Arquitectura Semântica para a Integração de Sistemas no Domínio do Turismo

Miguel Bruno Lemos de Gouveia

(Licenciado)

Tese Submetida à Universidade da Madeira para a

Obtenção do Grau de Mestre em Engenharia Informática

Funchal – Portugal

Dezembro 2007

UNIÃO EUROPEIA
FUNDO SOCIAL EUROPEU

PROGRAMA OPERACIONAL
PLURIFUNDOS DA REGIÃO
AUTÓNOMA DA MADEIRA

CENTRO DE CIÊNCIA E TECNOLOGIA DA MADEIRA

Orientador:

Professor Doutor António Jorge Silva Cardoso

Professor Auxiliar do Departamento de Matemática e Engenharia da Universidade da Madeira

UNIÃO EUROPEIA
FUNDO SOCIAL EUROPEU

PROGRAMA OPERACIONAL
PLURIFUNDOS DA REGIÃO
AUTÓNOMA DA MADEIRA

CITMA
CENTRO DE CIÊNCIA E TECNOLOGIA DA MADEIRA

ABSTRACT

The tourism industry is characterized by its heterogeneity and by a high volume of online transactions. Small tourist operators develop their own booking systems more and more to avoid paying commissions to the big tourist information system owners. Due to this fact, many new sources of tourist information are appearing in the Internet and the escalation of these new sources makes the planning of holidays more complex to the tourists. The development of a system that enables the integration of different sources becomes an urgent necessity when it comes to help tourists plan their holidays and at the same time gives means to the tourist operators to develop new marketing strategies. One of them is most certainly the development of Dynamic Packaging. Dynamic Packaging allows the tourist or the agent to freely choose the products that will compose a holiday package. At the same time the tourist operators will be able to set business rules that will be dynamically applied when the packages are being created.

SEED architecture describes the implementation of an integration system and it also intents to be a base to develop a system that will support Dynamic Packaging. The integration of different sources of information can be made by using technologies related to the Semantic Web. The implementation of Dynamic Packaging is supported by reasoning engines that allow the definition and interpretation of semantic rules.

KEYWORDS

Systems Integration, Semantic Web, Semantic Rules, Dynamic Packaging, Tourist Products, Ontology, Inference.

RESUMO

A indústria do turismo caracteriza-se pela sua heterogeneidade e pelo grande volume de transacções realizadas on-line. Cada vez mais os pequenos operadores turísticos optam por desenvolver os seus pequenos sistemas de reservas, para não terem de estar submetidos ao pagamento de comissões às entidades que gerem os grandes sistemas de informação turística. Devido a este facto, têm surgido um grande número de novas fontes de informação turística na Internet. A proliferação de informação turística torna complexo o planeamento das férias por parte do turista. A implementação de sistemas de integração de informação turística torna-se uma necessidade urgente. Ao mesmo tempo que ajudam o turista no planeamento das férias, também permitem aos operadores implementarem novas estratégias de marketing. Uma destas novas estratégias de marketing passa pela implementação do conceito de "Dynamic Packaging". O "Dynamic Packaging" permite ao turista, ou ao agente turístico, a construção de pacotes que incluem produtos turísticos escolhidos por este sem qualquer limitação. Aos operadores turísticos, permite a criação de regras de negócio sobre a constituição de um pacote. As regras são depois aplicadas dinamicamente à medida que os pacotes são definidos.

A arquitectura SEED define a implementação de um sistema de integração de informação turística. Pretende também disponibilizar a base para a implementação de sistemas que suportem o "Dynamic Packaging". A integração da informação é realizada através da utilização das tecnologias associadas à Web Semântica. A implementação do "Dynamic Packaging" é suportada pela utilização de motores de inferência que permitem a definição e interpretação de regras semânticas.

PALAVRAS CHAVE

Integração de Sistemas, Web Semântica, Regras Semânticas, Dynamic Packaging, Produtos Turísticos, Ontologia, Inferência.

AGRADECIMENTOS

Quero agradecer ao meu orientador, professor Jorge Cardoso, pelo tempo despendido e pela ajuda no desenvolvimento do projecto. Quero também agradecer a toda a equipa que esteve envolvida no projecto SEED, pelas reuniões realizadas onde foram discutidos assuntos relacionados com a arquitectura SEED. Finalmente quero agradecer à minha mulher e ao meu pai por tudo o apoio que me deram durante a realização desta tese de mestrado.

Dedico este trabalho ao meu filho João Francisco.

ÍNDICE

Lista de Figuras

ACRÓNIMOS

API - Application Programming Interface

ASCII – American Standard Code for Information Interchange

CERN – Conseil Européen pour la Recherche Nucléaire

COG - Corporate Ontology Grid

CRM - Customer Relationship Management

CRS – Computerized Reservation System

DAML - DARPA Agent Markup Language

DIG - DL Implementation Group

DMS – Destination Management Systems

DSSs – Decision support systems

DTD - Document Type Definition

EBCDIC – Extended Binary Coded Decimal Interchange Code

EDI - Electronic Data Interchange

FTP - File Transfer Protocol

GDS – Global Distribution System

GUI - Graphical User Interface

HDS – Hotel Distribution System

HL7 - Health Level Seven

HTML - HyperText Markup Language

HTTP - HyperText Transfer Protocol

ISBN - International Standard Book Number

METU - Middle East Technical University

nRQL - New RacerPro Query language

OTA - Open Travel Alliance

OIL - Ontology Inference Layer

OWL - Web Ontology Language

OWL QL - OWL Query Language

OWL-S – Semantic Markup for Web Services

OWLmt – OWL Ontology Mapping Tool

RacerPro - Renamed ABox and Concept Expression Reasoner Professional

RDF – Resource Description Framework

RSS - Really Simple Syndication

S2SQL – Semantic To Structured Query Language

SEED - SEmantic E-tourism Dynamic packaging

SIM - Semantic Information Management

SMS - Short Message Service

SOAP - Simple Object Access Protocol

SQL – Structured Query Language

SWRL - Semantic Web Rule Language

TCP - Transmission Control Protocol

URI - Universal Resource Identifier

URL - Uniform Resource Locator

URN - Uniform Resource Name

W3C - World Wide Web Consortium

WSDL – Web Services Description Language

WYSIWYG - What You See Is What You Get

WWW - World Wide Web

XML - Extensible Markup Language

XSD - XML Schema Definition

XSLT - eXtensible Stylesheet Language Transformations

1 INTRODUÇÃO

1.1 MOTIVAÇÃO

A Internet tornou-se num gigante repositório onde podemos obter informação sobre quase todos os assuntos. As empresas aproveitam este meio de informação para disponibilizarem informação sobre os seus produtos e serviços. Hoje em dia é quase obrigatório a promoção e venda de produtos e serviços na Internet. Estes factos levaram a um grande aumento do volume de informação sobre produtos e serviços. Esta massificação de informação provocou um aumento na exigência dos clientes face ao que lhes é oferecido e na forma como é feita essa oferta. Cada vez mais os clientes pretendem obter a informação sobre os produtos e/ou serviços o mais completo e rápido possível.

Esta tarefa torna-se particularmente complexa quando se está a falar sobre turismo. Um turista geralmente pretende obter informação sobre um grande leque de produtos e serviços. Quando o turista pretende programar as suas férias pode necessitar obter informações sobre acomodação, transporte, eventos, meteorologia, monumentos ou outros. Actualmente é necessário aceder a vários sistemas para obter toda esta informação. A dispersão de informação torna difícil a escolha do turista pois não pode obter a informação desejada agregada numa só pesquisa. Existem alguns sistemas que tentam agregar diferentes produtos e serviços turísticos, no entanto, nunca conseguem disponibilizar toda a oferta turística existente numa determinada região. Existem muitas pequenas empresas que optam por não introduzir os seus produtos e/ou serviços nestes sistemas, ou porque não pretendem pagar comissão sobre a venda ou porque não têm recursos para implementarem uma integração com sistemas deste tipo. A disponibilização de informação nestes sistemas implica sempre um investimento de recursos na inserção e na actualização da informação no novo sistema.

Para os operadores turísticos também surgiram novos desafios com a massificação da oferta turística na Internet. A crescente exigência dos clientes e a concorrência cada vez mais apertada, exigem que se encontre novas formas de promoção dos produtos. Uma das formas mais comuns de promoção na área do turismo é a oferta de pacotes turísticos. Um pacote turístico consiste na oferta de um conjunto de produtos ou

serviços turísticos agregados num só, com um único valor monetário que esconde o valor monetário de cada um dos produtos ou serviços constituintes. Uma das vantagens da oferta de pacotes turísticos é a possibilidade de vender os produtos ou serviços menos procurados agregados aos mais procurados. Os pacotes são geralmente pré-definidos e podem ser constituídos por produtos e/ou serviços de diferentes operadores. Para o cliente, a grande vantagem está na possibilidade de usufruir de um conjunto de produtos e/ou serviços com um valor monetário mais em conta. O grande problema nestes pacotes está na pouca flexibilidade que eles oferecem. O cliente está sempre limitado aos produtos e/ou serviços pré-definidos para o pacote, sem poder substituí-los por outros do seu agrado. Para ultrapassar esta limitação surgiu o conceito de "Dynamic Packaging". O termo "Dynamic Packaging" pressupõe a possibilidade de criação de pacotes dinâmicos onde o cliente é que define quais os produtos e/ou serviços que constituem o pacote. O pacote é criado em tempo real e o valor monetário é calculado dinamicamente conforme os produtos escolhidos. Nos pacotes dinâmicos, tal como nos pacotes tradicionais, poderão ser aplicados descontos aos produtos e/ou serviços constituintes, tornando assim o valor total do pacote inferior à soma dos valores dos constituintes. A criação de um sistema que suporte "Dynamic Packaging" requer a integração de produtos e serviços de vários sistemas. Só assim consegue-se oferecer uma grande variedade de pacotes. Necessita também de um processo de definição de regras de negócio de forma flexível. As regras são necessárias para a definição dos pacotes dinâmicos, nomeadamente, para definir quais os possíveis constituintes de um pacote ou para definir o cálculo das tarifas associadas aos pacotes. As regras deverão ser flexíveis de forma a ser possível alterá-las conforme as tendências do mercado.

Todos estes problemas sugerem a definição de um sistema que os consiga resolver, um sistema que permita agregar produtos e serviços turísticos de forma a facilitar ao turista a procura de soluções para a programação das suas viagens. Tornar possível o suporte ao "Dynamic Packaging" é um dos desafios interessantes e que tornará o sistema muito mais apetecido, não só na perspectiva do turista mas também na perspectiva dos operadores turísticos.

1.2 Principais Objectivos

O trabalho realizado, no contexto da tese de mestrado, tem como principal objectivo a criação de uma arquitectura de integração de sistemas na área do turismo. Esta arquitectura deverá permitir a pesquisa de informação sobre produtos e serviços turísticos que poderão estar em diferentes sistemas e sobre diferentes formatos. A arquitectura deverá suportar a integração de informação estruturada sobre os formatos mais comuns para a troca de informação tais como XML, modelo relacional, HTML ou mensagens SOAP.

A configuração da integração com os sistemas externos deverá ser simples e não deverá consumir muito tempo. Desta forma não será necessário às empresas disponibilizarem muitos recursos para a integração dos seus sistemas com a arquitectura criada. Da mesma forma, a manutenção e actualização da informação integrada não deverá ser nem muito complexa nem deverá implicar muito tempo por parte da entidade integradora. A questão da actualização é muito importante, na área do turismo questões como o preço dos produtos e serviços ou então a disponibilidade destes podem ser alterados a qualquer momento. A arquitectura tem de conseguir uma actualização em tempo real para que a informação que chega ao cliente seja a mais correcta possível. Não é agradável para o cliente quando é informado que um determinado produto está disponível para consumo, e só depois de ser preenchida toda a informação necessária para a reserva deste, ser informado que a disponibilidade já não existe.

Um dos maiores problemas com que se defronta a integração de sistemas é a identificação de registos duplicados de uma mesma instância em sistemas diferentes. Este problema deverá ser tomado em conta. A constante duplicação de informação poderá complicar as pesquisas realizadas pelo cliente ou sistemas clientes. Sendo um ponto importante, a resolução deste nunca deverá prejudicar a simplicidade no processo de integração dos sistemas externos. É mais importante convencer as entidades que gerem os sistemas externos a associarem-se ao sistema do que a existência de instâncias duplicadas neste. É muito mais atractivo a existência de um

sistema que engloba um grande número de produtos e serviços turísticos do que um sistema com poucos produtos e serviços turísticos mas que nunca duplica a informação apresentada.

A arquitectura deverá disponibilizar funções de pesquisa de produtos e serviços de forma agregada, ou seja, funções que permitam obter um conjunto de produtos e serviços de sistemas diferentes, ou de áreas diferentes, através de apenas uma única pesquisa. Por exemplo, deverá permitir pesquisa de acomodação e informação meteorológica referente a uma determinada região através de um único filtro indicado pelo utilizador ou sistema cliente. O resultado desta pesquisa deverá ser o conjunto de todo o tipo de acomodação e de toda a informação meteorológica referente à região pretendida. A informação retornada deverá respeitar um modelo de dados pré-definido, independentemente do sistema externo que a contém. Desta forma o cliente ou o sistema cliente, poderá utilizar o sistema como se fosse único e não como um sistema composto pela integração de vários outros sistemas de informação. Para o cliente deverá ser completamente transparente a forma como os sistemas externos são integrados.

Sendo o conceito de "Dynamic Packaging" uma questão importante no futuro da indústria turística, a arquitectura deverá estar preparada para suportar este novo conceito. Devido à grande complexidade na implementação de um sistema com suporte ao "Dynamic Packaging", optamos por nos centrar apenas no problema da agregação da informação, que só por si é já um aspecto complexo. No entanto, a arquitectura deverá ser criada de forma a que no futuro seja fácil uma extensão para o suporte ao "Dynamic Packaging". Com este intuito, a arquitectura deverá permitir a definição de regras de negócio de forma flexível e em tempo de execução. Ou seja, deverá ser possível a definição de regras sobre o modelo de dados utilizado mesmo quando o sistema esteja operacional e em utilização. As regras definidas deverão permitir alterar os resultados das pesquisas dos clientes. Para o completo suporte ao conceito de "Dynamic Packaging" ficará a faltar, não só a possibilidade de criação de pacotes de forma dinâmica mas também a possibilidade de reservar os pacotes criados. A agregação da informação sobre produtos e serviços turísticos e a possibilidade de definição de regras sobre estes, são já os primeiros passos na criação de um sistema para o total suporte à criação de pacotes dinâmicos.

A tentativa de agregação de informação e o problema de integração de sistemas não é exclusivo da área do turismo. Este problema existe em muitos outros domínios e geralmente com as mesmas características que as existentes no domínio do turismo. A criação de uma arquitectura de integração geral, que não seja exclusivamente utilizada na área do turismo, traz muitas vantagens. Sem prejudicar o principal objectivo, integração de sistemas de informação na área do turismo, a arquitectura deverá manter-se independente do domínio. Assim, conseguimos alargar a utilidade da arquitectura permitindo a utilização noutros domínios que partilham dos mesmos problemas existentes no domínio do turismo.

1.3 Descrição da Estrutura do Documento

Este documento está divido em secções que por sua vez são divididas em subsecções. Começa com a introdução onde é definida a motivação para o trabalho realizado e identifica quais os principais objectivos a serem atingidos.

Na secção 2 começamos por apresentar o estado actual da indústria turística, identificando a importância das tecnologias de informação nos processos de negócio referentes ao turismo. Apresentamos também uma visão sobre o problema da integração de sistemas de informação e quais as vantagens na criação destes sistemas. Numa outra subsecção fazemos uma apresentação da tecnologia associada à Web Semântica mostrando qual a aplicabilidade desta nova tecnologia, nomeadamente na integração de sistemas. Terminamos a secção com a apresentação de outros trabalhos relacionados com o nosso.

Na secção 3 apresentamos a arquitectura para a integração de sistemas de informação no domínio do turismo. Aqui identificamos os módulos e sub-módulos que constituem a arquitectura. Para cada módulo e sub-módulo são descritas as responsabilidades que cada um irá tomar na arquitectura. A descrição da arquitectura é realizada sem ser definida qualquer pormenor de implementação, concentrando-se mais na análise do problema.

Depois de realizada a análise passamos à descrição de como foi implementada a arquitectura. Esta descrição é feita na secção 4. Descrevemos como foram implementados os módulos e sub-módulos identificados na secção anterior. São também identificadas as tecnologias utilizadas e é explicado o porquê da opção de utilização das referidas tecnologias. Em alguns casos são apresentadas soluções alternativas de implementação que poderiam ser escolhidas em detrimento das opções tomadas. Para cada uma das opções tomadas é explicado o porquê da sua escolha.

Descrita a arquitectura em pormenor, apresentamos um exemplo de utilização através da definição de um possível cenário. O cenário é apresentado na secção 5. Nesta secção,

são descritos todos os passos que deverão ser tomados para que a arquitectura seja correctamente configurada e utilizada para o cenário apresentado.

Por fim terminamos com a secção 6. Nesta secção apresentamos as principais limitações da arquitectura implementada e explicamos como estas limitações poderão ser resolvidas. Definimos também quais os possíveis melhoramentos e extensões que poderão ser aplicados à arquitectura. A terminar, apresentamos as conclusões finais sobre o trabalho realizado.

2 Estado da Arte

2.1 Sistemas de Informação no Domínio do turismo

A indústria do turismo nos nossos dias é caracterizada pela heterogeneidade do mercado e das fontes de informação e pelo grande volume de transacções on-line (Werthner and Klein, 2004). Existem muitos actores com interesses na área do turismo, existem as empresas que gerem os recursos como hotéis ou restaurantes, existem os operadores turísticos que reservam determinados recursos para a criação de produtos turísticos, as agências de viagem que tratam de vender os produtos turísticos dos operadores e por fim o turista que tem como objectivo adquirir produtos turísticos. Para além destes actores directos poderemos ter outras entidades com interesses no mercado turístico como por exemplo entidades governamentais interessadas em aumentar as entrada de receitas da região ou do país que governam. A grande quantidade de actores aumenta a complexidade do mercado do turismo. O cada vez maior acesso à informação, permitido pelas tecnologias de informação, veio alterar a tradicional segmentação do mercado implicando um reordenar mais adequado às exigências do turista. Os utilizadores da Internet constroem as suas viagens a partir de casa ou do escritório, 24 horas por dia, sem recorrer aos meios de apoio tradicionais como por exemplo os folhetos ou as agências de viagem. A Internet é já a principal fonte de informação para os turistas. Cerca de 95% dos utilizadores da Web utilizam a Internet para obterem informação relacionada com o turismo e 93% indicam que visitam Web Sites de turismo quando pensam em planear as suas férias (Fernandes, 2005).

Os operadores turísticos já se aperceberam da grande vantagem na utilização dos sistemas de informação e começaram já a disponibilizar os seus produtos na Internet para poderem ser reservados directamente pelos turistas ou por outros parceiros. A British Airways indica que 42% das suas vendas são provenientes do seu Web Site enquanto a Aer Lingus indica valores na ordem dos 70% (Wilson, 2005). As Agências de Viagens também tentam tirar partido das tecnologias de informação para poderem sobreviver a este novo modelo de turismo. É inevitável que o número de Agências de Viagens terá de diminuir, não devendo no entanto desaparecerem por completo. No futuro, prevê-se que estas utilizem aplicações para disponibilizar os seus serviços on-

line mas que também ofereçam serviços personalizados de ajuda ao turista em pontos de informação físicos que permitam um contacto directo entre turista e agência (Buhalis e Costa, 2006).

2.1.1 Tipos de Sistemas de Informação

As tecnologias de informação são utilizadas para disponibilizar informação acerca dos produtos e serviços turísticos, aos turistas e aos agentes de viagens. Informação sobre as características dos hotéis, preços e disponibilidades de viagens de avião, são alguns exemplos da informação disponibilizada (Cardoso e Lange, 2005). A informação disponibilizada por estes sistemas tem de estar sempre actualizada, principalmente no caso de sistemas que gerem disponibilidades dos produtos ou serviços turísticos. Por exemplo, no caso da venda de viagens de avião é essencial que a informação seja actualizada em tempo real para que o turista não reserve uma viagem num voo que afinal já não tinha disponibilidade. Existem cinco tipos de sistemas de informação utilizados no domínio do turismo:

Computerized Reservation System (CRS): Um CRS é um sistema de reservas pertencente a um determinado operador turístico. É muito utilizado pelas Agências de Viagens para procurarem os produtos ou serviços que o cliente pretende. Estes sistemas contêm informação sobre horários de voos, disponibilidades dos voos, taxas aplicadas e informação sobre serviços associados à reserva de uma viagem de avião. Alguns destes sistemas oferecem a possibilidade de criar reservas e de impressão de bilhetes. Estes sistemas começaram a aparecer nos anos 50 como sistemas internos às empresas. Com o desenvolvimento das tecnologias de informação passaram a estar disponíveis para as Agências de Viagens ou para outras organizações. Os sistemas CRS são principalmente populares, e muito utilizados, pelas companhias aéreas. Estima-se que 70% das reservas de voos são realizados através dos sistemas CRS.

Global Distribution System (GDS): É um sistema que liga vários sistemas CRS. Um sistema GDS integra informação turística sobre companhias aéreas, hotéis, aluguer de automóveis, cruzeiros e outros. É utilizado, quase em exclusivo, pelas Agências de Viagens. Estes sistemas apareceram pela primeira vez nos anos 60, criados pelas companhias aéreas. Antes as Agências de Viagens perdiam muito tempo a criar as reservas de produtos e serviços turísticos pois tinham de consultar e introduzir os dados do turista em vários sistemas CRS. Os sistemas GDS trouxeram uma maior

automatização do processo de criação de reservas, levando a uma expansão nas vendas de viagens aéreas (HotelOnline, 2002). A desvantagem de utilização destes sistemas é a comissão cobrada por cada reserva efectuada. Existem quatro principais sistemas GDS: Amadeus, Galileo, Sabre e Worldspan. Actualmente nos Estados Unidos, 90% dos bilhetes de avião são vendidos utilizando um destes quatro sistemas.

Hotel Distribution System (HDS): São sistemas que permitem reservar acomodação num determinado hotel. Estão directamente ligados a sistemas GDS para que uma reserva de acomodação seja realizada da mesma forma que uma reserva de uma viagem de avião. Desta forma, a criação de uma reserva de avião e alojamento será muito mais simples e rápida de realizar por parte das Agências de Viagens. Existem essencialmente dois tipos de sistemas HDS:

1) O sistema HDS está directamente ligado ao sistema de reservas próprio de um determinado hotel. O sistema depois é ligado a um sistema de GDS, oferecendo assim a possibilidade aos agentes de viagens da criação de reservas de acomodação e viagens.

2) O sistema HDS é oferecido por uma determinada companhia e depois é associado a um sistema GDS.

Destination Management Systems (DMS): São sistemas que oferecem informações sobre regiões turísticas. Permitem a promoção das regiões turísticas através da caracterização da oferta turística que essa região tem para oferecer. Oferecem ao turista informações variadas tais como atracções turísticas, festivais ou eventos culturais. Associada a esta informação estão sistemas que permitem a criação de reservas de acomodação. Através destes sistemas o turista pode, por exemplo, obter relatórios sobre o estado do tempo nos destinos turísticos. Também poderão estar associados às regiões turísticas vídeos que mostram as principais atracções turísticas ou então câmaras Web que mostram, em tempo real, imagens sobre a região turística. Os sistemas DMS têm como objectivo desenvolver produtos turísticos integrados, flexíveis e especializados. O desenvolvimento de um sistema deste tipo pode ser muito complexo pois agrega um grande número de actores com diferentes interesses. Podemos identificar sete grupos de actores (Buhalis e Spada 2000):

1) Clientes/visitantes (existentes e potenciais) que pretendem obter informações sobre as regiões turísticas;

2) Entidades que gerem os recursos turísticos e que tentam que os seus recursos sejam utilizados;

3) Operadores turísticos que pretendem vender os seus produtos;

4) Agentes de Viagens que têm o objectivo de vender produtos e serviços turísticos;

5) Entidades públicas que têm o intuito de promover as suas regiões;

6) Os investidores que pretendem que o sistema gere receitas;

7) As entidades que desenvolvem o sistema que desejam que o sistema responda com sucesso aos requisitos propostos.

O objectivo de cada um dos actores perante os sistemas pode provocar conflitos. O sucesso de um sistema DMS pode estar na boa gestão dos conflitos que poderão existir. Dois exemplos de sucesso são o sistema Tiscover (Áustria) e Gulliver (Irlanda).

Distribuição directa utilizando Web sites: O desenvolvimento da Internet e das tecnologias da informação está a revolucionar a indústria do turismo. Actualmente as pequenas empresas já conseguem desenvolver os seus Web sites e oferecer um acesso com uma boa largura de banda, situação que anteriormente só era possível através de grande investimento. Anteriormente, muitas companhias criavam os seus próprios sistemas de reservas mas só conseguiam disponibilizá-los através de outros sistemas existentes tais como sistemas GDS. Actualmente estas companhias disponibilizam o acesso aos sistemas de reservas através do seu próprio Web Site. Desta forma, deixam de utilizar os sistemas GDS. Esta solução é muito mais económica pois assim deixam de pagar as comissões sobre as reservas vendidas. Actualmente 95% das cadeias de hotéis têm um Web Site e destas 90% oferecem a possibilidade aos turistas da criação de reservas directamente aos hotéis (O'Connor, 2003).

2.1.2 Dynamic Packaging

Cada vez mais as pequenas empresas ligadas ao turismo tentam vender os seus produtos directamente através dos seus Web Sites, evitando a utilização dos sistemas GDS que implicam o pagamento de uma comissão por cada reserva realizada. Esta mudança leva a que cada vez mais produtos e serviços turísticos estejam disponíveis

ao cliente final para reservar através da Internet. Contudo, planear uma viagem utilizando a Web é uma tarefa que consome muito tempo e que em certos casos pode tornar-se complicada. A maior parte dos sites oferecem informações isoladas sobre voos, hotéis, aluguer de carro ou previsões meteorológicas, deixando para o cliente a complicada tarefa de combinar toda esta informação de forma a escolher o pacote de produtos turísticos que desejam para a sua viagem (Kabbaj, 2003). Para cada um dos produtos a reservar, o cliente terá de indicar os seus dados pessoais e os dados referentes ao pagamento.

Uma das formas mais tradicionais de adquirir uma viagem de férias é através dos pacotes turísticos pré-definidos. Estes pacotes são constituídos por diferentes produtos turísticos que poderão ser: viagem de avião, alojamento, aluguer de carro, etc. A definição dos pacotes é geralmente realizada com meses de antecedência e são disponibilizados através de catálogos ou através da Web. Permitem aos operadores e às entidades que gerem os recursos turísticos a criação de produtos compostos, muito atractivos aos turistas. A grande desvantagem neste tipo de pacotes é a pouca flexibilidade que oferecem. Os itinerários são fixos, as datas de viagem inflexíveis e as opções em relação aos produtos que constituem o pacote são limitadas. Para o típico turista, o ideal seria a possibilidade de encontrar e reservar o conjunto de produtos turísticos que lhe servisse exactamente, em termos de datas, horas, lugares e de forma a conseguir reservar estes produtos pelo menor valor monetário possível. Claramente os pacotes pré-definidos não conseguem satisfazer estes parâmetros.

De forma a satisfazer melhor o turista, surgiu o conceito de "Dynamic Packaging". Entende-se por "Dynamic Packaging " a possibilidade que um turista, ou um agente turístico, tem de construir um itinerário costumizável, adicionando vários componentes escolhidos por este e completar esta transacção em tempo real (Lofgren, 2005). O "Dynamic Packaging" caracteriza-se também pela atribuição de um único valor monetário para o pacote turístico (escondendo o valor de cada um dos componentes) em 5 a 15 segundos (Fitzgerald, 2005). Um sistema que suporte "Dynamic Packaging" deverá permitir a criação de pacotes incluindo componentes tais como voos, alojamento, aluguer automóvel, excursões locais, bilhetes para o teatro, e eventos desportivos (Cardoso e Lange, 2005). Neste documento iremos denominar de pacotes dinâmicos os pacotes criados segundo a perspectiva do "Dynamic Packaging". A grande diferença entre os pacotes pré-definidos e dos pacotes dinâmicos encontra-se

no processo de criação. Enquanto os primeiros são criados com bastante tempo de antecedência, os pacotes dinâmicos são criados aquando da apresentação dos requisitos do cliente. Esta diferença é ilustrada pela figura 2.1.

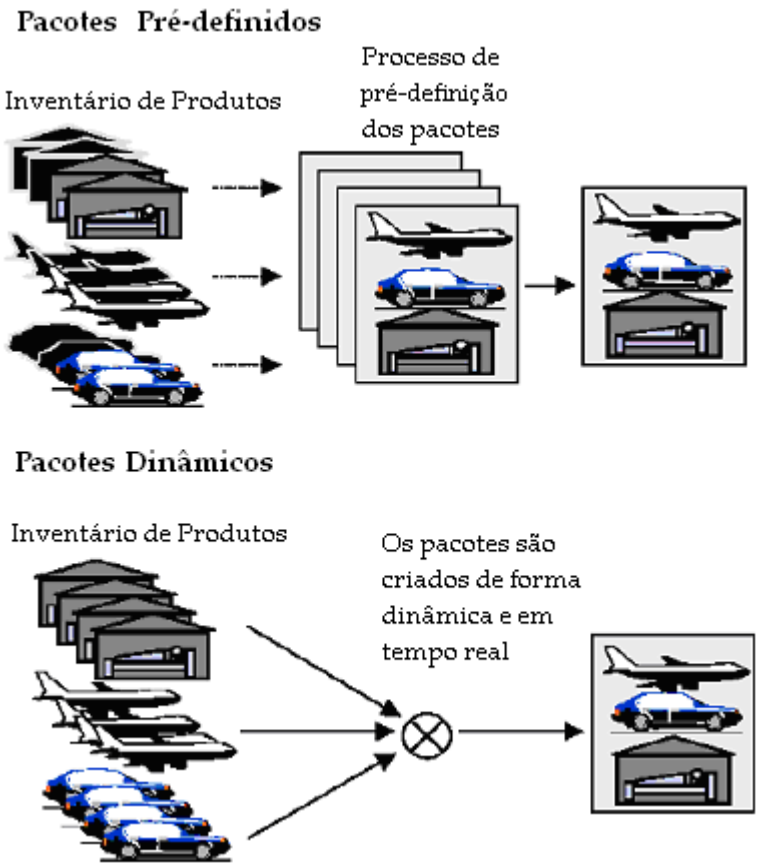

Figura 2.1 – Pacotes Pré-definidos e Pacotes Dinâmicos (Kabbaj, 2003).

Para além das grandes vantagens que os pacotes dinâmicos trazem aos turistas, estes trazem também vantagens para os operadores turísticos e para as entidades que gerem os recursos turísticos. O facto de os pacotes pré-definidos serem definidos com muito tempo de antecedência e com preços fixos restringe a habilidade do operador turístico de adaptar o preço dos pacotes com base nas informações das reservas efectuadas e das tendências de mercado. Por vezes são criadas cinco ou mais versões dos pacotes pré-definidos para que seja possível a alteração dos preços dos produtos. No entanto, esta solução provou ser cara e complexa de operar. O suporte ao "Dynamic Packaging" permite a alteração dos valores dos pacotes em tempo real. Assim, os operadores turísticos podem adaptar os preços dos pacotes às alterações de mercado quase de imediato (Cardoso e Lange, 2005).

A implementação de um sistema que suporte a criação de pacotes dinâmicos é uma tarefa muito complexa. Para que um sistema suporte o conceito de "Dynamic Packaging" terá de implementar os seguintes requisitos (Kabbaj, 2003):

1) Deverá ter acesso a um grande número de produtos turísticos. Estes poderão pertencer a diferentes entidades e poderão ser disponibilizados por diferentes sistemas. Cada sistema terá o seu processo específico de comunicação e a informação disponibilizada poderá ser apresentada de diferentes formas de sistema para sistema.

2) Deverá permitir a pesquisa de produtos turísticos e deverá oferecer a possibilidade de agregação destes produtos de forma a criar um pacote.

3) Deverá permitir ao cliente a escolha de qualquer combinação de produtos na criação dos pacotes.

4) A criação dos pacotes deverá ser realizada através de um único ponto de comunicação sem ser necessário interacção directa do cliente com outros sistemas.

5) Deverá calcular o valor total do pacote criado em tempo real, aplicando as regras de descontos para a combinação dos produtos escolhidos.

As três maiores Agências de Viagens on-line aperceberam-se da importância do conceito de "Dynamic Packaging " e já implementaram-no nos seus sistemas. A agência Expedia foi a pioneira e desde o ano de 2002 que suporta o "Dynamic Packaging". Actualmente 30% do seu lucro provém da venda de pacotes turísticos. A grande vantagem do sistema de "Dynamic Packaging" da Expedia é a possibilidade de ser possível criar pacotes dinamicamente incluindo produtos diferentes dos produtos tradicionais (hotel, viagem e aluguer de carro). O sistema da Expedia permite a criação de pacotes dinâmicos que podem incluir outros produtos relacionados com diversão e actividades (Fitzgerald, 2005). As outras Agências de Viagens on-line também implementaram o suporte ao "Dynamic Packaging" para não perderem a sua quota-parte no mercado do turismo. A Travelocity será a primeira a disponibilizar um sistema de "Dynamic Packaging" que permite escolher o lugar no avião e escolher um determinado quarto de hotel. A Orbitz tenta valorizar o seu sistema de "Dynamic Packaging" através do apoio ao cliente após a reserva dos pacotes. Depois do cliente

fazer a sua reserva, recebe informações importantes que podem influenciar a sua viagem. Informações como as condições atmosféricas, atrasos nos transportes ou outros, são enviados ao cliente através de correio electrónico, ou através do envio de SMS para o telemóvel (Cardoso e Lange, 2005).

2.2 Integração de Sistemas de Informação

Num mundo cada vez mais globalizado, onde a informação toma um papel decisivo, é cada vez mais necessário a troca de informação entre as empresas e dentro das empresas. Com o objectivo de tratar toda a informação existente, as empresas foram construindo sistemas de informação e sistemas de apoio à decisão. De início, estes sistemas eram criados para gerir a informação referente aos departamentos dentro de uma organização. Cada departamento possuía o seu sistema de informação independente e autónomo. A construção, operação e manutenção destes sistemas era complicada e implicava muito tempo despendido. Estes problemas aumentam à medida que aumenta o número de sistemas existentes (Lawrence e Barker, 1999). Além disso, cada um dos sistemas independentes continha informação que era comum a vários departamentos. A integração dos sistemas de informação dentro de uma empresa veio resolver todos estes problemas. A integração dos sistemas de informação torna possível a obtenção de uma visão unificada de toda a informação existente numa empresa. Uma visão unificada dos recursos de informação de uma empresa traz muitas vantagens:

1) **Procura de informação:** é mais fácil aceder à informação pretendida e é mais simples obter informação associada a esta, independentemente do sistema onde esta se encontra.

2) **Integração dos dados:** o acesso e a manipulação de diferentes fontes de dados pode ser suportado e automatizado através de uma visão unificada. O impacto das alterações pode ser gerido mais facilmente.

3) **Qualidade dos dados:** a consistência dos dados poderá ser mais facilmente verificada e gerida, porque as regras que forçam a consistência dos dados podem ser mantidas de forma central. É também mais fácil detectar e eliminar a informação redundante existente entre as várias fontes de informação.

A integração dos sistemas de informação dentro de uma empresa pode ser transportada para fora destas e ser aplicada na relação entre empresas. Através da integração de sistemas de diferentes empresas podemos ter as mesmas vantagens existentes quando uma empresa integra toda a sua informação interna. A integração dos dados internos com os dados dos sistemas dos fornecedores, dos clientes ou de parceiros, irá trazer as vantagens já enumeradas anteriormente.

A integração de dados torna-se assim um aspecto muito importante no sucesso das empresas actuais. A grande competição existente leva a que as empresas tentem a todo o custo aperfeiçoar a gestão da informação. A existência de sistemas centralizados que forneçam informações importantes para as decisões de negócio, é uma prioridade no melhoramento da gestão da informação. A solução passa pela aquisição de sistemas que utilizam os dados existentes sem prejudicarem as aplicações já existentes. Estes sistemas irão melhorar os resultados e a produtividade no acesso e manutenção dos dados referentes às empresas (Hit Software, 2005). No entanto, a integração de informação proveniente de diferentes fontes de dados pode tornar-se numa tarefa muito complexa. Actualmente, as cooperações europeias gastam mais de 10 biliões de Euros resolvendo problemas na área da integração de sistemas (Alexiev, Breu, Bruijn, Fensel, Lara, e Lausen, 2005). De acordo com estudos recentes, mais de 30% de todo o investimento em tecnologias de informação será aplicado na integração de aplicações empresariais (Sink, 2002).

O grande problema existente na integração de sistemas, prende-se com a heterogeneidade dos dados a serem integrados. Como cada sistema foi desenhado para um determinado propósito, é natural que a estrutura dos dados e a forma como são guardados difere de sistema para sistema. Podem existir quatro diferentes tipos de heterogeneidade entre diferentes fontes de dados: heterogeneidade de sistema, heterogeneidade sintáctica, heterogeneidade estrutural e heterogeneidade semântica (Cardoso e Amit, 2006).

Heterogeneidade de sistema: As aplicações e os dados poderão estar guardados em diferentes plataformas e/ou sistemas operativos.

Heterogeneidade sintáctica: As aplicações poderão utilizar diferentes representações e codificações de dados. Por exemplo, uma aplicação poderá utilizar o sistema ASCII e outra, o sistema Unicode.

Heterogeneidade estrutural: Acontece quando diferentes sistemas de informação utilizam diferentes modelos de dados ou diferentes estruturas e esquemas de dados. Por exemplo, num sistema os dados poderão ser estruturados em documentos XML enquanto noutro poderão ser estruturados utilizando o modelo relacional.

Heterogeneidade Semântica: O significado dos dados pode ser expresso de diferentes formas. Por exemplo, numa empresa que opera na área do turismo poderá existir, num determinado sistema, o conceito "cliente" que pode representar os turistas e os agentes de viagens que interagem com a empresa. Noutro sistema, o mesmo conceito poderá representar apenas os agentes de viagens.

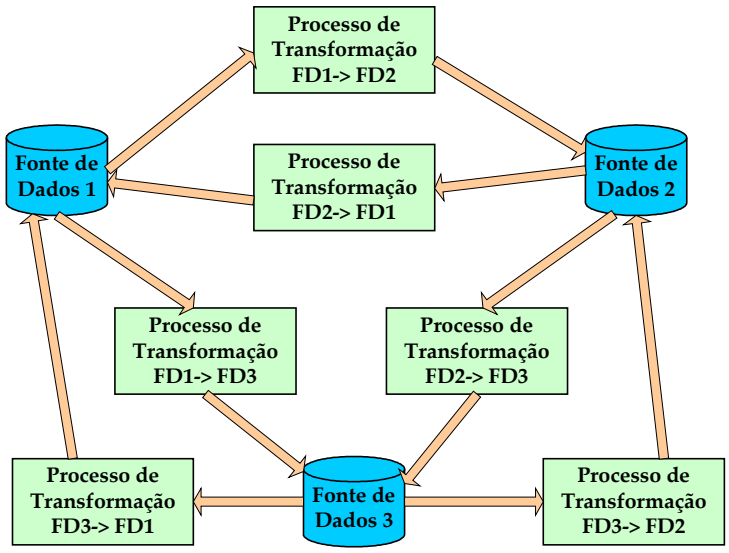

Figura 2.2 – Integração "Ad-hoc".

Existem três tipos de integração que podem ser utilizados para resolver o problema da heterogeneidade entre sistemas: integração "ad-hoc", integração usando uma Ontologia partilhada ou integração utilizando "clustering" de ontologias (Alexiev, Breu, Bruijn, Fensel, Lara, e Lausen, 2005). Na integração "ad-hoc" são criados programas de transformação que têm a função de transformar a informação de um formato para outro. Este tipo de transformação é difícil de manter, no caso de um dos formatos de informação ser alterado é necessário detectar essa alteração e depois actualizar o programa de transformação em conformidade com a alteração ocorrida. Este problema torna-se ainda maior quando existem vários programas de transformação dentro de uma empresa. A integração "ad-hoc" não é escalável porque para cada duas fontes de dados que têm de ser integradas é necessário implementar um processo de transformação próprio, que depois deverá ser mantido. Caso a

estrutura de uma determinada fonte de dados seja alterada, é necessário alterar todos os processos de transformação que têm como ponto de partida esta fonte de dados e todos os processos de transformação que têm esta fonte de dados como ponto de destino. A integração "ad-hoc" encontra-se representada na figura 2.2

A integração utilizando uma Ontologia partilhada passa pela utilização de uma Ontologia central onde todos os modelos de dados, pertencentes às fontes de dados, são mapeados. Uma Ontologia é uma especificação formal e explícita de um modelo abstracto de conceitos partilhados (Gruber, 1993). A desvantagem de utilizar uma integração deste tipo é a dificuldade de criar uma Ontologia que reuna o consenso de todos os interessados, principalmente se estes utilizam diferentes terminologias para o mesmo domínio. A grande vantagem prende-se com uma manutenção muito mais simples que o caso da integração "ad-hoc". Neste caso, se o modelo de dados de uma fonte de dados for alterado, é necessário apenas alterar o mapeamento desta com a Ontologia partilhada. Na figura 2.3 podemos observar a representação de uma integração através da utilização de uma Ontologia partilhada.

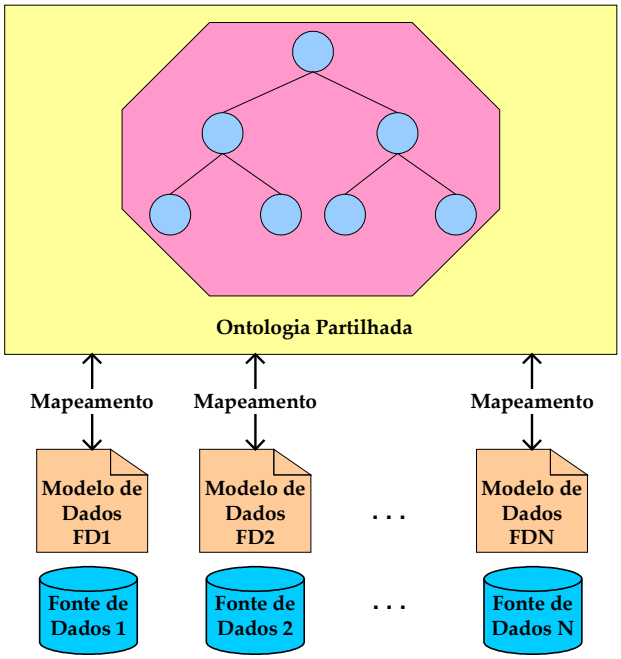

Figura 2.3 – Integração utilizando uma Ontologia Partilhada (Alexiev, Breu, Bruijn, Fensel, Lara, e Lausen, 2005).

A integração utilizando "clustering" de Ontologias, é baseada nas semelhanças entre conceitos conhecidos dos diferentes agentes. Tal como nas relações interpessoais, recursos que partilham conceitos similares poderão ter conversas mais "profundas"

que com aqueles recursos que não partilham tantos conceitos (Visser e Tamma, 1999). Neste caso, em vez de existir apenas uma Ontologia partilhada, o conhecimento partilhado é colocado em várias pequenas Ontologias. Os recursos não se comprometem com apenas uma única Ontologia global, em vez disso, são agrupados com base na similaridade dos conceitos que têm sobre o domínio em questão. As várias Ontologias são então organizadas de forma hierárquica. Nos níveis de topo encontram-se os conceitos mais genéricos, que são partilhados por todos os recursos. Nos níveis mais baixos, os conceitos mais genéricos são expandidos e caracterizados de forma a representar os conceitos específicos de um ou mais recursos. Como cada conceito pode ser expandido de diferentes formas pelos vários recursos, neste tipo de integração é possível a existência de Ontologias heterogéneas. Na figura 2.4 encontra-se representado um modelo possível para a integração de recursos utilizando o "clustering" de Ontologias.

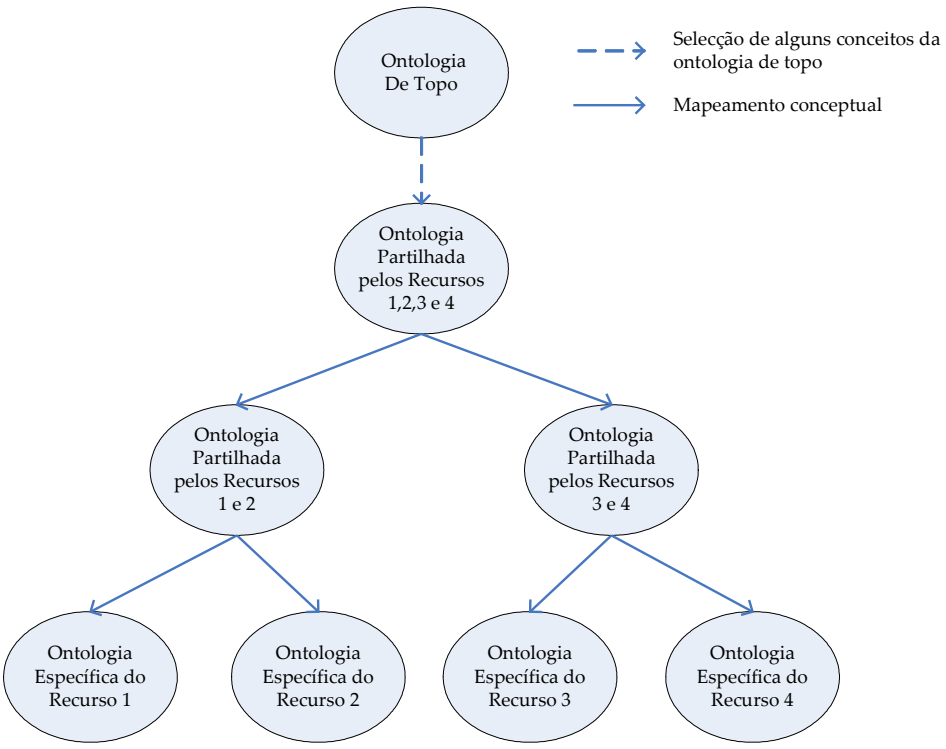

Figura 2.4 – Integração utilizando "Clustering" de Ontologias (Visser e Tamma, 1999).

2.3 WEB SEMÂNTICA

O problema da integração de sistemas heterogéneos é claramente mais fácil de resolver através da utilização de uma Ontologia global que define o domínio do problema. A Ontologia deverá ser definida numa linguagem que possa ser compreendida pelas máquinas, de forma a automatizar ao máximo o difícil e moroso trabalho de integração da informação. As tecnologias relacionadas com a Web Semântica poderão dar uma grande ajuda na definição e manipulação de Ontologias. Na nossa arquitectura iremos utilizar estas tecnologias para resolver o problema da integração de informação. Nesta secção, descrevemos a Web Semântica e explicamos quais as principais vantagens na utilização das tecnologias associadas a esta.

2.3.1 Definição da Web Semântica

O inventor da WWW (World Wide Web), Tim Berners-Lee, tem uma visão dupla do futuro desta. A primeira parte da visão é tornar a Web, ainda mais, num meio de colaboração. A segunda parte da visão é tornar a Web compreensível e processável pelas máquinas. Na figura 2.5 apresentamos a proposta inicial que Tim Berners-Lee apresentou ao CERN para a criação da primeira Web.

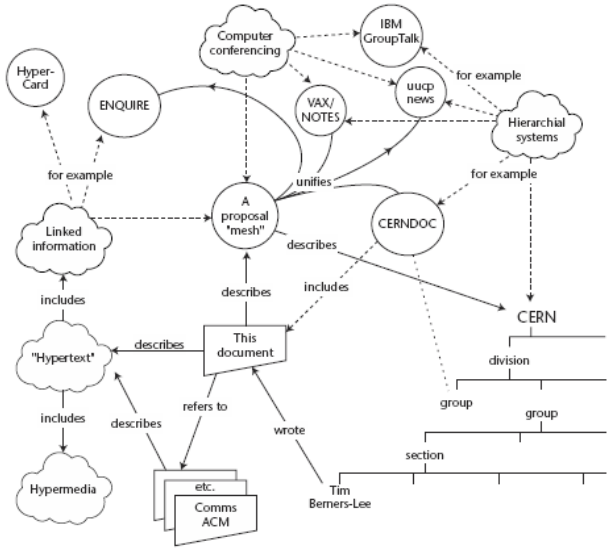

Figura 2.5 – Web original proposta ao CERN por Tim Berners-Lee (Daconta, Obrst e Smith, 2003)

Claramente, a visão de Berners-Lee não é suportada pela actual linguagem utilizada na Web, o HTML. Na figura 2.5 podemos observar relações entre unidades de informação tais como "includes", "describes" ou "wrote". Actualmente, a Web está em evolução e diferentes ideias estão a ser desenvolvidas de forma a adicionar semântica aos recursos existentes nesta. Na figura 2.6 podemos observar, no lado esquerdo, a representação da Web sintáctica. Neste caso, os recursos são ligados formando a Web. Não existe qualquer distinção entre os recursos ou entre as ligações que unem os recursos. Para adicionar significado aos recursos e às ligações, novos standards e linguagens estão a ser investigados e desenvolvidos. As regras e informação descritiva oferecida por estas linguagens permite caracterizar individualmente e com precisão o tipo de recursos existentes na Web e as ligações entre recursos (Cardoso, 2005). Esta Web semântica é apresentada no lado direito da figura 2.6.

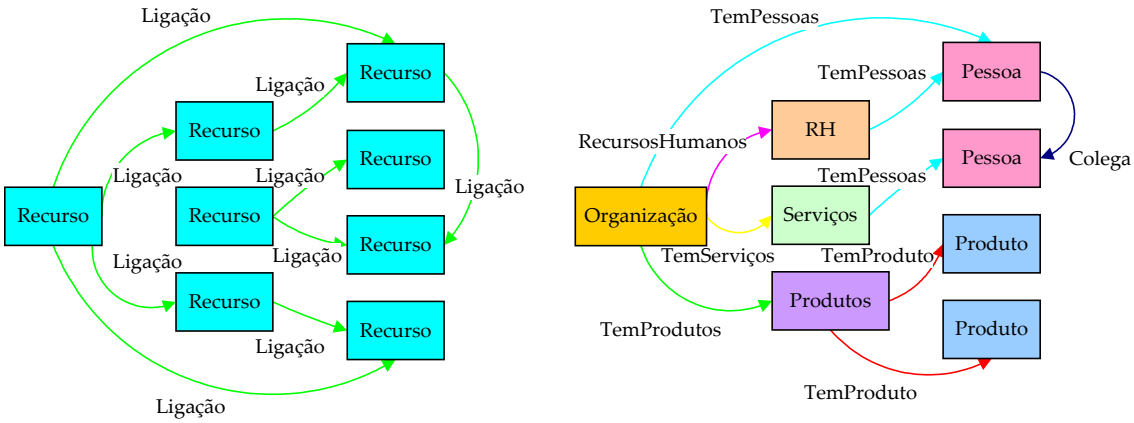

Figura 2.6 – Evolução da Web (Cardoso, 2005).

Podemos definir a Web Semântica como sendo uma Web processada por máquinas composta por dados inteligentes. Dados inteligentes são dados independentes de qualquer plataforma tecnológica, que podem ser compostos, classificados e fazendo parte de uma grande Ontologia. A Web Semântica não se refere apenas à World Wide Web, representa um conjunto de tecnologias que poderão ser aplicadas localmente nos sistemas de informação das empresas (Daconta, Obrst e Smith, 2003).

2.3.2 Vantagens da Web Semântica

A possibilidade de adicionar semântica à informação de forma a ser entendida pelas máquinas, leva a que muitas das acções antes realizadas pelos humanos possam ser automatizadas através de acção das máquinas. Este facto traz muitas vantagens

nomeadamente na gestão de conhecimento. Eis alguns exemplos de vantagens na utilização das tecnologias associadas à Web Semântica (Daconta, Obrst e Smith, 2003):

(1) **Apoio à Decisão:** O acesso ao conhecimento leva-nos a tomar boas decisões. As organizações são compostas por sub-organizações, divisões, grupos e projectos. Cada um destes conjuntos tem as suas fontes de informação. Para tomar o máximo partido destes grupos, é necessário combinar toda a informação referente a cada um dos grupos e compreender a relação entre estes. Existe trabalho a decorrer para a criação de sistemas de suporte à decisão baseados na semântica (DSSs). Baseiam-se na análise de agentes de software e na interacção entre o utilizador e o computador para a tomada de decisão, com o intuito de ajudar o utilizador final a tomar decisões com base em informação correcta e útil (Casey e Austin, 2001). Mesmo sem a utilização de sistemas de apoio à decisão, agentes de software poderão monitorizar a base de conhecimento de forma a transmitir alertas sobre determinados factos.

(2) **Desenvolvimento do Negócio:** É importante que todos os membros da organização tenham informação actualizada ao minuto, para que a organização tenha sucesso no seu negócio. Na maioria dos casos, não é possível que as pessoas que tratam das vendas consigam ter o conhecimento total do que se passa dentro da organização. Esta falta de conhecimento poderá levar à não concretização de negócios que poderiam ser vantajosos para a organização. Se existir uma base de conhecimento interna sobre a organização, o pessoal das vendas poderá aceder a esta e recolher informação importante para tornar possível e aproveitar todas as oportunidades de negócio. Outra forma de tentar melhorar o negócio é através da utilização dos sistemas de gestão de relacionamento com o cliente (CRM). Os sistemas CRM permitem a colaboração entre parceiros, clientes e empregados, disponibilizando informação relevante e personalizada a partir de várias fontes de dados existentes numa organização. Existem alguns problemas quando se tenta integrar toda a informação que se encontra espalhada por vários sistemas, que por vezes são sistemas proprietários fechados. A capacidade de comparação entre informação referente a diferentes domínios dentro de uma organização é

outra das acções que os sistemas actuais de CRM têm dificuldade em realizar. As tecnologias associadas à Web Semântica irão permitir às empresas a criação de soluções de CRM mais inteligentes.

(3) **Partilha de Informação e Procura de Conhecimento:** A partilha de informação e a comunicação é primordial em qualquer organização. O problema é que à medida que a organização cresce maior informação é recolhida e mais comunicação é necessária. Até que chega a um ponto em que a informação é tanta que é difícil obter a informação que desejamos. Este estrangulamento de informação leva a que por vezes seja reinventada a roda dentro da organização. A criação de uma base de conhecimento em que exista a descrição de cada um dos projectos e/ou dos trabalhos realizados, poderá evitar muito trabalho duplicado. A inclusão na base de conhecimento das resoluções de problemas já anteriormente resolvidos também é uma outra forma de ganhar tempo nos processos.

(4) **Administração e Automação:** Outras das vantagens na criação de uma base de conhecimento é a possibilidade que os programas de software têm de automatizar tarefas administrativas. Por exemplo, a reserva de produtos turísticos é uma tarefa que se pode tornar muito complexa e demorada. Todos nós temos preferências nos produtos e serviços turísticos a consumir. Preferências no transporte (carro, comboio, avião, etc.), no tipo de hotel ou no tipo de carro a alugar. Juntar todas estas preferências às restrições temporais e de orçamento, criam-se um conjunto de requisitos que poderá implicar muito tempo de procura para encontrar os produtos e serviços que respondam a todos os requisitos. Através da Web Semântica, muita desta procura poderá ser realizada através de processos automáticos.

2.3.3 Arquitectura da Web Semântica

Uma das principais premissas arquitecturais da Web Semântica é o diagrama de linguagens apresentado pela primeira vez por Tim Berners-Lee em 2001. Este diagrama é ilustrado na figura 2.7.

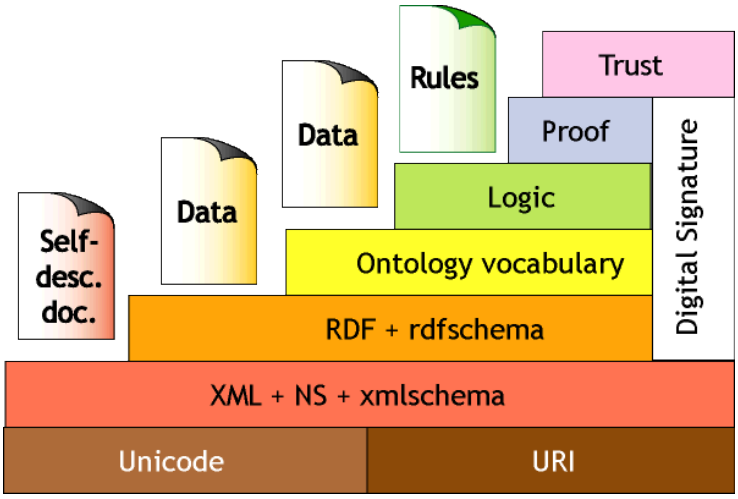

Figura 2.7 – Arquitectura da Web Semântica (Berners-Lee, 2000).

URI e Unicode

URI (Universal Resource Identifier) é um conjunto de caracteres formatados, utilizados para identificar recursos físicos ou abstractos. Um URI pode ser classificado como uma localização, um nome ou ambos. URL (Uniform Resource Locator) refere-se a um subconjunto do URI, identifica recursos através da representação do seu mecanismo de acesso primário. O URN (Uniform Resource Name) refere-se a um subconjunto do URI, é necessário para a identificação única, global e persistente, mesmo quando o recurso deixe de estar disponível ou mesmo que este deixe de existir (Cardoso, 2005). Por exemplo:

- O URL http://www.uma.pt identifica a localização onde uma determinada página Web poderá ser acedida.

- O URN urn:isbn:0-465-05673-3 identifica um determinado livro usando o seu ISBN (International Standard Book Number)

O sistema de codificação Unicode atribui um número único para cada caracter independentemente da plataforma, do programa ou da língua. Antes da criação do Unicode, existiam vários sistemas de codificação de caracteres. Era necessário que os computadores suportassem vários sistemas de codificação para apresentarem a informação correctamente. Mesmo assim, existiam conflitos entre tipos de codificação, um mesmo código poderia ser utilizado por diferentes sistemas de codificação para representar diferentes caracteres. Os sistemas de codificação ASCII e EBCDIC são

alguns exemplos de sistemas existentes antes da criação do sistema Unicode. Actualmente o sistema Unicode é suportado por muitos sistemas operacionais, por todos os browsers modernos e por muitos outros produtos. A utilização do Unicode oferece uma redução significativa nos custos quando comparado com a utilização dos outros sistemas de codificação. O Unicode permite a troca de informação entre sistemas e plataformas, sem a necessidade de processos de conversão (Unicode Consortium, 2007).

XML e XML Schema

XML (Extensible Markup Language) é um conjunto de regras sintácticas para a criação de linguagens de marcação (Daconta, Michael, Obrst e Smith, 2003). Nas linguagens de marcação são utilizadas marcas descritivas que definem o início e o fim do texto marcado como unidade ou elemento de informação. O XML é aceite como sendo um standard para a troca de dados na Web, permitindo a estruturação dos dados mas sem comunicar o seu significado semântico. Em contraste com o HTML, com o XML é possível a criação de novos tipos de marcação que carregam alguma semântica. Por exemplo, podemos criar a marcação <hotel> que na perspectiva humana claramente carrega informação semântica. No entanto, na perspectiva computacional a marcação <hotel> é muito semelhante à marcação HTML <h1>. Ou seja, a semântica existente no XML é a mesma que podemos encontrar no HTML. De qualquer forma, a utilização do XML resolve muitos problemas que antes não eram possíveis de resolver utilizando apenas o HTML. Nomeadamente problemas na área da troca de informação ou na integração de sistemas (Cardoso, 2005).

Um documento XML bem formatado é constituído por uma árvore de grupos de aberturas e de fechos de marcações. Para cada uma abertura e fecho de marcação podem existir vários pares de atributos e valor. No próximo exemplo apresentamos um documento XML que contem informações sobre um determinado hotel. O documento inclui várias marcações tais como <nome>, <estrelas> e <telefone> que adicionam vários detalhes sobre o hotel.

```
<hotel hotel_id="1">
      <nome> Hotel Madeira</nome>
      <estrelas>3</estrelas>
      <telefone>351 29175160</telefone>
</hotel>
```

Não existe nenhuma limitação quanto à utilização das marcações num documento XML. Na versão 1.0 podemos utilizar documentos DTD (Document Type Definition) de forma a restringir a quantidade de marcações a serem utilizadas e de como estas deverão ser combinadas, qual a ordem e quais as marcações que poderão ser incluídas dentro de outras marcações. Actualmente existe a vontade de substituir os documentos DTD pelos documentos XML Schema. O XML Schema oferece muitas vantagens sobre os DTDs pois fornece mais formas de restrições sobre os documentos XML. De qualquer forma, ambos têm o mesmo fim, definir uma gramática para os documentos XML (Davies, Fensel e Harmelen, 2003).

RDF e RDF Schema

Muitas vezes chamada de linguagem, o RDF é essencialmente um modelo de dados. Baseia-se em triplas de objecto-atributo-valor chamadas de frases. A sintaxe utilizada para representar e transmitir as frases RDF é baseada no XML. Assim, o RDF herda os benefícios associados ao XML. No entanto são possíveis representações de RDF sem ser baseadas no XML.

Os elementos fundamentais existentes no RDF são os recursos, as propriedades e as frases. Podemos pensar nos recursos como sendo um objecto sobre o qual queremos falar. Exemplos de recursos podem ser hotel, viagem, turista etc. Cada recurso contém um URI associado que poderá ser um URL ou um URN. As propriedades são um tipo especial de recurso. Elas descrevem relações sobre recursos. Por exemplo, "gerido por", "reservado por" e "localizado em", são exemplo de possíveis propriedades. Tal como no caso dos recursos, as propriedades também são identificadas através de um URI. As frases atribuem valores às propriedades dos recursos. Uma frase é composta por uma tripla objecto-atributo-valor, constituídos por um recurso, uma propriedade e um valor. Os valores poderão ser um recurso ou então um valor atómico como um valor inteiro ou uma "string".

O exemplo de uma frase RDF poderá ser: "O Hotel Madeira tem o endereço Funchal". Nesta frase o "Hotel Madeira" é o recurso. O recurso pode ter o URI referente à localização da página Web do hotel "www.hotelmadeira.pt" e tem a propriedade "Endereco" que tem o valor "Funchal". Esta frase está representada na figura 2.8.

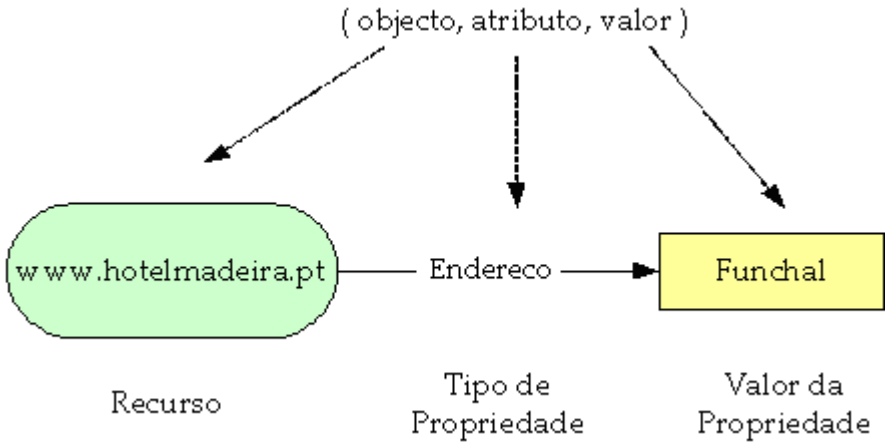

Figura 2.8 – Representação gráfica de uma frase RDF.

O RDF Schema é uma linguagem de descrição de vocabulário utilizada para descrever propriedades e classes dos recursos RDF. O RDF Schema toma um passo em frente no enriquecimento da representação formal e introduz primitivas de modelação ontológica. Através do RDF Schema podemos falar sobre classes, subclasses, domínio e restrições sobre propriedades. Apesar da semelhança nos nomes, o RFD Schema toma um papel diferente em relação ao XML Schema. O XML Schema, e também os DTDs, definem a ordem e a combinação de marcações possíveis num documento XML. Em contraste, o RDF Schema apenas disponibiliza informação sobre a interpretação das frases presentes num modelo de dados RDF. O RDF Schema não define restrições sintácticas sobre um modelo de dados RDF. Através do RDF Schema podemos definir um vocabulário específico para os dados RDF, também podemos definir os tipos de objectos nos quais estes atributos podem ser aplicados. Ou seja, o mecanismo RDF Schema disponibiliza um sistema de tipos básico para os modelos RDF. Este sistema de tipos utiliza alguns termos pré-definidos tais como "Class", "subPropertyOf" e "subClassOf". Através destes termos podemos definir classes, subclasses das classes definidas e sub-propriedades de propriedades definidas (Davies, Fensel e Harmelen, 2003). No exemplo seguinte, apresentamos a definição da classe "Acomodacao" e a definição da subclasse "Hotel" da classe "Acomodacao".

```
<rdfs:Class rdf:ID="Acomodacao"/>
<rdfs:Class rdf:ID="Hotel">
     <rdfs:subClassOf rdf:resource="#Acomodacao"/>
</rdfs:Class>
```

As propriedades podem ser definidas definindo o domínio no qual podem ser aplicadas e definindo qual o conjunto de valores que podem tomar. Também podem

ser organizadas hierarquicamente através da utilização do termo "subPropertyOf". De seguida apresentamos um exemplo RDF Schema onde definimos a propriedade "Endereco". O domínio da propriedade (rdfs:domain) é a classe "Hotel" e o conjunto de valores que pode tomar (rdfs:range) é definido pelo tipo XSD "string".

```
<rdf:Property rdf:ID="Endereco"/>
      <rdfs:domain rdf:resource="#Hotel"/>
      <rdfs:range rdf:resource="&xsd;string"/>
</rdfs:Property>
```

Ontologia

A expressividade do RDF e do RDF Schema é claramente limitada. O RDF limita-se à definição de triplas "objecto-atributo-valor" e o RDF Schema limita-se à definição de uma hierarquia de subclasses e à definição de hierarquia de propriedades, onde para cada propriedade é definido o domínio e o tipo de valores que a propriedade pode tomar. O grupo Web Ontology Working Group, que trabalha na área das tecnologias semânticas para a Web e que pertence à W3C, identificou um conjunto de casos de utilização característicos da Web Semântica que requerem muito mais expressividade que a existente no RDF e no RDF Schema. Vários grupos de pesquisa dos Estados Unidos e da Europa já tinham identificado a necessidade da existência de uma linguagem de modelação que permitisse um maior poder semântico. Isto levou à criação da linguagem DAML+OIL que foi criada por estes grupos de investigação. O nome desta linguagem surgiu através da junção da proposta americana DAML-ONT e da proposta Europeia OIL. A linguagem DAML+OIL foi tomada pelo grupo W3C Web Ontology Working Group como ponto de partida para a criação da linguagem OWL. O OWL pretende ser a linguagem para a definição de ontologias standard da Web Semântica. Uma linguagem semântica deverá permitir definir conceptualizações de modelos de domínio, de forma formal e explícita. Os principais requisitos de uma linguagem semântica são (Antoniou, e Harmelen 2004):

- Uma sintaxe bem definida.
- Uma semântica formal para descrever precisamente o significado do conhecimento de forma a não gerar diferentes interpretações.
- Deverá suportar inferência de conhecimento de forma a automatizar o raciocínio sobre a Ontologia.

- Deverá possuir um bom poder de expressividade de forma a possibilitar uma boa representação do conhecimento.

Uma linguagem semântica deverá ter um bom poder de expressividade, mas normalmente quanto maior é a expressividade de uma linguagem menos eficiente é o seu suporte à inferência. Quando o poder de expressividade é muito, podemos mesmo chegar ao ponto de não conseguirmos utilizar sistemas de inferência.

Na tentativa de responder a todos os requisitos de uma linguagem semântica, o grupo W3C Web Ontology Working Group definiu três diferentes sub-linguagens, cada uma pretende responder a determinados aspectos dos requisitos identificados. As três sub-linguagens criadas são:

1) **OWL Full:** Utiliza a linguagem OWL completa permitindo a utilização de todas as primitivas existentes. Permite também a combinação destas primitivas com o RDF e o RDF Schema de forma arbitrária. Isto inclui a possibilidade de alterar o significado de primitivas de OWL ou RDF aplicando as primitivas da outra linguagem, RDF ou OWL. Por exemplo, no OWL Full podemos impor uma restrição de cardinalidade na classe que representa todas as classes, limitando o número de classes que poderá existir numa Ontologia. A grande vantagem do OWL Full é ser compatível com o RDF, tanto a nível sintáctico como a nível semântico. Qualquer documento RDF válido é também um documento OWL Full válido. A desvantagem na utilização do OWL Full é o facto de não suportar a inferência. Esta incapacidade deve-se à sua grande capacidade de expressividade.

2) **OWL DL:** É uma sub-linguagem do OWL Full. Foi criada de forma a termos maior eficiência computacional. Restringe a utilização dos construtores de OWL e RDF. A utilização dos construtores sobre outros construtores não é permitida. Isto garante o suporte a sistemas de inferência eficientes. A desvantagem desta sub-linguagem reside no facto de não ser completamente compatível com o RDF. Um documento RDF terá de ser estendido ou restringido de forma a tornar-se num documento OWL DL válido. Todos os documentos OWL DL válidos são também documentos RDF válidos.

3) **OWL Lite**: É uma sub-linguagem que cria ainda mais restrições que aquelas impostas pela OWL DL. Por exemplo, o OWL Lite exclui a possibilidade de utilização das classes enumeradas, das funções de disjunção e cardinalidade arbitrária. Esta sub-linguagem tem a vantagem de ser de simples compreensão para os utilizadores e de ser simples de implementar por quem desenvolve ferramentas com base nesta sub-linguagem. A grande desvantagem é a fraca expressividade que esta oferece.

O OWL Lite permite a utilização de todas as funções existentes no RDF Schema. Entre estas incluem-se a definição de classes, de subclasses, de propriedades e de instâncias de classes. No caso das propriedades, podemos definir dois tipos diferentes. Ou a propriedade refere-se a tipo de dados como, por exemplo um inteiro ou uma "string", ou então a propriedade refere-se a uma classe ou a um grupo de classes. No caso de referir-se a tipo de dados chamamos Propriedade de Tipo de Dados e no caso de referir-se a uma classe ou grupo de classes chamamos Propriedade de Objecto.

Para além das funcionalidades do RDF Schema, O OWL Lite também suporta (McGuinness e Harmelen, 2004):

- A definição de igualdade e desigualdade entre classes e propriedades.
- A definição de características de propriedades, como por exemplo, propriedade simétrica ou transitiva.
- A definição de restrições sobre os valores que uma propriedade pode tomar.
- A definição de intersecção entre classes e restrições.

O OWL DL e o OWL Full utilizam o mesmo vocabulário, a única diferença entre estas duas sub-linguagens é o facto do OWL DL obrigar a separação de tipos, ou seja, uma classe não pode ser uma instância ou propriedade e uma propriedade não pode ser uma instância ou uma classe. Isto implica que não seja possível aplicar restrições a elementos primários da linguagem OWL, algo que é possível no OWL Full. As funcionalidades existentes no OWL DL e OWL Full e que não existem no OWL Lite são:

- Utilização de classes enumeradas.

- Restrições sobre propriedades de objectos, indicando que uma propriedade deverá ter uma determinada instância como valor.

- Definição de classes disjuntas, indicando que uma instância não pode pertencer ao mesmo tempo às classes definidas como disjuntas.

- Suporte completo às restrições por cardinalidade, permitindo cardinalidade arbitrária utilizando valores inteiros não negativos.

- Definição de classes complexas, permitindo a definição de classes através de descrições arbitrárias e complexas constituídas por classes enumeradas, restrições de propriedades e por combinações lógicas.

Para o mesmo exemplo apresentado no caso do RDF Schema, apresentamos o OWL correspondente.

```
<owl:Class rdf:ID="Acomodacao"/>
<owl:Class rdf:ID="Hotel">
      <rdfs:subClassOf rdf:resource="#Acomodacao"/>
</owl:Class>
<owl:DatatypeProperty rdf:ID="Endereco"/>
      <rdfs:domain rdf:resource="#Hotel"/>
      <rdfs:range rdf:resource=""&xsd;string""/>
</owl:DatatypeProperty>
```

Lógica e Prova

Os níveis referentes à lógica e à prova pretendem disponibilizar funcionalidades iguais às que se pode encontrar nos sistemas de lógica de primeira ordem. Com estes sistemas, a partir da declaração de um qualquer conjunto de princípios lógicos, o computador pode raciocinar e inferir utilizando estes princípios. Por exemplo, um operador turístico pode decidir criar uma regra em que a reserva de um quarto em hotéis de 4 ou mais estrelas tem uma redução de 10% no valor final. Um sistema que suporte lógica poderá utilizar esta regra e chegar à seguinte conclusão: "Foi reservado um quarto no Hotel Savoy que tem 5 estrelas, logo esta reserva terá uma redução de 10% no seu valor final" (Cardoso, 2005).

As aplicações de software que permitem o suporte à lógica e prova têm o nome de motores de inferência ou de motores de raciocínio. Estas aplicações conseguem descobrir novos factos ou associações entre factos, a partir da informação já existente. A inferência e as regras de inferência permitem obter novos dados a partir de dados já

conhecidos. Assim, novo conhecimento pode ser adicionado baseado no conhecimento já existente. Ao criarmos um modelo da informação e das relações entre informação, fornecemos aos motores de inferência a possibilidade de criar conclusões lógicas baseadas no modelo definido. A utilização de motores de inferência na Web Semântica permite às aplicações perguntarem o porquê de se ter chegado a uma determinada conclusão, ou seja, as aplicações semânticas conseguem dar provas das conclusões a que chegaram. As provas indicam ou explicam os passos tomados para chegar a uma determinada conclusão lógica.

Actualmente existem muitos motores de inferência disponíveis. Alguns exemplos são:

- **Jena Reasoner:** Inclui um motor de inferência genérico baseado em regras juntamente com um conjunto de regras configuradas para RDFS e para OWL. É uma "framework" "Open Source" em Java para implementação de aplicações para a Web Semântica. Foi desenvolvido pelos Laboratórios HP (Jena, 2005).

- **Jess:** Utilizando a "framework" Jess (Gandon e Sadeh, 2003) é possível implementar software Java com a capacidade de raciocínio utilizando conhecimento disponibilizado sobre a forma de regras declarativas. O Jess é uma "framework" "leve" e é um dos motores de regras mais rápido disponível. Foi desenvolvido pela Universidade Carnegie Melon.

- **SWI-Prolog Semantic Web Library:** Prolog é uma linguagem natural para trabalhar com RDF e OWL. Foi implementado pela equipa que criou o SWI-Prolog, um conjunto de ferramentas para a criação e edição de aplicações RDF e OWL, incluindo um pacote que permite a utilização do motor de raciocínio (Wielemaker, 2005).

- **FaCT++:** Este é um motor de raciocínio baseado na lógica de descrição. É a reimplementação do motor de raciocínio FaCT. Permite o raciocínio sobre a linguagem OWL (FaCT, 2005).

- **Racer:** É também um motor de inferência baseado na lógica de descrição. Consegue ler e raciocinar sobre bases de conhecimento definidas nas linguagens DAML+OIL e OWL, estando estas bases de conhecimentos localizadas em ficheiros locais ou em Sites remotos. O motor Racer está

disponível gratuitamente para projectos de investigação e pode ser acedido através dos protocolos HTTP ou TCP. É um dos mais rápidos motores de raciocínio para bases de conhecimentos definidas na linguagem OWL DL (Haarslev e Möller, 2003).

Confiança

A confiança é o nível mais alto da arquitectura semântica. Este nível disponibiliza autenticação de identidade e a evidência de confiança nos dados e nos serviços. Ao contrário dos outros níveis da Web Semântica, que têm recebido muita atenção por parte da comunidade científica, o nível da confiança tem merecido pouca atenção. A ideia neste nível é permitir às pessoas fazerem questões sobre a confiança da informação disponibilizada na Web. Por exemplo, deverá ser possível definir que se tem confiança na informação proveniente de terminados Web Sites ou Serviços Web e que não temos confiança noutros Web Sites ou Serviços Web específicos (Cardoso, 2005).

2.4 Trabalhos Relacionados

A integração de informação é um problema muito comum. Existem muitos trabalhos que abordam a integração de informação. Muitos dos trabalhos mais recentes utilizam as tecnologias semânticas. Nesta secção apresentamos alguns exemplos de trabalhos realizados na área da integração de informação, e que recorrem à tecnologia semântica para resolver o problema da integração.

2.4.1 TDS Biological Modeler

TDS Biological Modeler (Teranode, 2007) é uma aplicação colaborativa para análises biológicas. Permite a integração de diferentes modelos de dados de forma a obter informação agregada para análise científica. Utiliza a linguagem RDF para definir relações entre a informação proveniente de diferentes modelos de dados. Os modelos de dados são definidos através da linguagem OWL formando Ontologias. Para a pesquisa de conhecimento, utiliza a linguagem de query SPARQL (Prud'hommeaux e Seaborne, 2007). É possível definir queries em SPARQL de forma a obter informação referente a diferentes Ontologias presentes no sistema. É também utilizada a linguagem SWRL (Horrocks, Patel-Schneider, Boley, Tabet, Grosof e Dean, 2004) para a criação de regras que permitem a tradução de elementos entre Ontologias. A arquitectura utilizada por esta aplicação encontra-se representada na figura 2.9.

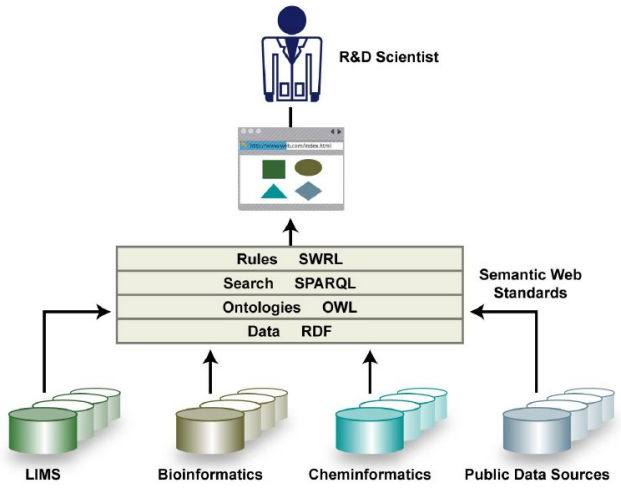

Figura 2.9 – Arquitectura da aplicação TDS Biological Modeler (Teranode, 2007).

2.4.2 Arquitectura de Integração no Domínio da Saúde

Um grupo de investigadores da universidade METU (Middle East Technical University) desenvolveu uma arquitectura para integração de sistemas na área da saúde (Bicer, Laleci, Dogac e Kabak, 2005). O principal objectivo é a criação de um sistema que permita a interoperabilidade entre os diversos sistemas existentes numa instituição de Saúde. Os sistemas a integrar são geralmente sistemas proprietários e servem apenas um departamento dentro da instituição.

O sistema desenvolvido, para resolver o problema da integração dos vários sistemas, utiliza tecnologias semânticas. Este sistema é utilizado para a transformação de mensagens no formato HL7 v2 para o formato HL7 v3 e vice-versa. HL7 (HL7, 2007) é um standard para a especificação de mensagens de comunicação entre sistemas no domínio da saúde. Entre a versão 2 e a versão 3 existem diferenças significativas que tornam necessário a implementação de um sistema mediador. O sistema implementado pode também ser utilizado como mediador entre outros tipos de modelos de informação que não os modelos HL7. A arquitectura do sistema é apresentada na figura 2.10.

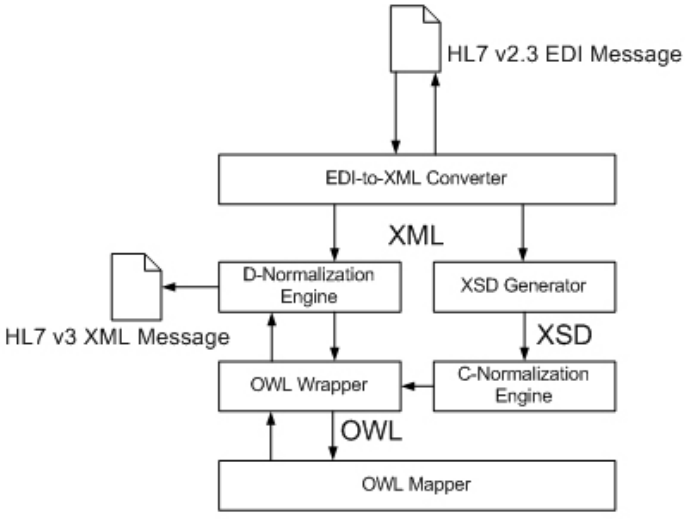

Figura 2.10 – Arq. de Integração de sist. de Saúde (Bicer, Laleci, Dogac e Kabak, 2005).

A arquitectura utilizada é composta pelos seguintes componentes:

- **OWL Ontology Mapping Tool (OWLmt):** É o componente principal do sistema e permite a criação de mapeamentos entre duas Ontologias diferentes. Os mapeamentos são utilizados para transformar as instâncias de uma das Ontologias mapeadas em instâncias da uma outra Ontologia

mapeada. Os mapeamentos são definidos utilizando uma interface GUI denominada OWLmt GUI.

- **EDI to XML Converter:** Componente utilizado para converter as mensagens no formato HL7 versão 2 em mensagens XML. As mensagens HL7 v2 são definidas através do standard EDI (Electronic Data Interchange).

- **XML Schema Generator:** Utilizado para obter o XML Schema referente a um determinado documento XML.

- **C-Normalization engine:** Permite obter o documento RDFS correspondente a um determinado documento XML Schema.

- **OWL Wrapper:** Componente utilizado para obter o documento OWL referente a um determinado documento RDFS.

- **D-Normalization engine:** Permite transformar instâncias XML em instâncias OWL e vice-versa. As transformações são realizadas através da consulta de um documento denominado de "Normalization Map" que define como cada componente XML deverá ser transformado num componente OWL e vice-versa.

2.4.3 Projecto COG

O projecto COG (Corporate Ontology Grid) (Alexiev, Breu, Bruijn, Fensel, Lara e Lausen, 2005) tem como objectivo a criação de um sistema para tornar possível a unificação da informação presente em fontes de dados heterogéneas, dentro de uma empresa. O projecto foi aplicado aos sistemas de produção da companhia automóvel Fiat. Para a unificação da informação é utilizado um modelo global, ou seja, uma Ontologia. São criados os modelos de dados referentes às fontes de dados a integrar no sistema. Cada um dos modelos de dados é mapeado com a Ontologia de forma precisa. Os mapeamentos permitem a localização da informação entre as aplicações existentes na empresa. O mapeamento permite também a transformação automática de queries definidas para a Ontologia em queries referentes às fontes de informação. Também é possível a transformação automática entre instâncias de dois diferentes modelos de dados que se encontrem mapeados com a Ontologia. O sistema implementado para a

integração da informação tem o nome de Unicorn Workbench. A arquitectura deste sistema é apresentada na figura 2.11.

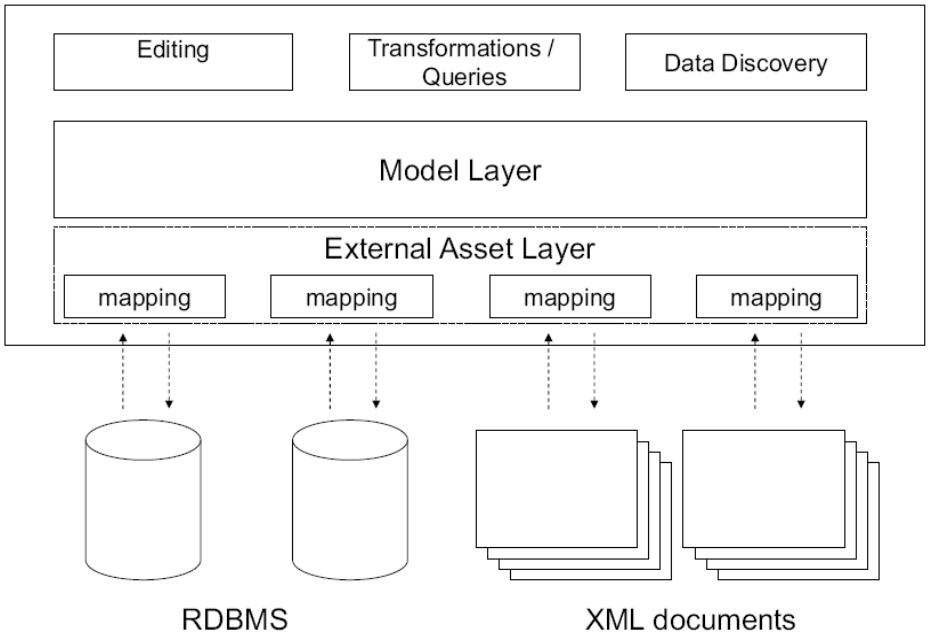

Figura 2.11 – Unicorn Workbench (Alexiev, Breu, Bruijn, Fensel, Lara e Lausen, 2005).

A arquitectura do sistema Unicorn Workbench é constituída por dois principais níveis:

- **External Assets Layer:** É o nível onde são definidos todos os mapeamentos com os modelos de dados da fontes de dados externas. Actualmente são suportadas fontes de dados referentes a bases de dados e a ficheiros XML. A integração de outros tipos de fontes de dados podem ser adicionadas, para isso é necessário implementar a integração utilizando uma API disponibilizada pelo sistema.

- **Model Layer:** É o nível onde se encontra a Ontologia. A Ontologia inclui a definição de toda a informação que reside nas fontes de dados a integrar.

Sobre este dois níveis foram implementados três módulos que disponibilizam três diferentes funções ao utilizador:

- **Edição:** Função que permite criar e manter a Ontologia e os mapeamentos do sistema.

- **Descoberta de dados:** Permite ao utilizador conhecer a localização dos dados ao nível das fontes de dados integradas.

- **Transformação e criação de queries:** Função que permite ao utilizador realizar transformações de instâncias entre diferentes modelos de dados e criação de queries sobre a Ontologia.

O sistema Unicorn Workbench segue a metodologia SIM (Semantic Information Management) (Schreiber, 2003). Esta metodologia encontra-se apresentada na figura 2.12.

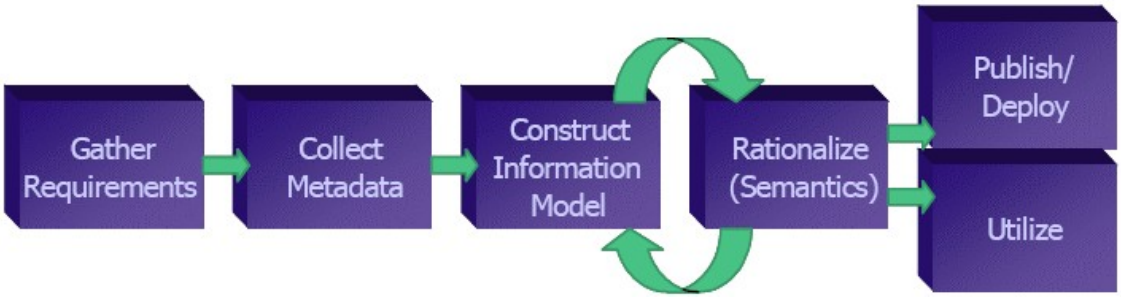

Figura 2.12 – Metodologia SIM (Schreiber, 2003).

A metodologia SIM segue claramente uma abordagem "Bottom-Up". Primeiro são recolhidos os modelos de dados dos sistemas a integrar ("Collect Metadata") e só depois é definida a Ontologia ("Construct Information Model").

2.4.4 Projecto ANOTA

O projecto ANOTA (Murua, Lladó e Llodrá, 2005) tem como objectivo a criação de um sistema que permita disponibilizar informação turística, proveniente de diferentes fontes, de forma agregada e de acordo com as pesquisas realizadas pelos utilizadores e de acordo com as estratégicas de marketing. A informação turística deverá ser disponibilizada através de um portal Web.

A integração das fontes de dados é realizada através da criação de "wrappers". "Wrappers" são componentes que permitem a extracção e formatação de dados existentes numa determinada fonte. Para cada uma das fontes a integrar é necessário a criação de um "wrapper". Cada um dos "wrappers" deverá transformar os dados formatados conforme o modelo de dados original em dados formatados segundo a Ontologia definida para o sistema. Os dados obtidos das fontes de dados externas deverão ser retornados utilizando a linguagem RDF. A Ontologia do sistema define o

domínio do turismo e foi baseada na especificação standard OTA (Open Travel Alliance) (Open Travel Alliance, 2007).

A responsabilidade da programação dos "wrappers" fica a cargo das empresas que pretendem disponibilizar os seus produtos turísticos através deste sistema. As empresas poderão registar as suas fontes de dados no sistema preenchendo um formulário próprio. Um administrador do sistema irá depois validar a informação disponibilizada pela empresa e irá enriquecê-la semanticamente, utilizando os conceitos definidos na Ontologia do sistema. Este processo de enriquecimento é denominado de anotação. O modelos das fontes de dados enriquecidos são guardados numa base de dados denominada "Sources directory". Quando um dos utilizadores do portal define uma pesquisa, o sistema terá de obter, através do "Sources directory", as fontes de dados que disponibilizam a informação correspondente à pesquisa indicada. O processo de obtenção da informação turística de forma agregada é apresentado na figura 2.13.

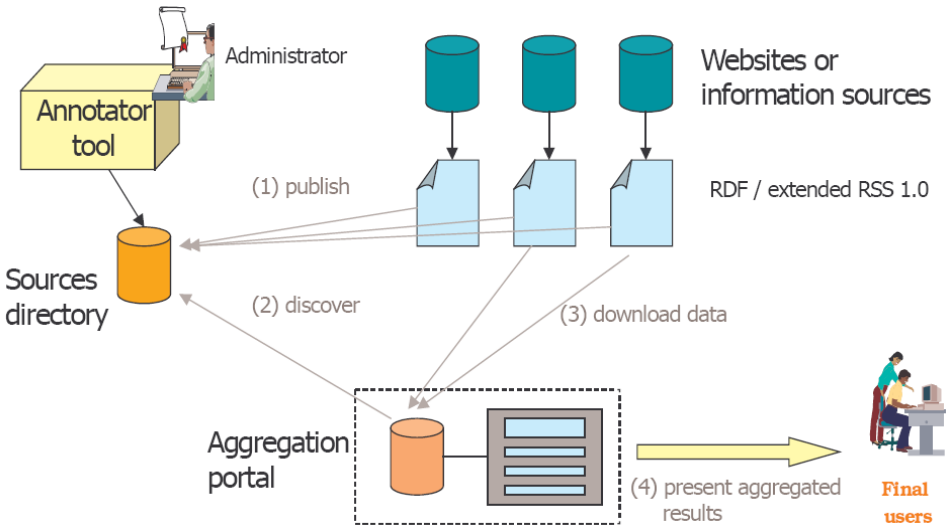

Figura 2.13 – Processo de pesquisa no projecto ANOTA (Murua, Lladó e Llodrá, 2005).

3 Definição da Arquitectura SEED

3.1 REQUISITOS

Nesta secção descrevemos a arquitectura para a integração de sistemas no domínio do turismo. A arquitectura tem o nome de SEED - SEmantic E-tourism Dynamic packaging. Esta arquitectura pretende resolver o problema da integração de informação turística proveniente de diferentes fontes de dados. Também pretende servir de base a sistemas que pretendam suportar o conceito de "Dynamic Packaging".

A arquitectura SEED foi definida de forma a suportar os seguintes requisitos:

- Integração de informação proveniente de diferentes tipos de fontes de dados. A arquitectura deverá suportar os tipos de fontes de dados mais comuns: Bases de Dados relacionais, ficheiros XML, Páginas Web e Web Services.

- Deverá permitir o suporte a outros tipos de fontes de dados através de uma extensão simples da arquitectura.

- Integração de informação com diferentes estruturas e diferentes semânticas. A informação deverá se integrada de forma a que as diferenças estruturais e semânticas sejam resolvidas. Toda a informação deverá ser traduzida num modelo de dados global único.

- O registo das fontes de dados a integrar deverá ser realizado de forma simples e rápida. Desta forma, será mais fácil convencer as entidades que gerem os sistemas turísticos a adicionarem a sua informação à arquitectura SEED.

- A arquitectura deverá ser simples de manter. Ou seja, sempre que haja alguma alteração nas fontes de dados, não deverá ser complicado nem demorado o processo de actualizar o sistema de forma a suportar tal alteração.

- Deverá permitir a obtenção da informação presente nas fontes de dados integradas sem que seja necessário conhecer os modelos de dados destas fontes. A pesquisa de informação deverá ser baseada apenas no modelo de dados global.

- A informação proveniente das fontes de dados deverá ser obtida em tempo real. Este requisito é necessário pois informação como disponibilidade e tarifas associadas aos produtos turísticos estão geralmente em constante actualização.

- A arquitectura deverá permitir a criação de regras de negócio. As regras deverão ser implementadas de forma a que seja possível alterá-las em tempo de execução. Este requisito é necessário para o suporte ao "Dynamic Packaging". As regras serão utilizadas para a definição dos pacotes dinâmicos e para a definição do cálculo do valor monetário para cada um dos pacotes. Desta forma é possível alterar a definição dos pacotes conforme as regras do mercado.

- A arquitectura deverá ser definida de forma a prever a futura extensão ao suporte de reservas de produtos turísticos e à implementação do conceito de "Dynamic Packaging".

3.2 Visão Geral da Arquitectura

A arquitectura SEED está dividida em quatro camadas. Por sua vez, cada uma das camadas é constituída por módulos. Na figura 3.1 apresentamos uma visão de alto nível da arquitectura.

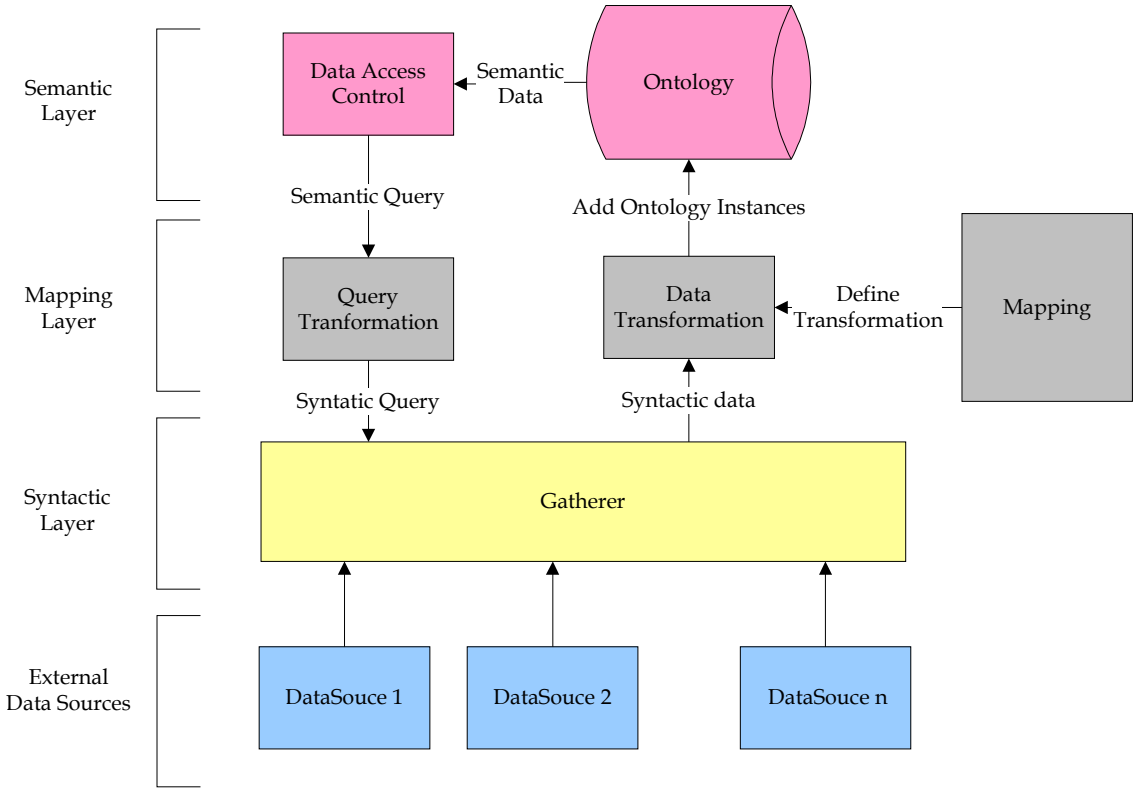

Figura 3.14 – Arquitectura SEED.

Numa visão por camadas a arquitectura fica dividida em quatro níveis:

- **External Data Sources (Fontes de Dados Externas):** Esta camada da arquitectura inclui todas as fontes de dados que serão integradas no sistema. A arquitectura permite a integração com fontes de dados do tipo HTML, ficheiros XML, bases de dados relacionais e Web Services.

- **Syntactic Layer (Nível Sintáctico):** Esta camada é inteiramente composta pelo módulo *Gatherer*. Este módulo trata da integração das fontes de dados heterogéneas. É responsável pela obtenção da informação através de uma determinada linguagem de query. É também responsável por retornar a informação num formato pré-definido.

- **Mapping Layer (Nível de Mapeamento):** É a camada onde a estrutura de dados sintáctica é mapeada com a estrutura de dados semântica. É no processo de mapeamento que adicionamos semântica ao modelo de dados sintáctico. É também nesta camada onde são processadas a transformações entre a camada sintáctica e a camada semântica. As queries semânticas são transformadas em queries sintácticas e os dados sintácticos são transformados em dados semânticos.

- **Semantic Layer (Nível Semântico):** É a camada superior da arquitectura, logo, é esta camada que trata de disponibilizar os métodos pelos quais as aplicações clientes poderão usar para interagir com a arquitectura. Esta camada também trata da definição das regras de negócio e pela aplicação destas regras à Ontologia definida para o sistema.

Existem dois fluxos de dados que atravessam estas camadas, é o fluxo das queries e o fluxo dos resultados. O fluxo das queries é inicializado na camada semântica. A aplicação cliente invoca um método disponibilizado pela camada semântica de forma a obter a informação desejada. A camada semântica trata de criar as queries semânticas correspondentes à pesquisa. As queries semânticas são depois transformadas em queries sintácticas, para depois serem transformadas em queries referentes a cada uma das fontes de dados mapeadas com a informação pretendida. O fluxo de resultados é iniciado na camada referente às fontes de dados. Primeiro, os dados vindos das fontes de dados são transformados de forma a respeitarem o modelo de dados sintáctico. Os dados sintácticos são depois transformados em dados semânticos. Finalmente, os dados semânticos são retornados à aplicação cliente que iniciou o pedido de informação. Estes dois fluxos de dados são mostrados na figura 3.2.

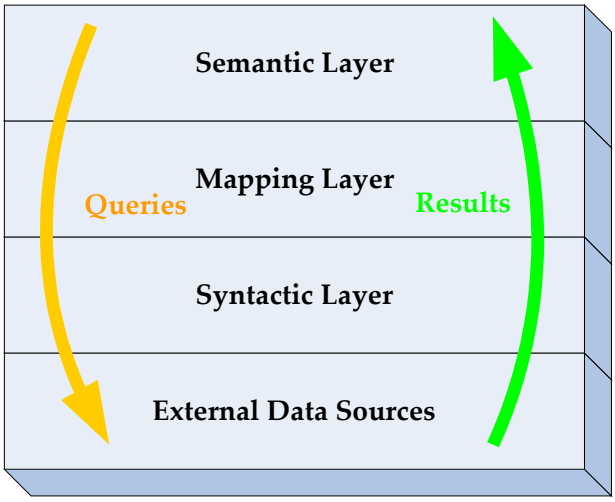

Figura 3.15 – Fluxos de dados da arquitectura SEED.

Cada uma das camadas anteriormente descritas é dividida em vários módulos. Na próxima secção iremos descrever cada um destes módulos.

3.3 Fontes de Dados Externas

O nível mais baixo da arquitectura refere-se às fontes de dados a serem integradas. É a partir destas fontes que a arquitectura irá obter os dados que deverão responder às pesquisas sobre informação turística.

As fontes de dados que disponibilizam informação turística são heterogéneas. Heterogéneas na forma como permitem o acesso à informação e heterogéneas no formato com que essa informação é retornada. De forma a considerar o maior número possível de fontes de dados, a arquitectura permite a integração com os tipos de fontes de dados mais comuns. Na figura 3.3 estão representadas os tipos de fontes de dados suportados pela arquitectura.

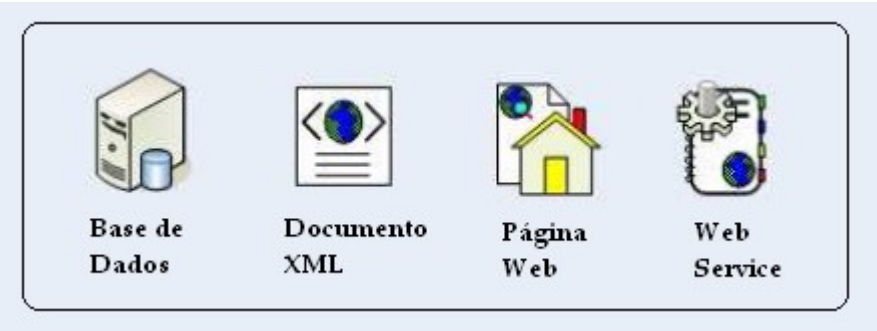

Figura 3.16 – Tipos de fontes de dados suportadas pela arquitectura SEED.

- **Base de Dados:** Inclui todas as bases de dados relacionais que permitam o acesso através da linguagem SQL.

- **Documento XML:** Documentos XML disponibilizados através de ficheiros locais ou através de acesso remoto através dos protocolos HTTP ou FTP.

- **Página Web:** Páginas Web definidas através da linguagem HTML e acessíveis através dos protocolos HTTP ou FTP.

- **Web Service:** Serviços Web que permitem comunicação através do protocolo SOAP. Deverão estar disponíveis através de pedidos HTTP.

3.4 Nível Sintáctico

Este nível é responsável pela extracção dos dados das fontes de dados externas. A integração é realizada através do mapeamento dos atributos das fontes de dados com um esquema de dados definido para a arquitectura. Partindo do princípio que toda a informação será adicionada a uma Ontologia, fazia todo o sentido que o esquema de dados a utilizar, para o mapeamento com as fontes de dados, fosse o modelo descrito pela Ontologia. No entanto esta não foi a nossa opção. Optamos antes por criar um modelo sintáctico intermédio. As fontes de dados são então mapeadas para um modelo sintáctico. O modelo sintáctico é depois mapeado com a Ontologia de forma a que toda a informação obtida das fontes de dados seja automaticamente adicionada à Ontologia. Na figura 3.4 apresentamos o esquema de mapeamentos entre os modelos existentes na arquitectura SEED.

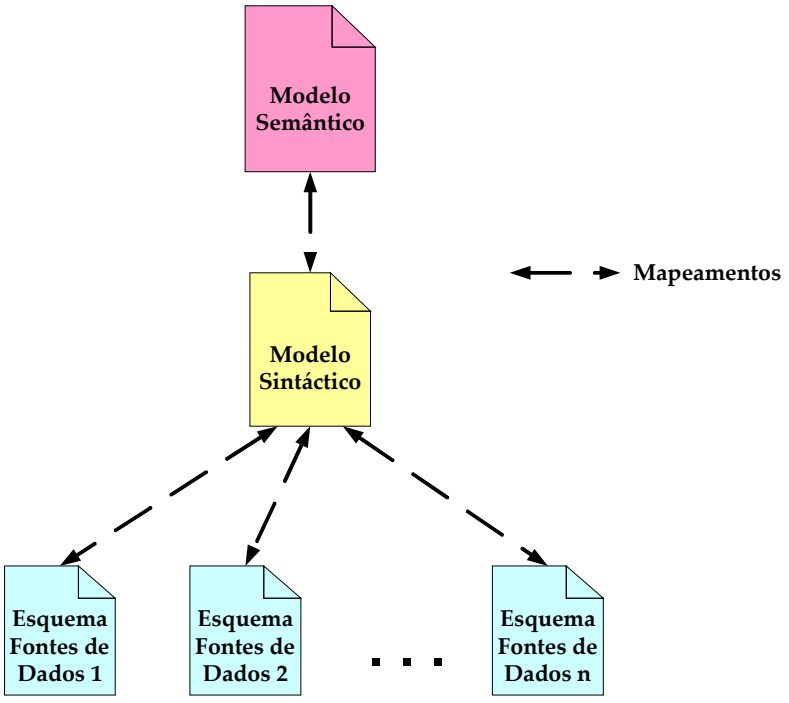

Figura 3.17 – Mapeamentos na Arquitectura SEED.

A opção pela utilização de um esquema sintáctico intermédio deve-se à pouca utilização de esquemas de dados semânticos nos sistemas actuais. Existe ainda pouco conhecimento sobre as estruturas de dados definidas através de linguagens que

permitem a adição de semântica aos dados. Tornando o mapeamento das fontes de dados com um esquema de dados que apenas define a sintaxe, leva a que o processo de mapeamento seja mais simples. Actualmente as linguagens sintácticas, como o XML, são muito utilizadas principalmente na resolução de problemas de integração de informação. É assim mais fácil encontrar pessoas capazes de definir o mapeamento entre as fontes de dados e o esquema sintáctico. O mapeamento entre o modelo sintáctico e o modelo semântico terá de ser realizado por alguém que tenha conhecimento na área das linguagens semânticas. No entanto, entre o modelo sintáctico e o modelo semântico temos de definir apenas um mapeamento, enquanto que para as fontes de dados temos de definir tantos mapeamentos quanto o número de fontes de dados existentes. A única desvantagem com a utilização de um modelo sintáctico intermédio prende-se com a necessidade de maior processamento. Maior processamento devido à existência de uma transformação de dados adicional entre o nível sintáctico e o nível semântico.

O nível sintáctico é composto apenas por um único módulo, o módulo *Gatherer*. O módulo *Gatherer* é depois subdividido em sub-módulos. A subdivisão em módulos encontra-se representada na figura 3.5.

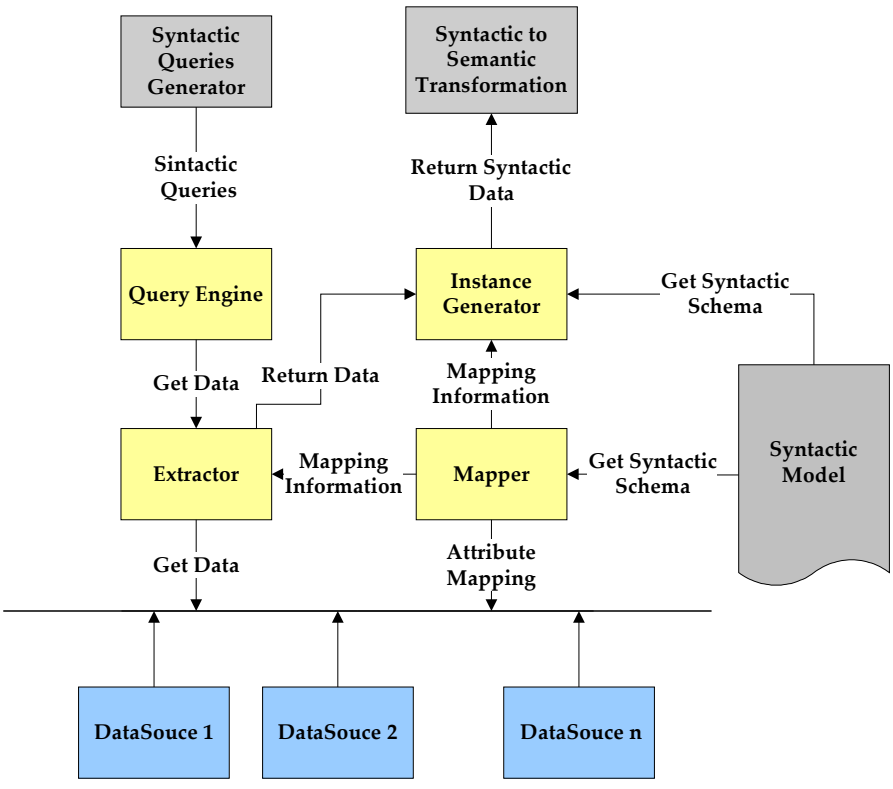

Figura 3.18 – Módulo *Gatherer*.

O módulo *Gatherer* é composto por quatro sub-módulos, *Extrator*, *Mapper*, *Query Engine* e *Instance Generator*. Estes sub-módulos são descritos de seguida.

3.4.1 Extrator

Extrator é o módulo responsável pela extracção dos dados das fontes de dados externas. Interage com o módulo *Mapper* para obter a informação sobre onde e como poderá obter os dados referentes aos itens definidos no modelo sintáctico. Trata também do acesso às fontes de dados de forma a extrair os dados necessários. É neste módulo que se encontra a informação referente ao acesso às fontes de dados. Associado a cada fonte de dados deverá existir a informação necessária para o acesso aos dados desta. Por exemplo, no caso de se tratar de uma fonte de dados do tipo Base de Dados teremos de possuir o URL de acesso ao servidor, o nome da base de dados e o utilizador e password de acesso à base de dados. Depois de extrair os dados das fontes de dados, este módulo trata de enviar os dados obtidos ao módulo *Instance Generator*.

3.4.2 Mapper

De forma a obter a informação das fontes de dados formatada conforme o modelo sintáctico definido, é necessário mapear os modelos de dados das fontes de dados externas com o modelo sintáctico. É o módulo *Mapper* que tem a responsabilidade de permitir este mapeamento. O mapeamento é realizado através da definição de atributos de mapeamento. A definição dos atributos de mapeamento é realizada através da acção humana. É o utilizador que define quais os mapeamentos existentes entre os atributos das fontes de dados e os atributos referentes ao modelo sintáctico. A definição de um atributo de mapeamento é realizada em três passos:

1. **Atribuir um nome ao atributo:** o nome do atributo irá identificar qual o atributo do modelo sintáctico que estamos a mapear.

2. **Definir regra de extracção:** A regra de extracção é um conjunto de código que permite obter os dados existentes numa determinada fonte de dados e necessários para preencher um determinado atributo. Associado à regra de extracção deverá existir a identificação da fonte de dados sobre a qual iremos obter os dados. A linguagem a ser utilizada para a definição das regras de extracção irá depender do tipo da fonte de dados em questão. Por exemplo, se a fonte de dados for do tipo

Base de Dados, então deveremos utilizar a linguagem SQL para a definição das regras de extracção. Associado a cada uma das regras deverá estar a identificação da fonte de dados de onde serão extraídos os dados

3. **Associar a regra de extracção ao atributo:** Para que a definição do atributo de mapeamento esteja finalizada é necessário associar a regra de extracção ao atributo. Depois desta associação o sistema sabe exactamente que código deverá executar para obter os dados referentes a um determinado atributo do modelo sintáctico.

O módulo *Mapper* obtém informação das fontes de dados e do modelo sintáctico para obter as estruturas de dados a mapear. Interage com os módulos *Extractor* e *Instance Generator* de forma a retornar a informação de mapeamento.

3.4.3 Query Engine

O módulo *Gatherer* disponibiliza uma linguagem de query de forma a permitir que os módulos cliente possam definir os seus pedidos de informação. Ao definirmos a interacção com este módulo através de uma linguagem de query, tornamos este módulo mais independente do resto da arquitectura. Se um dia pretendermos substituir este módulo, desde que o novo módulo suporte a linguagem de query, não será necessário alterar o resto da arquitectura. A desvantagem desta opção prende-se com o aumento de complexidade e de processamento na integração com os outros módulos.

É o módulo *Query Engine* que trata de processar os pedidos definidos na linguagem de query. A linguagem de query baseia-se na linguagem SQL. É constituída por uma primeira parte onde são indicados os elementos a retornar. Esta primeira parte é definida pela palavra "SELECT". É também constituída por uma segunda parte, marcada pela palavra "WHERE", onde são definidos os filtro a considerar. A sintaxe da linguagem é a seguinte:

```
SELECT <element>[1..*]
WHERE <attribute><operator><constraint>
       AND/OR <attribute><operator><constraint>[1..*]
```

Onde "element" refere-se a um elemento definido no modelo sintáctico, "attribute" refere-se a um atributo pertencente ao elemento. O item "operator" representa um

operador lógico e o item "constraint" um valor que irá limitar os possíveis valores para o atributo indicado.

Para além de processar a linguagem de query, o módulo *Query Engine* também tem a responsabilidade de realizar os pedidos de extracção ao módulo *Extractor*. Só são permitidos pedidos ao nível dos elementos, ou seja, não podemos pedir os dados referentes a apenas um determinado atributo de um elemento. Esta limitação permite tornar o processamento da linguagem de query mais simples. Grande parte do custo de obtenção dos dados está na acção de conexão às fontes de dados. Mais vale agir por antecipação e obter todos os dados referentes a um determinado elemento, que obter os dados para parte do elemento, e na seguinte interacção ter que obter os restantes dados para o mesmo elemento.

3.4.4 Instance Generator

O módulo *Instance Generator* tem a responsabilidade de formatar os dados obtidos conforme o modelo sintáctico definido. Obtém os dados retornados pelo módulo *Extractor*. Para cada unidade de informação obtida procura qual o atributo de mapeamento associado. Coleccionado os vários valores de atributos retornados, vai construindo uma instância de dados referente ao modelo sintáctico. A instanciação criada é retornada como resposta à query inicialmente enviada ao módulo *Gatherer*.

3.5 Nível de Mapeamento

Toda a informação obtida através do módulo *Gatherer* deverá ser adicionada à Ontologia definida para o sistema. Para isso é necessário transformar as instâncias referentes ao modelo sintáctico em instâncias da Ontologia. Para além desta transformação, também é necessário transformar as queries provenientes do nível semântico em queries compreendidas pelo módulo *Gatherer*. Em resumo, este nível tem a função de servir de mediador entre o nível semântico e o nível sintáctico.

O nível de mapeamento é composto por três módulos principais. Dois deles são responsáveis pelos processos de transformação existentes entre o nível semântico e o nível sintáctico. O módulo *Query Transformation* trata da transformação das queries semânticas em queries sintácticas. Por sua vez, o módulo *Data Transformation* trata da transformação dos dados sintácticos em dados semânticos. O outro módulo, o módulo *Mapping*, é responsável pela definição do mapeamento entre os modelos de dados sintácticos e semânticos. Cada um destes módulos será detalhado de seguida.

3.5.1 Mapping

O módulo *Mapping* trata de disponibilizar as funções que permitem a definição do mapeamento entre o modelo semântico e o modelo sintáctico. Através do mapeamento são definidas regras de mapeamento que depois são utilizadas para a definição do processo de transformação entre os dados sintácticos e os dados semânticos. Existem três diferentes estratégias de mapeamento. Estas distinguem-se pelo grau de interacção necessária por parte do utilizador. As três estratégias são:

- **Mapeamento manual:** No mapeamento manual o utilizador tem de definir o mapeamento completo entre os dois modelos. Geralmente existe uma interface gráfica que auxilia a definição dos mapeamentos. Actualmente, a maior parte das ferramentas de mapeamento utilizam esta estratégia.

- **Mapeamento semi-automático:** Nesta estratégia de mapeamento o sistema apresenta sugestões de mapeamento ao utilizador. O utilizador pode ou não aceitar as sugestões.

- **Mapeamento automático:** Neste caso o mapeamento é realizado sem a intervenção humana. O mapeamento é realizado apenas através de um processo automático.

Os mapeamentos automáticos permitem eliminar a intervenção humana na acção de definição dos mapeamentos. Infelizmente os sistemas que permitem o mapeamento automático ainda não garantem uma eficácia de 100%. O mapeamentos semi-automáticos permitem automatizar parte do processo, além disso, dão a possibilidade ao utilizador de corrigir os mapeamentos que foram definidos incorrectamente. No entanto, os sistemas semi-automáticos podem tornar a verificação dos mapeamentos pouco rigorosa por parte do utilizador, levando à existência de erros de mapeamento. A estratégia referente ao mapeamento manual obriga o utilizador a definir todos os mapeamentos. Apesar de ser um processo mais moroso, torna-se mais eficaz.

Na nossa arquitectura optamos por utilizar uma estratégia de mapeamento manual. Escolhemos esta estratégia pois pretendemos garantir que não existe nenhum erro no processo de mapeamento. Assim, o utilizador terá de definir todos os mapeamentos entre os elementos do modelo semântico e os elementos do modelo sintáctico. A definição de um mapeamento passa por definir em primeiro lugar a associação entre os elementos de cada modelo. Depois, para cada associação entre elementos deveremos definir as associações entre as propriedades de cada elemento. Para cada associação entre elementos é ainda necessário indicar qual o identificador que irá distinguir os elementos no modelo semântico. A identificação de um elemento no modelo semântico é definida através de uma ou mais propriedades sintácticas mapeadas com este elemento. As propriedades sintácticas que irão compor o identificador de cada elemento semântico são escolhidas pelo utilizador e na altura em que define o mapeamento.

Os identificadores e as associações definem as regras de mapeamento entre os dois modelos. As regras de mapeamento são guardadas de forma a permitir no futuro a edição dos mapeamentos definidos. As regras de mapeamento também são utilizadas para a definição do processo de transformação das as instâncias do modelo sintáctico

em instâncias do modelo semântico. O módulo *Mapping* tem a responsabilidade de definir as regras de mapeamento e de definir o processo de transformação. No entanto, a acção de transformação dos dados está a cargo do módulo *Data Transformation*.

Depois de definidos os mapeamento, o módulo *Mapping* gera dois documentos, que serão depois utilizados por outros módulos da arquitectura. Os dois documentos são:

- **Regras de mapeamento:** Documento que inclui as definições de todos os mapeamentos criados. É utilizado para a edição futura dos mapeamentos criados e para ser utilizado no processo de transformação das queries sintácticas em queries semânticas.

- **Documento de transformação:** Documento que define a transformação dos dados do modelo sintáctico em dados do modelo semântico. É utilizado no processo de transformação de dados.

3.5.2 Data Transformation

O módulo *Data Transformation* tem a responsabilidade de transformar os dados sintácticos em dados semânticos. Os dados sintácticos deverão respeitar o modelo sintáctico definido, e os dados semânticos deverão respeitar o modelo semântico definido. O processo de transformação é realizado através da utilização do documento de transformação gerado pelo módulo *Mapping*. O processo de transformação é realizado durante o tempo de execução do sistema.

Este módulo também tem a responsabilidade de adicionar as instâncias referentes ao modelo semântico, e obtidas através do processo de transformação, à Ontologia definida para o sistema. Antes de adicionar uma instância à Ontologia é necessário analisar se esta respeita as regras da Ontologia e se esta já existe na Ontologia. Se a instância não estiver conforme as regras da Ontologia é ignorada e não é adicionada à Ontologia. Se a instância já existir na Ontologia não é adicionada. Neste caso, existe uma actualização dos valores das propriedades da instância que se encontra na Ontologia. São sempre considerados os valores das propriedades obtidas mais recentemente. Para verificar se uma instância já existe na Ontologia, comparamos o identificador da instância a adicionar com os identificadores das instâncias existentes na Ontologia. Os identificadores das instâncias do modelo semântico são definidos no módulo *Mapping*. O processo de obtenção dos identificadores fica registado no

documento de transformação. O identificador para cada instância semântica é criado durante o processo de transformação dos dados sintácticos em dados semânticos.

O módulo *Data Transformation* é subdividido em sub-módulos. Estes sub-módulos estão representados na figura 3.6.

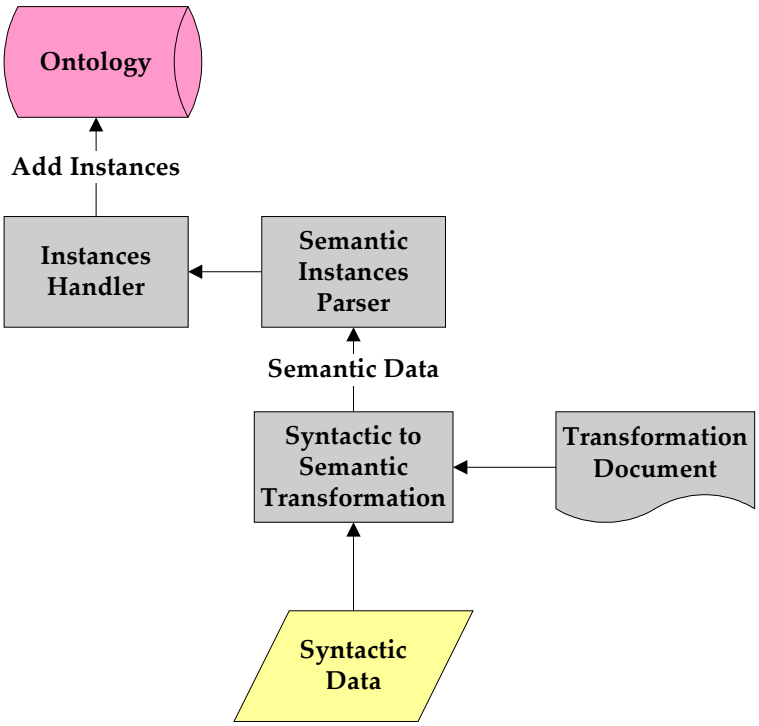

Figura 3.19 – Módulo *Data Transformation*.

Os sub-módulos do módulo *Data Transformation* têm as seguintes responsabilidades:

- ***Syntactic to Semantic Transformation*:** Tem a responsabilidade de transformar os dados sintácticos em dados semânticos, obtendo os dados sintácticos através do módulo *Gatherer*. Posteriormente, utiliza o documento de transformação gerado pelo módulo *Mapping* e transforma os dados em instâncias da Ontologia.

- ***Semantic Instances Parser*:** Trata de processar as instâncias obtidas do módulo *Syntactic to Semantic Transformation*. Para isso são criadas estruturas próprias para incluírem as instâncias obtidas e para facilitar o seu processamento. As estruturas estão representadas na figura 3.7.

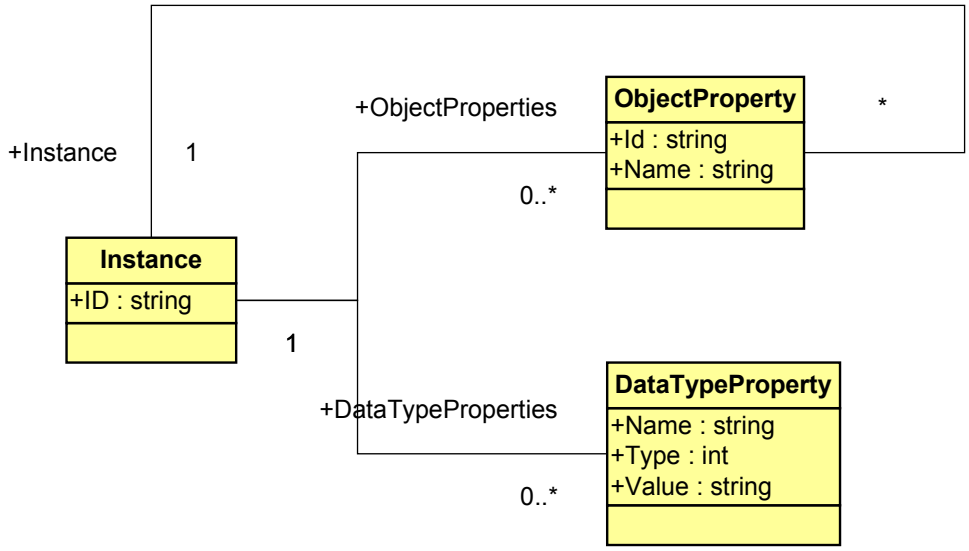

Figura 3.20 – Estrutura de dados para as instâncias da Ontologia.

A classe "Instance" representa uma instância do modelo semântico que será adicionada à Ontologia. A classe "DataTypeProperty" define as propriedades da instância que contêm tipos simples. A classe "ObjectProperty" representa as propriedades que têm como tipo uma classe. A classe "ObjectProperty" está associada a uma instância da Ontologia que representa o valor tomado pela propriedade.

- ***Instances Handler*:** Tem a responsabilidade de adicionar as instâncias à Ontologia. Contém uma colecção de instâncias incluídas na estrutura definida no módulo *Semantic Instances Parser*. Trata também da validação das instâncias perante as regras da Ontologia e da verificação da existência das instâncias na Ontologia.

3.5.3 Query Transformation

O módulo *Query Transformation* tem a responsabilidade de transformar as queries semânticas em queries sintácticas. As queries semânticas são criadas no nível semântico para depois serem enviadas para o nível de mapeamento. As fontes de dados registadas no sistema não compreendem as queries semânticas. Assim, é necessário transformar estas queries de forma a serem compreendidas. É o módulo *Gatherer* que trata da integração das fontes de dados. Este módulo disponibiliza a linguagem de query S2SQL que permite obter os dados das fontes de dados registadas no sistema. Assim, as queries semânticas deverão ser transformadas em queries sintácticas definidas através da linguagem S2SQL.

De forma a simplificar o processo de transformação de queries, a linguagem de query semântica foi limitada. A linguagem de query semântica só permite obter dados referentes a uma única classe da Ontologia. Além disso, só são permitidos filtros que utilizam restrições simples sobre as propriedades das classes. Também não é possível indicar a obtenção de apenas parte da informação referente a uma classe da Ontologia. É sempre retornada toda a informação associada à classe pedida. Novas funcionalidades de query poderão ser adicionadas futuramente. No entanto, temos de ter sempre presente que quanto maior complexidade adicionarmos à linguagem de query mais complexo será o processo de transformação das queries.

O processo de transformação das queries utiliza o documento de regras de mapeamento definido no módulo *Mapping*. Através das regras de mapeamento, o módulo *Query Transformation* cria as queries sintácticas correspondentes às queries semânticas em questão. A transformação é quase directa pois cada um dos elementos do modelo semântico poderá estar mapeado a um ou mais elementos do modelo sintáctico. Quando não existe mapeamento com a classe semântica utilizada na query, não existe transformação da query semântica. Nestes casos não será obtida informação de resposta à query. O processo de transformação de queries torna-se mais complexo quando existe a definição de herança associada à classe utilizada na query. Quando uma query semântica inclui classes que são super classes de outras, é necessário que o processo de transformação de queries tenha este facto em conta. O processo de transformação deverá transformar as queries semânticas em queries sintácticas de forma a que consigam obter todos os dados necessários para que as queries semânticas sejam executadas com sucesso. Assim, no caso de pesquisa de uma super classe, o processo de transformação deverá obter todos os elementos mapeados com a classe pesquisada e todos os elementos mapeados com as subclasses da super classe em questão. Outras das situações em que o processo de transformação de queries torna-se complexo é quando pesquisamos por uma classe semântica que não tem mapeamento directo com o modelo sintáctico. Neste caso, o processo de transformação tem de criar as queries sintácticas que obtenham os dados referentes aos elementos mapeados com a super classe da classe pesquisada. Este comportamento acontece porque o sistema prevê que exista alguma regra semântica que defina a subclasse em questão. As regras semânticas são geralmente utilizadas para definir determinadas classes a partir das suas super classes. Por exemplo, na Ontologia podemos definir uma classe "Hotel" e uma subclasse desta com o nome "HotelLuxo". A classe "Hotel" pode ser mapeada

directamente ao modelo sintáctico. Por sua vez, a classe "HotelLuxo" poderá não conter qualquer mapeamento com o modelo sintáctico. No mesmo exemplo é criada uma regra semântica que diz que todos os hotéis com 4 ou mais estrelas são um hotel de luxo. Quando pesquisamos pela classe "HotelLuxo", o processo de transformação de queries cria uma query sintáctica para obter todos os hotéis. Aplicando a regra definida descartamos todos os hotéis com menos de 4 estrelas. Desta forma conseguimos obter os dados referentes aos hotéis de luxo.

O módulo *Query Transformation* é composto por um conjunto de sub-módulos, estes sub-módulos estão representados na figura 3.8.

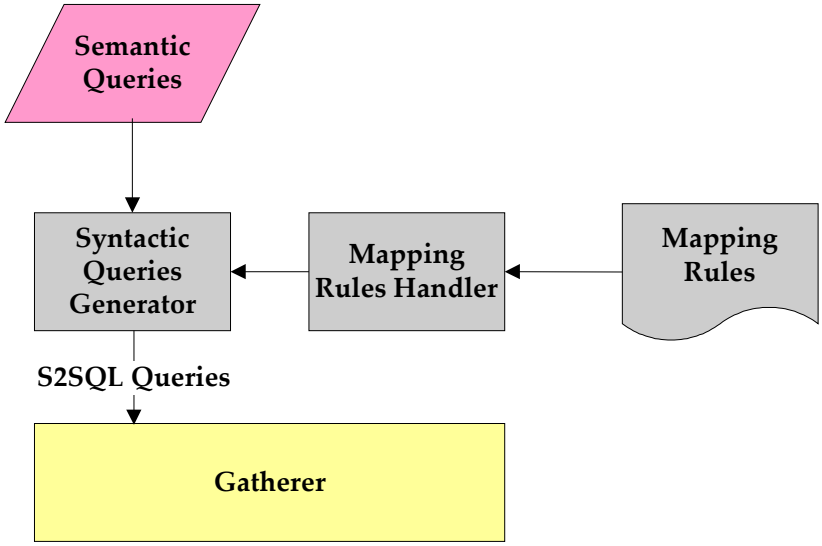

Figura 3.21 – Módulo *Query Transformation*.

Cada um dos módulos apresentados é descrito de seguida:

- ***Mapping Rules Handler*:** Este módulo tem a responsabilidade de carregar as regras de mapeamento numa estrutura específica. Esta estrutura irá facilitar o processamento das regras de mapeamento. A estrutura utilizada para suportar as regras de mapeamento é apresentada na figura 3.9.

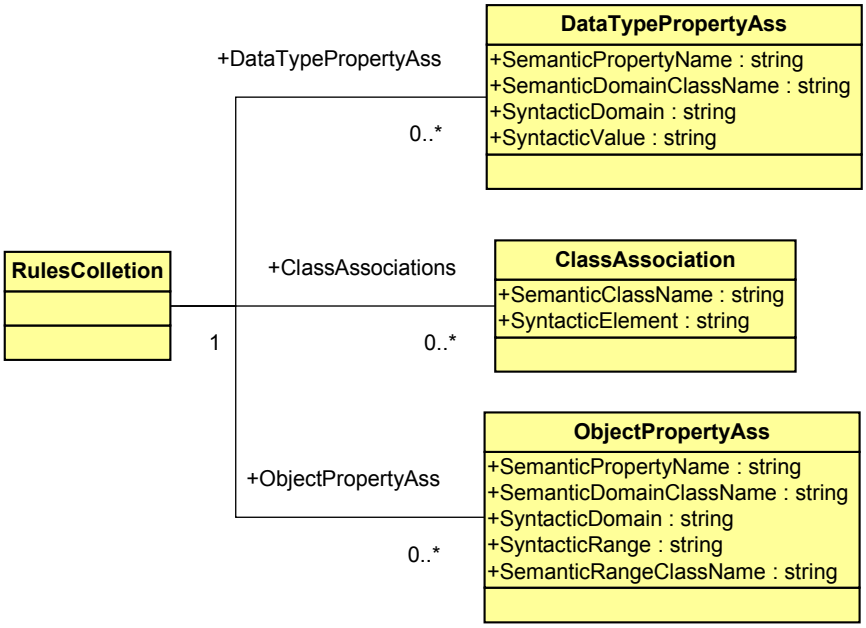

Figura 3.22 – Estrutura de dados para as Regras de Mapeamento.

A classe "RulesCollection" representa a colecção de regras de mapeamento definidas para o sistema. As classes "ClassAssociation", "ObjectPropertyAss" e "DataTypePropertyAss" representam os diversos tipos de mapeamento que são possíveis definir. Podemos definir mapeamentos entre Elementos sintácticos e classes semânticas ("ClassAssociation"), mapeamentos entre propriedades de tipos simples dos elementos e das classes ("DataTypePropertyAss") e mapeamentos entre propriedades com tipos complexos, tais como elementos sintácticos ou classes do modelo semântico ("ObjectPropertyAss").

- ***Syntactic Queries Generator*:** Este módulo tem a função de transformar as queries semânticas em queries sintácticas. As queries sintácticas têm de ser criadas de forma a obter todos os dados necessários à execução correcta das queries semânticas. O módulo *Syntactic Queries Generator* interage com o módulo *Mapping Rules Handler* para obter as regras de mapeamento e com o módulo *Gatherer* de forma a enviar as queries sintácticas. As queries sintácticas são enviadas para o módulo *Gatherer* definidas na linguagem S2SQL.

3.6 Nível Semântico

É no nível semântico que é definida a Ontologia, e consequentemente onde definimos todos os conceitos a serem considerados no sistema. A Ontologia é composta pelas classes, pelas propriedades das classes e pelas relações entre as classes definidas. Através da Ontologia também é possível a definição de restrições sobre as propriedades e regras semânticas, utilizando as classes e respectivas propriedades. Todas as instâncias adicionadas à Ontologia deverão respeitar as restrições e regras definidas.

Para além da Ontologia, o nível semântico tem também a responsabilidade de disponibilizar as funções que permitem manipular os conceitos existentes na Ontologia. Disponibiliza funções que permitem obter informação da Ontologia e que permitem a criação e activação/desactivação de regras semânticas.

Os módulos que compõem o nível semântico estão representados na figura 3.10.

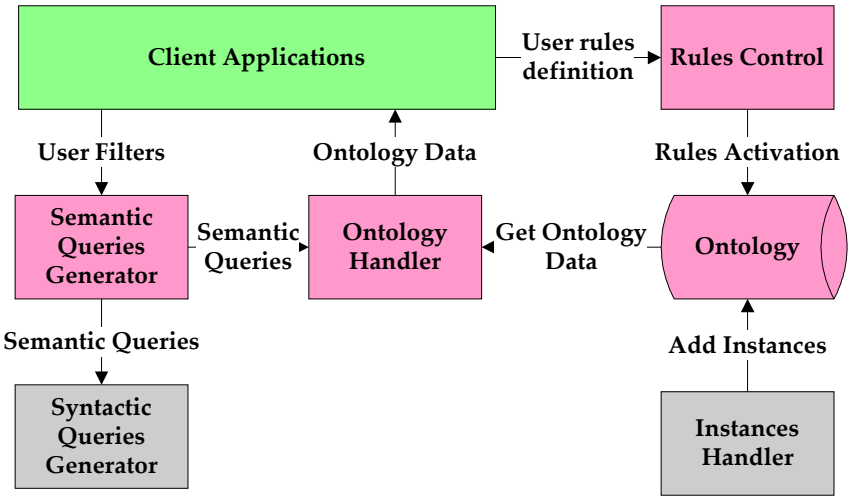

Figura 3.23 – Módulos do Nível Semântico.

Cada um dos módulos que constituem o nível semântico é explicado de seguida.

3.6.1 Ontology Handler

Este módulo tem a responsabilidade de obter a informação que se encontra na Ontologia definida para o sistema. Os pedidos de informação são enviados pelo

módulo *Semantic Queries Generator* na forma de queries semânticas. O módulo *Ontology Handler* processa as queries semânticas e obtém os dados referentes a estas. Os dados obtidos são retornados às aplicações clientes da arquitectura. As aplicações clientes representam as aplicações que poderão utilizar a arquitectura para a integração de fontes de dados. Na figura 3.10 estas são representadas pelo módulo *Client Applications*.

3.6.2 Semantic Queries Generator

Semantic Queries Generator é o módulo que cria as queries semânticas referentes aos pedidos realizados pelas aplicações clientes. Os pedidos de informação são realizados através de uma estrutura de dados própria. O módulo *Semantic Queries Generator* tem a função de transformar esta estrutura de dados em queries semânticas. As queries semânticas são depois enviadas para os módulos *Syntactic Queries Generator* e *Ontology Handler*. Através do módulo *Syntactic Queries Generator* as queries são transformadas em queries sintácticas de forma a obtermos os dados referentes às fontes de dados externas. Depois de várias transformações, os dados obtidos são adicionados à Ontologia na forma de instâncias. Depois de termos as instâncias adicionadas à Ontologia, as queries semânticas são aplicadas à Ontologia através do módulo *Ontology Handler*. Seguindo este processo, permite-nos utilizar toda a expressividade que poderá ser definida através da Ontologia.

A estrutura de dados disponibilizada pelo módulo *Semantic Queries Generator* para a realização dos pedidos de informação é apresentada na figura 3.11.

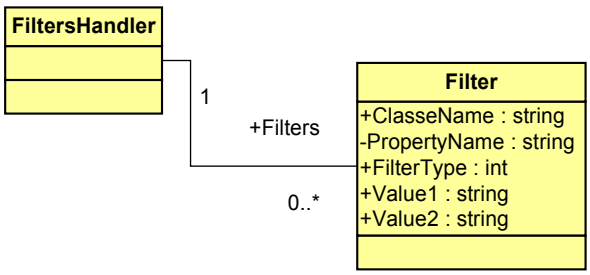

Figura 3.24 – Estrutura de dados para definição de pedidos de informação.

A classe "FiltersHandler" representa um conjunto de filtros criados para a obtenção de informação referente a uma ou mais classes da Ontologia. A Classe "Filter" representa um filtro específico sobre uma determinada classe da Ontologia.

3.6.3 Rules Controls

Este módulo é responsável pela definição das regras de negócio que poderão ser criadas no sistema. As regras de negócio poderão ser utilizadas, por exemplo, para definirmos o que é um hotel de luxo. Poderemos criar uma regra que indique que um hotel de luxo deverá ter 4 ou mais estrelas. Para ser possível a definição desta regra é necessário que na nossa Ontologia estejam representadas as classes "Hotel" e "HotelLuxo". As regras são sempre baseadas em conceitos existentes na Ontologia. A definição da classe "HotelLuxo" poderia claramente ser definida na própria Ontologia e sem o recurso às regras de negócio. No entanto, a vantagem na criação de uma regra está no facto desta poder ser alterada ou activada/desactivada enquanto o sistema está em execução. Ou seja, as regras de negócio poderão ser alteradas em tempo de execução.

O módulo *Rules Control* interage com a Ontologia para a activação e verificação das regras. Interage também com as aplicações clientes, recebendo as regras definidas nestas aplicações. Para a definição das regras é utilizada a classe apresentada na figura 3.12.

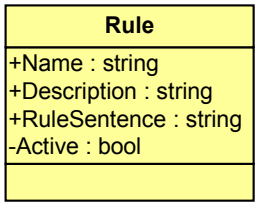

Figura 3.25 – Estrutura de dados para definição de regras de negócio.

A classe "Rule" representa uma regra de negócio que poderá estar activa ou desactiva.

3.7 METODOLOGIA DE INTEGRAÇÃO

A arquitectura aqui descrita pode ser configurada sobre diferentes abordagens. As abordagens definem qual o procedimento para a criação dos modelos de dados para a integração. Existem três abordagens principais: abordagem "Top-Down", abordagem "Bottom-Up" e abordagem Mista. Cada uma das abordagens tem as suas vantagens e desvantagens, pelo que a escolha da melhor abordagem irá depender das características do sistema a implementar. Cada uma das abordagens será descrita de seguida.

3.7.1 Abordagem "Top-Down"

Na abordagem "Top-Down" começa-se por definir a Ontologia para então depois definir o modelo sintáctico do sistema. A Ontologia deverá ser criada por especialistas no domínio referente ao sistema a implementar. Na Ontologia deverão ser definidos todos os conceitos a serem utilizados pelo sistema. Nesta abordagem o modelo sintáctico é baseado na Ontologia criada.

A vantagem desta abordagem está no facto da Ontologia ser criada de forma a definir o domínio do problema e não criada baseada nas estruturas de dados das fontes de dados externas. As estruturas de dados das fontes de dados externas geralmente são criadas para uma determinada aplicação, contendo informação necessária apenas para essa aplicação. Uma Ontologia que define o domínio e que é independente de qualquer aplicação específica, permite uma mais fácil integração de novas fontes de dados. A desvantagem de utilizar esta abordagem está na dificuldade de criação das Ontologias de forma a definirem um determinado domínio. É necessário encontrar pessoas especialistas no domínio. Além disso, é difícil definir uma Ontologia que consiga obter a aprovação de todos os especialistas no domínio. Outra das desvantagens está na maior complexidade existente na integração das fontes de dados. Como a Ontologia foi criada sem serem consideradas as fontes de dados a integrar, é possível encontrarmos pontos de divergência entre a Ontologia e os modelos de dados das fontes externas. Esta abordagem deverá ser utilizada quando se prevê a integração de novas fontes de dados num futuro próximo. Também devemos utilizar esta abordagem quando

pretendemos utilizar modelos de domínio próprios, que se distinguem dos existentes nas fontes de dados a integrar. Na figura 3.13 apresentamos a ordem de acções a realizar quando escolhemos uma abordagem "Top-Down".

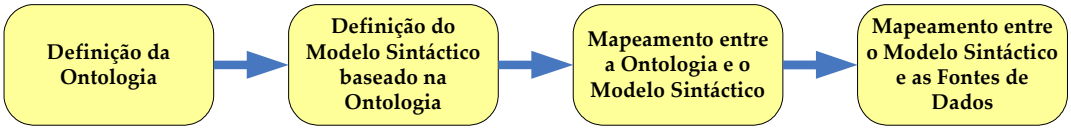

Figura 3.26 – Abordagem "Top-Down".

3.7.2 Abordagem "Bottom-Up"

Na abordagem "Bottom-Up" começamos por definir o Modelo Sintáctico. Neste caso, o Modelo Sintáctico é baseado nas estruturas de dados das fontes de dados a integrar. Primeiro é realizado um levantamento dos modelos de dados da fontes a integrar. Os modelos são depois integrados num único modelo de dados. Este modelo tenta representar toda a informação necessária a ser integrada no sistema. O modelo é baseado nos modelos de dados das fontes de dados externas. A Ontologia para o sistema é baseada no Modelo de Dados Sintáctico.

A vantagem desta abordagem está na maior facilidade de criação do Modelo Sintáctico e da Ontologia. A desvantagem está na dificuldade de integração de novas fontes de dados externas. O Modelo Sintáctico é criado baseado apenas nas fontes de dados conhecidas na altura da criação do modelo. A integração de futuras fontes de dados poderá tornar-se numa tarefa muito complexa ou mesmo impossível sem alterarmos os modelos anteriormente definidos. Esta abordagem deverá ser utilizada quando estamos a integrar um número limitado de fontes de dados e quando não se prevê adicionar novas fontes de dados no futuro. Na figura 3.14 apresentamos a ordem de acções que definem a abordagem "Bottom-Up".

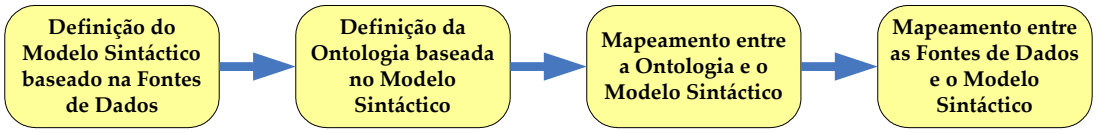

Figura 3.27 – Abordagem "Bottom-Up".

3.7.3 Abordagem Mista

Uma abordagem de integração mista baseia-se na utilização das duas abordagens anteriores em paralelo. Ou seja, a Ontologia e o Modelo de Dados Sintáctico são

definidos em paralelo. A Ontologia é definida sem conhecimento das estruturas de dados da fontes de dados a integrar e o Modelo Sintáctico é definido sem conhecimento sobre a Ontologia. A Ontologia é definida por peritos no domínio do sistema, tal como na abordagem "Top-Down". A definição do Modelo Sintáctico é baseada nos modelos de dados da fontes de dados a integrar.

Esta abordagem tem a vantagem de simplificar a definição do Modelo Sintáctico e da definição da Ontologia ser independente dos esquemas de dados das fontes de dados a integrar. Uma Ontologia independente dos modelos das fontes de dados está mais preparada para a integração de novas fontes de dados. O grande problema desta abordagem está no processo de mapeamento entre o Modelo Sintáctico e a Ontologia. Como ambos o modelos foram definidos sem qualquer conhecimento um do outro, a probabilidade de existência de inconsistências entre os dois modelos é muito grande. Normalmente as inconsistências são resolvidas com redefinições da Ontologia ou do Modelo Sintáctico. Os processos de redefinição dos modelos são complexos e morosos. A abordagem Mista deverá ser utilizada quando temos duas equipas de modelação, uma especialista nas fontes de dados a integrar e outra especialista no domínio do sistema. Esta abordagem também deverá ser utilizada quando queremos ter integração de dados simples e ao mesmo tempo termos a capacidade de integração de novas fontes de dados sem provocar grande impacto no sistema. Na figura 3.15 está representada a abordagem Mista.

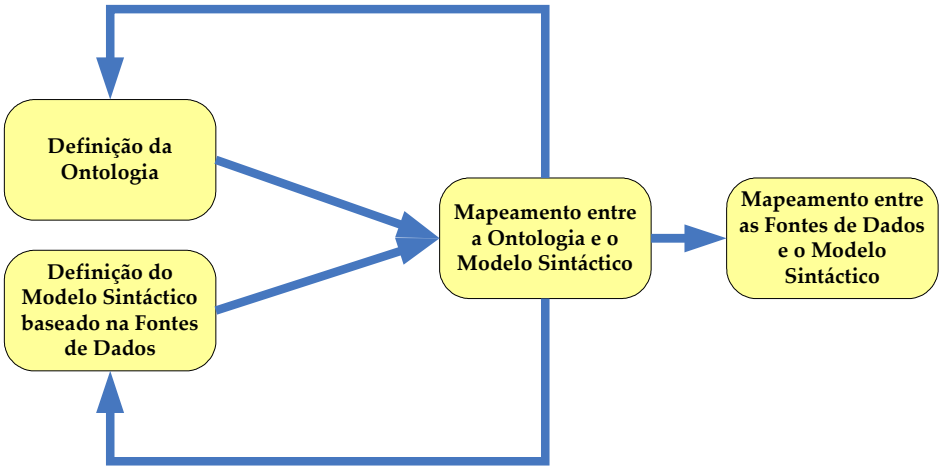

Figura 3.28 – Abordagem Mista.

4 IMPLEMENTAÇÃO DA ARQUITECTURA

4.1 APLICAÇÃO "GATHERER"

Para a implementação do nível sintáctico, optamos por desenvolver uma aplicação completamente independente do resto dos módulos da arquitectura. Esta opção deveu-se ao facto de este módulo ser desenvolvido por uma equipa diferente daquela que desenvolveu a arquitectura. Tornando o nível sintáctico suportado por uma aplicação independente, possibilita a utilização da aplicação para integração de fontes de dados sem a utilização das funcionalidades semânticas. A aplicação desenvolvida para implementar o nível sintáctico tem o nome de "Gatherer" (Silva, 2006).

A aplicação "Gatherer" foi implementada na linguagem Java. Esta aplicação permite a integração de fontes de dados de quatro tipos deferentes: Páginas Web, Bases de Dados, Ficheiros XML e Web Services. Para cada um dos tipos de fontes de dados foi implementado um processo de extracção de dados. Os processos são definidos de seguida.

- **Páginas Web:** No caso das Páginas Web, a extracção dos dados é realizada utilizando a linguagem WebL (Marais 1999). WebL é uma linguagem de script criada para processar e extrair dados de páginas Web. Suporta os protocolos HTTP e FTP para a obtenção dos dados. Na aplicação "Gatherer", para cada item de informação a extrair, é necessário criar uma função em WebL que permita obter e retornar os dados necessários para o respectivo item. A informação extraída é guardada num ficheiro temporário que é depois utilizado pela aplicação. Por exemplo, o código WebL para obter o nome do hotel referente à uma determinada página Web é o seguinte:

```
import Str, Files;
var P = GetURL("http://....../hotel.htm", nil, nil,
                [.mimetype="text/plain".]);
var t = Text(P);
var St = Str_Search(t,
           "<div class=\"hotel\"><b class=\"sans\">" +
          `(?)[0-9a-zA-Z'/., ()]+`);
var spliter = Str_Split(St[0][0],"<>");
var nome = Select(spliter[2],0,6);
Files_SaveToFile( "output.dat" ,nome);
```

Através da função "GetURL" definimos qual a página Web a considerar. Utilizando a função "Str_Search", e através da utilização de expressões regulares, definimos qual a parte do documento HTML que queremos extrair. A informação a retornar é depois guardada no ficheiro temporário "output.dat".

- **Bases de Dados:** Para a extracção de dados referentes a bases de dados relacionais é utilizada a linguagem SQL. Para cada tipo de base de dados é necessário obter o driver em Java que permita o acesso à base de dados em questão. Por exemplo, para bases de dados do tipo MySQL é necessário utilizar o driver com.mysql.jdbc.Driver. A única restrição na utilização dos drivers é estes suportarem a linguagem SQL. Para obter os dados referentes a um determinado item, é necessário definir a query SQL que permite extrair os dados da base de dados. Por exemplo, Para obter os nomes dos hotéis que se encontram numa determinada base de dados na tabela "Hotel", poderíamos definir a seguinte query SQL:

```
SELECT nome_hotel FROM Hotel
```

  Na query SQL, a palavra "Hotel" refere-se à tabela onde se encontra guardada a informação dos hotéis e a palavra "nome_hotel" o campo da tabela "Hotel" onde se encontram definidos os nomes dos hotéis.

- **Ficheiros XML:** No caso dos ficheiros XML utilizamos a linguagem XQuery (XQuery, 2006). Esta linguagem permite obter informação referente aos elementos ou atributos existentes num determinado ficheiro XML. O sistema suporta a integração tanto de ficheiros XML locais como de ficheiros XML remotos. Por exemplo, para obter o nome dos hotéis existentes no ficheiro XML local "hoteis.xml", poderíamos definir a seguinte instrução em XQuery:

```
for $x in doc("hoteis.xml")/hoteis/hotel
     return $x/nome
```

  Na frase XQuery do exemplo, primeiro definimos a variável "x" que aponta para todos os elementos XML que se encontram no ficheiro "hoteis.xml" e em que o seu XPath é "/hoteis/hotel". No fim, pedimos para ser retornado o sub-elemento "nome" do XPath definido.

- **Web Services:** A integração com Web Services é realizada a partir do processamento do WSDL referente a cada Web Service a ser integrado. Basta

indicar qual o URL de acesso ao WSDL que descreve o Web Service e a aplicação "Gatherer" trata de criar os métodos de acesso ao serviço. No entanto, existe uma grande limitação na integração de Web Services. O resultado da invocação de um Web Service só pode ser associado a um único atributo do modelo sintáctico. Ou seja, se um Web Service retornar uma estrutura de dados que inclua informação referente a vários atributos do modelo sintáctico, não é possível associar a informação retornada pelo Web Service aos vários atributos. Geralmente os Web Services retornam estruturas de dados referentes a vários conceitos e raramente retornam informação referente apenas a um atributo de um conceito. Apesar de a documentação referente à aplicação "Gatherer" indicar que suporta a integração com Web Services, optamos por não considerar este tipo de integração na implementação da arquitectura SEED.

4.1.1 Registo das Fontes de Dados

O registo das fontes de dados é realizado através da indicação da informação necessária ao acesso aos dados de uma determinada fonte. As fontes registadas são mantidas num ficheiro XML denominado "DataSources.xml". O ficheiro XML é constituído por quatro diferentes tipos de elementos: "dataBase", "webPage", "localXML", "remoteXML" e "webService". Cada um destes elementos é composto por um atributo de nome "ID" que identifica cada uma das fontes de dados registadas no sistema. Em cada um dos elementos existem sub-elementos que são utilizados para definir a informação de acesso às fontes de dados. Por exemplo, no caso de uma fonte do tipo Base de Dados é necessário indicar o URL de acesso ao motor de base de dados, o driver a ser utilizado, o nome da base de dados e o utilizador e a password de acesso. Exemplo de um registo de uma fonte de dados do tipo Base de Dados:

```
<?xml version="1.0" encoding="UTF-8"?>
<dataSource>

      . . .

      <dataBase ID="DB1">
            <driver>com.mysql.jdbc.Driver</driver>
            <url>jdbc:mysql://localhost/</url>
            <dbName>Hoteis</dbName>
            <login>root</login>
            <pass>123</pass>
      </dataBase>

      . . .
```

```
</dataSource>
```

4.1.2 Mapeamento com o Modelo Sintáctico

Depois do registo das fontes de dados, é necessário definir o mapeamento entre os dados das fontes externas com o Modelo Sintáctico. O Modelo Sintáctico é definido utilizando a linguagem XML Schema (XML Schema, 2007). O mapeamento entre as fontes de dados e o XML Schema, é realizado através da criação de um ficheiro XML denominado "MappedElements.xml". Neste ficheiro são definidos os mapeamentos entre os dados das fontes externas e os elementos definidos no XML Schema. O documento que define os mapeamentos contém a estrutura definida pelo XML Schema. Ou seja, se no XML Schema definirmos o elemento "Hotel", para criarmos um mapeamento para o elemento "Hotel" criamos este elemento no documento XML de mapeamento. Existem dois tipos de elementos definidos no documento de mapeamento, os elementos que contêm sub-elementos e os elementos que não contêm sub-elementos. Os elementos que contêm sub-elementos referem-se a estruturas de dados que agregam outras estruturas ou elementos com valores simples. Os elementos que não contêm sub-elementos referem-se a valores simples que serão obtidos através das fontes de dados. Quando criamos uma entrada de mapeamento para um elemento com sub-elementos, temos de definir qual o identificador deste elemento na fonte de dados que irá preencher os seus sub-elementos. Este identificador é definido pelo atributo "ID" e é utilizado para relacionar com os possíveis sub-elementos constituídos por estruturas de dados. Aos elementos que não contêm sub-elementos, é associado o script que define como serão obtidos os dados que irão atribuir valor a este elemento. Para além do script, existe um atributo de nome "DATASOURCE", onde deverá ser indicada a fonte de dados que irá ser utilizada para adicionar o valor ao elemento mapeado. Exemplo de um mapeamento para o elemento do Modelo Sintáctico "hotel":

```
<?xml version="1.0" encoding="utf-8"?>
<thing>
   . . .

   <hotel ID="hotel_id">
     <name DATASOURCE="DB1" >SELECT nome FROM hotel</name>
     <stars DATASOURCE="DB1" >SELECT estrelas FROM hotel</stars>
     <room ID="hotel_id">
        <type DATASOURCE="DB1" >SELECT type FROM room</type>
       <tv DATASOURCE="DB1">SELECT tv FROM room</tv>
     </room>
   </hotel>

   . . .
```

</thing>

4.1.3 Processo de Extracção dos Dados

O processo de extracção dos dados das fontes externas irá utilizar o documento XML Schema que define o Modelo Sintáctico, o ficheiro XML dos registos das fontes de dados e o ficheiro XML de mapeamento. A ordem de acções que compõem o processo de extracção dos dados é apresentada na figura 4.1.

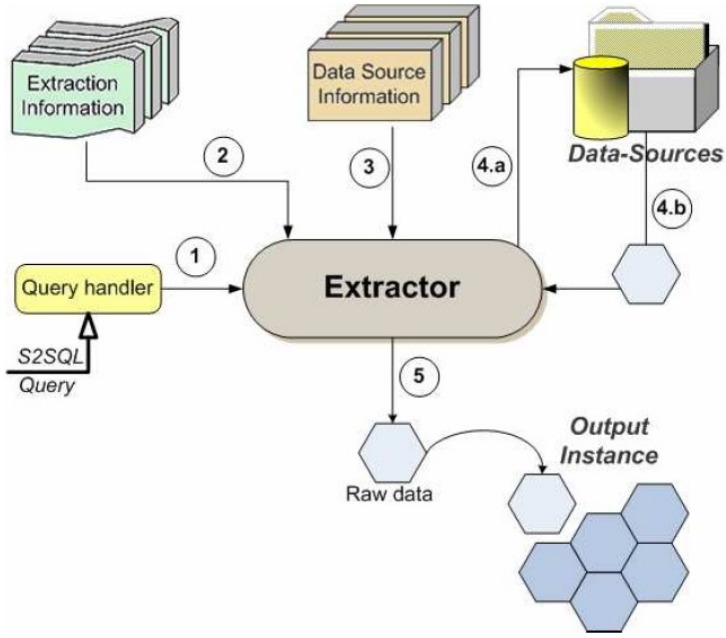

Figura 4.29 – Processo de Extracção dos Dados (Silva, 2006).

O processo de extracção dos dados é realizado pela seguinte ordem:

(1) A query sintáctica definida na linguagem S2SQL é processada e são obtidos os elementos e atributos a serem extraídos;

(2) São obtidos os scripts de extracção definidos no ficheiro XML que inclui a definição dos mapeamentos;

(3) É obtida a informação de acesso às fontes de dados externas definidas nos mapeamentos obtidos;

(4) É realizado o acesso à fonte de dados externa e é invocado o script de extracção. A fonte de dados externa retorna a informação em resposta ao script de extracção;

(5) A informação extraída é adicionada às instâncias do Modelo Sintáctico a serem retornadas em resposta à query S2SQL enviada.

4.1.4 Integração com Arquitectura

A integração da aplicação "Gatherer" com a arquitectura é muito simples. A aplicação "Gatherer" disponibiliza uma API que inclui o objecto "QueryEngine". Este objecto disponibiliza o método "Execute" que permite extrair informação mediante a indicação de uma query S2SQL. Os dados são retornados por este método no formato XML. O XML retornado respeita o Modelo Sintáctico definido para a aplicação.

4.2 APLICAÇÃO "JXML2OWL MAPPER"

Tal como aconteceu no caso do nível sintáctico, também o módulo *Mapping* foi implementado como sendo uma aplicação independente da arquitectura. A razão de criação de uma aplicação independente deveu-se ao facto deste módulo ser implementado por uma equipa diferente daquela que implementou o resto da arquitectura. Outra das razões foi a intenção de criar uma aplicação de definição de mapeamentos entre modelos que poderia ser utilizadas noutros contextos. A aplicação que implementa o módulo *Mapping* tem o nome de "JXML2OWL Mapper" (Rodrigues e Rosa, 2006). Foi implementada utilizando a linguagem de programação Java e permite definir o mapeamento entre documentos XML Schema e documentos OWL (McGuinness e Harmelen, 2004). Depois de definido o mapeamento, a aplicação também permite a criação de um documento XSLT (Clark, 1999) que define o processo de transformação de instâncias do modelo XML Schema em instâncias do modelo OWL.

4.2.1 Definição das Regras de Mapeamento

As regras de mapeamento são definidas na aplicação "JXML2OWL Mapper". Nesta aplicação deverão ser indicados os documentos XML Schema e OWL a serem mapeados. Depois de indicados, estes documentos são apresentados graficamente numa estrutura em árvore. No caso do documento XML Schema, são apresentados os nós XML definidos no documento XML Schema. No caso do documento OWL, são apresentadas as classes OWL. Os mapeamentos são definidos arrastando uma ligação entre os nós XML definidos no documento XML Schema e as classes definidas no documento OWL. Para cada mapeamento é ainda possível definir mapeamentos entre sub-nós do nó XML mapeado e propriedades referentes à classe OWL mapeada. Todos os mapeamentos definidos são guardados num ficheiro XML. Este ficheiro guarda todas as regras de mapeamento e permite reutilizar as regras definidas de modo a ser possível editá-las numa futura utilização. Na figura 4.2 apresentamos o exemplo de mapeamento utilizando a interface da aplicação "JXML2OWL Mapper". O

mapeamento é realizado entre um documento XML Schema e um documento OWL, ambos definindo informação no domínio do turismo.

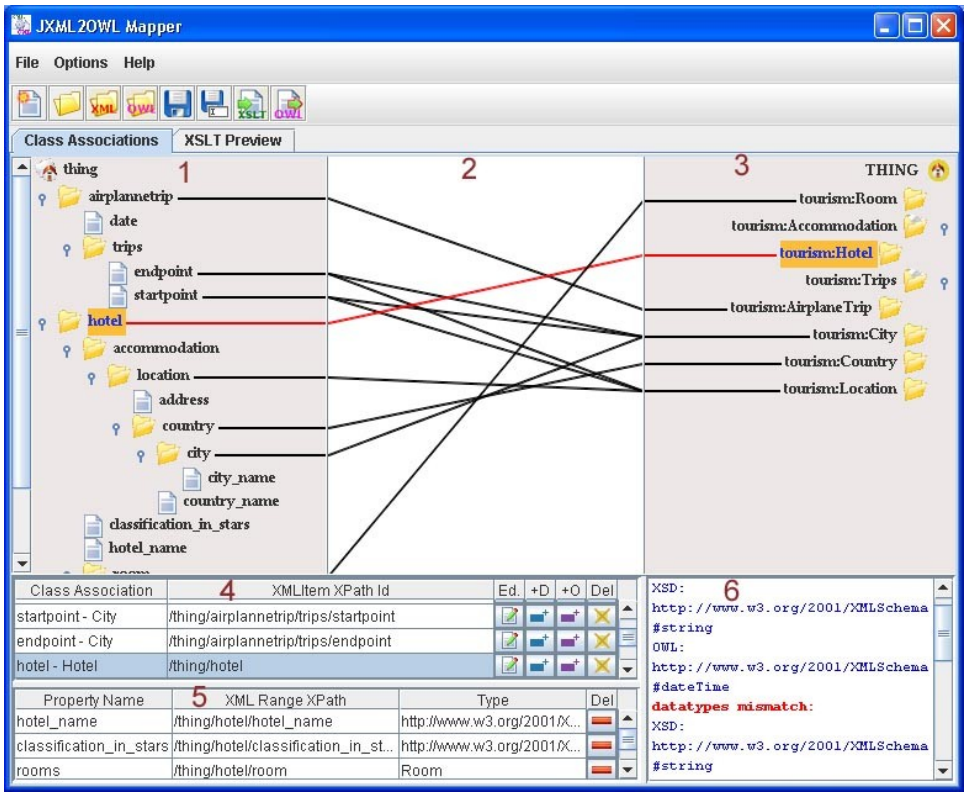

Figura 4.30 – "JXML2OWL", interface de mapeamento (Rodrigues e Rosa, 2006).

Através da figura 4.2 podemos detectar as seguintes zonas na interface de mapeamento:

(1) Estrutura em árvore do documento XML Schema.

(2) Linhas de mapeamento entre os nós XML e as classes do documento OWL.

(3) Estrutura em árvore do documento OWL.

(4) Tabela que apresenta os mapeamentos já definidos.

(5) Tabela que apresenta os mapeamentos das propriedades OWL referentes ao mapeamento entre o nó XML e classe OWL seleccionado.

(6) Janela onde os utilizadores obtêm informação sobre erros ou avisos referentes aos mapeamentos criados.

Através da interface da aplicação "JXML2OWL Mapper" poderemos definir três tipos de mapeamento:

- **Mapeamento entre um nó XML e uma classe OWL:** Este mapeamento é criado através de uma ligação directa entre um nó XML, apresentado na interface de mapeamento, e uma classe OWL, apresentada na mesma interface. Indica que os nós XML do tipo do nó mapeado deverão ser transformados em instâncias da classe OWL mapeada. O mapeamento é definido no ficheiro XML de regras de mapeamento através do nó "class". Este nó contém dois sub-nós que definem o mapeamento. O sub-nó "owlClassName" define a classe OWL mapeada e o sub-nó "elementXPath" define o nó XML mapeado. Exemplo de um mapeamento entre um nó XML e uma classe OWL:

```
<?xml version="1.0" encoding="UTF-8"?>
<mapping>
 <xmlURI>http://…/XMLTourism.xsd</xmlURI>
 <owlURI>http://…/tourism.owl</owlURI>
 <owlPrefix>tourism</owlPrefix>
 <associations>
   . . .
  <class>
   <owlClassName>Hotel</owlClassName>
   <elementXPath>/thing/hotel</elementXPath>
  </class>
   . . .
 </associations>
</mapping>
```

- **Mapeamento entre um nó XML e uma propriedade de tipo de dados OWL:** Este mapeamento é criado seleccionando um mapeamento entre um nó XML e uma classe OWL previamente definido. Indica que os valores dos nós do tipo do nó mapeado deverão ser atribuídos às propriedades do tipo da propriedade mapeada e pertencentes às instâncias da classe OWL seleccionada. No ficheiro XML de regras de mapeamento, este tipo de mapeamento é definido através do nó "datatypeProperty". Este nó é constituído por três sub-nós. O sub-nó "owlPropertyName" que indica a propriedade OWL a ser mapeada, o sub-nó "owlDomainClassName" que indica a que classe OWL pertence a propriedade e o sub-nó "valueXPath" que indica qual o nó XML onde será obtido o valor para a propriedade. Exemplo de um mapeamento entre um nó XML e uma propriedade de tipo de dados OWL:

```
<?xml version="1.0" encoding="UTF-8"?>
<mapping>
 <xmlURI>http://…/XMLTourism.xsd</xmlURI>
 <owlURI>http://…/tourism.owl</owlURI>
 <owlPrefix>tourism</owlPrefix>
 <associations>
   . . .
```

```
    <datatypeProperty>
     <owlPropertyName>hotel_name</owlPropertyName>
     <owlDomainClassName>Hotel</owlDomainClassName>
     <valueXPath>/thing/hotel/hotel_name</valueXPath>
    </datatypeProperty>
    . . .
  </associations>
</mapping>
```

- **Mapeamento entre um nó XML e uma propriedade de objecto OWL:** Uma propriedade de objecto em OWL representa uma propriedade que tem como tipo de valor uma classe OWL. Este tipo de mapeamento é criado também seleccionando um mapeamento entre um nó XML e uma classe OWL previamente definido. Um mapeamento deste tipo indica que os nós do tipo do nó mapeado deverão ser transformados nas propriedades de objecto do tipo da propriedade mapeada e pertencentes às instâncias da classe OWL seleccionada. No ficheiro XML de regras de mapeamento, este tipo de mapeamento é definido através do nó "objectProperty". Este nó é constituído por quatro sub-nós. O sub-nó "owlPropertyName" indica qual a propriedade mapeada, o sub-nó "owlDomainClassName" indica a que classe OWL pertence a propriedade, o sub-nó "rangeXPath" indica o nó XML mapeado e o sub-nó "owlRangeClassName" indica a classe OWL que define qual o tipo de instâncias que podem tomar o valor da propriedade. A aplicação só deixa mapear nós XML a uma propriedade de objecto OWL quando estes estejam previamente mapeados com a classe que define qual o tipo de instâncias que pode tomar o valor da propriedade. Exemplo de um mapeamento entre um nó XML e uma propriedade de objecto OWL:

```
<?xml version="1.0" encoding="UTF-8"?>
<mapping>
 <xmlURI>http://…/XMLTourism.xsd</xmlURI>
 <owlURI>http://…/tourism.owl</owlURI>
 <owlPrefix>tourism</owlPrefix>
 <associations>
    . . .
 <objectProperty>
    <owlPropertyName>location</owlPropertyName>
    <owlDomainClassName>Hotel</owlDomainClassName>
    <owlRangeClassName>Location</owlRangeClassName>
    <rangeXPath>
      /thing/hotel/accommodation/location
    </rangeXPath>
 </objectProperty>
    . . .
 </associations>
</mapping>
```

Neste exemplo, o nó XML "/thing/hotel/accommodation/location" só pode ser mapeado com a propriedade "location" se este nó já estiver previamente mapeado com a classe OWL "Location".

4.2.2 Definição do Processo de Transformação

Definidos os mapeamentos entre o Modelo Sintáctico e a Ontologia, é necessário definir o processo de transformação entre as instâncias destes dois modelos. O processo de transformação deverá transformar documentos XML, que respeitem o XML Schema definido, em instâncias OWL referentes à Ontologia. O processo de transformação é gerado pela aplicação "JXML2OWL Mapper". Baseado nos mapeamento definidos, a aplicação "JXML2OWL Mapper" gera um documento XSLT (Clark, 1999) que define a transformação entre os modelos. A linguagem XSLT é a linguagem recomendada pelo W3C (World Wide Web Consortium) para transformar documentos XML em outros documentos XML. Como um documento OWL também é um documento XML, logo, o XSLT pode ser utilizado para o nosso caso. Apesar de a transformação utilizando a linguagem XSLT não oferecer a melhor performance, traz-nos a vantagem de utilização de um standard. Para além de utilizarmos uma linguagem revista e utilizada por muita gente, também podemos utilizar as ferramentas já criadas para a utilização desta.

Tendo o documento XSLT criado, a transformação entre os dados sintáctico e os dados semânticos é quase directa. Basta utilizarmos um processador de documentos XSLT e aplicá-lo aos documentos XML. Na figura 4.3 está representado como o documento XSLT é gerado e como este é utilizado no processo de transformação.

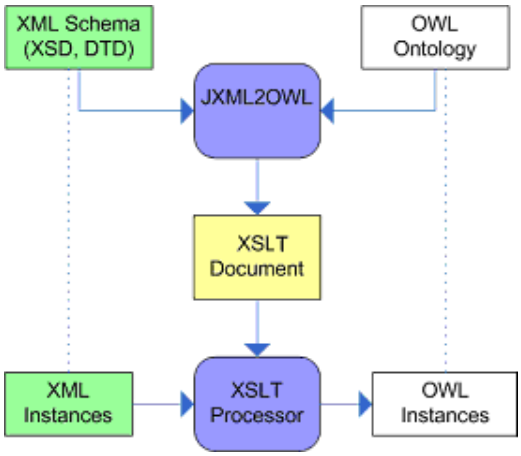

Figura 4.31 – Definição do processo de transformação (Rodrigues e Rosa, 2006).

4.3 Processos de Transformação

Os processos de transformação são processados no nível de mapeamento e têm a função de transformar, os dados e as queries, entre os níveis sintáctico e semântico. Existem dois processos de transformação, a transformação das queries semânticas em queries S2SQL e a transformação dos dados no formato XML em instâncias da Ontologia.

4.3.1 Processo de Transformação de Dados

A informação retornada pelas fontes de dados externas, é formatada em documentos XML que respeitam o Modelo Sintáctico definido pela linguagem XML Schema. Depois é necessário transformar esta informação em instâncias da Ontologia definida para o sistema. A transformação é obtida através da aplicação do documento XSLT, obtido através do mapeamento entre o Modelo Sintáctico e a Ontologia. Para proceder à transformação é aplicado o documento XSLT aos dados XML utilizando um processador XSLT. O processador XSLT retorna depois as instâncias OWL obtidas na transformação. Na figura 4.4 está representado como se processa esta transformação.

Figura 4.32 – Processo de transformação dos dados.

4.3.2 Processo de Transformação das Queries

As queries definidas no nível semântico, têm de ser transformadas em queries S2SQL de forma a serem compreendidas pela aplicação "Gatherer". Depois, na aplicação "Gatherer" as queries S2SQL são processadas para serem definidos quais os dados a serem extraídos das fontes de dados externas.

Para a definição das queries semânticas é utilizada a linguagem de query nRQL (New RacerPro Query Language). A linguagem nRQL (RacerPro, 2005) é a linguagem de query utilizada pelo motor de inferência RacerPro (Renamed ABox and Concept Expression Reasoner Professional). Pode ser utilizada para obter informação referente a documentos OWL. A escolha desta linguagem de query, deve-se principalmente ao facto de escolhermos como motor de inferência para a implementação da arquitectura o sistema RacerPro. Utilizando a linguagem de query associada ao motor de inferência, torna mais eficiente a interpretação das regras semânticas nas queries criadas. Além disto, a linguagem nRQL é considerada como sendo muito mais expressiva que as linguagens RDF típicas e, em determinados aspectos, que a semi-oficial linguagem de query para OWL, a linguagem OWL QL (RacerPro, 2005).

A linguagem nRQL permite a criação de queries conjuntivas. Numa query nRQL, são utilizadas variáveis que deverão ser mapeadas às instâncias OWL que satisfazem a query. As classes OWL providenciam o vocabulário específico do domínio que poderá ser utilizado nas queries nRQL. Uma query nRQL é constituída por duas partes:

- **Query Head:** É a parte inicial da query e indica quais as variáveis que deverão ser retornadas. As variáveis utilizadas na "Query Head" deverão ser um subconjunto das variáveis utilizadas na "Query Body".

- **Query Body:** É a segunda parte da query e é aqui que são definidas as expressões que irão definir as variáveis utilizadas.

Na figura 4.5 encontra-se representado o exemplo de uma query em nRQL onde são identificas as duas partes referidas.

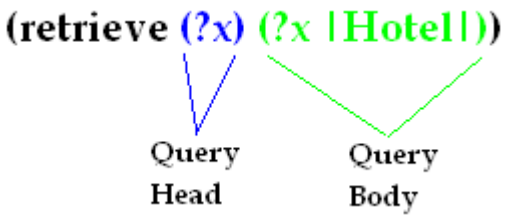

Figura 4.33 – Exemplo de uma query nRQL.

Neste exemplo, na "Query Head" é pedido para serem retornados os valores referentes à variável "x". Na parte da "Query Body", indicamos que a variável "x" deverá ser uma instância da classe"Hotel". Um possível resultado desta query seria:

```
((?X1 |HotelMadeira|)
 (?X1 |HotelFlorasol|) ... )
```

Na implementação da arquitectura pretendemos criar queries semânticas que representam a pesquisa que o cliente, ou aplicação cliente, pretende fazer. As queries geradas pelo sistema são sempre divididas por classes. Se o cliente pretender obter dados de duas classes distintas, a sua pesquisa será dividida em duas queries distintas, uma para cada classe. O sistema permite pesquisar instâncias de classes sem indicar qualquer restrição ou então pesquisas de classes onde é indicada uma ou mais restrições. As restrições são criadas utilizando as propriedades das classes. Por exemplo, podemos criar uma query que nos retorne todos os hotéis com classificação de 4 ou mais estrelas. Neste caso o sistema iria gerar a seguinte query nRQL:

```
(retrieve (?x1)
          (and (?x1 |Hotel|)
               (?x1 (at-least 1 |Stars|(and (min 4))))
          )
)
```

Para além das queries nRQL, o sistema também terá de criar as queries sintácticas definidas através da linguagem S2SQL. Só assim poderemos obter os dados das fontes externas, que irão ser adicionados à Ontologia na forma de instâncias desta mesma Ontologia. As queries S2SQL são criadas de forma a que sejam obtidos todos os dados necessários à criação das instâncias, que serão obtidas através da query semântica utilizada. Por exemplo, se definirmos uma query semântica para retornar todas as instâncias da classe "Hotel" da Ontologia com 4 ou mais estrelas, então temos de criar a query S2SQL que permita obter todos os dados referentes aos elementos do Modelo Sintáctico mapeados com a classe "Hotel". Supondo que existem dois elementos do Modelo Sintáctico, "hotel1" e "hotel2", mapeados com a classe "Hotel" da Ontologia. Neste caso, o sistema tem de criar as seguintes queries S2SQL:

- ```
  SELECT hotel1
  WHERE (hotel1.stars >= "4")
  ```

- ```
  SELECT hotel2
  WHERE (hotel2.stars >= "4")
  ```

Para determinar quais os elementos do Modelo Sintáctico mapeados com uma determinada classe da Ontologia, é utilizado o ficheiro XML que guarda as regras de mapeamento. Este ficheiro é gerado pela aplicação "JXML2OWL Mapper". É também através deste ficheiro que conseguimos determinar que elementos do Modelo Sintáctico devemos utilizar para a criação das restrições de propriedades. Por exemplo,

para criar a restrição que indica que queremos obter apenas os hotéis com 4 ou mais estrelas, temos de obter qual o nó XML mapeado com a propriedade "Stars" da classe "Hotel " da Ontologia.

A criação de queries em S2SQL torna-se mais complexa quando estamos a tratar de classes, que estão relacionadas com outras classes através de relações de herança. Existem essencialmente três situações diferentes quando estamos a tratar de relações de herança:

- **Pesquisa de super classe ou subclasse mapeada no Modelo Sintáctico:** Neste caso pretendemos obter uma determinada classe, definida na Ontologia como super classe ou subclasse de outras classes, e esta classe encontra-se mapeada directamente com um elemento do Modelo Sintáctico. Quando isto acontece, criamos a query S2SQL normalmente, criando queries S2SQL para obter os elementos do Modelo Sintáctico mapeados com a classe em questão. Quando existe mapeamento de uma super classe com um ou mais elementos do Modelo Sintáctico, a responsabilidade de obter todos os dados necessários a esta classe é de quem define o mapeamento.

- **Pesquisa de super classe não mapeada no Modelo Sintáctico:** Como a super classe não se encontra mapeada, no processo de criação das queries S2SQL temos de obter os dados para a classe através das subclasses desta. Assim, o sistema tem de criar as queries S2SQL, utilizando os elementos do Modelo Sintáctico que se encontram mapeados com subclasses da classe em questão.

- **Pesquisa de subclasse não mapeada com o Modelo Sintáctico:** Quando uma subclasse não se encontra mapeada com o Modelo Sintáctico, significa que só poderão existir instâncias desta classe se existir uma regra de negócio que defina esta subclasse através de características especiais da sua respectiva super classe. Por exemplo, se definirmos a classe "HoteldeLuxo" na Ontologia como subclasse da classe "Hotel" e não mapearmos a classe "HoteldeLuxo" com qualquer elemento do Modelo Sintáctico. Criamos depois uma regra que indica que todas as instâncias da classe "Hotel" que tenham o valor da propriedade "Stars" igual ou superior a 4 são consideradas também como sendo uma classe "HoteldeLuxo". Neste cenário, para obtermos instâncias da classe "HoteldeLuxo", temos de obter os dados sintácticos referentes à classe "Hotel". Assim, ao pesquisarmos por uma

subclasse não mapeada no Modelo Sintáctico, o sistema trata de criar as queries S2SQL referentes aos elementos do Modelo Sintáctico mapeados com a super classe da subclasse em questão.

Para além das relações de herança, existem outros tipos de relações que não são consideradas no processo de transformação de queries. Por exemplo, na Ontologia podemos definir uma classe através da união de outras duas classes. Na criação das queries sintácticas, teríamos de considerar este facto de forma a obter todos os dados necessários para criarmos todas as instâncias referentes a esta classe especial. Na implementação da arquitectura não suportamos este tipo de definição de classe, principalmente devido à complexidade que tal implementação implicaria. No entanto, a implementação da criação das queries S2SQL, prevê o futuro suporte à interpretação de novas definições de classes.

4.4 Nível Semântico

No nível semântico tivemos de implementar dois dos principais componentes da arquitectura, a Ontologia e o motor de inferência. É também implementado neste nível todo o processo de interacção entre as possíveis aplicações clientes e o sistema.

4.4.1 Ontologia

Para a definição da Ontologia optamos por utilizar a linguagem OWL (Bechhofer, Harmelen, Hendler, Horrocks, McGuinness, Patel-Schneider e Stein, 2004). Para além de ser uma recomendação do grupo W3C para a definição de Ontologias, é também uma linguagem semântica que oferece maior expressividade que as linguagens RDF ou RDF Schema.

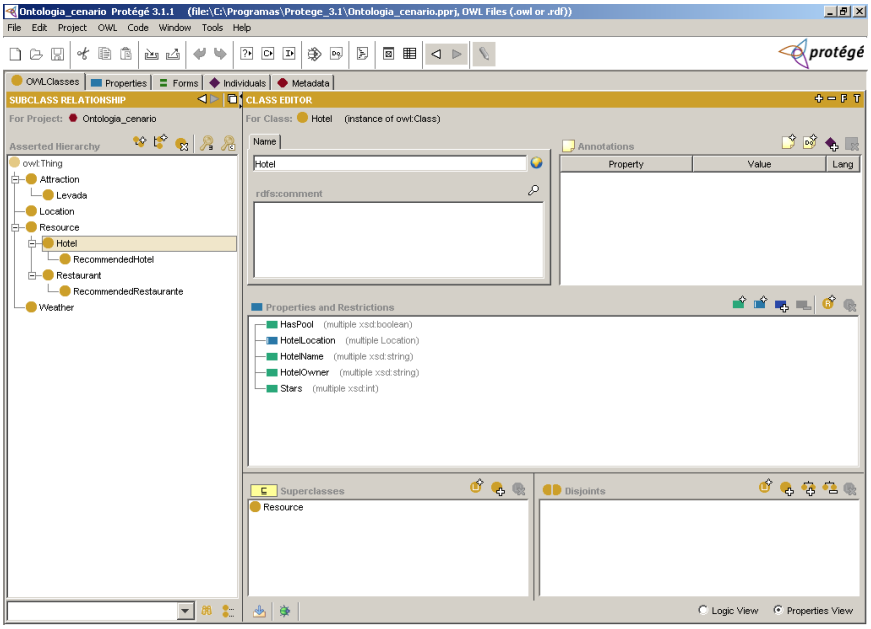

Figura 4.34 – Interface da aplicação Protégé-OWL.

A definição da Ontologia para o sistema pode ser realizada por qualquer aplicação que permita a criação de documentos OWL. Esta até poderá ser definida utilizando um simples processador de texto. No entanto, aconselhamos a utilização da aplicação Protégé-OWL (Knublauch, Fergerson, Noy e Musen, 2004). O Protégé-OWL permite a definição de Ontologias utilizando a linguagem OWL. Disponibiliza uma interface

onde permite a definição de classes e propriedades de forma simples. Na figura 4.6 apresentamos o aspecto geral da interface da aplicação Protégé-OWL.

4.4.2 Motor de Inferência

De forma a validar a Ontologia e para verificar se as instâncias adicionadas à Ontologia são válidas, utilizamos um motor de inferência na implementação do nível semântico. É também o motor de inferência que irá processar e interpretar as regras de negócio definidas no sistema. O motor de inferência permite ainda a aquisição de novo conhecimento baseado no conhecimento definido na Ontologia. Através da interpretação de relações entre elementos, o motor de inferência consegue criar novas relações que anteriormente não estavam presentes na Ontologia de maneira explícita.

No nosso caso, optamos por utilizar o motor de inferência RacerPro (RacerPro 2005). É um motor de inferência comercial que disponibiliza versões de demonstração e versões gratuitas para fins académicos. Para o nosso sistema utilizamos a versão 1.9. Suporta a linguagem OWL-DL com tipos de dados simples. Ou seja, apenas suporta propriedades de tipo de dados definidos com os tipos int, string, float e boolean. Escolhemos este motor de inferência pelas seguintes razões:

- Pode ser integrado com o Protégé de forma a funcionar como o motor de inferência da aplicação. Desta forma poderemos testar a consistência da Ontologia na altura em que esta é criada.

- Disponibiliza uma interface DIG (Bechhofer, 2003) que permite a utilização em paralelo com a API do Protégé. Ou seja, podemos carregar uma Ontologia através da API do Protégé e depois utilizar o RacerPro sem que haja a necessidade de carregar a Ontologia no motor de inferência. O RacerPro consegue carregar a Ontologia directamente da API do Protégé. Este facto torna o processo de inferência muito mais rápido nos casos em que utilizamos a API do Protégé para a manipulação dos conceitos OWL.

- Consegue processar não só documentos OWL-Lite mas também documentos OWL-DL com apenas algumas pequenas restrições.

- Disponibiliza um processo de acesso através do protocolo HTTP. Desta forma permite uma integração mais simples.

- Inclui processos de cálculo optimizados para lógica de descrição com grande expressividade.

- Inclui uma linguagem de query que permite obter conhecimento da Ontologia, incluído o conhecimento inferido.

4.4.3 Controlo de Acesso ao Sistema

De forma a permitir uma utilização mais simples do sistema por parte da aplicações clientes, foi definido o módulo *Data Access Control*. Este módulo trata das queries enviadas pelas aplicações clientes e do processamento das regras de negócio.

Na implementação deste módulo utilizamos a API do Protégé. Esta API permite-nos, de forma simples, processar documentos OWL e manipular o seu conteúdo. Através desta API, podemos obter informação sobre as classes e respectivas propriedades que se encontram definidas numa determinada Ontologia. Também podemos obter as instâncias presentes na Ontologia. Os dados são retornados numa estrutura de dados simples, à imagem da estrutura que existe nos documentos OWL. A API do Protégé utiliza a API Jena, suportando as funcionalidades desta mas definindo uma interface de interacção muito mais simples. Na figura 4.7 apresentamos algumas das classes presentes na API do Protégé.

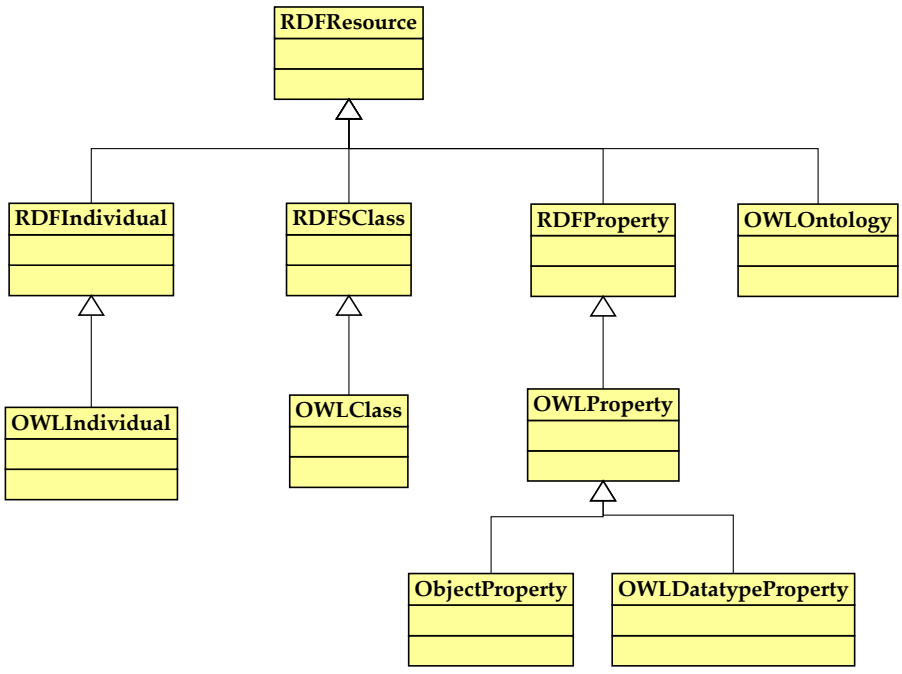

Figura 4.35 – Classes da API do Protégé.

A primeira acção a ser realizada pelo módulo *Data Access Control*, passa pelo carregamento do documento OWL que define a Ontologia para o sistema. É aqui que começamos a utilizar a API do Protégé. A Ontologia é carregada, em memória, através API do Protégé. Depois, iniciamos o motor de inferência RacerPro a partir do modelo da Ontologia já carregada na API do Protégé. A partir da Ontologia, são obtidas as classes e as propriedades definidas. Utilizando novamente a API do Protégé, disponibilizamos a informação de todas as classes da Ontologia, de forma a que as aplicações clientes possam definir as pesquisas à Ontologia. As pesquisas criadas são transformadas em queries nRQL e depois enviadas às fontes de dados externas. Os dados referentes às queries, são recebidos no nível semântico na forma de instâncias da Ontologia. Para a adição das classes à Ontologia utilizamos uma vez mais a API do Protégé. Para isso basta criar um objecto "OWLIndividual" e utilizar o método disponibilizado pela API que adiciona este à Ontologia. Depois de todas as instâncias serem adicionadas, são aplicadas as regras de negócio definidas na linguagem nRQL. As regras são aplicadas utilizando o motor de inferência RacerPro e através de pedidos HTTP. A partir deste momento é possível executar as queries nRQL de forma a obtermos a informação pedida inicialmente. Para a execução das queries também utilizamos o motor de inferência RacerPro e, novamente, através de pedidos HTTP ao motor de inferência. Em resposta às queries, obtemos os identificadores das instâncias da Ontologia que respeitam as restrições presentes nas queries e que respeitam as regras de negócio definidas. Através dos identificadores, e utilizando a API do Protégé, obtemos toda a informação referente às instâncias referidas. As instâncias são apresentadas às aplicações clientes numa estrutura de dados baseada na estrutura da classe "OWLIndividual" da API do Protégé.

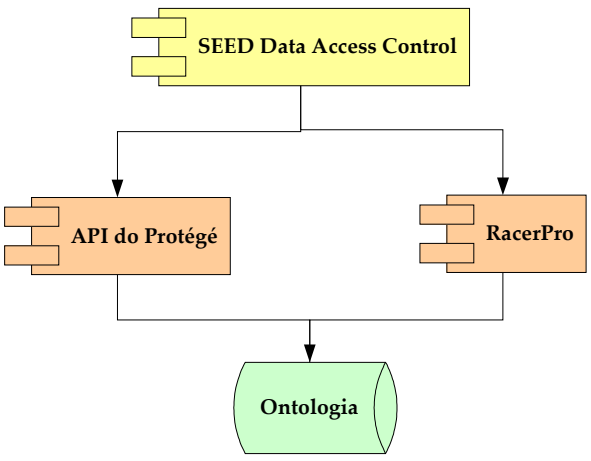

Figura 4.36 – Componentes do nível semântico.

Podemos depreender da descrição anterior que a implementação do módulo *Data Access Control* é, em grande parte, implementada através da utilização da API do Protégé e do motor de inferência RacerPro. Desta forma conseguimos abstrair-nos de grande parte da complexidade associada ao processamento dos sistemas semânticos. Na figura 4.8 está representada a relação entre os principais componentes que compõem o nível semântico.

4.5 PROTÓTIPO

Sobre o sistema implementado foi desenvolvido um protótipo ao qual denominamos por aplicação SEED. A aplicação SEED é um protótipo simples que pretende demonstrar como os componentes da arquitectura SEED interagem entre si. Também poderá ser utilizada no processo de configuração, testes e manutenção da arquitectura implementada. Esta aplicação disponibiliza uma interface de utilizador para a definição dos parâmetros de configuração do sistema. Disponibiliza também uma interface que permite o registo das fontes de dados, a criação de regras de negócio e a criação de queries e a verificação dos resultados destas.

A aplicação SEED foi implementada em Java utilizando a ferramenta de desenvolvimento Eclipse (www.eclips.org). Utiliza o plugin Jigloo GUI Builder (Jigloo, 2007) que permite a criação e gestão de interfaces num modo gráfico e através de um editor "WYSIWYG". Utiliza controlos gráficos Swing ou controlos gráficos SWT do Eclipse. No nosso caso utilizamos apenas os controlos Swing.

4.5.1 Configuração do Sistema

O primeiro passo para a utilização da aplicação SEED passa pela definição dos ficheiros de configuração. Enquanto os ficheiros de configuração não estiverem correctamente indicados não é possível ter acesso às outras funcionalidades da aplicação. Os ficheiros de configuração necessários são: o ficheiro OWL que define a Ontologia, o ficheiro XSD que define o Modelo Sintáctico, o ficheiro XSL que inclui a definição da transformação dos dados do nível sintáctico para o nível semântico e o ficheiro XML que contém as regras de mapeamento entre o Modelo Sintáctico e a Ontologia. Na figura 4.9 apresentamos a interface que permite definir estes ficheiros.

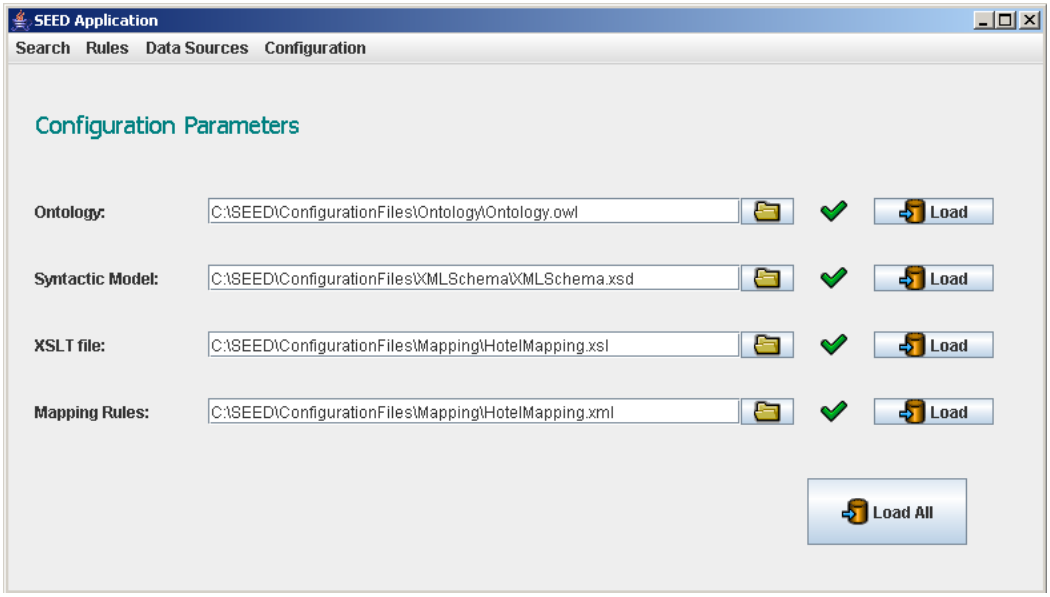

Figura 4.37 – Interface de configuração da aplicação SEED.

4.5.2 Registo das Fontes de Dados

Depois de definirmos todos os ficheiros de configuração, podemos começar a adicionar fontes de dados ao sistema. Para registar uma fonte de dados no sistema é necessário, em primeiro lugar, identificar os parâmetros de acesso e o tipo da fonte de dados. Na figura 4.10 apresentamos a interface de registo de fontes de dados.

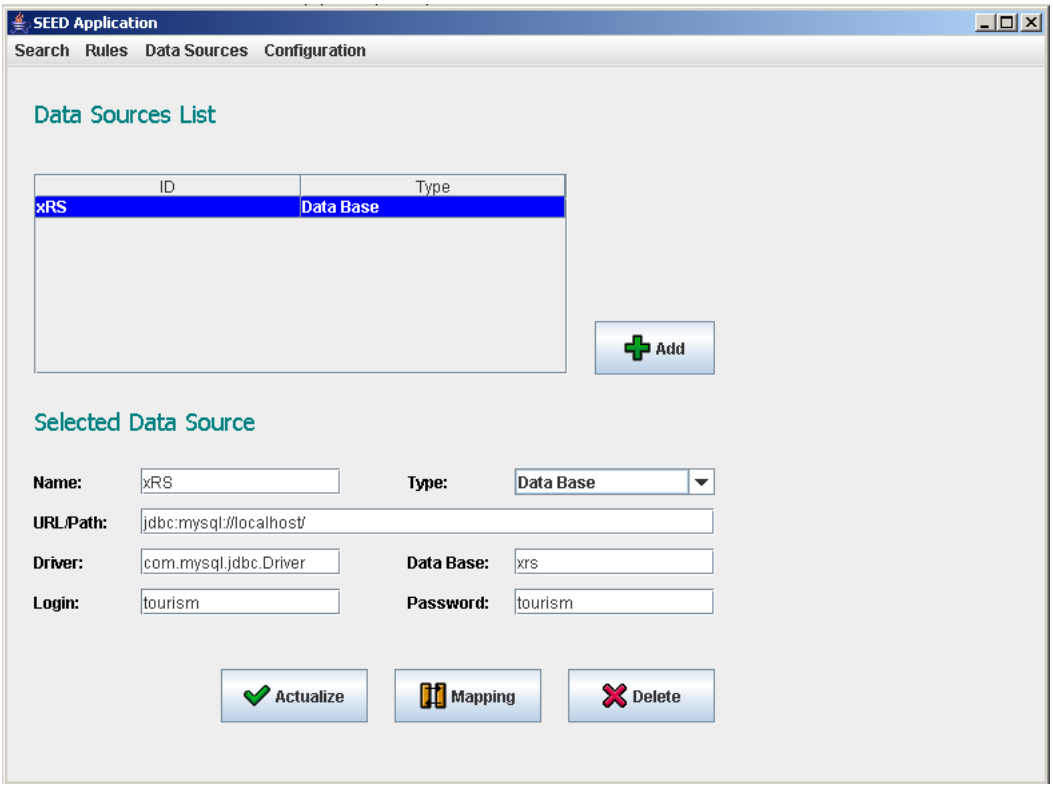

Figura 4.38 – Interface de registo de fontes de dados da aplicação SEED.

Após o registo de uma fonte de dados, é necessário definir os elementos de mapeamento associados a esta. Os elementos de mapeamento definem quais os elementos do Modelo Sintáctico que irão ser preenchidos com os dados da fonte de dados em questão. Para cada elemento de mapeamento é necessário criar os atributos de mapeamento. Para cada atributo de mapeamento é indicado o script que define quais os dados a extrair da fonte de dados mapeada. Na figura 4.11 apresentamos a interface que permite a criação dos elementos de mapeamento e os respectivos atributos.

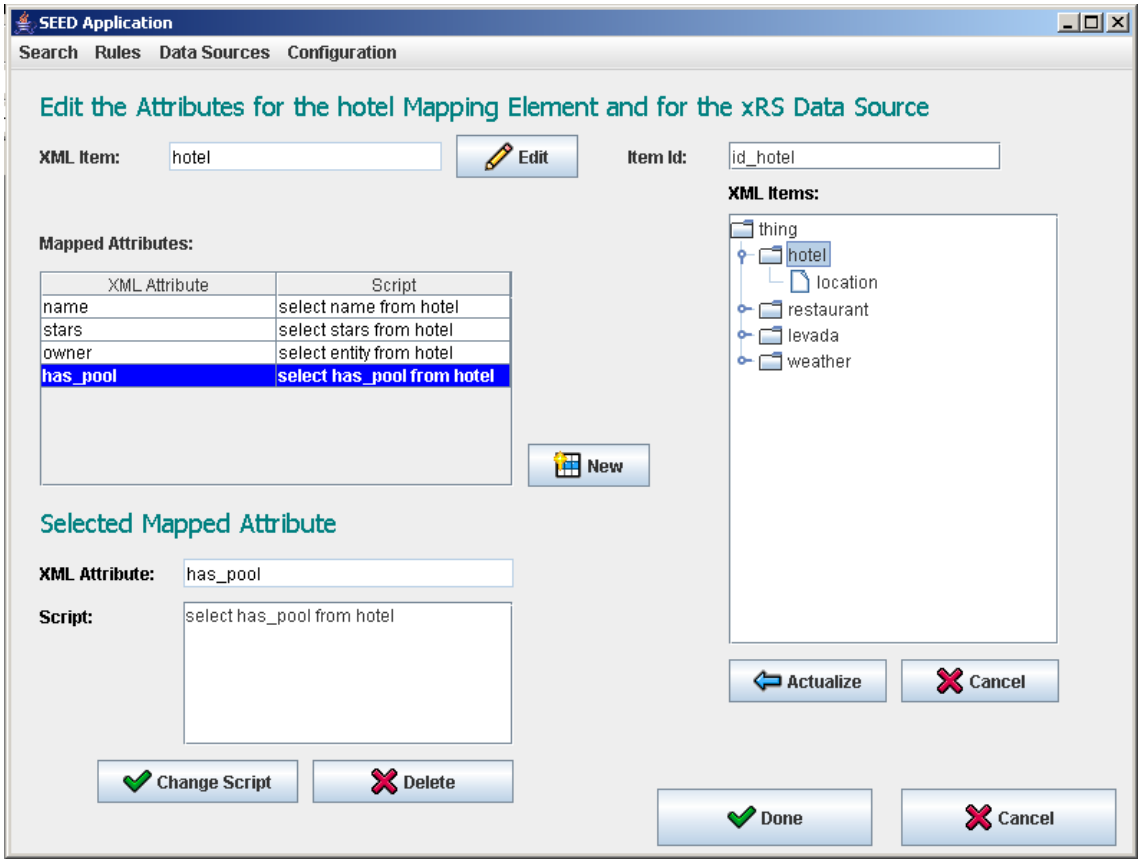

Figura 4.39 – Definição dos elementos de mapeamento na aplicação SEED.

4.5.3 Definição das Regras de Negócio

A aplicação SEED também disponibiliza uma interface para a criação das regras de negócio. As regras de negócio devem ser definidas através da linguagem nRQL e deverão utilizar os elementos presentes na Ontologia. Sempre que uma regra é definida esta é validada. Caso a regra esteja mal definida a aplicação retorna uma mensagem de aviso indicando a existência de um problema na definição da regra. A aplicação não permite que sejam criadas regras inválidas. Uma regra pode ainda ser

definida como activa ou inactiva. Quando inactiva não terá qualquer influência na Ontologia definida. As regras podem ser editadas sem ser necessário recarregar a Ontologia ou reiniciar a aplicação. As alterações nas regas têm efeito imediato depois das mesmas serem confirmadas. Na figura 4.12 apresentamos a interface que permite a criação e edição das regras de negócio.

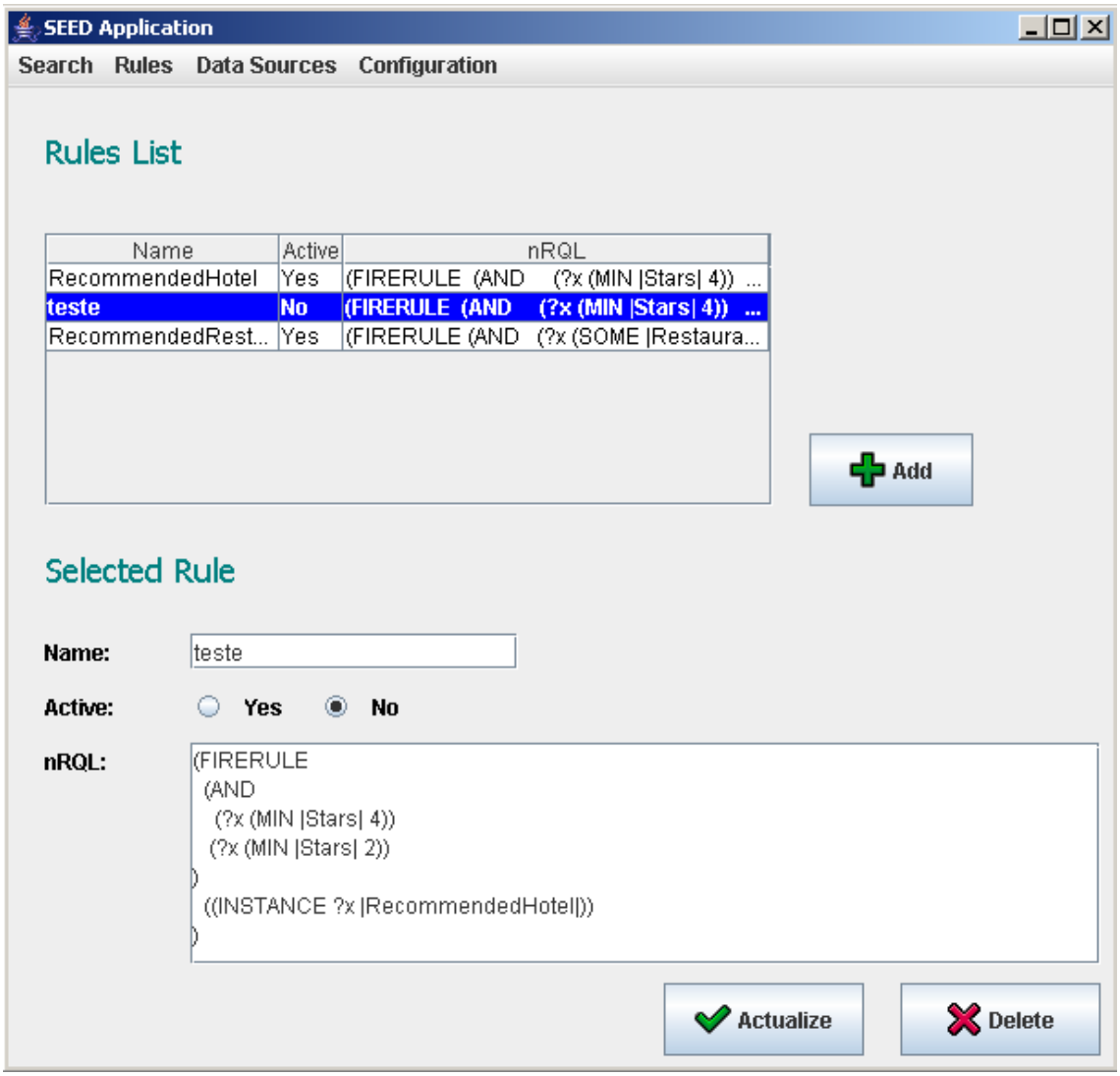

Figura 4.40 – Interface da aplicação SEED para a definição de regras.

4.5.4 Criação e Execução de Pesquisas

Depois de indicarmos os ficheiros de configuração, de registarmos as fontes de dados, de definirmos os elementos de mapeamento e de definirmos as regras de negócio, podemos criar queries de forma a testar o sistema. A aplicação SEED disponibiliza uma interface que permite a definição simples e rápida de queries. As queries são definidas

através da utilização das classes e propriedades presentes na Ontologia definida para o sistema. Na figura 4.13 apresentamos a interface que permite a criação das queries.

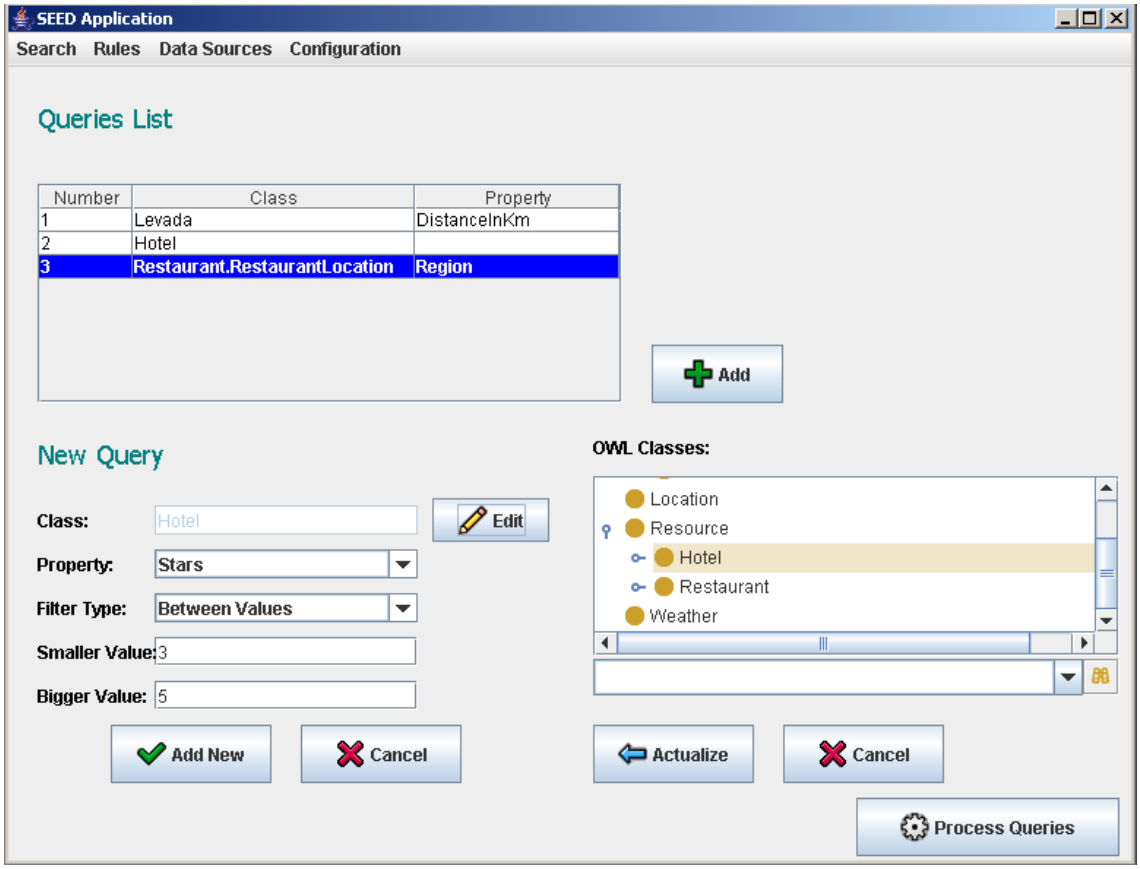

Figura 4.41 – Interface da aplicação SEED para a definição de queries.

Definidas as queries podemos mandar executá-las através do botão "Process Queries". Os resultados são apresentados na forma de árvore. São apresentadas as instâncias, e respectivas propriedades, que respeitem os filtros indicados nas queries. Na figura 4.14 podemos observar a interface que apresenta os resultados referentes a um determinado conjunto de queries.

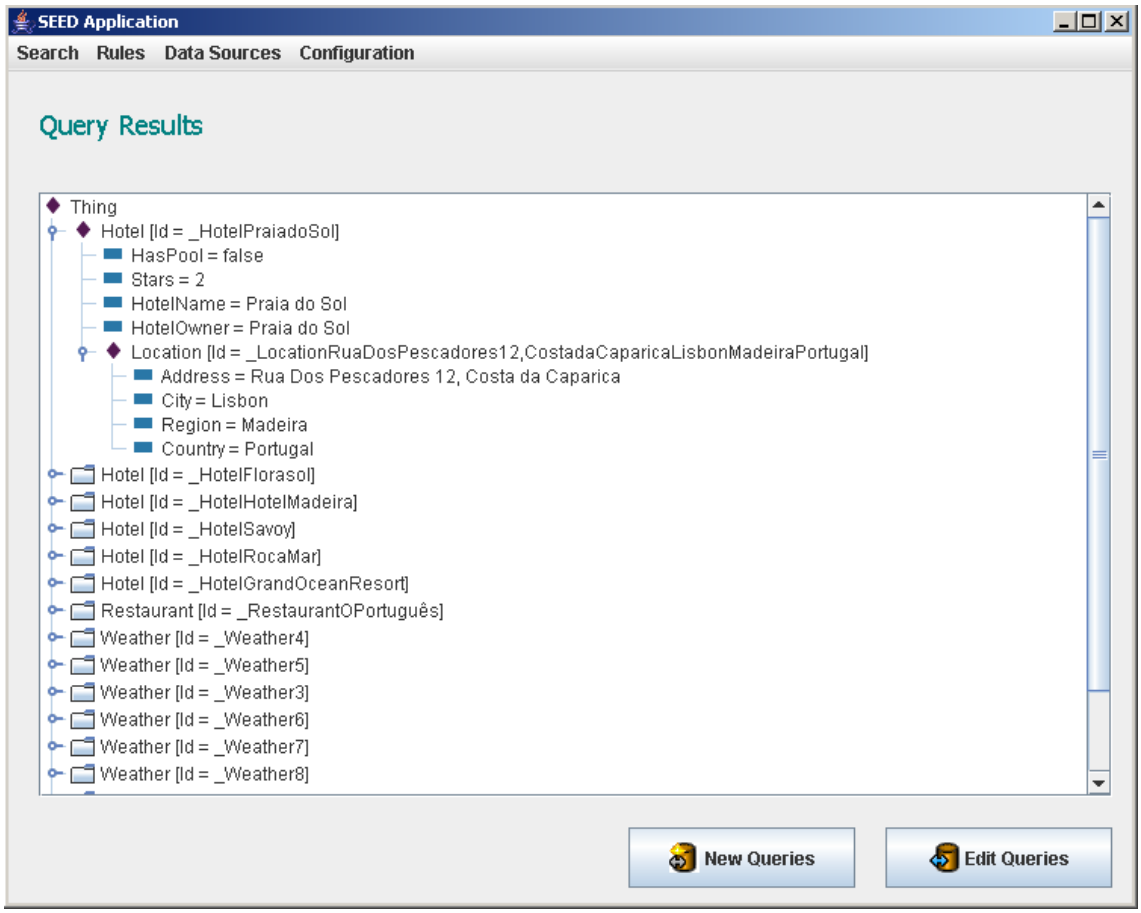

Figura 4.42 – Interface da aplicação SEED que apresenta os resultados das queries.

A partir dos resultados podemos verificar se o sistema está bem configurado. É possível verificar se a Ontologia está bem definida, se as fontes de dados estão correctamente registadas e mapeadas no Modelo Sintáctico, se o processos de transformação de dados e de queries estão bem definidos e se as regras estão a ser interpretadas da forma esperada. Se não obtivermos as instâncias esperadas, como resultado às queries definidas, significa que algo na configuração não está correcto.

A aplicação SEED também poderá ser muito útil na manutenção do sistema. Caso seja necessário alterar alguns dos parâmetros de configuração, poderemos testar novamente o sistema aplicando um novo conjunto de queries. Por exemplo, se pretendermos adicionar uma nova fonte de dados ao sistema, podemos utilizar a aplicação para registar e mapear a nova fonte de dados. Através da interface de pesquisa, podemos criar queries para verificar se estamos a obter dados provenientes da nova fonte de dados registada.

5 Cenário de Utilização do Sistema

5.1 Descrição do Cenário

Para exemplificar uma possível utilização do sistema implementado, vamos apresentar um exemplo. Neste exemplo, pretendemos integrar vários produtos turísticos com atracções turísticas referentes ao destino turístico Madeira. Com esta integração pretendemos adicionar valor aos produtos turísticos oferecidos através da associação a atracções turísticas. Desta forma os possíveis turistas poderão obter os produtos turísticos que poderão adquirir de forma a poderem desfrutar da melhor forma a sua visita ao destino Madeira.

A Região Autónoma da Madeira é constituída por um conjunto de ilhas que se situam no oceano Atlântico e pertencentes a Portugal. As principais atracções turísticas desta região são as Levadas e as boas condições climatéricas. A temperatura é amena durante a maior parte do ano e o Sol está quase sempre presente. Levadas são canais de irrigação. Começaram a ser criados no século XVI e eram utilizados para transportar água para regiões agrícolas onde não existiam recursos hídricos. Actualmente oferecem bonitas zonas para passeios pedestres que são muito apreciados pelos turistas. Uma das principais razões da beleza destes trilhos deve-se ao facto de estes passarem por entre as altas montanhas da ilha da Madeira e pela floresta Laurissilva. Laurissilva é a classificação para a floresta endémica da ilha da Madeira, e pertence ao património da humanidade desde 1999.

Neste exemplo iremos apresentar apenas parte da estrutura de dados que seria necessária para a implementação de uma aplicação de integração na área do turismo, pois o objectivo é manter o exemplo simples e de fácil compreensão. Os dados referentes à disponibilidade e tarifas dos produtos turísticos não serão apresentados. Tanto a definição da disponibilidade como o cálculo das tarifas dos produtos turísticos caracterizam-se por serem complexos e por serem compostos por estruturas de dados extensas que iriam complicar a compreensão do cenário. Não obstante, estes dados são essenciais para a criação de uma aplicação que permita ao turista obter toda a informação necessária à escolha dos produtos a reservar. Desta forma, iremos integrar apenas os conceitos de hotel, restaurante, levada, condição climatérica e localização.

A configuração da arquitectura irá processar-se pela seguinte ordem:

1) Definição da Ontologia.

2) Definição do modelo de dados sintácticos.

3) Mapeamento do modelo de dados sintáctico com a Ontologia.

4) Geração do documento de transformação entre os dados sintácticos e os dados semânticos.

5) Registo das fontes de dados a serem integradas.

6) Definição dos atributos das fontes de dados e respectivo mapeamento com o modelo de dados sintáctico.

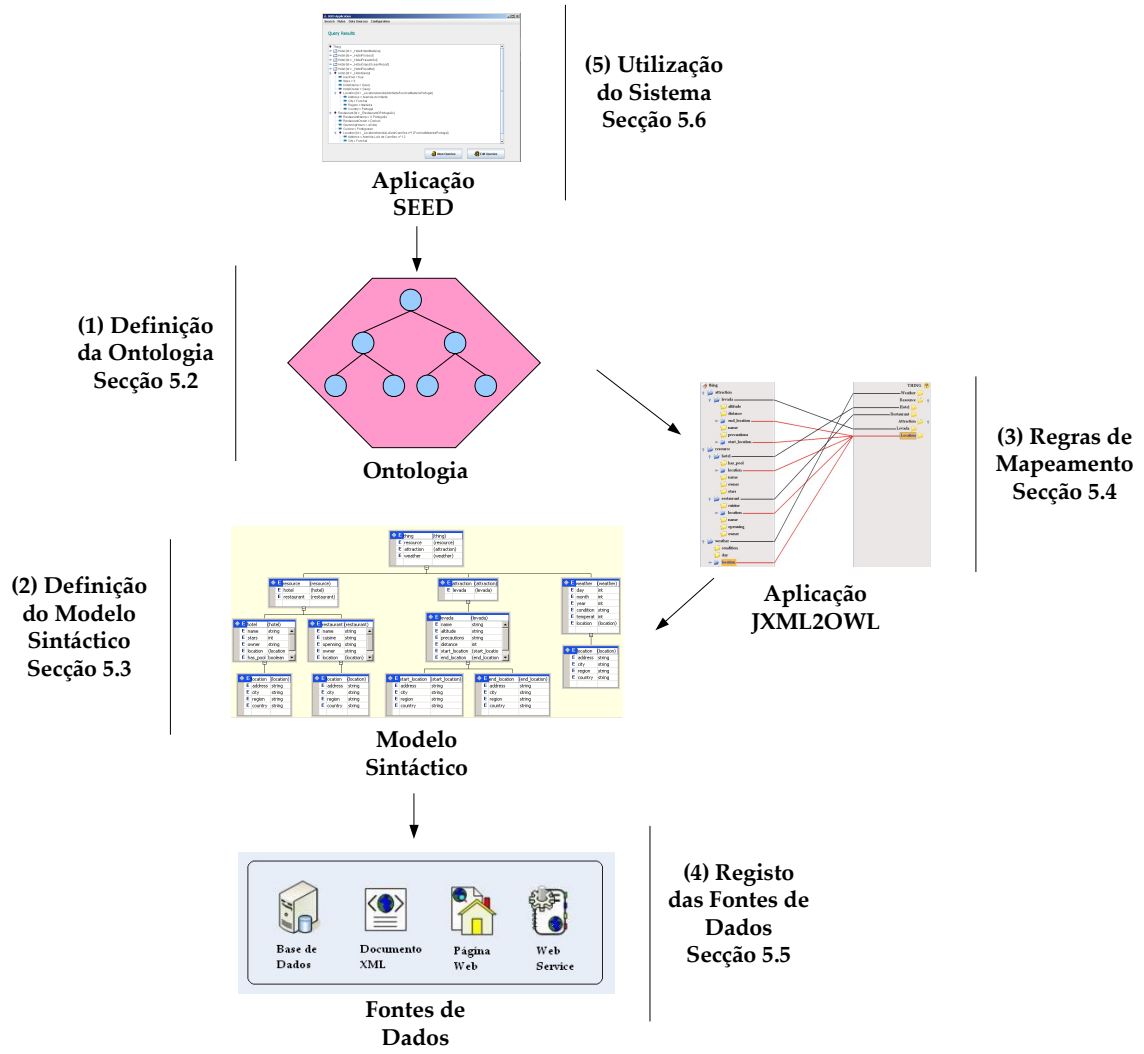

Figura 5.43 – Ordem de acções a realizar no exemplo.

Depois de termos o sistema configurado, iremos apresentar alguns exemplos de regras que poderão ser definidas bem como de algumas queries. Neste exemplo, algumas imagens aparecem com aspecto de recortadas. Com esta representação pretendemos denotar que estas são apenas parte de uma interface da aplicação SEED. A ordem de acções a realizar neste exemplo é apresentado na figura 5.1.

5.2 Modelo Semântico

A configuração do sistema começa com a definição da Ontologia. É importante começar pela definição da Ontologia antes de tratar das fontes de dados a integrar. Sem conhecer as fontes de dados, evitamos o erro de criar uma Ontologia baseada apenas nas fontes de dados a integrar. Neste exemplo a Ontologia deverá ser geral e definir o domínio sem se basear em modelos de dados proprietários.

As classes definidas para a Ontologia a ser utilizada estão representadas na figura 5.2. Outras classes poderiam ser definidas, tais como Disponibilidade ou Tarifas, mas vamos limitarmo-nos às classes apresentadas nesta figura.

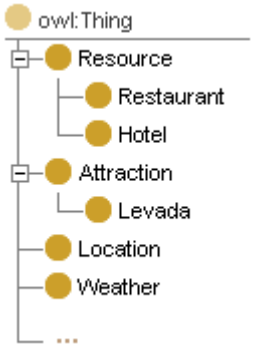

Figura 5.44 – Hierarquia de classes da Ontologia.

Foram definidas duas classes distintas para representar as atracções turísticas e os recursos turísticos. Esta opção torna claro a separação entre os dois conceitos. Os recursos turísticos são representados pela classe "Resource". Como recursos turísticos foram definidas duas subclasses, "Restaurant" e "Hotel", que representam respectivamente restaurantes e hotéis. As levadas estão representadas através da classe "Levada", que se encontra definida como uma subclasse da classe "Attraction", representando efectivamente as atracções turísticas. No caso das condições climatéricas, optamos por não considerar este conceito como uma atracção turística pois nem sempre poderão ser consideradas como tal e diferem das atracções turísticas por serem variáveis. Assim, estas são representadas pela classe "Weather". A classe "Location" refere-se a uma determinada localização. Esta classe é utilizada apenas como propriedade das outras classes definidas.

Para cada uma das classes apresentadas foram criadas propriedades. Também aqui o número de propriedades para cada uma das classes é reduzido para manter o exemplo simples. No entanto, muitas outras propriedades poderiam ser definidas de forma a caracterizar melhor cada uma das classes. Na figura 5.3 apresentamos o Modelo de Classes UML que define a Ontologia criada. Neste modelo são mostradas quais as propriedades definidas para cada uma das classes e quais as relações entre as classes.

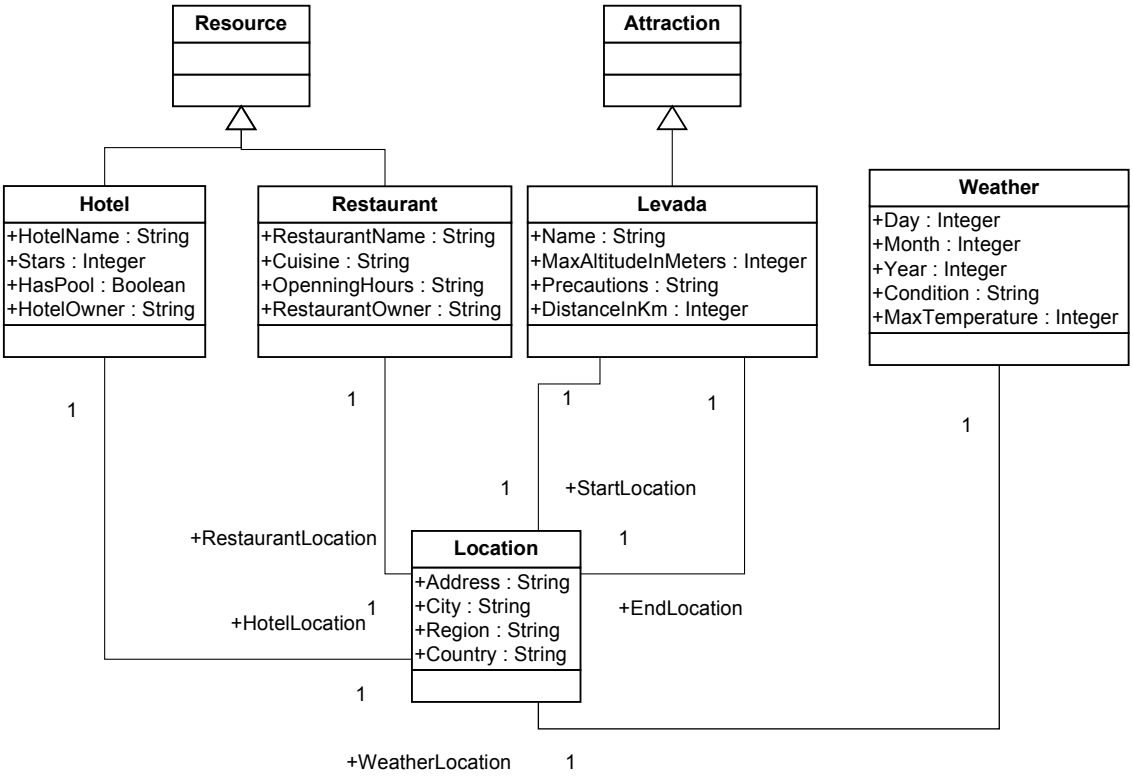

Figura 5.45 – Modelo de Classes UML para a Ontologia.

Através do modelo de classes podemos observar que a classe "Location" encontra-se associada a todas as outras classes. Esta classe é essencial para a Ontologia pois é através desta que iremos relacionar as outras classes. Por exemplo, podemos associar um hotel com uma levada através da região onde se encontra o hotel e da região onde começa a levada. Neste caso a associação é realizada através da propriedade "Region" da classe "Location". Podemos observar também que no caso da classe "Weather" a indicação da data da previsão meteorológica está dividida em três diferentes propriedades, "Day", "Month" e "Year". Isto deve-se à impossibilidade de utilização

do tipo de dados "Date". Esta restrição é necessária pois o motor de inferência utilizado pelo sistema não suporta o tipo de dados "Date".

Além das classes anteriormente apresentadas, serão acrescentadas mais duas classes à Ontologia. Estas classes terão a função de representar produtos turísticos recomendados pelos administradores da aplicação. As duas classes a acrescentar são a classe "RecommendedHotel" e a classe "RecommendedRestaurant", que representam respectivamente os hotéis e os restaurantes recomendados. Estas classes podem ser utilizadas não só para ajudar o turista a escolher mais rapidamente os produtos turísticos mas também para promover produtos turísticos que tenham maior interesse para a entidade que gere a aplicação. A relação destas duas classes com as restantes classes está definida na figura 5.4.

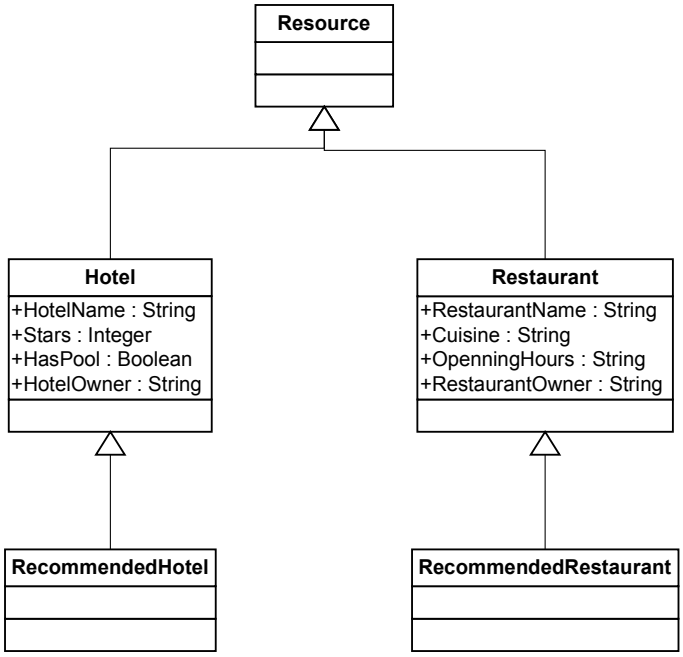

Figura 5.46 – Modelo de Classes UML para as classes "Resource".

Os parâmetros que definem tanto um hotel recomendado como um restaurante recomendado podem variar. Os contratos com as entidades que gerem os recursos turísticos podem mudar e as tendências de mercado podem variar. Este facto leva a que a definição de um hotel recomendado ou de um restaurante recomendado seja definido através de regras semânticas. Desta forma é possível alterar a sua definição em tempo real e sem necessidade de recompilar ou reiniciar o sistema.

5.3 Modelo Sintáctico

O modelo sintáctico é utilizado como mediador entre a camada semântica e os modelos das fontes de dados. Neste exemplo, optamos por utilizar um modelo sintáctico à imagem da Ontologia definida ao invés de aproximar o modelo sintáctico às fontes de dados a integrar. Desta forma, retiramos complexidade ao processo de mapeamento e adicionamos complexidade à integração das fontes de dados. Esta opção foi realizada tendo em conta que as fontes de dados terão modelos de dados pouco variáveis e supondo que a entidade integradora tem um bom conhecimento sobres as fontes de dados a integrar.

O modelo de dados sintáctico é definido através da linguagem XML Schema. O modelo encontra-se apresentado na figura 5.5.

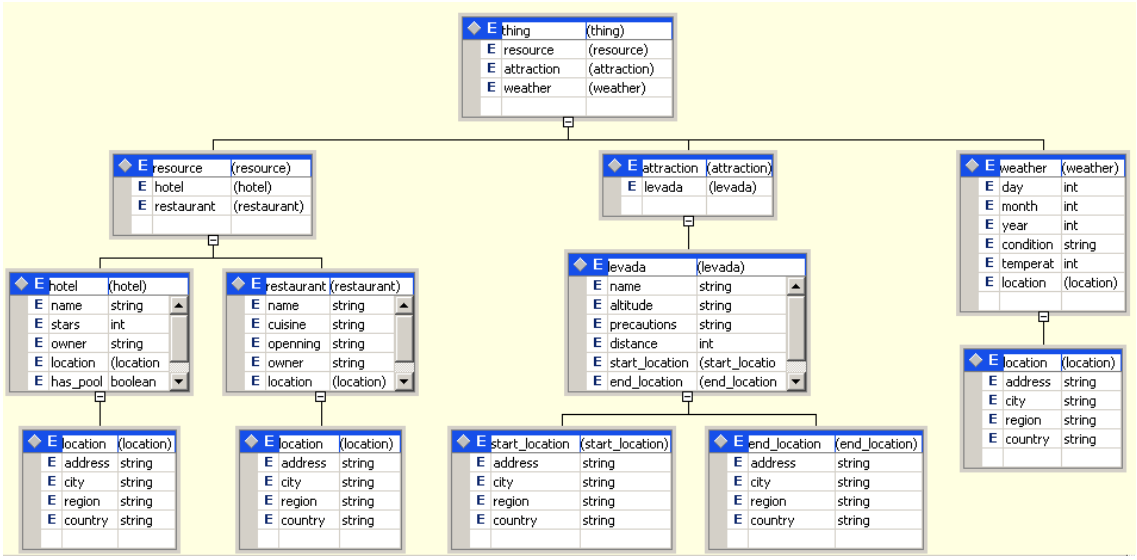

Figura 5.47 – Modelo de dados sintáctico.

De notar que no modelo sintáctico não existe o conceito de herança. No nível sintáctico a herança é substituída por uma simples relação de classe e propriedade. Não é possível também o reaproveitamento de definição de estruturas. No modelo sintáctico tivemos de repetir o conceito "location" várias vezes. Outra das diferenças que notamos é a não presença de uma representação das classes "RecommendedHotel" e "RecommendedRestaurant". Estas classes são definidas através das regras definidas ao

nível semântico logo, não existe necessidade de estarem representadas ao nível sintáctico. No modelo sintáctico, só existe necessidade de representar as classes que serão utilizadas no mapeamento com as fontes de dados.

O modelo sintáctico poderá ser definido num qualquer editor de XML Schema. Neste exemplo utilizamos o editor de XML Schema existente na ferramenta de desenvolvimento Visual Studio .Net (Visual Studio Developer Center, 2007).

5.4 MAPEAMENTO

Depois de definirmos qual a Ontologia e o modelo sintáctico a utilizar, temos de mapear estes dois modelos. O mapeamento é realizado utilizando a ferramenta "XML2OWL Mapper". Na figura 5.6 está representado o mapeamento realizado entre os dois modelos.

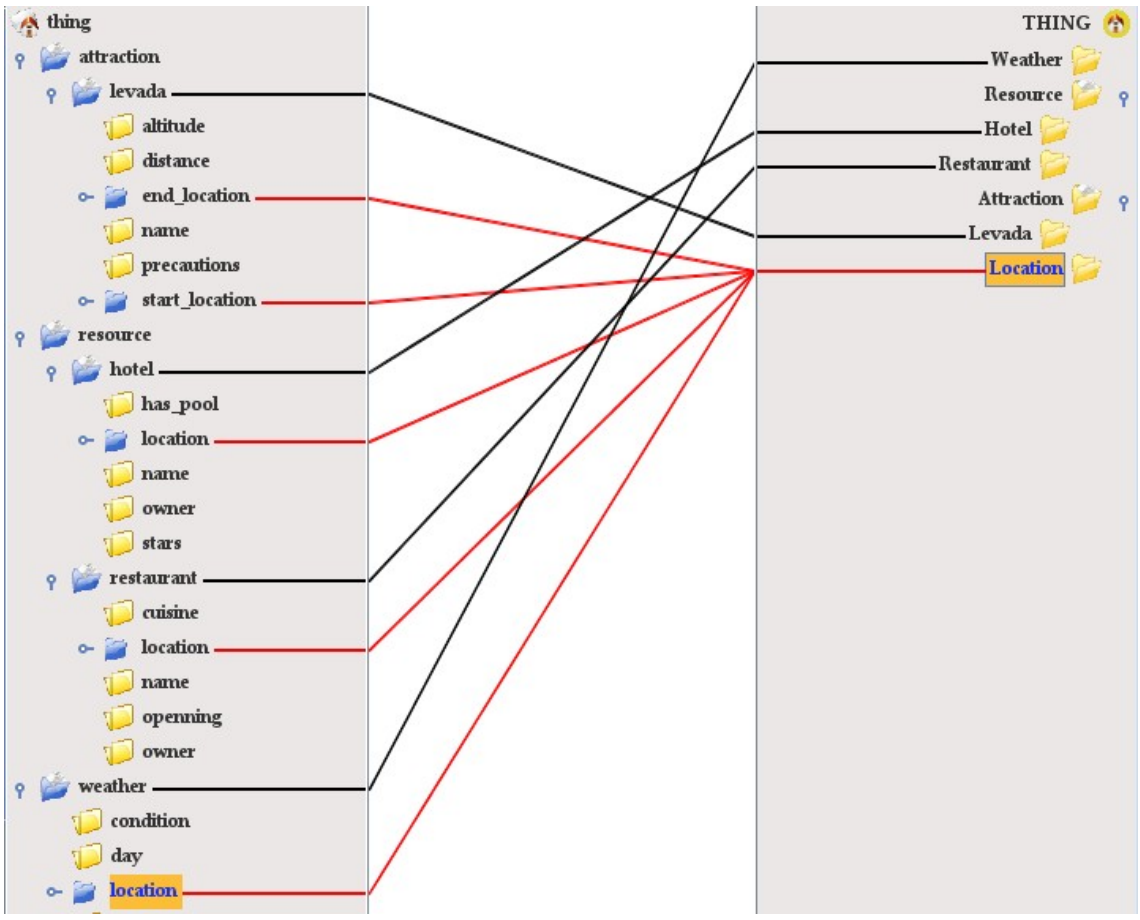

Figura 5.48 – Mapeamento entre a Ontologia e o modelo sintáctico.

Como o modelo sintáctico foi baseado na Ontologia, o mapeamento é praticamente directo. Primeiro são mapeados os elementos XML com as correspondentes classes OWL. As únicas classes que não são mapeadas são as classes "Resource" e "Attraction". Apesar de existir um elemento XML correspondente a estas, o mapeamento não é necessário pois tratam-se de classes abstractas que nunca irão ser preenchidas por dados das fontes de dados externas. Para cada classe OWL ainda são

mapeadas as propriedades de tipo de dados e as propriedades objecto. As propriedades tipo de dados irão corresponder a atributos XML. Por exemplo, a propriedade "Stars" é mapeada ao atributo "stars" referente ao elemento XML "hotel". As propriedades objecto irão corresponder a elementos XML. Por exemplo, a propriedade "HotelLocation" é mapeada ao sub-elemento "location" referente ao elemento "hotel".

Através do mapeamento definido na aplicação "XML2OWL Mapper" é criado um documento XSLT. Este documento será necessário para a transformação dos dados XML em instâncias OWL. Para além do documento XSLT, também é criado um documento XML que inclui as regras de mapeamento definidas. Ambos os documentos deverão ser guardados para serem indicados na configuração do sistema.

A partir deste momento temos definido todos os dados necessários para a configuração do sistema. Estes dados deverão ser indicados no menu configuração da aplicação SEED. Na figura 5.7 apresentamos a interface de configuração e a indicação dos respectivos ficheiros de configuração.

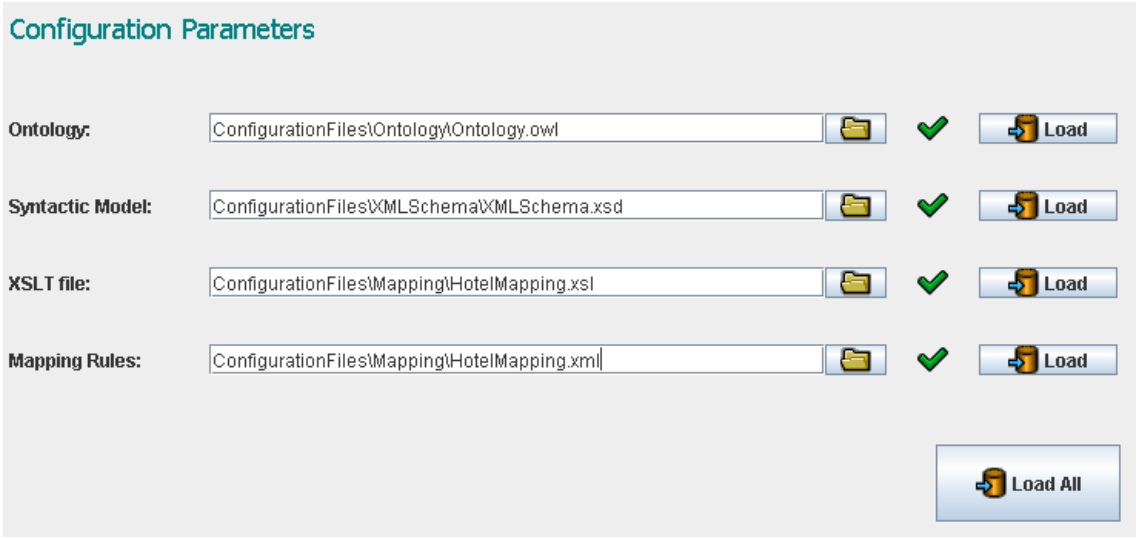

Figura 5.49 – Interface de configuração da aplicação SEED.

Os parâmetros necessários para a configuração são:

- **Ontology:** Localização do documento OWL que contém a Ontologia definida para o sistema.

- **Syntactic Model:** Localização do documento XML Schema onde se encontra definido o modelo de dados sintáctico.

- **XSLT file:** Localização do documento XSLT obtido através da aplicação "XML2OWL Mapper" e que define a transformação dos dados no formato XML em instâncias da Ontologia.

- **Mapping Rules:** Localização do documento XML que contém as regras de mapeamento definidas através da aplicação "XML2OWL Mapper".

5.5 Fontes de Dados

Depois de configurado o sistema, é necessário adicionar as fontes de dados que irão disponibilizar os dados com que o sistema irá trabalhar. Deveremos procurar fontes de dados que contenham informação relacionada com a Ontologia definida. Para o nosso exemplo, temos de encontrar fontes de dados que disponibilizem informação sobre atracções da região turística Madeira, bem como de produtos turísticos relacionados com estas atracções. Necessitamos também de fontes de dados que nos dêem informação sobre a situação climatérica desta região. De seguida apresentamos as fontes de dados que escolhemos para integrar no nosso sistema e descrevemos como estas fontes deverão ser registadas e quais os elementos de mapeamento definidos.

5.5.1 ISNOVA

ISNOVA é um projecto que se enquadra na iniciativa europeia INTERREG IIIB SUDOE (ISNOVA, 2005). Neste projecto participaram as Ilhas Baleares, as Ilhas Canárias e a Madeira. Cada região desenvolveu plataformas tecnológicas na área do Bioturismo. Como é óbvio, vamos centrar-nos na plataforma implementada pela região da Madeira.

A plataforma implementada pela região Madeira destina-se à caracterização de atracções turísticas. A plataforma caracteriza-se pela forma flexível como são definidas as estruturas de dados referentes às atracções. Antes de começarmos a criar as atracções, é necessário definir uma hierarquia de classificações de atracções para permitir uma navegação e pesquisa de atracções mais eficaz. Para cada classificação é necessário definir um esquema de dados. O esquema é definido através da linguagem XML Schema (XML Schema, 2007) e é utilizado para definir as estruturas referentes às atracções associadas às classificações. Cada classificação terá um determinado esquema de dados, e todas as atracções associadas a uma determinada classificação deverão respeitar o esquema de dados desta. O esquema de cada classificação poderá ser alterado futuramente de forma a adaptar as atracções aos novos requisitos, e sem ser necessário reiniciar ou recompilar a plataforma. As estruturas de dados das atracções são definidas em XML (XML, 2007), permitindo assim uma validação automática

quando comparadas com os esquemas de dados definidos nas classificações associadas.

Sobre a plataforma ISNOVA foi criado o Web Site "Levadas e Veredas da Madeira", que disponibiliza informação sobre as levadas da Madeira (ISNOVA Web Site, 2006). A plataforma disponibiliza uma interface de comunicação com sistemas externos. A interface é definida através de um conjunto de Web Services (Web Services Activity, 2002) que permitem a obtenção de informação referente às atracções existentes no sistema. As atracções poderão ser pesquisadas a partir de filtros sobre as propriedades que as constituem e a informação é retornada sobre a forma de mensagens SOAP. Através destas mensagens, poderemos extrair a informação sobre as atracções no formato XML.

Devido às limitações existentes actualmente no sistema para a integração com Web Services, a integração com a plataforma ISNOVA foi realizada através de documentos XML. Foi implementado um processo que diariamente invoca o Web Services e que obtém a informação sobre todas as levadas existentes no ISNOVA. Este ficheiro é guardado localmente para depois ser integrado com o sistema. O documento XML tem a estrutura igual ao seguinte exemplo:

```
<?xml version="1.0" encoding="ISO-8859-2"?>
<root>
   <Classifications>
      <Levadas>
         <Levada>
            <title>Levada dos Piornais</title>
            <precaution>Use appropriate footwear</precaution>
            <distance>5</distance>
            <maxalt>300</maxalt>
            <start>Piornais</start>
            <end>Câmara de Lobos</end>
            <start_city>Funchal</start_city>
            <end_city> Câmara de Lobos</end_city>
            <country>Portugal</country>
            <region>Madeira</region>
         </Levada>
         <Levada>
            <title>Levada do Furado</title>
            <precaution>Tunnels en route, carry a
                        torch
            </precaution>
            <distance>11</distance>
            <maxalt>870</maxalt>
            <start>E.R. 103 (Ribeiro Frio)</start>
            <end>E.R. 102 (Portela)</end>
            <start_city>Santana</start_city>
            <end_city>Machico</end_city>
```

```
            <country>Portugal</country>
            <region>Madeira</region>
         </Levada>
      </Levadas>
   </Classifications>
</root>
```

O processo de obtenção da informação em XML, cria uma estrutura de dados que apenas inclui a informação necessária para a integração. Todas as propriedades que não estavam representadas na Ontologia foram eliminadas. As tags XML "region" e "country" foram acrescentadas de forma a facilitar a integração com o modelo sintáctico definido para o sistema. Como a plataforma ISNOVA apenas inclui levadas da região Madeira estas duas tags terão sempre os mesmos valores, region="Madeira" e country="Portugal".

Para integrarmos a plataforma ISNOVA com o nosso sistema, temos primeiro de a registar como uma fonte de dados utilizando a aplicação SEED. Se dermos o nome "levadas" ao ficheiro XML que inclui a informação obtida da plataforma ISNOVA, então o registo desta fonte de dados externa deverá ser definida como as configurações apresentadas na figura 5.8.

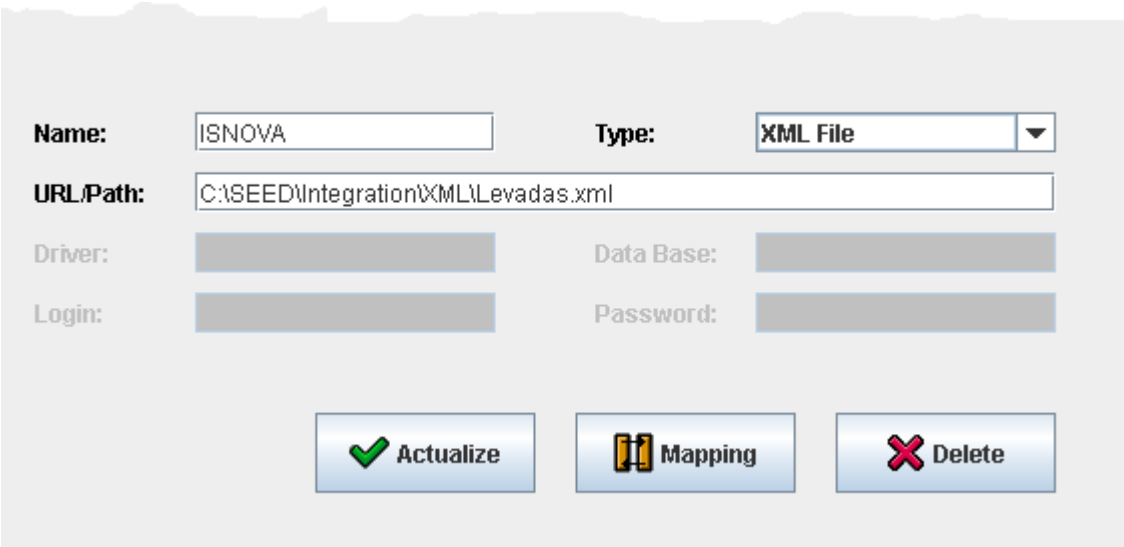

Figura 5.50 – Registo da fonte de dados ISNOVA através de aplicação SEED.

Depois de registada a fonte de dados, é necessário definir os elementos de mapeamento. A fonte de dados ISNOVA irá preencher o elemento XML "Levada" definido no modelo sintáctico. Assim é necessário associar às propriedades do elemento XML "Levada" os scripts de obtenção da informação existente na fonte de dados ISNOVA. Como se trata de uma fonte de dados XML, os scripts deverão ser indicados através da linguagem XQuery (Boag, Chamberlin, Fernandez, Florescu,

Robie, e Simeon, 2007). Na figura 5.9 apresentamos a criação do elemento de mapeamento "levada" e do atributo de mapeamento "name", utilizando a aplicação SEED. Na criação do atributo de mapeamento "name", podemos verificar que o script para obtenção dos dados é uma instrução em XQuery .

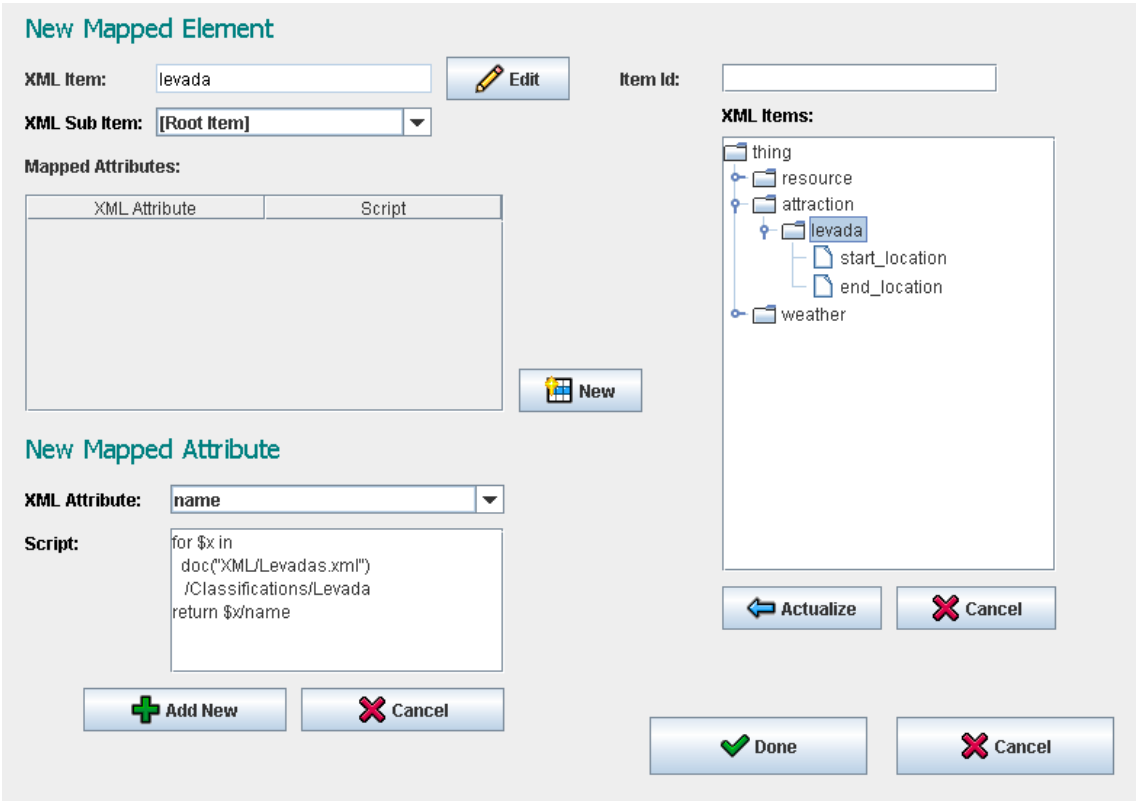

Figura 5.51 – Definição dos atributos de mapeamento para a fonte de dados ISNOVA.

Deverão ser criadas as outras instruções XQuery para os restantes atributos pertencentes ao elemento XML "Levadas" ("altitude", "precautions" e "distance"). Também deverão ser criados elementos de mapeamento para os sub-elementos XML "start_location" e "end_location". Estes elementos de mapeamento deverão ser indicados como sub itens XML do elemento de mapeamento "Levadas". Depois de serem criados todos os elementos e atributos de mapeamento, a integração com a fonte de dados ISNOVA fica completa. A partir deste momento é possível obtermos informação referente às levadas a partir do sistema.

5.5.2 xRS

xRS (Plataforma xRS, 2007) é uma plataforma utilizada para a definição de oferta turística de forma a permitir a reserva de produtos turísticos on-line e em tempo real. Esta plataforma foi implementada pela empresa Expedita e foi utilizada para a

implementação de projectos para importantes empresas na área do turismo da Região Autónoma da Madeira.

Na plataforma xRS, através do conceito de produto podemos representar qualquer tipo de produto turístico tais como viagens, eventos, aluguer de automóvel ou reserva de quartos. Cada produto está associado a um determinado recurso. Um recurso pode representar um hotel, um avião ou um comboio. Através do conceito de condição de venda podemos definir a disponibilidade e as tarifas referentes a um produto turístico. Uma das mais interessantes funcionalidades que esta plataforma oferece é a possibilidade de definir pacotes dinâmicos. Os pacotes dinâmicos permitem a reserva de uma determinada combinação de produtos, chamados de pacotes, os quais são seleccionados pelo cliente, ou agente turístico, quando é definida a reserva. Os pacotes deverão respeitar certas regras pré-definidas pelos operadores turísticos.

Tal como acontece com as atracções na plataforma ISNOVA, os recursos e os produtos encontram-se associados a uma ou mais classificações. As classificações são definidas na forma de hierarquias, sendo que os mais altos níveis são utilizados para classificar os recursos e os níveis mais baixos utilizados para classificar os produtos. Outra característica interessante na plataforma xRS é a forma como são definidas as propriedades dos recursos. Como um recurso pode representar um qualquer recurso turístico, as propriedades têm de ser criadas de maneira flexível de forma a permitir a caracterização de diferentes tipos de recursos. Os recursos são caracterizados através da associação a propriedades pré-definidas. Para cada associação recurso/propriedade é definido um determinado valor. A ideia é podermos definir diferentes propriedades para recursos, dependentemente da classificação a que estes estão associados. Por exemplo, para os recursos classificados como "Hotel" temos de definir propriedades como o número de quartos ou o número de estrelas. Para recursos classificados como "Avião" temos de definir outro grupo de propriedades como número de lugares ou velocidade máxima.

A integração com a plataforma xRS será realizada através do acesso directo à base de dados que serve de suporte à informação persistente da plataforma. Este tipo de integração é pouco flexível porque a estrutura da bases de dados é geralmente mais complexa e menos estável que as estruturas definidas através de documentos XML. A vantagem deste tipo de integração é uma permitir uma boa performance. Sem existir qualquer tipo de transformação entre a base de dados e o módulo de integração, é

possível a utilização do motor de base de dados para obter a informação desejada. Devido à grande utilização dos sistemas de base de dados, estes possuem um motor de queries muito desenvolvido e com muita boa performance. No entanto, devido à complexidade da estrutura de tabelas existente na base de dados da plataforma xRS, foi necessário criar um conjunto de views para simplificar a integração. Além de funcionarem como um nível de abstracção sobre a estrutura da base de dados, estas views tornam o processo de integração mais imune às alterações produzidas nas estruturas de dados internas à plataforma. O modelo das views de integração criadas está representado na figura 5.10.

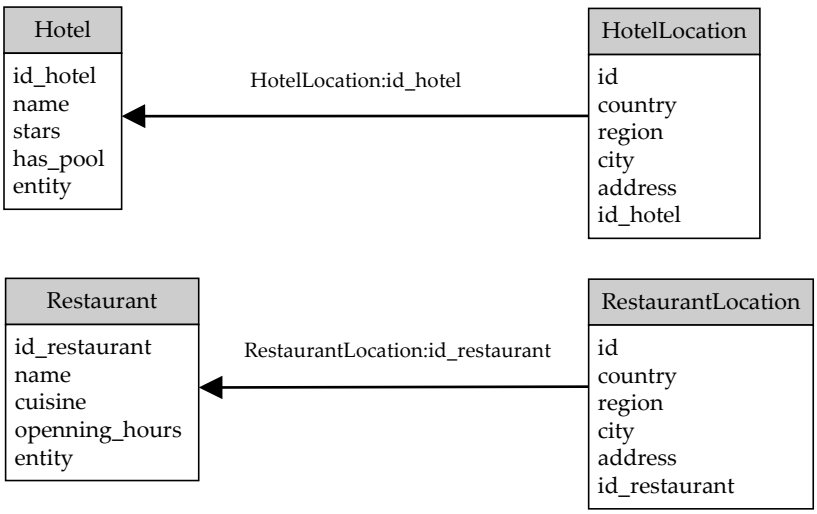

Figura 5.52 – Views de integração para a Plataforma xRS.

O registo de fontes de dados do tipo base de dados implica a indicação de mais informação que no caso de fontes de dados do tipo XML. O registo da fonte de dados para a plataforma xRS está apresentado na figura 5.11.

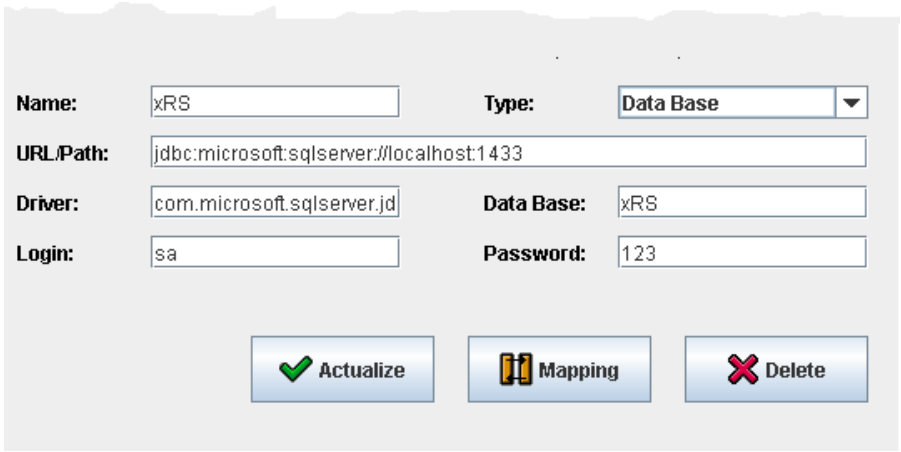

Figura 5.53 – Registo da fonte de dados xRS utilizando a aplicação SEED.

A plataforma xRS utiliza o motor de base de dados SQL Server. Para o registo desta fonte de dados temos de utilizar o driver correspondente a este motor (com.microsoft.sqlserver.jdbc.SQLServerDriver). O driver pode ser obtido directamente através da página Web da Microsoft (http://www.microsoft.com/downloads/details.aspx?FamilyID=07287B11-0502-461A-B138-2AA54BFDC03A&displaylang=en).

A fonte de dados xRS irá disponibilizar informação referente aos elementos XML "hotel" e "restaurant" do modelo sintáctico. Desta forma, temos de criar dois elementos de mapeamento, um para os hotéis e outro para os restaurantes. Os atributos de mapeamento deverão ser definidos utilizando instruções de SQL para a obtenção dos dados. Na figura 5.12 apresentamos o exemplo da definição do atributo de mapeamento para o atributo "stars" referente ao elemento "hotel".

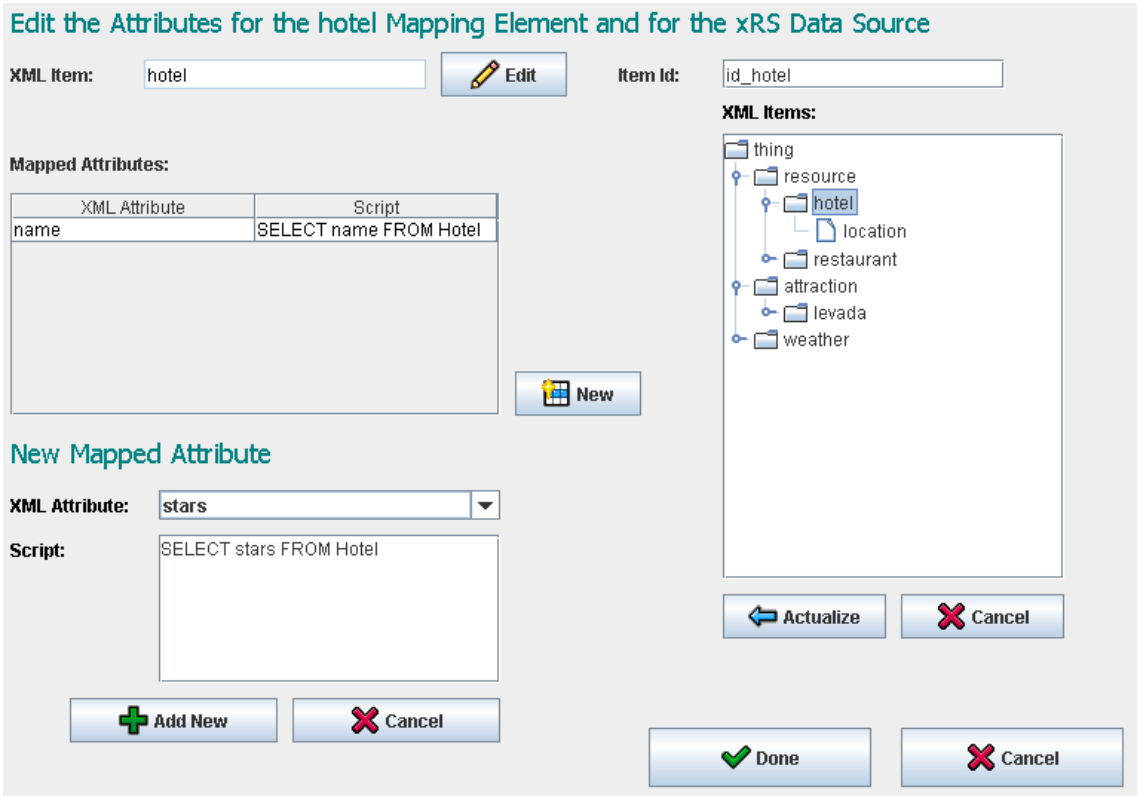

Figura 5.54 – Definição dos atributos de mapeamento para a fonte de dados xRS.

Depois da definição de todos os elementos de mapeamento e dos respectivos atributos de mapeamento, finalizamos a integração da fonte de dados xRS no nosso sistema.

5.5.3 Weather.com

Weather.com é uma página Web que disponibiliza informação sobre as condições climatéricas para todas as cidades do mundo. Iniciou-se em 1982 como um canal de televisão, inteiramente dedicado à meteorologia que transmitia 24 horas por dia e 7 dias por semana. Expandiu o seu negócio por outros meios de comunicação tais como a Internet e os telemóveis.

Através da página Web do Weather.com, podemos obter informações meteorológicas tais como a temperatura, a precipitação ou uma descrição geral sobre as condições climatéricas. Podemos obter as previsões sobre as condições climatéricas com dez dias de antecedência. A informação apresentada é actualizada ao minuto.

A Weather.com disponibiliza através da sua página Web o acesso à informação através da tecnologia RSS (RSS Specifications, 2007). RSS é um método que utiliza o XML para distribuir conteúdo Web referente a uma determinada página Web. Poderíamos utilizar o XML existente no RSS disponibilizado pelo Weather.com. O problema é que o RSS não disponibiliza a previsão meteorológica com dez dias de antecedência, disponibiliza apenas a previsão para o dia actual. No domínio do turismo é importante dispormos de informação sobre as condições climatéricas com alguns dias de antecedência, pois o turista geralmente não planeia as suas férias para o dia seguinte. Não existindo outra alternativa, a integração com o sistema Weather.com terá de ser realizada através do acesso à informação disponibilizada pela página Web.

Acedendo directamente à página Web, conseguimos obter a informação referente às previsões das condições climatéricas para dez dias, incluindo o dia actual. Para cada dia existe uma página Web que disponibiliza esta informação. Por exemplo, para obter informação meteorológica sobre a região Madeira para o dia seguinte ao dia actual (amanhã), temos de aceder à página:

http://www.weather.com/outloqqqok/travel/businesstraveler/wxdetail/POXX0081?dayNum=1

O parâmetro "dayNum" existente no URL indica qual o número do dia a considerar. O número do dia é sempre em relação ao dia actual. Para o dia actual devemos colocar "dayNum= 0". Para os dias seguintes, vamos aumentando o valor do parâmetro até chegarmos ao décimo dia onde devemos indicar "dayNum= 9". Na figura 5.13

mostramos parte da página referente à informação meteorológica para um determinado dia e para a região Madeira.

Figura 5.55 – Parte da página do Weather.com.

Como necessitamos de aceder a dez páginas Web diferentes para obter informação para os dez dias, então teremos de registar dez diferentes fontes de dados no sistema. Na figura 5.14 apresentamos como deverá ser registada a fonte de dados referente ao dia seguinte ao dia actual.

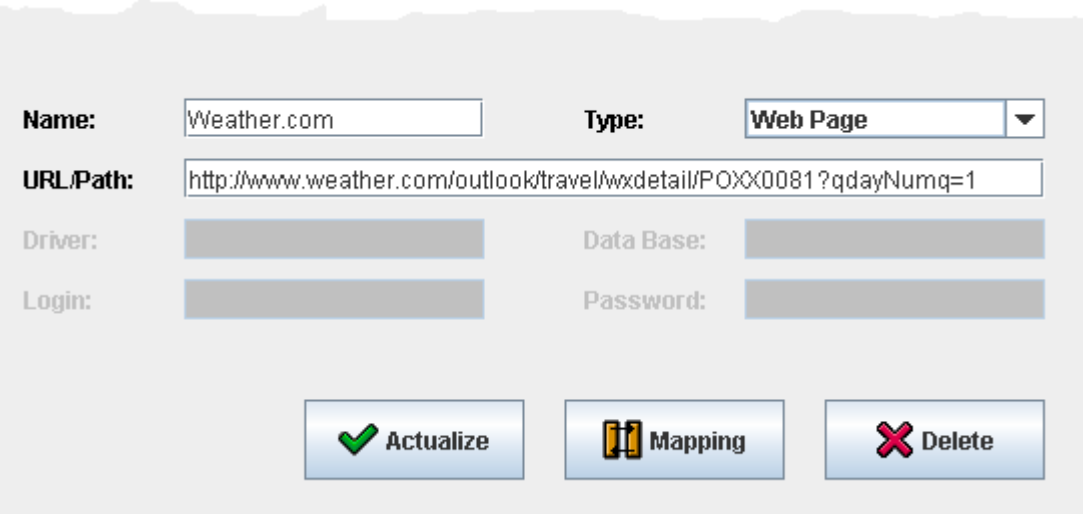

Figura 5.56 – Registo de uma das fontes de dados Weather.com.

As fontes de dados referentes ao Weather.com irão ser utilizadas para fornecer dados para o elemento XML "weather" do modelo sintáctico. Assim, é necessário criar um elemento de mapeamento para o elemento XML "weather" e um sub-elemento de mapeamento para o elemento XML "location". Deverão ser também definidos os atributos de mapeamento que irão disponibilizar os dados para preencherem os

atributos XML referentes aos elementos "weather" e "location". Os atributos serão definidos utilizando a linguagem WebL (Marais, 1999). Para cada atributo será criada uma função WebL que será responsável por obter a informação referente ao atributo. As funções WebL deverão ser guardadas em ficheiros locais de forma a poderem ser acedidas pelo sistema. A figura 5.15 mostra como deverá ser criado o atributo de mapeamento referente ao atributo "temperature" do elemento "weather" do modelo sintáctico. Nesta figura estamos a definir os elementos de mapeamento para a fonte de dados referente ao dia actual.

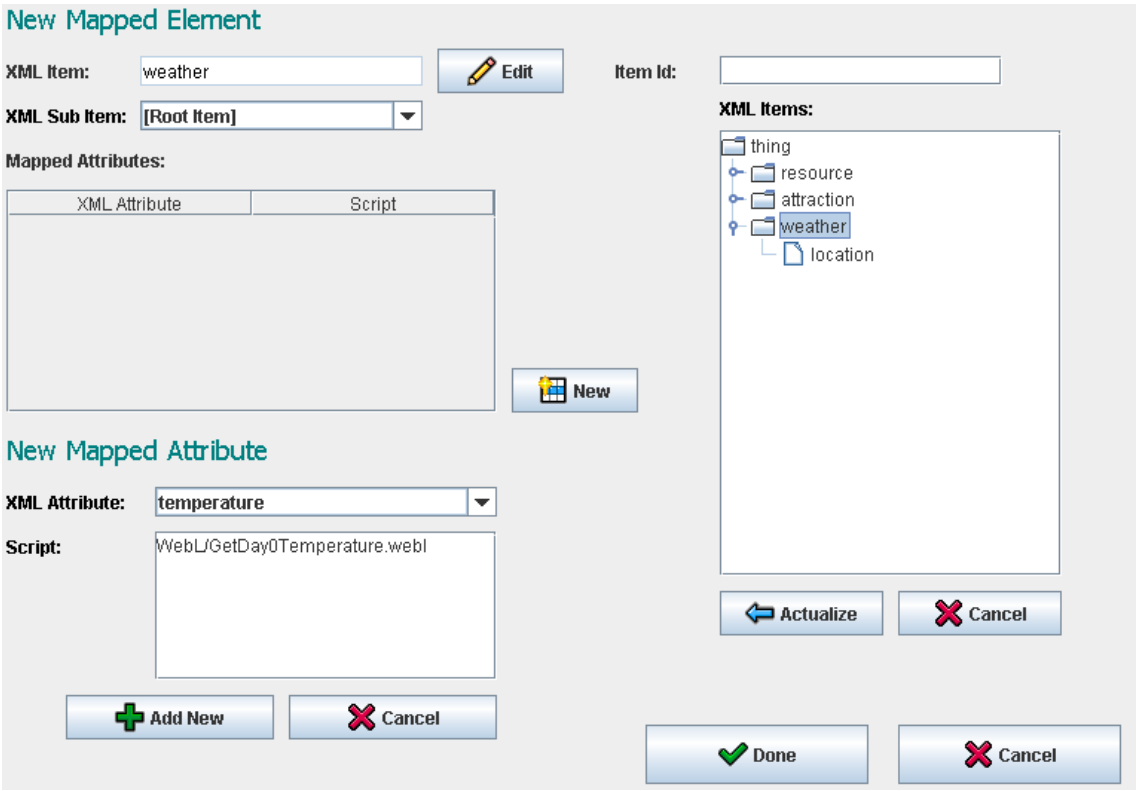

Figura 5.57 – Mapeamento de atributos para uma das fontes de dados Weather.com.

Como podemos observar na figura 5.15, para cada atributo de mapeamento temos de indicar qual o ficheiro WebL que será responsável por extrair a informação da página Web, necessária para preencher o atributo.

Depois de definirmos todos os atributos de mapeamento para as dez fontes de dados referentes ao sistema Weather.com, terminamos a integração com todas as fontes de dados necessárias para o nosso exemplo.

5.6 Utilização do Sistema

Depois de terminada a configuração do sistema e de termos terminado o registo das fontes de dados, podemos começar a testar e a utilizar o sistema. Utilizando a aplicação SEED é possível criar regras e queries sobre a Ontologia definida. As regras deverão ser criadas utilizando a linguagem nRQL. As queries são definidas utilizando os controlos da aplicação SEED. A aplicação tem depois a responsabilidade de transformar estas queries em queries semânticas, também definidas na linguagem nRQL.

5.6.1 Regras

O sistema SEED permite-nos a criação de regras. As regras são definidas utilizando as classes e propriedades existentes na Ontologia. Para o nosso exemplo iremos criar regras para a definição de duas classes, a classe "RecommendedHotel" e a classe "RecommendedRestaurant". Estas classes representam os hotéis e os restaurantes recomendados pela entidade que gere o sistema. A definição destas classes através de regras permite a alteração da definição destas enquanto o sistema está a correr e em tempo real.

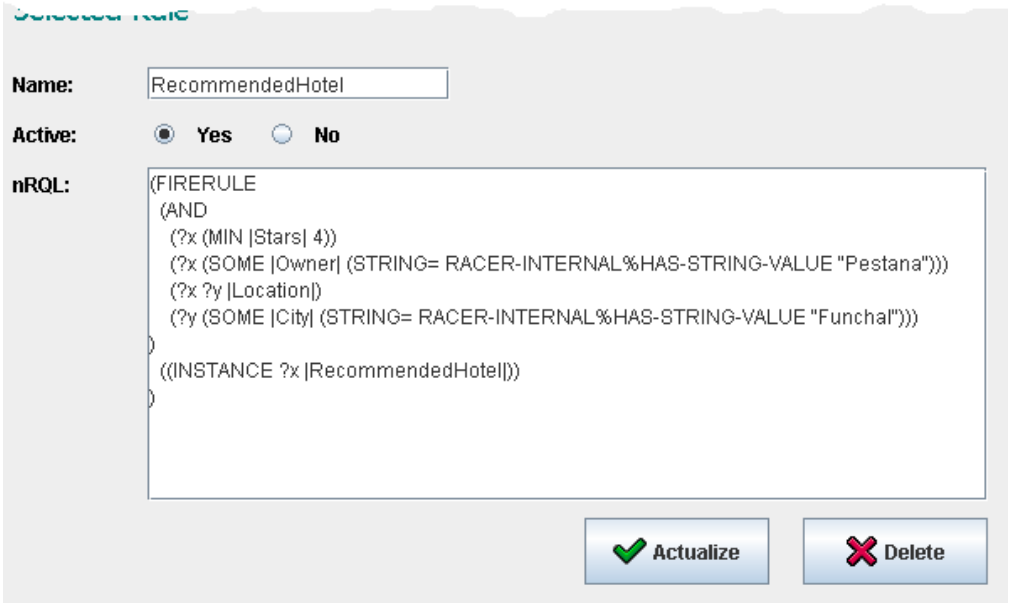

Figura 5.58 – Definição da regra "RecommendedHotel".

As regras deverão ser criadas através da linguagem nRQL e utilizando a aplicação SEED. Por exemplo, para definir os hotéis recomendados podemos criar uma regra que indique que um hotel recomendado tem de pertence à entidade Pestana, deve ter pelo menos 4 estrelas e deve estar localizado na cidade do Funchal. Na figura 5.16 temos representado esta regra que define a classe "RecommendedHotel".

A regra definida é composta pelo operador "AND", por um conjunto de expressões e por uma consequência. As expressões irão definir quais as condições necessárias para se chegar à consequência. As expressões têm a seguinte correspondência com a regra anteriormente indicada:

- **Pertence à entidade Pestana:**

```
?x (SOME |Owner| (STRING= RACER-INTERNAL%HAS-STRING-
VALUE "Pestana"))
```

- **Ter pelo menos 4 estrelas:**

```
?x (min |Stars| 4)
```

- **Estar localizado na cidade do Funchal:**

```
?x ?y |Location|
?y (SOME |City| (STRING= RACER-INTERNAL%HAS-STRING-
VALUE "Funchal"))
```

A consequência da regra é definida pela seguinte expressão:

```
INSTANCE ?x |RecommendedHotel|
```

Esta expressão indica que todas as instâncias da Ontologia que possam tomar o valor da variável "x", serão consideradas como instâncias da classe "RecommendedHotel".

Também devemos criar uma regra de forma a definir quais as características a serem consideradas para termos um restaurante recomendado. Definimos um restaurante recomendado como um restaurante que pertença à entidade Dorisol e que o tipo de cozinha seja Madeirense. A definição da regra é realizada da mesma forma que a regra anterior. Esta regra encontra-se apresentada na figura 5.17.

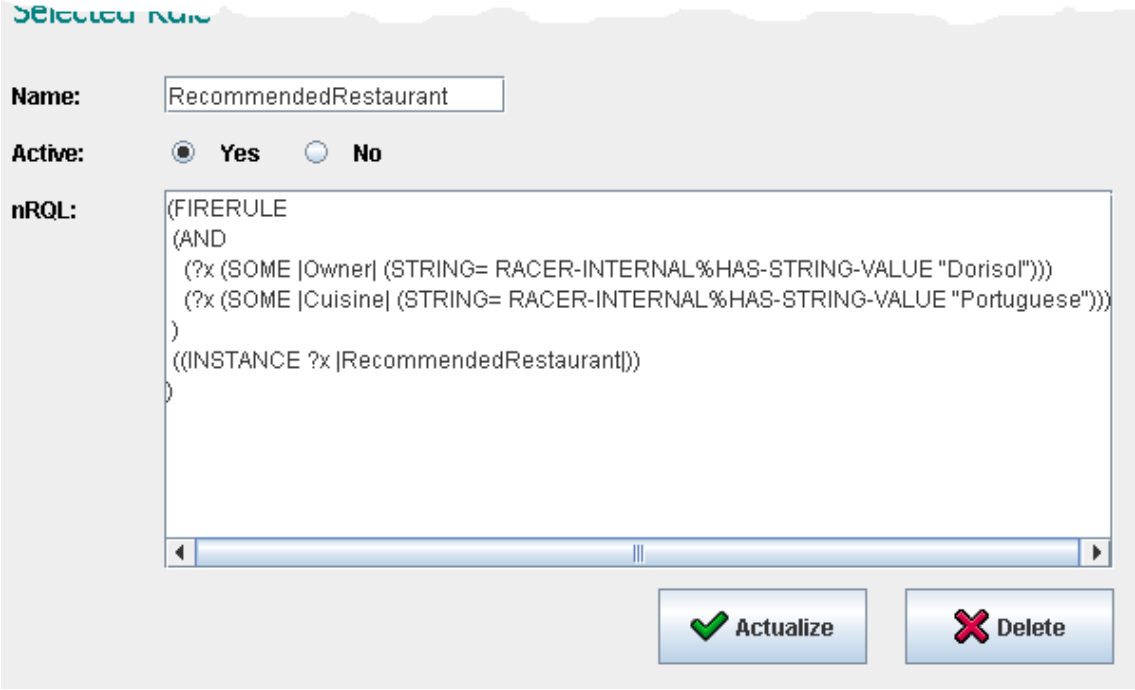

Figura 5.59 – Definição da regra "RecommendedRestaurant".

As regras definidas irão alterar os resultados das queries realizadas. Por exemplo, ao pesquisarmos por um hotel recomendado é a regra "RecommendedHotel" que irá definir quais os resultados a obter. As regras depois de definidas poderão ser activadas ou desactivadas enquanto o sistema está em execução. Além disto, as regras poderão ser modificadas em tempo real sem que seja necessário reiniciar o sistema. Por exemplo, a partir de determinado momento poderemos alterar a regra "RecommendedHotel" de forma a considerar como hotel recomendado apenas os hotéis com cinco estrelas. Esta alteração poderá ser realizada através da aplicação SEED e sem parar o sistema. A partir do momento da alteração, através de uma pesquisa por hotéis recomendados só iremos obter hotéis com cinco estrelas.

5.6.2 Queries

Depois de configurarmos o sistema e de definirmos as regras a utilizar, finalmente podemos testar o sistema. Os testes poderão ser realizados utilizando a aplicação SEED. Para testarmos o sistema deveremos criar várias queries para verificar se o resultado obtido é o esperado. Por exemplo, podemos criar uma query que permita obter todas as levadas da região Madeira com distância entre os 4 e os 12 Km. Através desta query queremos também obter as condições climatéricas entre os dias 4 e 6 de Novembro. A definição de todos estes filtros poderá ser realizada através da aplicação SEED utilizando a interface de definição de queries. Cada filtro deverá ser criado

individualmente, depois a aplicação trata de juntá-los de forma a obter o resultado da intersecção de todos estes filtros. Na figura 5.18 podemos observar como pode ser criado o filtro que indica que queremos obter todas as levadas com distância entre os 4 e os 12 Km.

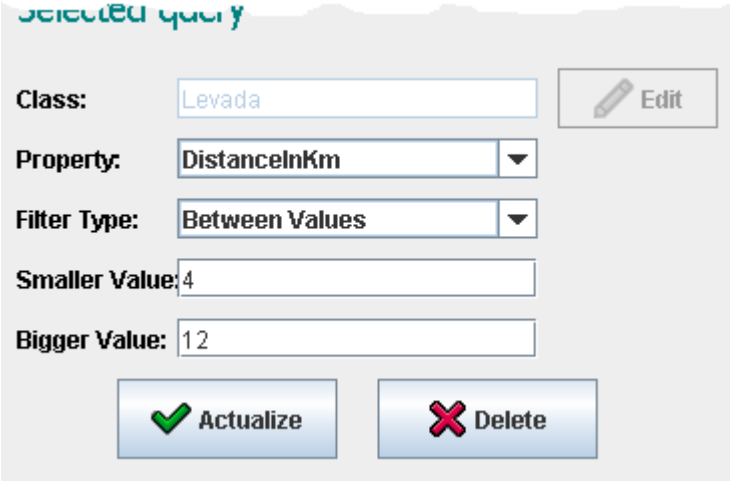

Figura 5.60 – Definição de filtro para a classe "Levada".

Depois de todos os filtros serem explicitamente definidos através da interface da aplicação SEED, o sistema gera as queries semânticas na linguagem nRQL. Para as levadas com distância entre 4 e 12 Km o sistema gera a seguinte query semântica:

```
retrieve (?x1)
(and
      (?x1 |Levada|)
      (?x1 (at-least 1 |DistanceInKm|(and (min 4))))
      (?x1 (at-least 1 |DistanceInKm|(and (max 12)))
)
```

Para obter as condições climatéricas entre os dias 4 e 6 de Novembro o sistema gera a seguinte query semântica:

```
retrieve (?x1)
(and
      (?x1 |Weather|)
      (?x1 (at-least 1 |Day|(and (min 4))))
      (?x1 (at-least 1 |Day|(and (max 6))))
      (?x1 (at-least 1 |Month|(and (equal 11))))
      (?x1 (at-least 1 |Year|(and (equal 2007))))
)
```

As queries em nRQL são depois transformadas em queries sintácticas definidas na linguagem de query da aplicação "Gatherer" (S2SQL).

```
SELECT levada
WHERE (levada.distance >= "4" AND
       levada.distance <= "12")

SELECT weather
WHERE (weather.day >= "4" AND
       weather.day <= "6" AND
       weather.month = "11" AND
       weather.year = "2007")
```

Aplicando a query anteriormente descrita, obtemos a informação apresentada na figura 5.19. Através desta query, um turista pode não só escolher qual o melhor percurso pedestre que poderá realizar bem como qual o melhor dia para o realizar.

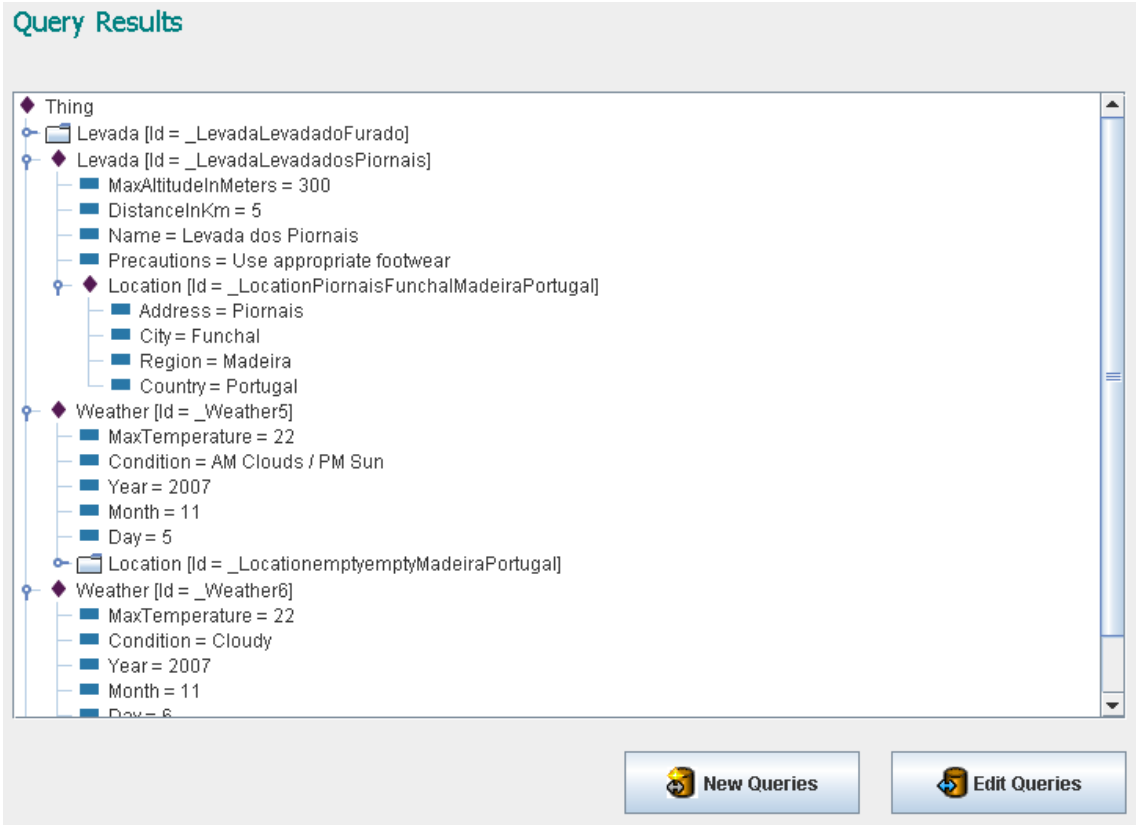

Figura 5.61 – Resultados da pesquisa de levadas e condições climatéricas.

Depois de escolher uma levada e um dia para a percorrer, faz todo o sentido procurar por um restaurante para jantar e por um hotel para poder passar a noite. Utilizando a aplicação SEED poderemos testar estas duas pesquisas. Por exemplo, para o hotel o turista poderá optar por verificar quais os hotéis recomendados na região Madeira. Para isto basta criar o filtro apresentado na figura 5.20.

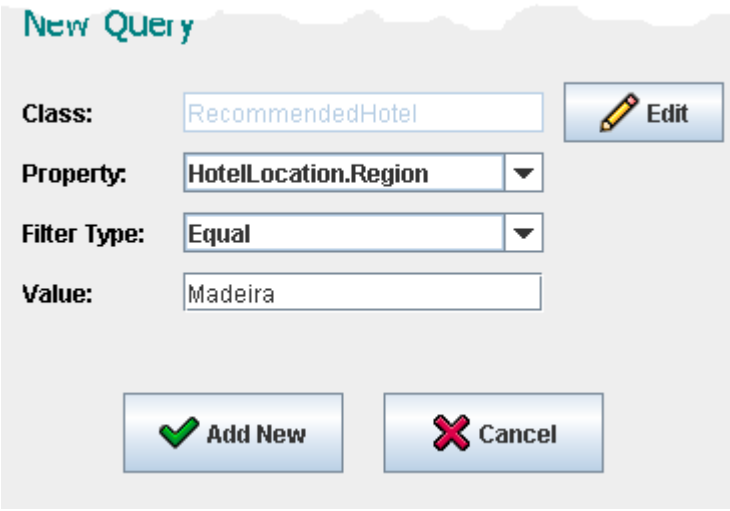

Figura 5.62 – Definição de filtro para a classe "RecommendedHotel".

De notar que ainda que estejamos a utilizar a classe "RecommendedHotel", o filtro criado utiliza uma propriedade da classe "HotelLocation". Isto só é possível porque a classe "RecommendedHotel" foi definida como subclasse da classe "Hotel" e porque existe uma propriedade objecto na classe "Hotel" que tem como domínio a classe "HotelLocation".

O filtro apresentado na figura 5.20 irá resultar na seguinte query semântica definida em nRQL:

```
retrieve (?x1)
(and
      (?x1 |RecommendedHotel|)
      (?x1 ?x2 |HotelLocation|)
      (?x2 (at-least 1 |Region|(STRING= "Madeira")))
)
```

A query semântica apresentada é depois transformada na seguinte query sintáctica definida na linguagem S2SQL.

```
SELECT hotel
WHERE (hotel.location.region="Madeira")
```

De notar, que apesar da classe "RecommendedHotel" não estar mapeada com nenhum elemento do modelo sintáctico, é criada uma query sintáctica que pede o retorno de instâncias do elemento "hotel". Isto acontece porque não foi definido qualquer mapeamento para a classe "RecommendedHotel" e porque esta classe é subclasse da classe "Hotel". Nestes casos o sistema, prevendo a possível definição das subclasses através das regras semânticas, considera o mapeamento com a super classe e cria uma query sintáctica sobre o elemento sintáctico mapeado com a super classe. O resultado

da pesquisa sobre hotéis recomendados na região Madeira é apresentado na figura 5.21.

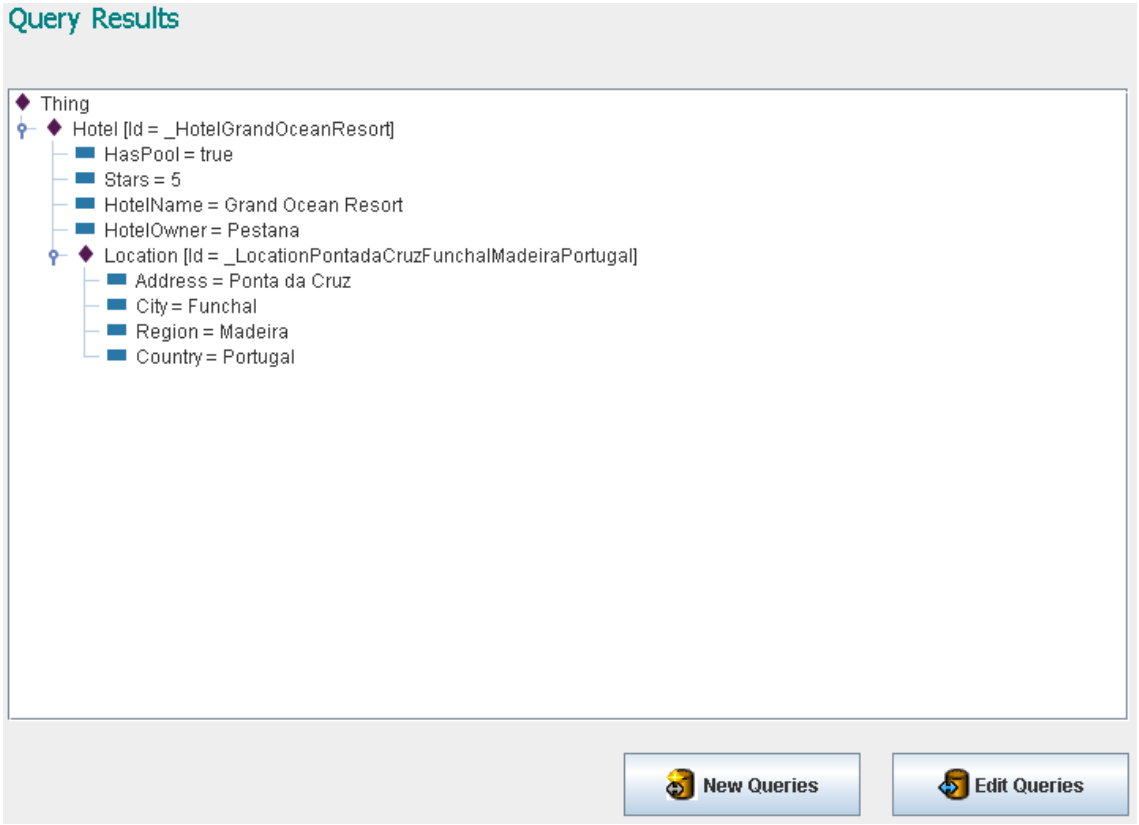

Figura 5.63 – Resultados da pesquisa de hotéis recomendados.

Através dos resultados podemos observar que apenas um hotel foi retornado. Este hotel foi obtido através da interpretação da regra "RecommendedHotel" por parte do sistema, logo, respeita as características definidas nesta regra (pertence à entidade Pestana, deve ter pelo menos 4 estrelas e deve estar localizado na cidade do Funchal).

Depois do sistema estar testado, termina o processo de preparação deste. A partir deste momento o sistema encontra-se pronto a ser utilizado. Poderá ser utilizado por aplicações ou sistemas Web que pretendam disponibilizar, de forma agregada, a informação existente nas fontes de dados integradas. A utilização da aplicação SEED não termina aqui, é uma boa ferramenta de auxílio à manutenção do sistema.

6 CONCLUSÕES

6.1 Limitações actuais do Sistema

O sistema implementado responde aos principais requisitos definidos. No entanto, existem algumas limitações detectadas que achamos serem de importante resolução para tornar o sistema mais perfeito. Para cada uma das limitações descrevemos como estas poderão ser ultrapassadas ou minimizadas. As limitações detectadas são:

- **Integração com Web Services limitada:** Só é possível a integração de Web Services que retornem um único valor simples. No domínio do turismo é raro encontrar-se Web Services que apenas retornem um valor simples. Actualmente, a única maneira de ultrapassarmos o problema passa pela implementação de um sistema moderador que permita invocar o Web Service e transformar a resposta destes num simples documento XML. Desta forma, podemos proceder à integração como se trata-se de um ficheiro XML. O problema desta solução está no trabalho despendido na implementação do sistema moderador. Para a resolução desta limitação é necessário que o sistema permita a criação de elementos de mapeamento para os elementos presentes na estrutura de resposta dos Web Services a integrar. Isto é possível através do processamento dos documentos WSDL associados aos Web Services.

- **Perda de performance na transformação dos dados sintácticos em dados semânticos:** Para a transformação das instâncias do Modelo Sintáctico em instâncias da Ontologia utilizamos o documento XSLT gerado pela aplicação "JXML2OWL Mapper". Apesar de se tratar de um standard, o XSLT torna-se lento quando aplicado a documentos muito longos. A implementação de um processo de transformação de dados próprio não seria muito complexo e iria optimizar em muito a transformação. O processo de transformação poderia utilizar as regras de mapeamento geradas também pela aplicação "JXML2OWL Mapper". Actualmente, para ultrapassar este problema aconselhamos a definição de um Modelos Sintáctico não muito extenso e com uma estrutura não muito distinta da estrutura da Ontologia.

- **O motor de inferência apenas suporta os tipos de dados simples mais comuns:** O motor de inferência utilizado apenas suporta os seguintes tipos de dados simples: "boolean", "float", "int", e "string". Existem outros tipos de dados suportados pela linguagem OWL tais como "date" ou "decimal". No domínio do turismo é muito comum a utilização de datas, logo, a existência de um tipo de dados "date" iria facilitar o processamento. Para resolver este problema teríamos de mudar de motor de inferência. No entanto, achamos que as vantagens em utilizar o RacerPro não justificam tal alteração. Através dos tipos de dados suportados pelo RacerPro conseguimos sempre representar os outros não suportados. Por exemplo, no caso do tipo de dados "date" pode ser representado por três variáveis do tipo inteiro, uma para representar o dia, outra para representar o mês e outra para representar o ano. Com esta solução conseguimos representar uma data, não conseguimos é obter a mesma facilidade de manipulação de datas que conseguimos quando utilizamos o tipo de dados "date".

- **Queries a fontes de dados do tipo Base de Dados não utilizam o motor da base de dados para a execução total das queries:** Os motores de base de dados encontram-se geralmente bem implementados e oferecem um processo de execução de queries muito optimizado. A utilização destes motores para a extracção dos dados seria vantajoso. Actualmente o sistema SEED utiliza os motores de base de dados apenas para obter a informação, a interpretação dos filtros é executada pelo sistema SEED. Se passarmos esta responsabilidade para os motores de base de dados iremos ganhar performance. Esta alteração iria tornar a extracção de dados para Base de Dados diferente dos outros tipos de fontes de dados. Para o caso de integração de Bases de Dados não poderíamos reutilizar os métodos existentes para a extracção dos dados nem poderíamos limitar a integração à interface de integração existente. Devido à grande utilização actual de sistemas de Base de Dados achamos que este diferente tratamento se justificaria.

6.2 Trabalho Futuro

A arquitectura definida foi pensada para a integração de sistemas turísticos de forma a apresentar a informação turística de forma agregada. No entanto, a arquitectura foi definida prevendo uma futura extensão, de forma a permitir a criação de pacotes dinâmicos. Outras funcionalidades poderão ser desenvolvidas de forma a tornar o sistema mais completo. Aqui indicamos as que achamos serem mais importantes.

- **Motor para a criação de pacotes dinâmicos:** A integração de sistemas turísticos de forma a serem acedidos de forma agregada, é apenas um primeiro passo para a implementação de um sistema que suporte o "Dynamic Packaging". A utilização de uma Ontologia para a agregação da informação e a possibilidade de definição de regras de negócio sobre a Ontologia, são funcionalidade do sistema SEED que irão simplificar o suporte aos pacotes dinâmicos. Através das Ontologia podemos definir a constituição dos pacotes dinâmicos e através das regras de negócio definir características dos pacotes que podem ser alteradas em tempo de execução do sistema. Através dos critérios definidos pelo utilizador e consultando a definição dos pacotes e as regras associadas a estes, é possível a criação de pacotes de forma dinâmica e automática. O motor para a criação de pacotes dinâmicos deverá ser implementado sobre o nível semântico.

- **Reserva dos pacotes dinâmicos:** Na definição de "Dynamic Packaging" incluem-se as reservas dos produtos turísticos constituintes do pacote através de uma única transacção. O suporte a reservas de produtos turísticos implica a existência de serviços que permitam a criação destas reservas. Estes serviços deverão ser disponibilizados pelas entidades que gerem os produtos turísticos e deverão ser registados no sistema. O registo dos serviços deverá permitir ao sistema reconhecer qual o serviço a invocar para a reserva de um determinado produto turístico. O sistema deverá ainda conhecer quais os parâmetros a indicar de forma a criar a reserva. Como se pode constatar, o suporte a reservas de produtos não é uma tarefa fácil. Cada tipo de sistema poderá disponibilizar uma forma diferente para a reserva dos produtos. O recurso à utilização de linguagens como o OWL-S (Martin,

Burstein, Hobbs, Lassila, McDermott, McIlraith, Paolucci, Parsia, Payne, Sirin, Srinivasan e Sycara, 2004) poderá ser uma das hipóteses possíveis para o suporte a reservas de produtos turísticos. Através da linguagem OWL-S é possível adicionar significado aos serviços de forma a que os sistema possa reconhecer os métodos próprios para a reserva dos produtos turísticos.

- **Funções de mapeamento e mapeamento condicional:** O sistema SEED permite o mapeamento de componentes do Modelo Sintáctico com componentes da Ontologia. O mapeamento é utilizado para a transformação das instâncias sintácticas em instâncias da Ontologia. Actualmente apenas é possível definir mapeamentos directos, ou seja, que um tipo de elemento do Modelo Sintáctico irá ser transformado numa determinada classe da Ontologia. A possibilidade de criação de regras de mapeamentos mais complexas torna-se necessário quando existe uma maior discrepância entre os modelos a mapear. A possibilidade de transformar valores referentes a atributos mapeados é uma da funcionalidades úteis a serem implementadas. Por exemplo, se no Modelo Sintáctico tivermos a representação do valor de desconto em percentagem e na Ontologia tivermos o mesmo valor na forma decimal, podemos definir uma função de mapeamento que permita a transformação correcta entre estes dois atributos. Por vezes também necessitamos de definir condições sobre as quais um determinado componente do Modelo Sintáctico poderá ser mapeado com um elemento da Ontologia. Por exemplo, podemos definir a classe "HotelLuxo" na Ontologia para representar os hotéis de luxo. Se no Modelo Sintáctico apenas existir o elemento "Hotel", então poderíamos definir um mapeamento condicional que indicasse que todos os elementos "Hotel" deverão ser transformados em instâncias da classe "HotelLuxo", caso tenham mais de 4 estrelas.

- **Sistema de cache:** A opção de extracção da informação turística em tempo de execução deveu-se às características da informação turística que necessita de actualização permanente. Informação como os valores de tarifas ou informação referente à disponibilidade dos produtos são exemplos de informação que necessita de actualização permanente. No entanto, existe outra informação que não necessita desta constante actualização. As características dos hotéis ou a informação associada às atracções de uma determinada localidade são exemplos de informação que não necessita de ser actualizada com tanta frequência. Para optimizar a

pesquisa de informação, a informação que não necessita de actualização tão frequente poderia ser adicionada à Ontologia em processos assíncronos e em ocasiões em que o sistema denotasse menor actividade. Para tornar isto possível é necessário a implementação de um sistema de cache que permitisse definir qual a informação a ser obtida em tempo real e qual aquela que poderia ser adicionada previamente à Ontologia.

6.3 Conclusão Final

O sistema SEED permite a pesquisa de informação turística que poderá pertencer a diferentes sistemas que utilizam diferentes tecnologias. Através da utilização das tecnologias associadas à Web Semântica, preparamos o sistema para o futuro suporte à implementação do "Dynamic Packaging".

Durante a definição da arquitectura SEED e durante a implementação do sistema SEED surgiram alguns problemas de complexa resolução. De seguida enumeramos estes problemas:

- **Integração de fontes de dados heterogéneas numa Ontologia de forma simples:** A integração das fontes de dados no sistema teria de ser um processo simples. Deveria permitir que as entidades sem conhecimentos na área da Web Semântica conseguissem configurar as suas fontes de dados no sistema. Conseguimos ultrapassar este desafio criando um Modelo Sintáctico entre as fontes de dados e a Ontologia. A introdução do Modelo Sintáctico tornou a definição e implementação do sistema mais complexo mas resolveu o problema da complexidade na integração das fontes de dados.

- **Propagação das queries semânticas até às fontes de dados:** A utilização de uma Ontologia para a integração dos dados provenientes das várias fontes de dados externas sugere a utilização de uma linguagem de query semântica para o acesso aos dados. Este facto levanta o problema de as fontes de dados não conhecerem tais queries. Foi necessário a implementação de um processo de transformação entre as linguagens de query nRQL e S2SQL. A transformação das queries não foi uma tarefa fácil devido a diferença sintáctica entre as linguagens e devido ao facto de serem utilizados nas queries componentes referentes a diferentes modelos de dados. A existência do documento XML com as regras de mapeamento facilitou a transformação das queries.

- **Utilização da linguagem OWL para a definição da Ontologia:** O não conhecimento da linguagem OWL tornou a implementação do sistema um pouco

mais complicado. Foi necessário um investimento de tempo no estudo desta linguagem. O estudo da linguagem OWL não foi tarefa fácil devido à grande expressividade que esta linguagem oferece.

- **Utilização da linguagem nRQL para a definição das queries e das regras de negócio:** A linguagem nRQL faz parte do motor de inferência RacerPro e foi utilizada para a definição das queries semânticas e das regras de negócio. Também por falta de conhecimento na utilização de motores de inferência, o estudo da linguagem nRQL tornou-se numa tarefa complicada. O facto de ser uma linguagem poderosa, tanto na definição de queries como na definição de regras, tornou também a sua compreensão mais complexa.

O sistema implementado pode ser utilizado para a criação de Ontologias simples que possibilitem a integração de informação turística presente em diferentes fontes de dados. É um sistema ideal para reunir informação turística presente em pequenos sistemas informáticos que disponibilizam informação turística. O sistema oferece uma aplicação que permite a configuração do sistema de forma simples e rápida. É apenas necessário a existência de noções básicas de Ontologias por parte do administrado do sistema e conhecimentos ao nível do XML por parte dos integradores das fontes de dados externas. O sistema SEED também pode ser utilizado para a integração de sistemas de informação noutros domínios, que necessitem de informação actualizada em tempo real.

O sistema implementado também poderá ser utilizado para a criação de sistemas de "Dynamic Packaging" através da implementação de um motor para a criação de pacotes dinâmicos. Achamos que é no suporte ao "Dynamic Packaging" que esta aplicação realmente ganhará valor. A unificação dos dados numa Ontologia e a possibilidade de definição de regras de negócio é o primeiro passo ao suporte do "Dynamic Packaging". Acreditamos que a arquitectura SEED define a melhor forma de implementação de um sistema para a integração de informação turística e que serve de base para sistemas que pretendem implementar o conceito de "Dynamic Packaging".

REFERÊNCIAS

Alexiev, V., Breu, M., Bruijn, J., Fensel, D., Lara, R. and Lausen H. (2005), *Information Integration with Ontologies – Experiences from an Industrial Showcase*. John Wiley & Sons Ltd.

Antoniou, Grigoris e Harmelen, Frank van (2004*). A Semantic Web Primer*. The MIT Press. Cambridge, Massachusetts, London, England.

Bechhofer, Sean (Fevereiro de 2003). *The DIG Description Logic interface: DIG/1.1*. University of Manchester.

Bechhofer, S., Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D., Patel-Schneider, P. e Stein, L. (Fevereiro de 2004). *OWL Web Ontology Language Reference*. W3C Recommendation.

Berners-Lee, Tim (2000). *Semantic Web – XML2000*. (http://www.w3.org/2000/Talks/1206-xml2k-tbl/slide10-0.html.

Bicer, V., Laleci, G., Dogac, A. e Kabak Y. (2005). *Providing Semantic Interoperability in the Healthcare Domain through Ontology Mapping*. Sofware Research and Development Center. Middle East Technical University (METU).

Boag, S., Chamberlin, D., Fernandez, M., Florescu, D., Robie, J. e Simeon, J. (Janeiro de 2007) XQuery 1.0: An XML Query Language. W3C Candidate Recommendation. (http://www.w3.org/TR/xquery/).

Buhalis, D. e Costa, C. (2006). *Tourism Business Frontiers*. Oxford et al., Elsevier.

Buhalis, D. e Spada, A. (2000). Destination Management Systems: Criteria for Success – an Exploratory Research. *Information Technology & Tourism, vol. 3* (1): 41-58.

Cardoso, Jorge (2005). *Semantic Web: Technologies and Applications*. Departamento de Matemática e Engenharia da Universidade da Madeira.

Cardoso, Jorge e Lange, Carola (2005). *Strategies and Technologies for Dynamic Packaging Applications in e-Tourism*.

Cardoso, Jorge e Sheth, Amit (2006). *Semantic Web Services, Processes and Applications*. Springer.

Casey, M. e Austin, M. (Novembro de 2001). *Semantic Web Methodologies for Spatial Decision Support*. University of Maryland, Institute for Systems Research and department of civil and Environmental Engineering.

Clark, James (Novembro de 1999). XSL Transformations (XSLT). Version 1.0. W3C Recommendation. (http://www.w3.org/TR/xslt).

Daconta, Michael C., Obrst, Leo J. e Smith, Kevin T. (2003). *The Semantic Web: A Guide to the Future of XML, Web Services, and Knowledge Management*. Wiley Publishing, Inc.

Davies, John, Fensel, Dieter e Harmelen, Frank van (2003), *Towards the Semantic Web. Ontology-driven Knowledge Management*. John Wiley & Sons, Ltd.

Extensible Markup Language (XML) (2003). (http://www.w3.org/XML/).

FaCT (2005). FaCT++, (http://owl.man.ac.uk/factplusplus/).

Fernandes, Jorge (Setembro de 2005). *Um Novo Modelo de Turismo em Portugal, Uma Abordagem Pragmática e Flexível, O Contributo da Expedita*.

Fitzgerald, Chicke (2005). Dynamic Packaging: The impact of technology on the sale of commodity products, both online and offline. The Solutionz Group International, Inc. (http://www.solutionz.com/pdfs/01-Dynamic_Packaging.pdf).

Gandon, F. L. e Sadeh, N. M. (2003). OWL inference engine using XSLT and JESS, (http://www-2.cs.cmu.edu/~sadeh/MyCampusMirror/OWLEngine.html).

Gruber, T. R. (1993). *A Translation Approach to Portable Ontology Specification*. Knowledge Acquisition.

Haarslev, Volker e Möller, Ralf (2003). *Racer: A Core Inference Engine for the Semantic Web*. Concordia University e Technical University Hamburg.

Hit Software (2005). *The Business Value of Hit Software Data Integration Tools: Allora and DBMoto.*

HL7 (2007). Health Level Seven Inc. (http://www.hl7.org/).

Horrocks, I., Patel-Schneider, P., Boley, H., Tabet, S., Grosof, B. e Dean, M. (Maio de 2004). *SWRL: A Semantic Web Rule Language. Combining OWL and RuleML*. W3C Member Submission. (http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/).

HotelOnline (2002). *Global Distribution Systems in Present Times*, Hotel Online (http://www.hotel-online.com/News/PR2002_4th/Oct02_GDS.html).

ISNOVA (2005). (http://www.isnova.net/index.php).

ISNOVA Web Site (2006). *Veredas e Levadas da Madeira*. (http://isnova.madeiratecnopolo.pt).

Jena(2005). Jena – A Semantic Web Framework for Java, (http://jena.sourceforge.net/).

Jigloo (2007). SWT/Swing GUI for Eclipse and WebSphere. CloudGarden. (http://www.cloudgarden.com/jigloo/index.html).

Kabbaj, Mohamed Youssef (Junho de 2003). Strategic and Policy Prospects for Semantic Web Services Adoption in US Online Travel Industry.

Knublauch, H., Fergerson, R., Noy, N. e Musen, M. (2004). *The Protégé OWL Plugin: An Open Development Environment for Semantic Web Applications*. Stanford Medical informatics, Stanford School of Medicine.

Lawrence , Ramon e Barker, Ken (1999). *Integrating Data Sources Using A Standardized Global Dictionary.* Department of Computer Science, University of Manitoba e Department of Computer Science, University of Calgary, Canada.

Lofgren, S. (2005). Metadata for Improving Commercialisation of Dynamic Tourist Packages. (http://www.ibit.org/dades/doc/864_ca.pdf).

Marais, Hannes (1999). *Compaq's Web Language. A programming Language for the Web*. Compaq System Research Center (SRC).

Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinivasan, N.e Sycara, K. (Novembro de 2004). OWL-S: Semantic Markup for Web Services. W3C Member Submission.

McGuinness, Deborah L. e Harmelen, Frank van (Fevereiro de 2004). *OWL Web Ontology Language Overview*. W3C Recommendation.

Mullaney, T. J. (2004). Design Your Own Discount Getaway, Business Week – Online (http://www.businessweek.com/magazine/content/04_10/b3873101_mz070.htm).

Murua, I., Lladó, E. e Llodrá, B. (2005). The Semantic Web for Improving Dynamic Tourist Packages Commercialisation. (http://www.ibit.org/dades/doc/1108_ca.pdf).

O'Connor, P., *Online Pricing – An Analysis of Hotel Company Practices*. Cornell Hotel & Restaurant Administration Quarterly, 2003. 44(1): p. 10-19.

Plataforma xRS (2007). XRS – Central de Reservas On-Line. Expedita. (http://www.expedita.com/expedita/index.php?option=com_content&task=view&id=52&Itemid=74).

Prud'hommeaux, Eric e Seaborne, Andy (Novembro de 2007). *SPARQL Query Language for RDF*. W3C Proposed Recommendation. (http://www.w3.org/TR/2007/PR-rdf-sparql-query-20071112/).

RacerPro (Dezembro de 2005). RecerPro User's Guide. Version 1.9. Racer Systems GmbH & Co. KG. (http://www.racer-systems.com).

Riebeek, Holli. (2003). The Ticket Chase, IEEE Spectrum online (http://www.spectrum.ieee.org/WEBONLY/publicfeature/jan03/tair.html).

Rodrigues, Toni e Rosa, João (Julho de 2006). *JXML2OWL – Java XML to OWL*. Final Gegree Project Report. Departamento de Matemática e Engenharia. Universidade da Madeira.

RSS Specifications (2007). Everything you need to know about RSS. (http://www.rss-specifications.com/rss-specifications.htm).

Schreiber, Zvi (2003). Semantic Information Architecture: Solving the Enterprise Data Problem. Executive White Paper. Unicorn Solutions, Inc. (http://datawarehouse.ittoolbox.com/pub/ZS050903.pdf).

Silva, Bruno (2006). *Integrating Web Tourism Information Sources. Gatherer Syntactic-to-Semantic.* Departamento de Matemática e Engenharia. Universidade da Madeira.

Sink, David (Maio de 2002), program director for IBM emerging technologies, cited in InformationWeek (http://www.informationweek.com/story/IWK20020524S0005).

Teranode (2007). TERANODE Design Suite: Biological Modeler (http://www.teranode.com/products/tds/biological_modeler.php).

Unicode Consortium (Junho de 2007). *What is Unicode?* (http://www.unicode.org/standard/WhatIsUnicode.html).

Visser, Pepijn R. S. e Tamma, Valentina A. M. (1999). An Experience with Ontology-based Agent Clustering. CORAL - Conceptualisation and Ontology Research at Liverpool Department of Computer Science, University of Liverpool.

Visual Studio Developer Center (2007). Microsoft (http://msdn2.microsoft.com/en-us/vstudio/default.aspx).

Web Services Activity (2002). (http://www.w3.org/2002/ws/).

Werthner, H. e Klein (2004). S., *Information Technology and Tourism: A Challenging Relationship*. Berlin etc., Springer.

Wielemaker, J. (2005). SWI-Prolog Semantic Web Library, (http://www.swi-prolog.org/packages/semweb.html).

Wilson, Gordon (2005). *Dynamic Packaging and the Future of Online Travel.*

XML Schema W3C (2007). (http://www.w3.org/XML/Schema).

XQuery (Junho de 2006). XQuery 1.0: An XML Query Language. W3C Candidate Recommendation . (http://www.w3.org/TR/xquery/).