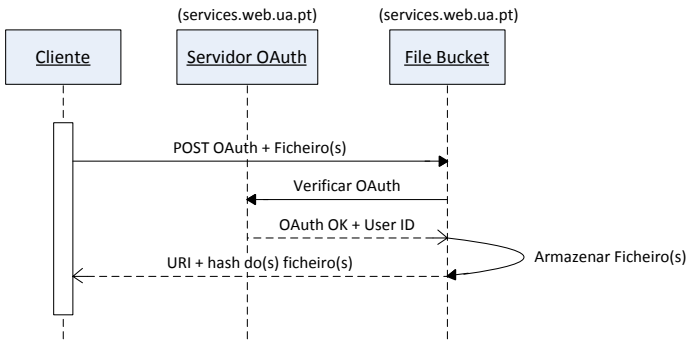


Figura 3.9: Diagrama de sequência do *upload* de ficheiro(s) para o serviço *File Bucket*, primeira versão

3.2.3.2 Segunda Versão

Após a migração dos servidores do projecto APIn para os STIC, foi alocado um domínio exclusivo para o acesso ao servidor OAuth: identity.ua.pt. Embora tanto o servidor OAuth como os serviços web continuem no mesmo servidor, o acesso ao OAuth é realizado através do domínio identity.ua.pt, enquanto que para aceder aos serviços web é utilizado o domínio services.web.ua.pt. Esta separação dos serviços invalida, à partida, a solução descrita anteriormente. Como a assinatura OAuth é baseada no URI do *endpoint* a ser invocado, esta é considerada inválida: o URI utilizado para invocar a verificação do OAuth tem como domínio services.web.ua.pt, enquanto que o servidor OAuth está alojado no domínio identity.ua.pt. Dada esta condicionalidade, foi necessária a implementação de outra solução para o *upload* de ficheiros.

Analisando alguns *service providers* existentes actualmente, como o *Twitter* ou *Flickr*, que expõem API's REST e o protocolo OAuth, facilmente encontramos operações dedicadas ao *upload* de imagens através de OAuth. Por exemplo, no caso do *Twitter*, o ficheiro deverá ser submetido através de *multipart*. Devido à utilização deste tipo MIME para submeter o ficheiro (tal como na primeira versão do *File Bucket*), este parâmetro não deverá ser considerado para a assinatura da mensagem OAuth [Twitter 2012]. Tal facto deve-se ao pré-processamento dos argumentos POST por parte do *web server*, invalidando a assinatura OAuth. O método utilizado pelo *Filickr* é em tudo semelhante.

O método implementado nesta segunda versão do *File Bucket* é em tudo semelhante à solução utilizada pelo *Flickr* e *Twitter*. O pedido OAuth deverá ser assinado com os parâmetros descritos no passo 6 da Protocolo OAuth 1.0a (4.1). No mesmo pedido deverão ser enviados os ficheiros pretendidos, utilizando a mesma técnica da primeira versão deste serviço, até um máximo de 5 ficheiros por invocação.

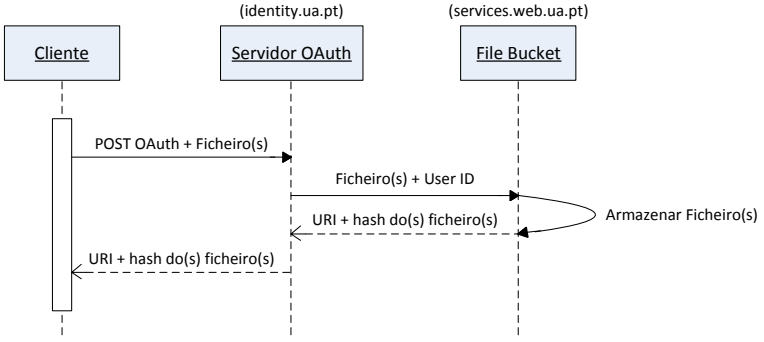


Figura 3.10: Diagrama de sequência do *upload* de ficheiro(s) para o serviço *File Bucket*, segunda versão

Em relação à primeira implementação deste serviço, é acrescentada uma sobrecarga ao servidor OAuth. Além de receber o(s) ficheiro(s), necessita de o(s) encaminhar para o servidor onde está a ser executado o *File Bucket*. Por outro lado este modelo permite uma validação prévia do pedido OAuth antes de proceder à invocação do pedido.

Para o correcto alojamento dos ficheiros no serviço são necessários vários parâmetros, tais como o *User ID* (neste caso o Utilizador Universal obtido através do IdP da UA) e o tipo MIME do ficheiro. Este último parâmetro é bastante importante: caso o utilizador aceda ao recurso através de um *browser*, a correcta definição do tipo MIME do ficheiro permite a visualização deste no próprio *browser* ou a execução de uma aplicação predefina para lidar com o ficheiro. Opcionalmente é possível definir um prazo de expiração do ficheiro, a partir do qual deixa de estar acessível, assim como o seu nome. Caso o nome não seja especificado, é atribuído um valor aleatório.

Este serviço foi desenhado para receber ficheiros até 5MB. Para a obtenção dos ficheiros basta invocar o URI devolvido pelo serviço através do método GET. Aquando do acesso ao recurso, é definida uma *cache* no lado do cliente com a duração de um mês. Como o ficheiro não pode ser alterado no serviço, não se coloca o problema da desactualização do ficheiro obtido previamente pelo cliente.

O *File Bucket* também permite apagar ficheiros presentes neste serviço. Como a cada recurso está associado um proprietário, a permissão para apagar um ficheiro não é pública, implicando a utilização de OAuth nesta operação. Dado este serviço web ser REST, deverá ser utilizado o método HTTP DELETE, em conjunto com o URI disponibilizado pelo serviço no momento em que foi submetido, para apagar o recurso.

3.2.3.3 Considerações Finais Sobre o *File Bucket*

Em ambas as implementações descritas para este serviço, existe uma potencial falha de segurança. Como foi referido anteriormente, a assinatura de uma invocação OAuth permite assegurar a integridade dos parâmetros transmitidos. No caso de a mensagem ser interceptada, uma entidade mal intencionada não tem capacidade para adulterar os parâmetros desta sem invalidar a assinatura. Na figura 3.8 é possível ver que o ficheiro a ser enviado para o serviço, embora acoplado no mesmo pedido POST, não faz parte dos parâmetros utilizados pelo OAuth para assinar o a mensagem. Na eventualidade da mensagem ser interceptada através de um ataque *man-in-the-middle*, por exemplo, é possível alterar o ficheiro a ser armazenado sem invalidar a mensagem OAuth. O pedido seria aceite, tanto pelo serviço *File Bucket*, como pelo servidor OAuth. Noutras implementações referidas anteriormente (*Twitter* e *Flickr*), o utilizador é alertado para a encriptação das mensagens através de SSL. Esta solução resolveria a vulnerabilidade de alteração do ficheiro. Durante a elaboração desta dissertação não houve a possibilidade de disponibilização de um certificado SSL pelos STIC, impossibilitando a encriptação no invocação do serviço.

Com foi descrito na primeira versão do *File Bucket*, os ficheiros são transmitidos em modo binário através do método HTTP POST. Segundo o RFC 4648, que especifica a codificação de dados nos formatos Base16, Base32 e Base64, "a codificação de dados no formato *base* é utilizada em muitas situações para armazenar ou transferir dados em ambientes restritos ao conjunto de dados US-ASCII" [RFC 4648]. Con-

siderando que, numa invocação de um serviço através do formato HTTP POST, o formato MIME atribuído à mensagem é de texto, tal permite o envio de um ficheiro, codificado em Base64, como parâmetro da invocação. Esta solução apresenta, à partida, uma vantagem: como o ficheiro faz parte dos parâmetros da invocação, no formato texto, é possível gerar uma assinatura que o inclua. Desta forma, e considerando o envio da mensagem num ambiente não seguro, a mensagem pode ser interceptada e o ficheiro extraído e descodificado, tal como acontece nas versões apresentadas anteriormente. No entanto não é possível modificar o ficheiro a ser enviado para o serviço: a sua alteração invalidaria a mensagem OAuth.

No entanto, a utilização de Base64 para codificação de dados acresce em $1/3$ o tamanho do ficheiro a ser transmitido. Como foi referido, a codificação *base* tem como pressuposto a conversão de dados para um formato de texto. Os caracteres disponíveis para a conversão são 64: `A-Z` (26), `a-z` (26), `0-1` (10), `+` e `/` (adicionalmente o caracter `=` é utilizado para *padding* da conversão) [RFC 4648]. Ou seja, é necessário converter octetos para conjuntos de 6 bits, aumentando o tamanho dos dados convertidos em 33%. Embora não esteja disponível literatura que apoie a implementação deste formato de transferência de dados através de OAuth, nos casos em que não esteja disponível um canal seguro, este deve ser uma opção a considerar.

Ao contrário das duas soluções identificadas anteriormente, este método apenas permite o envio de um ficheiro. Além disso, o utilizador necessita de especificar como parâmetros todos os dados referentes ao ficheiro (nome e tipo MIME).

Este serviço abre a possibilidade de desenvolvimento de inúmeras aplicações de armazenamento na *cloud*. Por exemplo, poderá ser implementado um portal em HTML5 para a partilha de documentos académicos. Um outro aspecto a considerar seria a alocação do espaço reservado ao serviço ARCA para o serviço *File Bucket*, com a possibilidade de criar um serviço semelhante ao *DropBox* ou *SugarSync*.

3.2.4 Operações *dump*, *help* e *licence*

Para além das operações que já foram descritas posteriormente, os serviços Ementas (3.2.1) e Senhas SAC (3.2.2) contam com mais três operações: a operação *help*, que permite que o consumidor obtenha ajuda sobre como utilizar o serviço, a operação *dump*, que permite que o consumidor obtenha todos os dados guardados desde o "epoch" de ambos os serviços (19 de Dezembro de 2011), e a operação *licence* que permite obter, no formato RDF, a licença de utilização aplicada ao serviço em causa.

3.2.4.1 Operação *help*

A operação `help` permite ao consumidor obter ajuda sobre como utilizar o serviço em causa. Ao invocar o serviço com o parâmetro `help`, o consumidor é redireccionado para a página inicial do servidor `http://services.web.ua.pt/`.

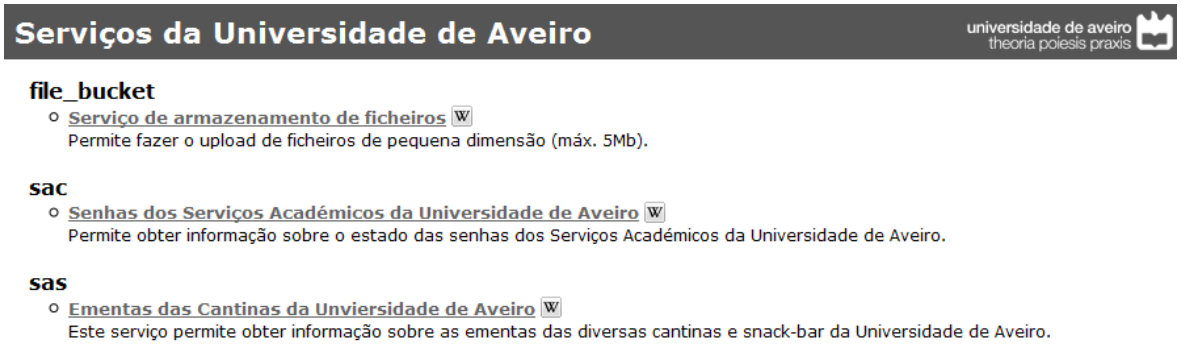


Figura 3.11: Página inicial do servidor `http://services.web.ua.pt/`

Como é possível ver na figura 3.11, ao lado direito do título do serviço é possível ver um ícone da *wiki*. Ao carregar sobre este ícone o consumidor é redireccionado para a *wiki* do serviço presente no portal APIn. As *wikis* destes serviços têm diversas informações pertinentes para o desenvolvimento de aplicações, tais como os parâmetros de cada serviço, exemplos de invocações e respectivas respostas assim como as possíveis mensagens de erro e a sua possível causa.

***Wiki* do Serviço Ementas** `http://api.web.ua.pt/pt/services/universidade_de_aveiro/ementas`

***Wiki* do Serviço Senhas SAC** `http://api.web.ua.pt/pt/services/universidade_de_aveiro/senhas_sac`

Wiki* do *File Bucket `http://api.web.ua.pt/pt/services/universidade_de_aveiro/file_bucket`

3.2.4.2 Operação *dump*

A operação `dump` permite obter todos os dados guardados das ementas desde o início da salvaguarda dos mesmos (19 de Dezembro de 2011). Os dados são retornados num ficheiro CSV. Para tal, basta invocar o serviço pretendido juntamente com o argumento `dump`. Devido à grande quantidade de dados a exportar, o CSV revela-se como a melhor opção para a divulgação da base de dados. Um outro formato a considerar seria o XML. No entanto, além deste formato ser muito mais redundante em comparação com o CSV, o processamento de documentos XML de grandes dimensões conduz à utilização excessiva do processador e requer bastante memória [Yoshida 2004].

Por forma a minimizar o esforço computacional exigido por esta operação, sempre que esta operação é requerida o ficheiro com os dados é guardado em *cache*, para a eventualidade de a operação ser novamente executada.

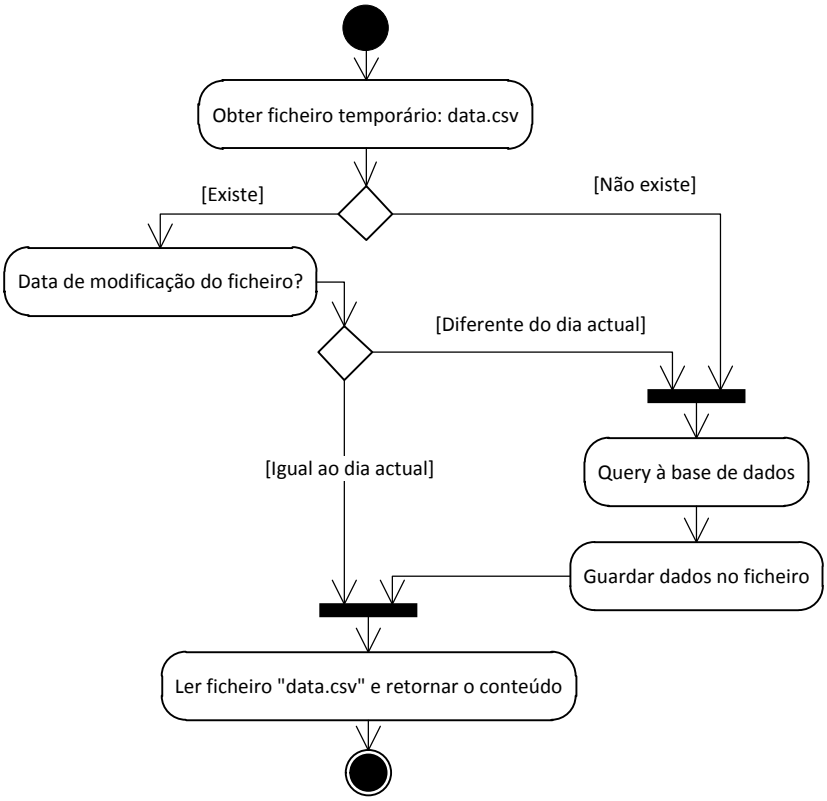


Figura 3.12: Diagrama de actividade da operação *dump*

3.2.4.3 Operação *licence*

A operação *licence* permite obter a licença aplicada aos serviços Ementas (3.2.1) e Senhas SAC (3.2.2). As licenças, devolvidas no formato RDF, permitem definir as permissões, obrigações e proibições associadas aos dados devolvidos pelo serviço. A licença atribuída a ambos os serviços foi a CC0. Esta licença, como já foi estudado anteriormente, foi criada pela CC com o objectivo de aplicar uma cada legal exclusivamente a dados e/ou conjuntos de dados.

As licenças atribuídas a ambos os serviços são meramente exemplificativas e podem ser consultadas no anexo Licenças dos Serviços Web (B).

3.3 Outras Fontes de Informação Identificadas

No seguimento da identificação e desenvolvimento de serviços web para a UA, foram surgindo outras fontes de informação com potencial para a implementação de interfaces de acesso programático. Uma das fontes identificadas está relacionada com os parques de estacionamento da UA. A universidade detém inúmeros parques para veículos que apenas estão acessíveis a pessoas devidamente autorizadas. O acesso aos mesmos é feito através de *Radio-Frequency Identification* (RFID). Seria interessante a disponibilização de dados estatísticos de utilização dos mesmos, o que permitiria, por exemplo, a criação de *mashups* com previsões sobre lotações, melhorando assim a sua gestão. Com a disponibilização do protocolo OAuth também seria interessante disponibilizar o histórico de acesso aos parques do utilizador

em causa.

Além dos parques de estacionamento, também os edifícios da UA estão equipados com dispositivos RFID nas suas entradas. Uma pessoa com acesso através do seu cartão da UA, poderá entrar nesse edifício a qualquer hora do dia. A disponibilização desses dados, de forma estatística, permitiria a criação de um serviço web sobre informações de acesso aos vários departamentos.Tal como no serviço dos parques de estacionamento, e numa perspectiva mais pessoal, poderia ser disponibilizado o histórico de acesso aos edifícios da UA por parte de um dado utilizador, abrindo a possibilidade de desenvolvimento de aplicações para o controlo de horários.

Além dos dispositivos RFID, alguns edifícios da UA estão equipados com dispositivos de medição do consumo energético. Tendo em conta a conjuntura económica actual e a necessidade de cortar nas despesas, com a implementação deste serviço surge a oportunidade do desenvolvimento de aplicações estatísticas que permitam gerir com mais exactidão os consumos energéticos no *campus*. Em parceria com o projecto CliM@UA, do Grupo de Meteorologia e Climatologia da UA, que permite obter previsões meteorológicas a curto e médio prazo, seria possível prever consumos energéticos para um período limitado de tempo no futuro.

No *website* `http://climetua.fis.ua.pt/legacy/main/current_monitor/cesamet.htm`, do Centro de Estudos do Ambiente e do Mar da UA, são disponibilizados os dados meteorológicos obtidos por uma estação presente no Departamento de Física da UA. Visto esta informação ser pública, seria interessante a implementação de um serviço web de acesso a estes dados. A disponibilização desta informação seria uma mais valia, com diversas áreas de aplicação: informação sobre o estado do tempo no *campus* da UA em páginas institucionais, caracterização meteorológica do *campus*, especificação de modelos matemáticos de previsão local, etc.

A UA também dispõe de uma lista de contactos gerais. Na página web `http://www.ua.pt/yellowPages.aspx` estão disponíveis os contactos de funcionários e docentes, incluindo o nome da pessoa, afiliação, gabinete, email e contacto telefónico (extensão da UA). Esta informação tem bastante potencial para a implementação de um serviço web. No entanto, e devido ao carácter sensível dos dados, este serviço apenas deveria ser disponibilizado através de OAuth.

Como já foi referido no serviço *File Bucket*, todos os alunos e docentes da UA têm acesso a um espaço pessoal de armazenamento de ficheiros. Através do portal my.ua (`https://my.ua.pt/`), e após autenticação, o utilizador tem acesso a uma lista de ficheiros que estão presentes nesse espaço pessoal. Há igualmente a possibilidade de execução de várias operações: armazenar, editar e eliminar ficheiros e directórios. Através de OAuth seria possível a implementação de um serviço web REST que permitisse uma interacção programática com este serviço da UA, assim como a criação de aplicações *desktop* para a representação dos ficheiros no disco do utilizador e sincronização com o serviço. Uma outra solução para a implementação de um serviço de partilha de ficheiros académicos seria a utilização do *File Bucket*, após alocação do espaço disponibilizado através do serviço ARCA.

4 Autenticação e Autorização

No capítulo Serviços Web (3) foram identificados alguns serviços da UA de carácter público e privado. Neste capítulo será retratada a implementação de um servidor OAuth para o acesso a esses serviços privados. O acesso a esses serviço implica a atribuição de uma autorização de invocação por parte do detentor da informação. Também deverá ser disponibilizada uma forma de revogar autorização concedidas. O servidor OAuth a seguir descrito foi implementado no servidor identity.ua.pt.

Será igualmente apresentada uma implementação do standard *Web Services Security* (WSS). Definido pela OASIS em 2004, o WSS é uma extensão do protocolo SOAP que permite a assinatura e encriptação das mensagens deste protocolo. Através de uma série de *standards*, como a Encriptação XML e Assinatura XML, o WSS permite adicionar uma camada de segurança para além de protocolos de transporte, como o HTTPS.

4.1 Protocolo OAuth 1.0a

Alguns dos serviços web identificados no capítulo anterior exportam informação que é considerada privada. Essa informação pessoal pertence a uma entidade e não pode ser partilhada com terceiros, sem a respectiva autorização por parte do seu detentor. Com base nesta necessidade foi implementado o protocolo OAuth. A escolha deste protocolo em relação ao XACML deve-se à simplicidade e objectividade do OAuth. O XACML é um protocolo de autorização que se baseia em políticas de acesso para permitir ou negar o acesso a um recurso por parte de uma entidade. Segundo [Anderson 2006], o "XACML não foi desenhado para ser uma linguagem de políticas de acesso para serviços web". Este facto deve-se a ainda não estar definido um perfil para a utilização do XACML para serviços web. Um outro aspecto a ter em conta para a escolha do protocolo a implementar foi a aplicabilidade do protocolo para o cenário em causa. Enquanto que o XAMCL não é dirigido para o problema em causa, o OAuth encaixa-se nos requerimentos necessários, tanto de segurança como de interoperabilidade e escalabilidade, para o acesso a serviços privados.

4.1.1 Modelo de Acesso a Recursos

O protocolo OAuth, na sua primeira especificação (1.0), implementa uma arquitectura orientada à aplicação [Leiba 2012]. Isto significa que um utilizador, ao conceder autorização a uma aplicação através de OAuth, esta passa a ter acesso a todos os serviços e/ou recursos disponíveis.

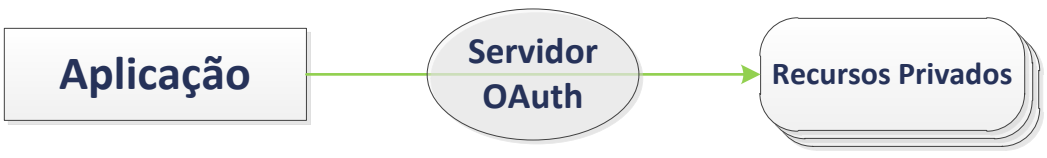


Figura 4.1: Modelo de acesso a recursos na versão 1.0(a) do OAuth

Este modelo representa um problema de segurança. Embora o acesso aos recursos esteja circunscrito à aplicação previamente autorizada, este modelo impossibilita a divisão dos recursos disponíveis por aplicação. Por exemplo, a aplicação X poderá ter acesso ao recurso A, B e C, enquanto que a aplicação Y apenas poderá aceder ao recurso A. Outro mecanismo que fica excluído através deste modelo são os níveis de acesso. Por exemplo, a aplicação X poderá apenas ter permissões de leitura, enquanto que a Y poderá ter de leitura e escrita.

Segundo a secção 4.8 do RFC do OAuth [RFC 5849], este protocolo não dispõe de nenhum método de qualificar os direitos de acesso concedidos a uma aplicação. Aquando da implementação deste protocolo, deverá ser tomada em consideração o desenvolvimento de mecanismos que permitam ao utilizador especificar a que recursos as aplicações podem aceder.

Com base nesta recomendação do RFC 5849, e devido à natureza dos serviços privados que são disponibilizados, foi necessária a adopção de um mecanismo que permita definir o acesso por parte de aplicações apenas a determinados recursos.

4.1.2 Modelo Proposto

De modo a colmatar a lacuna encontrada no modelo de acesso a recursos do OAuth, foi proposta a implementação de uma arquitectura orientada ao recurso. Neste caso, os recursos protegidos não são vistos como um todo, mas sim como peças singulares.

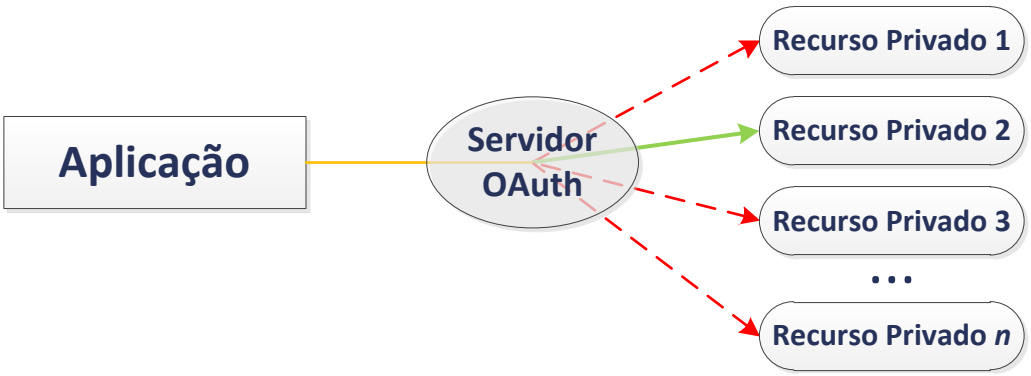


Figura 4.2: Modelo proposto de acesso a recursos para o protocolo OAuth

A implementação desta solução foi dividida em três fases. Numa primeira instância é necessário associar os serviços à aplicação. Isto é conseguido no registo da aplicação (ver Aplicações (4.2.1)). Nesta fase, o proprietário da aplicação a registar deve indicar quais os recursos que a aplicação necessita de aceder. Após a aplicação ser aprovada, é aplicado um mapeamento entre a aplicação e os recursos pretendidos. A segunda fase, e a mais importante [Leiba 2012], é a apresentação ao utilizador dos recursos que a aplicação pretende aceder. Na figura 4.4 está ilustrado um exemplo de um pedido de autorização. O utilizador é informado a que aplicação está a conceder autorização e a que recursos esta irá ter acesso. A última fase desta solução está relacionada com o acesso ao recurso por parte da aplicação. Após o utilizador conceder autorização, é necessário que a aplicação especifique o serviço ao qual deseja aceder.

Esta selecção é realizada através de um parâmetro extra na invocação OAuth (ver passo 6 da secção Implementação do Protocolo OAuth 1.0a (4.1.3)).

Este tipo de arquitectura é implementada na versão 2.0 do protocolo OAuth [draft-ietf-oauth-v2-26].

4.1.2.1 Mapeamento de Recursos no Servidor OAuth

De modo a aplicar a arquitectura definida na secção anterior, foi necessário efectuar uma mapeamento dos recursos disponíveis no lado do servidor. Neste mapeamento são definidos todos os atributos necessários para a obtenção do recurso ou invocação do serviço web em causa.

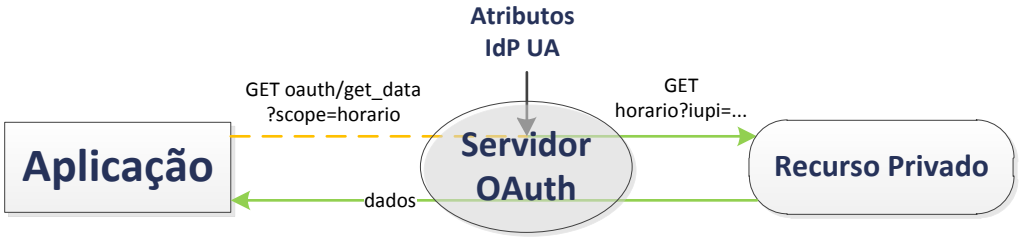


Figura 4.3: Mapeamento de serviços/recursos no protocolo OAuth

O **identificador do recurso** deve ser único e permite identificar inequivocamente o recurso em causa. É este identificador que deve ser introduzido no parâmetro `scope` quando o recurso é invocado. O **nome** e **descrição** devem reflectir toda a informação necessária para que o utilizador reconheça a informação que está prestes a partilhar.

Caso o recurso a aceder seja um serviço web, é necessário especificar os parâmetros que este necessita para a sua correcta invocação. Caso seja um serviço web SOAP, é necessário especificar o endereço do ficheiro WSDL, a função especificada no WSDL assim como o nome do elemento da resposta. No caso de ser um serviço web REST é necessário especificar o endereço do *endpoint* e o método HTTP para a invocação do serviço. Em ambas as situações é necessário definir os argumentos que os serviços web requerem. Os argumentos necessários poderão ser atributos do IdP da UA.

Caso o recurso seja um serviço web, fica a cargo do servidor OAuth a invocação do mesmo. A resposta obtida é então reencaminhada para a aplicação, através do formato mais conveniente (JSON ou XML).

4.1.3 Implementação do Protocolo OAuth 1.0a

De modo a expor de forma coerente a implementação do OAuth, é utilizado um exemplo de um serviço acessível através deste protocolo. Um dos serviços web privados identificados posteriormente é o dos horários escolares dos alunos. Este serviço, disponível no servidor do PACO, permite obter o horário escolar do aluno. Devido a este serviço conter informações pessoais do aluno, este tem o direito de autorizar, ou não, o acesso a esta informação por qualquer entidade.

Como já foi analisado anteriormente na secção Open Authorization Protocol (2.3.2), o protocolo OAuth permite o acesso a informação privada de uma entidade, sempre depois da última facultar autorização

para que a informação pretendida seja partilhada. De modo a explicar como foi implementado este protocolo na UA, tomemos em consideração o seguinte exemplo: a aplicação web myCalendar (cliente) pretende obter o horário na UA de um aluno (utilizador). O serviço que disponibiliza o horário está num servidor de recursos. O servidor OAuth está no servidor identity.ua.pt.

1. A aplicação myCalendar (cliente) regista-se no servidor OAuth da UA. Esta aplicação especificou que necessita de aceder a três serviços/recursos: o horário escolar, o utilizador universal e o nome do aluno. Também definiu o URI de *callback* como `http://mycalendar.ua.pt/oauth`. À aplicação são atribuídos dois *tokens* de identificação: um `consumer_key` e um `consumer_secret` (ver Aplicações (4.2.1)).

2. O utilizador acede à aplicação myCalendar. Nesta aplicação existe um botão que permite importar o horário do aluno na UA. O utilizador carrega neste botão.

3. O cliente contacta o servidor OAuth requerendo um *request token*. Para tal, é invocado o seguinte URL:

```
GET oauth/request_token HTTP/1.1
Host: identity.ua.pt
Params:
  oauth_consumer_key=_b0e742818dfe3e9d3cb287e131f24d219bebcd4a7e&
  oauth_nonce=pJNoLltekyqbeQ&
  oauth_signature=QqulrhY4%2B%2BR7pwoqjJj9diNTUf4%3D&
  oauth_signature_method=HMAC-SHA1&
  oauth_timestamp=1336864308&
  oauth_version=1.0a
```

   A resposta do servidor contém dois *tokens*: `oauth_token` e um `oauth_token_secret`, necessários para os próximos passos:

```
HTTP/1.1 200 OK
Content-Type: application/x-www-form-urlencoded

oauth_token=_36ea5de5d69f001d33546115aa2423514d7617e9c7&oauth_token_secret=
    _27d9f78a8742adb3d099aff7fecfb51d0f298b6a52&oauth_callback_confirmed=true
```

   Os *tokens* recebido devem ter um tempo limitado de validade, vistos que estas credenciais são apenas temporárias, não podendo ser utilizadas para aceder a recursos. Na configuração actual do servidor OAuth, estes *tokens* têm uma validade de 30 minutos.

4. O próximo passo é a autenticação e autorização por parte do utilizador. Deste modo, o cliente deverá redireccionar o utilizador para o seguinte endereço:

```
GET oauth/authorize HTTP/1.1
Host: identity.ua.pt
Params:
  oauth_token=_36ea5de5d69f001d33546115aa2423514d7617e9c7
```

   Caso o utilizador ainda não esteja autenticado, surgirá o formulário do IdP da UA para que este introduza as credenciais. De seguida é redireccionado para o URL anterior. Após a autenticação, o utilizador necessitará de conceder autorização para que o cliente aceda ao seu horário escolar:

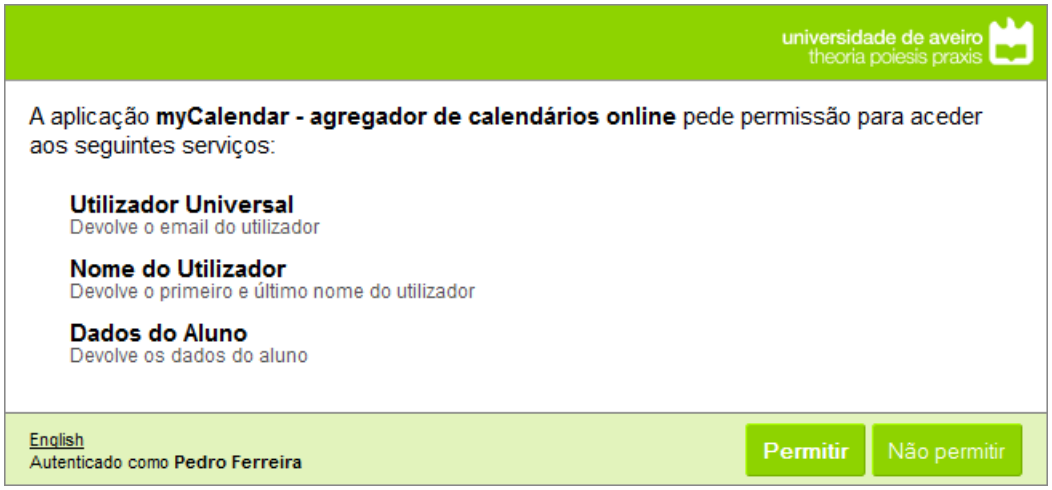


Figura 4.4: Formulário de autorização no protocolo OAuth

No formulário da figura 4.4, é indicado ao utilizador as informações a que o cliente deseja aceder. Caso o utilizador carregue no botão "Não permitir", este será redireccionado para a página do cliente, com indicação (para o cliente através de um parâmetro) que a operação não foi autorizada. Caso este autorize o acesso aos dados, será redireccionado para a página do cliente, sendo acrescentados ao URL dois parâmetros: `oauth_token` (com o mesmo valor da requisição) e `oauth_verifier`, que contém um código de verificação necessário para o próximo passo. O formulário apresentado na figura 4.4 também está disponível em Inglês.

Resposta do servidor após autorização do utilizador:

```
GET oauth/?oauth_verifier=_5e9d4e8b0d0d4d532e92c094f5d33f49cd9859ea9d&oauth_token=
    _36ea5de5d69f001d33546115aa2423514d7617e9c7
Host: mycalendar.ua.pt
```

5. Neste momento o cliente já detém autorização, por parte do utilizador, para aceder aos seus dados. No entanto o processo ainda não esta concluído. A próxima etapa será trocar o *request token* obtido no passo 3 por um *access token*. Para tal, o cliente invoca o seguinte URL:

```
GET oauth/access_token HTTP/1.1
Host: identity.ua.pt
Params:
  oauth_consumer_key=_b0e742818dfe3e9d3cb287e131f24d219bebcd4a7e&
  oauth_nonce=wKZaxRQvXUSpDn&
  oauth_signature=gHeLpZTH77eflWwQ21kQK3%2B13FU%3D&
  oauth_signature_method=HMAC-SHA1&
  oauth_timestamp=1336866004&
  oauth_token=_36ea5de5d69f001d33546115aa2423514d7617e9c7&
  oauth_verifier=_5e9d4e8b0d0d4d532e92c094f5d33f49cd9859ea9d&
  oauth_version=1.0a
```

Na resposta são devolvidos dois novos *tokens*: `oauth_token` e um `oauth_token_secret`:

```
HTTP/1.1 200 OK
Content-Type: application/x-www-form-urlencoded

oauth_token=_23b74c165e1ade11cf804f220699fcc463aadb6898&oauth_token_secret=
    _651fb19cffaa2cd3346018a6e9bacc7e14232a82f5
```

Com estes *tokens* o cliente já poderá aceder aos recursos privados do utilizador. Estes *tokens* têm uma validade de 30 dias.

6. Neste último passo, o cliente acede aos recursos do utilizador. Para tal, invoca o seguinte URL:

```
GET oauth/get_data HTTP/1.1
Host: identity.ua.pt
Params:
  oauth_consumer_key=_b0e742818dfe3e9d3cb287e131f24d219bebcd4a7e&
  oauth_nonce=1YOXEGTE7EuqGr&
  oauth_signature=gNdQniGtVYUCqT5TYaclEOuqQHI%3D&
  oauth_signature_method=HMAC-SHA1&
  oauth_timestamp=1336866452&
  oauth_token=_23b74c165e1ade11cf804f220699fcc463aadb6898&
  oauth_version=1.0a&
  scope=student_schedule
```

   Neste passo, o cliente deverá adicionar um parâmetro extra, que não faz parte do protocolo OAuth. Como já foi referido anteriormente, o OAuth 1.0a não faz a distinção de recursos. Foi, portanto, necessário a implementação de um mecanismo de acesso a recursos (ver Modelo Proposto (4.1.2)). Cada recurso privado é identificado no servidor OAuth por uma *string*. O cliente, ao invocar um recurso, necessita de identificá-lo através do parâmetro `scope`.

   Resposta da invocação do URL:

```
HTTP/1.1 200 OK
Content-Type: text/xml

<?xml version="1.0"?>
<diffgr:diffgram xmlns:msdata="urn:schemas-microsoft-com:xml-msdata" xmlns:diffgr="urn
    :schemas-microsoft-com:xml-diffgram-v1">
   <NewDataSet xmlns="">
      <ObterHorarioAluno diffgr:id="ObterHorarioAluno1" msdata:rowOrder="0">
         <CodDisciplina>43309</CodDisciplina>
         <NomeDisciplina>COMPUTAÇÃO RECONFIGURÁVEL</NomeDisciplina>
         <Dep>DET</Dep>
         <IDTurma>44338</IDTurma>
         <D_Turma>T1</D_Turma>
         <DiaSemana>2</DiaSemana>
         <HoraInicio>9.000000</HoraInicio>
         <duracao>2.000000</duracao>
         <Sala>ANF_V</Sala>
      </ObterHorarioAluno>
      <ObterHorarioAluno diffgr:id="ObterHorarioAluno2" msdata:rowOrder="1">
         <CodDisciplina>43309</CodDisciplina>
         <NomeDisciplina>COMPUTAÇÃO RECONFIGURÁVEL</NomeDisciplina>
         <Dep>DET</Dep>
         <IDTurma>44340</IDTurma>
         <D_Turma>P3</D_Turma>
         <DiaSemana>3</DiaSemana>
         <HoraInicio>14.000000</HoraInicio>
         <duracao>2.000000</duracao>
         <Sala>4.1.32</Sala>
      </ObterHorarioAluno>
   </NewDataSet>
</diffgr:diffgram>
```

Após este processo inicial, o cliente não necessita de voltar a repetir todos os passos quando quiser voltar a aceder à informação do utilizador. Basta apenas executar o passo 6.

Como é possível reparar pelos passos descritos anteriormente, este processo pressupõe que a aplicação cliente é uma aplicação web. E se a aplicação for uma aplicação móvel ou *desktop*? Esta problemática é discutida na próxima secção.

4.1.4 OAuth em Aplicações Móveis/Desktop

Para utilizar o serviço OAuth descrito anteriormente a partir de uma aplicação móvel ou *desktop*, é necessário ter em conta certas particularidades. Em primeiro lugar será assumido que esta aplicação não tem capacidade de replicar o funcionamento de um *browser*. É também importante referir que parte deste processo tem de ser realizado através de um *browser*, nomeadamente o processo 4 (autenticação e autorização).

De modo a utilizar o OAuth neste modelo, é necessário fazer algumas alterações nos parâmetros descritos na secção Protocolo OAuth 1.0a (4.1):

1. No passo 3 deve ser definido o seguinte parâmetro: `oauth_callback=oob`. Este parâmetro indica ao servidor OAuth que não existe um URI de *callback.*
2. Após a autenticação e autorização no passo 4, será indicado ao utilizador um código de verificação, idêntico ao da figura 4.5.

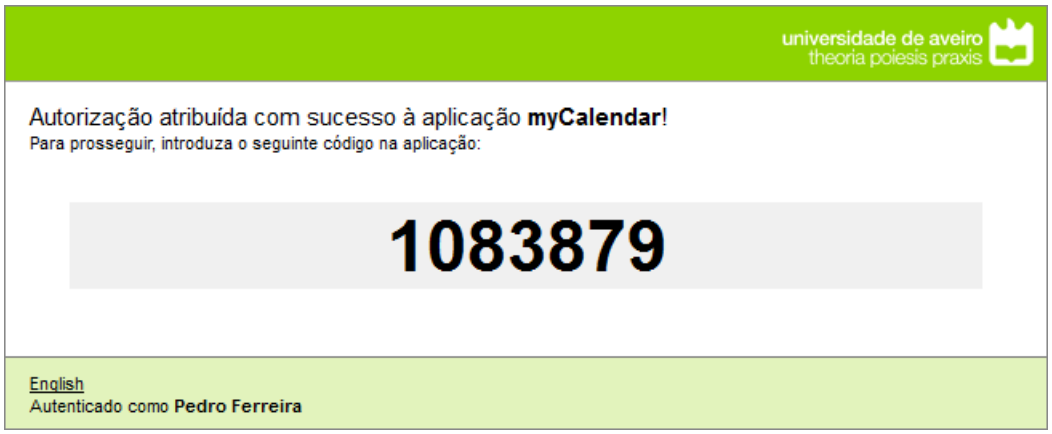


Figura 4.5: Código de verificação de um pedido no OAuth

   A aplicação cliente deverá disponibilizar um formulário onde o utilizador tenha a possibilidade de introduzir o código de verificação.

O restante processo é em tudo idêntico ao processo para uma aplicação web.

A *framework* SimpleSAMLphp, utilizada para a comunicação com o IdP da UA, contém um módulo de OAuth. Foi com base neste módulo que foi desenvolvido o servidor OAuth para a UA, utilizando a linguagem de programação PHP.

4.2 Portal de Gestão

Para a gestão do protocolo OAuth, foi necessária a criação de um portal. Este portal, à semelhança do portal APIn, está integrado com o IdP da UA. Existem dois tipos de utilizadores: administradores, que são responsáveis pela gestão do protocolo, e utilizadores normais. O *frontend* do portal foi desenvolvido em

HTML4, HTML5, JS (*framework* jQuery) e CSS1-3. O *backend* foi desenvolvido em PHP, com ligação a uma base de dados MySQL.

Figura 4.6: Página inicial do portal OAuth

Este portal esta dividido por duas categorias principais: aplicações e autorizações.

4.2.1 Aplicações

Nas aplicações, o utilizador tem acesso às suas aplicações. Também é através desta página que um utilizador deverá submeter a sua aplicação para utilização do protocolo OAuth (passo 1 da secção Protocolo OAuth 1.0a (4.1)). Após aprovação, serão atribuídos os *tokens* necessários à utilização do OAuth.

Figura 4.7: Aplicações no OAuth

Nesta página também é possível ter acesso a algumas estatísticas, tais como o número de *access*

tokens activas e o número de utilizadores que autorizaram a aplicação a aceder aos seus dados. A edição de uma aplicação previamente aprovada implica uma nova aprovação até as novas alterações serem aplicadas.

Cada utilizador conseguirá ver as aplicações que requisitou. Um administrador tem acesso a todas as aplicações, activas ou por aprovar, registadas na plataforma. Assim que uma aplicação é removida, todos os *tokens* por si gerados serão revogados. Também existe a possibilidade de um administrador suspender temporariamente a actividade de uma aplicação.

4.2.2 Autorizações

Na página das autorização é possível listar as autorizações concedidas a todas as aplicações. O utilizador tem a capacidade de revogar as autorizações quando quiser. Por defeito, cada autorização tem uma validade de 30 dias.

Figura 4.8: Autorizações no OAuth

Por cada aplicação é listada a data da última autorização concedida e a data de expiração.

4.3 Web Services Security

Numa arquitectura cliente-servidor, utilizada por serviços web, as questões de segurança não devem ser levadas de ânimo leve. Todas as decisões referentes à segurança devem ser tomadas tendo em conta as ameaças a que o sistema está exposto. Embora estejam disponíveis um vasto conjunto de tecnologias com o intuito de tornar os serviços web seguros, estes podem não ser os mais adequados para as necessidades de uma determinada entidade [Singhal 2007]. A melhor forma de proteger um serviço web contra ataques passa, numa primeira fase, por estudar as vulnerabilidades este pode estar exposto [Schwarz 2005]:

- **Modificação da Mensagem**: uma mensagem interceptada pode permitir a inserção, alteração ou remoção de dados contidos nesta, induzindo em erro o destinatário da mensagem.
- **Confidencialidade**: o conteúdo da mensagem pode estar exposto a outras pessoas que não o destinatário. Este ponto é bastante importante quando a mensagem transporta conteúdos sensíveis.

- **Falsificação de Mensagens**: um atacante poderá forjar mensagens e transmiti-las para um destinatário, simulando um remetente confiável.
- ***Man in the Middle***: este ataque passa por uma entidade mal intencionada interceptar a invocação e resposta de um serviço web. Este método permite a essa entidade ver os conteúdos das mensagens, assim como modificá-los.
- **Falsificação de Identidade**: uma mensagem é construída utilizando as credenciais de outra entidade. Os métodos de ataque e de prevenção são os mesmos que o ataque de modificação da mensagem.
- **Reenvio de Mensagem (ou de partes de mensagens)**: neste ataque, mensagens ou parte de mensagens que já foram enviadas são novamente submetidas.
- ***Denial of Service* (DoS)**: este ataque caracteriza-se por induzir o sistema a consumir recursos desproporcionalmente, impossibilitando de satisfazer novos pedidos.

A forma como estas vulnerabilidades são tratadas depende da própria instituição. Por exemplo, num serviço de informação sobre atendimento ao público, a confidencialidade é mínima. Mas, num serviço que trate de transferências bancárias para os empregados de uma organização, devem ser aplicadas todas as medidas de segurança possíveis. Independentemente do contexto e importância do serviço web, é fundamental perceber a que riscos este está sujeito e que tecnologias existem para minimizar a sua exposição a ataques.

4.3.1 Contextualização

O projecto APIn consistiu na implementação de um portal agregador de serviços web, proveniente de diversos fornecedores de serviços. Embora a maioria dos serviços disponibilizados no portal sejam de carácter público, existem alguns cuja informação é sensível ou privada. Surgiu, então, a questão sobre que tecnologias poderiam ser adoptadas para a disponibilização, de forma segura, dos serviços web em causa. Devido à estreita colaboração com a PTin neste projecto, o sistema myPersonas revelou-se como uma solução viável para o problema encontrado. O sistema myPersonas, da PTin, é um agregador de identidades. Este sistema permite uma gestão por parte do utilizador de várias identidades num cenário de multi-provedores de serviços. Neste cenário, o IdP da UA apresenta-se como um provedor de identidades para este sistema. Com a integração do myPersonas no portal APIn é possível não só o acesso a serviços web privados da UA, como também a integração com outros serviços e redes sociais. Por exemplo, no portal APIn estão disponíveis inúmeras *wikis* de serviços web. Por questões de segurança, apenas os proprietários do serviço ou administrados podem editar o conteúdo da *wiki*. Com o myPersonas seria possível a disponibilização de serviços de comentários, como o *Disqus* ou até o próprio *Facebook*, o que incentivaria à colaboração e melhoramento das *wikis* existentes no portal.

De modo a que várias entidades possam usufruir do sistema myPersonas, este expõe uma série de serviços web públicos que permitem a realização de diversas operações. Estes serviços SOAP são

utilizados por aplicações para a obtenção de dados referentes aos utilizadores do sistema. Por exemplo, uma aplicação poderá necessitar de obter todas as identidades de um utilizador através do serviço `get_identities(userID)`. Este serviço web é público, logo qualquer aplicação o poderia invocar. Dado este facto, poderão ser levantados dois problemas de segurança. O primeiro prende-se com a confidencialidade e privacidade dos dados transmitidos na invocação e resposta do serviço. O segundo problema de segurança esta relacionado com a identificação da entidade que invoca o serviço web. Aplicações distintas poderão ter níveis de acesso distintos. Surge assim a necessidade de implementação de um sistema de identificação da entidade invocadora do serviço.

Um método seguro para a transmissão de uma mensagem SOAP seria a utilização de encriptação na camada de transporte. Mecanismos como SSL ou TLS permitem encriptação ponto-a-ponto de uma ligação, assegurando confidencialidade, integridade da mensagem e autenticação [Bisel 2007]. No entanto, a utilização deste mecanismo de segurança poderá ser limitada por diversos motivos. Segundo [Chappell 2002], tanto os serviços web como SOAP devem ser independentes do protocolo de transporte, o que invalida a utilização deste mecanismo de segurança. Um outro aspecto a considerar prende-se com a possibilidade da intervenção de intermediários numa transacção SOAP. Estes intermediários poderão necessitar de remover ou inserir cabeçalhos na mensagem SOAP. Caso a mensagem esteja encriptada por SSL ou TLS, o intermediário não poderá modificar os cabeçalhos [Nordbotten 2009]. Por outro lado a utilização deste tipo de segurança obriga à obtenção de um certificado digital assinado, o que envolve custos avultados.

Um outro mecanismo a considerar para a transmissão segura de uma mensagem SOAP é o *Web Services Security*. Segundo [Lavarack 2010], o "WSS providencia segurança ponto-a-ponto a serviços web ao permitir a inclusão de mecanismos de segurança no SOAP, tais como assinatura e encriptação do XML, de modo a implementar uma *framework* que proporcione segurança em mensagens SOAP e que seja neutra em relação ao transporte". Este mecanismo de segurança permite a encriptação e assinatura do XML presente no elemento `body` de uma mensagem SOAP, conferindo confidencialidade, integridade, autenticação e interoperabilidade à mensagem [Nordbotten 2009].

4.3.2 Segurança ao Nível da Mensagem

O WSS implementa segurança no SOAP ao nível da mensagem [Singh 2004]. Isto significa que os elementos de segurança necessários são incluídos no cabeçalho da mensagem SOAP. Além disso, a segurança ao nível da mensagem permite aplicar mecanismos de segurança aos dados da mensagem, tais como encriptação e assinatura digital. Este tipo de mecanismo permite partilhar informação sobre os métodos, protocolos e procedimentos utilizados para tornar a mensagem segura, permitindo ao destinatário aplicar os processos necessários para a verificação e leitura da mensagem SOAP. A figura 4.9 ilustra como a segurança ao nível da mensagem é aplicada.

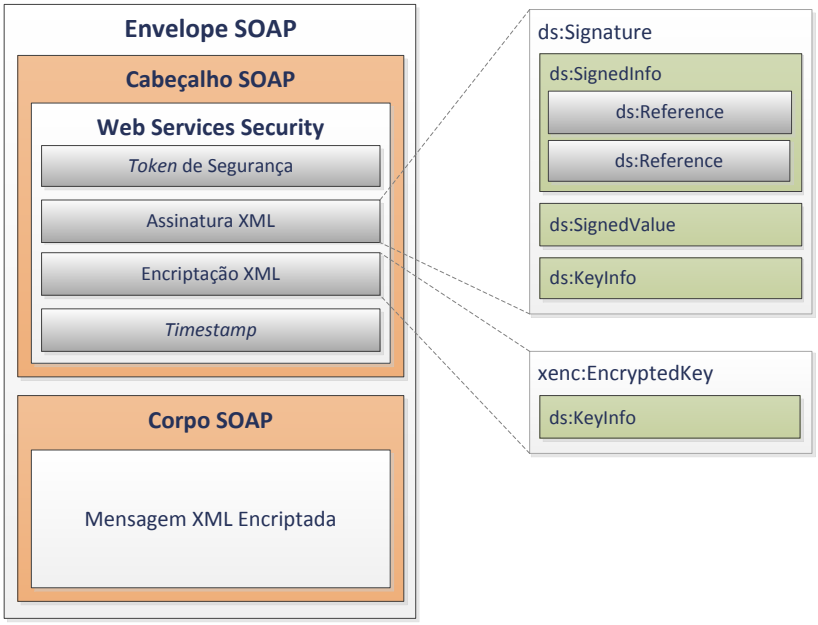


Figura 4.9: Estrutura de uma mensagem SOAP que implementa WSS

4.3.2.1 Encriptação do XML

A encriptação XML, definida pelo W3C, surgiu no final de 2002. Este mecanismo tem como objectivo a encriptação de dados e representação dos mesmos em XML. Os dados a serem encriptados podem incluir um documento XML, um elemento XML ou o conteúdo de um elemento XML. O resultado desta encriptação é um elemento XML `EncryptedData`, que contém ou identifica os dados cifrados [Imamura 2002]. Para a encriptação do XML é necessário a utilização de uma chave. Existem três tipos de chaves que podem ser utilizadas: chaves simétricas, chaves assimétricas e certificados X.509.

A encriptação através de **chaves simétricas** pressupõe que tanto o cliente, como o servidor partilham a mesma chave que é usada para a encriptação e desencriptação das mensagens. A grande desvantagem desta técnica consiste na necessidade de partilha da chave entre entidades, que muitas vezes tem de ser feita pela Internet.

A encriptação utilizando **chaves assimétricas** determina que o servidor deve partilha com o(s) cliente(s) uma chave pública que permita ao cliente encriptar a mensagem a ser enviada. Apenas o detentor da chave privada (servidor) consegue desencriptar a mensagem. Este mecanismo normalmente é utilizado para a partilha de uma chave simétrica.

Os **certificados X.509**, definidos pelo IETF, consistem num formato de partilha de informação, protegidos por uma assinatura digital. Existem 6 campos que devem ser preenchidos: o número de série, o identificador do algoritmo usado para assinar o certificado, o nome da entidade que gerou o certificado, a sua validade, a sua chave pública e o nome do detentor. A encriptação do XML através de certificados X.509 é idêntica às chaves assimétricas: o cliente utiliza a chave pública do certificado do servidor para encriptar os dados. O servidor usa a chave privada para os desencriptar. Embora ambas as chaves,

pública e privada, estejam relacionadas matematicamente, uma não pode ser usada para derivar a outra [Kuhn 2001].

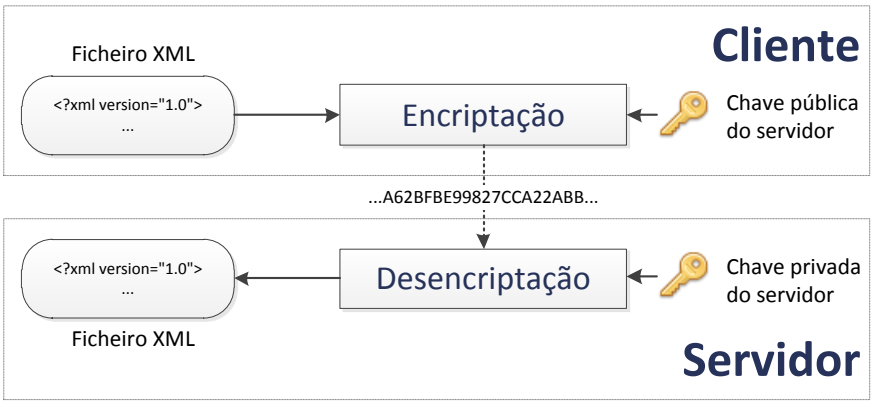


Figura 4.10: Encriptação/desencriptação através de chave pública-privada

O elemento `EncryptedKey`, presente no cabeçalho da mensagem SOAP, permite representar as chaves encriptadas.

Exemplo:

```
<soapenv:Envelope xmlns:ser="http://servico/" xmlns:soapenv="http://schemas.xmlsoap.org/
    soap/envelope/" xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
  <soapenv:Header>
    <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
        wssecurity-secext-1.0.xsd">
      <wsse:BinarySecurityToken
        EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-
            security-1.0#Base64Binary"
        ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile
            -1.0#X509v3"
        wsu:Id="D086A2FA2C762748F0133700530213694"
        xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility
            -1.0.xsd">
        MIIBoTCC(...)5kZH2jqJOw==
      </wsse:BinarySecurityToken>
      <xenc:EncryptedKey Id="EncKeyId-D086A2FA2C762748F0133700530213695">
        <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
        <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
          <wsse:SecurityTokenReference>
            <wsse:Reference
              URI="#D086A2FA2C762748F0133700530213694"
              ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-
                  profile-1.0#X509v3"/>
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
        <xenc:CipherData>
          <xenc:CipherValue>KqGTfK/ZZH(...)zZOxEp0=</xenc:CipherValue>
        </xenc:CipherData>
        <xenc:ReferenceList>
          <xenc:DataReference URI="#EncDataId-76"/>
        </xenc:ReferenceList>
      </xenc:EncryptedKey>
      (...)
    </wsse:Security>
  </soapenv:Header>

  <soapenv:Body wsu:Id="id-75" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis
      -200401-wss-wssecurity-utility-1.0.xsd">
    <xenc:EncryptedData Id="EncDataId-76" Type="http://www.w3.org/2001/04/xmlenc#Content">
      <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc"/>
      <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
```

```
35        <wsse:SecurityTokenReference xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis
              -200401-wss-wssecurity-secext-1.0.xsd">
36          <wsse:Reference URI="#EncKeyId-D086A2FA2C762748F0133700530213695"/>
37        </wsse:SecurityTokenReference>
38      </ds:KeyInfo>
39      <xenc:CipherData>
40        <xenc:CipherValue>+BeGR4rPmnRu(...)KKn6vs</xenc:CipherValue>
41      </xenc:CipherData>
42    </xenc:EncryptedData>
43  </soapenv:Body>
44 </soapenv:Envelope>
```

Linhas (02) - (29) Especificação do cabeçalho SOAP.

Linhas (03) - (28) Elemento de segurança definido no cabeçalho. Este elemento contém informação relacionada com a segurança da mensagem (encriptação, assinatura, *timestamp*, etc).

Linhas (04) - (10) Elemento que define o *token* de segurança da mensagem. Neste caso, representa um certificado X.509v3, codificado em Base64.

Linhas (11) - (26) No elemento `EncryptedKey` é especificada a chave usada para encriptar o corpo da mensagem. Visto esta chave ser simétrica, esta é encriptada pelo algoritmo definido na linha (12). nas linhas (13) a (19) é identificada a chave que é utilizada para a encriptação da chave simétrica. As linhas (20) a (22) especificam a chave simétrica encriptada. Finalmente, na linha (24) é referenciado o elemento que é encriptado (neste exemplo é o corpo da mensagem).

Linhas (31) - (43) Especificação do corpo da mensagem SOAP. O corpo da mensagem está encriptado e assinado.

Com a encriptação de mensagens é possível evitar muitas das vulnerabilidades atrás descritas. O conteúdo torna-se ilegível, prevenindo a visualização e alteração do mesmo.

4.3.2.2 Assinatura do XML

A assinatura de mensagens é outro método de segurança definido pelo WSS. Especificado como standard pelo W3C, as assinaturas de XML permitem assegurar a autenticidade e integridade das mensagens [Bartel 2008]. Este método segue os mesmos princípios da encriptação de mensagens: o cliente assina a mensagem com a sua chave privada, e o servidor verifica a identidade da mensagem através da chave pública do cliente (figura 4.11). Ao contrário da encriptação do XML, a assinatura da mensagem permite ver o seu conteúdo. No entanto, é impossível alterar o conteúdo da mensagem sem a invalidar.

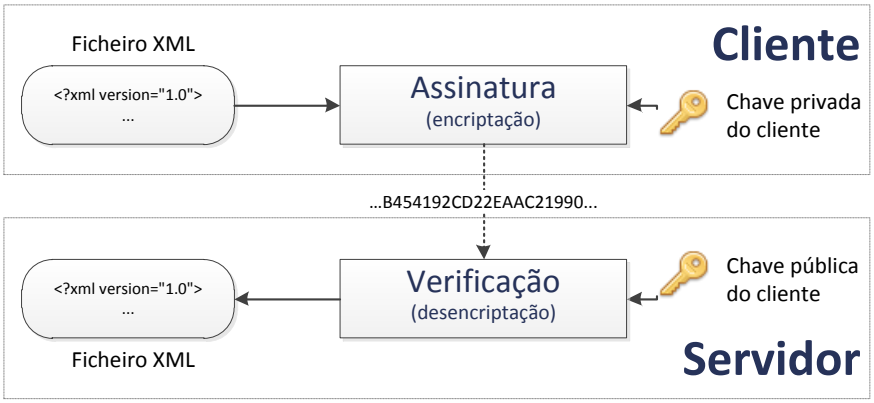


Figura 4.11: Assinatura/verificação através de chave pública-privada

Exemplo:

```
<soapenv:Envelope xmlns:ser="http://servico/" xmlns:soapenv="http://schemas.xmlsoap.org/
    soap/envelope/" xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
  <soapenv:Header>
    <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
        wssecurity-secext-1.0.xsd">
      (...)
      <wsse:BinarySecurityToken
       EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-
           security-1.0#Base64Binary"
       ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile
           -1.0#X509v3"
       wsu:Id="CertId-D086A2FA2C762748F0133700530211291"
       xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility
           -1.0.xsd">
        MIIBmT(...)9p3RLf4M=
      </wsse:BinarySecurityToken>
      <ds:Signature Id="Signature-74" xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
       <ds:SignedInfo>
        <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
        <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
        <ds:Reference URI="#Timestamp-73">
         <ds:Transforms>
          <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
         </ds:Transforms>
         <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
         <ds:DigestValue>pyQvFTSTiQve6TdB27AcQzmNaK8=</ds:DigestValue>
        </ds:Reference>
        <ds:Reference URI="#id-75">
         <ds:Transforms>
          <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
         </ds:Transforms>
         <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
         <ds:DigestValue>HbKGwtTp+Lv/d2I6nTvESwJduFw=</ds:DigestValue>
        </ds:Reference>
       </ds:SignedInfo>
       <ds:SignatureValue>VGb52Bb7aT(...)ZxHgqyU=</ds:SignatureValue>
       <ds:KeyInfo Id="KeyId-D086A2FA2C762748F0133700530211392">
        <wsse:SecurityTokenReference
         wsu:Id="STRId-D086A2FA2C762748F0133700530211393"
         xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
             utility-1.0.xsd">
         <wsse:Reference
          URI="#CertId-D086A2FA2C762748F0133700530211291"
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-
              profile-1.0#X509v3"/>
        </wsse:SecurityTokenReference>
       </ds:KeyInfo>
      </ds:Signature>
```

```
42      <wsu:Timestamp wsu:Id="Timestamp-73" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/
            oasis-200401-wss-wssecurity-utility-1.0.xsd">
43        <wsu:Created>2012-05-14T14:21:42.094Z</wsu:Created>
44        <wsu:Expires>2012-05-14T14:21:52.094Z</wsu:Expires>
45      </wsu:Timestamp>
46    </wsse:Security>
47  </soapenv:Header>
48
49  <soapenv:Body wsu:Id="id-75" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis
        -200401-wss-wssecurity-utility-1.0.xsd">
50    <xenc:EncryptedData Id="EncDataId-76" Type="http://www.w3.org/2001/04/xmlenc#Content">
51      <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc"/>
52      <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
53        <wsse:SecurityTokenReference xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis
              -200401-wss-wssecurity-secext-1.0.xsd">
54          <wsse:Reference URI="#EncKeyId-D086A2FA2C762748F0133700530213695"/>
55        </wsse:SecurityTokenReference>
56      </ds:KeyInfo>
57      <xenc:CipherData>
58        <xenc:CipherValue>+BeGR4rPmnRu(...)KKn6vs</xenc:CipherValue>
59      </xenc:CipherData>
60    </xenc:EncryptedData>
61  </soapenv:Body>
62 </soapenv:Envelope>
```

Linhas (02) - (47) Especificação do cabeçalho SOAP.

Linhas (03) - (46) Elemento de segurança definido no cabeçalho. Este elemento contém informação relacionada com a segurança da mensagem (encriptação, assinatura, *timestamp*, etc).

Linhas (05) - (11) Elemento que define o *token* de segurança da mensagem. Neste caso, representa um certificado X.509v3, codificado em Base64. Este *token* é distinto do *token* utilizado para a encriptação.

Linhas (12) - (41) Especificação da assinatura digital, baseada no certificado X.509. As linhas (13) a (30) indicam os elementos que são assinados. Na linha (14) é indicado o algoritmo de canonização e na linha (15) é indicado o algoritmo de assinatura.

Linhas (16) - (29) Nestas linhas são definidos os elementos que são assinados: a *timestamp* e o corpo da mensagem.

Linha (31) Nesta linha é especificado o valor da assinatura

Linhas (32) - (40) Nestas linhas é especificada a chave utilizada para a assinatura. Neste caso, é utilizada a chave X.509 contida na mensagem.

Linhas (49) - (61) Especificação do corpo da mensagem SOAP. O corpo da mensagem está encriptado e assinado.

De modo a assegurar que o conteúdos da mensagem não são alteráveis, este mecanismo produz uma *hash* (denominada de *digest*) por cada elemento do XML que necessita de ser assinado. No caso anterior são geradas duas *digest*: uma para o elemento *Body* e outra para o elemento *Timestamp*. A *digest* está presente no elemento `<ds:DigestValue>` (linhas 21 e 28) e o algoritmo utilizado para gerar esta *hash* no elemento `<ds:DigestMethod>` (linhas 20 e 27). De modo a produzir uma assinatura, as *digests* geradas são encriptadas juntamente com os elementos a que se referem. A *hash* da assinatura é então

colocada no elemento `<ds:SignatureValue>` (linha 31). Os métodos necessários à desencriptação da *hash* da assinatura são colocados no elemento `<ds:KeyInfo>` (linhas 32 a 40).

Para a verificação da consistência de uma mensagem assinada por parte de um destinatário, é necessário, num primeiro passo, proceder à desencriptação da *hash* da assinatura. De seguida é necessário que o destinatário recrie todos os passos efectuados pelo remetente de modo a obter uma *digest*. A comparação da *digest* desencriptada com a que foi gerada pelo destinatário da mensagem permite averiguar se os elementos assinados da mensagem foram modificados.

Uma *hash* gerada por um algoritmo como o SHA1 permite detectar, através de assinatura do XML, a mais pequena alteração do documento. Este característica é essencial para garantir a segurança da mensagem. No entanto este tipo de segurança representa um problema para o XML [Chappell 2002]. Tal como foi referido anteriormente, o envio de uma mensagem SOAP poderá estar sujeita a alterações por parte de intermediários. Estas alterações, por mais significativas que sejam (como por exemplo a remoção de um espaço em branco de um elemento), poderão originar uma *hash* do documento diferente da original. Para evitar este tipo de situações, foi introduzido pelo W3C um mecanismo denominado de Canonicalização. Este método define uma série de normas e formatos a serem aplicados a documentos XML, de modo a que os elementos abrangidos por este mecanismo num documento permaneçam iguais até à última iteração da mensagem. As normas definidas incluem a normalização de quebras de linha e valores de atributos, alteração de elementos CDATA por caracteres codificados, definição da codificação UTF-8, eliminação de espaços em branco fora de elementos e adição de atributos por defeito ao documento.

4.3.2.3 *Timestamp*

As *timestamps* permitem definir valores temporais numa mensagem SOAP. Parte integrante do WSS, este método é utilizado essencialmente para prevenir ataques de replicação. Além disso, também é utilizada como mecanismo de detecção de mensagens antigas que, ao serem executadas, poderão criar conflitos no sistema. No cabeçalho SOAP, dentro do elemento `wsse:Security`, é possível definir o elemento `wsu:Timestamp`. Dentro deste, é possível definir outros 3 elementos [Seely 2002]:

- **`wsu:Created`**: define a data e hora em que a mensagem foi criada.
- **`wsu:Expires`**: define a data e hora em que a mensagem expira.
- **`wsu:Received`**: define a data e hora em que a mensagem foi recebida por um determinado intermediário.

No exemplo da secção Assinatura do XML (4.3.2.2) está definido um *timestamp*:

```
1 <wsu:Timestamp wsu:Id="Timestamp-73" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/
      oasis-200401-wss-wssecurity-utility-1.0.xsd">
2   <wsu:Created>2012-05-14T14:21:42.094Z</wsu:Created>
3   <wsu:Expires>2012-05-14T14:21:52.094Z</wsu:Expires>
4 </wsu:Timestamp>
```

No exemplo anterior a mensagem tem um período de duração de 10 segundos. No caso do documento XML estar sujeito a assinatura, este elemento deverá ser assinado de modo a prevenir alterações antes da chegada da mensagem ao destino. Este mecanismo de segurança exige sincronização entre os relógios do cliente e do servidor.

4.3.3 Implementação do WSS no myPersonas

Com o desenvolvimento da versão 3.0 do sistema myPersonas, surgiu a necessidade de acrescentar WSS à invocação de serviços web através de SOAP. O projecto *Web Services Security for Java* (WSS4J), da fundação Apache, providencia uma implementação do standard WSS. Com base nesta biblioteca, e no projecto *Java API for XML Web Services* (JAX-WS), foi implementado um serviço para:

1. Verificar se a mensagem SOAP da invocação de um serviço web está assinada e encriptada;
2. Desencriptar a mensagem e verificar a identidade do cliente;
3. Executar o serviço web;
4. Enviar a resposta para o cliente, assinada e encriptada.

Para executar o passo 1 é necessário adicionar uma camada lógica ao serviço web denominada de interceptor. Um interceptor permite a manipulação de uma mensagem SOAP antes do serviço ser executado. Isto permite, por exemplo, desencriptar e verificar a assinatura de uma mensagem. Um interceptor não só permite efectuar operações sobre uma mensagem SOAP de entrada, como permite alterar uma mensagem SOAP de resposta à invocação do serviço.

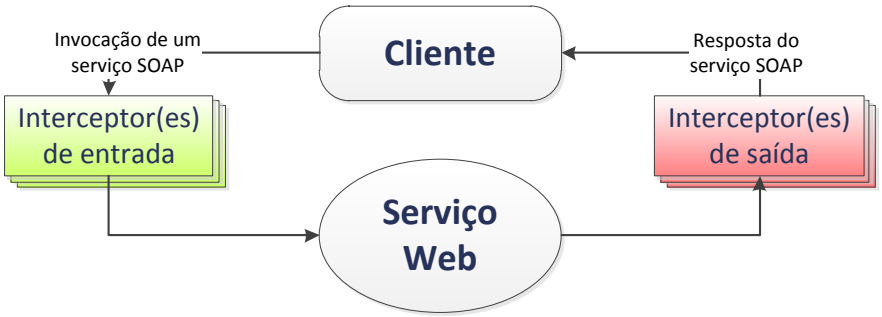


Figura 4.12: Diagrama de interceptores no JAX-WS

Nesta implementação vão ser necessários dois interceptores: um de entrada e outro de saída.

O interceptor de entrada tem a função de desencriptar a mensagem através da chave criptográfica privada do servidor, verificar se a assinatura é válida e, se for, identificar o cliente que invocou o serviço. Também é verificado se o *timestamp* da mensagem é válido, caso este esteja definido.

O interceptor de saída tem como função fazer o oposto do interceptor de entrada: encriptar a mensagem com a chave criptográfica pública do cliente e assinar a mensagem com a chave privada do servidor.

4.3.4 Testes e Resultados

De modo a avaliar a performance da solução implementada, foram executados alguns teste utilizando diversos mecanismos de segurança. Os testes foram realizados num computador com processador Core 2 Duo de 1,3GHz, com 4GB de memória RAM. O sistema operativo foi o Windows 7. O serviço web foi executado no *web server GlassFish* 3.1 e a versão do JRE é a 1.6.0_31.

O serviço web utilizado para obtenção dos resultados obtidos apenas soma dois números. Na mensagem SOAP do pedido devem ser especificados dois números inteiros. A resposta à invocação do serviço contém a soma dos números introduzidos. Os testes foram executados através de 10 *threads* que efectuaram invocações contínuas do serviço web durante 10 segundos.

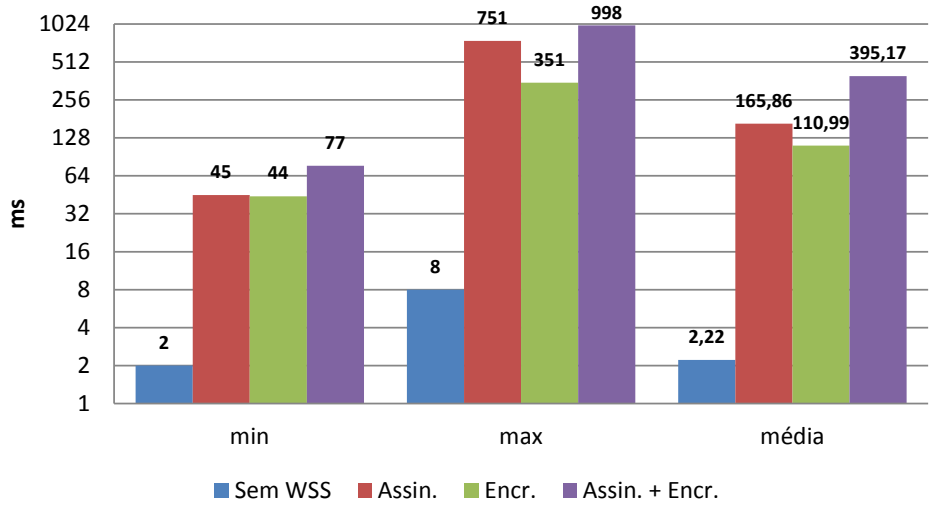


Figura 4.13: Tempos de execução de diversos testes aos mecanismos de segurança implementados pelo WSS

Na figura 4.13[1] estão ilustrados os tempos médios necessários à execução de quatro testes distintos. No primeiro teste, o serviço web descrito foi executado sem qualquer tipo de mecanismo de segurança, servindo os seus valores como padrão para os restantes testes. O tempo médio de chamado do serviço e respectiva resposta foi de 2,22 milissegundos. O tamanho do pacote recebido foi de 215 bytes.

No segundo teste foi implementado apenas assinatura do XML. Este teste pode ser dividido em cinco passos: assinatura da mensagem SOAP por parte do cliente, verificação da assinatura no servidor, execução do serviço, assinatura da mensagem de resposta pelo servidor e verificação da assinatura pelo cliente. Na assinatura foram utilizados os seguintes algoritmos:

- **Assinatura**: RSA-SHA1.
- **Canonicalização**: Canonical XML.
- ***Digest***: SHA1.dono

O tipo de chave utilizada para a assinatura foi *Binary Security Token*. Como é possível verificar na figura

[1] A figura 4.13 está representada com uma escala logarítmica.

anterior, o tempo de execução do pedido e obtenção da respectiva resposta aumentou consideravelmente em relação ao primeiro teste, na ordem dos 7500%.

No terceiro teste foi implementada apenas a encriptação das mensagens SOAP. Tal como no teste dois, este processo é dividido em cinco passos: encriptação da mensagem SOAP de invocação do serviço web, desencriptação da mensagem no servidor, execução do serviço web, encriptação da mensagem SOAP de resposta e, for fim, desencriptação da mensagem por parte do cliente. Na encriptação das mensagens forma utilizados os seguintes algoritmos:

- **Codificação Simétrica**: AES-128.
- **Encriptação da Chave**: RSA-v1.5.
- **Canonicalização**: Canonical XML.

Tal como no teste anterior, o tipo de chave utilizada para a encriptação foi *Binary Security Token*. À semelhança da assinatura do XML, também a encriptação tem um custo computacional associado bastante elevado. Em relação ao primeiro teste, o tempo de processamento das mensagens é da ordem dos 5000%. No entanto, e em comparação com a assinatura do XML, este mecanismo é 1,5 vezes mais rápido. Esta diferença de valores entre a assinatura e encriptação de mensagens XML pode ser explicado pelo algoritmo utilizado. Na publicação [Liu 2005] é realizada uma comparação da performance de vários algoritmos de assinatura e encriptação. Segundo os autores, para *arrays* de até 10KBytes, o algoritmo RSA-SHA1 demora 19ms a assinar e 1,3ms a verificar. Para o mesmo tamanho de *arrays*, o algoritmo AES-128 demora, em média, 0,3ms a encriptar e 0,5ms a desencriptar. Nesta diferença de tempos entre os dois algoritmos reside a discrepância de valores entre a assinatura e encriptação de mensagens SOAP. No mesmo estudo é concluído que os algoritmos utilizados nos testes 2 e 3 são os mais eficazes para os processos de assinatura e encriptação do XML.

No quarto e último teste são implementadas tanto encriptação como a assinatura da mensagem SOAP. Os algoritmos utilizados são os mesmos dos testes 2 e 3. Comparativamente, o tempo de latência obtido no primeiro teste é uma fracção da latência inerente à utilização de WSS.

Na figura 4.14 estão ilustrados os tamanhos das mensagens SOAP dos testes realizados anteriormente.

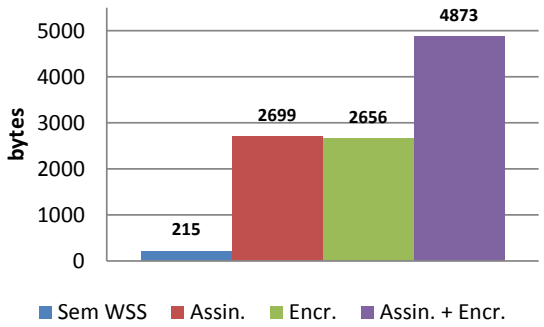


Figura 4.14: Tamanhos das mensagens SOAP dos testes aos mecanismos de segurança implementados pelo WSS

Os resultados obtidos assemelham-se aos testes de latência realizados. A utilização de encriptação e assinatura do XML tem repercussões no tamanho da mensagem: o tamanho da mensagem SOAP recebida pelo servidor no teste sem WSS representa 4,4% do tamanho da mensagem recebida com WSS. É expectável que os valores apresentados na figura 4.14 tendam a aumentar à medida que o XML da mensagem SOAP seja mais complexo [Liu 2005].

4.3.5 Considerações Finais

Na utilização de serviços web SOAP, e caso seja necessária uma solução de encriptação das mensagens ponto-a-ponto, o WSS é uma solução a considerar. Devido à complexidade da encriptação XML, o *overhead* no sistema, tanto no servidor como nos demais clientes, poderá contribuir significativamente para a degradação da comunicação e experiência do utilizador. Por exemplo, na invocação de um serviço web que some dois números (sendo ambos os números parâmetros definidos pelo cliente), a mensagem SOAP ocupa 215 bytes. Aplicando WSS à mensagem (encriptação + assinatura), a mensagem passa para os 4873 bytes, cerca de 22 vezes o tamanho da mensagem sem WSS.

Caso a performance do serviço web seja uma prioridade, então deve ser considerada a utilização de uma alternativa ao WSS. No caso de não ser necessária a alteração da mensagem SOAP por intermediários, então deverá ser considerada encriptação via SSL ou TLS.

5 Academic Playground & Innovation

O projecto Academic Playground & Innocation (APIn), financiado pela PTin e SAPO, consiste na criação de um portal agregador de diversas API's, incluindo documentação e exemplos. Este projecto, iniciado em Maio de 2011 e com duração de um ano, agrega serviços web de três fornecedores principais: PTin, SAPO e UA. Devido à vertente académica deste projecto, é dada a possibilidade aos alunos da UA de requisitarem servidores (máquinas virtuais) para o desenvolvimento de serviços web e/ou aplicações *mashup*. Qualquer aluno da UA que desenvolva um aplicação que utilize serviços presentes no portal APIn, tem a possibilidade de publicitar essa mesma aplicação através do portal.

O portal APIn é um portal Web 2.0. O *frontend* foi desenvolvido em HTML4, HTML5, JavaScript (*framework* jQuery) e CSS1-3. O *backend* foi desenvolvido em PHP, com ligação a uma base de dados MySQL.

5.1 Visão Geral

Numa fase inicial do projecto APIn, foram disponibilizados dois servidores: um servidor *Linux*, de domínio `http://services.web.ua.pt/` e um servidor *Windows*, de domínio `http://api.web.ua.pt/`. O servidor *Linux* destina-se ao desenvolvimento de serviços web. O servidor *Windows* além de alojar o portal APIn e a sua base de dados, também serve de suporte ao desenvolvimento de serviços em tecnologias incompatíveis com *Linux*, tal como a *framework* .NET, da *Microsoft*.

O portal API está disponível em duas linguagens, Português e Inglês, encontrando-se neste momento em versão *beta*. Devido a ser utilizado em meio académico, especificamente na UA, foi adoptado o design das páginas institucionais. Além de um serviço de *tickets*, suportado pelo Code.UA (`https://code.ua.pt/`), que permite reportar *bugs* ou requerer suporte e funcionalidades, o portal APIn conta com um *blog* e está presente na rede social *Twitter*. Existem três páginas com visibilidade pública:

- **Início**: página inicial do portal, que conta com uma pequena descrição do projecto e as últimas notícias publicadas no *Twitter* e *blog*.

- **Serviços**: página com o directório de serviços. Organizado por fornecedores, é através desta página que um utilizador pode consultar informações detalhadas sobre um serviço (ver Serviços (5.2)).

- **Aplicações**: nesta secção são apresentadas algumas aplicações que utilizam serviços disponibilizados através pelo portal APIn (ver Aplicações (5.3)).

Figura 5.1: Página inicial do portal APIn

Neste projecto existe um máquina virtual de *staging*, alojada no Instituto de Telecomunicações (IT), pólo de Aveiro, de domínio api.av.it.pt. Qualquer eventual modificação do portal APIn é primeiro efectuada no servidor de *staging* e, após terminados todos os testes, a actualização é colocada no servidor de produção.

O portal APIn conta com dois níveis de acesso e categorização dos seus utilizadores. Além dos utilizadores normais, existem os administradores do portal, que têm permissões para controlar (quase) todas as definições do portal através do próprio portal. Esta característica será detalhada na secção Área Administrativa (5.5).

Também está disponível uma área privada do utilizador. Ao aceder ao endereço `http://api.web.ua.pt/home`, o utilizador poderá obter algumas informações suas no portal, tais como a data de registo, data de último acesso, os serviços que publicou e o servidores que lhe estão alocados.

No endereço `http://api.web.ua.pt/faq` está disponível um conjunto de perguntas e respectivas respostas sobre a utilização do portal APIn.

5.1.1 Autenticação

5.1.1.1 Arquitectura Utilizada no APIn

O sistema de autenticação da UA consiste num IdP *Shibboleth*. No início do projecto optou-se por apenas pedir autorização aos STIC para que a máquina *Linux* tivesse acesso ao IdP da UA. Esta decisão prendeu-se com o facto de os serviços identificados no Serviços Web (3) poderem necessitar de aceder a recursos protegidos. Deste modo, o servidor *Linux* passa a ser um *proxy* para o IdP da UA.

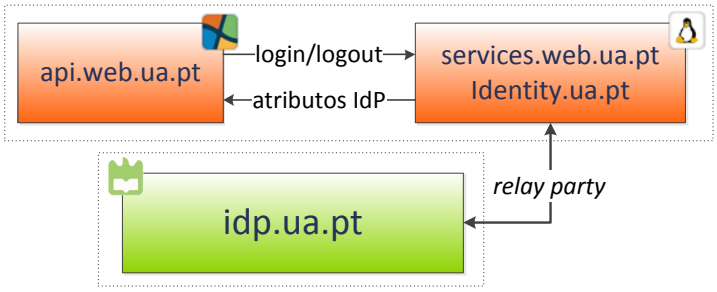


Figura 5.2: Esquema dos servidores APIn e comunicação com o IdP da UA

Na figura 5.2 é possível observar a interacção que existe entre o servidor api.web.ua.pt, o services.web.ua.pt/identity.ua.pt e o IdP da UA para a autenticação e *logout* no portal APIn.

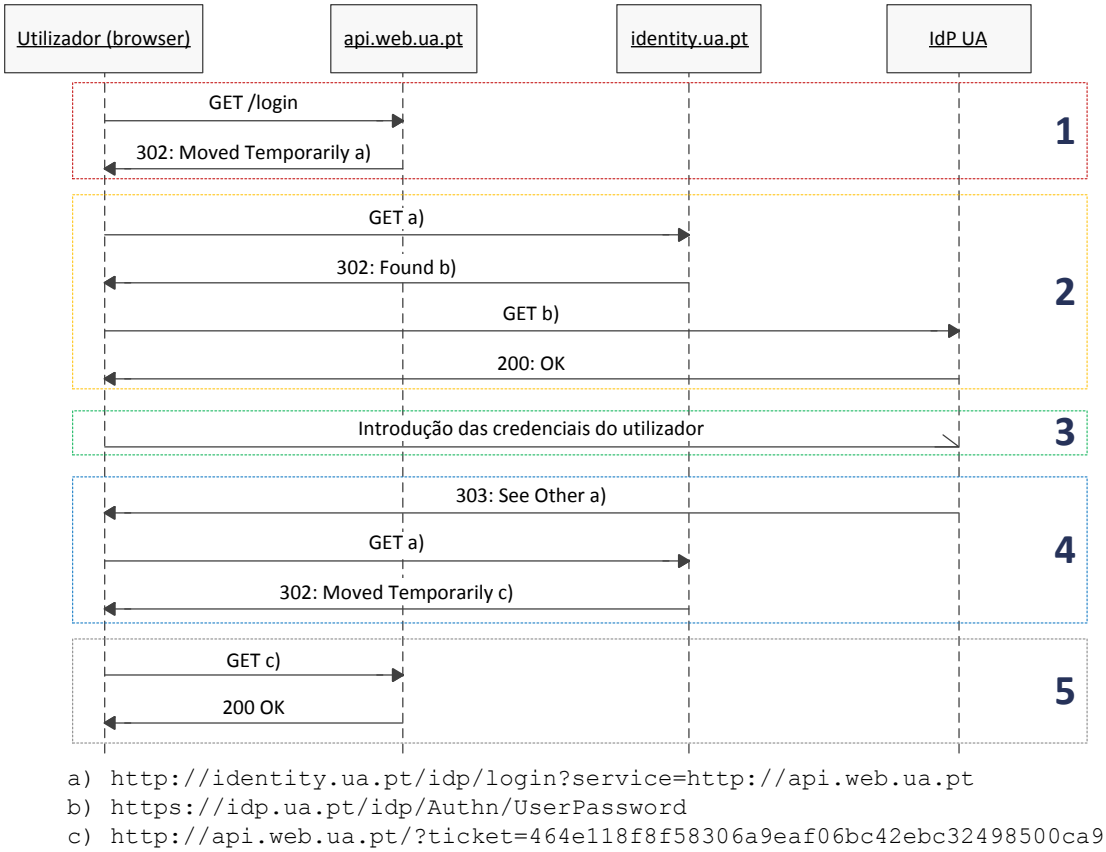


Figura 5.3: Diagrama de sequência do processo de autenticação no portal APIn

Na figura 5.3 é descrito o processo de autenticação no portal APIn através do IdP da UA, com o servidor *proxy* identity.ua.pt:

1. O utilizador, através de um *browser,* inicia o processo de autenticação. Para tal é invocado o URL `http://api.web.ua.pt/login`. Como o processo de autenticação tem de ser realizado através do servidor identity.ua.pt, o utilizador é redireccionado para o URL `http://identity.ua.pt/idp/login?service=http://api.web.ua.pt`. No parâmetro `service` do URL anterior especifica o URL de retorno após autenticação no IdP.

2. Após contactar o servidor identity.ua.pt, o utilizador é novamente redireccionado, desta vez para o IdP da UA. No *browser* do utilizador surgirá uma página web com um formulário para a introdução das credenciais (figura 5.6);

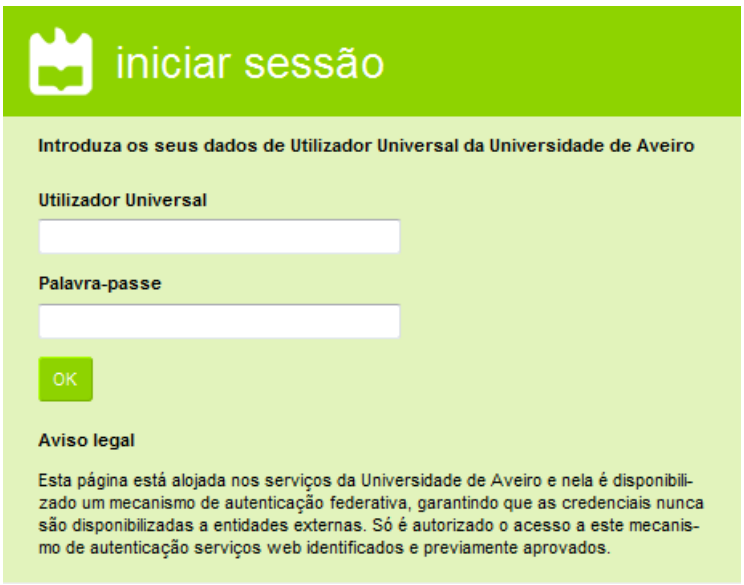


Figura 5.4: Página web de autenticação no IdP da UA

3. Caso o utilizador já tiver iniciado a sessão através de outro portal da UA (por exemplo, PACO ou Moodle), e caso a sessão ainda não tenha expirado, o utilizador não necessita de introduzir novamente as suas credenciais.

4. Após a validação das credenciais pelo IdP, o utilizador é novamente redireccionado para o URL `http://identity.ua.pt/idp/login?service=http://api.web.ua.pt`. Os atributos do utilizador disponibilizados pelo IdP foram comunicados ao servidor identity.ua.pt. É gerada uma *hash* única que permite identificar inequivocamente o pedido. Através de um sistema de *cache* denominado *Memcached*, os atributos devolvidos pelo IdP são guardados, sendo a chave de entrada da *cache* a *hash* gerada anteriormente.

5. Neste último passo, o utilizador é redireccionado para o portal APIn (`http://api.web.ua.pt/?ticket=464e118f8f58306a9eaf06bc42ebc32498500ca9`. No endereço é acrescentado o parâmetro `ticket`, que contém a chave necessária para para obter os atributos do utilizador. No lado do servidor api.web.ua.pt, e antes do portal APIn ser enviado para o *browser*, é estabelecida uma comunicação com o servidor identity.ua.pt para obter os atributos do utilizador. É invocado o URL `http://identity.ua.pt/idp/get?ticket=464e118f8f58306a9eaf06bc42ebc32498500ca9` e, caso a entrada exista na *Memcached* do servidor identity.ua.pt, os atributos são retornados. A comunicação dos atributos por parte do servidor identity.ua.pt para o portal APIn é feita de forma encriptada.

O processo de *logout* é bastante semelhante ao processo de *login* descrito anteriormente:

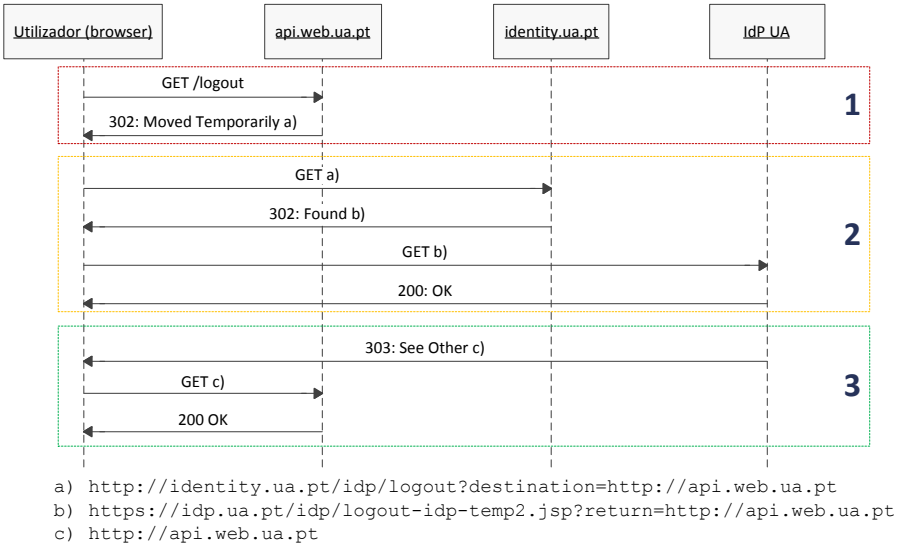


Figura 5.5: Diagrama de sequência do processo de *logout* no portal APIn

1. O utilizador, através do *browser*, inicia o processo de *logout*. Para tal é necessário aceder ao endereço `http://api.web.ua.pt/login`. Tal como no processo de autenticação, o utilizador tem de ser redireccionado para o servidor identity.ua.pt (`http://identity.ua.pt/idp/logout?destination=http://api.web.ua.pt`).

2. No servidor identity.ua.pt, o utilizador é novamente redireccionado, desta vez para o IdP da UA, através do URL `https://idp.ua.pt/idp/logout-idp-temp2.jsp?return=http://api.web.ua.pt`. No *browser* do utilizador surgirá uma página web, semelhante à da figura 5.6, que indica ao utilizador a operação que esta a ser realizada e o URL para onde será feito o redireccionamento. A sessão existente do utilizador no IdP será terminada.

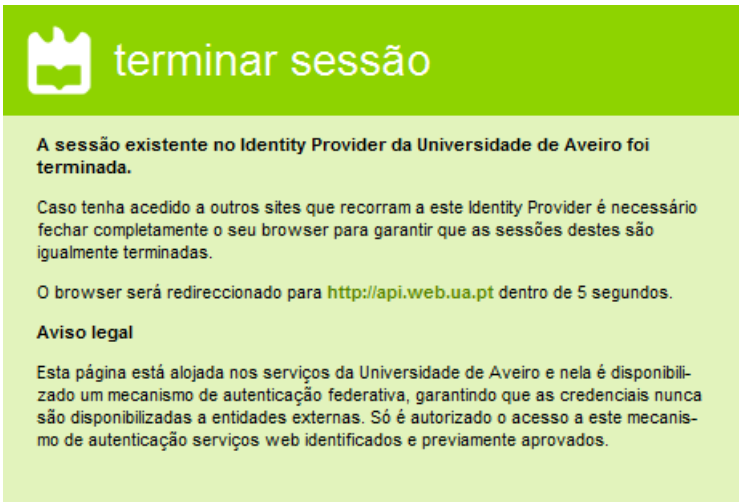


Figura 5.6: Página web de *logout* no IdP da UA

3. Após estar concluído o processo de *logout* no IdP da UA, o utilizador será redireccionado para o URL definido no passo 1 através do parâmetro `destination`.

Para a comunicação com o IdP da UA foi utilizado, no servidor identity.ua.pt, a *framework* de autenticação SimpleSAMLphp (`http://simplesamlphp.org/`). Desenvolvida em PHP, esta *framework* lida essencialmente com autenticações através dos protocolos de identidade Shibboleth 1.3, A-Select, CAS, OpenID, WS-Federation e OAuth.

5.2 Serviços

Na página web `http://api.web.ua.pt/services` é possível aceder aos serviços publicado no portal APIn. Esta página esta dividida em duas categorias: fornecedores e serviços:

- **Fornecedores**: entidades responsáveis pelos serviços web. Existem três fornecedores que representam empresas ou instituições: PTin, SAPO e UA. Apenas os administradores podem adicionar fornecedores, fornecedores estes que podem representar disciplinas (na figura 5.7 existem duas disciplinas: Arquitecturas Distribuídas e Engenharia de Serviços). É também possível associar *tags* a um fornecedor, para uma categorização e pesquisa mais eficientes, *tags* essas que são herdadas pelos serviços do fornecedor. Podem ser definidos níveis de visibilidade e acessibilidade para os fornecedores:
  - **Não visível**: um fornecedor definido como não visível (barra vermelha ao lado esquerdo do mesmo), apenas está visível para os administradores da página. Esta propriedade é herdada pelos serviços web que este detém.
  - **Privado**: um fornecedor considerado privado (barra amarela ao lado esquerdo do mesmo) esta acessível publicamente. No entanto, as *wikis* dos serviços web contidos neste apenas são visíveis para utilizadores autenticados.
- **Serviços**: definição dos serviços representados por uma entidade. Estes serviço são representados por um titulo, descrição, visibilidade, acessibilidade e, opcionalmente, *tags*. Os níveis de visibilidade e acessibilidade são iguais aos níveis definidos para os fornecedores. Para cada serviço é possível associar uma *wiki*. Esta *wiki* permite definir tópicos de ajuda para o consumo do serviço web.

Figura 5.7: Página web dos serviços no portal APIn

Também é possível mostrar todos os serviços disponíveis, como é possível ver na figura 5.7, e obter uma ordenação por *tags* dos serviços.

Nos fornecedores definidos como disciplinas, é possível definir que utilizadores têm permissão para adicionar serviços, visível na figura 5.8. Esta opção apenas esta disponível para os administradores.

Figura 5.8: Permissões de publicação de serviços

5.2.1 Wiki

Para cada serviço adicionado ao portal APIn é possível adicionar uma *wiki*. Para o suporte destas *wiki* utilizou-se o motor de um software colaborativo para *wikis* denominado "DokuWiki" (`http://www.dokuwiki.org/dokuwiki`). Desenvolvido por Andreas Gohr desde 2004, o DokuWiki utiliza a sua própria sintaxe para a criação de *wikis*. É possível a coexistência de várias revisões para a mesma *wiki* assim como definir contas de utilizador e grupos. Também tem suporte para *plugins* [Gohr 2004].

Quando é adicionado um novo serviço web, é criada automaticamente uma *wiki* referente a esse serviço web. A *wiki* criada apenas poderá ser editada pelo dono do serviço ou pelos administradores.

Figura 5.9: *Wiki* do serviço OAuth da UA

Na figura 5.9 é possível ver a *wiki* do serviço OAuth, da UA. Existe uma *dropdown* onde estão presentes todas as versões da *wiki* em causa. Também está definido uma tabela de conteúdos, que permite uma navegação mais rápida pela *wiki*. Para a edição da *wiki* é utilizado um editor *What You See Is What You Get* (WYSIWYG).

Quando é seleccionado um serviço web, é feita uma chamada AJAX ao servidor a requer a *wiki* do serviço. Na resposta são enviados uma serie de parâmetros para além da *wiki* em si, já em formato HTML. Os parâmetros incluem uma lista das revisões da *wiki*, informações sobre a revisão actual e as permissões para edição.

Para editar uma *wiki*, é utilizado uma característica do HTML que permite a edição visual de elementos HTML. No entanto, a *wiki* tem de ser guardada no formato definido pelo DokuWiki. Para tal é necessário fazer uma conversão de HTML para esse formato. Esta conversão é feita do lado do cliente através de JS.

Figura 5.10: Guardar *wiki* de um serviço

5.3 Aplicações

O projecto APIn é utilizado em algumas disciplinas da UA, que culminam com a criação de projectos por parte dos alunos. De modo a que esses projectos/aplicações que usam serviços web disponibilizados pelo APIn sejam publicitados, existe uma secção no portal que permite a publicação de aplicações.

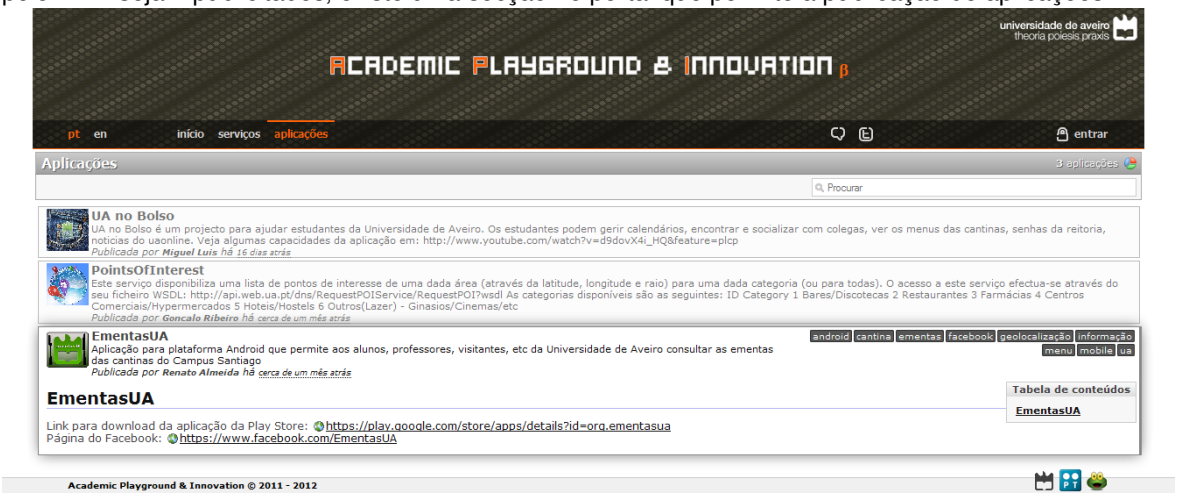


Figura 5.11: Aplicações publicadas no portal APIn

As aplicações partilham maior parte dos elementos presentes nos serviços web. É possível definir um título para a aplicação, uma descrição, associar *tags* assim como uma *wiki*. Depois de adicionada, uma aplicação necessita de ser aprovada por um administrador. Só depois de ser aprovada é que a aplicação fica publicamente visível.

5.4 Redes Sociais e Serviço de *Tickets*

Como já foi referido, o portal APIn utiliza outros serviços, tanto para divulgação de notícias como de apoio e interacção com os utilizadores. Para a divulgação de notícias, são utilizados dois meios:

- ***Blog***: o *blog*, de endereço `http://api.web.ua.pt/blog`, permite a divulgação de notícias categorizadas. Integrado com o IdP da UA, também é possível que qualquer utilizador escreva um artigo sobre a aplicação que publicou no portal APIn.

- ***Twitter***: o portal APIn também utiliza o *Twitter* para divulgar notícias por esta rede social. O endereço é `https://twitter.com/#!/api_ua`.

As notícias publicadas no *blog* e no *Twitter* também podem ser acedidas pelo portal APIn. Na página inicial (figura 3.11) está disponível a última notícia publica em cada plataforma. Através da barra de ligações do portal é possível aceder a todas as noticias, como é possível ver na figura 5.12.

Figura 5.12: Últimas notícias no portal APIn

O portal APIn também utiliza o sistema de *tickets* do Code.UA (`https://code.ua.pt/`). Este sistema permite a qualquer utilizador expor as suas questões, tanto sobre o portal como sobre os serviços web que são disponibilizados. Do ponto de vista dos administradores, o Code.UA também é utilizado como repositório, tanto para o portal como para alguns serviços web desenvolvidos (ver Serviços Web (3)). As páginas do projecto no Code.UA são `https://code.ua.pt/projects/api/` (projecto público para a colocação de questões e indicação de erros), `https://code.ua.pt/projects/apisource/` (projecto privado com o intuito de servir de repositório para o código fonte do portal APIn) e `https://code.ua.pt/projects/apiservices/` (projecto privado para repositório de serviços web desenvolvidos no âmbito do projecto APIn).

5.5 Área Administrativa

Devido à dimensão do portal API, foi necessário a criação de uma área administrativa. Assim que o utilizador se autentica, é verificado se este é administrador ou não. Caso seja, ficam disponíveis locais no portal que estão interditos aos outros utilizadores. Nas figuras figura 5.13 e figura 5.14 é possível ver uma comparação entre a barra de ligação de um utilizador normal e de um administrador.

Figura 5.13: Barra de ligações de um utilizador normal

Figura 5.14: Barra de ligações de um administrador

As páginas web disponíveis para os administradores são:

- **Estatísticas**: página onde estão disponíveis algumas estatísticas de utilização do portal APIn. Está disponível uma classificação dos serviços web mais consultados assim como estatísticas de acesso ao servidor services.web.ua.pt.

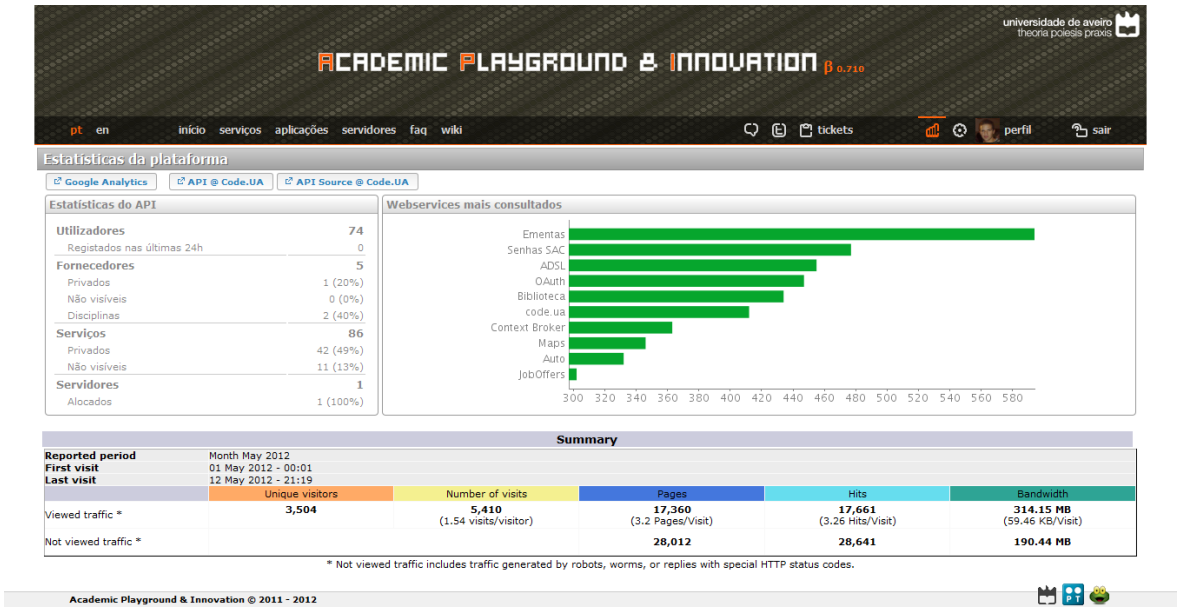


Figura 5.15: Página de estatísticas do portal APIn

- **Definições**: nesta página é possível configurar algumas definições do portal APIn. Esta página esta dividida em quatro sub-páginas:

  - **Plataforma**: configurações relativas à própria plataforma. Podem, por exemplo, ser configurados o título das páginas do portal e o código do *Google Analytics*.
  - **Utilizadores**: nesta página é possível ter acesso a uma lista de todos os utilizadores que já se autenticaram no portal, assim como definir o nível de acesso do utilizador ou as imagens das

máquinas virtuais pelas quais é responsável. É também possível listar os serviços e servidores de cada utilizador.

– **Histórico**: no histórico são listadas todas as operações do portal. Cada operação contém uma descrição, o utilizador responsável, o endereço IP do pedido e a data em que foi efectuada. As operações estão divididas em três categorias: informação, aviso ou erro.

– ***Change log***: nesta página é possível listar todas as alterações feitas ao portal APIn.

5.6 Dados Estatísticos

Durante o decorrer do projecto foram recolhidos diversos dados estatísticos sobre a utilização do portal APIn. Foram igualmente recolhidos dados sobre o consumo dos serviços disponibilizados no servidor services.web.ua.pt. Estes dados são interessantes com vista ao futuro desenvolvimento, tanto do portal APIn como dos serviços.

O "epoch", tanto do portal APIn como do servidor services.web.ua.pt foi no dia 11 de Outubro de 2011.

5.6.1 Portal APIn

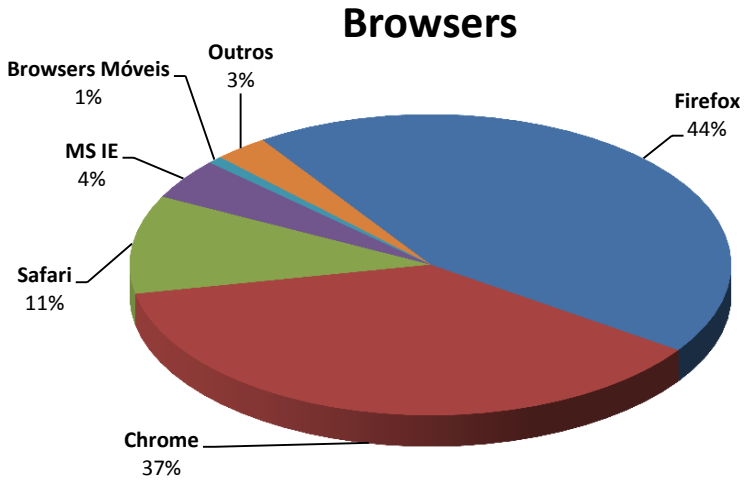


Figura 5.16: *Browsers* utilizados no acesso ao portal

Na figura 5.16 é possível ver os tipos de *browsers* que acederam ao portal entre 11 de Outubro de 2011 e 31 de Maio de 2012. Embora o Firefox seja o *browser* com mais acessos, esta estatística poderá estar influenciada por ambos os criadores do portal usarem Firefox. É também de salientar o baixo número de acessos por dispositivos móveis. Este facto poderá estar relacionado por o portal não disponibilizar uma interface para este tipo de dispositivos, degradando a experiência para o utilizador.

5.6.1.1 Utilizadores Registados

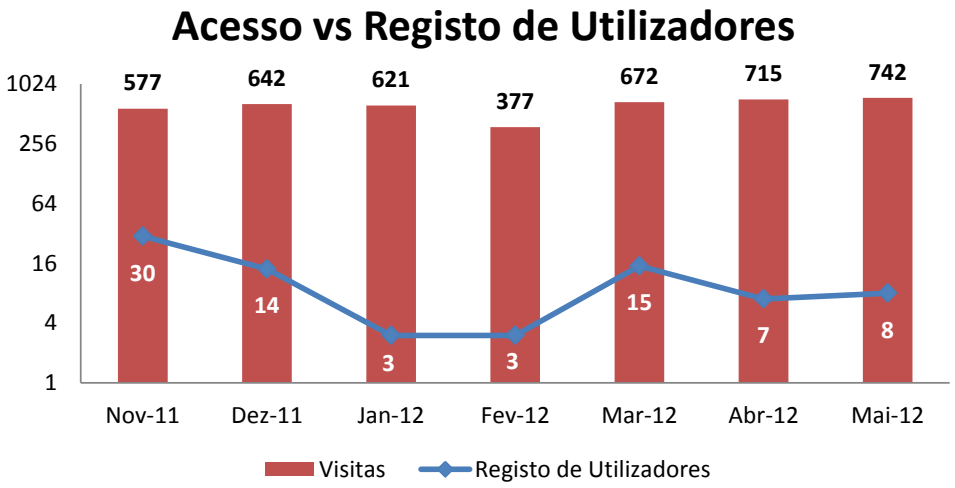


Figura 5.17: Acessos vs número de utilizadores registados

No gráfico da figura 5.17 é apresentada uma comparação entre o número de acessos ao portal APIn e o registo de utilizadores. Os valores do registo de utilizadores evidenciam com bastante clareza a utilização do portal para fins académicos. Há uma primeira vaga de registos que coincide com a implementação da autenticação no portal. A segunda vaga coincide com o início do segundo semestre na UA, em Março de 2012. Nota: A escala do número de visitas é logarítmica.

5.6.1.2 Acessos ao Portal

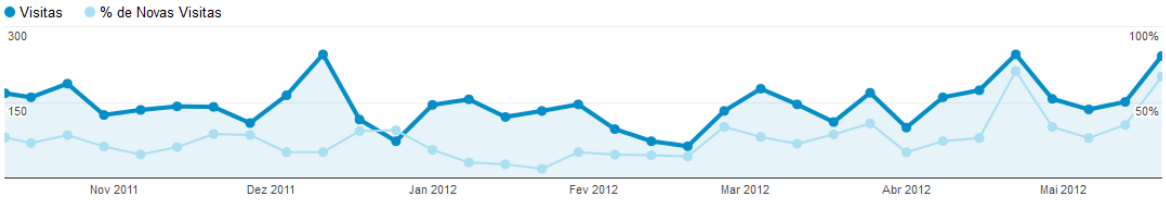


Figura 5.18: Visitas e percentagem de novas visitas ao portal

No gráfico da figura 5.18 é feita uma comparação entre o número de visitas e a percentagem de novas visitas. É possível considerar o acesso ao portal constante ao longo dos 8 meses da amostragem.

Numa análise mais minuciosa ao gráfico anterior, é evidente o pico de visitas a meio do mês de Dezembro. Este pico pode ser explicado pela entrega de trabalhos no âmbito de várias disciplinas. Após o fim do semestre, no início de Fevereiro, há uma quebra de acessos ao portal. Após a introdução do portal em novas disciplinas, no início de Março, o número de visitas volta a subir.

Os dados foram recolhidos através do serviço *Analytics* do Google.

5.6.2 Servidor services.web.ua.pt

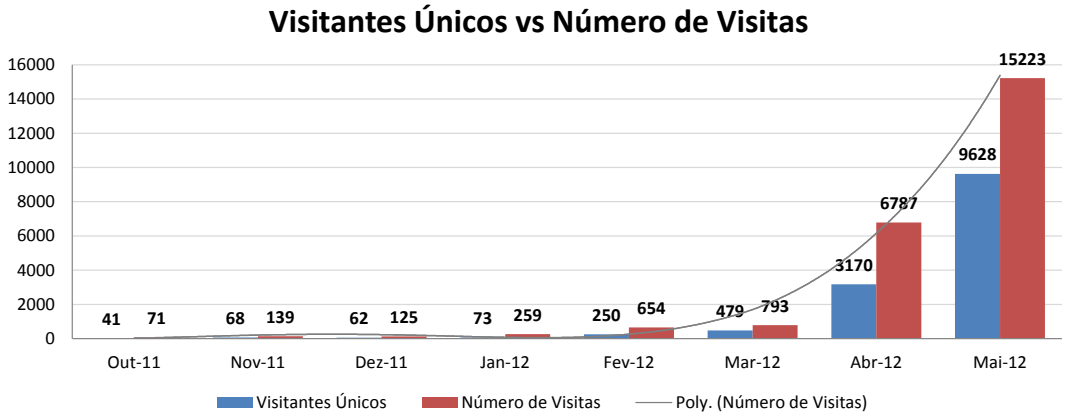


Figura 5.19: Gráfico visitantes únicos vs número de visitas

No gráfico da figura 5.19 é possível obter uma comparação entre o número de visitas e os visitantes únicos mensais no servidor services.web.ua.pt. É bastante notória a evolução exponencial que se verifica a partir de Março de 2012. Este aumento pode ser explicado pela utilização de serviços por entidades externas (televisões do Departamento de Electrónica, Telecomunicações e Informática da UA (DETI), informações das senhas na página web do PACO (`http://paco.ua.pt/`) e acesso ao serviço das ementas por aplicações móveis).

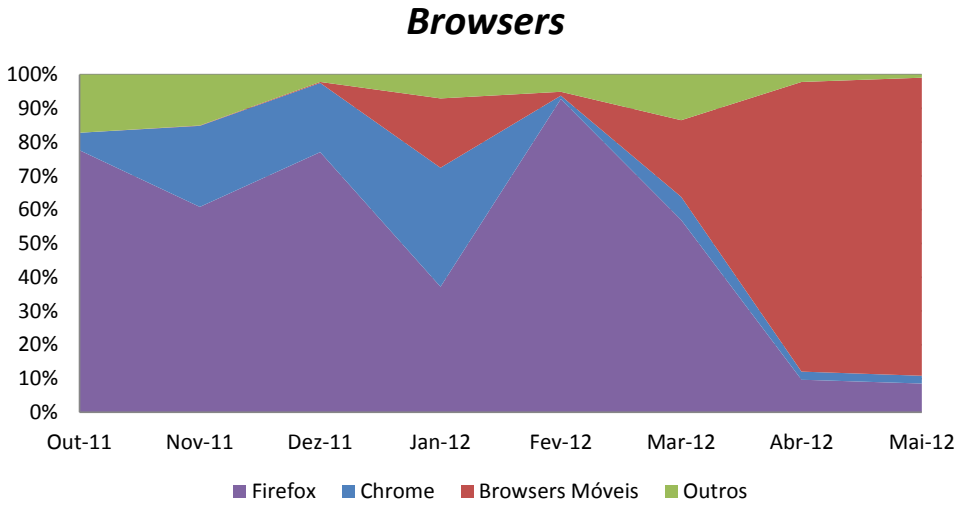


Figura 5.20: Relação entre os tipos de *browsers* utilizados

Na figura 5.20 está ilustrada a utilização de *browsers* para o acesso aos serviços do servidor services.web.ua.pt. É evidente a grande utilização de *browsers* para dispositivos móveis a partir de Fevereiro de 2012. Este facto deve-se ao desenvolvimento de aplicações para *smartphones* com o intuito de mostrar as ementas das cantinas da UA.

6 Conclusões e Trabalho Futuro

Na implementação de serviços web há inúmeras variáveis que se devem ter em consideração. Antes de iniciar a implementação do serviço web, é necessário proceder a uma categorização rigorosa da informação a ser disponibilizada. Isto é, proceder a uma clara distinção entre informação pública e informação privada. A informação que é considerada de carácter pessoal deverá ser tratada com o máximo de cuidado, necessitando da implementação de mecanismo de controlo de acesso. Depois de efectuada esta caracterização, deverá ser realizada uma análise cuidada dos melhores mecanismos para divulgação das fontes de dados no meio. Embora a arquitectura REST comece a ganhar vantagem sobre implementações SOAP, poderão ocorrer casos particulares em que exista uma necessidade de adoptar a arquitectura SOAP.

Um outro factor que não deve ser descurado é a aplicação de uma licença que defina, clara e concisamente, as permissões de utilização, redistribuição e, sobretudo, que tipos de referências devem ser atribuídas aos donos dos dados. Tendo em conta a legislação de várias nações, poderão existir barreiras legais à distribuição de dados ou conjuntos de dados. A aplicação de uma licença ajuda a prevenir este tipo de situações. Considerando a internacionalidade destas licenças, o problema de partilha de dados e informação entre estados passa a estar solucionado.

O acesso a serviços cuja informação é considerada sensível deparou-se como um desafio ao longo desta dissertação. Quando o projecto APIn foi apresentado em algumas disciplinas, o acesso a serviços privados da UA foi bastante requisitado. Uma dessas informações foi o horário escolar do aluno. Sendo o aluno proprietário desta informação, coloca-se um problema de partilha da mesma. Evidentemente que dono de um recurso deve exercer total controlo sobre a sua partilha. Ou seja, para partilhar informação privada, são necessários dois mecanismos: um de autorização e outro de controlo. Após alguma pesquisa, o protocolo OAuth apresenta-se como uma boa solução para este problema. Embora a versão OAuth não seja a mais actual, este está a funcionar em pleno e, através do portal de gestão, é possível ter um controlo total sobre os recursos que são partilhados com terceiros.

6.1 Trabalho Futuro

Existem alguns pontos que podem ser melhorados neste projecto. No portal APIn podem ser definidas novas interfaces de visualização, que abranjam mais dispositivos com acesso à Internet. Em relação aos serviços, existem uma série de informações espalhadas por diversas páginas web da UA que são candidatas à criação de um serviço web.

Com a criação do servidor identity.ua.pt, foi possível a implementação de um servidor OAuth. A versão implementada (1.0a) padece de algumas limitações que são ultrapassadas na versão 2.0 deste protocolo. No entanto, a implementação da versão 2.0 implica a instalação de um certificado SSL genuíno no servidor, o que é neste momento impossível devido aos custos associados. Esta opção deverá ser considerada no futuro, pois este servidor também é *relay party* do IdP da UA.

6.1.1 Portal APIn

Embora o portal APIn já apresente alguma maturação, existe ainda uma série de aspectos que podem ser desenvolvidos. Com a grande expansão dos dispositivos móveis no mercado, há lugar a uma necessidade de os *websites* se adaptarem às novas formas de visualização e interacção que surgem. Por exemplo, os *smartphones* apresentam resoluções de ecrã bastante reduzidas, dificultando as visualização de páginas web que não estejam preparadas para este tipo de dispositivos. A interacção nestes dispositivos também é diferente, sendo utilizado o dedo indicador para a navegação. Desta forma, há lugar à implementação de uma interface de navegação para dispositivos móveis no portal APIn que melhore a experiência do utilizador.

Com o método de edição visual das *wikis* dos serviços web, é necessária a transformação do formato HTML para a sintaxe definida pela *DokuWiki*. Este processo, embora seja executado no lado do cliente, é bastante susceptível a erros de conversão assim como o tempo de processamento é proporcional à dimensão *wiki*. Como a *DokuWiki* não permite a salvaguarda de *wikis* no formato HTML, mais cedo ou mais tarde deverá ser adoptado outro mecanismo para a gestão de *wikis*.

Um dos objectivos traçados no início deste projecto passava pela utilização do sistema *myPersonas*, da PTin, para o controlo de identidades através do IdP da UA. A integração deste sistema no portal possibilitaria a utilização de serviços úteis para partilha de conhecimento. Por exemplo, existe um serviço de comentários, denominado *Disqus* (`http://disqus.com/welcome/`), que permite adicionar uma secção de comentários a qualquer página. Como foi referido anteriormente, a *wiki* de um serviço web apenas pode ser editada pelo criador do serviço ou por um administrador. O *Disqus*, através do sistema *myPersonas*, proporcionaria um ambiente colaborativo no desenvolvimento e especificação de *wikis* para serviços web Devido a atrasos no lançamento de uma nova versão do *myPersonas*, não houve oportunidade de utilização do mesmo. Se o projecto APIn for continuado, a integração do sistema *myPersonas* no portal deverá ser uma prioridade.

Referências

[Abelson 2008] **Hal Abelson, Ben Adida, Mike Linksvayer & Nathan Yergler**. *ccREL: The Creative Commons Rights Expression Language*. In Creative Commons, volume 1, Março 2008.

[Alm 2010] **Christopher Alm & Roland Illig**. *Translating High-level Authorization Constraints to XACML*. IEEE 6th World Congress on Services, pages 629 – 636, 2010.

[Anderson 2006] **Anne Anderson**. *Web services policies*. Security & Privacy, IEEE, vol. 4, pages 84 – 87, 2006.

[Austin 2002] **Daniel Austin, Abbie Barbir, Christopher Ferris & Sharad Garg**. *Web Services Architecture Requirements*, Em linha, Disponível na Internet, `http://www.w3.org/TR/2002/WD-wsa-reqs-20020819`, Consultado a 7 de Maio, 2012.

[Ball 2011] **Alex Ball**. *How to License Research Data*, Em linha, Disponível na Internet, `http://www.dcc.ac.uk/webfm_send/332`, Consultado a 5 de Maio, 2012.

[Bartel 2008] **Mark Bartel, John Boyer, Barb Fox, Brian LaMacchia & Ed Simon**. *XML Signature Syntax and Processing (Second Edition)*. W3C, Em linha, Disponível na Internet, `http://www.w3.org/TR/2008/REC-xmldsig-core-20080610/`, Consultado a 14 de Abril, 2012.

[Bisel 2007] **Larry D. Bisel**. *The Role of SSL in Cybersecutiry*. IT Professional, vol. 9, pages 22 – 25, Abril 2007.

[Booth 2004] **David Booth, Hugo Haas, Francis McCabe, Eric Newcomer, Michael Champion, Chris Ferris & David Orchard**. *Web Services Architecture*. W3C, Em linha, Disponível na Internet, `http://www.w3.org/TR/ws-arch/`, Consultado a 7 de Maio, 2012.

[Box 2000] **Don Box, David Ehnebuske, Gopal Kakivaya, Andrew Layman, Noah Mendelsohn, Henrik Frystyk Nielsen, Satish Thatte & Dave Winer**. *Simple Object Access Protocol (SOAP) 1.1*, Em linha, Disponível na Internet, `http://www.w3.org/TR/2000/NOTE-SOAP-20000508/`, Consultado a 7 de Maio, 2012.

[Bray 2008] **Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler & François Yergeau**. *Extensible Markup Language (XML) 1.0 (Fifth Edition)*, Em linha, Disponível na Internet, `http://www.w3.org/TR/xml/`, Consultado a 2 de Maio, 2012.

[Chappell 2002] **David Chappell & Tyler Jewell**. *Java Web Services*. O'Reilly, Março 2002.

[Chatti 2011] **Mohamed Amine Chatti, Matthias Jarke, Marcus Specht, Ulrik Schroeder & Daniel Dahl**. *Model-Driven Mashup Personal Learning Environments*. Int. J. Technology Enhanced Learning, vol. 3, pages 1 – 23, 2011.

[Comission 2011] **European Comission**. *Digital Agenda: Turning Government Data Into Gold*, Em linha, Disponível na Internet, `http://europa.eu/rapid/pressReleasesAction.do?reference=IP/11/1524&format=HTML&aged=0&language=EN&guiLanguage=en`, Consultado a 22 de Março, 2012.

[Commons 2003] **Creative Commons**. *About The Licenses*, Em linha, Disponível na Internet, `http://creativecommons.org/licenses/`, Consultado a 1 de Maio, 2012.

[Connolly 2002] **Dan Connolly**. *URIs, Addressability, and the use of HTTP GET*. W3C, Em linha, Disponível na Internet, `http://www.w3.org/2001/tag/doc/get7`, Consultado a 7 de Abril, 2012.

[Dietrich 2010] **Daniel Dietrich, Jonathan Gray, Tim McNamara, Antti Poikola, Rufus Pollock, Julian Tait & Ton Zijlstr**. *The Open Data Handbook*, Em linha, Disponível na Internet, http://opendatahandbook.org/en/index.html, Consultado a 18 de Maio, 2012.

[draft-ietf-oauth-v2-26] **Ed. E. Hammer, D. Recordon, Facebook, D. Hardt & Microsoft**. *The OAuth 2.0 Authorization Framework*. draft-ietf-oauth-v2-26, Internet Engineering Task Force (IETF), Maio 2012.

[Fielding 2000] **Roy Thomas Fielding**. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, Irvine, 2000.

[Figueira 2011] **Diogo Figueira**. *Protocolo de Comunicação GEF - PACO*. Micro I/O, Janeiro 2011.

[Gohr 2004] **Andreas Gohr**. *DokuWiki*, Em linha, Disponível na Internet, http://www.dokuwiki.org/dokuwiki, Consultado a 10 de Fevereiro, 2012.

[Halpin 2012] **Harry Halpin**. *Web Authentication: The next step in the evolving identity ecosystem?* Web 2.0 Security & Privacy, Maio 2012.

[Hammer 2009] **Eran Hammer**. *Explaining the OAuth Session Fixation Attack*, Em linha, Disponível na Internet, http://hueniverse.com/2009/04/explaining-the-oauth-session-fixation-attack/, Consultado a 9 de Maio, 2012.

[Hammersley 2005] **Ben Hammersley**. *Developing Feeds with RSS and Atom*. O'Reilly, Abril 2005.

[Hickson 2012] **Ian Hickson**. *HTML5 Web Messaging*. W3C, Em linha, Disponível na Internet, http://dev.w3.org/html5/postmsg/, Consultado a 20 de Janeiro, 2012.

[Hilaiel 2011] **Lloyd Hilaiel**. *How BrowserID Works*, Em linha, Disponível na Internet, http://lloyd.io/how-browserid-works, Consultado a 29 de Maio, 2012.

[Hoxha 2011] **Julia Hoxha & Armand Brahaj**. *Open Government Data on the Web: A Semantic Approach*. Emerging Intelligent Data and Web Technologies (EIDWT), pages 107 – 113, Setembro 2011.

[Huston 1999] **Geoff Huston**. *Web Caching*. The Internet Protocol Journal, vol. 2, Setembro 1999.

[IBM 2002] **IBM**. *Web Services Security*, Em linha, Disponível na Internet, http://www.ibm.com/developerworks/library/specification/ws-secure/, Consultado a 10 de Abril, 2012.

[Imamura 2002] **Takeshi Imamura, Blair Dillaway & Ed Simon**. *XML Encryption Syntax and Processing*. W3C, Em linha, Disponível na Internet, http://www.w3.org/TR/xmlenc-core/, Consultado a 19 de Novembro, 2011.

[Internet2 2005] **Internet2**. *What's Shibboleth?*, Em linha, Disponível na Internet, http://shibboleth.net/about/index.html, Consultado a 2 de Dezembro, 2012.

[JSON 2006] **JSON**. *http://www.json.org/*, Consultado a 2 de Janeiro, 2012.

[Kambayashi 2001] **Kai Kambayashi & Mukesh Mohania**. *Efficient Management of Data in Proxy Cache*. Database and Expert Systems Applications, pages 479 – 483, Setembro 2001.

[Kaplan 2005] **Jeff Kaplan**. *Roadmap for Open ICT Ecosystems*. Berkman Center Publication Series, 2005.

[Korn 2011] **Naomi Korn & Charles Oppenheim**. *Licensing Open Data: A Practical Guide*, Em linha, Disponível na Internet, http://discovery.ac.uk/files/pdf/Licensing_Open_Data_A_Practical_Guide.pdf, Consultado a 2 de Fevereiro, 2012.

[Kuhn 2001] **D. Rchard Kuhn, Vincent C. Hu, W. Timothy Polk & Shu-Jen Chang**. *Introduction to Public Key Technology and the Federal PKI Infrastructure*. NIST, Em linha, Disponível na Internet, http://csrc.nist.gov/publications/nistpubs/800-32/sp800-32.pdf, Consultado a 22 de Fevereiro, 2012.

[Lang 2010] **Jean-Philippe Lang**. *Rest api - Redmine*, Em linha, Disponível na Internet, http://www.redmine.org/projects/redmine/wiki/Rest_api, Consultado a 10 de Novembro, 2011.

[Lavarack 2010] **Tristan Lavarack & Marijke Coetzee**. *Considering web services security policy compatibility*. In Information Security for South Africa (ISSA), pages 1 – 8, Agosto 2010.

[Leiba 2012] **Barry Leiba**. *OAuth Web Authorization Protocol*. IEEE Internet Computing, vol. 16-1, Janeiro/Fevereiro 2012.

[Libris 2011] **Ex Libris**. *List of X-Services*, Em linha, Disponível na Internet, http://catalogo.up.pt/X?op=explain&file=list, Consultado a 11 de Novembro, 2011.

[Liu 2005] **Hongbin Liu, Shrideep Pallickara & Geoffrey Fox**. *Performance of Web Services Security*. In 3th Annual Mardi Gras Conference, Fevereiro 2005.

[Madsen 2005] **Paul Madsen & Eve Maler**. *SAML V2.0 Executive Overview*. OASIS, Em linha, Disponível na Internet, http://www.oasis-open.org/committees/download.php/13525/sstc-saml-exec-overview-2.0-cd-01-2col.pdf, Consultado a 20 de Maio, 2012.

[Malik 2011] **Sanjay Kumar Malik & SAM Rizvi**. *Information Extraction using Web Usage Mining, Web Scrapping and Semantic Annotation*. Computational Intelligence and Communication Networks, pages 465 – 469, Outubro 2011.

[Nordbotten 2009] **Nils Agne Nordbotten**. *XML and Web Services Security Standards*. Communications Surveys & Tutorials, IEEE, vol. 11, pages 4 – 21, 2009.

[OASIS 2005] **OASIS**. *Profiles for the OASIS Security Assertion Markup Languages (SAML) V2.0*, Em linha, Disponível na Internet, http://docs.oasis-open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf, Consultado a 1 de Maio, 2012.

[OASIS 2010] **OASIS**. *eXtensible Access Control Markup Language (XACML) Version 3.0*, Em linha, Disponível na Internet, http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-cs-01-en.pdf, Consultado a 1 de Maio, 2012.

[Obama 2010] **Administração Obama**. *Data.gov*, Em linha, Disponível na Internet, http://www.data.gov/, Consultado a 24 de Abril, 2012.

[Oppliger 2003] **Rolf Oppliger**. *Microsoft .Net Passport: a security analysis*. Computer, vol. 36, pages 29 – 35, Julho 2003.

[O'Reilly 2005] **Tim O'Reilly**. *What Is Web 2.0*, Em linha, Disponível na Internet, http://oreilly.com/web2/archive/what-is-web-20.html, Consultado a 20 de Abril, 2012.

[Pereira 2011a] **Rui Pereira**. *acesso.ua.pt - XML*, Em linha, Disponível na Internet, http://acesso.ua.pt/xml/help.html, Consultado a 2 de Novembro, 2011.

[Pereira 2011b] **Rui Pereira**. *Jornal Online da Universidade de Aveiro - XML*, Em linha, Disponível na Internet, http://uaonline.ua.pt/xml/help.html, Consultado a 2 de Novembro, 2011.

[Raggett 1999] **Dave Raggett, Arnaud Le Hors & Ian Jacobs**. *HTML 4.01 Specification*. W3C, Em linha, Disponível na Internet, http://www.w3.org/TR/1999/REC-html401-19991224/interact/scripts.html, Consultado a 3 de Maio, 2012.

[Ragouzis 2008] **Nick Ragouzis, John Hughes, Rob Philpott, Eve Maler, Paul Madsen & Tom Scavo**. *Security Assertion Markup Language (SAML) V2.0 Technical Overview*. OASIS, Em linha, Disponível na Internet, `http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0-cd-02.html`, Consultado a 22 de Maio, 2012.

[Rehman 2007] **Rafeeq Rehman**. *The OpenID Book*. Conformix Books, 2007.

[RFC 2045] **N. Freed, Innosoft, N. Borenstein & First Virtual**. *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*. RFC 2045, Internet Engineering Task Force (IETF), Novembro 1996.

[RFC 2388] **L. Masinter**. *Returning Values from Forms: multipart/form-data*. RFC 2388, Internet Engineering Task Force (IETF), Agosto 1998.

[RFC 2616] **R. Fielding, UC Irvine, J. Gettys, Compaq/W3C, J. Mogul, Compaq, H. Frystyk, W3C/MIT, L. Masinter, Xerox, P. Leach, Microsoft & T. Berners-Lee**. *Hypertext Transfer Protocol – HTTP/1.1*. RFC 2616, Internet Engineering Task Force (IETF), Junho 1999.

[RFC 3023] **M. Murata, IBM Tokyo Research Laboratory, S. St.Laurent, simonstl.com, D. Kohn & Skymoon Ventures**. *XML Media Types*. RFC 3023, Internet Engineering Task Force (IETF), Janeiro 2001.

[RFC 3529] **W. Harold**. *Using Extensible Markup Language-Remote Procedure Calling (XML-RPC) in Blocks Extensible Exchange Protocol (BEEP)*. RFC 3529, Internet Engineering Task Force (IETF), Abril 2003.

[RFC 3986] **T. Berners-Lee, W3C/MIT, R. Fielding, Day Software, L. Masinter & Adobe Systems**. *Uniform Resource Identifier (URI): Generic Syntax*. RFC 3986, Internet Engineering Task Force (IETF), Janeiro 2005.

[RFC 4180] **Y. Shafranovich & Inc. SolidMatrix Technologies**. *Common Format and MIME Type for Comma-Separated Values (CSV) Files*. RFC 4180, Internet Engineering Task Force (IETF), Outubro 2005.

[RFC 4627] **D. Crockford**. *The application/json Media Type for JavaScript Object Notation (JSON)*. RFC 4627, Internet Engineering Task Force (IETF), Julho 2006.

[RFC 4648] **S. Josefsson**. *The Base16, Base32, and Base64 Data Encodings*. RFC 4648, Internet Engineering Task Force (IETF), Outubro 2006.

[RFC 5849] **Ed. E. Hammer-Lahav**. *The OAuth 1.0 Protocol*. RFC 5849, Internet Engineering Task Force (IETF), Abril 2010.

[RFC 6454] **A. Barth & Inc. Google**. *The Web Origin Concept*. RFC 6454, Internet Engineering Task Force (IETF), Dezembro 2011.

[Richardson 2007] **Leonard Richardson & Sam Ruby**. *RESTful Web Services*. O'Reilly, 2007.

[Scavo 2005] **Tom Scavo & Scott Cantor**. *Shibboleth Architecture*. Internet2, Em linha, Disponível na Internet, `http://shibboleth.internet2.edu/docs/draft-mace-shibboleth-tech-overview-latest.pdf`, Consultado a 10 de Janeiro, 2012.

[Schwarz 2005] **Jerry Schwarz, Bret Hartman, Anthony Nadalin, Chris Kaler, Mark Davis, Frederick Hirsch & Frederick Hirsch**. *Security Challenges, Threats and Countermeasures Version 1.0*. WS-I, Em linha, Disponível na Internet, `http://www.ws-i.org/profiles/basicsecurity/securitychallenges-1.0.pdf`, Consultado a 15 de Abril, 2012.

[Seely 2002] **Scott Seely**. *Understanding WS-Security*. Microsoft, Em linha, Disponível na Internet, http://msdn.microsoft.com/en-us/library/ms977327.aspx, Consultado a 12 de Novembro, 2011.

[Severance 2012] **Charles Severance**. *Discovering JavaScript Object Notation*. Computer, vol. 45, pages 5 – 8, Abril 2012.

[Singh 2004] **Inderjeet Singh, Sean Brydon, Greg Murray, Vijay Ramachandran, Thierry Violleau & Beth Stearns**. *Designing Web Services with the J2EE 1.4 Platform JAX-RPC, SOAP, and XML Technologies*. Addison Wesley, Junho 2004.

[Singhal 2007] **Anoop Singhal, Theodore Winograd & Karen Scarfone**. *Guide to Secure Web Services*. NIST, Em linha, Disponível na Internet, http://csrc.nist.gov/publications/nistpubs/800-95/SP800-95.pdf, Consultado a 1 de Abril, 2012.

[Stair 2012] **Ralph Stair & George Reynolds**. *Information Systems*. Course Technology, Cengage Learning, 2012.

[Teixeira 2009] **Cláudio Teixeira**. *Infra-estrutura Para Portal Internet Integrador de Serviços*. PhD thesis, Universidade de Aveiro, 2009.

[Twitter 2011] **Twitter**. *PIN-based authorization*, Em linha, Disponível na Internet, https://dev.twitter.com/docs/auth/pin-based-authorization, Consultado a 3 de Março, 2012.

[Twitter 2012] **Twitter**. *POST statuses/update-with-media | Twitter Developers*, Em linha, Disponível na Internet, https://dev.twitter.com/docs/api/1/post/statuses/update_with_media, Consultado a 28 de Abril, 2012.

[van Kesteren 2012] **Anne van Kesteren**. *Cross-Origin Resource Sharing*. W3C, Em linha, Disponível na Internet, http://www.w3.org/TR/cors/, Consultado a 9 de Janeiro, 2012.

[Vickery 2011] **Graham Vickery**. *Review of Recent Studies on PSI Re-use and Related Market Developments*, Em linha, Disponível na Internet, http://ec.europa.eu/information_society/policy/psi/docs/pdfs/report/final_version_study_psi.docx, Consultado a 18 de Abril, 2012.

[Vollmer 2011] **Timothy Vollmer & Diane Peters**. *Creative Commons and Public Sector Information: Flexible Tools to Ssupport PSI Creators and Re-users*. In Topic Report 23. European Public Sector Information Platform, Fevereiro 2011.

[Ying 2010] **Wang Ying**. *Research on Multi-level Security of Shibboleth Authentication Mechanism*. Information Processing (ISIP), pages 450 – 453, Outubro 2010.

[Yoshida 2004] **Shigeru Yoshida, Hironori Yahagi & Junichi Odagiri**. *CSV compaction to improve data-processing performance for large XML documents*. Data Compression Conference (DCC '04), page 574, 2004.

[Zervaas 2008] **Quentin Zervaas**. *Practical Web 2.0 Applications with PHP*. Apress, 2008.

[Özses 2009] **Seda Özses & Salih Ergül**. *Cross-domain communications with JSONP, Part 1: Combine JSONP and jQuery to quickly build powerful mashups*, Em linha, Disponível na Internet, http://www.ibm.com/developerworks/library/wa-aj-jsonp1/, Consultado a 4 de Maio, 2012.

A Serviços Web Identificados

Nesta secção são descritos os serviços web identificados na UA. Os parâmetros dos serviços web descritos neste capítulo que estejam sublinhados são de carácter obrigatório. Todos os restantes são parâmetros opcionais. Os valores por defeito dos parâmetros opcionais, caso existam, estão a negrito.

Nota: devido a algumas respostas serem demasiado extensas, optou-se por truncar as mesmas. No local onde as respostas foram truncadas, é possível observar a notação `(...)`. Tal significa que o excerto da resposta que foi ocultado tem exactamente o mesmo padrão do excerto visível, mudando apenas os valores do mesmo, os quais não representam nenhum valor acrescentado para o âmbito desta dissertação.

A.1 Biblioteca da UA

A.1.1 Find

Esta operação permite obter o número de registos para uma determinada pesquisa.

URL `http://opac.ua.pt/X?op=find`

Método HTTP GET

Formato da Resposta XML

Argumentos:

- `base`: A base (biblioteca) onde se pretende executar a pesquisa.
- `request`: O valor a ser pesquisado
- `adjacent`: Define o modo de pesquisa quando o valor a ser pesquisa contém mais que uma palavra:
  - **`N`**: A pesquisa é efectuada por palavra.
  - `Y`: A pesquisa é efectuada com o valor completo.
- `code`: Local onde a pesquisa deve ser efectuada:
  - **`wrd`**: Lista todos os documentos que contêm a *string* definida no parâmetro `request`.
  - `wti`: A pesquisa é efectuada no título.
  - `wau`: A pesquisa é efectuada por autor.
  - `wln`: A pesquisa é efectuada por língua.
  - `wco`: A pesquisa é efectuada por colecção.
  - `wyr`: A pesquisa é efectuada por ano de publicação.
  - `issn`: A pesquisa é efectuada pelo campo International Standard Serial Number (ISSN).

  - `isbn`: A pesquisa é efectuada pelo campo International Standard Book Number (ISBN).
  - `sys`: A pesquisa é efectuada por número de registo.
  - `bar`: A pesquisa é efectuada por código de barras.
  - `cot`: A pesquisa é efectuada por localização (cota).

Exemplo `http://opac.ua.pt/X?op=find&code=wrd&request=portugal&base=BUA01`

Resposta

```
<?xml version = "1.0" encoding = "UTF-8"?>
<find>
  <set_number>269296</set_number>
  <no_records>000024263</no_records>
  <no_entries>000001000</no_entries>
  <session-id>DFNQHSB1YRYNBQXNLJMN83GPM52DFUCQMGI1GRLJMMXFAJSJJC</session-id>
</find>
```

A.1.2 Present

Esta operação permite listar informações detalhadas sobre os documentos obtidos pela operação Find (A.1.1).

URL `http://opac.ua.pt/X?op=present`

Método HTTP GET

Formato da Resposta XML

Argumentos:

- `base`: A base (biblioteca) onde se pretende executar a pesquisa.
- `set_number`: Campo `set_number` obtido através da operação Find (A.1.1).
- `set_entry`: Número de entradas a serem devolvidas. Este argumento pode ter dois formatos distintos:
  - `000000001-0000000012` (Conjunto): Lista as entras 1 a 12.
  - `000000001,000000005,0000000010` (Singular): Listas as entradas 1, 5 e 10. Estas devem ser separadas por vírgula.

Exemplo `http://opac.ua.pt/X?op=present&set_entry=000000001&set_number=269296&base=BUA01`

Resposta

```
<?xml version = "1.0" encoding = "UTF-8"?>
<present>
  <record>
    <record_header>
      <set_entry>000000001</set_entry>
    </record_header>
    <doc_number>000248495</doc_number>
    <metadata>
```

```
 9      <oai_marc>
10        <fixfield id="FMT">BK</fixfield>
11        <fixfield id="LDR">-----nam--22--------450-</fixfield>
12        <fixfield id="001">000248495</fixfield>
13        <varfield id="100" i1=" " i2=" ">
14          <subfield label="a">20120410d1949----k--a0pory0103----ba</subfield>
15        </varfield>
16        <varfield id="101" i1="0" i2=" ">
17          <subfield label="a">por</subfield>
18        </varfield>
19        <varfield id="102" i1=" " i2=" ">
20          <subfield label="a">PT</subfield>
21        </varfield>
22        <varfield id="200" i1="1" i2=" ">
23          <subfield label="a">Ultamar Português</subfield>
24          <subfield label="f">António Mendes Corrêa</subfield>
25        </varfield>
26        <varfield id="210" i1=" " i2=" ">
27          <subfield label="a">Lisboa</subfield>
28          <subfield label="c">Agência Geral das Colónias. Divisão de Publicações e
                Biblioteca</subfield>
29          <subfield label="d">1949</subfield>
30        </varfield>
31        <varfield id="215" i1=" " i2=" ">
32          <subfield label="a">^^vol</subfield>
33        </varfield>
34        <varfield id="317" i1=" " i2=" ">
35          <subfield label="a">Doação Aldónio Gomes</subfield>
36        </varfield>
37        <varfield id="327" i1="0" i2=" ">
38          <subfield label="a">Vol. 1: Síntese da África. - 400, [33] p. : il., CXVII est
                ., 1 carta geográf. desdob.</subfield>
39        </varfield>
40        <varfield id="615" i1=" " i2=" ">
41          <subfield label="a">Expansão portuguesa</subfield>
42        </varfield>
43        <varfield id="606" i1=" " i2=" ">
44          <subfield label="a">História de Portugal</subfield>
45          <subfield label="y">Africa</subfield>
46        </varfield>
47        <varfield id="675" i1=" " i2=" ">
48          <subfield label="a">94(469):94(6)</subfield>
49          <subfield label="v">BN3</subfield>
50          <subfield label="z">por</subfield>
51        </varfield>
52        <varfield id="700" i1=" " i2="1">
53          <subfield label="a">Correia,</subfield>
54          <subfield label="b">António Mendes,</subfield>
55          <subfield label="f">1888-1960</subfield>
56        </varfield>
57        <varfield id="801" i1=" " i2="0">
58          <subfield label="a">PT</subfield>
59          <subfield label="b">UAVSD</subfield>
60          <subfield label="c">20120410</subfield>
61          <subfield label="g">RPC</subfield>
62        </varfield>
63        <fixfield id="BAS">01</fixfield>
64        <varfield id="CAT" i1=" " i2=" ">
65          <subfield label="a">MSILVA</subfield>
66          <subfield label="b">02</subfield>
67          <subfield label="c">20120410</subfield>
68          <subfield label="l">BUA01</subfield>
69          <subfield label="h">1225</subfield>
70        </varfield>
71        <varfield id="CAT" i1=" " i2=" ">
72          <subfield label="a">DANIEL</subfield>
73          <subfield label="b">02</subfield>
74          <subfield label="c">20120411</subfield>
75          <subfield label="l">BUA01</subfield>
```

```
76          <subfield label="h">1408</subfield>
77        </varfield>
78      </oai_marc>
79    </metadata>
80  </record>
81  <session-id>J16HX74VTHCVXJ4S42VMDB383CBB3ESEMB1473SKREI4L9DT4R</session-id>
82 </present>
```

A.1.3 Finddoc

Esta operação permite obter informações detalhadas sobre um documento.

URL `http://opac.ua.pt/X?op=find-doc`

Método HTTP GET

Formato da Resposta XML

Argumentos:

- `base`: A base (biblioteca) onde se pretende executar a pesquisa.
- `doc_number` / `doc_num`: Número de publicação. Este valor pode ser obtido através dos resultados da operação *Present*.

Exemplo `http://opac.ua.pt/X?op=find-doc&doc_num=000234151&base=BUA01`

Resposta

```
1  <?xml version = "1.0" encoding = "UTF-8"?>
2  <find-doc>
3    <record>
4      <metadata>
5        <oai_marc>
6          <fixfield id="FMT">BK</fixfield>
7          <fixfield id="LDR">-----nam--22--------450-</fixfield>
8          <varfield id="001" i1=" " i2=" ">
9            <subfield label="1">000234151</subfield>
10         </varfield>
11         <varfield id="010" i1=" " i2=" ">
12           <subfield label="a">978-972-21-2076-0</subfield>
13           <subfield label="b">brochado</subfield>
14         </varfield>
15         <varfield id="100" i1=" " i2=" ">
16           <subfield label="a">20100702d2009----k--y0pory0103----ba</subfield>
17         </varfield>
18         <varfield id="101" i1="0" i2=" ">
19           <subfield label="a">por</subfield>
20         </varfield>
21         <varfield id="102" i1=" " i2=" ">
22           <subfield label="a">PT</subfield>
23         </varfield>
24         <varfield id="200" i1="1" i2=" ">
25           <subfield label="a">Caim</subfield>
26           <subfield label="e">romance</subfield>
27           <subfield label="f">José Saramago</subfield>
28         </varfield>
29         <varfield id="205" i1=" " i2=" ">
30           <subfield label="a">5ª ed</subfield>
31         </varfield>
```

```
      <varfield id="210" i1=" " i2=" ">
        <subfield label="a">Alfragide</subfield>
        <subfield label="c">Caminho</subfield>
        <subfield label="d">cop. 2009</subfield>
      </varfield>
      <varfield id="215" i1=" " i2=" ">
        <subfield label="a">181 p.</subfield>
      </varfield>
      <varfield id="675" i1=" " i2=" ">
        <subfield label="a">821.134.3</subfield>
        <subfield label="v">BN3</subfield>
        <subfield label="z">por</subfield>
      </varfield>
      <varfield id="700" i1=" " i2="1">
        <subfield label="a">Saramago,</subfield>
        <subfield label="b">José,</subfield>
        <subfield label="f">1922-2010</subfield>
      </varfield>
      <varfield id="801" i1=" " i2="0">
        <subfield label="a">PT</subfield>
        <subfield label="b">UAVSD</subfield>
        <subfield label="c">20100702</subfield>
        <subfield label="g">RPC</subfield>
      </varfield>
      <fixfield id="BAS">01</fixfield>
      <varfield id="CAT" i1=" " i2=" ">
        <subfield label="a">OLGA</subfield>
        <subfield label="b">02</subfield>
        <subfield label="c">20100621</subfield>
        <subfield label="l">BUA01</subfield>
        <subfield label="h">1614</subfield>
      </varfield>
      <varfield id="CAT" i1=" " i2=" ">
        <subfield label="a">SUSY</subfield>
        <subfield label="b">40</subfield>
        <subfield label="c">20100623</subfield>
        <subfield label="l">BUA01</subfield>
        <subfield label="h">1412</subfield>
      </varfield>
      <varfield id="CAT" i1=" " i2=" ">
        <subfield label="a">SUSY</subfield>
        <subfield label="b">00</subfield>
        <subfield label="c">20100701</subfield>
        <subfield label="l">BUA01</subfield>
        <subfield label="h">1353</subfield>
      </varfield>
      <varfield id="CAT" i1=" " i2=" ">
        <subfield label="a">ICARVALHO</subfield>
        <subfield label="b">02</subfield>
        <subfield label="c">20100701</subfield>
        <subfield label="l">BUA01</subfield>
        <subfield label="h">1353</subfield>
      </varfield>
      <varfield id="CAT" i1=" " i2=" ">
        <subfield label="a">CSA</subfield>
        <subfield label="b">02</subfield>
        <subfield label="c">20100702</subfield>
        <subfield label="l">BUA01</subfield>
        <subfield label="h">1641</subfield>
      </varfield>
      <varfield id="CAT" i1=" " i2=" ">
        <subfield label="a">SUSY</subfield>
        <subfield label="b">40</subfield>
        <subfield label="c">20100705</subfield>
        <subfield label="l">BUA01</subfield>
        <subfield label="h">0917</subfield>
      </varfield>
    </oai_marc>
  </metadata>
```

```
  </record>
  <session-id>DD4376RTHAS4ACKXQ7EEF1FCMXJCJPJQJEHVJC31U2IPQU9K7X</session-id>
</find-doc>
```

A.2 Code.UA

A.2.1 Ocorrências

Esta operação permite listar todas as ocorrências dos projectos definidos como públicos. Também é possível listar as ocorrências para um determinado projecto.

URL `http://code.ua.pt/issues.[xml|json]`

Método HTTP GET

Formato da Resposta XML / JSON

Argumentos relevantes:

- `project_id`: Se este argumento for definido, são listadas as ocorrências apenas desse projecto. Caso o projecto seja privado, é necessário introduzir as credencias de autenticação no IdP da UA.
- `status_id`: Define o estado das ocorrências a serem listadas. Possíveis valores: `open`, `closed`, `*`. Caso este argumento não seja definido, são listadas apenas as ocorrências no estado `open`.
- `assigned_to_id`: Permite listar as ocorrências atribuídas a um determinado utilizador.

Exemplo `http://code.ua.pt/issues.xml?project_id=api&status_id=*`

Resposta

```
<?xml version="1.0" encoding="utf-16"?>
<issues type="array" offset="0" total_count="48">
  <issue>
    <id>2879</id>
    <project name="API" id="562" />
    <tracker name="Suporte" id="3" />
    <status name="Fechado" id="5" />
    <priority name="Urgente" id="6" />
    <author name="David Miguel Vicente Ferreira" id="317" />
    <assigned_to name="Francisco Alexandre de Gouveia" id="760" />
    <category name="Serviços SAPO" id="44" />
    <subject>Serviço de SMS da SAPO</subject>
    <description>É possível obter o permissões para o serviço de SMS da SAPO até dia 1
        de fevereiro? Tenho uma entrega e apresentação de um trabalho de redes móveis
        nesse dia ao qual usa este serviço. </description>
    <start_date>2012-01-23</start_date>
    <due_date></due_date>
    <done_ratio>100</done_ratio>
    <estimated_hours></estimated_hours>
    <created_on>2012-01-23T19:45:09+00:00</created_on>
    <updated_on>2012-02-01T17:53:50+00:00</updated_on>
  </issue>
  (...)
</issues>
```

A.2.2 Ocorrência

Além de listar todas as ocorrências de um projecto, é também possível listar uma única ocorrência através do seu identificador. Caso a ocorrência pretendida pertença a um projecto privado, é necessária uma prévia autenticação através do IdP da UA.

URL `http://code.ua.pt/issues/[id].[xml|json]`

Método HTTP GET

Formato da Resposta XML / JSON

Exemplo `http://code.ua.pt/issues/2879.json`

Resposta

```
{
  "issue": {
    "priority": {
      "name": "Urgente",
      "id": 6
    },
    "tracker": {
      "name": "Suporte",
      "id": 3
    },
    "spent_hours": 0,
    "status": {
      "name": "Fechado",
      "id": 5
    },
    "start_date": "2012/01/23",
    "created_on": "2012/01/23 19:45:09 +0000",
    "description": "É possível obter o permissões para o serviço de SMS da SAPO até dia
        1 de fevereiro? Tenho uma entrega e apresentação de um trabalho de redes móveis
         nesse dia ao qual usa este serviço. ",
    "category": {
      "name": "Serviços SAPO",
      "id": 44
    },
    "assigned_to": {
      "name": "Francisco Alexandre de Gouveia",
      "id": 760
    },
    "subject": "Serviço de SMS da SAPO",
    "author": {
      "name": "David Miguel Vicente Ferreira",
      "id": 317
    },
    "done_ratio": 100,
    "updated_on": "2012/02/01 17:53:50 +0000",
    "project": {
      "name": "API",
      "id": 562
    },
    "id": 2879
  }
}
```

A.2.3 Projectos

Esta operação permite listar todos os projectos definidos como públicos no Code.UA.

URL `http://code.ua.pt/projects.[xml|json]`

Método HTTP GET

Formato da Resposta XML / JSON

Exemplo `http://code.ua.pt/projects.xml`

Resposta

```
<?xml version="1.0" encoding="utf-16"?>
<projects type="array" offset="0" total_count="53" limit="25">
  <project>
    <id>562</id>
    <name>API</name>
    <identifier>api</identifier>
    <description>Academic Playground &amp;amp; Innovation</description>
    <created_on>2011-07-28T17:02:22+01:00</created_on>
    <updated_on>2011-11-10T16:10:19+00:00</updated_on>
  </project>
  (...)
</projects>
```

A.2.4 Projecto

À semelhança das operações dedicadas às ocorrências, também é possível listar um único projecto. Caso o projecto pretendido esteja definido como privado, é necessária uma prévia autenticação através do IdP da UA.

URL `http://code.ua.pt/projects/[id].[xml|json]`

Método HTTP GET

Formato da Resposta XML / JSON

Argumentos:

- `include`: Quando definido, o argumento `include` permite adicionar à listagem do projecto algumas categorias que estejam activas no mesmo. Categorias possíveis: `trackers` e `issue_categories`.

Exemplo `http://code.ua.pt/projects/api.json?include=trackers`

Resposta

```
{
  "project": {
    "trackers": [
      {
        "name": "Bug",
        "id": 1
      },
```

```
      {
        "name": "Funcionalidade",
        "id": 2
      },
      {
        "name": "Suporte",
        "id": 3
      }
    ],
    "created_on": "2011/07/28 17:02:22 +0100",
    "description": "Academic Playground & Innovation",
    "homepage": "http://api.web.ua.pt/",
    "identifier": "api",
    "name": "API",
    "updated_on": "2011/11/10 16:10:19 +0000",
    "id": 562
  }
}
```

A.2.5 Entradas Temporais

Esta operação permite listar todas as entras temporais associadas a projectos públicos.

URL `http://code.ua.pt/time_entries.[xml|json]`

Método HTTP GET

Formato da Resposta XML / JSON

Exemplo `http://code.ua.pt/time_entries.xml`

Resposta

```
<?xml version="1.0" encoding="utf-16"?>
<time_entries type="array">
  <time_entry>
    <id>163</id>
    <project name="weDraw" id="504" />
    <issue id="1087" />
    <user name="Paulo Coelho dos Santos de Lima Alves" id="709" />
    <activity name="Desenvolvimento" id="9" />
    <hours>1.0</hours>
    <comments></comments>
    <spent_on>2011-05-31</spent_on>
    <created_on>2011-05-31T14:48:54+01:00</created_on>
    <updated_on>2011-05-31T14:48:54+01:00</updated_on>
  </time_entry>
</time_entries>
```

A.2.6 Notícias

Esta operação permite obter todas as notícias associadas a projectos públicos.

URL `http://code.ua.pt/news.[xml|json]`

Método HTTP GET

Formato da Resposta XML / JSON

Exemplo `http://code.ua.pt/news.xml`

Resposta

```
<?xml version="1.0" encoding="utf-16"?>
<news total_count="19" type="array" offset="0" limit="25">
  <news>
    <id>160</id>
    <project name="labmm4_ta_g04" id="1045" />
    <author name="Rui Filipe Raposo Vidal" id="810" />
    <title>Site LabMM4_g04</title>
    <summary>Site dos exercícios de LabMM4 do grupo 04</summary>
    <description>Está online o site do grupo 04, utiliza jquery, html5 e php.</
        description>
    <created_on>2012-02-18T21:23:22+00:00</created_on>
  </news>
  (...)
</news>
```

A.2.7 Notícias do Projecto

Também é possível listar apenas as notícias de um determinado projecto.

URL `http://code.ua.pt/projects/[id]/news.[xml|json]`

Método HTTP GET

Formato da Resposta XML / JSON

Exemplo `https://code.ua.pt/projects/codeua/news.xml`

Resposta

```
<?xml version="1.0" encoding="utf-16"?>
<news type="array" offset="0" total_count="5" limit="25">
  <news>
    <id>143</id>
    <project name="Code.UA" id="1" />
    <author name="João Paulo Silva Barraca" id="8" />
    <title>1000 Utilizadores Activos!</title>
    <summary></summary>
    <description>O CodeUA está de parabéns! Criado à pouco mais de um ano com o objetivo
         de promover a utilização de ferramentas de gestão de projetos, o CodeUA
        atingiu os 1000 utilizadores activos. Suportamos actualmente Webdav, GIT e SVN
        para mais de 700 projetos pessoais, de disciplinas ou de investigação. Como
        sempre, comentários relacionados com novas funcionalidades ou problemas
        existentes são bem vindos. Para isso, utilizem o projecto CodeUA. Obrigado por
        utilizarem este serviço. Esperamos que o continuem a achar útil. </description>
    <created_on>2011-10-27T15:26:48+01:00</created_on>
  </news>
  (...)
</news>
```

A.3 Guia de Acesso

A.3.1 Cursos

Uma das operações disponível é o cursos, que permite obter uma listagem de todos os cursos da UA.

URL `http://acesso.ua.pt/xml/cursos.asp`

Método HTTP GET

Formato da Resposta XML

Argumentos:

- `lg`: Linguagem da resposta. Valores possíveis: **`pt`** (Português), `en` (Inglês).
- `c`: Ciclo de estudos. Valores possíveis: `1`, `2`, `3`.

Resposta

```
<?xml version="1.0" encoding="utf-16"?>
<cursos>
  <curso>
    <guid>23</guid>
    <codigo>0300/9361</codigo>
    <nome>Engenharia de Computadores e Telemática</nome>
    <grau>Mestrado Integrado</grau>
    <ciclo>1</ciclo>
    <departamentos>
      <departamento>Departamento de Electrónica, Telecomunicações e Informática</
          departamento>
    </departamentos>
    <areascientificas>
      <areacientifica>Informática</areacientifica>
    </areascientificas>
    <areasformacao>
      <areaformacao>Ciências da Engenharia e Tecnologias</areaformacao>
    </areasformacao>
    <regime>laboral</regime>
    <local>Campus Universitário de Santiago, Aveiro</local>
    <provas>Matemática A (19)</provas>
    <m23>23</m23>
  </curso>
  (...)
</cursos>
```

A.3.2 Cursos Param

Semelhante à operação Cursos (A.3.1),a operação cursos param permite obter uma versão compacta da lista dos cursos da UA.

URL `http://acesso.ua.pt/xml/cursos_param.asp`

Método HTTP GET

Formato da Resposta XML

Argumentos:

- `d`: Identificador do departamento.
- `g`: Grau.
- `o`: Prioridade de ordenação. Valores possíveis: `d` (ordenação por departamento), `grau` (ordenação por grau).

Resposta

```
<?xml version="1.0" encoding="utf-16"?>
<cursos>
  <curso>
    <guid>23</guid>
    <codigo>0300/9361</codigo>
    <nome>Engenharia de Computadores e Telemática</nome>
    <grau>Mestrado Integrado</grau>
    <departamento>Departamento de Electrónica, Telecomunicações e Informática</
        departamento>
    <area>Ciências da Engenharia e Tecnologias</area>
  </curso>
  (...)
</cursos>
```

A.3.3 Curso

Esta operação permite obter os detalhes sobre um dado curso.

URL `http://acesso.ua.pt/xml/curso.asp`

Método HTTP GET

Formato da Resposta XML

Argumentos:

- `i`: Identificador do curso.
- `lg`: Linguagem da resposta. Valores possíveis: **`pt`** (Português), `en` (Inglês).

Exemplo `http://acesso.ua.pt/xml/curso.asp?i=23`

Resposta

```
<?xml version="1.0" encoding="utf-16"?>
<curso>
  <guid>23</guid>
  <nome><![CDATA[Engenharia de Computadores e Telemática]]></nome>
  <imagem><![CDATA[http://acesso.ua.pt/upload/imagens/c_23.jpg]]></imagem>
  <codigo>0300/9361</codigo>
  <grau><![CDATA[Mestrado Integrado]]></grau>
  <ciclo>1</ciclo>
  <duracao>5 anos lectivos, 10 semestres (300 ECTS)</duracao>
  <ano_inicio>2006/2007</ano_inicio>
  <regime><![CDATA[laboral]]></regime>
  <horario><![CDATA[]]></horario>
  <propinas><![CDATA[999,71?/ano]]></propinas>
  <subsistema><![CDATA[Universitário]]></subsistema>
  <areas>
```

```
16      <area><![CDATA[Informática]]></area>
17    </areas>
18    <director>
19      <nome>Prof. Doutor Joaquim João Estrela Ribeiro Silvestre Madeira</nome>
20      <email>jmadeira@ua.pt</email>
21    </director>
22    <departamentos>
23      <departamento>
24        <nome>Departamento de Electrónica, Telecomunicações e Informática</nome>
25        <url>http://www.ua.pt/deti</url>
26      </departamento>
27    </departamentos>
28    <medias>
29      <fase1>136</fase1>
30      <fase2>0</fase2>
31      <fase3>0</fase3>
32    </medias>
33    <vagas>
34      <fase1>70</fase1>
35      <fase2>0</fase2>
36      <fase3>0</fase3>
37    </vagas>
38    <provas><![CDATA[Matemática A (19)]]></provas>
39    <ano_info><![CDATA[2011/2012]]></ano_info>
40    <saidas_profissionais><![CDATA[Os Mestres em Engenharia de Computadores e Telem&
          aacute;tica est&atilde;o preparados para intervir nas novas &aacute;reas que
          integram as tecnologias de informa&ccedil;&atilde;o, a ci&ecirc;ncia e engenharia
           de computadores e as telecomunica&ccedil;&otilde;es. <p>Assim, poder&atilde;o
          desempenhar fun&ccedil;&otilde;es em organiza&ccedil;&otilde;es p&uacute;blicas e
           privadas que realizem projectos de investiga&ccedil;&atilde;o, desenvolvimento e
           integra&ccedil;&atilde;o/utiliza&ccedil;&atilde;o de novas tecnologias da
          informa&ccedil;&atilde;o e comunica&ccedil;&atilde;o para a cria&ccedil;&atilde;o
           de novos servi&ccedil;os ligados &agrave; economia digital, de sistemas de
          informa&ccedil;&atilde;o ou para o projecto de equipamento.</p><p>Ser&atilde;o
          assim respons&aacute;veis tecnicamente por projectar, adaptar ou gerir solu&
          ccedil;&otilde;es, sistemas e redes inform&aacute;ticas complexas. As sa&iacute;
          das profissionais s&atilde;o m&uacute;ltiplas desde o sector dos servi&ccedil;os
          - onde &eacute; de real&ccedil;ar a banca, os seguros, o turismo, a cultura, o
          multim&eacute;dia - ao sector industrial, passando pela Administra&ccedil;&atilde
          ;o P&uacute;blica Central, Local e Regional.</p>]]></saidas_profissionais>
41    <objectivos><![CDATA[&Eacute; indiscut&iacute;vel, hoje, que a independ&ecirc;ncia e
          o desenvolvimento de um pa&iacute;s passa grandemente pela capacidade de se
          autonomizar do ponto de vista tecnol&oacute;gico. Mat&eacute;rias como os
          Computadores, Telem&aacute;tica e Inform&aacute;tica s&atilde;o inevitavelmente
          de import&acirc;ncia primordial, quer pela sua relev&acirc;ncia como elementos
          coadjuvantes de outras t&eacute;cnicas, quer pela sua interven&ccedil;&atilde;o
          directa em sistemas que condicionam cada vez mais a vida das pessoas.
          Computadores, redes de telecomunica&ccedil;&otilde;es, sistemas embutidos, telem&
          oacute;veis s&atilde;o alguns dos exemplos importantes. <p>O curso de Mestrado
          Integrado em Computadores e Telem&aacute;tica da Universidade de Aveiro (MIECT-UA
          ) tem por finalidade fornecer uma forma&ccedil;&atilde;o s&oacute;lida em ci&
          ecirc;ncias de base e de espectro largo nas &aacute;reas das tecnologias da
          computa&ccedil;&atilde;o, informa&ccedil;&atilde;o e das telecomunica&ccedil;&
          otilde;es, as quais t&ecirc;m vindo a assumir um papel dominante no sector
          industrial e o principal motor de desenvolvimento nas economias avan&ccedil;adas.
          </p><p>A tradi&ccedil;&atilde;o da Universidade de Aveiro neste dom&iacute;nio &
          eacute; forte. Aqui nasceu, por exemplo, o Servi&ccedil;o de Apontadores
          Portugueses (SAPO). A escolha de Aveiro como primeira Cidade Digital reflecte
          tamb&eacute;m o capital de conhecimentos adquiridos nesta &aacute;rea, n&atilde;o
           s&oacute; pela Universidade como tamb&eacute;m por outras institui&ccedil;&
          otilde;es da cidade. Nos pr&oacute;ximos anos, os projectos que, no &acirc;mbito
          do programa Cidades Digitais, aqui se forem desenvolvendo dar&atilde;o, por certo
          , um excelente enquadramento &agrave; forma&ccedil;&atilde;o fornecida pela
          Mestrado Integrado em Computadores e Telem&aacute;tica.</p><p>A responsabilidade
          por esta licenciatura &eacute; assumida pelo Departamento de Electr&oacute;nica,
          Telecomunica&ccedil;&otilde;es e Inform&aacute;tica da Universidade de Aveiro.
          Fundado em 1974, este departamento &eacute; detentor de um prestigiante historial
           nas &aacute;reas dos computadores e das telecomunica&ccedil;&otilde;es. O curr&
          iacute;culo do curso recorre, pois, &agrave;s compet&ecirc;ncias adquiridas para
```

```
      construir um perfil que fornece uma s&oacute;lida forma&ccedil;&atilde;o geral
      nas &aacute;reas da Engenharia Inform&aacute;tica.<br  >< p>]]>< objectivos>
42  <observacoes><  CDATA É conferido o grau de licenciado em Ciências da Engenharia de
      Computadores e Telemática aos alunos que tenham realizado os     ECTS
      correspondentes aos primeiros seis semestres curriculares de trabalho   ><
      observacoes>
43  <testemunho><  CDATA   >< testemunho>
44  <pre_requisitos><  CDATA   >< pre_requisitos>
45  <local><  CDATA Campus Universitário de Santiago  Aveiro  >< local>
46  <urlmaisinfo><  CDATA http     >< urlmaisinfo>
47  <urlcandidaturas><  CDATA http     >< urlcandidaturas>
48  <urlmobilidade><  CDATA http     >< urlmobilidade>
49  <m23>
50    <provas><  CDATA Matemática  a href "http://www.ua.pt/ensino/PageText.aspx?id=15132"
        >programa e exames< a>]]>< provas>
51    <vagas><  CDATA   >< vagas>
52    <urlinfo><  CDATA http   www ua t m23   >< urlinfo>
53    <observacoes><  CDATA Júri br> Presidente: Lucília Tavares dos Santos<br> Director:
        Joaquim Silvestre Madeira<br> Vogal: Ernesto Ventura Martins<br> Vogal (
        Suplente): Alexandre Nunes da Mota<br> Vogal (Suplente): Joaquim de Sousa Pinto
         ]]>< observacoes>
54  < m23>
55  <posgraduacoes>
56    <periodo><  CDATA   >< periodo>
57    <corpo_docente><  CDATA   >< corpo_docente>
58    <destinatarios><  CDATA   >< destinatarios>
59    <vagas><  CDATA   >< vagas>
60    <vagas_obs><  CDATA   >< vagas_obs>
61    <urlmobilidade><  CDATA   >< urlmobilidade>
62    <ano_info><  CDATA   >< ano_info>
63    <edital><  CDATA   >< edital>
64  < posgraduacoes>
65 < curso>
```

A.3.4 Plano

Operação que permite obter o plano de um determinado curso.

URL `http://acesso.ua.pt/xml/plano.asp`

Método HTTP GET

Formato da Resposta XML

Argumentos:

- `i`: Identificador do curso.
- `lg`: Linguagem da resposta. Valores possíveis: **`pt`** (Português), `en` (Inglês).

Exemplo `http://acesso.ua.pt/xml/plano.asp?i=23`

Resposta

```
1 <?xml version="1.0" encoding="utf-16"?>
2 <planoEstudos curso="23">
3   <ano n="1">
4     <semestre n="1">
5       <disciplina guid="2193">
6         <nome>Álgebra Linear</nome>
7         <id>2193</id>
```

```
      <area>M</area>
      <cargahoraria>
        <horas tipo="TP">5</horas>
      </cargahoraria>
      <creditos>8</creditos>
      <percurso></percurso>
      <minor></minor>
    </disciplina>
    (...)
  </semestre>
 </ano>
 (...)
</planoEstudos>
```

A.3.5 Unidade Curricular

Esta operação permite obter informações detalhadas sobre uma determinada unidade curricular.

URL `http://acesso.ua.pt/xml/uc.asp`

Método HTTP GET

Formato da Resposta XML

Argumentos:

- `i`: Identificador da unidade curricular.
- `p`: Código do PACO da unidade curricular.
- `lg`: Linguagem da resposta. Valores possíveis: **`pt`** (Português), `en` (Inglês).

Exemplo `http://acesso.ua.pt/xml/uc.asp?i=6304`

Resposta

```
<?xml version="1.0" encoding="utf-16"?>
<disciplina>
  <guid>6304</guid>
  <nome>Engenharia de Serviços</nome>
  <paco>42578</paco>
  <codigo>42578</codigo>
  <creditos>6</creditos>
  <objectivos><![CDATA[&lt;p&gt;Fornecer uma vis&amp;atilde;o global dos aspectos
      fundamentais que norteiam o desenvolvimento de arquitecturas de servi&amp;ccedil;
      o em telecomunica&amp;ccedil;&amp;otilde;es. Apresentar arquitecturas de
      desenvolvimento de software e servi&amp;ccedil;os em redes avan&amp;ccedil;adas.&
      lt;/p&gt;]]></objectivos>
  <conteudos><![CDATA[<ol> <li>Arquitecturas de Telecomunica&ccedil;&otilde;es
      orientadas a servi&ccedil;o.</li> <li>Estandardiza&ccedil;&atilde;o de Servi&
      ccedil;os em Telecomunica&ccedil;&otilde;es (3GPP, OMA) </li> <li>Tecnologias
      para suporte a desenvolvimento de servi&ccedil;os: SMS, WAP, MMS, SIP IMS.</li> <
      li>Servi&ccedil;os em redes SOA: Instant Messaging, Presence, virtual PABX, servi
      &ccedil;os de localiza&ccedil;&atilde;o.</li> <li>Tecnologias e APIs para SOA:
      XMPP, OSA/Parlay, Parlay-X, SIPServlets</li> <li>Web services e Widgets para
      ambientes de telecomunica&ccedil;&otilde;es.</li> <li>Sistemas de mensagens.</li>
       </ol>]]></conteudos>
  <avaliacao><![CDATA[&lt;p&gt;Avalia&amp;ccedil;&amp;atilde;o mista:&lt;/p&gt; &lt;p&
      gt;Exame Te&amp;oacute;rico (30%) + Avalia&amp;ccedil;&amp;atilde;o Continua
      (20%) + 1 Projecto (50%)&lt;/p&gt;]]></avaliacao>
```

```
11  <requisitos><![CDATA[&lt;p&gt;Conceitos de Redes e de Programa&amp;ccedil;&amp;atilde
        ;o Distribuida&lt;/p&gt;]]></requisitos>
12  <metodologia><![CDATA[&lt;p&gt;Ensino baseado em aulas te&amp;oacute;ricas e
        componente pr&amp;aacute;tica, com projecto de sintese acompanhado.&lt;/p&gt;]]><
        /metodologia>
13  <bibliografiabase><![CDATA[SOA Design Patterns / T. Erl / 2009 Understanding SIP
        Servlets 1.1" / C. Boulton, A. Kristoffer, B. Gronowski/ 2009 The 3G IP
        Multimedia Subsystem (IMS): Merging the Internet and the Cellular Worlds / G.
        Camarillo, M. Garcia-Martin / 2006]]></bibliografiabase>
14  <bibliografia><![CDATA[&lt;ul&gt; &lt;li&gt;T. Erl, &amp;ldquo;SOA Design Patterns&
        amp;rdquo;, Prentice Hall, ISBN 0136135161, 2009&lt;/li&gt; &lt;li&gt;C. Boulton,
         A. Kristoffer, B. Gronowski, &amp;ldquo;Understanding SIP Servlets 1.1&amp;rdquo
        ;, Artech house, ISBN 978-1-59693-428-3, 2009&lt;/li&gt; &lt;li&gt;G. Camarillo,
        M. Garcia-Martin, &amp;ldquo;The 3G IP Multimedia Subsystem (IMS): Merging the
        Internet and the Cellular Worlds&amp;rdquo;, Wiley, ISBN 0470018186, 2006&lt;/li&
        gt; &lt;/ul&gt;]]></bibliografia>
15  <competencias><![CDATA[&lt;p&gt;Deve ser capaz de compreeender e discutir:&lt;/p&gt;
        &lt;p&gt;1.&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp; Arquitecturas de
        Telecomunica&amp;ccedil;&amp;otilde;es orientadas a servi&amp;ccedil;o.&lt;/p&gt;
         &lt;p&gt;2.&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp; Estandardiza&amp;
        ccedil;&amp;atilde;o de Servi&amp;ccedil;os em Telecomunica&amp;ccedil;&amp;
        otilde;es (3GPP, OMA)&lt;/p&gt; &lt;p&gt;3.&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;
        nbsp;&amp;nbsp; Tecnologias para suporte a desenvolvimento de servi&amp;ccedil;os
        : SMS, WAP, MMS, SIP IMS.&lt;/p&gt; &lt;p&gt;4.&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp
        ;nbsp;&amp;nbsp; Servi&amp;ccedil;os em redes SOA: Instant Messaging, Presence,
        virtual PABX, servi&amp;ccedil;os de localiza&amp;ccedil;&amp;atilde;o.&lt;/p&gt;
         &lt;p&gt;5.&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp; Tecnologias e APIs
         para SOA: XMPP, OSA/Parlay, Parlay-X, SIPServlets&lt;/p&gt; &lt;p&gt;6.&amp;nbsp
        ;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp; Web services e Widgets para ambientes
        de telecomunica&amp;ccedil;&amp;otilde;es.&lt;/p&gt; &lt;p&gt;7.&amp;nbsp;&amp;
        nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp; Sistemas de mensagens.&lt;/p&gt; &lt;p&gt;
        Deve ser capaz de desenvolver servi&amp;ccedil;os em termos pr&amp;aacute;ticos&
        lt;/p&gt;]]></competencias>
16  <area>Electrotecnia / Telecomunicações</area>
17  <carga>0|3|0</carga>
18  <responsavel>
19    <nome>Diogo Nuno Pereira Gomes</nome>
20    <email>dgomes@ua.pt</email>
21  </responsavel>
22 </disciplina>
```

A.3.6 Departamentos

Operação que permite listar os departamentos existentes na UA.

URL `http://acesso.ua.pt/xml/deps.asp`

Método HTTP GET

Formato da Resposta XML

Argumentos:

- `lg`: Linguagem da resposta. Valores possíveis: **`pt`** (Português), `en` (Inglês).

Resposta

```
1 <?xml version="1.0" encoding="utf-16"?>
2 <departamentos>
3   (...)
4   <departamento>
```

```
    <guid>1</guid>
    <nome>Departamento de Electrónica, Telecomunicações e Informática</nome>
    <sigla>DET</sigla>
    <url>http://www.ua.pt/deti</url>
  </departamento>
  (...)
</departamentos>
```

A.3.7 Graus

Operação permite obter uma lista dos graus académicos existentes na UA.

URL `http://acesso.ua.pt/xml/graus.asp`

Método HTTP GET

Formato da Resposta XML

Argumentos:

- `lg`: Linguagem da resposta. Valores possíveis: **`pt`** (Português), `en` (Inglês).

Resposta

```
<?xml version="1.0" encoding="utf-16"?>
<graus>
  <grau>
    <guid>2</guid>
    <nome>Licenciatura</nome>
  </grau>
  (...)
</graus>
```

A.4 Jornal Online UA

A.4.1 Conteúdos

Esta operação permite obter um resumo das notícias, podendo estar incluídas imagens no texto (através da respectiva *tag* no formato HTML) [Pereira 2011b].

URL `http://uaonline.ua.pt/xml/contents_xml.asp`

Método HTTP GET

Formato da Resposta XML

Argumentos:

- `lid`: Identificador da língua pretendida. Valor por defeito: **`18`** (Português). Consultar Linguagens (A.4.2).

- `dt`: *Flag* indicadora da inclusão de destaques (`0` ou `1`). Valor por defeito: **0**.
- `n`: Número máximo de notícias a serem listadas. Valor por defeito: **6**.
- `di`: data de início.
- `df`: Data de fim.
- `i`: Identificador da categoria do conteúdo. Consultar Categorias (A.4.3).
- `d`: Identificador do departamento. Consultar Departamentos (A.4.4).

Exemplo `http://uaonline.ua.pt/xml/contents_xml.asp?n=2`

Resposta

```
<?xml version="1.0" encoding="utf-16"?>
<?xml-stylesheet type="text/xsl" href="style.xsl" version="1.0"?>
<rss version="2.0">
  <channel>
    <title>@UA_online</title>
    <link>http://uaonline.ua.pt</link>
    <description>Jornal Online da Universidade de Aveiro</description>
    <language>pt-pt</language>
    <copyright>2012Universidade de Aveiro</copyright>
    <lastBuildDate>Fri, 13 Apr 2012 15:24 GMT</lastBuildDate>
    <ttl>2</ttl>
    <image>
      <title>www.ua.pt</title>
      <url>http://uaonline.ua.pt/images/logo_ua_44.gif</url>
      <link>http://uaonline.ua.pt</link>
      <width>44</width>
      <height>44</height>
    </image>
    <item>
      <guid>http://uaonline.ua.pt/detail.asp?c=23722</guid>
      <title>Novo ranking de Leiden destaca boa performance da UA</title>
      <link>http://uaonline.ua.pt/detail.asp?c=23722</link>
      <description>&lt;img src="http://uaonline.ua.pt/upload/img/thumb_img_c2_9638.jpg"
          alt="" title="" /&gt;A UA é a universidade portuguesa que consegue um impacto
          mais elevado com a sua produção científica (artigos publicados), no período
          2011/2012, de acordo com os critérios definidos no Ranking da Universidade de
          Leiden, posição cimeira que repete a avaliação do ranking do Times Higher
          Education (THE) feita em outubro do ano passado. De resto, o ranking de Leiden
           atribui boas prestações às universidades portuguesas, mesmo no contexto
          internacional.</description>
      <pubDate>Fri, 13 Apr 2012 00:00 GMT</pubDate>
    </item>
    (...)
  </channel>
</rss>
```

A.4.2 Linguagens

Operação que permite obter uma lista das linguagens disponíveis para os conteúdos.

URL `http://uaonline.ua.pt/xml/languages_xml.asp`

Método HTTP GET

Formato da Resposta XML

Resposta

```
<?xml version="1.0" encoding="utf-16"?>
<linguas>
  <item>
    <guid>18</guid>
    <nome>português</nome>
    <sigla>pt</sigla>
  </item>
  <item>
    <guid>19</guid>
    <nome>inglês</nome>
    <sigla>en</sigla>
  </item>
  <item>
    <guid>43</guid>
    <nome>espanhol</nome>
    <sigla>es</sigla>
  </item>
</linguas>
```

A.4.3 Categorias

Esta operação permite obter uma lista das categorias de conteúdos disponíveis para um dado site, e numa determinada linguagem.

URL `http://uaonline.ua.pt/xml/categs_xml.asp`

Método HTTP GET

Formato da Resposta XML

Argumentos:

- `lid`: Identificador da linguagem pretendida. Valor por defeito: **`18`** (Português). Consultar Linguagens (A.4.2).

Resposta

```
<?xml version="1.0" encoding="utf-16"?>
<categorias>
 <item>
  <guid>60</guid>
  <nome>notícias</nome>
 </item>
 <item>
  <guid>67</guid>
  <nome>Distinções</nome>
 </item>
 (...)
</categorias>
```

A.4.4 Departamentos

Operação que permite obter uma lista dos departamentos aos quais as notícias são referenciadas.

URL `http://uaonline.ua.pt/xml/deps_xml.asp`

Método HTTP GET

Formato da Resposta XML

Resposta

```
<?xml version="1.0" encoding="utf-16"?>
<departamentos>
 <item>
  <guid>0</guid>
  <nome>@UA_online</nome>
  <sigla>uaonline</sigla>
 </item>
 <item>
  <guid>1</guid>
  <nome>Departamento de Ambiente e Ordenamento</nome>
  <sigla>dao</sigla>
 </item>
 (...)
</departamentos>
```

A.4.5 Banners

Operação que permite obter os *banners* publicitários institucionais. É também possível obter os *banners* publicitários associados a um determinado departamento.

URL `http://uaonline.ua.pt/xml/banners_xml.asp`

Método HTTP GET

Formato da Resposta XML

Argumentos:

- `d`: Identificador do departamento. Valor por defeito: **0** (*banners* do Jornal Online). Consultar Departamentos (A.4.4).

Resposta

```
<?xml version="1.0" encoding="utf-16"?>
<pub>
 <item>
  <guid>551</guid>
  <descricao>forum 3e_2012</descricao>
  <url_ficheiro>http://uaonline.ua.pt/upload/pub/banner_551.swf</url_ficheiro>
  <url_site>http://</url_site>
  <posicao>0</posicao>
  <peso>10</peso>
 </item>
 <item>
  <guid>547</guid>
  <descricao>revista turismo</descricao>
  <url_ficheiro>http://uaonline.ua.pt/upload/pub/banner_547.swf</url_ficheiro>
  <url_site>http://</url_site>
  <posicao>0</posicao>
  <peso>1</peso>
 </item>
 (...)
</pub>
```

A.4.6 Pesquisa

Esta operação permite efectuar pesquisas de texto nos campos `Título`, `Ante-título` e `Resumo`, tanto nas notícias como nos eventos.

URL `http://uaonline.ua.pt/xml/pesquisa_xml.asp`

Método HTTP GET

Formato da Resposta XML

Argumentos:

- `str`: Texto a ser pesquisado.
- `lid`: Identificador da linguagem pretendida. Valor por defeito: **`18`** (Português). Consultar Linguagens (A.4.2).
- `d`: Identificador do departamento. Valor por defeito: **`0`** (*banners* do Jornal Online). Consultar Departamentos (A.4.4).
- `di`: Permite especificar uma data de início para os resultados da pesquisa. A data deve estar no formato `dd-mm-aaaa`. Valor por defeito: 1 ano precedente à data da pesquisa.
- `df`: Permite especificar uma data de fim para os resultados da pesquisa. A data deve estar no formato `dd-mm-aaaa`. Valor por defeito: 1 ano procedente à data da pesquisa.
- `t`: Tipo de pesquisa a ser efectuada:
  - **`and`**: Devolve os resultados de todas as palavras presentes no argumento `str`.
  - `or`: Devolve os resultados que contenham qualquer uma das palavras presentes no argumento `str`.
- `e`: Indica a categoria onde deve ser feita a pesquisa:
  - **`j`**: Notícias.
  - `e`: Eventos.

Exemplo `http://uaonline.ua.pt/xml/pesquisa_xml.asp?str=deti&lid=18`

Resposta

```
<?xml version="1.0" encoding="utf-16"?>
<resultados>
 <item guid="23571">
  <titulo><![CDATA[Robôs humanoides da UA à conquista do pódio na Alemanha]]></titulo>
  <data>3/29/2012</data>
  <url>http://uaonline.ua.pt/detail.asp?lg=pt&amp;c=23571</url>
 </item>
 <item guid="23005">
  <titulo><![CDATA[Novidades tecnológicas podem ser conhecidas no DETI]]></titulo>
  <data>2/1/2012</data>
  <url>http://uaonline.ua.pt/detail.asp?lg=pt&amp;c=23005</url>
 </item>
 (...)
</resultados>
```

A.4.7 Eventos

Através desta operação é possível obter os eventos disponíveis no Jornal Online. Estes eventos são devolvidos no formato vCalendar.

URL `http://uaonline.ua.pt/xml/events_vcal.asp`

Método HTTP GET

Formato da Resposta XML

Argumentos:

- `lid`: Identificador da linguagem pretendida. Valor por defeito: **`18`** (Português). Consultar Linguagens (A.4.2).
- `d`: Identificador do departamento. Valor por defeito: **`0`** (*banners* do Jornal Online). Consultar Departamentos (A.4.4).
- `di`: Permite especificar uma data de início para os resultados da pesquisa. A data deve estar no formato `dd-mm-aaaa`.
- `df`: Permite especificar uma data de fim para os resultados da pesquisa. A data deve estar no formato `dd-mm-aaaa`.
- `i`: Tipo de evento a ser devolvido.

Exemplo `http://uaonline.ua.pt/xml/events_vcal.asp?d=23&di=01-04-2012`

Resposta

```
<?xml version="1.0" encoding="utf-16"?>
<eventos>
 <evento guid="16692">
  BEGIN:VCALENDAR
    PRODID: ua_online
    VERSION: 1.0
    METHOD:PUBLISH
    BEGIN:VEVENT
      SUMMARY: Provas do Programa Doutoral em Música, de Ana Veloso
      DESCRIPTION: As Provas do Programa Doutoral em Música de Ana Luísa Setas Veloso,
          no âmbito do Programa Doutoral em Música, decorrem a 18 de abril, a partir das
           15h00, no Departamento de Engenharia Mecânica. «Voar até ao comboio dos
          Segredos: A construção de significados partilhados no desenvolvimento do
          pensamento musical em crianças do 1º ciclo do EB.» é o título da tese a
          apresentar.
      LOCATION:
      DTSTART: 4/18/2012 3:00:00 PM
      DTEND: 4/18/2012 11:59:00 PM
      CATEGORIES: Provas Académicas
    END:VEVENT
  END:VCALENDAR </evento>
 (...)
</eventos>
```

A.5 PACO

A.5.1 Horário do Aluno

Esta operação permite obter o horário escolar do aluno.

Operação ObterHorarioAluno

Formato da Resposta XML

Argumentos:

- `iupi`: identificador privado do aluno na UA.

Exemplo de Resposta

```
<?xml version="1.0" encoding="utf-16"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://
    www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ObterHorarioAlunoResponse xmlns="https://paco.ua.pt/">
      <ObterHorarioAlunoResult>
        <xsd:schema>
          <xs:schema xmlns="" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:msdata="
              urn:schemas-microsoft-com:xml-msdata" id="NewDataSet">
            <xs:element name="NewDataSet" msdata:IsDataSet="true">
              <xs:complexType>
                <xs:choice maxOccurs="unbounded">
                  <xs:element name="ObterHorarioAluno">
                    <xs:complexType>
                      <xs:sequence>
                        <xs:element name="CodDisciplina" type="xs:int" minOccurs="0" />
                        <xs:element name="NomeDisciplina" type="xs:string" minOccurs="0" />
                        <xs:element name="Dep" type="xs:string" minOccurs="0" />
                        <xs:element name="IDTurma" type="xs:int" minOccurs="0" />
                        <xs:element name="D_Turma" type="xs:string" minOccurs="0" />
                        <xs:element name="DiaSemana" type="xs:int" minOccurs="0" />
                        <xs:element name="HoraInicio" type="xs:decimal" minOccurs="0" />
                        <xs:element name="duracao" type="xs:decimal" minOccurs="0" />
                        <xs:element name="Sala" type="xs:string" minOccurs="0" />
                      </xs:sequence>
                    </xs:complexType>
                  </xs:element>
                </xs:choice>
              </xs:complexType>
            </xs:element>
          </xs:schema>
        </xsd:schema>
        <diffgr:diffgram xmlns:msdata="urn:schemas-microsoft-com:xml-msdata" xmlns:diffgr
            ="urn:schemas-microsoft-com:xml-diffgram-v1">
          <NewDataSet xmlns="">
            <ObterHorarioAluno diffgr:id="ObterHorarioAluno1" msdata:rowOrder="0">
              <CodDisciplina>43309</CodDisciplina>
              <NomeDisciplina>COMPUTAÇÃO RECONFIGURÁVEL</NomeDisciplina>
              <Dep>DET</Dep>
              <IDTurma>44338</IDTurma>
              <D_Turma>T1</D_Turma>
              <DiaSemana>2</DiaSemana>
              <HoraInicio>9.000000</HoraInicio>
              <duracao>2.000000</duracao>
              <Sala>ANF_V</Sala>
            </ObterHorarioAluno>
```

```
<ObterHorarioAluno diffgr:id="ObterHorarioAluno2" msdata:rowOrder="1">
  <CodDisciplina>43309</CodDisciplina>
  <NomeDisciplina>COMPUTAÇÃO RECONFIGURÁVEL</NomeDisciplina>
  <Dep>DET</Dep>
  <IDTurma>44340</IDTurma>
  <D_Turma>P3</D_Turma>
  <DiaSemana>3</DiaSemana>
  <HoraInicio>14.000000</HoraInicio>
  <duracao>2.000000</duracao>
  <Sala>4.1.32</Sala>
</ObterHorarioAluno>
</NewDataSet>
</diffgr:diffgram>
</ObterHorarioAlunoResult>
</ObterHorarioAlunoResponse>
</soap:Body>
</soap:Envelope>
```

A.5.2 Horário do Docente

Esta operação permite obter o horário do docente (leccionação e atendimento).

Operação ObterHorarioDocente

Formato da Resposta XML

Argumentos:

- `iupi`: identificador privado do docente na UA.

A.5.3 Turmas do Aluno

Esta operação permite obter uma lista das turmas do aluno.

Operação ObterListaTurmasAluno

Formato da Resposta XML

Argumentos:

- `iupi`: identificador privado do aluno na UA.

Exemplo de Resposta

```
<?xml version="1.0" encoding="utf-16"?>
<eventos>
 <evento guid="16692">
  BEGIN:VCALENDAR
    PRODID: ua_online
    VERSION: 1.0
    METHOD:PUBLISH
    BEGIN:VEVENT
      SUMMARY: Provas do Programa Doutoral em Música, de Ana Veloso
```

```
10      DESCRIPTION: As Provas do Programa Doutoral em Música de Ana Luísa Setas Veloso,
            no âmbito do Programa Doutoral em Música, decorrem a 18 de abril, a partir das
             15h00, no Departamento de Engenharia Mecânica. «Voar até ao comboio dos
            Segredos: A construção de significados partilhados no desenvolvimento do
            pensamento musical em crianças do 1º ciclo do EB.» é o título da tese a
            apresentar.
11      LOCATION:
12      DTSTART: 4/18/2012 3:00:00 PM
13      DTEND: 4/18/2012 11:59:00 PM
14      CATEGORIES: Provas Académicas
15    END:VEVENT
16  END:VCALENDAR </evento>
17  (...)
18 </eventos>
```

A.5.4 Turmas do Docente

Esta operação permite obter uma lista das turmas do docente.

Operação ObterListaTurmasDocente

Formato da Resposta XML

Argumentos:

- `iupi`: identificador privado do docente na UA.

A.5.5 Disciplinas do Aluno

Esta operação permite obter uma lista das disciplinas do aluno.

Operação ObterListaDisciplinasAluno

Formato da Resposta XML

Argumentos:

- `iupi`: identificador privado do aluno na UA.

Exemplo de Resposta

```
1 <?xml version="1.0" encoding="utf-16"?>
2 <soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://
      www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
3   <soap:Body>
4     <ObterListaDisciplinasAlunoResponse xmlns="https://paco.ua.pt/">
5       <ObterListaDisciplinasAlunoResult>
6         <xsd:schema>
7           <xs:schema xmlns="" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:msdata="
                urn:schemas-microsoft-com:xml-msdata" id="NewDataSet">
8             <xs:element name="NewDataSet" msdata:IsDataSet="true">
9               <xs:complexType>
10                <xs:choice maxOccurs="unbounded">
11                  <xs:element name="ObterListaDisciplinasAluno">
```

```
12                <xs:complexType>
13                  <xs:sequence>
14                    <xs:element name="CodDisciplina" type="xs:int" minOccurs="0" />
15                    <xs:element name="NomeDisciplina" type="xs:string" minOccurs="0" />
16                  </xs:sequence>
17                </xs:complexType>
18              </xs:element>
19            </xs:choice>
20          </xs:complexType>
21        </xs:element>
22      </xs:schema>
23    </xsd:schema>
24    <diffgr:diffgram xmlns:msdata="urn:schemas-microsoft-com:xml-msdata" xmlns:diffgr
          ="urn:schemas-microsoft-com:xml-diffgram-v1">
25      <NewDataSet xmlns="">
26        <ObterListaDisciplinasAluno diffgr:id="ObterListaDisciplinasAluno1" msdata:
              rowOrder="0">
27          <CodDisciplina>49952</CodDisciplina>
28          <NomeDisciplina>DISSERTAÇÃO</NomeDisciplina>
29        </ObterListaDisciplinasAluno>
30      </NewDataSet>
31    </diffgr:diffgram>
32  </ObterListaDisciplinasAlunoResult>
33 </ObterListaDisciplinasAlunoResponse>
34 </soap:Body>
35 </soap:Envelope>
```

A.5.6 Disciplinas do Docente

Esta operação permite obter uma lista das disciplinas do docente.

Operação ObterListaDisciplinasDocente

Formato da Resposta XML

Argumentos:

- `iupi`: identificador privado do docente na UA.

A.5.7 Dados do Aluno

Esta operação permite obter os dados pessoais do aluno (número mecanográfico, curso, ano curricular e fotografia no formato Base64).

Operação ObterDadosAluno

Formato da Resposta XML

Argumentos:

- `iupi`: identificador privado do aluno na UA.

Exemplo de Resposta

```xml
<?xml version="1.0" encoding="utf-16"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ObterHorarioAlunoResponse xmlns="https://paco.ua.pt/">
      <ObterHorarioAlunoResult>
        <xsd:schema>
          <xs:schema xmlns="" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:msdata="urn:schemas-microsoft-com:xml-msdata" id="NewDataSet">
            <xs:element name="NewDataSet" msdata:IsDataSet="true">
              <xs:complexType>
                <xs:choice maxOccurs="unbounded">
                  <xs:element name="ObterDadosAluno">
                    <xs:complexType>
                      <xs:sequence>
                        <xs:element name="NMec" type="xs:int" minOccurs="0" />
                        <xs:element name="Curso" type="xs:string" minOccurs="0" />
                        <xs:element name="AnoCurricular" type="xs:short" minOccurs="0" />
                        <xs:element name="Foto" type="xs:base64Binary" minOccurs="0" />
                      </xs:sequence>
                    </xs:complexType>
                  </xs:element>
                </xs:choice>
              </xs:complexType>
            </xs:element>
          </xs:schema>
        </xsd:schema>
        <diffgr:diffgram xmlns:msdata="urn:schemas-microsoft-com:xml-msdata" xmlns:diffgr="urn:schemas-microsoft-com:xml-diffgram-v1">
          <NewDataSet xmlns="">
            <ObterDadosAluno diffgr:id="ObterDadosAluno1" msdata:rowOrder="0">
              <NMec>33326</NMec>
              <Curso>8240 - MESTRADO INTEGRADO EM ENGENHARIA DE COMPUTADORES E TELEMÁTICA</Curso>
              <AnoCurricular>5</AnoCurricular>
              <Foto>Qk3WXwA(...)0AAA</Foto>
            </ObterDadosAluno>
          </NewDataSet>
        </diffgr:diffgram>
      </ObterHorarioAlunoResult>
    </ObterHorarioAlunoResponse>
  </soap:Body>
</soap:Envelope>
```

A.6 Ementas SASUA

```xml
<?xml version="1.0" encoding="utf-16"?>
<result request="/sas/ementas" request_timestamp="1334843931">
 <menus zone="rest" type="day">
  <menu canteen="Restaurante Universitário" meal="Almoço" date="Thu, 19 Apr 2012 14:58:51 +0100" weekday="Thursday" weekdayNr="4" disabled="0">
   <items>
    <item name="Sopa">Sopa de lentilhas</item>
    <item name="Prato de carne">Bifinhos de porco com ananás</item>
    <item name="Prato de peixe">Pescada à braz</item>
    <item name="Buffet de entradas">Buffet de entradas variadas</item>
    <item name="Buffet de saladas">Buffet de saladas variadas</item>
    <item name="Buffet de sobremesas">Buffet de sobremesas variadas</item>
   </items>
  </menu>
 </menus>
</result>
```

A.7 Senhas SAC

```
<?xml version="1.0" encoding="utf-16"?>
<result request="/sac/senhas" request_timestamp="1336950132">
  <items count="2">
    <item enabled="0" info="Fora de horário">
      <id>4593</id>
      <letter>A</letter>
      <desc>Lic. (1º ciclo), Mestrado (2º ciclo)</desc>
      <latest>134</latest>
      <ast>265</ast>
      <awt>1477</awt>
      <wc>1</wc>
      <post_number>3</post_number>
      <date>2012-01-27 16:09:04</date>
    </item>
    <item enabled="0" info="Fora de horário">
      <id>4592</id>
      <letter>A</letter>
      <desc>Lic. (1º ciclo), Mestrado (2º ciclo)</desc>
      <latest>133</latest>
      <ast>265</ast>
      <awt>1478</awt>
      <wc>2</wc>
      <post_number>4</post_number>
      <date>2012-01-27 16:07:21</date>
    </item>
  </items>
</result>
```

B Licenças dos Serviços Web

Nesta secção estão listadas as licenças aplicáveis a alguns serviços web desenvolvidos para a UA.

B.1 Ementas SASUA

```
<?xml version="1.0" encoding="utf-8"?>

<rdf:RDF xmlns="http://web.resource.org/cc/"
 xmlns:dc="http://purl.org/dc/elements/1.1/"
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">

 <Work rdf:about="http://services.web.ua.pt/sas/ementas">
  <dc:title>University of Aveiro's Menus RESTful API</dc:title>
  <dc:date>2011-12-20</dc:date>
  <dc:language>Portuguese</dc:language>
  <dc:description>A RESTful API to retrieve canteens' and snack bar's menus of University
      of Aveiro</dc:description>
  <dc:subject>API,REST,University of Aveiro,Menus,Canteen,Snack Bar,Restaurant</dc:subject
      >
  <dc:creator>
   <Organization>
    <dc:title>University of Aveiro</dc:title>
   </Organization>
  </dc:creator>
  <dc:publisher>
   <Organization>
    <dc:title>University of Aveiro</dc:title>
   </Organization>
  </dc:publisher>
  <dc:rights>
   <Organization>
    <dc:title>University of Aveiro</dc:title>
   </Organization>
  </dc:rights>

  <dc:identifier>http://services.web.ua.pt/sas/ementas</dc:identifier>
  <dc:format>text/xml</dc:format>
  <dc:type rdf:resource="http://purl.org/dc/dcmitype/Dataset" />
  <license rdf:resource="http://web.resource.org/cc/PublicDomain" />
 </Work>

 <License rdf:about="http://web.resource.org/cc/PublicDomain">
  <permits rdf:resource="http://web.resource.org/cc/Reproduction" />
  <permits rdf:resource="http://web.resource.org/cc/Distribution" />
  <permits rdf:resource="http://web.resource.org/cc/DerivativeWorks" />

  <requires rdf:resource="http://web.resource.org/cc/Attribution" />

  <prohibits rdf:resource="http://web.resource.org/cc/CommercialUse" />
 </License>

</rdf:RDF>
```

B.2 Senhas SAC

```
<?xml version="1.0" encoding="utf-8"?>

<rdf:RDF xmlns="http://web.resource.org/cc/"
      xmlns:dc="http://purl.org/dc/elements/1.1/"
```

```
5   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
6
7   <Work rdf:about="http://services.web.ua.pt/sac/senhas">
8     <dc:title>University of Aveiro's Academic Services Tickets RESTful API</dc:title>
9     <dc:date>2011-12-20</dc:date>
10    <dc:language>Portuguese</dc:language>
11    <dc:description>A RESTful API to retrieve tickts information from University of Aveiro's
          Academic Services</dc:description>
12    <dc:subject>API,REST,University of Aveiro,Academic Services,tickets</dc:subject>
13    <dc:creator>
14      <Organization>
15        <dc:title>University of Aveiro</dc:title>
16      </Organization>
17    </dc:creator>
18    <dc:publisher>
19      <Organization>
20        <dc:title>University of Aveiro</dc:title>
21      </Organization>
22    </dc:publisher>
23    <dc:rights>
24      <Organization>
25        <dc:title>University of Aveiro</dc:title>
26      </Organization>
27    </dc:rights>
28
29    <dc:identifier>http://services.web.ua.pt/sac/senhas</dc:identifier>
30    <dc:format>text/xml</dc:format>
31    <dc:type rdf:resource="http://purl.org/dc/dcmitype/Dataset" />
32    <license rdf:resource="http://web.resource.org/cc/PublicDomain" />
33  </Work>
34
35  <License rdf:about="http://web.resource.org/cc/PublicDomain">
36    <permits rdf:resource="http://web.resource.org/cc/Reproduction" />
37    <permits rdf:resource="http://web.resource.org/cc/Distribution" />
38    <permits rdf:resource="http://web.resource.org/cc/DerivativeWorks" />
39
40    <requires rdf:resource="http://web.resource.org/cc/Attribution" />
41
42    <prohibits rdf:resource="http://web.resource.org/cc/CommercialUse" />
43  </License>
44
45 </rdf:RDF>
```