6
Arquitetura de Implementação

Neste capítulo apresentamos o esboço de uma arquitetura de implementação capaz de oferecer suporte para as idéias apresentadas nesta tese. É importante lembrar que este trabalho está inserido no contexto da Web Semântica e, portanto, muitas questões práticas ainda precisam ser solucionadas.

6.1.
Visão Geral da Arquitetura de Implementação

Em nossa arquitetura trabalhamos com os artefatos do método SHDM mencionados ao longo desta tese. O projetista da aplicação elabora o Esquema Conceitual SHDM e pode usar uma ferramenta de mapeamento (SHDM2OWL) [31]para automatizar a tradução do esquema para a Ontologia Conceitual SHDM. Os Esquemas de Classes Navegacionais e de Contextos SHDM também são elaborados pelo projetista e a ferramenta pode ser novamente utilizada para traduzir os esquemas em uma Ontologia Navegacional SHDM, conforme ilustrado na Figura 37. Enquanto a ferramenta SHDM2OWL não está disponível, o projetista pode utilizar os Editores de Ontologias existentes que oferecem suporte ainda limitado a DAML+OIL, mas acreditamos que em breve teremos editores compatíveis com a linguagem de ontologias OWL. O importante é observar que fornecemos esquemas SHDM com abstrações mais amigáveis para permitir que a aplicação Web seja projetada.

Uma vez que a Ontologia Conceitual é definida, seja através de uma ferramenta ou de um processo manual, ela está disponível para validar as instâncias correspondentes. Estas instâncias, por sua vez, podem ser provenientes de diversas fontes de dados. Por exemplo, poderíamos ter instâncias relativas a imagens que fossem provenientes de uma base de dados específica, enquanto outra base de dados poderia fornecer outras informações textuais.

[31] A ferramenta também pode ser SHDM2DAMLOIL.

Como as Ontologias (Conceitual e Navegacional) são definidas em linguagens processáveis, as mesmas são importadas para o ambiente de persistência, onde temos serviços de armazenamento, inferência e consulta. Os detalhes deste ambiente serão apresentados em seguida.

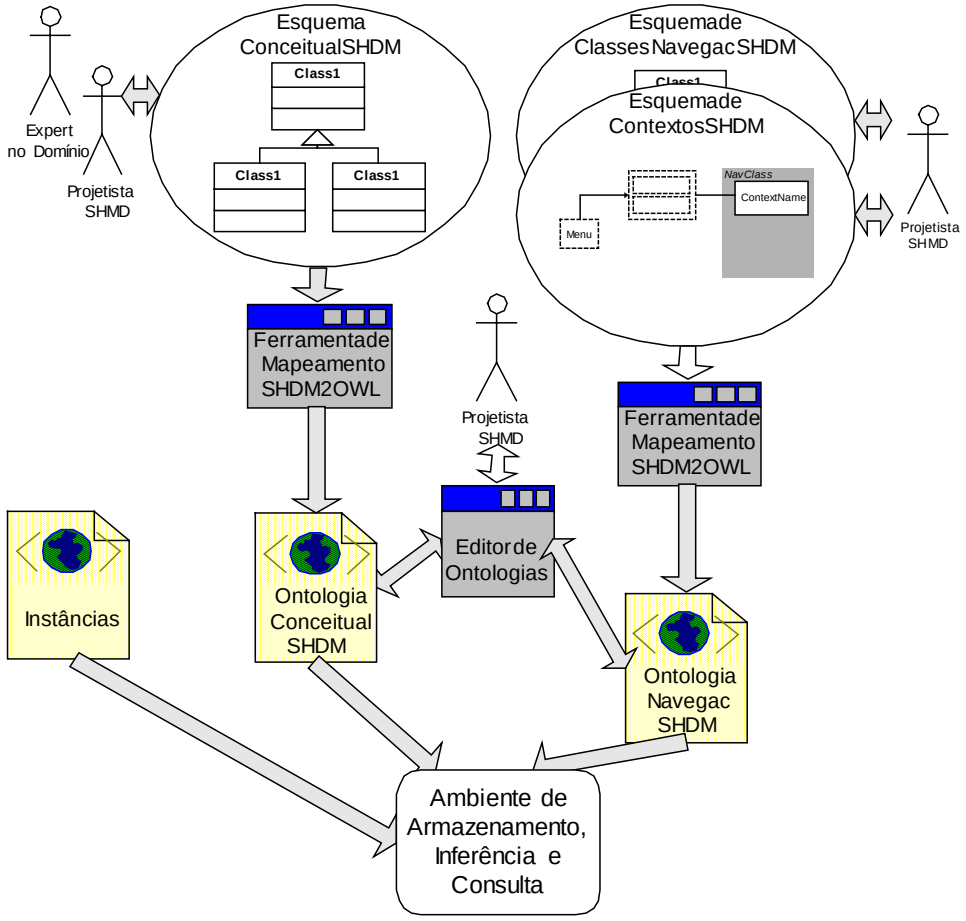

Figura 37 – Arquitetura de implementação para os artefatos SHDM

Outra pesquisa muito interessante que pretendemos realizar como trabalho futuro é a elaboração do Esquema Conceitual (e conseqüentemente da Ontologia Conceitual) utilizando uma arquitetura de mediadores [Wiederhold, 1992]. Será preciso tratarmos questões relacionadas à integração de ontologias. O tópico de integração vem sendo estudado pela comunidade de Banco de Dados há muitos anos e existem diversas propostas para elaboração de modelos canônicos em mediadores. Pretendemos avaliar as propostas que vêm sendo realizadas na área de Lógica Descritiva com extensões para integração de ontologias [Calvanese & DeGiacomo, 2003]. Nossa tarefa será viabilizar um

Projeto Navegacional de aplicações Web para estas Ontologias Conceituais Integradas.

Na Figura 38 esboçamos o ambiente de armazenamento, inferência e consulta, cuja arquitetura é composta de diversas camadas (claramente inspiradas na arquitetura genérica Sesame [Broekstra & Kampman, 2001]). A camada mais externa (1) é responsável por fornecer a funcionalidade necessária para acesso Web aos usuários, recebendo requisições de diferentes protocolos. O módulo Direcionador de Requisições recebe solicitações que podem fazer uso de protocolos como HTTP, RMI ou SOAP e direciona estas requisições para os módulos funcionais adequados.

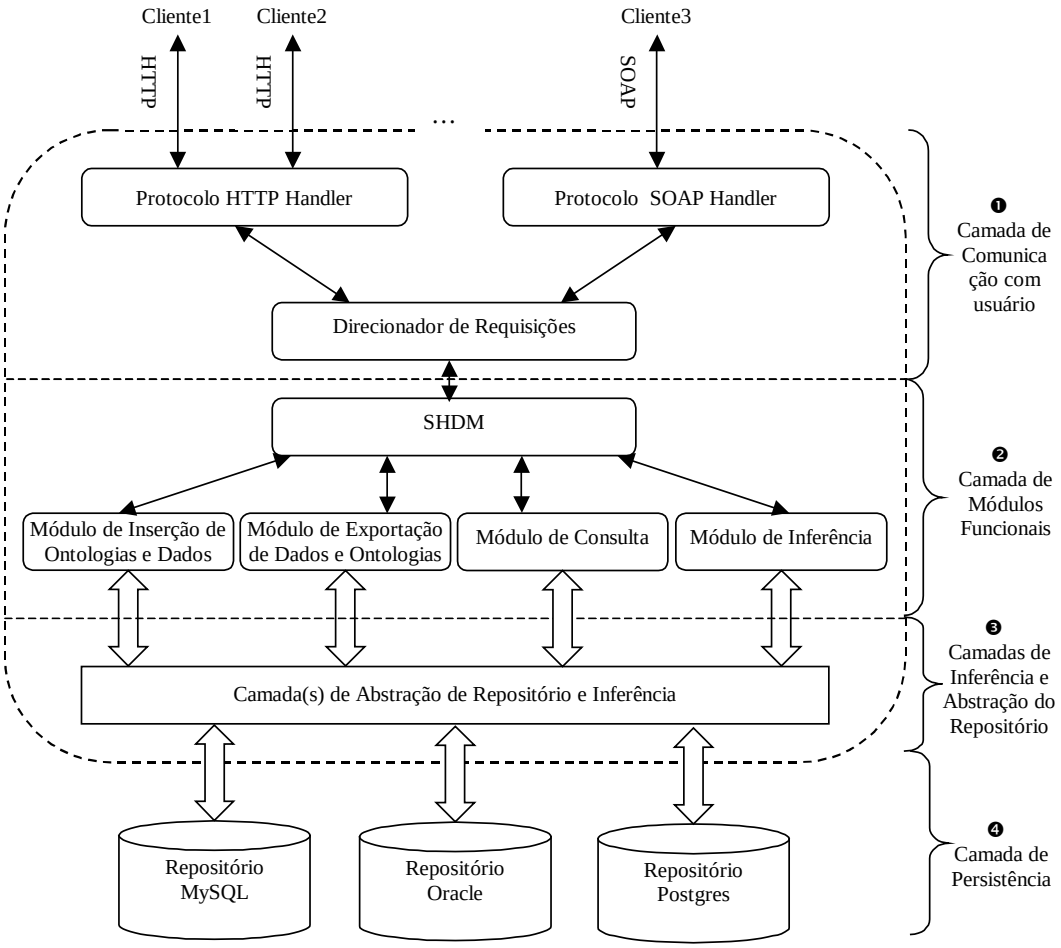

Figura 38 - Ambiente de armazenamento, inferência e consulta

Os módulos funcionais residem em uma camada intermediária (2) e a arquitetura permite a criação de novos módulos a serem incorporados nesta camada. Cada um destes módulos implementa suas funcionalidades básicas através de chamadas à camada inferior.

Para que o ambiente seja independente do repositório escolhido para persistência, existe uma camada abstrata (3), que na verdade é uma interface que oferece métodos específicos para tratar RDF nos clientes e traduzir estes métodos para chamadas ao banco de dados que tiver sido escolhido para persistência. Desta forma é possível implementar esta arquitetura em repositórios distintos sem alterar as camadas superiores.

Finalmente a última camada (4) é a de persistência, a ser preenchida com qualquer tipo de repositório concreto capaz de armazenar RDF: SGBDs (Relacionais, Orientados a Objetos, Objeto-Relacionais), repositórios RDF, etc.

A princípio, estes repositórios podem estar fisicamente próximos ao restante do ambiente. Mas nada impede que trabalhemos com repositórios distribuídos na Web (um outro possível trabalho futuro). Neste caso, seus serviços poderiam ser oferecidos através de *Web Services*. Uma consulta a um servidor poderia ser emitida e este servidor poderia contactar outros servidores na Web gerando uma arquitetura distribuída para armazenamento e consulta RDF.

6.2. Implementação Atual

Atualmente estamos utilizando a arquitetura Sesame[32] em nossos experimentos de armazenamento de ontologias conceituais e navegacionais SHDM. Este ambiente, desenvolvido em Java, oferece armazenamento RDF e RDF(S) e suporte à linguagem de consulta RQL. Além disto, o ambiente básico é capaz de realizar inferências nas definições RQL(S) (como transitividade) e persistir as novas tuplas inferidas, tanto em relação ao esquema quanto às instâncias.

Inicialmente, investigamos o ambiente de implementação ICS-FORTH RDFSuite [Alexaki et al., 2001], cuja equipe contém autores da linguagem RQL. No entanto esta implementação não possui a mesma flexibilidade de Sesame, tendo sido criada para persistência apenas no SGBD PostgreSQL e sistemas operacionais Mandrake Linux e Solaris. Optamos por usar o Sesame devido a independência de plataforma (Sistema Operacional), uma vez que foi desenvolvido em Java e também devido a flexibilidade na escolha de SGDBs.

[32] http://sesame.aidministrator.nl

A Figura 39 apresenta detalhes da arquitetura Sesame mais recente. O módulo OMM (*Ontology Middleware Module*) foi desenvolvido para oferecer suporte a três questões importantes: (1) inferências para linguagem mais expressivas que RDF(S) como DAML+OIL, (2) segurança e (3) versionamento. Os módulos funcionais que tratam estas questões aparecem na figura como caixas cinzas na camada de módulos funcionais. Já na camada de "Abstração de Repositório e Inferência", novas camadas concretas foram criadas para tratar cada módulo correspondente. Maiores detalhes podem ser obtidos em [Kiryakov et al., 2002a e 2002b].

A implementação de um raciocinador DAML+OIL chamado BOR [Simov & Jordanov, 2002] foi integrada ao Sesame através de uma camada de inferência adicional, capaz de tratar ontologias definidas em DAML+OIL e obter novas tuplas RDF(S) inferidas. Os processos de inferência ocorrem tanto no nível do esquema quanto das instância, portanto novas tuplas RDF também são obtidas e persistidas em qualquer repositório que estiver em uso. A linguagem de consulta RQL continua habilitada a lidar com o repositório, uma vez que as tuplas persistidas utilizam RDF e RDF(S).

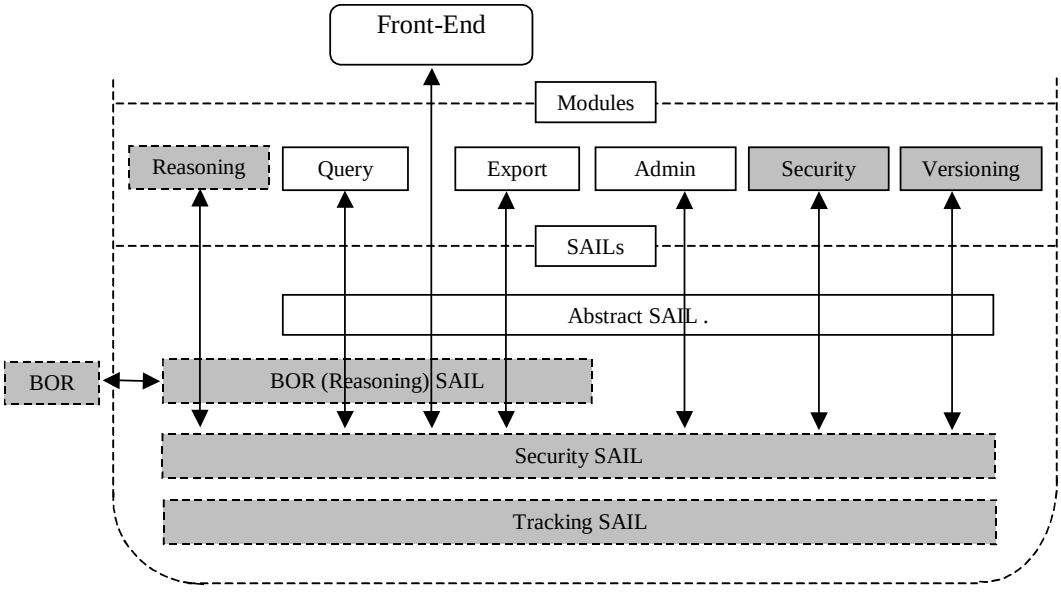

Figura 39 – Arquitetura Sesame com OMM e BOR [Kiryakov et al., 2002a e 2002b].

Vale comentar que atualmente o módulo BOR oferece suporte a parte das definições DAML+OIL, conforme detalhado em [Jordanov & Simov, 2002].

6.3. Experimento Realizado

Já implementamos a ontologia de Artes apresentada neste trabalho e validamos uma versão preliminar, onde as ontologias produzidas nas etapas de projeto conceitual e navegacional foram utilizadas na etapa de implementação.

A implementação faz uso de *templates*, como o esboçado na Figura 40. Apesar de não discutirmos o assunto neste trabalho, a especificação da interface concreta é derivada a partir de uma especificação de interface abstrata, onde os principais elementos de interface são definidos com base na capacidade de interação. A implementação concreta é responsável pela aparência e comportamento (*look and feel*). Portanto, o *template* inclui elementos abstratos como <<Referência Ativa>>, que irá gerar um caminho lógico para a página corrente, com referências a pontos intermediários relevantes, como por exemplo, Artistas>Pintores>Cubistas. De forma similar, <<anterior>> e <<prox>> geram links para os elementos anterior e próximo dentro de um mesmo contexto. O elemento <<escolhedor>> gera um *widget* de interface que permite uma escolha dentre vários elementos.

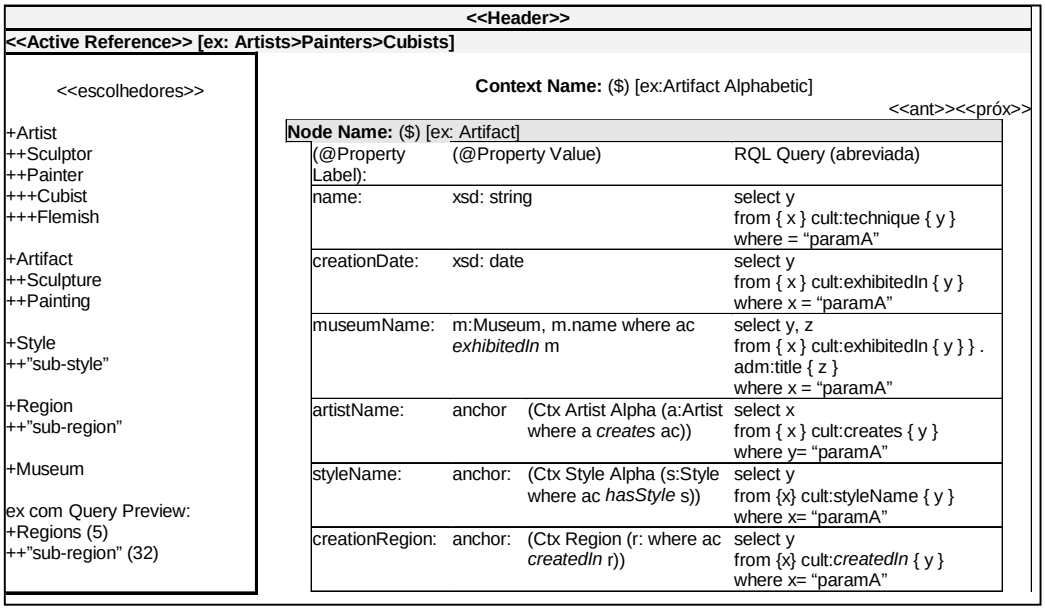

Figura 40 – Esboço de *template* para Contexto de Artefato Alfabético

O esboço de *template* para páginas HTML é apresentado na Figura 40 e mostra as consultas que devem ser feitas para recuperar os itens de informação apropriados. Este *template* diz respeito ao contexto de `Artefatos` em ordem alfabética, onde as variáveis que aparecem com os símbolos `$` e `@` são

preenchidas com nomes de classes e propriedades, respectivamente. Este esboço indica que as instâncias da classe navegacional `Artefato` dentro de um contexto de `Artefatos` em ordem Alfabética podem ser apresentada desta forma. A parte direita da tela contém uma tabela com três colunas, sendo que a terceira aparece no esboço apenas para indicar qual a consulta RQL responsável pelo correto preenchimento dos valores de propriedades (propriedades indicadas na 2ª coluna).

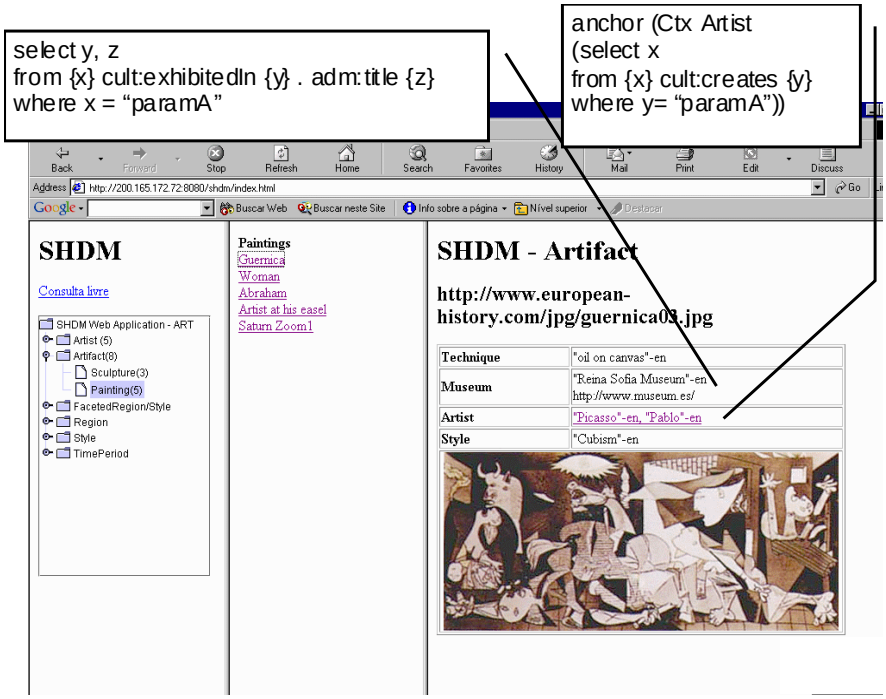

Figura 41 – Instância do template para a implementação da ontologia de artes