Universidade de Aveiro Electrónica, Telecomunicações e Informática,
2010

Nuno José São Pedro Rosa

Serviços de gestão de dados pessoais em arquitectura "user-centric"

Nuno José São Pedro Rosa

Serviços de gestão de dados pessoais em arquitectura "user-centric"

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requesitos necessários à obtenção do grau de Mestre em Engenharia Electrónica e de Telecomunicações, realizada sob a orientação científica do Prof. Dr. Francisco Fontes, Professor Auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro e com colaboração do Mestre Ricardo Azevedo Pereira da PT Inovação.

o júri / the jury

presidente / president	**Professor Doutor Rui Luis Andrade Aguiar** Professor Auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro.
vogal - arguente principal	**Professor Doutor Paulo Alexandre Ferreira Simões** Professor Auxiliar do Departamento de Engenharia Informática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra
vogal - orientador	**Professor Doutor Francisco Manuel Marques Fontes** Professor Auxiliar convidado do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

agradecimentos / acknowledgements

Ao Professor Doutor Francisco Fontes e Mestre Ricardo Azevedo Pereira pela sua orientação e apoio ao longo do trabalho desenvolvido nesta Dissertação.

À minha família e amigos pela compreensão e apoio incondicional.

palavras-chave

XRI, XDI, OASIS, *semantic web*, *data portability*, *user-centric*, identificadores.

resumo

O paradigma da gestão de informação pessoal está a mudar. Até agora sempre esteve associado a uma gestão centrada nas organizações em que o utilizador cede os seus dados pessoais a uma entidade e delega, segundo os termos da organização, a sua gestão. Com o aparecimento de novos serviços *online* e a apetência cada vez maior dos indivíduos para a sua utilização surgem algumas preocupações como fragmentação, privacidade, interoperabilidade, para nomear alguns. Tecnologias *web* emergentes procuram trazer avanços em questões como "*semantic web*", "*trust computing*", "*data portability*", "*identity management*" com o intuito de mudar para um paradigma centrado no indivíduo em controlo da informação que lhe diz respeito mas mantendo a interoperabilidade para que estes objectivos se mantenham entre os múltiplos intervenientes naquilo que designa como a identidade digital.

Esta Dissertação de mestrado aborda esta temática segundo dois aspectos: (i) a identificação e descoberta dos vários serviços associados à identidade digital através de um identificador abstracto (ii) a representação de dados, num formato *machine-readable*, agregação e partilha de dados de uma forma consistente e governada pelo utilizador. Como prova de conceito foi desenvolvido um protótipo baseado no *framework* XRI/XDI com o propósito de servir como adaptador para serviços de redes sociais permitindo a descoberta, agregação e partilha de dados.

O protótipo consiste em três módulos, aplicação cliente, módulo servidor e *endpoint* XDI. A aplicação cliente permite, através de uma interface *web*, associar contas de serviços e gerir relações/acessos sobre os dados do utilizador. O módulo de servidor contém a lógica necessária para processar os pedidos recebidos pelo cliente ou *endpoint* nos dados do utilizador. O *endpoint* XDI permite que qualquer aplicação com capacidades XDI aceda, se autorizado, aos dados do utilizador incluindo aqueles, através de mapeamento XDI, que se encontram guardados remotamente num dos serviços associados. A implementação realizada evidencia algumas das vantagens do *framework* XRI/XDI mesmo em coexistência com os sistemas actuais.

keywords XRI, XDI, OASIS, *semantic web*, *data portability*, *user-centric*, *identifiers*.

abstract The personal information management paradigm is shifting. Until now, it's was always associated with organizations. Individuals hand over their personal data and delegate the management, under organizational policies, to an external entity. With the spread of new online services and the increasing acceptance of their value by individuals, some concerns come along: fragmentation, privacy, interoperability, to name few. Emergent web technologies seek progress on concepts like "Sematic Web", "Trust Computing", and "Data Portability" and "Identity Management" to a paradigm centered in the individual. This paradigm allows total control over the information individual shares online without compromising interoperability between services, i.e., building a unified digital identity.

This thesis addresses these concerns in two goals. The identification and discovery of all the online services associated with the individual digital identity through a digital identifier and abstract representation of data, a machine-readable format, aggregating and sharing data in a consistent way and governed by the user. As a proof-of-concept was developed a prototype based on the framework XRI/XDI to serve as an interface to social network services allowing discovery, aggregation and data sharing. The prototype consists of three modules, client, server module and XDI endpoint. The client application allows, through a web interface, associate social web accounts and manage granular relationships/accesses to user data. The server module contains logic needed to process requests from the web interface or XDI endpoint and synchronize data with social network services. The XDI endpoint allows that any application with XDI capabilities to access, if authorized, to user data represented in XDI, including remote data located on social network. The work developed on the proof-of-concepts show some of the framework XRI/XDI advantages even in coexistence with current "walled garden" systems.

...

Conteúdo

Lista de Figuras

Lista de Tabelas

Acrónimos

AJAX Asynchronous Javascript And XML

API Application Programming Interface

ARPANET Advanced Research Projects Agency Network

CRM Customer Relationship Management

DNSKEY DNS Public Key

DNSSEC DNS Security

DNS Domain Name System

DS Delegation Signer

DoS Denial of Service

GCS Global Context Symbol

GWT Google Web Toolkit

HTML HyperText Markup Language

HTTPS Hypertext Transfer Protocol Secure

HTTP Hypertext Transfer Protocol

HXRI HTTP XRI

ICANN Internet Corporation for Assigned Names and Numbers

IETF Internet Engineering Task Force

IM Instant Messaging

IP Internet Protocol

IRI Internationalized Resource Identifier

ISP Internet Service Provider

JVM Java Virtual Machine

NSEC Next Secure

OASIS Organization for the Advancement of Structured Information Standards

OP OpenID Provider

OWL Web Ontology Language

PDF Portable Document Format

PDS Personal Data Services

PKI Public Key Infrastructure

RDF Resource Description Framework

RP Relying Party

RPC Remote Procedure Call

RRSIG Resource Record Signature

RR Resource Record

SAML Security Assertion Markup Language

SEP Service Endpoint

SOA Start of Authority

SO Sistema Operativo

SRI-NIC Standford Research Institute - Network Information Center

SSL Secure Sockets Layer

SSO Single Sign On

TLD Top Level Domain

TLS Transport Layer Security

TTL Time To Live

UDP User Datagram Protocol

URI Uniform Resource Identifier

URL Uniform Resource Locator

URN Uniform Resource Name

VRM Vendor Relationship Management

W3C World Wide Web Consortium

XDI XRI Data Interchange

XML eXtensible Markup Language

XRDS eXtensible Resource Descriptor Sequence

XRD eXtensible Resource Descriptor

XRI eXtensible Resource Identifier

Capítulo 1

Introdução

O crescente número de serviços disponibilizados em plataformas *online* aumentou significativamente na última década, desde *e-Commerce*, redes sociais, entertenimento, *homebanking*, email, entre outros. Apesar da enorme variedade de serviços disponibilizados todos eles mantêm algo em comum, a necessidade de ter informação sobre o utilizador para, eventualmente, prestar um melhor serviço.
A gestão da informação, mesmo que pessoal, sempre esteve do lado das organizações num modelo centralizado em que o utilizador cede a informação necessária para a interacção com o serviço e a gestão dos dados é delegada às organizações de acordo com políticas de privacidade definidas por esta, ou seja, o utilizador não controla o modo como os seus dados são geridos apenas aceita ou recusa a política imposta pelo prestador do serviço. Este modelo gera do lado do utilizador a sensação de perda de controlo e a fragmentação da informação muitas vezes duplicada de forma inconsistente nas várias localizações, como por exemplo, o simples acto de modificar o endereço de residência implica ao utilizador actualizar essa informação nos diversos serviços que utiliza para as suas compras *online*. Do lado das organizações, estas têm de efectuar um esforço para manter a informação actualizada e em alguns casos verificar a veracidade desses dados. Ao nível dos serviços a tendência é para fomentar a interoperabilidade entre os vários disponíveis, partilhando informação entre domínios. O desafio passa por evitar que a ubiquidade do acesso à informação seja feita à custa da privacidade do indivíduo.

1.1 Motivação

A enorme quantidade de informação acessível *online* e a necessidade de interligar e relacionar dados de várias fontes independentemente da sua localização ou meios de interacção exige uma reformulação, uma nova abordagem, ao modo como a informação é identificada, representada e disponibilizada. A arquitectura de identificação e resolução de recursos actual, *Domain Name System* (DNS), soluciona a abstracção da localização de máquinas e, em parte, protocolos de interacção. No entanto é necessário unificar o processo de identificação e resolução para todo o tipo de recurso, seja ele uma máquina, fotografia, artigo ou conceito abstracto. Os esforços para que tal seja possível passam por três princípios chave:

- A necessidade de uma nova camada, que inclua identificadores abstractos, únicos, persistentes e que tenha em consideração questões de segurança e privacidade, claramente afastados do DNS actual.

- A existência de formatos *machine-readable* capazes de descrever não só o objecto (pessoas, grupos, produtos, entre outros) como a relação entre eles e em diferentes contextos melhorando a automação na descoberta de dados e relacções.

- A mediação de políticas de acesso aos recursos orientadas à privacidade e um controlo granular do acesso à informação.

Enquanto que as organizações já possuem um vasto leque de ferramentas que permitem gerir a informação e as relações com os seus clientes, situando-se na área de *Customer Relationship Management* (CRM), o contrário não é uma realidade. Urge o aparecimento de ferramentas que permitam ao indivíduo gerir a sua informação e as relações que mantém com organizações, ou seja, *Vendor Relationship Management* (VRM), ou outros indivíduos. De um modo mais geral pode ser considerado como uma camada mediadora, controlada pelo utilizador, de acesso à informação [13]. Do lado da mediação de acessos a tecnologia que tem tido melhor receptividade por parte dos implementadores, OAuth [14], permite retirar a necessidade da partilha das credênciais especifícas a cada domínio e associar autorizações, centradas no utilizador.
Do lado da representação dos objectos de forma *machiche readable* está a designada "*Semantic Web*"[15], particularmente as tecnologias associadas ao conceito de "*Linked Data*"[15] e assente principalmente na especificação *Resource Description Framework* (RDF)[16] desenvolvida sob alçada do *World Wide Web Consortium* (W3C). A adopção destes mecanismos tem tido maior proliferação em serviços relacionados com redes sociais já que abordam um tipo de informação muito ligada ao conceito de relações entre identidades (pessoas) e onde o utilizador está mais consciencializado para a importância da privacidade devido ao nível cada vez maior de integração entre as plataformas mais populares.
Conjungando as várias considerações já enunciadas, está em desenvolvimento no âmbito da *Organization for the Advancement of Structured Information Standards* (OASIS) um conjunto de especificações [8, 11, 12, 17], que consiste num *framework* XRI/XDI, que tentam colmatar a ligação entre identificação, descrição e mediação de acesso a recursos podendo operar em conjunto com as tecnologias descritas anteriormente para criar uma pilha de tecnologias abertas e livres na realização da visão para uma *web* mais semântica, portabilidade da informação e centrada no indivíduo.

1.2 Objectivos

Pretende-se com a elaboração desta Dissertação estudar as tecnologias que permitam desenvolver serviços, baseados numa arquitectura "*user-centric*", focados na gestão da informação pessoal. Para tal é necessário o estudo prévio das tecnologias implementadas com maior expressão nos serviços actuais.
Sendo o *framework* XRI/XDI relativamente recente, principalmente a tecnologia XDI cuja especificação ainda está em desenvolvimento, o estudo incide principalmente no seu estado actual e possibilidades de integração com as tecnologias existentes demonstrado através de alguns casos de uso. Outro dos objectivos desta Dissertação é a implementação de um protótipo como prova de conceito para demonstrar as potencialidades da tecnologia XDI na intergração em serviços existentes, nomeadamente dados contidos em serviços de redes sociais.

1.3 Contribuições

A visibilidade da tecnologia XDI em situações reais ainda é relativamente escassa devido à não existência de uma especificação estável que assegure o correcto funcionamento dos serviços que a implementem. Para além de uma especificação estável é também necessário definir ontologias e dicionários para os vários tipos de recursos a descrever.
O trabalho desenvolvido ao longo desta Dissertação fornece propostas tanto para a definição de alguns dicionários usados na integração com serviços de redes sociais assim como a adição de código que permita manipular esse tipo de dados de forma mais conveniente. O protótipo desenvolvido acrescenta a plataforma Google App Engine como uma opção viável e atractiva (baixo custo, escalabilidade, entre outros) para o desenvolvimento de aplicações que façam uso da tecnologia XDI e criar uma abstração para a localização da informação podendo estar distribuída por vários serviços.

1.4 Estrutura do documento

A estrutura desta Dissertação está está dividida em quatro partes distintas.

- Introdução, Capítulo 1, enquadra o estudo e trabalho desenvolvido ao longo da elaboração da Dissertação.
- Estado da arte, Capítulos 2, 3, 4 e 5, aborda as tecnologias relevantes para a identificação, resolução e descrição de recursos.
- Prova de conceito, Capítulo 6, explana através de uma prova de conceito a visão proposta como mediação e agregação de acesso a dados contidos em serviços *web* sociais já existentes.
- Conlusões, o Captítulo 7 apresenta as conclusões finais resultantes do trabalho desenvolvido e sugere possibilidades de trabalho futuro.

Capítulo 2

Domain Name System (DNS)

O DNS tem um papel importante nas redes de hoje e é utilizado há mais de 20 anos como sistema de resolução. Permite uma abstracção entre um identificador, simples e estruturado de maneira hierárquica, e um recurso. É criado um identificador passível de ser utilizado independentemente do local da rede onde se encontra o recurso ou a entidade que pede a sua resolução. Este processo de associação entre um identificador de alto nível ,*domain name*, e um identificador de baixo nível ,*resource address*, é chamado de resolução de nome.

2.1 Perspectiva histórica e motivações

A primeira arquitectura de resolução de nomes [18] surgiu na rede ARPANET quando esta ainda continha apenas cerca de uma centena de máquinas. Mesmo um número tão reduzido tornava impraticável decorar todos os endereços das máquinas disponíveis na rede. A solução da altura foi uma arquitectura centralizada que continha todos os mapeamentos endereço-*hostname* num ficheiro único (`host.txt`)[1] situado num servidor administrado pelo SRI-NIC. No inicio dos anos 80 o número de máquinas na rede subiu muito rapidamente, o que evidenciou os problemas de implementar uma arquitectura centralizada para este tipo de serviço; elevado tráfego, colisão de nomes, inconsistência e dependência de um ponto central. A necessidade de encontrar um solução para o problema levou à especificação do DNS que se tornou o *standard* [19] de hoje. Os objectivos a que o sistema se propôs foram:

- **Consistência**: O *domain namespace* é distribuído hierarquicamente e pode ser subdivido em várias partes (*domain names*), esta hierarquia permite consistência garantindo identificadores únicos para cada recurso dentro de uma zona administrativa.
- **Escalabilidade**: É uma arquitectura distribuída cliente-servidor com a possibilidade de delegar a administração de uma zona do *domain namespace* a outra entidade retirando assim carga a entidades administrativas a montante.
- **Eficiência**: Devido à sua natureza distribuida a maioria dos pedidos de resolução são resolvidos através de uma entidade local que implementa mecanismos de *caching*, evitando a necessidade de contactar elementos acima na hierarquia para resolver pedidos *domain names* já conhecidos. No entanto se estas entradas residirem para sempre na

[1] Actualmente este mecanismo ainda está presente nos sistemas operativos actuais, como complemento. Por exemplo, em sistemas UNIX a sua localização é `/etc/hosts`.

cache o problema da consistência dos dados volta pôr-se, assim a cada entrada também está associado um parâmetro *Time To Live* (TTL) que define qual a validade temporal do registo.

2.2 Organização do *Domain namespace*

O espaço de domínios DNS está organizado como uma estrutura em árvore invertida que permite a divisão em vários domínios no sentido *top-down* e endereçar qualquer elemento na rede. Cada nó é identificado por um nome e o identificador global de um *domain name* será a concatenação de todos os nomes até à raiz, separadas por um ponto. e.g., `workstation04.labcom.deti.ua.pt`. Uma parte do *namespace* que esteja sobre a autoridade da mesma entidade (*name server*) é designada por zona, o que não significa o mesmo que domínio que pode ter uma ou mais zonas (ver Figura 2.1).

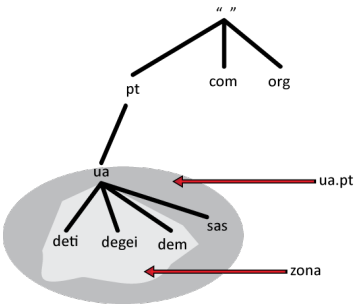

Figura 2.1: Exemplo, *Domain namespace*.

2.3 Arquitectura

A característica distributiva da arquitectura DNS implica que qualquer entidade detentora dos direitos de utilização sobre um domínio pode gerir o seu *name server* ou delegar esse domínio para outra entidade. A arquitectura cliente-servidor do DNS é composta por 3 elementos base: cliente (*resolver*), servidor (*name server*) e registos (RR). A Figura 2.2 ilustra a relação entre entes elementos.

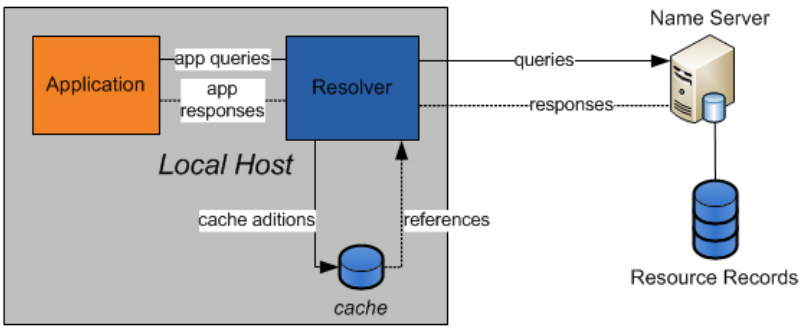

Figura 2.2: Arquitectura cliente-servidor,DNS.

2.3.1 *Resource Record* (RR)

Como o nome indica é um formato de dados que representa um recurso disponível na rede, sendo a informação trocada entre clientes e servidores. Por exemplo, quando o utilizador digita um *Uniform Resource Locator* (URL) no *browser*, e.g. `http://www.ua.pt`, a aplicação necessita saber qual a localização (endereço IP do recurso) para que lhe possa enviar pedidos HTTP. A Tabela 2.1 lista os vários atributos que formam um RR.

Atributo	Descrição
NAME	Identificador da localização no *namespace*
TYPE	O tipo de dados que contém, sendo os mais comuns: • A - endereço IPv4. • AAAA - endereço IPv6. • NS - endereço de um *name server* que tem autoridade para a classe e domínio especificados. • SOA - É a primeira entrada em cada zona e identifica o *name server* como sendo autoritário para o domínio. • PTR - ponteiro para outra localização no *name space*. • CNAME - *Canonical Name*, domínio principal do recurso. • MX - recurso com a capacidade de *mail exchange* para o domínio. • SVR - serviço, `protocol:address:port`. • TXT - texto.
CLASS	classe a que pertence o registo, quase exclusivamente a classe `Internet (IN)` é utilizada actualmente. Outras classes possíveis podem ser consultadas na especificação [19].
TTL	validade temporal do registo, se tiver expirado é descartado da *cache*.

Tabela 2.1: Atributos de RR.

2.3.2 *Name Server*

Toda a informação acerca do *domain namespace* está distribuída pelos vários *name servers*. A sua principal funcionalidade é responder a pedidos de resolução utilizando os dados da zona sobre a qual detém autoridade ou dados contidos em *cache*, adquiridos em pedidos anteriores pelo *resolver* local. Mesmo que não exista o resultado de resolução, em *cache*, pode ser dada uma resposta que encurte o processo de resolução, por exemplo o endereço do *name server* autoritário para o nome que se pretende resolver (registo NS). Para a existência de redundância e tolerância a falhas é comum existirem dois *name servers* por zona, primário e

secundário. Para manter a consistência necessária o DNS prevê mecanismos de sincronização (*zone transfer*). Quando os dados de zona contidos no primário são actualizados é incrementado o campo *serial number* no RR SOA que indica a versão dos dados, o secundário compara esta entrada com a que contém e inicia uma transferência de zona se os dados forem mais recentes.

2.3.3 *Resolver*

O *resolver* é o elemento cliente na arquitectura do DNS, efectuando os pedidos de resolução ao servidor a pedido de uma aplicação do sistema. Para que o *resolver* inicie um processo de resolução necessita de conhecer pelo menos o endereço de um *name server* para iniciar o processo (*bootstrap*). O processo pode terminar logo após a primeira resposta ou receber informação de quais os *name servers* a contactar que possam melhor resolver o pedido. Por questões de eficiência podem implementar um mecanismo de *cache* (*non-stub resolvers*) para guardar registos de resoluções anteriores evitando contactar um *name server*, se o TTL do registo não tiver expirado.

2.4 Resolução

O pedido de resolução é um processo iniciado sempre pelo cliente. Este deseja que dado um *domain name* lhe possam indicar qual o recurso associado. Por exemplo, o utilizador insere no *browser* um URL (e.g. `http://www.ua.pt`), este delega à implementação do *resolver* que envie um pedido de resolução ao *name server* e quando devolvido o RR contendo o endereço IP do *web server*, o *browser* pode estabelecer um ligação com este.
O Exemplo 1 é a resposta recebida para o pedido anterior. As *flags* `QR`, `RD` e `RA` sinalizam que a mensagem é uma resposta, esta é final e que o modo recursivo está disponível, respectivamente. A *flag* `AUTHORITY` a zero indica que o *name server* não é autoritário sobre o domínio, neste caso servidor do *Internet Service Provider* (ISP). A secção `Answer` contém o endereço IPv4 correspondente ao *domain name* `www.ua.pt`.

```
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 14299
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;www.ua.pt. IN A

;; ANSWER SECTION:
www.ua.pt. 27959 IN A 193.136.173.25

;; Query time: 6 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Mon Dec 14 05:55:00 2009
;; MSG SIZE  rcvd: 43
```

Exemplo 1: Resposta de resolução recursiva.

Existem dois modos possíveis de configurar a relação entre cliente e servidor, resolucção

iterativa e recursiva. Embora seja sempre o servidor a ditar qual o modo a utilizar o cliente pode expressar qual a sua preferência através da *flag* `RD`.
Num pedido **recursivo** é esperado que a resposta seja a resolução final do pedido, isto é, se o *name server* não for autoritário para o domínio cabe-lhe efectuar o resto do processo de resolução começando pelo *domain name* mais acima na hierarquia (*root name server*).
Num pedido **iterativo** se o *name server* não for autoritário para o *domain name* pedido nem tiver o resultado em *cache* devolve os registos que acha úteis para a sua resolução. Por exemplo, o pedido a um *root name server* para resolver `www.ua.pt` devolve o registo `NS` associado ao *name server* responsável pelo domínio `pt`, este em seguida devolve o registo `NS` associado ao domínio `ua.pt` e finalmente o *name server* autoritário devolve o endereço do *webserver*. O Exemplo 2 mostra as repostas de um pedido de resolução feito ao *root name server*.

```
(...)
[d.root-servers.net]
pt. 172800 IN NS AUTH210.NS.UU.NET.
(...)
[AUTH200.NS.UU.NET]
ua.pt. 28800 IN NS ns.ua.pt.
ua.pt. 28800 IN NS ns2.ua.pt.
(...)
[ns.ua.pt]
www.ua.pt. 28800 IN A 193.136.173.25
(...)
```

Exemplo 2: Respostas de resolução iterativa.

O método recursivo implica uma carga muito maior para o *name server* que recebe o pedido. É conveniente que o *name server*[2] para o qual o *resolver* está configurado aceite pedidos recursivos, enquanto que os restantes apenas iterativos. A Figura 2.3 monstra o aumento considerável de tráfego nos *name servers* do nível *Top Level Domain* (TLD) quando a maioria dos *name servers* de um ISP ficaram indisponíveis. Isto mostra que sempre que possível a resolução deve ser resolvida o mais perto possível de quem efectuou o pedido.

2.4.1 Resolução inversa

Um pedido DNS é normalmente para dado um *domain name* obter um endereço IP. No entanto existem ocasiões em que é útil a situação inversa, dado um endereço IP obter o *hostname* correspondente. Algumas vezes por uma questão de diagnóstico para rastrear actividades de *hacking* e *spamming*, tanto que hoje em dia alguns sistemas de *email* usam resolução inversa como método de autenticação simples que valida o mapeamento nos dois sentidos [20].
Um endereço IPv4 é representado por quatro grupos de dígitos separados por um ponto, a sua representação é facilmente mapeada no *domain namespace*. Existe um *domain name* reservado para esse efeito, o `in-addr.arpa.`, com a diferença de ter a ordem invertida porque ao contrário do *domain namespace* um endereço IP vai ficando mais específico da esquerda

[2]Normalmente, o servidor DNS de uma empresa ou do ISP que estão geográficamente mais perto e já contêm um número signigicativo de resultados em *cache*

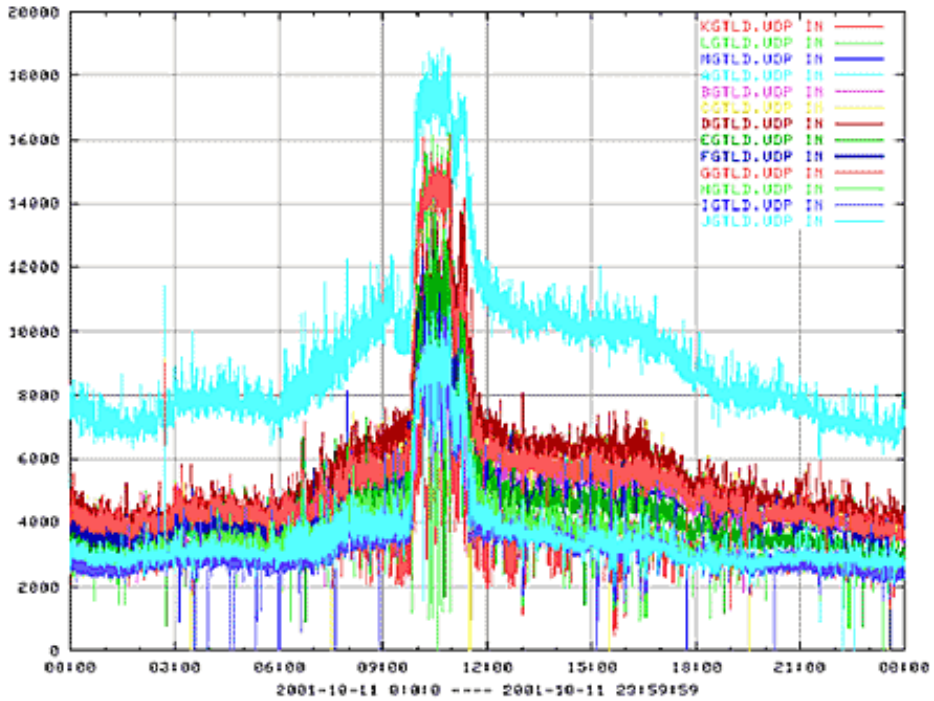

Figura 2.3: Incidente DNS, Outubro 2001. [1]

para a direita. A gestão deste *domain name* está a cargo do ICANN sendo necessário efectuar/actualizar o registo através desta organização para ser possível a resolução inversa, por isso é expectável que nem sempre seja possvel fazer a resolução inversa mesmo que exista o mapeamento no sentido directo.
Com a introdução de novos identificadores a necessidade de criar novos *domain namespaces* reservados para conter o mapeamento inverso é imperativo, por exemplo com a introdução do IPv6 foi necessário criar outra zona, a `ip6.arpa` [21][22].

2.5 *DNS Security* (DNSSEC)

A especificação inicial do DNS [19] não prevê qualquer mecanismo de segurança e é por isso vulnerável a ataques [23] como *cache poisining*, *name based authentication*, entre outros. Sendo o DNS um serviço crucial para o funcionamento da *Internet* nos dias de hoje houve necessidade de tentar introduzir alguns mecanismos. O DNSSEC [24] surgiu com o objectivo de introduzir **autenticação de origem**, **integridade dos dados** e **autenticação de não existência de recurso**, mantendo compatibilidade com o sistema já implementado. Baseia-se num mecanismo de assinaturas digitais assimétrico em que existem duas chaves, uma privada e uma pública. O *resolver* executa um pedido ao *name server* e este devolve uma assinatura assinada pela chave privada, se o *resolver* tiver conhecimento da chave pública pode verificar a validade desta. Devido à informação extra enviada numa resposta DNSSEC os intervenientes no processo necessitam de suportar os mecanismos de extensão [25] suportando datagramas UDP superiores e o suporte do bit `DO(DNSEEC OK)` no cabeçalho [26] para identificar o suporte do cliente. A Tabela 2.2 apresenta os novos RRs necessários.

Resource Record Signature **(RRSIG)**	Para cada RR existe um RRSIG que contém os dados assinados, o tipo de RR que assina, algoritmo utilizado, o TTL do registo, tempo de validade da assinatura, a *tag* da chave utilizada (pode haver mais do que um par) e o *owner* da chave utilizada para assinar o registo (geralmente o nome da zona).
DNS Public Key **(DNSKEY)**	Contém a *tag* do *owner*, o algoritmo e o valor da chave necessário para validar a informação de um RRSIG ou DS. A chave está associada a uma zona e não ao *name server*.
Delegation Signer **(DS)**	Existe numa zona acima da zona de confiança (*Delegation Point*) e contém a chave pública correspondente à chave privada utilizada para assinar a DNSKEY da zona de confiança.
Next Secure **(NSEC)**	Utilizado para validar a não existência de recurso para o pedido efectuado, dado que na especificação original [19] a resposta para um recurso não existente é uma resposta com campos vazios, foi necessário criar um novo registo que contivesse informação para assinar. No entanto como o registo aponta o próximo registo seguro, indesejávelmente permite, a possibilidade de enumeração dos recursos de uma zona, o que levou à introdução do `NSEC3` [27] que devolve o *hash* do *domain name*.

Tabela 2.2: RR introduzidos no DNSSEC.

2.5.1 Cadeia de Autenticação

Não sendo viável configurar as chaves públicas em cada *resolver* impõe-se a existência de um processo de descoberta, seguindo a estrutura hierárquica do DNS (zonas). Na situação ideal em que todo o *domain namespace* implementa DNSSEC, a cadeia começa na raiz ".". Por exemplo para `www.ua.pt.`, os registos DS da zona *root* são utilizados para verificar a chave da zona `pt.`, aceitando os registos pertencentes a `pt.` como sendo válidos se este contiver registos DS associados à zona `ua.pt.` é possível validar a DNSKEY para essa zona e finalmente validar os RRs devolvidos para o recurso com *domain name* `www.ua.pt.` verificando se a assinatura destes (RRSIG) é válida para a DNSKEY obtida.
A implementação do DNSSEC está ainda pouco disseminada [28] sendo o cenário anterior pouco viável no imediato. No entanto é possível aplicar o conceito a pequenas partes do *domain namespace* designadas por *Island of Security* (ver Figura 2.4). Nessas, a função do *root node* é efectuada pelo nó que define o início da zona que suporte DNSSEC designada por *Trust Anchor* e que precisa ter conhecimento da chave pública que não por meio do mecanismo de

descoberta descrito anteriormente. Apenas os registos pedidos dentro da zona e pertencentes à zona podem ser considerados seguros.

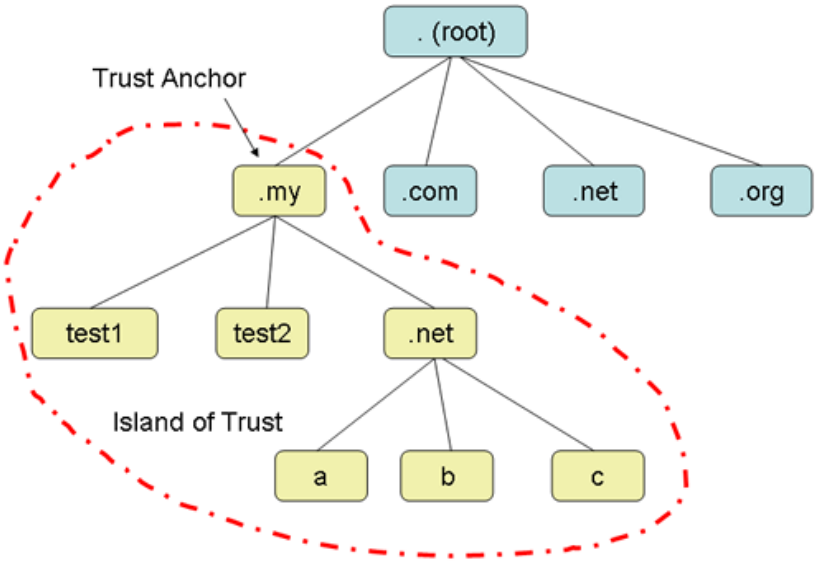

Figura 2.4: *Island of Trust* [2].

2.6 Considerações

O DNS é um protocolo de resolução de nomes escalável, robusto e extensível como se prova com o crescimento enorme da *Internet* e as extensões efectuadas para permitir novos serviços (DNSSEC, DNS ENUM, suporte IPv6) adaptando-se às necessidades que foram surgindo.
Apesar disso as exigências e flexibilidade necessárias para os serviços de hoje podem não ser resolvidas acrescentando ainda mais extensões ao protocolo. Para cada novo tipo de mapeamento é necessário criar uma nova zona (sobre o domínio `arpa.`) o que o torna pouco apetecível para mapear objectos digitais.
Outro aspecto importante é a mobilidade. O DNS não foi desenhado tendo em vista a mobilidade a um ritmo rápido, sendo uma base de dados distribuída é necessário que as mudanças se propaguem mas o TTL de um registo também não pode ser muito curto porque iria contrariar o propósito de se conservar registos em *cache* para aliviar o tráfego gerado. Mesmo com a possibilidade de efectuar *updates* dinâmicos a um *name server* a propagação barra sempre no tempo que um registo demora a ser descartado da *cache*.
O tamanho das mensagens mesmo com as extensões pode continuar a ser um problema. Com a introdução do DNSSEC e também do IPv6 é previsível que as mensagens aumentem de tamanho e encorajar o uso de TCP/IP não é solução. A coexistência de IPv6 e IPv4 pode duplicar os pedidos dos *resolvers* para aplicações que peçam os dois tipos de resolução, principalmente em SOs desactualizados.
Em termos confidencialidade da informação o DNSSEC não resolve o problema, este apenas permite verificar se aquela resposta teve origem numa fonte válida de informação.

Capítulo 3

OpenID

Com crescimento de serviços *online*, a identidade de um utilizador tende a ficar fracionada (ver Figura 3.1) entre os múltiplos serviços a que acede. Sendo que cada tem os seus próprios métodos de autenticação, obriga a criação de um perfil e credenciais em cada serviço (ver Figura 3.2).

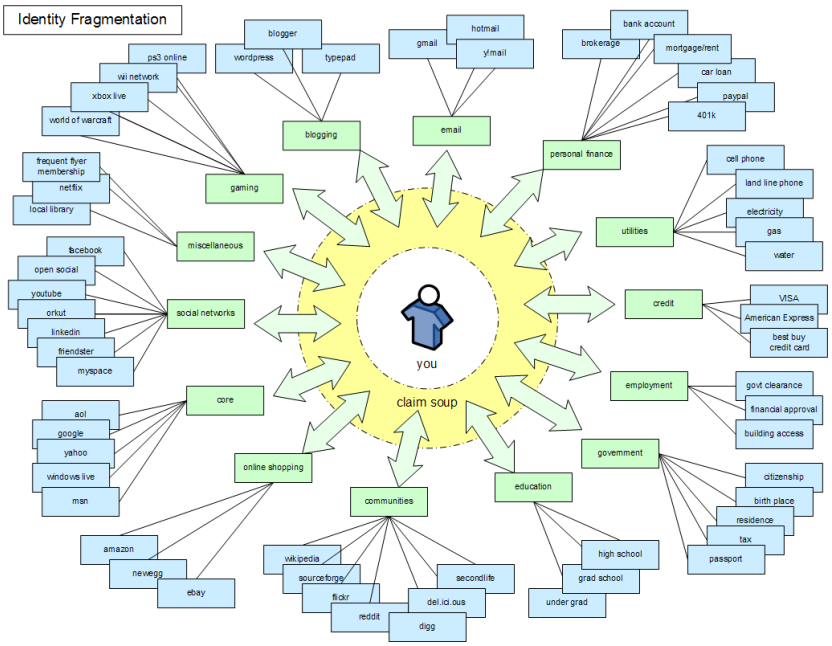

Figura 3.1: Fragmentação de identidade online. [3]

Os sistemas de *Single Sign On* (SSO), onde se inclui o OpenID, surgiram para colmatar o problema da fragmentação fornecendo mecanismos que facilitam a agregação da autenticação de uma identidade. A Figura 3.3 mostra a mudança de paradigma numa arquitectura SSO. Apenas um componente é responsável pela autenticação do utilizador, existindo credenciais únicas, enquanto que os restantes delegam essa funcionalidade para uma autoridade da sua confiança. O OpenID fornece mecanismos que transferem a autoridade sobre identificadores para o utilizador. Isto acontece sem que os serviços acedidos tenham qualquer conhecimentos das credenciais do utilizador ou qualquer outra informação associada ao identificador.

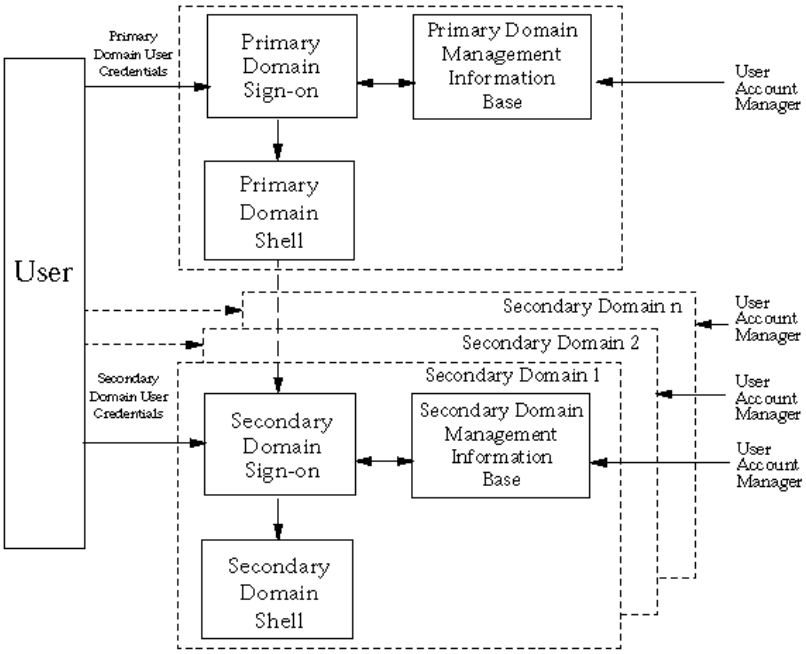

Figura 3.2: Múltiplos serviços sem SSO [4].

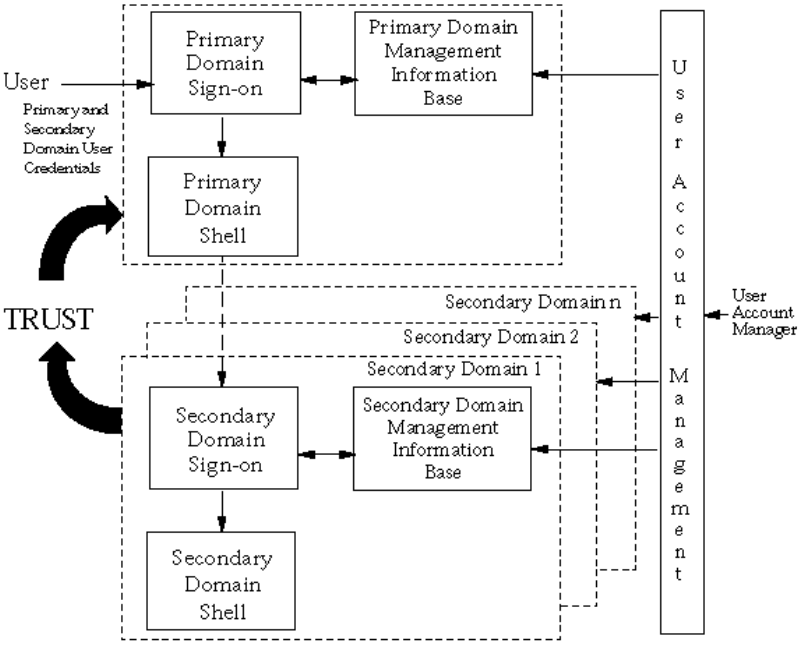

Figura 3.3: Múltiplos serviços com SSO [4].

3.1 Arquitectura

A sua arquitectura é descentralizada não existindo uma autoridade central reguladora sobre provedores de identitade OpenID ou serviços consumidores, permitindo ao utilizador trocar de provedor sem perder o identificador. Do mesmo modo, não havendo regulação cabe ao serviço que aceite identificadores OpenID, a escolha de aceitar determinado provedor para delegar a autenticação. O processo de autenticação é realizado sobre HTTP(S) pelo que não requer capacidades adicionais por parte da aplicação do utilizador, normalmente o *web browser*. A Figura 3.4 ilustra os componentes que constituem a arquitectura OpenID e a Tabela 3.1 descreve os seus papéis:

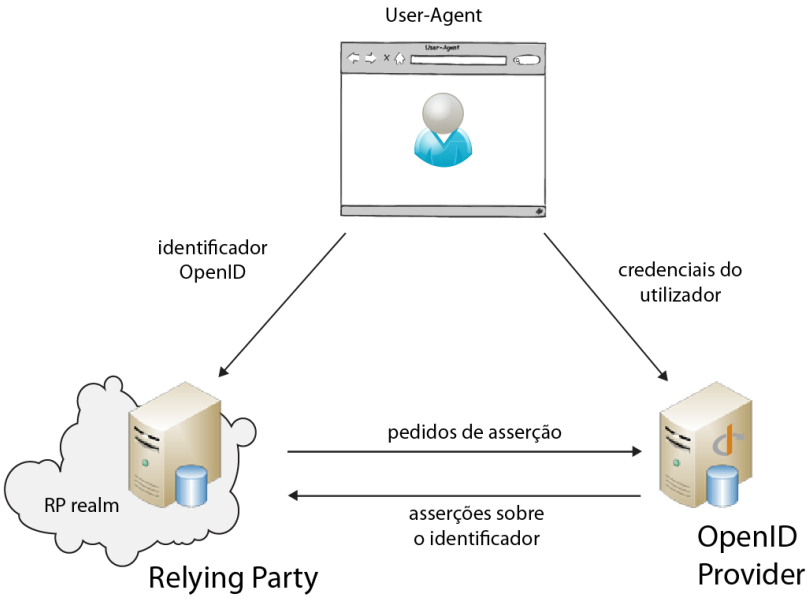

Figura 3.4: Arquitectura OpenID.

User-Agent	O *web browser*, implementa HTTP/1.1 [29], serve de *interface* para a interacção entre o utilizador e os restantes elementos, *Relying Party* e *OpenID Provider*.
Relying Party **(RP)**	Aplicação que necessita da prova que o utilizador detém autoridade sobre o identificador.
OpenID Provider **(OP)**	Servidor de autenticação OpenID, no qual o RP confia as suas asserções sobre a autoridade do identificador.

Tabela 3.1: Elementos da arquitectura OpenID.

3.1.1 Identificadores

Um identificador OpenID permite ao RP reconhecer o utilizador no seu domínio sem que necessite de implementar mecanismos próprios de asserção da identidade ou guardar qualquer informação acerca das suas credenciais. Este pode ser de dois tipos, HTTP(S) *Uniform Resource Identifier* (URI) - URL - ou *eXtensible Resource Identifier* (XRI) (ver Capítulo 4), podendo ter vários significados consoante o contexto. O utilizador não necessita obrigatóriamente de se apresentar perante o RP com o identificador sobre o qual é autoritário. Caso o OP suporte, pode apresentar o identificador do OP que assistirá o utilizador na escolha do identificador a utilizar, no caso de possuir mais do que um reflectindo as várias especificidades da sua identidade. A especificação [30] designa o identificador apresentado ao RP, por qualquer dos métodos anteriores, como *User-Supplied Identifier*.
O objectivo geral do protocolo é verificar o direito que o utilizador detém sobre o identificador. A este identificador a especificação designa por *Claimed Identifier*. A Tabela 3.2 contém alguns exemplos, e é obtido através de normalização do identificador fornecido ao RP como descrito na secção 3.1.1.1.

`http://nunorosa.com`	O utilizador controla um domínio no *namespace* do DNS e gere o descritor do recurso.
`http://www.google.com/profiles/nuno.sp.rosa`	É possível que o utilizador já possua um identificador OpenID nos vários serviços que utiliza. Este é o exemplo em que o URL do perfil no Google é também um identificador OpenID.
`@id*nunorosa`	Identificador XRI (ver Capítulo 4).

Tabela 3.2: Exemplos de identificadores OpenID.

3.1.1.1 Normalização

Um identificador fornecido pelo utilizador deve ser sempre normalizado pelo RP antes do dar ínicio ao processo de descoberta do OP. Essa normalização segue os passos pela ordem que se segue.

1. Se o identificador fornecido pelo utilizador conter o prefixo `xri://`, o RP tem de retirá-lo, para que o XRI seja utilizado na sua forma canónica.

2. Se o primeiro caracter do identificador for um XRI *Global Context Symbol* (GCS) ou o início de uma referência cruzada o identificador deve ser tratado como um XRI, ver secção 4.2.

3. Se o identificador não foi reconhecido como um XRI, deve ser tratado como um URL.
   Se o identificador não tiver um esquema `http` ou `https`, o prefixo *http://* deve ser acrescentado. No caso de ter um fragmento este tem de ser retirado juntamente com o caracter delimitador, '`#`'.

4. Identificadores do tipo URL, têm de ser normalizados seguindo os sucessivos redirecionamentos necessários para alcançar o conteúdo (ver secção 3.3.1) e normalizado para o identificador final que o RP assume como *Claimed Identifier* usado em pedidos de autenticação.

3.2 Inicialização

A Figura 3.5 ilustra a interacção entre os componentes da arquitectura OpenID durante o processo de autenticação de um utilizador perante o RP através de um OP. Esta sequência apenas ocorre completa na fase de inicialização, isto é, quando o utilizador não tem uma sessão iniciada no OP.

1. O utilizador acede à aplicação do RP através do *User-Agent* fornecendo o seu identificador, normalmente através de um formulário podendo também haver *plugins* no *browser* que automatizem o processo.

2. O RP efectua a descoberta do *endpoint* do OP e redireciona o *User-Agent* com o pedido de autenticação para o OP.

3. O utilizador autentica-se fornecendo ao OP as suas credenciais de acesso. Se no primeiro passo o identificador fornecido foi o do OP, é necessário um passo adicional, no qual o utilizador deve escolher qual o identificador a utilizar que lhe pertence.

4. O OP reencaminha o *User-Agent* de volta ao RP com o resultado da asserção sobre o identificador.

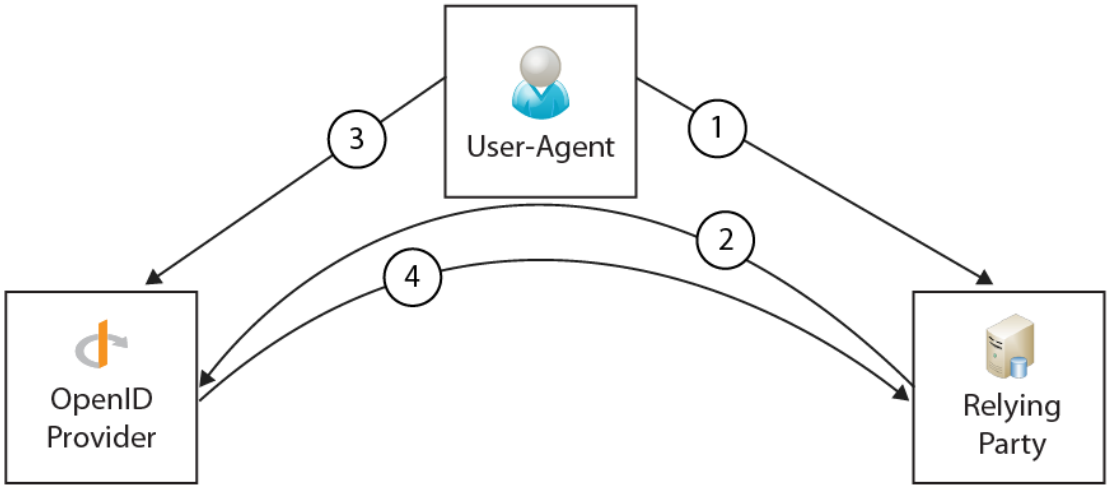

Figura 3.5: Interacções OpenID

3.3 Descoberta

É o processo em que o RP utiliza o identificador para descobrir a informação necessária para iniciar pedidos de autenticação, contendo o *endpoint* do OP indicando a localização do servidor OpenID para onde deve reencaminhar o *User-Agent*, a versão do protocolo suportado, e no caso de o utilizador não tiver fornecido o identificador do OP, o *Claimed Identifier* assim como o identificador local ao OP. Existem vários modos de descoberta consoante o tipo de identificador, resolução XRI (ver secção 4.3), resolução YADIS e resolução baseada em HTML.

3.3.1 Resolução YADIS

Yadis [31] é um sistema de descoberta de serviços que permite a partir de identificadores, *Yadis ID*, determinar sem a intervenção do utilizador qual o protocolo mais apropriado a usar para fins de autenticação, responsabilidades, entre outros. O resultado da resolução é um documento XRDS que descreve os vários serviços disponíveis, *Yadis Resource Descriptor*.
O descritor *Yadis Resource Descriptor* é obtido pelo RP efectuando um pedido HTTP, `GET` ou `HEAD`, para o URL correspondente ao *Yadis ID*, *Yadis URL*. No caso do identificador ser um URL estes são equivalentes. No entanto se o *Yadis ID* for um identificador de outro tipo, e.g. XRI, o correspondente *Yadis URL* será a transformação do identificador para um URL conforme a sua especificação.
Dependendo do suporte por parte do servidor que disponibiliza o recurso apontado pelo *Yadis URL*, pode ser necessário ao RP efectuar mais do que um pedido caso este não suporte negocição de conteúdo através do cabeçalho `Accept:application/xrds+xml`. O Exemplo 3 mostra o cabeçalho da resposta a um pedido HTTP GET sobre o identificador `http://www.google.com/profiles/nuno.sp.rosa`. O cabeçalho `X-XRDS-Location` contém a localização para o descritor do recurso correspondente ao Exemplo 4.
O documento *Yadis Resource Descriptor* lista os serviços associados ao identificador, o tipo suportado e o URI para estabelecer a comunicação com serviço que o RP deve contactar. Mais detalhes sobre a estrutura XRD encontra-se na secção 4.4.

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
X-XRDS-Location: https://www.google.com/accounts/o8/id?source=profiles
&id=http%3A%2F%2Fwww.google.com%2Fprofiles%2Fnuno.sp.rosa
```

Exemplo 3: Cabeçalho da resposta ao pedido de resolução.

3.3.2 Resolução baseada em HTML

Existem casos mais simples em que implementar suporte para o protocolo Yadis não é uma necessidade. Por exemplo, o utilizador quer utilizar o URL do seu *website* como identificador delegando a autenticação para o OP. Este mecanismo consiste em ter a informação de resolução no código HTML apontado pelo indentificador. Contido no elemento `<header>` terá de existir pelos menos o elemento `<link>` com o atributo `rel='openid2.provider'` e o atributo `href` indicando o *endpoint* do OP.

```
<xrds:XRDS xmlns:xrds="xri://$xrds" xmlns="xri://$xrd*($v*2.0)">
  <XRD>
  <Service priority="10">
  <Type>http://specs.openid.net/auth/2.0/signon</Type>
  <Type>http://openid.net/srv/ax/1.0</Type>
  <Type>http://specs.openid.net/extensions/ui/1.0/mode/popup</Type>
  <Type>http://specs.openid.net/extensions/ui/1.0/icon</Type>
  <Type>http://specs.openid.net/extensions/pape/1.0</Type>
  <URI>https://www.google.com/accounts/o8/ud?source=profiles</URI>
  </Service>
  </XRD>
</xrds:XRDS>
```

Exemplo 4: *Yadis Resource Descriptor*.

3.4 Associação

Opcionalmente, após a descoberta, pode ser realizada uma associação entre RP e OP criando uma chave partilhada que permite verificar a autenticidade das messagens seguintes e diminuir os tempos de *round-trip*. Caso o RP não tenha suporte para efectuar uma associação existe outro mecanismo conhecido por *Stateless Mode* em que o RP envia um pedido ao OP para verificar a assinatura através de uma associação interna, do OP, criada quando respondeu com uma asserção positiva. Uma sessão de associação é iniciada por um pedido directo do RP para o servidor do OP, sem intervenção do *User-Agent*.

3.5 Autenticação

Assim que o RP finaliza a descoberta, e opcionalmente uma associação, com sucesso sobre o identificador pode enviar pedidos de autenticação ao OP. O pedido é do tipo indirecto já que é feito através do *User-Agent* reencaminhado-o para o OP. A Tabela 3.3 lista os parâmetros necessários a enviar pelo RP e o Exemplo 5 mostra os detalhes de um pedido de autenticação. O OP responde a pedidos de autenticação através do *User-Agent* enviando um redirecionamento com o RP como destino, estas messagens devem ser assinadas pelo OP. O Exemplo 6 é a resposta ao pedido do Exemplo 5.

openid.ns	http://specs.openid.net/auth/2.0
openid.claimed_id	@!B1E8.C27B.E41C.25C3!6a64.5ff2.58f8.0404
openid.identity	@!B1E8.C27B.E41C.25C3!6a64.5ff2.58f8.0404
openid.return_to	https://login.sapo.pt/LoginWithOpenID.do?is_return=1&disco_id=cd2197c7-df2f-4846-b4fe-817ecafcc2b4&authProvider=OpenID
openid.realm	https://login.sapo.pt/LoginWithOpenID.do
openid.assoc_handle	1286465563779-0
openid.mode	checkid_setup

Exemplo 5: Pedido de autenticação pelo RP.

`openid.ns`	Identifica a versão do protocolo utilizado, sendo o valor `http://specs.openid.net/auth/2.0` utilizado pela última especificação [30].
`openid.mode`	Indica o tipo de pedido efectuado. Para pedidos de autênticação existem dois modos consoante o desejo de o utilizador inteagir com o OP, `checkid_setup`, ou no caso em que o pedido é assíncrono em aplicações AJAX, `checkid_immediate`.
`openid.claimed_id`	(Opcional) `Claimed Identifier`, o identificador sobre o qual o RP pretende uma asserção.
`openid.identity`	(Opcional) Identificador local ao OP. Se não existir deve ser igual ao valor do parâmetro `openid.claimed_id`. Existe ainda um valor especial para quando o identificador final deve ser escolhido no OP, `http://specs.openid.net/auth/2.0/identifier_select`.
`openid.assoc_handle`	(Opcional) A chave da associação entre RP e OP. Se não existir, a transação ocorre em *Stateless Mode*.
`openid.return_to`	(Opcional) URL para o qual o OP deve retornar o *User-Agent* com o resultado da autenticação.
`openid.realm`	(Opcional) O domínio em a que asserção do identificador é válida e que se espera a confiança do utilizador. O valor será indicado ao utilizador no OP. Por defeito o valor é o mesmo que o URL presente no parâmetro `openid.return_to`.

Tabela 3.3: Parâmetros de um pedido de autenticação.

is_return	1
disco_id	cd2197c7-df2f-4846-b4fe-817ecafcc2b4
authProvider	OpenID
openid.ns	http://specs.openid.net/auth/2.0
openid.op_endpoint	https://authn.freexri.com/authentication/
openid.claimed_id	@!B1E8.C27B.E41C.25C3!6a64.5ff2.58f8.0404
openid.response_nonce	2010-10-07T16:55:05Z0
openid.mode	id_res
openid.identity	@!B1E8.C27B.E41C.25C3!6a64.5ff2.58f8.0404
openid.return_to	https://login.sapo.pt/LoginWithOpenID.do?is_return=1&disco_id=cd2197c7-df2f-4846-b4fe-817ecafcc2b4&authProvider=OpenID
openid.assoc_handle	1286465563779-0
openid.signed	op_endpoint,claimed_id,identity,return_to,response_nonce,assoc_handle
openid.sig	OI+lxdh5KtGNo+leATIRHqzpaNUdsA+YU+Cuw+n15k0=

Exemplo 6: Resposta a autenticação pelo OP.

3.5.1 Verificação

Ao receber a reposta do OP, o RP necessita confirmar a validade da informação recebida. Concretizando e comparando com os Exemplos 5 e 6:

- O valor do parâmetro `openid.return_to` corresponde ao URL enviado no pedido.
- A informação da resposta corresponde à encontrada na fase de descoberta. *Claimed*

Identifier, identificador local ao OP, OP *endpoint* URL e versão do protocolo.

- Ainda não foi aceite nenhuma asserção do OP para o mesmo valor do `openid.response_nonce`.
- A assinatura é válida (`openid.sig`) e todos os parâmetros necessários estão assinados (`openid.signed`).

3.6 Segurança

3.6.1 Ataques de *Replay* e *Eavesdropping*

O RP mantem o registo dos valores do campo `openid.response_nonce`, por tempo determinado, e nunca aceita asserções positivas com valor repetido proveniente do mesmo OP *endpoint* URL. Esse valor é gerado pelo OP contendo uma marca temporal da resposta. Este mecanismo evita que um elemento mal intencionado possa escutar a mensagem e retransmiti-la para ganhar acesso. No caso do RP não verificar a repetição do parâmetro, a utilização de encriptação no transporte (TLS) da mensagem é suficiente.

3.6.2 Ataques *Man-in-the-Middle*

Um ataque deste tipo acontece quando um elemento mal intencionado intercepta a mensagem e manipula os parâmetros da mensagem antes de a reenviar para o destino, por exemplo mudar o identificador sobre a qual é efectuada a asserção pelo OP. O mecanismo que a especificação dispõe para prevenir o ataque é assinar todos os parâmetros sensíveis da mensagem, se algum for alterado a assinatura será inválida para o RP a ser não que a origem do ataque conheça o segredo partilhado, o que aumenta considerávelmente a probabilidade de sucesso do ataque.
Se a resolução de DNS ou a camada de transporte estiverem comprometidas a assinatura da mensagem deixa de ser eficaz já que o atacante pode fazer-se passar pelo OP e emitir as suas próprias associações.
O mesmo se passa se a informação na fase de descoberta for corrompida. Por exemplo, o utilizador tem o URL do seu *website* como identificador (ver secção 3.3.2), se o atacante conseguir ganhar acesso ao ficheiro descritor (XRDS) pode modificar o OP *endpoint* URL para um OP que controla, e sem precisar das credenciais do utilizador fazer passar-se por ele junto do RP. A solução é assinar digitalmente [32] o ficheiro descritor e deixar para o RP a responsabilidade de confiar na entidade que assina o documento.
Concluindo, apesar de não ser necessário SSL para o funcionamento do protocolo é aconselhável que pelo meno o OP assegure a integridade do seu *endpoint* e interacções com o *User-Agent* através de SSL com certificados assinados por uma autoridade confiável.

3.6.3 Ataques *Denial of Service* (DoS)

No protocolo OpenID não existe nenhum mecanismo que permita ao OP verificar se uma mensagem provem de um RP genuíno. Num ataque DoS são enviandas sucessivas mensagens de associação, autenticação ou verificação de uma assinatura. Como consequência, o OP pode estar a negar o serviço a um RP fidedigno enquanto processa os pedidos provenientes do RP malicioso. Apesar de não existirem mecanismos no protocolo que previnam este tipo

de ataques, o OP pode banir o acesso a um RP descartando pedidos provenientes do mesma origem, cabendo ao OP definir qual o critério a utilizar.

3.7 Extensões

Apesar do propósito inicial do OpenID ser apenas delegar a autenticacão do utilizador para outra entidade e reduzir o número de credenciais associadas à mesma identidade, o protocolo é versátil o suficiente para suportar extensões como troca de informações permitindo ao RP conhecer algo mais sobre o utilizador por detrás do identificador. Extensões como *OpenID Simple Registration* [33] ou *OpenID Attribute Exchange* [34] permitem a troca de informação básica como o *username*, nome, email, idoma, entre outros[1].

3.8 Considerações

A adopção do protocolo OpenID na *Internet* tem sido enorme, tanto ao nível de provedores de identidade como de serviços[2] que o suportam.
Apesar da natureza descentralizada do protocolo, é pouco espectável que o utilizador comum tenha o total controlo sobre o seu identificador e use identificadores fornecidos pelo OP. Isto pode ser um potencial incoveniente no caso de problemas em OPs menos fiáveis, principalmente em provedores de pequena dimensão, bloqueando o acesso aos mais variados serviços onde esses identificadores possam ser utilizados ou em casos mais graves o acesso indevido às credenciais do utilizador. Esses problemas podem surgir no caso de o OP não estar acessível, mudar o seu domínio ou a base de dados ser comprometida.

[1] `http://www.axschema.org/types/#sreg`
[2] `https://www.myopenid.com/directory`

Capítulo 4

eXtensible Resource Identifier (XRI)

A especificação para identificadores XRI [5][11], desenvolvida pelo OASIS XRI TC[1], fornece uma linguagem comum para identificadores estruturados, abtractos e *semantic aware* independentes de protocolo, domínio, localização ou aplicação. O XRI extende a estrutura e capacidades do IRI [35] e por consequência do URI [7], ver Figura 4.1. A principal motivação para o seu desenvolvimento é colmatar algumas das limitacões de outros identificadores, nomeadamente IRI, bem como disponibilizar normas para criar identificadores estruturados que facilitem o endereçamento, consulta e partilha de dados estruturados (ver Capítulo 5).

Figura 4.1: Relação entre XRI, IRI e URI. [5]

A abstração introduzida com o XRI é conseguida porque o indentificador por si só não aponta para a localização específica do recurso. Associado a um mecanismo de resolucção, permite descobrir o descritor do recurso que contém a concretização dos protocolos e meios de acesso ao recurso, ver Figura 4.2.

[1] `http://www.oasis-open.org/committees/xri/`

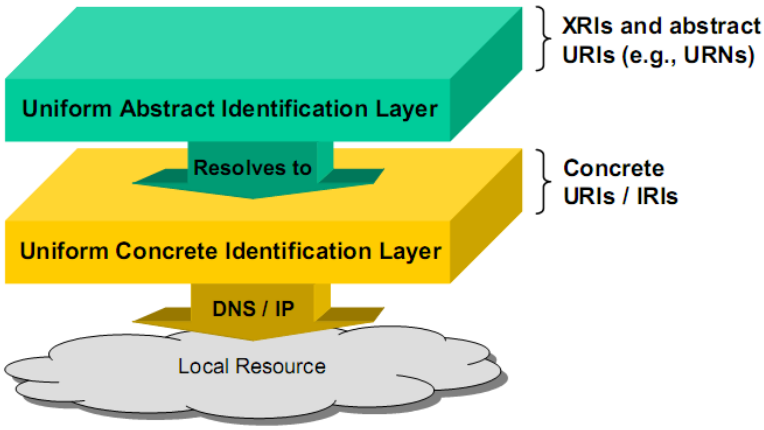

Figura 4.2: Camada de abstração XRI. [6]

4.1 Motivações

4.1.1 Persistência

O exemplo mais comum da utilização de identificadores abstractos é a necessidade de manter uma referência persistente para um recurso quando este muda de localização, evitando assim quebrar a ligação entre identificador e recurso.
A norma URI [7] aborda essa necessidade através de um subtipo designado por *Uniform Resource Name* (URN).

> *The term "Uniform Resource Name"(URN) has been used historically to refer to both URIs under the"urn" scheme [RFC2141], which are required to remain globally uniqueand persistent even when the resource ceases to exist or becomesunavailable, and to any other URI with the properties of a name. (p. 7)*

A principal diferença entre XRIs e URNs [36] [37] é que enquanto o primeiro lida com identificadores persistentes e reutilizáveis o segundo apenas lida com identificadores persistentes. Outro aspecto importante para recursos acessíveis *online* é a capacidade de resolver o identificador de modo a aceder a uma ou várias representações do objecto que identifica. Apesar de existir discussão sobre mecanismos de resolução de URNs ainda não existe um mecanismo comum capaz de resolver todo o *namespace* [38] ao contrário da resolução de XRIs.
A diferenciação entre identificadores persistentes e reutilizáveis, *i-number* e *i-name* respectivamente, garante que existe sempre um identificador de baixo nível (ver Figura 4.3) que irá identificar o recurso mesmo que mude de localização, contexto, entre outros. A relação entre os dois tipos de identificadores é estabelecida através do conceito de sinónimo (ver secção 4.4.2). A um *i-name* corresponde um *i-number*, ou seja, o resultado da sua resolução é o mesmo.

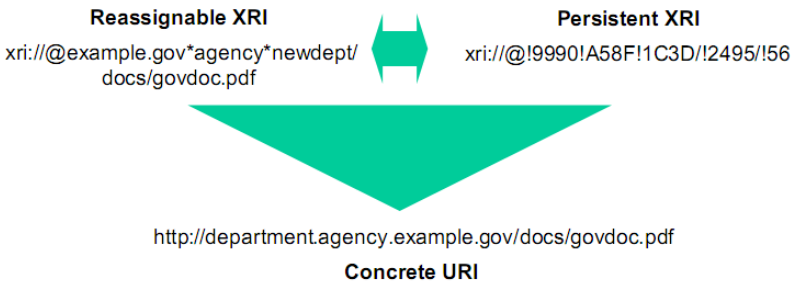

Figura 4.3: "Triângulo de Resolução" XRI. [6]

4.1.2 Delegação

Um subconjunto de identificadores URI são os URLs, e que permitem localizar qualquer recurso na *Internet* independentemente do domínio ou autoridade que os aloja. A secção autoritária de um URL pode ser um endereço IP que indica directamente a localização do recurso ou qualquer nome no *domain namesapce*. Esta característica permite resolver o URL recorrendo à delegação do DNS (ver Capítulo 2). A delegação no URL apenas ocorre dentro do segmento *authority*, ver Figura 4.4 enquanto o XRI suporta a delegação tanto no segmento *authority* como no segmento *path*.

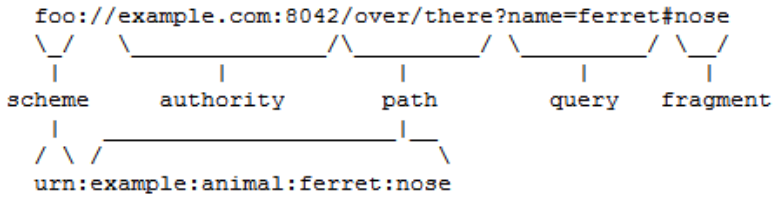

Figura 4.4: Segmentos do identificador URI. [7]

4.1.3 Multi-Contexto

Por vezes existe a necessidade de utilizar identificadores em diferentes contextos. O XRI facilita essa utilização através de referências cruzadas, (ver secção 4.2.3), que não são mais do que segmentos opacos que encapsulam o identificador do contexto original permitindo partilhar o identificador de forma consistente e com maior significado semântico. A Tabela 4.1 mostra a utilização de um identificador URN, `urn:ISBN:0-395-36341-1`, relativo a um "livro" em vários contextos.

`xri://@example.org*national.library/!(urn:ISBN:0-395-36341-1)`
`xri://@!9990!A58F!1C3D/!(urn:ISBN:0-395-36341-1)`
`xri://@example.bookstore/!(urn:ISBN:0-395-36341-1)`

Tabela 4.1: Exemplo do uso de referências cruzadas.

4.1.4 Agregação

O crescente número de meios de interacção entre pessoas ou serviços (telefone, email, *Instant Messaging* (IM), entre outros) obriga à utilização de diferentes identificadores relativos ao mesmo objecto mas com interfaces de interacção diferentes. Jamie Lewis, CEO do Burton Group, escreveu um artigo [39] onde sintetiza essa problemática.

> *Today, we use a wide variety of different mechanisms for identification, including e-mail addresses, IP addresses, phone numbers, and object identifiers. But most of these are specific to one means of interaction. None of them is persistent across the many different ways that people, applications, and devices can communicate, and so they dont function well as identifiers in the long run.*

O XRI permite a identificação de recursos de forma abstracta independentemente da localização ou protocolo de interacção. Isto significa que um único XRI pode ser utilizado para identificar um recurso nas mais diversas formas e localizações. A descoberta das capacidades e meios de interacção do recurso são relegadas para o processo de resolução e aplicações, (ver secções 4.3 e 4.4.2).

4.1.5 Metadados

Metadados, ou metainformação, é a descrição ou conjunto de características de um objecto, especialmente em relação a informação processada por computador, por exemplo, a data da última alteração de um documento.
Sendo o XRI um identificador abstracto, a descoberta de metadados sobre o recurso é relegada para o processo de descoberta tal como a descoberta de *endpoints* disponíveis. O resultado da resolução é um documento que descreve o recurso, XRD (ver secção 4.4).

4.2 Estrutura

Em termos estruturais o XRI partilha as características do URI, ver Figura 4.4, o que é de esperar já que é uma extensão às funcionalidades do URI.

4.2.1 *Authority*

Semelhante ao URI o identificador começa com o segmento *authority*. Na arquitectura XRI, ao contrário do URI, este segmento utiliza a mesma estrutura para os seus subsegmentos que o segmento *path*, com uma excepção, tem obrigatoriamente de começar com um subsegmento global.

4.2.2 Segmentos

Uma das diferenças fundamentais entre a sintaxe URI/IRI é a estrutura dos segementos do identificador. Enquanto que na arquitectura URI/XRI o segmento é um elemento opaco, na arquitectura XRI um segmento é formado por um ou mais subsegmentos. Esses subsegmentos podem ser de três tipos: globais, locais ou referências cruzadas.

4.2.2.1 Subsegmento global

O primeiro caracter tem de ser um símbolo de contexto global, ver Tabela 4.2, dando um contexto semântico ao identificador.

Símbolo	Contexto	Definição
=	Pessoal	Identificadores cuja autoridade pertence a um indivíduo.
@	Organizacional	Identificadores cuja autoridade pertence a uma organização
+	Geral	Identificadores para os quais não existe autoridade, não resolúveis, porque representam conceitos genéricos definidos em dicionários mantidos por consenso.
$	Especial	Identificadores cuja autoridade pertence à organização que define a norma. Apenas o OASIS XRI TC pode designar identificadores neste contexto.

Tabela 4.2: Símbolos de contexto global [11]

4.2.2.2 Subsegmento local

O carácter delimitador que precede o subsegmento define o tipo de subsegmento, ver Tabela 4.3. Um identificador apenas pode ser considerado persistente se todos os subsegmentos que o compõem forem persistentes.

Símbolo	Contexto	Definição
*	Reutilizável	A associação entre identificador e recurso é volátil, isto é, pode mudar se a autoridade do identificador o entender.
!	Persistente	A associação entre identificador e recurso é permanente, isto é, cumpre os mesmo requisitos especificados para o URN [40]

Tabela 4.3: Símbolos de contexto global [11]

4.2.3 Referência cruzada

Permite encapsular identificadores com outros esquemas que de outro modo não seriam considerados subsegmentos válidos de um XRI, permitindo manter a sintaxe original quando inseridos num identificador XRI, possibilitando a utilização e partilha de identificadores em diferentes contextos. Sintaticamente uma referência cruzada é distinguida ao inclui-la entre parênteses curvos.

4.3 Resolução

4.3.1 Arquitectura

A resolução de identificadores XRI segue a mesma arquitectura de resolução do DNS, ver secção 2.3 e 2.4, mas a um nível maior de abstração. Ao invés de utilizar UDP para resolver um *domain name* para um descritor baseado em texto, *Resource Record* (RR), utiliza HTTP para resolver o XRI num descritor baseado em XML, documento XRDS. Isto permite utilizar os mecanismo de *cache* já existentes na infraestrutura *web* actual assim como utilizar mecanismos de segurança existentes, HTTPS para garantia de confidencialidade e SAML para garantir autenticidade e integridade da informação.

A Tabela 4.4 contém uma comparação directa, onde possível, entre as arquitecturas de resolução XRI e DNS.

Resolution Component	DNS Architecture	XRI Architecture
Identifier	domain name	XRI (authority + path + query)
Resource record format	text (resource record)	XML (XRDS document)
Attribute identifier	string	anyURI
Network endpoint identifier	IP address	URI
Synonyms	CNAME	LocalID, EquivID, CanonicalID, CanonicalEquivID
Primary resolution protocol	UDP	HTTP(S)
Trusted resolution options	DNSSEC	HTTPS and/or SAML
Resolution client	resolver	resolver
Resolution server	authoritative nameserver	authority server
Recursing resolution	recursing nameserver	recursing authority server or proxy resolver

Tabela 4.4: Comparação entre a arquitectura de resolução DNS e XRI [8].

A Figura 4.5 ilustra as várias configurações possíveis em que os elementos da arquitectura podem interagir para resolver o identificador `xri://(tel:+1-201-555-0123)*foo*bar`. Qualquer um desses cenários pode envolver duas fases de resolução:

- **Resolução de autoridade**. Esta fase corresponde à resolução do segmento *Authority* do XRI para um descritor do recurso, esta é efectuada resolvendo sucessivamente os vários subsegmentos da esquerda para a direita. Para o XRI exemplificado necessita de resolver três subsegmentos: `(tel:+1-201-555-0123)`, `foo` e `bar`.
- **Selecção de *Service Endpoint* (SEP)**. Uma vez completa a resolução de autoridade existe uma fase opcional que possibilita a escolha de informação específica no descritor, XRD, consoante parâmetros cedidos por quem faz o pedido de resolução. Por exemplo, incluir no XRD apenas entradas relativas ao OpenID.

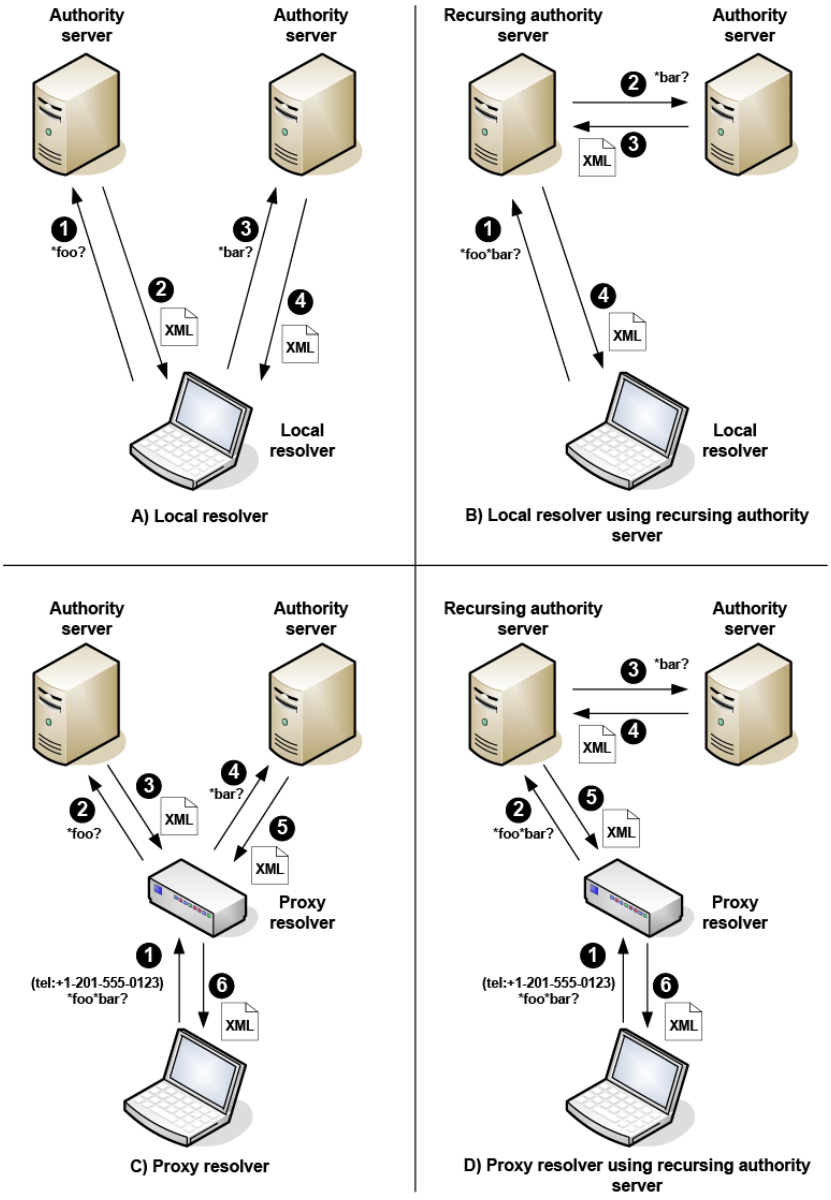

Figura 4.5: Cenários típicos de resolução XRI. [8]

4.3.2 *Proxy Resolver*

Para além de resolução recursiva, semelhante ao DNS, a arquitectura XRI suporta *proxy resolvers* disponibilizando um interface HTTP para um *resolver* local. Este mecanismo permite a aplicações resolver identificadores XRI mesmo que apenas suportem URLs.
O pedido de resolução é efectuado através de um URL especial, designado por *HTTP XRI* (HXRI), construído a partir do *endpoint* do *proxy resolver*, do XRI a resolver e os parâmetros de resolução pretendidos. Um exemplo da aplicação deste componente é a tecnologia OpenID, ver Capítulo 3, onde a aplicação RP pode contactar um *proxy resolver* ao invés de implementar um *resolver* local. A especificação [8] da resolução XRI contempla o mapeamento das opções de resolução num interface HTTP.

4.4 *eXtensible Resource Descriptor* (XRD)

Inicialmente desenvolvida sob a especificação referente à resolução XRI [8], a crescente utilização fora da arquitectura XRI (WebFinger, OpenID, OAuth, OExchange, entre outros) originou o desenvolvimento de uma especificação [12] independente com base no trabalho desenvolvido pelo *Internet Engineering Task Force* (IETF), *Web Linking* [41].
Um documento XRD é um formato simples e genérico, baseado em XML, com a função de descrever um recurso. Contém informação *machine-readable*, metadados, sobre recursos com o propósito de promover interoperabilidade assistindo a interacção entre recursos desconhecidos através de interfaces conhecidos.

4.4.1 Estrutura

Um documento XRD necessita ser um documento XML válido, a especificação apenas fornece os elementos base (Tabela 4.5) para suportar os casos de uso mais comuns.

`<XRD>`	Encapsula todo o descritor do recurso. Contém o atributo `xml:id` que fornece uma identificação única para o documento XRD e é utilizado como referência para assinatura do documento, caso exista.
`<Expires>`	Contém um valor temporal do tipo `xs:dateTime` definindo a validade temporal do documento XRD. Esta validade é independente do protocolo de transporte utilizado, por isso a validade no mecanismo de *cache* utilizado não deve ser superior ao do documento.
`<Subject>`	Contém um URI que identifica o recurso descrito pelo documento XRD. Não existir este elemento significa que o recurso deverá ser identificado por outro método. No caso de ser um XRI existe o elemento `CanonicalID`.
`<Alias>`	Fornece identificadores, URIs, adicionais para o mesmo recurso identificado pelo `<Subject>`.
`<Property>`	Declara uma propriedade do recurso quando utilizada como elemento filho do elemento `<XRD>` ou estabelece o tipo de relação quando é inserida dentro de um elemento `<Link>`. É sempre necessário incluir o atributo `type`, um URI específico de aplicações, que permite determinar o tipo de propriedade ou relação.
`<Link>`	Encapsula informação sobre a relação entre o recurso descrito no XRD e outro recurso.
`<Title>`	Não tem qualquer significado no documento XRD servindo apenas para contextualizar.

Tabela 4.5: Elementos base do XRD [12].

4.4.2 Extensões

É esperado que para algumas aplicações, consoante necessidade, se extenda o XRD para incluir mais informação sobre os recursos que descrevem. É o caso da arquitectura de resolução XRI que extende o esquema da especificação base para incluir novos elementos no documento XRD que suportem as funcionalidades desejadas.
Uma dessas funcionalidade é o suporte para uma arquitectura distribuída que requer mecanismos de *cache*, gestão de erros e delegação. A Tabela 4.6 contém os elementos mais comuns utilizados para esse efeito.

`<XRDS>`	Contentor para vários elementos `<XRD>`. No caso do pedido de resolução o explicitar, poderá receber os vários documentos XRD obtidos em cada passo da resolução encapsulados num elemento `<XRDS>`.
`<Query>`	Contém o identificador do qual resulta o documento descritor.
`<Status>`	Utilizado para conter mensagens de erro destinadas ao utilizador e indicar o resultado da verificação do `<CanonicalID>`.
`<ServerStatus>`	Indica o resultado do pedido de resolução semelhante ao utilizado no HTTP.
`<Redirect>`	Indica ao *resolver* para seguir uma localização onde pode encontrar outro documento XRD contendo informação sobre o recurso.

Tabela 4.6: Elementos de gestão.

A resolução XRI fornece alguns mecanismos que permitem verificar a autenticidade e integridade da informação devolvida sobre um determinado identificador, ver Tabela 4.7.

`<ProviderID>`	Contém um identificador persistente da autoridade que devolveu o descritor do recurso.
`<Assertion>`	Asserção SAML, sobre o documento XRD, emitida pela autoridade presente no elemento `<ProviderID>`.

Tabela 4.7: Elementos de autenticidade e integridade.

Outra extensão introduzida com a resolução XRI é o conceito de sinónimos que possibilita a camada de indirecção entre identificadores persistentes e identificadores reutilizáveis, ver secção 4.1.1, assim como a validação de sinónimos por parte do *resolver*. A Tabela 4.8 contém os elementos mais comuns.
Até agora os elementos apresentados apenas estão relacionados com o documento descritor do recurso e ainda não acrescentam informação de como interagir com o recurso descrito. Para esse efeito a resolução XRI introduz um grupo de elementos descritores de *Service Endpoint* (SEP), ver Tabela 4.9, com função de publicar *endpoints* que permitem a delegação de resolução, metadados, ou interagir directamente com o recurso. Adicionalmente declaram prioridades e tipos de serviços permitindo mecanismos de selecção automática.

`<LocalID>`	Contém um sinónimo persistente para o valor do elemento `<Query>`, ou seja, é um identificador local à autoridade do identificador.
`<CanonicalID>`	Contém um identificador absoluto atribuído pela autoridade ao recurso. Normalmente o identificador persistente equivalente. Por exemplo na arquitectura OpenID, após o processo de decoberta para um identificador XRI este deve ser o identificador para o qual o RP pede autenticação ao OP. Preferencialmente deverá ser o identificador utilizado pelas aplicações por ser, usualmente, um identificador persistente.

Tabela 4.8: Elementos de sinónimos.

`<Service>`	Elemento contentor para metadados sobre SEPs. A prioridade do SEP é definida através do atributo `priority`.
`<LocalID>`	Tem o mesmo significado que o elemento homónimo da Tabela 4.8 mas neste caso o identificador é atribuído pelo provedor do serviço e não pela autoridade do documento XRD.
`<URI>`	Indica um URI para acessar o serviço descrito pelo SEP. Pode existir mais do que um meio de comunicar com o serviço e ordenados consoante prioridades predefinidas.
`<Redirect>` **ou** `<Ref>`	Tal como no contexto `<XRD>` permite explicitar delegação.
`<ProviderID>`	Tem o mesmo significado que o elemento homónimo da Tabela 4.8 mas neste caso o identificador é atribuído pelo provedor do serviço e não pela autoridade do documento XRD. No caso de ser um serviço de resolução identifica a autoridade para a qual é delegada a resolução subsequente.
`<ds:KeyInfo>`	Contém a informação sobre a assinatura digital necessária para interagir com o recurso, normalmente um certificado digital.
`<Type>`	Um identificador único que identifica o tipo de capacidades disponíveis no SEP.
`<Path>`	Permite a seleção do SEP consoante o segmento `path` do identificador sobre o qual a resolução foi efectuada.
`<MediaType>`	Explicita o tipo de conteúdo [42] disponível.

Tabela 4.9: Elementos de *Service Endpoints*.

4.5 Considerações

A separação entre a especificação de resolução XRI [8] e a especificação do XRD [12] é talvez o indício maior de que a adopção do XRI ainda não é significativa nas emergentes tecnologias *web*. No entanto a influência de uma tecnologia com grande implementação como o OpenID pode trazer uma maior sensibilização para as vantagens do *framework* XRI, nomeadamente o suporte para confidencialidade/autenticidade na resolução e a utilização de identificadores abstractos, num cenário onde a necessidade de interoperabilidade entre os muitos serviços existentes é decisiva e necessária.
Até à data, existe pelo menos uma solução *open source*[2] para os vários componentes da arquitectura XRI, assim como uma implementação de I-Broker[3] que para além de fornecerem identificadores fornecem alguns serviços associados como *OpenID Provider* (OP).

[2]OpenXRI Project. `http://www.openxri.org/`
[3]ibrokerKit. `http://www.ibrokerkit.com`

Capítulo 5

XRI Data Interchange (XDI)

Existe uma preocupação crescente com a necessidade de tornar a informação acessível a máquinas (*machine-to-machine communication*) da mesma forma que está disponível para pessoas. Para que tal aconteça há a necessidade de existirem mecanismos que permitam a descoberta, contextualização e interoperabilidade entre as várias fontes. Com base nestes pressupostos têm sido feitos esforços em áreas como "*Semantic Web*" [15], "*Linked Data*"[15], entre outros. É neste contexto que surge a tecnologia XDI, ainda vista em alguns meios como uma tecnologia concorrente à *Resource Description Framework* (RDF) [16], mas que pode ser um passo importante para a implementação do conceito de "*Data Portability*" essencial numa arquitectura "*user-centric*".
O XDI é um *standard* aberto em desenvolvimento pelo OASIS XDI TC[1] que aplica os identificadores XRI de uma forma estruturada para resolver os problemas de partilha de dados, referência e sincronização independentemente do domínio, aplicação ou esquema. O objectivo é permitir que dados provenientes de qualquer fonte possam ser trocados, referenciados e sincronizados por um esquema passível de ser "*machine-readable*" do mesmo modo que o HTML tem feito na *Internet*.
Este capítulo aborda as funcionalidades presentes na especificação [17] até à data da elaboração desta Dissertação. Os Exemplos XDI presentes neste capítulo estão todos no formato "X3 Simple", mais legível. Para a transmissão de dados a representação utilizada deve ser o mais compacta possível como a "X3 Standard" ou "JSON" [17].

5.1 Modelo de representação XDI

O modelo de representação da informação presente no XDI tem a forma de um grafo estruturado através de declarações codificadas na forma de identificadores XRI compostos. Como consequência, todos os elementos presentes no grafo são endereçáveis. As declarações são estruturadas hierarquicamente na forma de um triplo: sujeito, predicado e objecto, sendo o contexto um conjunto de triplos (ver Figura 5.1). Ou seja, cada sujeito inicia um novo contexto o que implica que um grafo XDI é um conjunto de contextos distribuídos hierarquicamente.
O endereçamento de cada nó é obtido agregando o segmento XRI de cada nó com o caracter delimitador '/'. O grafo do Exemplo 7 contém dois contextos, o contexto raíz associado ao i-number do utilizador e o contexto relativo ao predicado `+personalinfo`. O sujeito com o segmento canónico '$' declara que os atributos do contexto são afirmações relativas ao

[1] `http://www.oasis-open.org/committees/xdi/`

predicado `+personalinfo`. Qualquer nó no grafo pode ser acedido através de um endereço XRI relativo a qualquer outro ponto. Por exemplo, o predicado `+firstname` é identificado pelo endereço `@!userinumber/+personalinfo//$/+firstname`. No caso em que o documento XDI está acessível *online* e a sua localização pode ser definida através de um *endpoint* concreto (ver secção 4.3) então qualquer nó, mediante autorização prévia, é acessível.

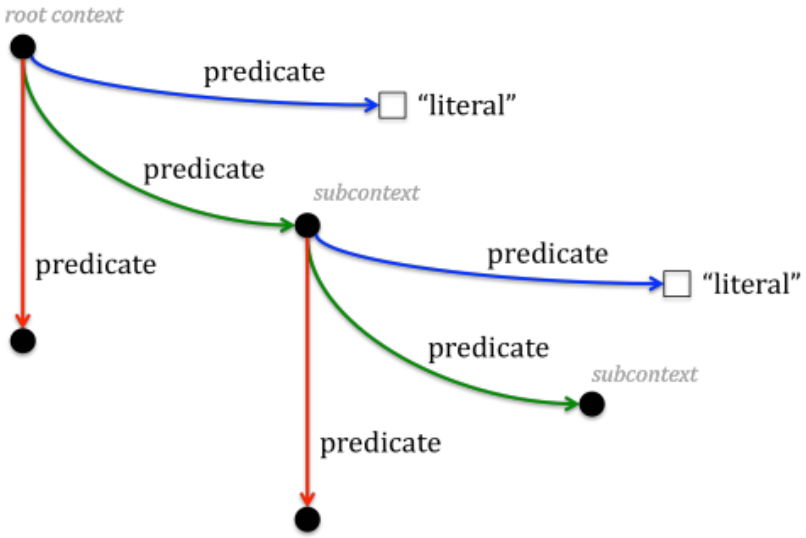

Figura 5.1: Grafo XDI [9].

```
@!userinumber
  +personalinfo
    /
      $
        +firstname
          "Nuno"
```

Exemplo 7: Grafo XDI com subcontexto.

5.2 Sintaxe base

A especificação [17] define quatro predicados base no dicionário reservado ao *standard* ,'$', que permitem expressar as relações fundamentais entre objectos XDI. São a base para definir novas ontologias. A Tabela 5.1 inclui cada um desses predicados e o seu inverso.

Predicado	Inverso	Relação
`$is`	`$is`	Equivalência, "o mesmo que"
`$a`	`$is$a`	Herança, "generilazação de"
`$has`	`$is$has`	Composição, "tem"
`$has$a`	`$is$has$a`	Agregação, "uma parte de"

Tabela 5.1: Predicados base XDI.

5.2.1 Operações

Existem quatro operações base, ver Tabela 5.2, que permitem operações atómicas no grafo XDI e uma operação abstracta com o intuito de extensibilidade para o suporte de novos tipo de operações. Para concretizar num exemplo: a extensão pode ser feita para um mecanismo de sincronização (e.g. `$sync`) que sendo uma extensão apenas seria possível entre sistemas que a implementassem.

Operação	Descrição
`$get`	Leitura de uma ou mais declarações presentes no grafo
`$add`	Escrita de uma ou mais declarações presentes no grafo
`$mod`	Modificação de uma ou mais declarações presentes no grafo
`$del`	Remoção de uma ou mais declarações presentes no grafo

Tabela 5.2: Operações base XDI.

5.2.2 Dicionário de tipos de dados

A raíz para este dicionário, (ver secção 5.5), é o predicado `$a` e serve para identificar o tipo de dados do sujeito que descreve. Não existem ainda dicionários de tipos estandardizados, apenas propostas sugeridas pelo XDI TC. Mais uma vez a extensibilidade é um objectivo de desenho e é expectável a evolução para dicionários definidos por consenso de utilização e provavelmente mapeados de outras implementações já disseminadas. Um exemplo de mapeamento sugerido pelo XDI TC são os "*IANA-specified MIME media types*" - `$a$mime` - e "*W3C-specified XML Schema datatypes*" - `$a$xsd`.
A conjugação com os predicados de operações possibilita a criação de mecanismos de negociação de conteúdo semelhante aos presentes no HTTP/1.1 [29]. O protocolo XDI não está dependente do protocolo de transporte, tornando este mecanismo uma mais valia quando não é utilizado HTTP. Por exemplo, uma operação `$get$mime$(application/pdf)` sobre o endereço `xri://@ua*deti/(nuno.rosa@ua.pt)+cv` retornaria o *Curriculum Vitae* do aluno no formato PDF.

5.2.3 *Link Contracts*

Um aspecto importante e assumido como objectivo de desenho pela especificação é o mecanismo de mediação do acesso aos dados, *link contract*. É também descrito através de

declarações XDI, o que o torna endereçável e parte do grafo. Esta característica facilita a portabilidade já que autorizações podem ser exportadas juntamente com os dados. Operações sobre os dados finais do grafo são realizadas através de contractos que após um processo de negociação entre ambas as partes e assinados digitalmente (nem sempre necessário) servem de mediador entre *caller* e *callee*, mantendo sempre um mediador para as operações. O *link contract* pode ser revogado a qualquer momento, dando um maior controlo a quem é autoritário sobre a informação. A Figura 5.2 ilustra o mecanismo conceptual de acesso a um recurso numa representação XDI. O nó endereçável pelo identificador `=alice$has$1$do$get` tem dois arcos que indicam relação (tracejado); um com o literal "`+1.206.111.1111`"indicando que a permissão para a operação `$get` se aplica a `=alice+tel/!1` e outro que se aplica à secção `$do` no *link contract* `=alice$has$1`. Este por sua vez tem um arco relacional que aponta para o sujeito `=bob` indicando que faz parte do contexto do *link contract*. O Exemplo 8 demostra o mesmo conceito mas através de um grafo XDI no formato "X3 Simple".

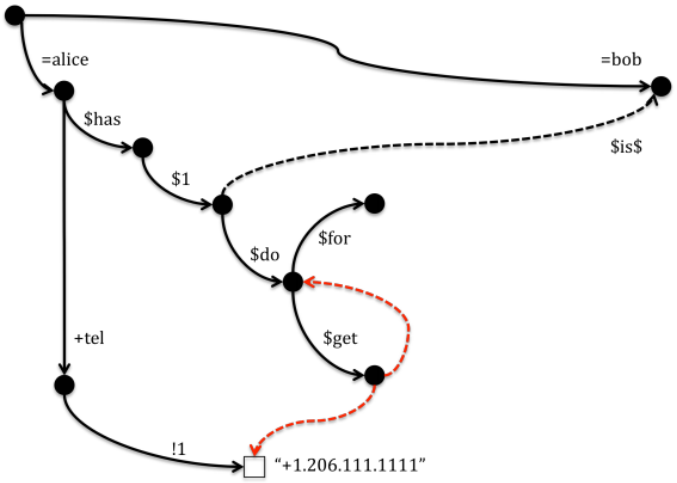

Figura 5.2: Padrão *link contract* [9].

Existe uma diferença na representação da referência para as partes envolvidas no contracto entre a Figura 5.2 e o Exemplo 8. A especificação está a evoluir para uma proposta em que o predicado `$` represente um conjunto de subcontextos retirando a necessidade de utilizar o `$has$a`, no entanto o grafo da Figura 5.2 ainda não consta do estado actual, até à data, da especificação. A representação seguida na implementação da prova de conceito segue o padrão do Exemplo 8.

```
@alice
  +tel$!1
    "+1.206.111.1111"
  $has$a <--Link Contract Aggregation-->

@alice$has$1 <--Link Contract-->
  $is$a
    $has
  $has$a
    $a        <--Link Contract Parties Aggregation-->
  $do$get
    /
      @alice
        +tel$!1

@alice$has$1$a <--Link Contract Parties-->
  $is$a
    $a
  $is$has
    =bob
```

Exemplo 8: Padrão *link contract*.

5.3 Mensagens

Com a permissa de que o XDI é um grafo global totalmente endereçável as mensagens também são representadas no mesmo formato e todos os seus nós endereçáveis. As mensagens seguem o seguinte padrão: o identificador XRI de quem faz o pedido (sujeito), um ou mais predicados representando operações que expressam subcontextos com declarações XDI sobre a porção do grafo onde deve ser executado. Este é o padrão base podendo conter outras informações como um predicado contendo uma assinatura, `$sig`, que possibilite a verificação da autenticidade e integridade da mensagem, ver Exemplo 9.

```
@!userinumber
  $sig
    "..."
  $get
    /
      @ua*deti
        +horario+eet
```

Exemplo 9: Padrão de mensagem XDI.

As operações presentes na mensagem não seguem uma ordem predefinida a não ser que isso seja explicitado. É portanto necessário ter em atenção, por parte da aplicação que envia, se as várias operações não entram em conflito. Para colmatar o problema existe uma sintaxe para expressar a ordem (não se restringe ao ordenamento das operações). Ver Exemplo 10.

```
@!userinumber
  $order
    /
      $
        $1
      $get
        $2
      $del
  $del
    /
      @!userinumber
        +tel+mobile
  $get
      /
        @!userinumber
          +tel+mobile
```

Exemplo 10: Padrão de mensagem XDI com operações ordenadas.

5.4 Versionamento

De modo a suportar sincronaização de dados a especificação XDI define formas de expressar versões de qualquer parte do grafo, de forma análoga ao que outros sistemas de versionamento como *wikis* suportam. O documento guarda informação para que mesmo após alterações no grafo seja possível reconstruir o estado que existia num determinado momento anterior (versionamento do tipo *log*). Outro tipo de versionamento, *snapshot*, guarda uma cópia integral de uma versão específica. A indicação de que um componente do grafo suporta versionamento é dada através de um segmento XRI composto com o predicado `$v` como se demonstra nos exemplos seguintes. O Exemplo 11 indica o estado inicial do grafo que após a mensagem do Exemplo 12 toma o padrão do Exemplo 13.

```
@!userinumber
  $has
    $v  <--versioning support-->
  $v
    $1  <--current version-->
  +tel+mobile
    "+351.961.234.567"
@!userinumber$v
  $has
    $1
  $has$a
    @!userinumber$v$1  <--version log-->
      /  <--copy of the operations performed -->
@!iserinumber$v$1  <--version snapshot-->
  +tel+mobile
    "+351.961.234.567"
```

Exemplo 11: Versionamento, estado inicial.

```
@!userinumber
  $sig
    "..."
  $mod
    /
      @!userinumber
        +tel+mobile
          "+351.967.654.321"
```

Exemplo 12: Versionamento, mensagem recebida.

```
@!userinumber
  $has
    $v  <--versioning support-->
  $v
    $2  <--current version-->
  +tel+mobile
    "+351.967.654.321"

@!userinumber$v
  $has
    $1
  $has$a
    @!userinumber$v$1  <--version log-->
      /  <--copy of the operations performed -->
    @!userinumber
      $sig
        "..."
      $mod
        /
          @!userinumber
            +tel+mobile
              "+351.967.654.321"

@!iserinumber$v$1  <--version snapshot-->
  +tel+mobile
    "+351.961.234.567"

@!iserinumber$v$2 <--version snapshot-->
  +tel+mobile
    "+351.967.654.321"
```

Exemplo 13: Versionamento, grafo após a execução da mensagem.

5.5 Dicionários

Do mesmo modo que o XML tem *schemas*, o RDF ontologias, o XDI recorre a dicionários. Em todos eles o objectivo é o mesmo, definir vocabulários que permitam a aplicações processar as relações presentes no conteúdo e não apenas apresentá-la ao utilizador. A representação XDI diferencia-se das restantes na forma como representa os dicionários. Também estes são documentos XDI construídos a partir da sintaxe base (ver secção 5.2). Para cada domínio específico cada comunidade de interesse deve chegar a consenso sobre os dicionários a adoptar. O Exemplo 14 apresenta uma pequena parte de um dicionário XDI directamente traduzido de uma ontologia RDF para representar cerveja[2].

```
+Beer
  $has
    +madeFrom
    +awarded
    +brews
    +hasAlcoholicContent
    +hasOriginalWortContent
+madefrom
  $has
    +Ingredient
+awarded
  $has
    +Award
+Brews
  $has
    +Brewary
+hasAlcoholicContent
  $is$a
    $a$xsd$float
+Award
  $has
    +awardedAt
    +awardCategory
+Water
  $is$a
    +Ingredient
+Malt
  $is$a
    +Ingredient
```

Exemplo 14: Parte de dicionário XDI para "cerveja".

Do dicionário parcial do Exemplo 14 retirara-se um conjunto de relações entre os vários objectos. A cerveja tem como ingredientes instâncias do tipo `+Ingredient` referenciados através do predicado `+madeFrom` e que o teor alcoólico da cerveja é apresentado como um

[2]Beer Ontology, `http://www.schemaweb.info/schema/SchemaInfo.aspx?id=99`

float. O Exemplo 15 ilustra como a partir da afirmação "*o* ***Utilizador gosta*** *de* ***Erdinger***" é possível descobrir mais informações sobre a cerveja, se houver conhecimento prévio do dicionário. A aplicação através da referência "`@ErdingerWeißbru+beer+erdinger`" faz um pedido de contexto à autoridade que administra o contexto global, `@`, sobre a localização do contexto XDI `@ErdingerWeibru` e envia um pedido a requerer o objecto `+erdinger` no contexto `+product`.

```
<--User XDI document-->
@!userinumber
  +likes
    @ErdingerWeibru+product+erdinger

<--XDI document stored in a diferent location-->
@ErdingerWeibru+product+erdinger
  $is$a
    +beer
  +madeFrom
    +Water
    +WheatMalt
    +Hops
    +Yeast
  +brews
    @ErdingerWeibru
  +hasAlcoholicContent
    "5.3"
  +hasOriginalWortContent
    "13"
```

Exemplo 15: Utilização do dicionário XDI para "cerveja".

5.6 *Resource Description Framework* (RDF) e paralelismos com o XDI

O conceito por detrás da representação de dados no *Resource Description Framework* (RDF) é semelhante ao descrito anteriormente no XDI já que o trabalho desenvolvido no RDF serviu como base para a especificação XDI. Esta secção explica sucintamente as funcionalidades associadas ao RDF e quando possível estabelece paralelismos com mecanismos XDI expostos nas secções anteriores.
O RDF surgiu com o propósito de criar uma linguagem estruturada capaz de caracterizar objectos, digitais ou não, de forma *machine-readable* mantendo o significado semântico entre aplicações. A especificação consiste num conjunto de documentos [16, 10, 43, 44, 45] desenvolvidos pelo RDF *Core Working Group* como parte do W3C *Semantic Web Activity* que estabelecem a sintaxe, vocabulário e a semântica para a descrição de objectos.

5.6.1 Modelo de representação RDF

A sua estrutura representa um grafo que agrega um conjunto de descrições na forma de triplos, ver Figura 5.3. Estabelecem a relação (predicado) entre um sujeito e um objecto. No RDF os recursos são identificados através de URIs e descritos em pares de propriedades e literais. Isto permite descrever recursos através de declarações na forma de um grafo de nós e arcos representando relações e propriedades.

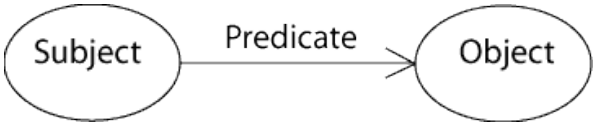

Figura 5.3: Declaração RDF [10].

Por exemplo, a expressão "o **Utilizador gosta** de **Erdinger**" pode ser traduzida por um triplo em que o sujeito é "Utilizador", "gosta" é o predicado e "Erdinger" é o objecto. Utilizando URIs para representar cada elemento da declaração obtém-se o seguinte em notação *triple* [10]: `<http://example.com/user> <http://example.com/social#likes> <http://www.erdinger.com/product#erdinger>`.

Para trocar informação entre domínios descrita no formato RDF a especificação [43] define uma sintaxe XML para os documentos. Aproveitando a funcionalidade dos *namespaces* do XML podemos agregar vocabulário comum sob o mesmo *namespacce*, apenas por razões de comodidade já que implementações de *parsing* expandem as referências URI para obter um URI completo que seja inequivocamente identificável. A Tabela 5.3 contém os elementos base do RDF/XML.

Elemento	Descrição
`<rdf:RDF>`	define o documento XML como sendo um RDF e define os *namespaces* presentes no documento
`<rdf:Description>`	elemento que define qual o objecto a ser descrito através do atributo `about` e contém os restantes elementos que descrevem o objecto.
`<namespace:property>`	elementos que descrevem o objecto, podem ser atributos da *tag* `<rdf:Description/>` ou elementos contidos na *tag* `<rdf:Description/>` em que o *namespace* e *property* dependem do vocabulário a utilizar.

Tabela 5.3: Elementos RDF/XML.

5.6.2 Ontologias

Como mencionado na secção 5.5 o equivalente aos dicionários XDI são as ontologias (*Web Ontology Language* (OWL)) que estendem o esquema RDF (RDFS) para definir taxinomias sobre os dados a representar e destinadas a serem processadas por aplicações. Aproveitando o exemplo dado na secção 5.5, o Exemplo 16 contém uma parte da ontologia[3] para descrever

[3] `http://www.schemaweb.info/webservices/rest/GetRDFByID.aspx?id=99`

objectos do tipo “cerveja” definindo restrições e estrutura para objectos do referido tipo.

```
<rdf:RDF>
 <owl:Ontology rdf:about="">
  <owl:versionInfo>
    beer_v0.4.owl, based on http://purl.org/net/ontology/beer_v0.3.owl
  </owl:versionInfo>
  <rdfs:label>Beer Ontology, OWL Lite</rdfs:label>
 </owl:Ontology>
 <owl:Class rdf:ID="Beer"/>
 <owl:ObjectProperty rdf:ID="madeFrom">
  <rdfs:range rdf:resource="#Ingredient"/>
  <rdfs:domain rdf:resource="#Beer"/>
 </owl:ObjectProperty>
 <owl:ObjectProperty rdf:ID="brews">
  <rdfs:range rdf:resource="#Beer"/>
  <rdfs:domain rdf:resource="#Brewery"/>
 </owl:ObjectProperty>
 <owl:DatatypeProperty rdf:ID="hasAlcoholicContent">
  <rdfs:subPropertyOf rdf:resource="#hasContent"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
  <rdfs:domain rdf:resource="#Beer"/>
  <rdf:type
    rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
 </owl:DatatypeProperty>
 <owl:Class rdf:ID="Hops">
  <rdfs:label xml:lang="en">Hops</rdfs:label>
  <rdfs:subClassOf rdf:resource="#Ingredient"/>
  <rdfs:label xml:lang="en">Vine</rdfs:label>
  <rdfs:label xml:lang="de">Hopfen</rdfs:label>
 </owl:Class>
</rdf:RDF>
```

Exemplo 16: Parte da ontologia para “cerveja”.

5.6.3 Acesso à informação

O Exemplo 17 é modelo RDF que descreve a relação entre o “Utilizador” e a cerveja “Erdinger”e está alojado no serviço associado ao utilizador. O Exemplo 18 no formato RDF descreve a cerveja enquanto produto e está alojado nos servidores da marca de cerveja que o disponibiliza para que possam ser feitas considerações sobre ele.

O mecanismo utilizado para que a partir da descrição da relação do utilizador descobrir os detalhes da cerveja é semelhante ao do exemplo dado para o XDI, mas neste caso não existe o processo de resolução para encontrar a localização do contexto. A aplicação consumidora do RDF através do atributo `rdf:resource` presente no predicado que descreve a relação (`<social:likes/>`) encontra a localização do objecto sobre o qual é feita a afirmação. Efectua um pedido HTTP para o URI `http://www.erdinger.com/product/beer/2345` que

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:social="http://example.com/social#"
  xmlns:beer="http://www.purl.org/net/ontology/beer#">
  <rdf:Description rdf:about="http://example.com/user/1234">
  <social:likes rdf:resource="http://www.erdinger.com/product/beer/2345"/>
  </rdf:Description>
</rdf:RDF>
```

Exemplo 17: Parte da descrição do utilizador em RDF/XML.

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:beer="http://www.purl.org/net/ontology/beer#">

  <rdf:Description rdf:about="http://www.erdinger.com/product/beer/2345">
  <beer:hasAlcoholicContent>5.3</beer:hasAlcoholicContent>
  <beer:hasOriginalWortContent>13</beer:hasOriginalWortContent>
  <beer:madeFrom>
    <rdf:Bag>
      <beer:Water />
      <beer:WheatMalt />
      <beer:Hops />
      <beer:Yeast />
    </rdf:Bag>
  </beer:madeFrom>
  </rdf:Description>
</rdf:RDF>
```

Exemplo 18: Descrição do produto "cerveja" em RDF/XML.

devolve o documento RDF do Exemplo 18. Este é o modelo simples de publicar "*Linked Data*", onde existem *named graphs* que são identificados por uma estrutura de URI definida pelo sistema que publica a informação. A aplicação pode explicitar que deseja receber a informação no formato RDF através do mecanismo de negociação de conteúdo presente no HTTP/1.1 [29] indicando no cabeçalho o tipo `application/rdf+xml` ou descobrir a localização da representação no documento HTML devolvido [46].
Ao contrário do XDI, o RDF não estipula nenhum controlo de acesso delegando a responsabilidade para a implementação do sistema que deve utilizar o mecanismo mais apropriado para mediar o acesso.

5.7 Considerações

O foco principal deste capítulo foi demonstrar não só as funcionalidades presentes na especificação mas também desmonstrar que todos os mecanismos presentes no XDI são representados da mesma forma que a informação. Esta é uma enorme vantagem em termos de portabilidade e de criação de ferramentas que partilhem um modelo de sintaxe único. O RDF no campo de descrição de recursos disponíbiliza mais ferramentas para uma *web* semântica, no entanto a sua complexidade e dispersão de especifícações tornam difícil a sua implementação, um dos objectivos do XDI é criar mecanismos simples abrangendo o essencial numa só especificação.

Capítulo 6

Prova de conceito, Gestão de Identidade baseada em XRI/XDI

Este capítulo aborda a implementação da prova de conceito desenvolvida tendo como objectivo demonstrar as potencialidades do *framework* XRI/XDI, em conjugação com outras tecnologias actuais, numa solução que coloca o utilizador no centro de decisão sobre a gestão dos seus dados pessoais. Os elementos desenvolvidos consistem numa aplicação *web* com a qual o utilizador interage para realizar todas as acções no sistema e um elemento servidor que comunica com a aplicação *web* ou outras instâncias que falem XDI.

6.1 Arquitectura

A ideia de criar um sistema de raíz que consiga satisfazer todas as funcionalidades encontradas nos serviços da *web* social actuais não é de momento um cenário actractivo. No entanto a inclusão desses serviços como repositórios de informação permite a sua inclusão como os silos de informação em que se transformaram e realizar uma agregação semântica numa camada superior independente. Por exemplo, o utilizador pode ter associadas contas em serviços como Flickr e Facebook que para o consumidor da informação não é notória esta distinção nem as permissões de acesso aplicadas no domínio de cada serviço. A Figura 6.1 representa a arquitectura proposta e os componentes intervenientes.

- **I-Broker**[1]: Este componente permite ao utilizador que obtenha um identificador único e universal XRI, essencial à agregação dos vários serviços associados. Ao identificador é associado um certificado digital numa estrutura *Public Key Infrastructure* (PKI) permitindo a garantia de autenticidade e integridade na troca de mensagens XDI assim como assinar *link contracts*. É este o componente a contactar para a resolução dos identificadores XRI sobre os quais detém autoridade, participando na descoberta do(s) *endpoint(s)* XDI de outro utilizador.

- ***Personal Data Services* (PDS)**: Não é obrigatório que toda a informação relacionada com o utilizador esteja na mesma localização, pode existir uma federação de vários. Por exemplo os dados de identificação num serviço disponibilizado pelo Estado e os contactos num serviço do ISP. Informação fora do mundo XDI, blogs, Facebook, Twitter, entre

[1] e.g. FullXRI em http://www.fullxri.com

outros, é acedida através de um módulo específico no PDS e mapeada numa representação XDI para cada serviço. Este último caso é o componente da arquitectura onde se situa o protótipo desenvolvido. Devido à diferente representação da informação assim como diferentes processos de autenticação torna-se necessário o desenvolvimento de um novo módulo no PDS para cada; excepção feita a serviços que partilhem de uma representação comum mesmo que não nativa ao XDI, e.g. Activity Streams [47].

- **Aplicações XDI**: Comunicam com o PDS através de um *endpoint* XDI, o acesso é mediado através do mecanismo de "*Link Contract*". As mensagens têm de estar assinadas digitalmente para garantir a autenticação de quem faz o pedido. Apenas podem ser realizadas operações sobre os dados autorizados pelo utilizador. As permissões podem ser geridas através do *front end* do PDS ou modificando o grafo directamente através do *endpoint* XDI, mediante autorização prévia.
- ***Front End***: Este elemento auxilia o utilizador na gestão dos seus dados e autorizações a partir de um interface *web* que comunica com o servidor recorrendo a chamadas remotas.
- **Serviços *web* sociais**: Dentro da arquitectura proposta servem como repositório de informação relacionada com o utilizador.

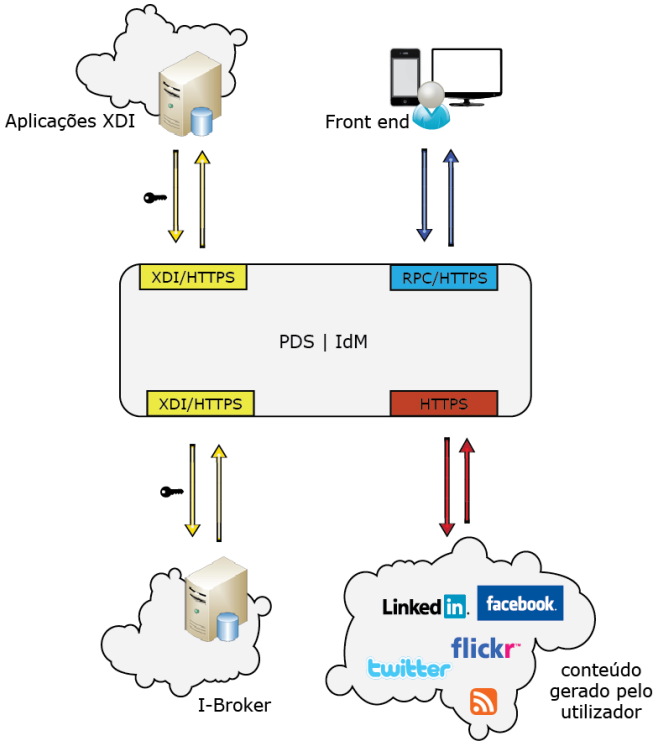

Figura 6.1: Arquitectura da prova de conceito.

6.2 Tecnologias e Ambiente de Desenvolvimento

A escolha de ferramentas a utilizar foi limitada pela estado embrionário da especificação XDI[17]. No XDI TC, responsável pela elaboração do documento, ainda se discutem alguns

mecanismos essenciais como formatos de serialização, sintaxe e semântica de "*link contracts*", entre outros, que farão parte da especificação final. Devido a este cenário, as bibliotecas disponíveis para lidar com XDI são reduzidas, pelo menos de livre acesso. A escolha recaiu para a biblioteca com o maior número de funcionalidades, integrada no projecto Higgins[48] e mantida por um membro do XDI TC, `XDI4J`[49]. Sendo a implementação da biblioteca XDI em `Java`, a escolha das restantes ferramentas foi para soluções baseadas na plataforma `Java`.

6.2.1 Google App Engine

É um serviço que permite correr aplicações *web* na estrutura da Google, pensado para aplicações com requisitos de escalibilidade e custo inicial baixo. A conjugação destes dois factores torna-o uma opção apetecível para que qualquer pessoa detentora de uma conta Google possa controlar a sua própria instância PDS. Finalmente uma das vantagens para o utilizador final, o processo de *deploy* resume-se a uma linha de comando facilitando a criação de ferramentas automáticas para a instalação do serviço para utilizadores com pouco *background* técnico.

6.2.2 OpenXri

É um projecto [50] *Open Source* com o objectivo de desenvolver aplicações e bibliotecas para a tecnologia XRI. Os módulos contidos abrangem sintaxe, clientes(*resolvers*) e todos os componenentes necessários para implementar um elemento de autoridade de resolução XRI. No protótipo implementado foi utilizada no módulo de descoberta de comunicação com o I-Broker permitindo a resolução de identificadores XRI quando necessário, tal como, descoberta de chave pública, *endpoint* XDI respectivo ao utilizador no I-Broker e descoberta do *endpoint* XDI da implementação PDS.

6.2.3 XDI4J

A biblioteca [49] permite realizar operações sobre dados modelados numa estrutura XDI, grafos. A Figura 6.2 apresenta a sua arquitectura.

As principais funcionalidades de interesse para a implementação são:

- **Grafos**: Interfaces para um modelo de dados nativo XDI na forma de grafo.
- ***Input/Ouput***: Serializar e desserializar dados XDI em vários formatos.
- **Enderaçamento**: Converter afirmações XDI para endereços XRI e vice-versa.
- ***Link Contracts***: Suporte base para o mecanismo de *link contracts* sobre grafos XDI
- **Messagens**: Envio e recepção de mensagens XDI suportando as operações base, `$get`, `$add`, `$do`, `$mod`.
- **Pesquisa**: Mecanismo simples de pesquisa num grafo XDI.
- **Assinaturas**: Assinatura e verifação de mensagens XDI.

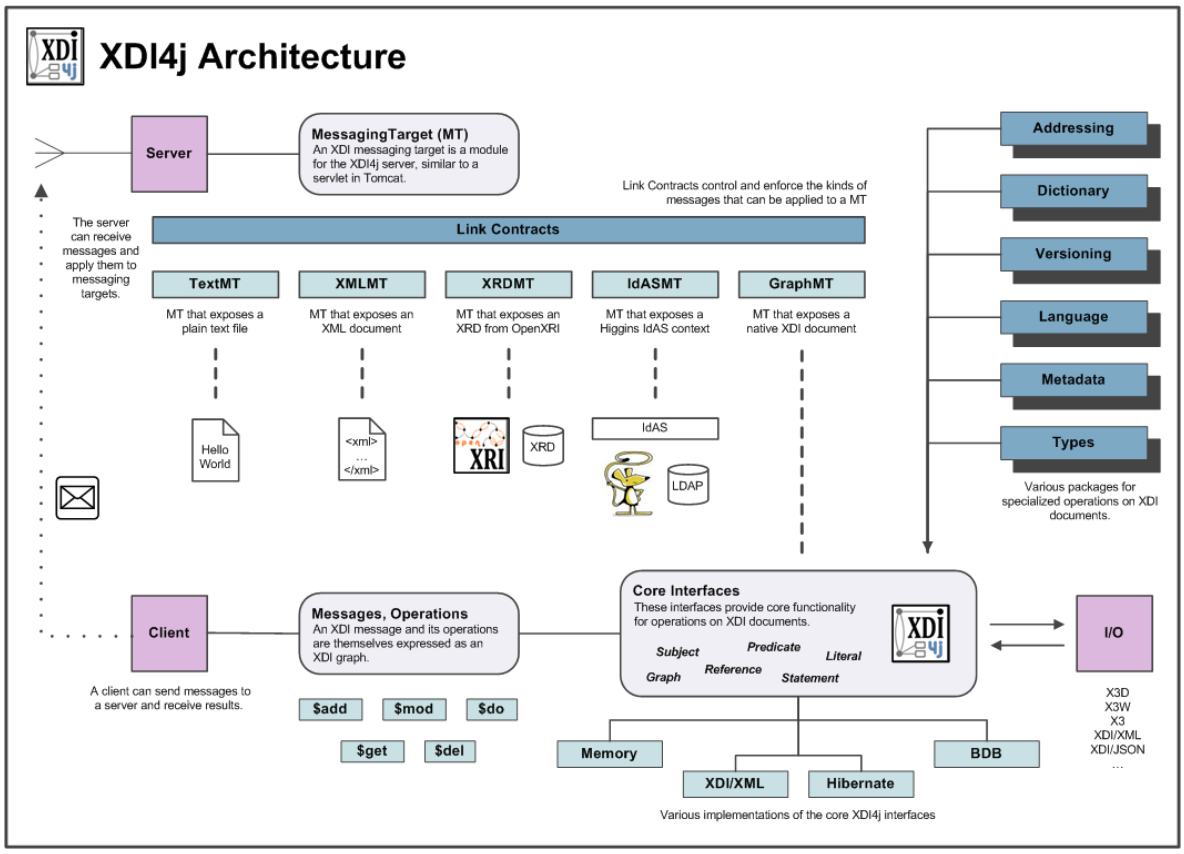

Figura 6.2: XDI4J - Arquitectura [49].

6.2.4 Google Web Toolkit (GWT)

Foi a escolha natural da plataforma para desenvolver o *front end* devido à integração, nomeadamente RPC, com aplicações alojadas no App Engine. Consiste num conjunto de ferramentas destinado a contruir aplicações *web* ricas utilizando as comodidades da plataforma Java (testes unitários, *strong typing*, entre outros) na fase de desenvolvimento e compilando o resultado final para javascript.

6.3 Perspectiva funcional

Para a demonstração da prova de conceito foram impostos que os seguintes requisitos funcionais fossem cumpridos.

- O Utilizador conseguir associar serviços *web* sociais ao seu perfil.
- O Utilizador visualizar as suas imagens alojadas em serviços associados.
- O Utilizador definir permissões para cada uma das imagens alojadas em serviços associados.

- O Utilizador definir permissões para cada atributo da sua informação pessoal (nome, sobrenome e email).
- O Utilizador definir permissões para a descoberta dos serviços que tem associados ao seu perfil.
- O Utilizador associar fontes ou amigos através de um i-name.
- O Utilizador visualizar as imagens, com permissão, de outro Utilizador.
- O Utilizador exportar os seus dados para uma representação XDI.
- O Sistema conseguir descobrir os detalhes relativos ao certificado associado ao i-name do Utilizador, nomeadamente chave privada e chave pública.
- O Sistema responder a pedidos XDI com suporte para "*link contracts*".
- O Sistema verificar autenticidade de mensagens recebidas.
- O Sistema realizar pedidos XDI a outros *endpoints* XDI.
- O Sistema descobrir o *endpoint* XDI de outros Utilizadores.
- O Sistema descobrir os serviços associados de outros Utilizadores.

6.4 Modelação de domínio

Da análise dos requisitos funcionais obtém-se uma representação alto nível dos dados envolvidos no sistema, as suas relações e quais têm representação XDI possibilitando o acesso através do *endpoint* XDI. A Figura 6.3 ilustra essa representação. O diagrama da Figura 6.4 ilustra as activades que ocorrem no sistema e actores com os quais interage.

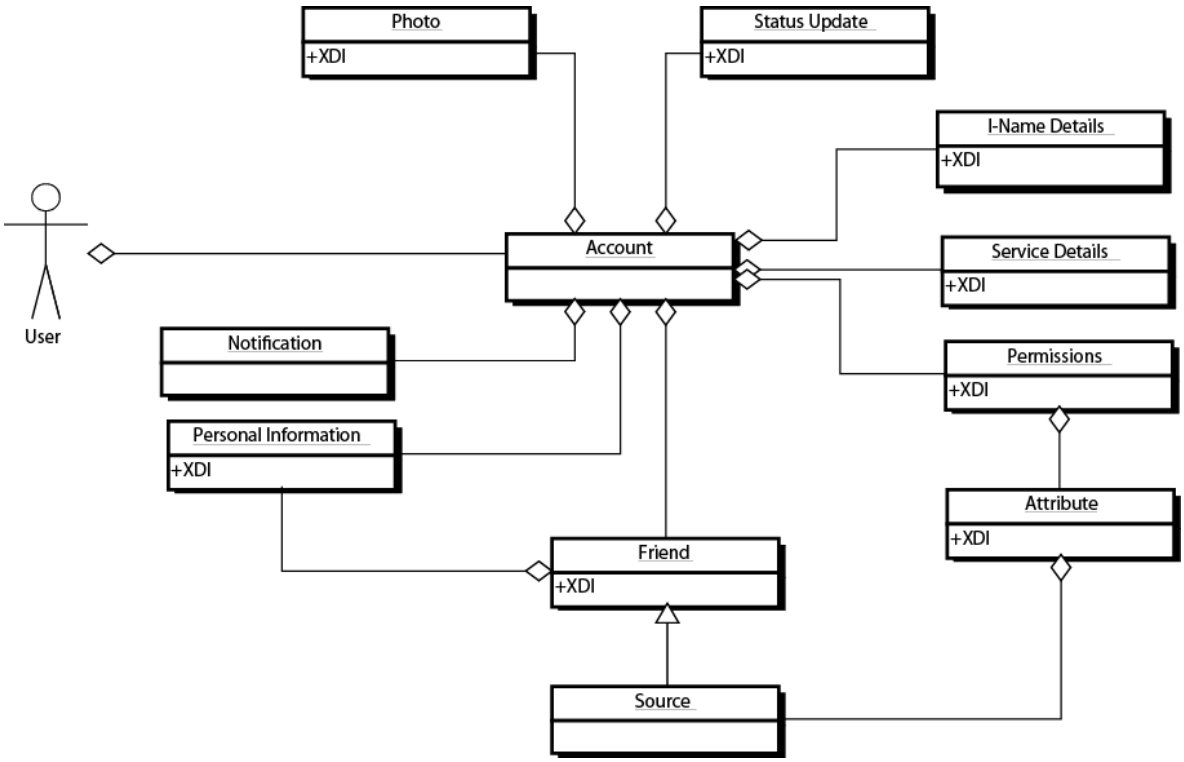

Figura 6.3: Modelo de domínio

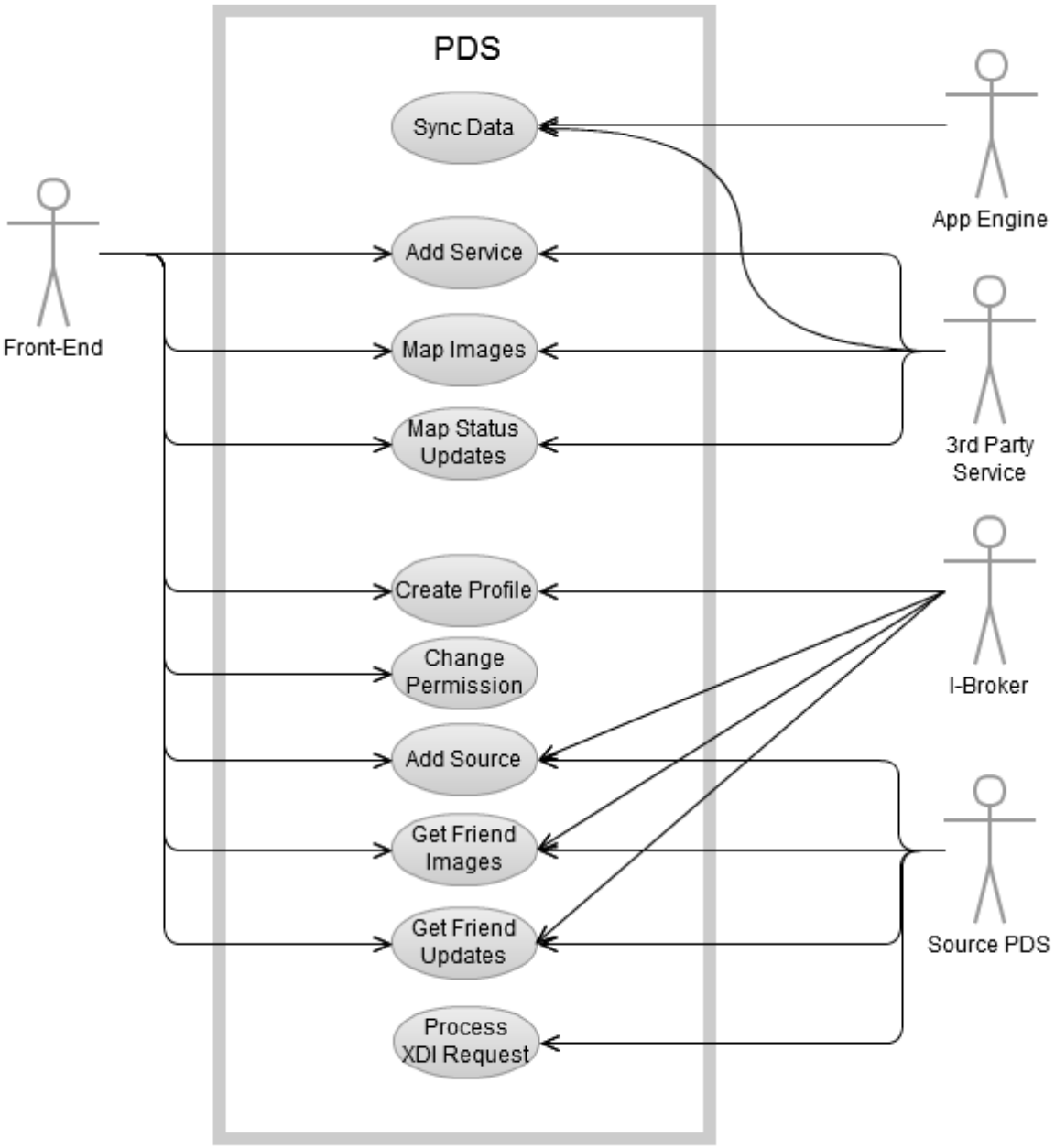

Figura 6.4: Diagrama de casos de utilização do sistema

6.5 Dicionários XDI

A partir do modelo da Figura 6.3 estabelecemos um conjunto de dicionários XDI que descrevem a identidade do utilizador. A definição de dicionários comuns permite a interoperabilidade entre elementos XDI assim como a portabilidade da informação, ver secção 5.5. Nesta Dissertação é apresentada uma proposta para dicionários e padrões XDI para a representação de dados pessoais localizados em redes sociais, particularmente fotografias, *status updates* e informação pessoal simples. O dicionário que descreve permissões sobre a informação faz parte do mecanismo de *link contracts* abordado no Capítulo 5. Nas subsecções seguintes são apresentados os dicionários propostos.

6.5.1 *personas*

O dicionário seguinte representa os elementos que uma classe `+person` ou que a extenda deve conter para representar uma identidade associada a serviços *web* sociais: informação relativa a certificado digital, informação pessoal simples do utilizador, serviços associados, fotografias, mensagens de *status* e indicação de interacção com outras identidades (*friending*).

```
+person
    $description
        "Person Object identified by a XRI"
    $has
        $key$private
        $key$public
        $certificate$x.509
        +personalinfo
        +onlineservice
        +photos
        +status
        +friends
$key$private
    $is$a
        $xsd$string
$key$public
    $is$a
        $xsd$string
$certificate$x.509
    $is$a
        $xsd$string
+friends
    $is$has
        +person
    $description
        "A container for friends or sources XRI identifiers"
```

6.5.2 Informações de contacto

Através deste dicionário aplicações XDI podem descobrir informação, sobre um i-name, a mostrar ao utilizador final. Por exemplo nesta prova de conceito, por defeito é explícito no *link contract* público o acesso ao primeiro (`@!userinumber/+personalinfo//$/+firstname`) e último nome (`@!userinumber/+personalinfo//$/+lastname`) para que ao adicionar um i-name como fonte/amigo poder mostrar ao utilizador informação com maior contexto do que apenas o i-name.

```
+personalinfo
    $description
        "Personal Information Container"
    $has
        +firstname
        +lastname
        +email
        +address
    $is$has
        +person

+firstname
    $is$a
        $a$xsd$string
+lastname
    $is$a
        $a$xsd$string
+email
    $is$a
        $a$uri$mailto
+address
    $is$a
        $a$xsd$string
```

6.5.3 Contas *web* sociais

Ao associar serviços *web* a informação que permite identificar o utilizador no serviço é guardada segundo o dicionário seguinte. Aplicações XDI podem contactar o PDS para saber a lista de serviços a que o i-name está associado, se autorizadas, através do endereço `@!userinumber/+onlineaccount$has$a` que lista as várias agregações do predicado `+onlineaccount`.

```
+onlineaccount
    $description
        "Online account details"
    $has
        +network
        +userid
    $is$has
```

```
        +person

+userid
    $is$a
        $xsd$string
+network
    $is$a
        $xsd$string
```

6.5.4 Imagens (Fotos)

O dicionário seguinte define a estrutura em que as imagens são representadas. O grafo do utilizador no PDS apenas contém o atributo `+id` e o atributo `+network` que permitem identificar o objecto no serviço onde está alojado. Quando aplicações XDI requerem os dados o serviço é contado para completar a estrutura.

```
+photos
    $is$has
        +person
    $description
        "Photo Container"
    $has
        +photo

+photo
    $is$has
        +photos
    $description
        "A XDI object representing a photo"
    $has
        +id
        +network
        +objectType
        +name
        +fullimage
        +thumbnail
        +description
        +permalink
+id
    $is$a
        $xsd$string
+name
    $is$a
        $xsd$string
+fullimage
    $is$a
        $uri$url
```

```
+thumbnail
    $is$a
        $uri$url
+description
    $is$a
        $xsd$string
+permalink
    $is$a
        $uri$url
```

6.5.5 "*status updates*"

O dicionário seguinte define a estrutura em que as mensagens de *status* são representadas. O grafo do utilizador no PDS apenas contém o atributo `+id` e o atributo `+network` que permitem identificar o objecto no serviço onde está alojado. Quando aplicações XDI requerem os dados o serviço é contado para completar a estrutura.

```
+status
    $is$has
        +person
    $description
        "A list of Status updates"
    $has
        +entry

+entry
    $is$has
        +status
    $description
        "A XDI Object representing a status update"
    $has
        +id
        +network
        +objectType
        +content
        +permalink
        $d
+objectType
    $is$a
        $uri$url
+content
    $is$a
        $xsd$string
+permalink
    $is$a
        $uri$url
$d
```

```
$is$a
    $xsd$timestamp
```

6.6 Implementação

6.6.1 Persistência

A solução de persistência presente no App Engine não suporta a implementação para os componentes nativos do grafo XDI presentes na biblioteca XDI4J, ver Figura 6.2. O *package* `pt.ua.xdi.impl.gae` contém a implementação, ver Figura 6.5, dos interfaces presentes no XDI4J com a adição da lógica necessária para guardar alterações no grafo correspondente do utilizador. A informação guardada na *datastore* está sempre na forma de grafo XDI. Para facilitar a manipulação dos grafos respeitando os dicionários descritos na secção 6.5, o *package* `pt.ua.xdi.dictionary` contém classes que abstraem a representação XDI, ver secção 6.5, e facilitam a sua manipulação construindo instâncias a partir de componentes do grafo, permitindo manipular o grafo XDI sem necessitar saber a sua estrutura.

- A Figura 6.6 mostra o diagrama de classes referente a indivíduos respeitando o dicionário para objectos do tipo `+person`.
- A Figura 6.7 mostra o diagrama de classes que abstraem os contentores, `+photos` e `+status`, para objectos XDI do tipo `+photo` e `+entry` que estendem a classe base para contentores de objectos XDI. Permite consultar e adicionar entradas na lista consoante o seu identificador ou serviço alojado.
- As Figuras 6.8, 6.9, 6.10 e 6.11 mostram os diagramas de classes que abstraem os objectos de dados XDI suportados pelo sistema.

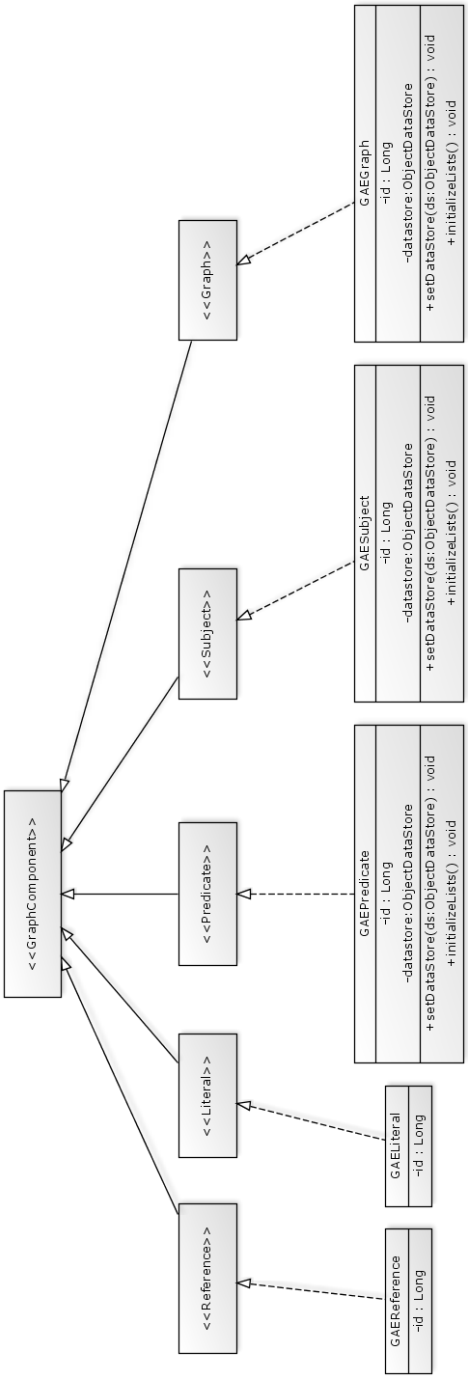

Figura 6.5: App Engine XDI core

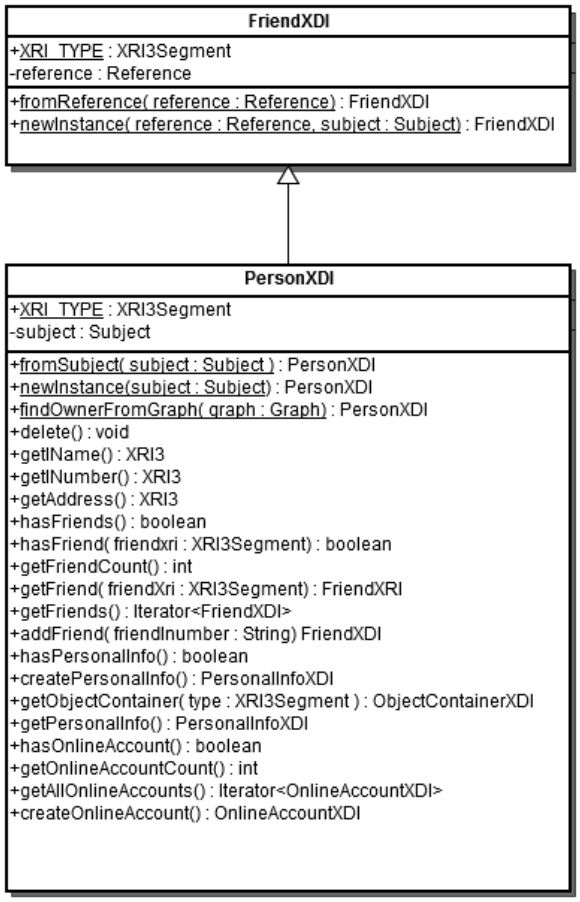

Figura 6.6: Diagrama de Classes, identidades.

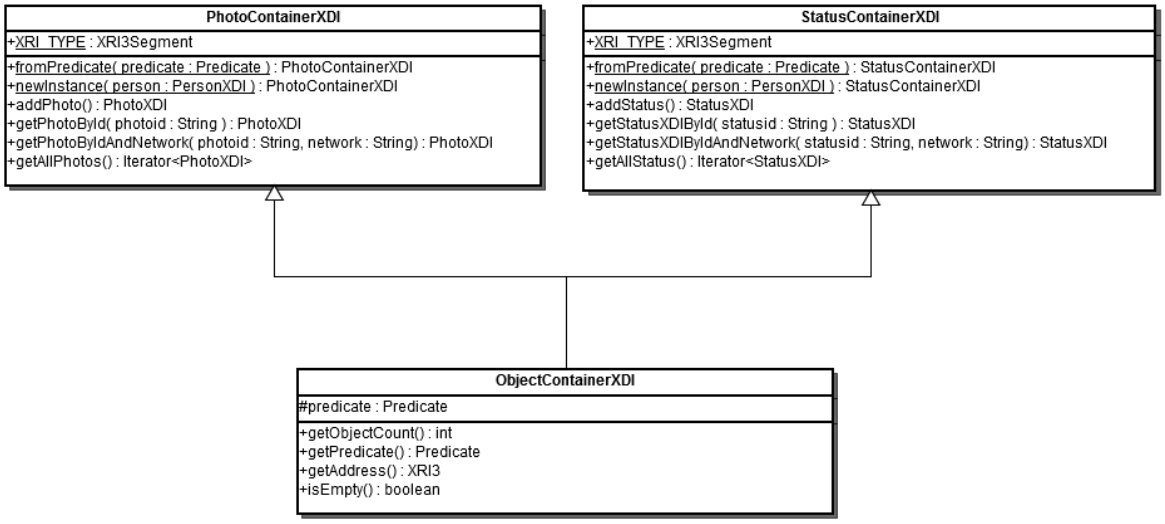

Figura 6.7: Diagrama de Classes, contentores de informação relativa a serviços.

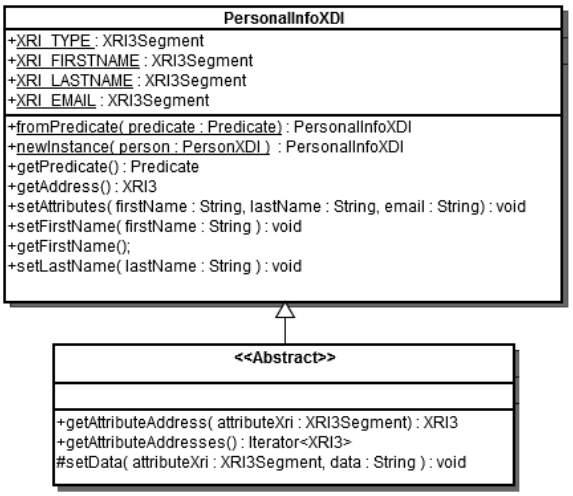

Figura 6.8: Diagrama de Classes, abstracção do tipo `+personalinfo`.

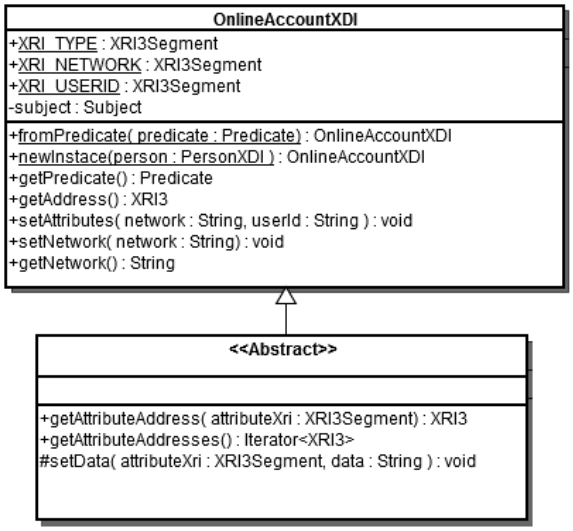

Figura 6.9: Diagrama de Classes, abstracção do tipo `+onlineaccount`.

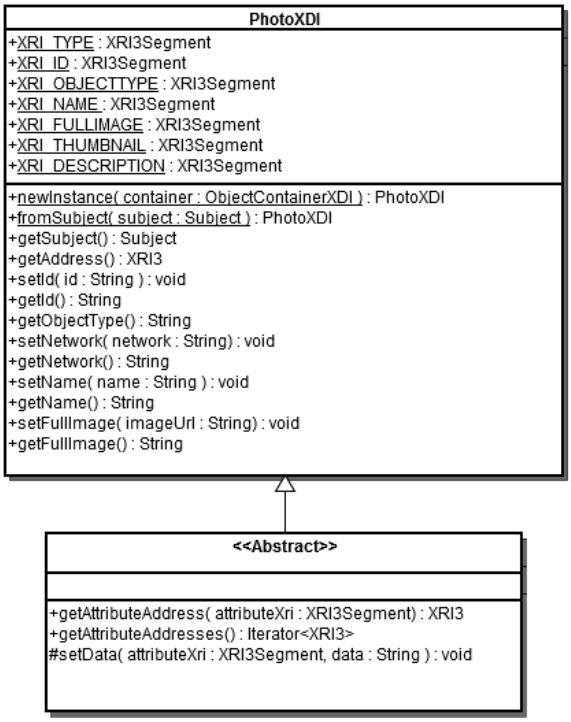

Figura 6.10: Diagrama de Classes, abstracção do tipo `+photo`.

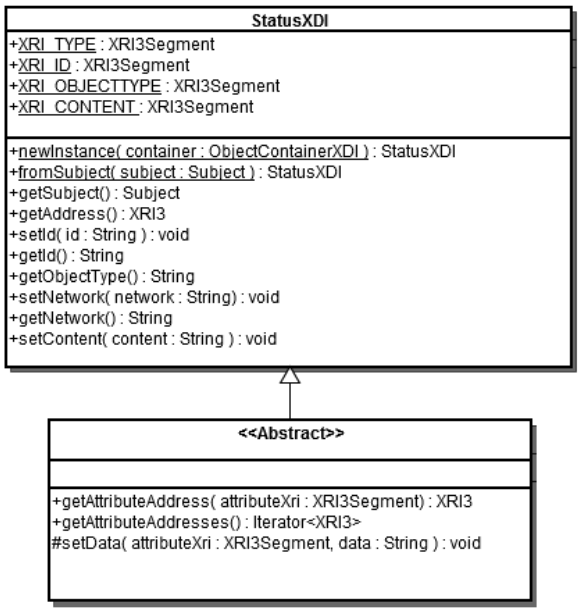

Figura 6.11: Diagrama de Classes, abstracção do tipo `+status`.

6.6.2 Módulo I/O

Sempre que o sistema necessita de contactar com o I-Broker, os pedidos são efectudados através de componenentes presentes neste módulo, a que corresponde o *package* `pt.ua.server.xdi`. A classe `IBrokerService`, ver Figura 6.12, é composta por um *resolver local* e um cliente XDI. Se o sistema necessita informações sobre determinado identificador XRI os pedidos são mediados através de uma instância desta classe, particularizando, a descoberta de *endpoints* XDI e detalhes sobre o certificado associado ao identificador.

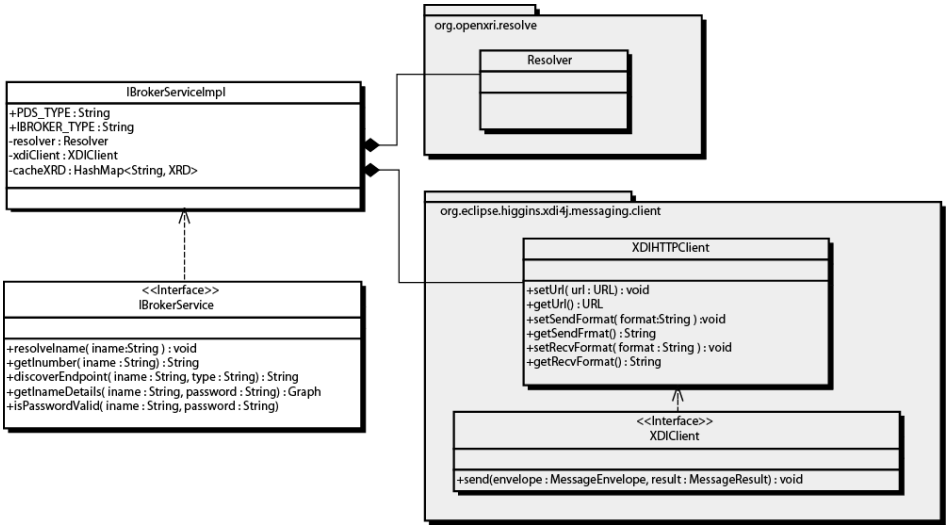

Figura 6.12: Implementação do interface `IBrokerService`.

6.6.3 Módulo de Serviços

A actividade de associar um serviço ao utilizador inicia-se sempre com o processo de autenticação com o provedor do serviço, ver Figura 6.13, para obter o *token* de acesso. Quando o utilizador adiciona um serviço na aplicação *web* é redireccionado para o *Servlet* correspondente que gere o processo de associação.
Para mapear a informação de cada serviço para o grafo XDI do utilizador no *package* `pt.ua.server.services` existe uma classe responsável, ver Figura 6.14, por devolver a instância adequada para mapear os dados conforme o serviço desejado. Instância essa que implementa o interface `DataMapper`. A Figura 6.15 demonstra o diagrama de sequência da associação de um serviço.
No final da operação o grafo apenas contém a descrição das várias contas associadas com a informação necessária para identificar os elementos no serviço, como demonstra o Exemplo 19. Por cada serviço associado é feita uma nova agregação com o predicado `+onlineaccount` e cada novo elemento, imagem no exemplo, adicionada no grafo interior do predicado que identifica o contentor de imagens.

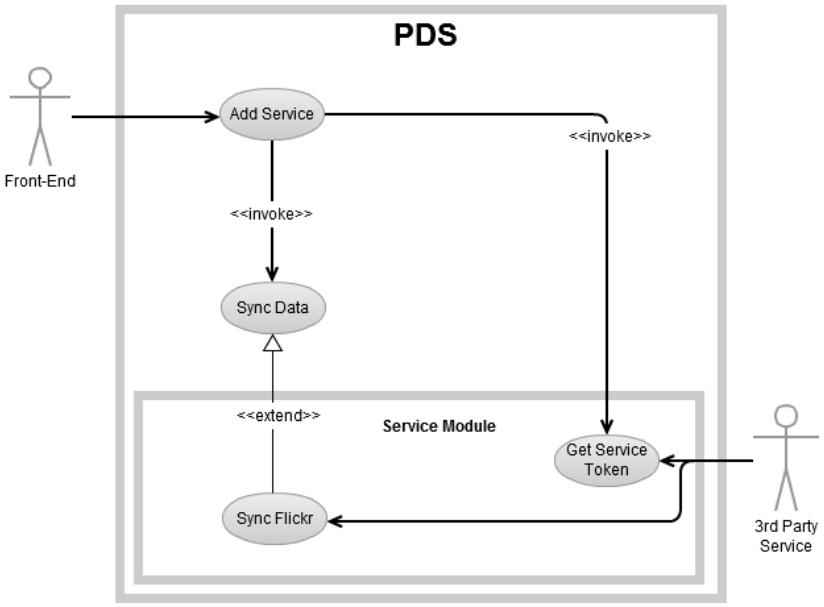

Figura 6.13: Detalhe do caso de utilização "Add Service".

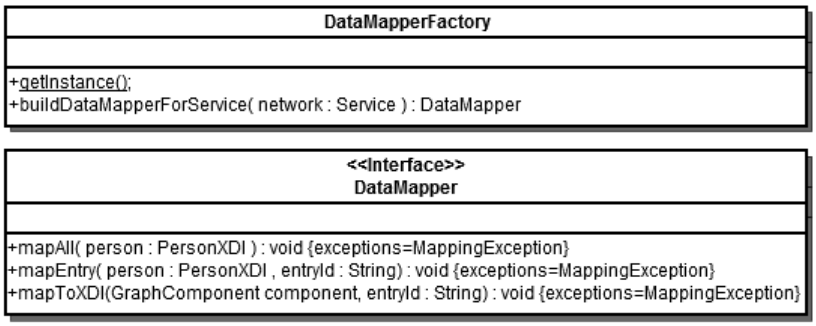

Figura 6.14: Diagrama de Classes responsáveis pelo mapeamento.

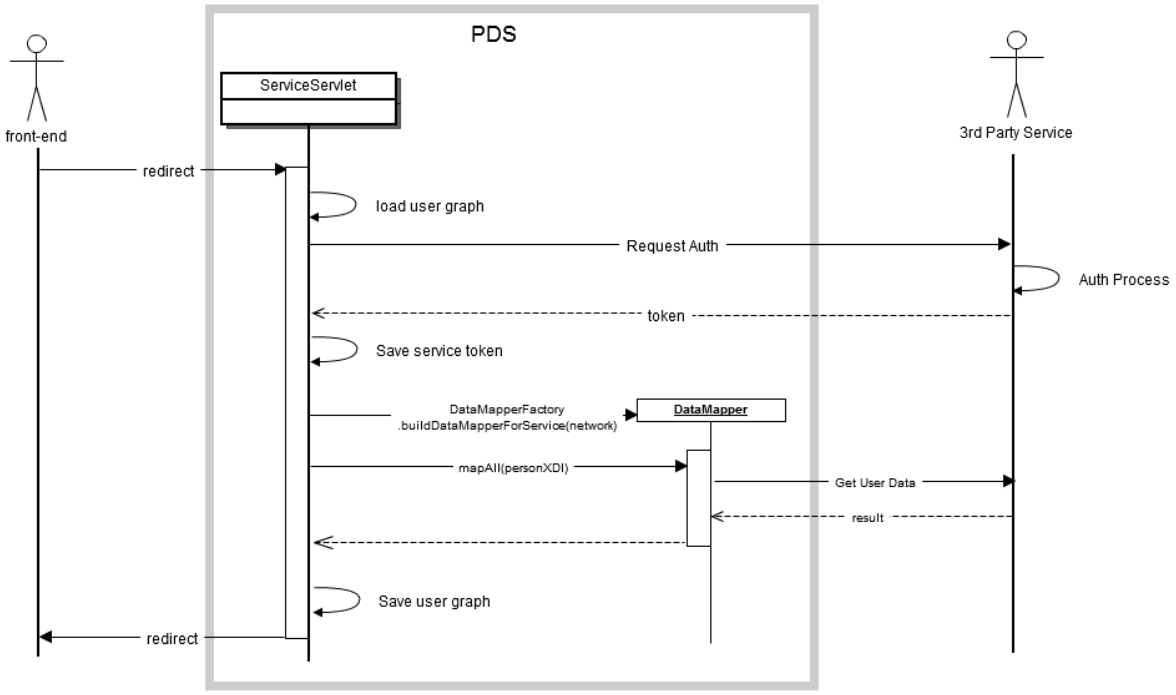

Figura 6.15: Diagrama de sequência da actividade "Add Service".

```
@userinumber
        +onlineaccount
                $!1
        +onlineaccount$!1
                /
                        $
                                +network
                                        "Flickr"
                                +userid
                                        "12345"
        +photos
                /
                        +photo$!1
                                +id
                                        "4864722466"
                                +network
                                        "Flickr"
                        +photo$!2
                                +id
                                        "4864105635"
                                +network
                                        "Flickr"
```

Exemplo 19: Grafo do utilizador resultante da associação de um serviço.

6.6.4 Módulo RPC

O módulo obedece à "*Command Pattern*" implementada no *framework* GWT-Platform como ilustrado na Figura 6.16. A aplicação cliente envia uma instância que extenda a classe `com.gwtplatform.dispatch.share.ActionImpl` identificando o tipo de acção e contendo os dados necessários para satisfazer o pedido. No servidor existe um mapeamento *action/handler* no módulo que implementa o método `configureHandlers()` da classe abstracta `com.gwtplatform.dispatch.server.guice.HandlerModule`. Cada actividade que consta no diagrama de casos de utilização da Figura 6.4, tendo o *front-end* envolvido como actor, corresponde à implementação de um *action handler* no servidor. Em seguida são apresentados os detalhes de cada actividade e o respectivo diagrama de sequência.

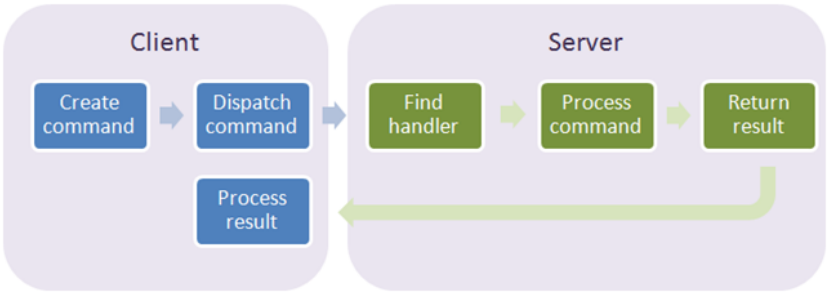

Figura 6.16: GWT-Plataform Command Pattern

6.6.4.1 Actividade "Create Profile"

Ocorre quando o utilizador efectua o *login* e não existe nenhum grafo associado. O *front-end* envia os dados inseridos pelos utilizador (informação pessoal, i-name, *password* e o *alias* desejado). Através do módulo de I/O é pedida a resolução do i-name e dos dados relativos ao certificado associado ao i-name. O sistema pede ao I-Broker[2] que adicione uma nova entrada no descritor do recurso(XRD), o que permite outros actores descobrirem o *endpoint* do utilizador no sistema com o seguinte formato: `https://{applicationdomain}/xdi/{alias}`. É criado o grafo inicial do utilizador e enviado o resultado para o *front-end*, ver Figura 6.17.

[2] À data de realização desta Dissertação a implementação dos I-Brokers disponíveis não suportam a adição de serviços dinâmicamente, no entanto, é expectável que venha a existir. Por enquanto o serviço tem de ser adicionado através do painel de administração do I-Broker pelo utilizador.

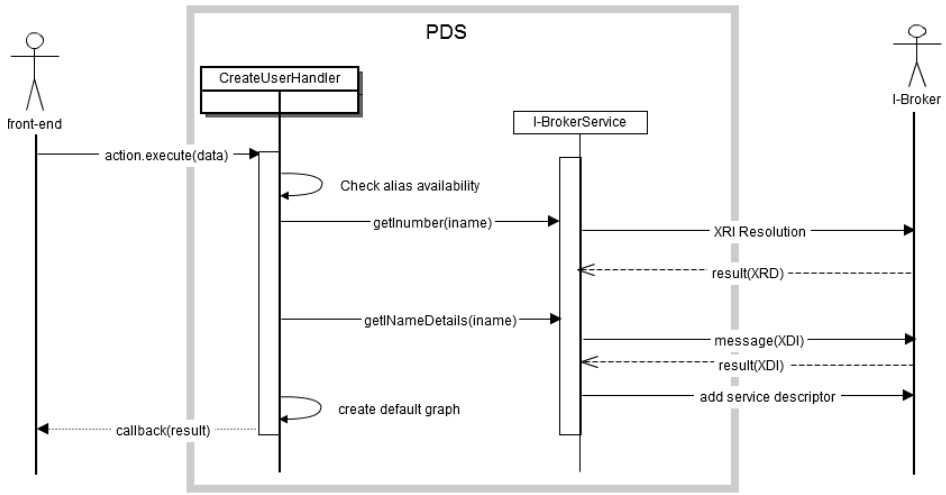

Figura 6.17: Diagrama de sequência da actividade "Create Profile"

O grafo do utilizador contém a informação pessoal assim como dois *link contracts*, o relativo ao utilizador que lhe permite alterar o seu grafo através de qualquer aplicação XDI e o *link contract* público. O estado actual da especificação XDI [17] não descreve um padrão para *link contracts* públicos. A proposta implementada nesta Dissertação sugere que o *link contract* associado ao i-name "@" seja considerado público. No final desta actividade o grafo do utilizador deve ter o padrão que se segue.

```
$<--XDI Context-->
    $is$a
        ($xdi$v$1)
        ($pds$v$1)
    $d$first
    $d$last
    $http$uri$1
    $https$uri$1
@!userinumber <--User Subject-->
    $certificate$x.509
    $key$private
    $key$public
    $is$a
        +person
    +personalinfo <--PersonalInfo Object-->
        /
            $<--PersonalInfo Context-->
                +firstname
                +lastname
                +email
    $has$a  <--Link Contract Aggregation-->
        $has
@!userinumber$has   <--Link Contract Root-->
```

```
    $is$a
        $has
    $has$a
        $1
        $2
@!userinumber$has$1 <--Owner Link Contract-->
    $is$a
        $has
    $has$a
        $a
    $get
        /
            $
            @!userinumber
    $add
        /
            $
            @!userinumber
    $del
        /
            @!userinumber
    $mod
        /
            $
            @!userinumber
@!userinumber$has$1$a
    $is$a
        $a
    $is$has
        @!userinumber
@!userinumber$has$2
    $is$a
        $has
    $has$a
        $a
    $get
        /
            $
            @!userinumber
                $is
                +personalinfo
                    /
                        $
                            +firstname
                            +lastname
@!userinumber$has$2$a
    $is$a
```

```
    $a
$is$has
    @
```

6.6.4.2 Actividade "Get Images"

Para promover a portabilidade dos dados, o grafo do utilizador guardado no sistema apenas contém a informação estritamente necessária para identificar o recurso no serviço associado. O diagrama de sequência da Figura 6.18 descreve a actividade de agregar a informação sobre os objectos alojados no(s) serviço(s) com a informação de acesso (*linkcontracts*) guardada no PDS.

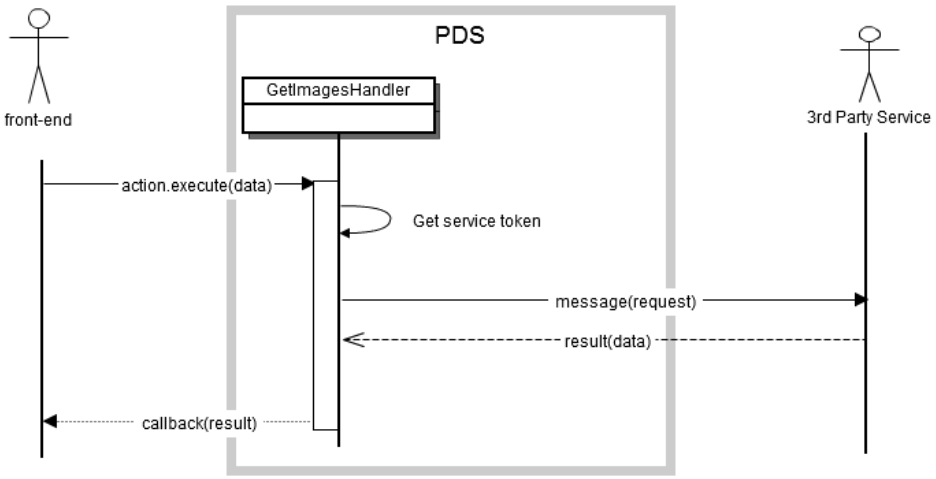

Figura 6.18: Diagrama de sequência da actividade "Get Images".

6.6.4.3 Actividade "Change Permissions"

Cada fonte (amigo) tem um *link contract* associado definindo as suas permissões de acesso. O *front-end* envia os atributos junto com a lista das várias partes que podem ter acesso. O sistema percorre os vários *link contracts* presentes no grafo do utilizador e adiciona/remove o acesso para a operação pretendida.

6.6.4.4 Actividade "Add Source"

O diagrama da Figura 6.19, mostra qual a estratégia para adiconar um(a) amigo/fonte e tentar ler alguma informação sobre este. Através da resolução XRI é possível descobrir se este contém um serviço PDS onde possa ser contactado. Caso exista, o sistema tenta ler a informação de contacto. Finalmente é associado um *link contract* ao i-number da fonte e adicionado como amigo do utilizador. De notar que a relação de amigo é assimétrica e não define nenhuma regra de acesso, apenas estabelece a relação de que X conhece Y.

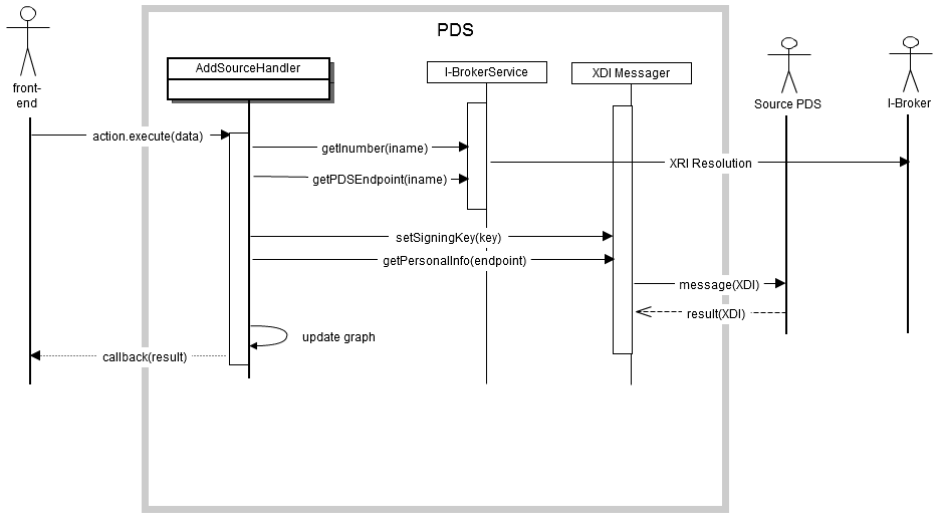

Figura 6.19: Diagrama de sequência da actividade "Add Source".

6.6.4.5 Actividade "Get Friend Images"

Através da resolução do i-number é descoberto o *endpoint* XDI correspondente ao PDS. A mensagem, assinada com a chave privada do utilizador, é enviada pelo sistema para outra instância PDS. O Exemplo 20 mostra a mensagem para a operação de leitura da lista de imagens. O diagrama de sequência é semelhante ao da Figura 6.19 excepto que o resultado não é gravado no grafo do utilizador mas sim serializado e enviado para o *front-end*.

```
@!senderinumber
        $sig
                "..."
        $get
                /
                        @!receiverinumber
                                +photos
```

Exemplo 20: Conteúdo de pedido de leitura para lista de imagens.

6.6.4.6 Actividade "Sync Data"

Para manter a consistência entre os dados presentes no serviço e no PDS do utilizador é necessário existirem mecanismos de sincronismo. Para tal o *package* `pt.ua.server.sync` contém implementações de *cronjobs* para cada serviço suportado que são chamados periódicamente pelo sistema. Ver diagrama da Figura 6.20.

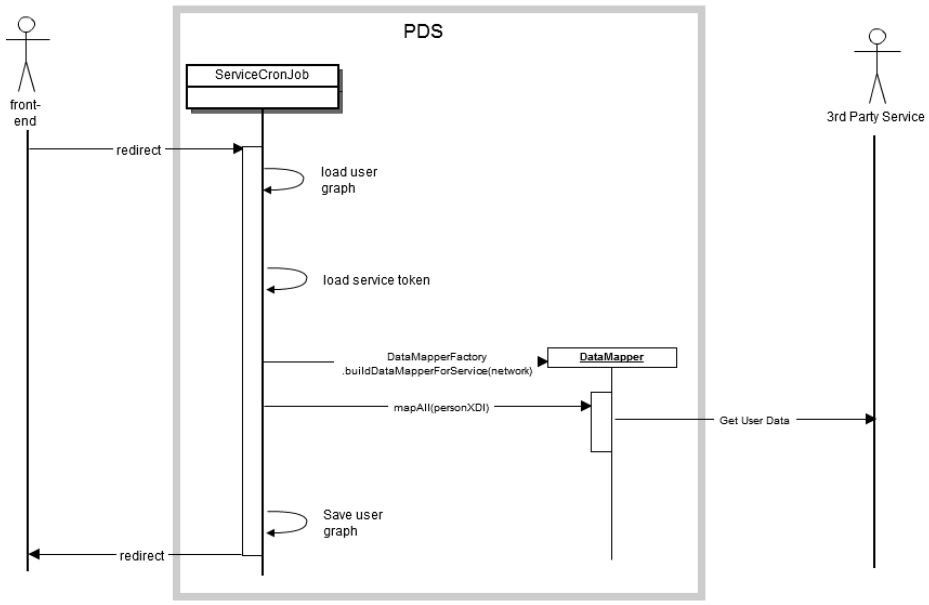

Figura 6.20: Diagrama de sequência da actividade "Sync Data"

6.6.5 Módulo *endpoint XDI*

Consiste num *Servlet* que mediante o parâmetro do *alias* definido pelo utilizador carrega o grafo correspondente e aplica as operações contidas na mensagem recebida. A implementação da biblioteca XDI4J contém a implementação do *Messaging Target* para grafos em memória 6.2 e interfaces para interceptores do envelope da mensagem, mensagem, endereços ou operações. Os interceptores realizam operações antes e depois da execução da mensagem no grafo, ver Figura 6.21.

Na implementação da prova de conceito existem concretizações de três tipos de interceptores.

6.6.5.1 Interceptor de assinatura

Para garantir a autenticidade da mensagem o sistema necessita de verificar a assinatura presente no envelope da mensagem, esse é o propósito deste interceptor. Para verificar a assinatura da mensagem é realizada a descoberta da chave pública associada ao identificador XRI (*sender*) que é obtida no processo de resolução do identificador. Este interceptor apenas implementa lógica para a fase anterior à execução da mensagem.

6.6.5.2 Interceptor de *link contracts*

As operações sobre o grafo do utilizador são mediadas pelo mecanismo de *link contracts*. O interceptor apenas contém lógica de verificação antes da execução da mensagem como é de esperar. A verificação ocorre para todos os endereços e operações presentes na mensagem recebida. Apenas quando todos os endereços de uma operação são negados é lançada uma

excepção, caso contrário o endereço que viola o *link contract* é retirado da mensagem antes da execução, sendo que, o primeiro *link contract* a ser verificado é o de domínio público que detém prioridade sobre os outros.

6.6.5.3 Interceptor de operações

Como já referido anteriormente, no grafo do utilizador apenas persiste informação estritamente necessária para identificar objectos alojados em serviços associados. Como consequência apenas quando for efectuado um pedido que necessite da descrição completa dos dados o serviço é contactado e contruída a resposta a devolver. Existe um interceptor para cada tipo de serviço suportado porque cada serviço implementa uma API específica. Estes interceptores apenas realizam operações após a execução da chamada de modo a completar a resposta.

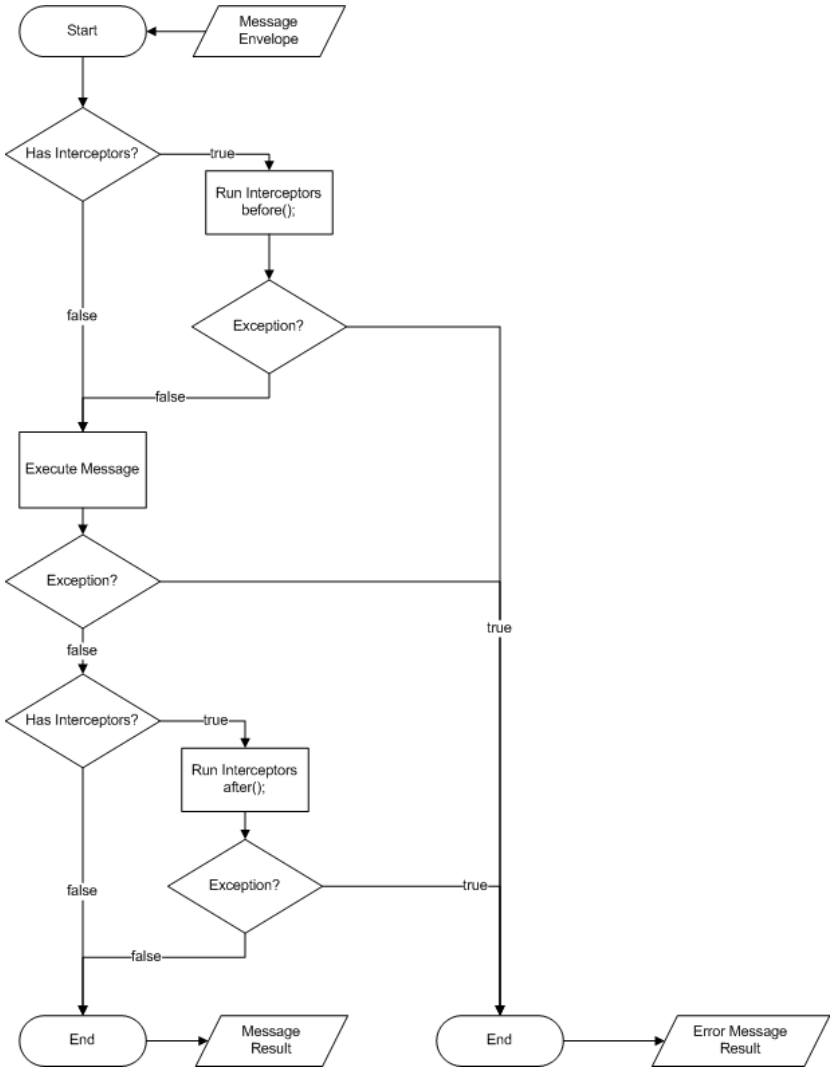

Figura 6.21: Diagrama de fluxo do *messaging target*.

6.7 Sumário

Este capítulo abordou a implementação realizada como prova de conceito para esta Dissertação abordando o cenário concreto de imagens, no entanto os mecanismos de comunicação mantêm-se apenas mudando o padrão do dicionário para cada novo tipo. A prova de conceito suporta dados, imagens, contidos no Flickr[3] ou Facebook[4] podendo ser estendidos para outros provedores com a implementação do *Servlet* e mapeador específicos. Foram definidos dicionários XDI para estruturas que representam um utilizador com serviços que suportem imagens e mensagens de *status*, assim como um padrão XDI que permite definir partes do grafo como públicas para que qualquer entidade identificada por um endereço XRI possa aceder. Através da solução proposta é possível trocar informação entre indivíduos independentemente da localização do serviço.
A implementação de componentes nativos XDI adiciona o suporte da biblioteca XDI4J para o Google App Engine. Infelizmente a plataforma é penalizadora para aplicações de baixo tráfego. A plataforma gere os seus recursos carregando ou desligando instâncias da aplicação consoante necessidade e a cada instância uma nova *Java Virtual Machine* (JVM) é inicializada para satisfazer o pedido. Uma aplicação com baixo tráfego passado poucos minutos pode não ter nenhuma instância activa e quando é recebido um pedido recomeça todo o processo de inicialização aumentando consideravelmente a latência e eventualmente o mau funcionamento das aplicações.

[3]Flickr, `http://www.flickr.com`
[4]Facebook, `http://www.facebook.com`

Capítulo 7

Conclusões

Este capítulo aborda as conclusões retiradas ao longo do trabalho desenvolvido nesta Dissertação e sugere abordagens para trabalho futuro. Em seguida são listadas as contribuições e resultados obtidos.

- A secção 6.6.5.2 propõe uma adição ao padrão definido na especifícação [17] para o mecanismo de *link contracts*. A proposta permite definir *link contracts* que representam operações de domínio público.
- A secção 6.5 propõe um conjunto de dicionários XDI que possibilite a associação de contas de serviços *web* sociais ao utilizador. Esta associação permite que outros utilizadores efectuem a descoberta, com autorização prévia, de identificadores específicos de cada serviço. Na mesma secção são propostos dicionários XDI para representar fotografias e mensagens do utilizador quer estejam ou não localizadas em serviços externos. A prova de conceito implementa estes dicionários.
- A performance obtida na implementação da prova de conceito ficou aquém do desejado. As causas para a fraca performance devem-se principalmente a dois factores: (i) a forma como o App Engine penaliza aplicações de baixo tráfego (ver secção 6.7), (ii) o elevado número de dependências da biblioteca XDI4J que obriga a uma carga adicional na inicialização.
- A abordagem de uma arquitectura centrada no utilizador foi demonstrada através da implementação da prova de conceito. As suas funcionalidades mostram claramente que é possível criar uma rede onde o utilizador tenha a liberdade de escolher o seu provedor, mantendo um identificador único e sem por em causa a interoperabilidade com instâncias de outros utilizadores.

7.1 Conclusões finais

Um novo ecossistema para a gestão de informação pessoal está a emergir, não só em novas formas de relacionamento *online* com os outros, através de redes sociais, mas também no relacionamento com instituições e organizações. Os designados *Personal Data Services* (PDS) são nucleares a um ecossistema orientado para o utilizador atribuindo melhor controlo sobre o acesso à sua informação. Não menos importante que um controlo granular da sua informação é a liberdade de escolha do prestador de serviço segundo critérios valorizados pelo utilizador.

Para que isso seja uma realidade impõe-se que a representação da informação seja portável e independente de plataformas, localização ou provedores.
O trabalho realizado nesta Dissertação centrou-se no estudo de algumas tecnologias que tentam solucionar o problema da identificação, resolução e representação de recursos que suportem esse ecosistema com principal foco no *framework* XRI/XDI. A prova de conceito proposta introduz alguns desses conceitos aproveitando a informação do utilizador já disponibilizada em redes sociais, criando uma agregação semântica de conteúdos. Como apenas é guardada a informação relevante para identificação do conteúdo, é favorecida a portabilidade dos dados para outras instâncias de modo simples apenas com a condição de existir um consenso nos dicionários e padrões XDI a utilizar, já que a portabilidade das autorizações de acesso está assegurada pelo mecanismo de *link contracts* presentes no XDI.
É expectável que soluções mistas, como a proposta na prova de conceito, sejam o primeiro passo para a criação de uma federação de redes sociais integrando serviços actuais que ainda funcionam como silos de informação. O utilizador gere toda a informação de forma granular numa camada superior. Os identificadores XRI, abstractos, garantem a escalabilidade de serviços associados ao identificador e a flexibilidade necessária para oferecer liberdade de escolha de provedores, deixando em aberto a possibilidade de gerir a sua própria instância PDS do mesmo modo que muitos indivíduos optam por alojar o seu *blog*. Para que isso seja uma realidade têm obrigatóriamente de surgir soluções livres, com baixo custo de manutenção e facilidade de instalação pelo utilizador final.
Apesar da solução abordada na implementação se focar na federação de redes sociais, os mecanismos de mapeamento e mediação de acesso podem ser aplicados noutros contextos como *Vendor Relationship Management* (VRM) que possibilita ao utilizador gerir o acesso à sua informação por parte de serviços ou organizações sobre os seu termos.
Para o aparecimento deste tipo de soluções, relativo ao *framework XRI/XDI*, é necessário solidificar alguns pontos. Existem poucos I-Brokers a operar e carece o amadurecimento de um mercado como o existente para a compra de domínios DNS. Outro ponto relacionado com I-Brokers é a necessidade de suporte para a manipulação XDI da informação relacionada ao identificador, isto porque, generalizando, a informação contida no I-Broker faz parte, conceptualmente, do grafo XDI do indivíduo.
O desenvolvimento da especificação XDI ainda está em desenvolvimento e sem uma especificação estável torna-se difícil gerar tracção para a sua adopção. A consequência é a fraca existência de ferramentas e a optimização necessária para melhorar a performance de aplicações.

7.2 Trabalho futuro

Como trabalho futuro, existe a necessidade de analisar qual o papel que outras tecnologias podem ter na implementação de serviços de gestão de informação pessoal, como por exemplo o papel que o OAuth pode adicionar ao mecanismo de *link contracts*. Existem actualmente vários esforços para a criação de redes sociais federadas. Um exemplo disso é o projecto OpenSocialWeb [51]. Tem como visão a interoperabilidade entre redes sociais. Agrega os utilizadores em *hubs* e a comunicação é efectuada via XMPP numa arquitectura PubSub (*publish/subscribe*). É um conceito particularmente interessante para operadores, sendo eles um *hub* que podem providenciar um serviço *web* social aos seus clientes de forma semelhante ao que já disponibilizam com o serviço de email. Em relação à integração com serviços já existentes, como os abordados na prova de conceito, delegam a responsabilidade de integração

para os serviços que queiram fazer parte da federação de redes. Tendo o XDI como objecto de desenho a extensibilidade, a sua integração e mapeamento com outras tecnologias pode ser uma mais valia. Nesta Dissertação foi abordado como o conceito de PDS pode facilitar a gestão e partilha de informação entre indivíduos. Sendo apenas uma parte da visão existente, deve ser feita a sua validação para o caso de relações entre organizações/serviços e o indivíduo assim como a integração com ferramentas tradicionais de *Customer Relationship Management* (CRM).

Referências

[1] ICANN, "Peak capacities.." http://www.icann.org/en/tlds/org/applications/uia/C17_10.html, 2010.

[2] http://www.dnssec.my/testbed-instruction.php.

[3] "Identity fragmentation." http://blog.myid.is/2007/11/identity-fragmentation/, 2007.

[4] "Opengroup." http://www.opengroup.org/security/sso/sso_intro.htm.

[5] D. McAlpin and D. Reed, *Extensible Resource Identifier (XRI) Syntax 2.0. Committe Specification 1.* OASIS Extensible Resource Identifier (XRI) TC, 2005.

[6] D. McAlpin and D. Reed, *An Introduction to XRIs. Working Draft 04.* OASIS Extensible Resource Identifier (XRI) TC, 2005.

[7] T. Berners-Lee, R. Fielding, and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax." RFC 3986 (Standard), Jan. 2005.

[8] G. Wachob and D. Reed, *Extensible Resource Identifier (XRI) Resolution Version 2.0. Committee Draft 03.* OASIS Extensible Resource Identifier (XRI) TC, 2008.

[9] D. Reed, *A New View of the XDI Graph Model.* 2010.

[10] G. Klyne and J. J. Carroll, "Resource description framework (rdf). concepts and abstract syntax." http://www.w3.org/TR/rdf-concepts/, 2004.

[11] P. Davis and D. Reed, *Extensible Resource Identifier (XRI) Version 3.0. Working Draft 3.* OASIS Extensible Resource Identifier (XRI) TC, 2010.

[12] P. Davis and D. Reed, *Extensible Resource Descriptor (XRD). Version 1.0. Committe Specification 01.* OASIS Extensible Resource Identifier (XRI) TC, 2010.

[13] A. Millar, *The Case for Personal Information Empowerment: The rise of the personal data store.* Mydex, 2010.

[14] E. Hammer-Lahav, "The OAuth 1.0 Protocol." RFC 5849 (Informational), Apr. 2010.

[15] L. M. Campbell and S. MacNeill, *The Semantic Web, Linked and Open Data.* JISC CETIS, 2010.

[16] F. Manola and E. Miller, "Rdf primer. w3c recommendation." http://www.w3.org/TR/rdf-primer/, 2004.

[17] M. Sabadello and D. Reed, *The XDI RDF Model*. OASIS Extensible Resource Identifier (XRI) TC, 2010.

[18] K. Harrenstien, M. Stahl, and E. Feinler, "DoD Internet host table specification." RFC 952, Oct. 1985.

[19] P. Mockapetris, "Domain names - implementation and specification." RFC 1035 (Standard), Nov. 1987. Updated by RFCs 1101, 1183, 1348, 1876, 1982, 1995, 1996, 2065, 2136, 2181, 2137, 2308, 2535, 2845, 3425, 3658, 4033, 4034, 4035, 4343, 5936, 5966.

[20] R. Aitchison, *Pro DNS and BIND*. Apress, 2005.

[21] S. Thomson and C. Huitema, "DNS Extensions to support IP version 6." RFC 1886 (Proposed Standard), Dec. 1995. Obsoleted by RFC 3596, updated by RFCs 2874, 3152.

[22] M. Crawford and C. Huitema, "DNS Extensions to Support IPv6 Address Aggregation and Renumbering." RFC 2874 (Experimental), July 2000. Updated by RFCs 3152, 3226, 3363, 3364.

[23] C. Schuba, "Addressing weaknesses in the Domain Name System Protocol," Master's thesis, Purdue University, West Lafayette, Indianapolis, USA, 1993.

[24] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, "DNS Security Introduction and Requirements." RFC 4033 (Proposed Standard), Mar. 2005.

[25] P. Vixie, "Extension Mechanisms for DNS (EDNS0)." RFC 2671 (Proposed Standard), Aug. 1999.

[26] D. Conrad, "Indicating Resolver Support of DNSSEC." RFC 3225 (Proposed Standard), Dec. 2001. Updated by RFCs 4033, 4034, 4035.

[27] B. Laurie, G. Sisson, R. Arends, and D. Blacka, "DNS Security (DNSSEC) Hashed Authenticated Denial of Existence." RFC 5155 (Proposed Standard), Mar. 2008.

[28] W. Jackson, "Growth in number of unmanaged dns servers raises security risk.." `http://gcn.com/Articles/2009/11/10/DNS-survey-DNSSEC.aspx?Page=1`, 2009.

[29] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext Transfer Protocol – HTTP/1.1." RFC 2616 (Draft Standard), June 1999. Updated by RFCs 2817, 5785.

[30] B. Fitzpatrick, D. Recordon, D. Hardt, and J. Hoyt, "Openid authentication 2.0 - final." `http://openid.net/specs/openid-authentication-2_0.html`, 2007.

[31] J. Millher, *Yadis Specification 1.0*. Yadis.org, 2006.

[32] D. Eastlake 3rd, J. Reagle, and D. Solo, "(Extensible Markup Language) XML-Signature Syntax and Processing." RFC 3275 (Draft Standard), Mar. 2002.

[33] J. Hoyt, J. Daugherty, and D. Recordon, "Openid simple registration extension 1.0." `http://openid.net/specs/openid-simple-registration-extension-1_0.html`, 2006.

[34] D. Hardt, J. Bufu, and J. Hoyt, "Openid attribute exchange 1.0 - final." http://openid.net/specs/openid-attribute-exchange-1_0.html, 2007.

[35] M. Duerst and M. Suignard, "Internationalized Resource Identifiers (IRIs)." RFC 3987 (Proposed Standard), Jan. 2005.

[36] R. Moats, "URN Syntax." RFC 2141 (Proposed Standard), May 1997.

[37] M. Mealling and R. Denenberg, "Report from the Joint W3C/IETF URI Planning Interest Group: Uniform Resource Identifiers (URIs), URLs, and Uniform Resource Names (URNs): Clarifications and Recommendations." RFC 3305 (Informational), Aug. 2002.

[38] L. Daigle, "The curious history of Uniform Resource Names." IETF Journal. Volume 6, Issue 1 (June 2010), 2010.

[39] BetaNews, "Oasis, amd push open identity management standard." http://www.betanews.com/article/OASIS-AMD-Push-Open-Identity-Management-Standard/1075980052, 2004.

[40] K. Sollins and L. Masinter, "Functional Requirements for Uniform Resource Names." RFC 1737 (Informational), Dec. 1994.

[41] M. Nottingham, "Web Linking." IETF Draft, https://tools.ietf.org/html/draft-nottingham-http-link-header-10, May 2010.

[42] N. Freed and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types." RFC 2046 (Draft Standard), Nov. 1996. Updated by RFCs 2646, 3798, 5147.

[43] D. Beckett, "Rdf/xml syntax specification. w3c recommendation." http://www.w3.org/TR/rdf-syntax-grammar/, 2004.

[44] P. Hayes, "Rdf semantics. w3c recommendation." http://www.w3.org/TR/rdf-mt/, 2004.

[45] D. Brickley and R. Guha, "Rdf vocabulary description language 1.0: Rdf schema. w3c recommendation." http://www.w3.org/TR/rdf-schema/, 2004.

[46] B. Adida and M. Birbeck, "Rdfa primer, bridging the human and data webs. w3c working group note." http://www.w3.org/TR/xhtml-rdfa-primer/, 2008.

[47] "Activity streams." http://activitystrea.ms/.

[48] H. Project, "Higgins. open source identity framework." http://www.eclipse.org/higgins/.

[49] H. Project, "Xdi4j 1.1." http://wiki.eclipse.org/XDI4j.

[50] O. Foundation, "Openxri project." http://www.openxri.org/.

[51] "Onesocialweb." http://onesocialweb.org.