5
Arquitetura de implementação

5.1
Visão geral

Nossa arquitetura é caracterizada pela construção de um ambiente para execução de aplicações hipermídia definidas segundo o método SHDM, definido sobre um ambiente de armazenamento, inferência e consulta de dados semânticos. A Figura 25 apresenta um modelo dessa arquitetura, que ilustra como os artefatos produzidos na modelagem de uma aplicação são utilizados na sua execução.

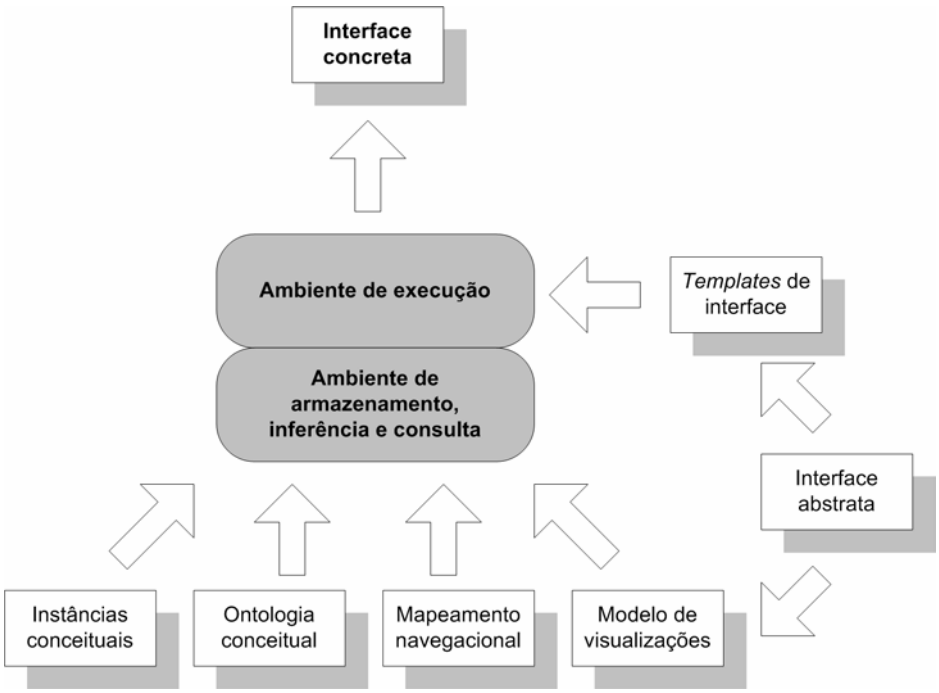

Figura 25 - Arquitetura para execução de aplicações SHDM

Neste modelo, após a etapa de modelagem que gera os artefatos descritos na Tabela 1, o modelo de visualizações e os *templates* de interface são gerados a partir da especificação da interface abstrata. Em seguida, a ontologia conceitual, as instâncias conceituais, a especificação do mapeamento navegacional e o modelo de visualizações são carregados no ambiente de armazenamento, inferência e consulta. Posteriormente, durante a execução da aplicação, as

informações inferidas desses artefatos são utilizadas na concretização das páginas da aplicação, com base nos *templates* de interface.

É importante observar que o modelo de visualizações não é um artefato do método, sendo apenas um recurso de implementação como explicado no decorrer da próxima seção.

5.2 Implementação

A arquitetura descrita na seção anterior foi implementada como uma aplicação *web*, seguindo o modelo cliente servidor da WWW, no qual um cliente – tipicamente um navegador – envia uma requisição HTTP ao servidor, que retorna – usualmente – uma página HTML como resposta. Dentro deste modelo a arquitetura geral do sistema foi baseada no padrão MVC (Krasner & Pope, 1988) com as adaptações pertinentes ao contexto da *Web*, onde o fluxo de processamento de uma requisição dá-se de acordo com a representação do diagrama de atividades mostrado na Figura 26.

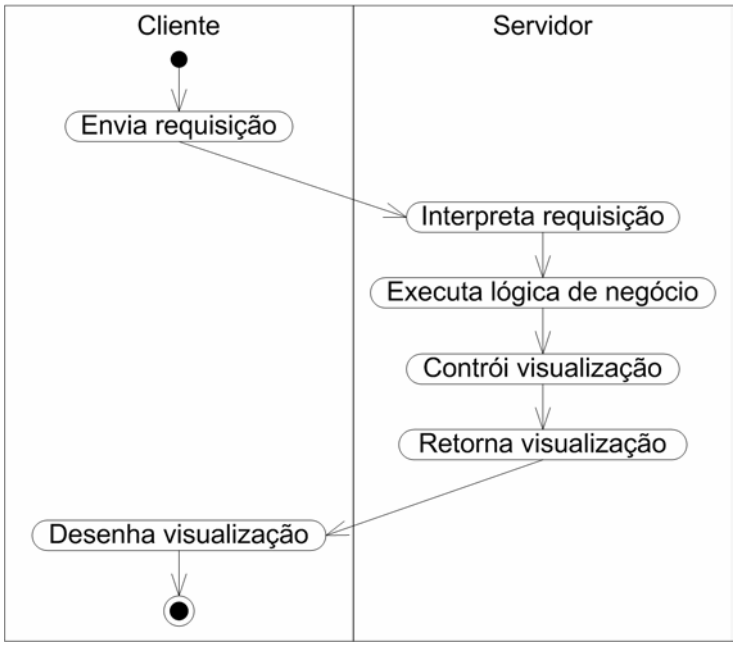

Figura 26 - Fluxo de processamento de uma requisição

O controlador recebe a requisição enviada pelo cliente, interpretando a mesma para então ativar as operações pertinentes da lógica de negócio (executadas sobre o modelo da aplicação). Em seguida o controlador determina

como os resultados da execução da lógica de negócio serão apresentados, ativando então os elementos responsáveis pela construção da visualização. Finalmente a visualização gerada é retornada ao cliente.

Nesse contexto o papel do ambiente de execução é atender requisições de nós, contextos de navegação e estruturas de acesso, cujos dados são então exibidos nas páginas HTML retornadas.

A aplicação em questão foi desenvolvida em Java sendo executada dentro da camada *web* da especificação J2EE, o que levou a adoção da arquitetura JSP Modelo 2 (Figura 27). Esta arquitetura é uma concretização do padrão MVC como descrito acima. Nela um *servlet* atua como o controlador, recuperando e instanciando objetos e *beans* que representam o modelo, cujos dados são então apresentados na visualização implementada por páginas JSP.

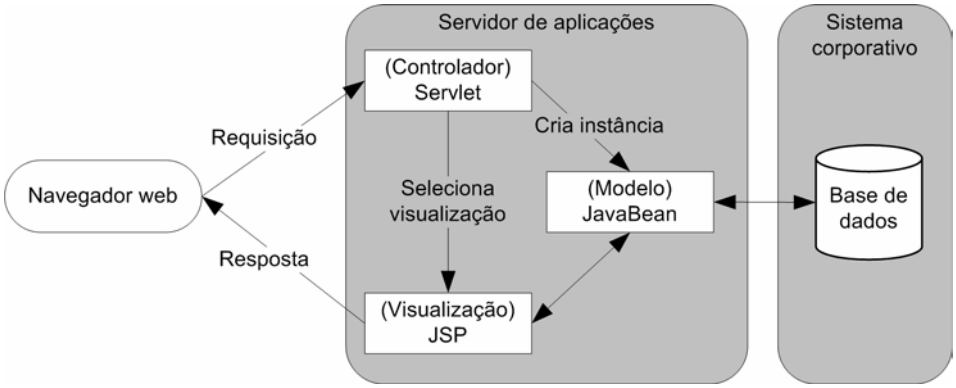

Figura 27 - Arquitetura JSP Modelo 2

Seguindo esta estrutura geral para tratamento de requisições, finalmente foi estabelecida para o sistema a arquitetura de módulos mostrada na Figura 28. O ambiente de armazenamento, inferência e consulta utilizado foi a API Jena 2[12], que fornece suporte para uso de ontologias definidas em RDFS e OWL, garantindo desta forma a capacidade do ambiente de executar aplicações SHDM cujas ontologias do modelo conceitual estejam definidas tanto em OWL quanto em RDFS.

[12] http://jena.sourceforge.net/

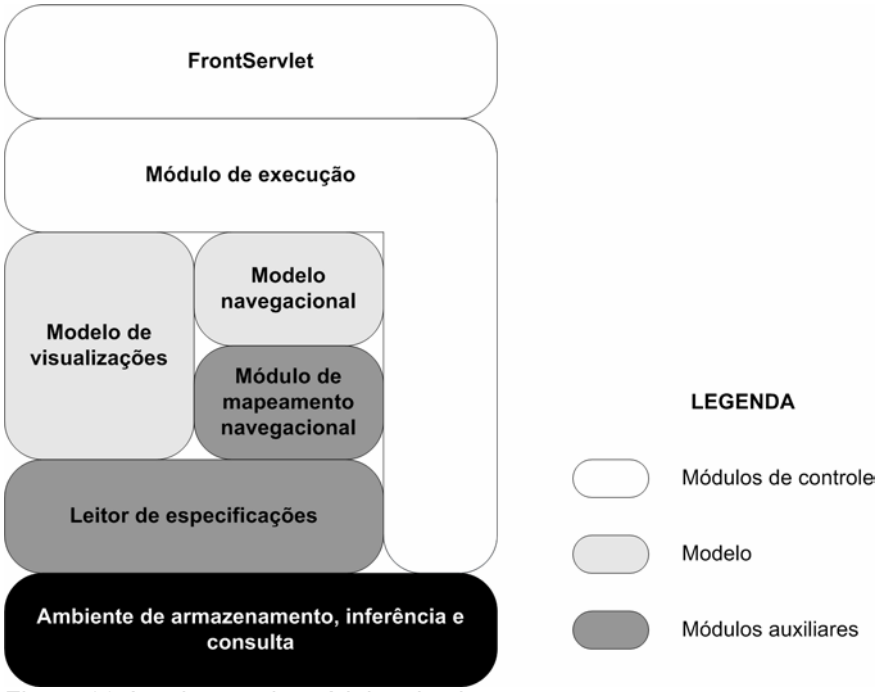

Figura 28 Arquitetura de módulos do sistema

Nesta arquitetura o módulo de controle, composto pelo *FrontServlet* e o módulo de execução, recupera os dados do modelo navegacional e do modelo de visualizações, que representam o modelo da aplicação, e então delega a apresentação da visualização para as páginas JSP, que correspondem aos *templates* de interface. Os módulos que compõem o modelo da aplicação são suportados por dois módulos auxiliares para leitura e manipulação da especificação do mapeamento navegacional, do modelo de visualizações e do modelo conceitual. Esta arquitetura de módulos é diretamente refletida na estrutura de pacotes do sistema, como mostrado na Figura 29.

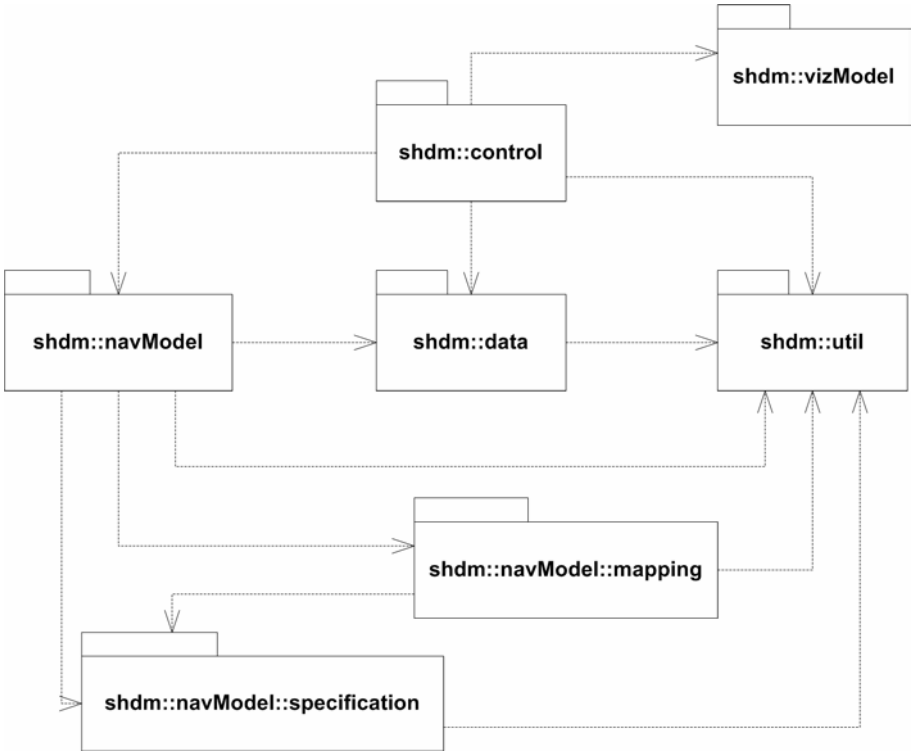

Figura 29 - Diagrama de pacotes do sistema

Apenas o pacote *shdm.util* que contém um conjunto de classes auxiliares não corresponde a nenhum dos módulos do sistema. O pacote *shdm.data* na verdade corresponde a um conjunto das classes do modelo navegacional que são utilizadas pelas páginas JSP. As classes deste pacote correspondem à interface pública do modelo navegacional e foram assim organizadas, para que os desenvolvedores das páginas JSP possam trabalhar com um conjunto mínimo de classes, sem a necessidade de tomar conhecimento de todos os detalhes das classes que implementam o modelo navegacional. A figura a seguir apresenta as classes e interfaces que compõem este conjunto mínimo.

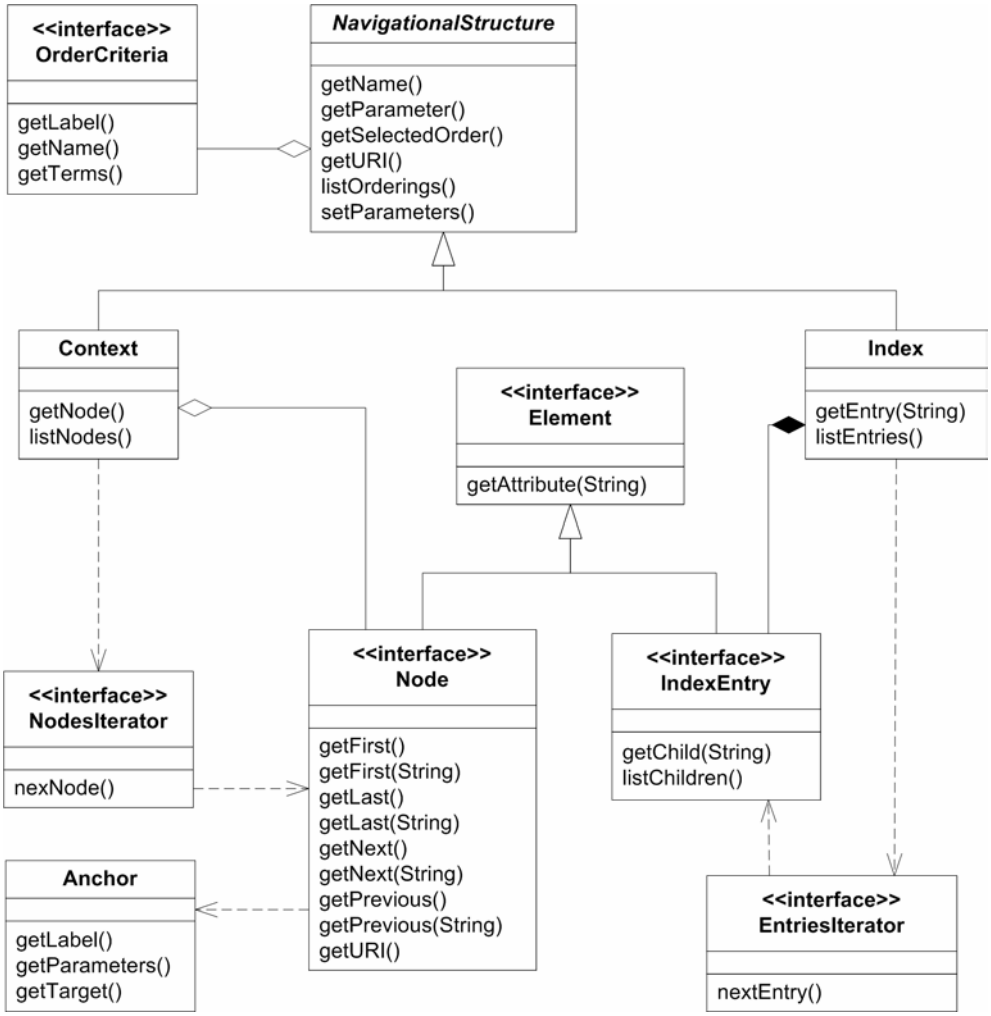

Figura 30 - Diagrama de classes do pacote *shdm.data*

A seguir são apresentados os fatores que levaram à evolução da arquitetura JSP Modelo 2 para a arquitetura de módulos final do sistema.

5.2.1
O modelo de visualizações

Como já foi dito anteriormente, o ambiente de execução atende requisições por nós, contextos de navegação e estruturas de acesso, retornando seus dados como páginas HTML. Entretanto, uma página pode exibir os dados de mais de um elemento SHDM, como no caso de uma página que apresente os dados de um nó em um contexto específico e um índice geral da aplicação. Para permitir que os dados exibidos em uma página desse tipo sejam recuperados sem a necessidade de múltiplas requisições HTTP, foi estabelecido o conceito de uma visualização. Uma visualização representa uma agregação de estruturas navegacionais

(contextos e índices) para exibição. Sendo assim as requisições tratadas pelo sistema são requisições por visualizações que retornam páginas HTML com os dados de todas as estruturas navegacionais que fazem parte da visualização requisitada.

5.2.2 O módulo de execução

Planejando a evolução do sistema para a futura incorporação de funcionalidades de hipermídia adaptativa, foi estabelecido o conceito de um módulo de execução (concretizado pela classe *Runtime*). Este módulo é utilizado pelo *servlet* que recebe as requisições por visualizações, para recuperação dos dados do modelo navegacional e do modelo de visualizações, encapsulando os mesmos através de uma aplicação do padrão *Facade* (Gamma et al., 1995). Com o isolamento dos sistemas subjacentes oferecido pelo módulo de execução, uma aplicação pode manter diferentes instâncias de *Runtime*, cada uma utilizando sua própria configuração de modelo navegacional e de visualizações.

5.2.3 Leitura e mapeamento

Como discutido no capítulo 3, o grafo de instâncias navegacionais resultante do mapeamento navegacional pode ser gerado em uma etapa de pré-processamento, ou as consultas de contextos e índices podem ser traduzidas em tempo de execução em consultas diretamente sobre as instâncias conceituais.

Em nossa implementação optamos pela alternativa do pré-processamento. Durante a inicialização da aplicação, os dados do modelo conceitual e a especificação do mapeamento navegacional são processados, para geração em memória de um grafo RDF de instâncias navegacionais que posteriormente é utilizado na recuperação dos dados de nós, contextos e estruturas de acesso.

5.2.4 O tratamento de uma requisição

Encerrando, apresentamos os passos envolvidos no tratamento de uma requisição, representados no diagrama de seqüência apresentado na figura a

seguir, que corresponde à execução do método *handleRequest()* do *FrontServlet*, que implementa o procedimento de tratamento de uma requisição por uma visualização.

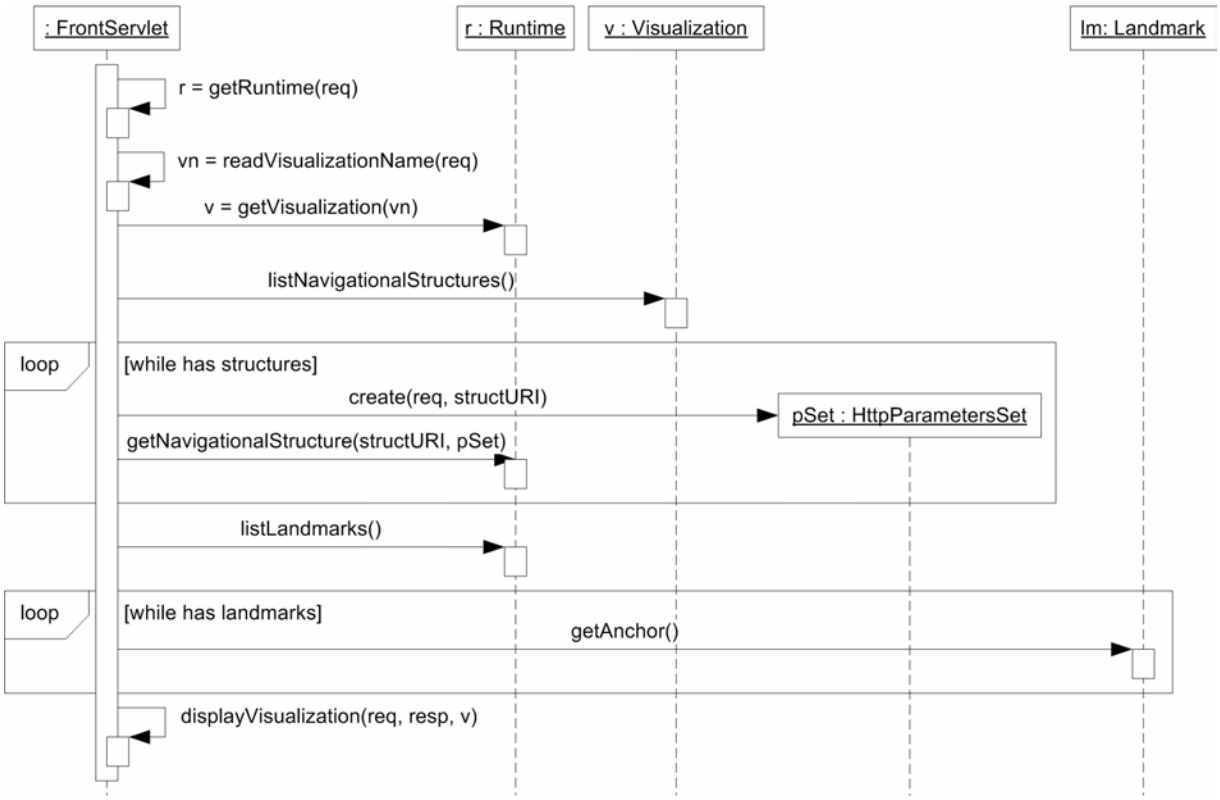

Figura 31 - Diagrama de seqüência do processamento de uma requisição

1. O *FrontServlet* recupera a instância do módulo de execução (*Runtime*) que será utilizada para recuperação dos dados do modelo.
2. O nome da visualização requisitada é recuperado dos dados da requisição.
3. A definição da visualização requisitada (*Visualization*) é recuperada através do módulo de execução.
4. Da definição da visualização obtém-se a lista das estruturas navegacionais que serão exibidas.
5. Para cada estrutura navegacional que consta na lista é criado um conjunto de parâmetros com os valores passados na requisição.
6. Através do módulo de execução os dados de cada estrutura navegacional são recuperados com base no respectivo conjunto de parâmetros. Os dados recuperados são armazenados no contexto da

requisição para posteriormente serem utilizados pelas páginas JSP na apresentação da visualização.

7. Através do módulo de execução é recuperada uma lista com todos os *landmarks* definidos para a aplicação.
8. A âncora de cada *landmark* é armazenada no contexto da requisição para posteriormente ser utilizada pelas páginas JSP na apresentação da visualização.
9. O processamento é redirecionado para a página JSP correspondente à visualização requisitada.

5.2.5 Funcionalidades implementadas

A implementação desenvolvida teve como foco principal os processos de mapeamento de instâncias navegacionais e concretização das abstrações navegacionais básicas. Sendo assim, os seguintes recursos não se encontram disponíveis nesta implementação:

- Navegação facetada.
- Leitura das especificações de assinaturas de operações específicas do domínio da aplicação.