PONTIFÍCIA UNIVERSIDADE CATÓLICA
DO RIO DE JANEIRO

Milena Ossorio Lamí

Um assistente dirigido por modelos para auxílio ao desenvolvimento de aplicações WWW

Dissertação de Mestrado

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós-graduação em Informática do Departamento de Informática da PUC-Rio.

Orientador: Prof. Daniel Schwabe

Rio de Janeiro
Março de 2015

PONTIFÍCIA UNIVERSIDADE CATÓLICA
DO RIO DE JANEIRO

Milena Ossorio Lamí

Um assistente dirigido por modelos para auxílio ao desenvolvimento de aplicações WWW

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre pelo Programa de Pós-graduação em Informática do Departamento de Informática do Centro Técnico e Científico da PUC-Rio. Aprovada pela Comissão Examinadora abaixo assinada.

Prof. Daniel Schwabe
Orientador
Departamento de Informática - PUC-Rio

Prof. Arndt von Staa
Departamento de Informática - PUC-Rio

Prof. Edward Hermann Haeusler
Departamento de Informática - PUC-Rio

Prof. José Eugênio Leal
Coordenador Setorial do Centro Técnico Científico - PUC-Rio

Rio de Janeiro, 16 de março de 2015

Milena Ossorio Lamí

Graduou-se em Ciência da Computação na Universidade de Havana (UH) em 2005. Trabalhou como pesquisadora no laboratório Tecweb da PUC-Rio.

Ficha Catalográfica

Lamí, Milena Ossorio

Um assistente dirigido por modelos para auxílio ao desenvolvimento de aplicações WWW / Milena Ossorio Lamí ; orientador: Daniel Schwabe. – 2015.

96 f. : il. (color.) ; 30 cm

Dissertação (mestrado)–Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2015.

Inclui bibliografia

1. Informática – Teses. 2. Ambiente de autoria. 3. Aplicações hipermídia. 4. SHDM. 5. Manipulação direta. 6. Programação por exemplo. 7. Wizard. 8. Desenvolvimento dirigido por modelos I. Schwabe, Daniel. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Para minha família, em especial minha avó,
A vocês todo o meu amor e carinho sempre.

Agradecimentos

Ao meu orientador, professor Daniel Schwabe, pelo ensino e toda a ajuda oferecida. Pelas constantes exigências, incentivo, seguimento permanente e implicação no meu processo de formatura.

Ao Jaisse Grela, o único, o melhor e imprescindível. Obrigada por teu apoio, paciência, amor sempre e por não me deixar fraquejar. Sem você, eu não teria tido forças.

Ao meu pai e a minha mãe, à Cecilia e o Nelson, ao meu tio Manuel e demais família que com muito amor e sacrifício fizeram possível minha educação e dedicação total a este objetivo.

Aos professores e funcionários do Departamento de Informática da PUC-Rio por esta oportunidade de aprendizado. À PUC-Rio e CAPES pelo suporte financeiro recebido durante a pesquisa.

Aos professores da Universidad de La Habana por me dar uma inestimável formação de base para poder realizar esta caminhada.

Ao Mauricio Henrique de Souza Bomfim e o Vagner Barbosa do Nascimento, pelas respostas prontas e precisas nos casos de dúvidas com o trabalho feito por eles.

Aos meus amigos todos.

Resumo

Lamí, Milena Ossorio; Schwabe, Daniel. **Um assistente dirigido por modelos para auxílio ao desenvolvimento de aplicações WWW.** Rio de Janeiro, 2015. 96p. Dissertação de Mestrado - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

As aplicações na WWW são exemplos de aplicações hipermídia. O desenvolvimento destas aplicações, mesmo utilizando metodologias de projeto, tem uma complexidade elevada. Existem propostas dirigidas por modelos para ajudar ao projetista, mas estas requerem de uma curva de aprendizado alta para os não familiarizados com os modelos. Este trabalho aborda este problema oferecendo uma abordagem que, fazendo uso de uma metodologia dirigida por modelos, permite a autoria de aplicações em um ambiente mais próximo à intenção do usuário. Se apresenta uma ferramenta com características de assistente (*wizard*) que permite a criação de aplicações através de exemplos, utilizando interfaces com dados concretos. O assistente usa aspectos da técnica de programação por exemplo e do estilo de interação de manipulação direta que contribuem para facilitar o desenvolvimento.

Palavras-chave

Ambiente de autoria; aplicações hipermídia; SHDM; manipulação direta; programação por exemplo; wizard; desenvolvimento dirigido por modelos

Abstract

Lamí, Milena Ossorio; Schwabe, Daniel (Advisor). **A Model-driven wizard to aid in developing Web applications.** Rio de Janeiro, 2015. 96p. MSc. Dissertation - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Web applications can be seen as examples of hypermedia applications. Developing such applications is a complex endeavor, even when using design methods. There are model-driven methods aimed at helping the designer, but they still require a steep learning curve for those unfamiliar with the models. This work addresses this problem through a model-driven wizard that helps the designer through the use of examples and concrete data-driven interfaces. This wizard uses direct manipulation techniques to help easing the designer's tasks.

Keywords

Authoring environments; Web applications; SHDM; model-driven development; direct manipulation; wizard.

Sumário

Lista de Figuras

Lista de Quadros

1
Introdução

A web é hoje um dos principais ambientes para desenvolver aplicações. Como a WWW é um ambiente hipermídia por definição, as aplicações obtidas nele são consequentemente hipermídia.

Hipermídia é um estilo de se construir sistemas para a criação, manipulação, apresentação e representação de conteúdo no qual a informação é armazenada em um conjunto estruturado de itens de informação. Entendemos, os sistemas de informação avançados como aqueles que permite a representação do conhecimento como uma coleção de nós interconectados por elos para o processamento pelo homem.

O desenvolvimento de aplicações hipermídias é um complexo, pois são muitos os fatores envolvidos que precisam ser controlados, incluindo entre outros a orientação, a interatividade e estruturação dos componentes.

O problema a abordar é que o desenvolvimento de aplicações hipermídia na web é de difícil realização. Este problema difere do problema do projeto de aplicações tradicionais pois uma característica das aplicações hipermídia que não existe em outras aplicações é o conceito de navegação.

A navegação hipermídia é uma operação que é definida sobre uma estrutura. Dado um conjunto de itens de informação que tem relações entre si, se trata de definir uma estrutura que permita ao usuário percorrer esse conjunto para apoiar determinadas tarefas. Percorrer esses conjuntos através da estrutura é o que se chama de navegação.

O problema específico é projetar essas estruturas de navegação de forma que elas suportem as tarefas a realizar. Existe um modelo de domínio com classes e relações entre elas. Mas nem sempre o suporte à tarefa é obtido diretamente do percorrimento direto do modelo de domínio. Para um conjunto de itens tem-se diversas estruturas possíveis, o seja, para um mesmo modelo de dominío tem-se vários hipertextos. A ideia é escolher o mais adequado para a tarefa. Em outras palavras, definir o modelo de navegação é especificar visões navegacionais sobre algum modelo de dados.

Já existem metodologias que propõem um vocabulário ou metamodelo de navegação em termo dos quais se pode definir um modelo de navegação, mais ainda são de difícil entendimento pela maioria dos desenvolvedores. Estas metodologias descrevem os passos necessários para o desenvolvimento de uma aplicação hipermídia, sendo utilizadas como forma de comunicação entre projetistas, implementadores e usuários. O processo de desenvolvimento é apresentado como uma sequência de atividades de modelagem que pode ser mapeada em quatro etapas: Modelagem Conceitual do domínio do problema, Projeto de Navegação, Projeto de Interface e Implementação. O processo se dá de forma iterativa e incremental seguindo o desenvolvimento de cada etapa e produzindo novos modelos ou enriquecendo os já existentes.

Uma das iniciativas mais proeminentes é o método *Semantic Hypermedia Design Method* (SHDM) (Lima, 2003), que é um método dirigido por modelos para o desenvolvimento de aplicações hipermídia (i.e., baseadas em dados semânticos na *Linked Data Cloud*) e o Synth, que é um ambiente de autoria de aplicações projetadas segundo o método SHDM.

Tanto o SHDM como outras propostas na literatura são dirigidas por modelos. Um desenvolvedor não acostumado a métodos dirigidos por modelos, normalmente requere passar por uma curva de aprendizado íngreme, o que frequentemente o leva a se desencorajar de usar e acaba não se beneficiando dos métodos.

Assim, a motivação é criar ferramentas que permitam ao desenvolvedor que desconhece a metodologia ou conhece muito pouco, desenvolver uma aplicação seguindo os modelos da mesma; e que este processo seja o mais natural, transparente e intuitivo possível.

O método SHDM é uma evolução do *Object-Oriented Hypermedia Design Method* (OOHDM) (Schwabe & Rossi, 1998), (Rossi, 1996) e, por esse motivo, manteve os seus fundamentos, além de acrescentar formalismos e primitivas introduzidos pela Web Semântica. Esta metodologia permite o projeto de aplicações hipermídia baseadas em ontologias, descritas através de metadados.

O método OOHDM é uma abordagem baseada em modelos e provê primitivas de projeto de alto nível e mecanismos de abstração do paradigma de orientação a objetos. O OOHDM apresenta uma maneira sistemática de modelagem, na qual se podem estabelecer aspectos do domínio, a estrutura da aplicação hipermídia e sua semântica navegacional, independentemente do que concerne à implementação. A importância desta metodologia reside no enfoque navegacional da aplicação, bem como o tratamento do projeto de interfaces de

forma independente da navegação, e também das tecnologias e ambientes de execução.

O Synth é um ambiente de desenvolvimento que dá suporte à construção de aplicações projetadas segundo o método dirigido por modelos SHDM. No SHDM a aplicação web é definida como uma visão navegacional sobre algum modelo específico de domínio, o seja, como acesso a visões navegacionais de algum modelo de dados através de interfaces. O metamodelo SHDM implementado no Synth é definido por uma ontologia. O ambiente Synth fornece um conjunto de módulos capazes de receber como entrada os modelos gerados na execução das etapas do método SHDM e produzir como saída uma aplicação hipermídia descrita por estes modelos. O Synth também dispõe de um ambiente de autoria que facilita a inserção e edição destes modelos através de uma interface gráfica de formulários que pode ser executada em qualquer navegador de internet. Através desta interface é possível executar a aplicação enquanto ocorre a sua construção e, dessa forma, validá-la a cada passo do seu desenvolvimento.

O problema com esta abordagem é que ainda a especificação dos modelos na interface de formulários requer um grau de abstração e conhecimentos tais que dificulta seu uso por um usuário não tão especialista. Com o Synth é preciso ter domínio do método SHDM, das classes e propriedades do metamodelo e seu uso, incluindo conhecimento das linguagens de manipulação de recursos RDF e da linguagem Ruby para a definição de operações, para assim acompanhar a representação da aplicação resultante.

Mesmo utilizando um método como SHDM para desenvolver aplicativos hipermídia, a complexidade do projeto é elevada. O projetista tem que ter domínio da metodologia e em cada etapa várias decisões de projeto precisam ser tomadas. É por isso que a ideia é continuar fazendo mais natural o processo de criação de aplicações ao tempo que se representam as funções do modelo SHDM.

De ai, surgiu a idéia de uma aplicação que fazendo uso do próprio Synth oferecera uma interface de desenvolvimento mais acessível ao usuário não especializado e mais próxima à sua aparência no produto terminado.

1.1. Objetivos

Esta dissertação apresenta uma abordagem para resolver o problema do projeto de navegação nas aplicações hipermídia. Nossa proposta, fazendo uso de uma metodologia dirigida por modelos, permite a autoria de aplicações mesmo por usuários não experientes. A ideia é encapsular a geração dos modelos próprios da metodologia e que este processo fique transparente ao usuário que vai interatuar com representações próximas à sua intenção na aplicação que quer desenvolver.

A abordagem inclui o desenvolvimento de um ambiente de autoria *What You See Is What You Get* (WYSIWYG) baseado em exemplos para devolver uma aplicação baseada em modelos. O produto obtido neste ambiente é o mesmo atingido com a interface de autoria do Synth atual, mas com um procedimento diferente. O processo começa com a construção de protótipos que são a definição da interface abstrata a partir de exemplos e escolhas do usuário, gradualmente gerando internamente os modelos e estruturas SHDM correspondentes aos exemplos informados. Portanto, a saída desta aplicação vai ser a entrada do Synth para obter a interface instanciada na aplicação real.

1.2. Estrutura da Dissertação

Os temas apresentados nessa dissertação estão organizados em capítulos da seguinte maneira:

- **Capítulo 2 – Fundamentos**: apresenta uma breve revisão dos principais conceitos envolvidos na pesquisa.
- **Capítulo 3 – Trabalhos relacionados**: descreve uma série de ferramentas para navegar na Web Semântica e para a apresentação de dados RDF, assim como outras abordagens que têm aspectos em comum com nossa proposta mais não reúnem os requisitos desejados.
- **Capítulo 4 – Assistente. Concepção e implementação**: apresenta os principais aspectos da construção do assistente: as abordagens para interação do usuário, requisitos iniciais, as decisões da arquitetura e detalhes relevante da implementação.

- **Capítulo 5 – Exemplos de Uso**: ilustra através de um exemplo de cenário desenvolvido com o ambiente Synth como representar no assistente este exemplo para obter o mesmo resultado.
- **Capítulo 6 – Avaliação**: Se faz uma análise sobre como se planejou o que se queria experimentar. Apresenta a avaliação do assistente.
- **Capítulo 7 – Conclusão e Trabalhos Futuros**: descreve as lições aprendidas, contribuições, e trabalhos futuros.

2
Fundamentos

A extensão da Web atual através da Web Semântica inicia uma revolução de novas possibilidades. As tecnologias da Web Semântica estão cada vez mais acessíveis e por isso, o cenário de uma aplicação que consuma dados publicados segundo os princípios de *Linked Data* é considerado comum.

A crescente disponibilização de dados e ontologias seguindo os padrões da Web Semântica tem levado à necessidade de criação de métodos e ferramentas de desenvolvimento de aplicações que considerem a utilização e disponibilização dos dados distribuídos na rede segundo estes princípios. Assim, evidencia-se que é preciso uma metodologia ou ferramenta adequada de projeto de sistemas que contemple além da idéia de hipertexto os dados semânticos, especificando e documentando o projeto de maneira clara e concisa.

O uso da metodologia *Semantic Hypermedia Design Method* (SHDM) na proposta serve a este propósito, dado que ajuda ao projeto de aplicações hipermídia em geral e deve seu qualificativo de Semantic ao fato de que propõe o uso de modelos para o desenvolvimento de aplicações que usam qualquer dado estruturado. Por exemplo, os dados semânticos da nuvem de *Linked Data* (*Linked Data Cloud*). Isto faz com que, além dos principais conceitos, seja considerada parte da terminologia usada na área da Web Semântica.

2.1.
Web semântica

A Web Semântica é um trabalho colaborativo liderado por *World Wide Web Consortium* (W3C), especificamente pelo projeto *W3C Linking Open Data* (LOD) (2007). Promovendo a inclusão de conteúdo semântico nas páginas web, a Web Semântica está dirigida a converter a web atual, dominada por documentos não estruturados e semiestruturados, em uma "Web de dados". Ela interliga significados de palavras e, neste escopo, tem como finalidade conseguir atribuir um significado (sentido) aos conteúdos publicados na Internet de modo que seja perceptível tanto pelo humano como pelo computador. O termo foi designado por

Tim Berners-Lee (inventor da WWW) para se referir a uma web de dados que pode ser processada por máquinas.

A Web Semântica é uma extensão da Web com um espaço de dados global baseado em padrões abertos. Ela provê um framework comum que permite que os dados sejam compartilhados e reusados entre aplicações. A integração das linguagens ou tecnologias *eXtensible Markup Language* (XML), *Resource Description Framework* (RDF), arquiteturas de meta-dados, ontologias, as linguagens para definição de vocabulários RDFS[1] e OWL[1], a linguagem de consultas SPARQL[2], as práticas Linked Data, entre outras, favorecem a interoperabilidade e cooperação.

RDF é um modelo padrão para intercambiar dados na Web. Provê uma linguagem comum que as aplicações podem utilizar para intercambiar informação, sem perda de significado. O modelo consiste no uso de identificadores da Web (chamados de *Uniform Resource Identifiers*, ou URIs) para designar recursos e na descrição destes recursos em termos de propriedades e seus valores. Esta descrição e feita por meio de afirmações em forma de triplas <Sujeito, Predicado, Objeto>.

Uma ontologia é um vocabulário formalizado de termos ou conceitos que cobrem um domínio específico do conhecimento e é compartilhado por uma comunidade de usuários. Especifica as definições dos termos descrevendo suas relações com outros termos.

Linked Data refere-se a um conjunto de melhores práticas para publicar e conectar dados estruturados na web. Estas melhores práticas têm sido adotadas por um número crescente de provedores de dados, levando à criação do espaço de dados global, a Web de Dados ou Web Semântica.

Os *datasets* publicados na Web como *Linked Data* são chamados *Linked Data Cloud*. (Heath & Bizer, 2011)

[1] RDFS e OWL são linguagens ou vocabulários utilizados na Web Semântica para a definição de ontologias.

http://www.w3.org/TR/rdf-schema

http://www.w3.org/TR/owl-guide

[2] Linguagem de consulta para fontes RDF. http://www.w3.org/TR/sparql11-query

2.2. OOHDM

Object-Oriented Hypermedia Design Method (OOHDM) é um método dirigido por modelos que utiliza práticas de orientação a objetos para o projeto de aplicações hipermídia (Rossi, 1996). Segundo este método a aplicação é modelada em várias etapas para estabelecer os aspectos do domínio, a estrutura e a semântica navegacional, independente da implementação. Estas etapas são: Levantamento de Requisitos, Modelagem Conceitual, Modelagem Navegacional, Projeto de Interface Abstrata e Implementação. O processo descrito por elas é iterativo, incremental e de prototipagem rápida, produzindo novos modelos ou enriquecendo os já existentes.

2.3. SHDM

Semantic Hypermedia Design Method (SHDM), proposto originalmente por (Lima, 2003), é uma abordagem baseada em modelos para projetar aplicações hipermídia. O SHDM é uma evolução do método OOHDM e, por esse motivo, manteve os seus fundamentos, enriquecendo cada etapa com os formalismos e primitivas introduzidas pela Web Semântica. No SHDM a aplicação web é definida como uma visão navegacional sobre algum modelo específico de domínio, o seja, como acesso a visões navegacionais de algum modelo de dados através de interfaces. Esta metodologia permite o projeto de aplicações hipermídia baseadas em ontologias, descritas através de metadados.

O método SHDM foi extensamente discutido em trabalhos anteriores, principalmente em (Lima, 2003), (Szundy, 2004) e (Nunes, 2005), assim como algumas outras mudanças introduzidas podem se consultar em (Bomfim, 2011)

2.4. Synth

O Synth é um ambiente de desenvolvimento que dá suporte à construção de aplicações projetadas segundo o método dirigido por modelos SHDM. No SHDM a aplicação web é definida como uma visão navegacional sobre algum modelo específico de domínio, o seja, como acesso a visões navegacionais de algum modelo de dados através de interfaces. O metamodelo SHDM implementado no Synth é definido por uma ontologia. O ambiente Synth fornece um conjunto de módulos capazes de receber como entrada os modelos gerados

na execução das etapas do método SHDM e produzir como saída uma aplicação hipermídia descrita por estes modelos. O Synth também dispõe de um ambiente de autoria que facilita a inserção e edição destes modelos através de uma interface gráfica de formulários que pode ser executada em qualquer navegador de internet. Através desta interface é possível executar a aplicação enquanto ocorre a sua construção e, dessa forma, validá-la a cada passo do seu desenvolvimento.

3
Trabalhos relacionados

A maioria das abordagens existentes relacionadas com o estilo de interação de manipulação direta se circunscrevem à edição de documentos para dar formato. Nos casos das propostas de programação por exemplo a literatura reflete só abordagens para generalizações a partir de exemplos usando técnicas de aprendizado de máquina (*machine learning*), onde o foco está no treinamento através de casos para obter funções, algoritmos e programas.

De modo que, não foram identificados outros trabalhos que tivessem os requerimentos pretendidos mas apresentaremos alguns que tem relação ou fazem parte do que se quer.

Uma série de ferramentas para navegar na Web Semântica e para a apresentação de dados RDF têm sido desenvolvidas até o momento. Os navegadores da Web Semântica servem a um propósito semelhante para os dados semânticos como os navegadores da Web convencional fazem para os documentos HTML.

3.1.
Navegadores da Web Semântica

Estas ferramentas permitem aos usuários explorar grandes quantidades de dados de forma associativa. No entanto, a apresentação dos dados é limitada principalmente a listas tabulares de todas as instâncias, propriedades e valores de propriedades. Exemplos de tais ferramentas são *Tabulator* (Berners-Lee, et al., 2008) e *Disco*.

As interfaces de usuário e ferramentas de busca da Web Semântica podem explorar a semântica da estrutura de dados subjacente para permitir aos usuários processar os dados sob diferentes perspectivas e formular consultas mais específicas e complexas. O potencial para fazer os dados semânticos exploráveis tem sido demonstrado por uma variedade de ferramentas. Estas utilizam as diferentes relações semânticas para a criação de facetas de busca, permitindo que os usuários filtrem os dados de forma flexível. O uso só dos atributos previamente definidos, organizados em facetas para construir as consultas, evita a ambiguidade

da linguagem natural. Técnicas visuais para formular consultas complexas têm sido desenvolvidas em trabalhos anteriores.

A maioria das abordagens exibe as facetas[3], assim como o conjunto de resultados, como listas em diferentes posições sobre a tela. Alguns exemplos são ferramentas como: *mSpace* (Schraefel, et al., 2005), *Flamenco* (Hearst, English, Sinha, Swearingen, & Yee, 2002), *Longwell* (Longwell, 2005) e *Haystack* (Quan, Huynh, & Karger, 2003). As facetas fornecidas por essas ferramentas estão limitadas a aquelas diretamente conectadas exclusivamente. A filtragem hierárquica, no entanto, que permite que também as facetas indiretamente conectadas sejam utilizadas para a construção de consultas, não é suportada por essas ferramentas.

Ferramentas como *Parallax* (Huynh & Karger, Parallax and companion: Set-based browsing for the Data Web, 2009), *Humboldt* (Kobilarov & Dickinson, 2008), *Tabulator*, *Nested Faceted Browser* (Huynh D. , Nested Faceted Browser, 2009) e *gFacet* (Heim, Ertl, & Ziegler, 2010), permitem uma filtragem hierárquica. Assim, as opções possíveis para a construção de uma consulta não se restringem à periferia direta em torno de um conjunto de resultados, mas podem incluir dimensões distantes ligadas por outras facetas.

Uma série de ferramentas, como *Exhibit* (Huynh, Karger, & Miller, 2007), *Dido* (Karger, Ostler, & Lee, 2009), *mSpace*, e */facet* (Hildebrand, Ossenbruggen, & Hardman, 2006) já fornecem formas rápidas de implementar interfaces mais elaboradas sobre dados estruturados. A maioria destas ferramentas trabalha sobre coleções de dados relativamente simples (uma coleção de recursos).

Por exemplo, *Exhibit* permite aos desenvolvedores criar uma poderosa interface de exploração de dados sem nenhum conhecimento de tecnologia de banco de dados ou programação, enquanto mSpace permite a instalação e configuração de um navegador facetado escalável sobre um SPARQL endpoint através de um assistente de instalação.

Dido não processa dados semânticos em formato RDF, mas fornece meios para a manipulação direta e criação de dados, assim como para a inclusão de diretivas de apresentação. Os dados embutidos são guardados como pares chave-valor em JSON (*JavaScript-Object Notation*). A seleção e visualização são definidas com lentes e visões. No editor, o usuário pode escolher diretamente a forma como

[3] Faceta – forma na qual um recurso pode ser classificado. A faceta representa uma propriedade no domínio de dado. Selecionando o valor de uma faceta o usuário automaticamente seleciona todos os elementos que tem tal propriedade com aquele valor selecionado.

os dados selecionados devem ser apresentados, por exemplo, como uma lista ou tabela.

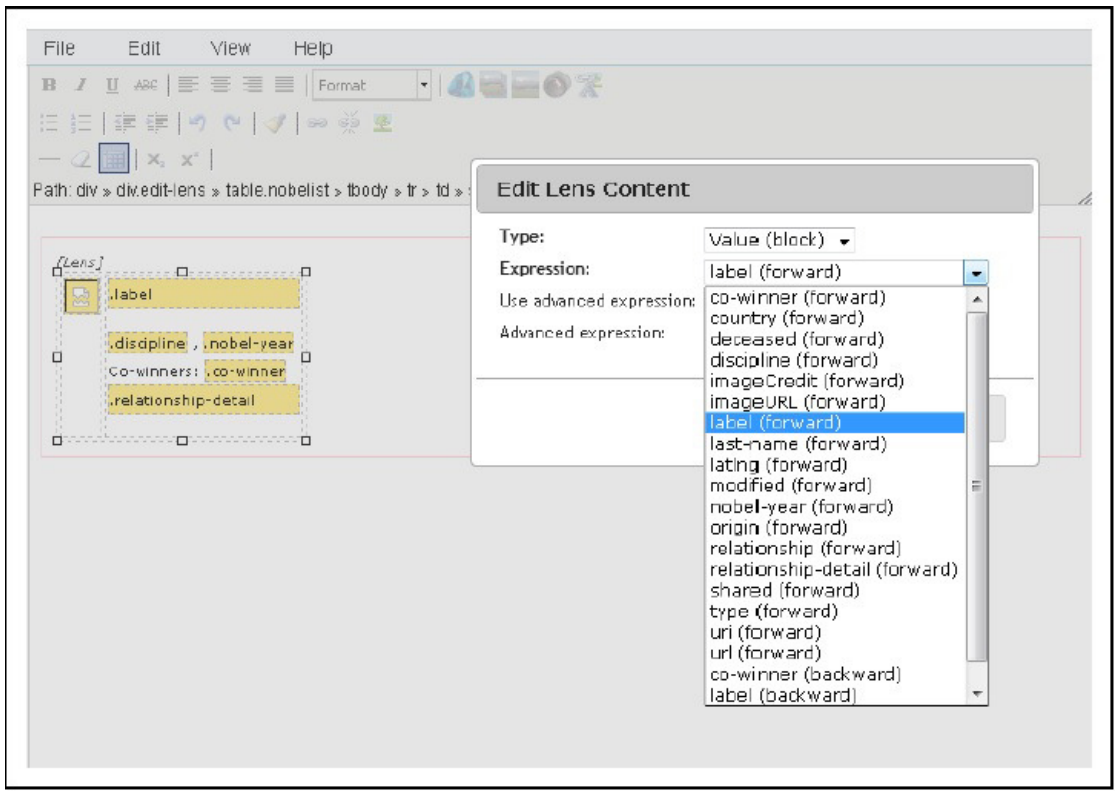


Figura 3.1 Dido. Editor de lentes

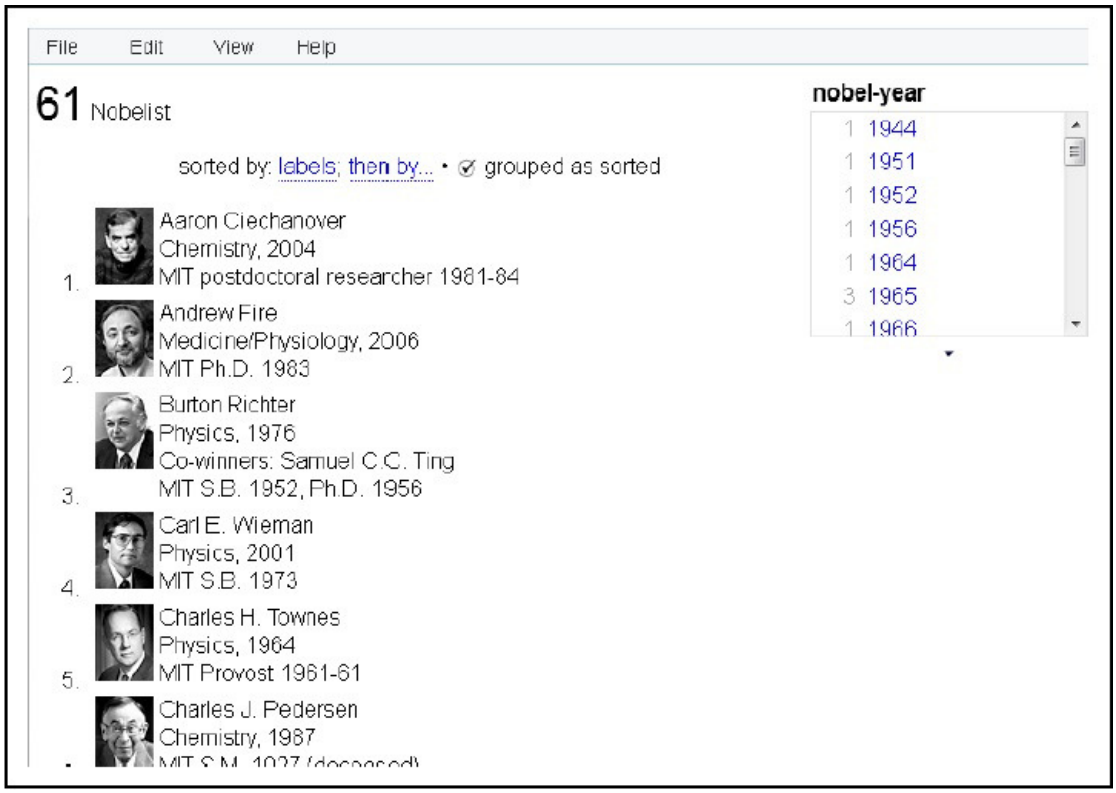


Figura 3.2 Dido. Resultado gerado

Por outra parte, *Tabulator* é um navegador de dados genérico. Ele pode ser usado para explorar a Web de dados e formular consultas representando os resultados em uma estrutura tabular. As consultas são feitas seguindo uma estratégia de "consulta por exemplo". Uma das ideias a contemplar em nosso trabalho.

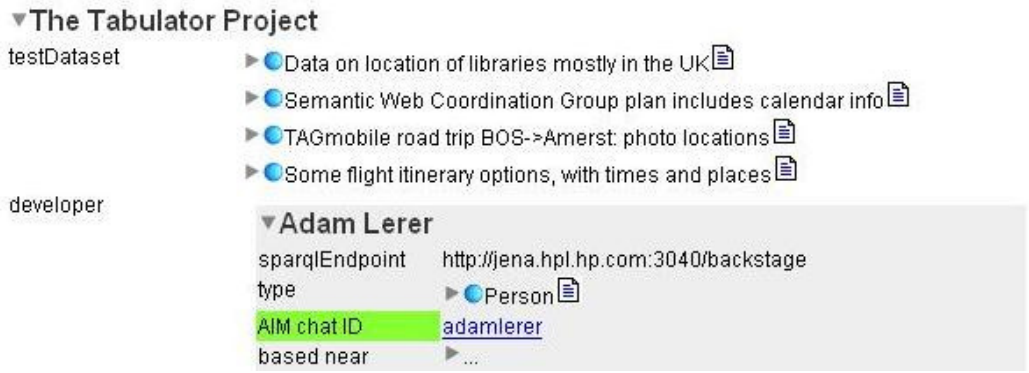


Figura 3.3 Tabulator. Consulta por exemplo mediante seleção da propriedade de um indivíduo

Figura 3.4 Resultado generalizado da consulta por exemplo

Navegadores como *Parallax*, *Humboldt*, *Explorator* (Araujo, Schwabe, & Barbosa, 2009), *gFacet* e *Visor* (Popov, Schraefel, Hall, & Shadbolt, 2011) introduzem a noção de pivoting e navegação orientada a conjunto. Assim mesmo, navegadores como *Parallax* e *Tabulator* permitem aos usuários selecionar dados arbitrários a partir do grafo explorado e visualizá-los através da entrada deles em *widgets* tais como mapas, gráficos, cronogramas e visões do calendário.

Os navegadores genéricos com frequência incorrem em representações também genéricas. Além disso, a interação e navegação estão estreitamente relacionadas ao modelo RDF subjacente, que muitas vezes é demasiado granular para o usuário final e exige transformações complexas, muitas seleções ou

encontrar representações adequadas antes que possam ser usadas para resolver uma necessidade de informação particular.

Eventualmente, o usuário pode alcançar o resultado desejado, no entanto, o processo pode ser longo e sujeito a erros. Para a maioria dos usuários finais, essas interações são muitas vezes demasiado complexas e demoradas. Além disso, os navegadores raramente capturam essa transformação dos dados em conhecimento útil produzido pelos usuários e perdem a oportunidade de oferecer resultados anteriores como uma visão a sugerir para os novos usuários do navegador.

Embora essas ferramentas sejam capazes de fornecer uma visão geral dos conjuntos de dados, elas não produzem estruturas de navegação e/ou contextos específicos para apoiar tarefas de um domínio dado. Estas limitações evitam uma interação mais rica com os dados semânticos e previnem de obter uma interface mais amigável para o usuário casual. Tudo o que constitui objetivo da abordagem perseguida.

3.2. Frameworks

Framework é simplesmente uma coleção de vários componentes únicos com uma cooperação predefinida entre eles. Alguns dos componentes de um "framework" são projetados para serem substituídos ou configurados e apresentam um refinamento predefinido. Estes componentes, chamados de *hot spots*, são a parte vital de um *framework*. Quando se consegue adaptar os *hot spots* de um *framework* permitindo atingir a flexibilidade desejada, então o *framework* é dito bem projetado (*well-designed*)

Aplicações que são construídas a partir de um *framework* podem reutilizar não apenas o código fonte mas o projeto de arquitetura aumentando a padronização da estrutura da aplicação. Esta característica do *framework* é considerada como a mais importante. Ao adaptar um *framework* para produzir uma aplicação específica têm-se também a vantagem da redução do volume de código fonte que precisa ser escrito por programadores que fazem a adaptação.

Este trabalho visa a especificação de um *framework* de projetos hipermídia, portanto antes de definir o que é um *framework* de projeto de aplicações hipermídia faz-se necessário descrever uma aplicação hipermídia.

Aplicação hipermídia: Coleção de objetos (itens de informação), com estrutura, que pode ser navegada e possivelmente processada pelos usuários para cumprir uma ou mais tarefas.

Framework hipermídia: Uma definição genérica dos possíveis objetos de uma aplicação, e das possíveis estruturas (arquiteturas) de navegação e processamento destes objetos.

Pode-se então definir *frameworks* de aplicação hipermídia como sendo um conjunto de itens de informação e o comportamento desta coleção de objetos. Os itens de informação definem o domínio do *framework* e o comportamento destes itens define a navegação e o processamento de uma aplicação hipermídia. O reuso proporcionado por estes *frameworks* é de mais alto nível que o provido pelos tradicionais, visto que os *frameworks* hipermídia permitem a reutilização de projeto e não apenas de código.

Quando se fala de reutilização de projeto deve-se ser capaz de caracterizar um domínio de aplicação, quais são as possíveis formas de navegação e como será dado suporte às tarefas que o usuário tem que cumprir. O domínio de uma aplicação hipermídia engloba todo o universo de informações relevantes para a aplicação em questão, e deve ser capturado tão imparcialmente quanto possível, sem ter em conta os tipos de usuários e suas tarefas.

A estrutura de navegação de uma aplicação hipermídia é o item que mais difere das aplicações convencionais. Os objetos de navegação são obtidos através de um mapeamento dos objetos do domínio e representam as informações que serão processadas pelo usuário, descrevendo ainda como estas informações serão apresentadas e a possível navegação entre elas.

Para poder reutilizar projetos de aplicações hipermídia é necessário generalizar estas questões de domínio e navegação.

Mesmo que os *frameworks* apresentados nos seguintes subepígrafes sequer falem sobre reuso de domínio e navegação, eles serão apresentados como evidência do estado da arte dos *frameworks* para aplicações semânticas e como justificativa da necessidade e o valor agregado da nossa abordagem.

3.2.1. Extensão Halo (Halo Extension)

A *Extensão Halo* é uma extensão de *Semantic MediaWiki* (SMW), a fim de facilitar a utilização de Wikis Semânticos por uma grande comunidade de usuários. O foco principal do desenvolvimento foi a criação de ferramentas que aumentam a facilidade de uso de recursos SMW e anunciar os benefícios imediatos de conteúdo semanticamente enriquecido.

Características da extensão Halo:

- Navegação wiki melhorada - funcionalidades para facilitar e acelerar a navegação e acesso a artigos, assim como aos dados semânticos na wiki
- Autoria de conhecimento melhorada - funcionalidades para permitir a adição fácil e expressiva de dados semânticos à wiki
- Recuperação de conhecimento simplificada – funcionalidades para consultar dados semânticos e permitir o acesso à informação armazenada na wiki.

Funcionalidades de navegação:

- Navegador de ontologia baseado em uma interface gráfica de usuário, permite a navegação intuitiva e a adaptação da ontologia da wiki para a pesquisa de informações de instâncias e de propriedades.

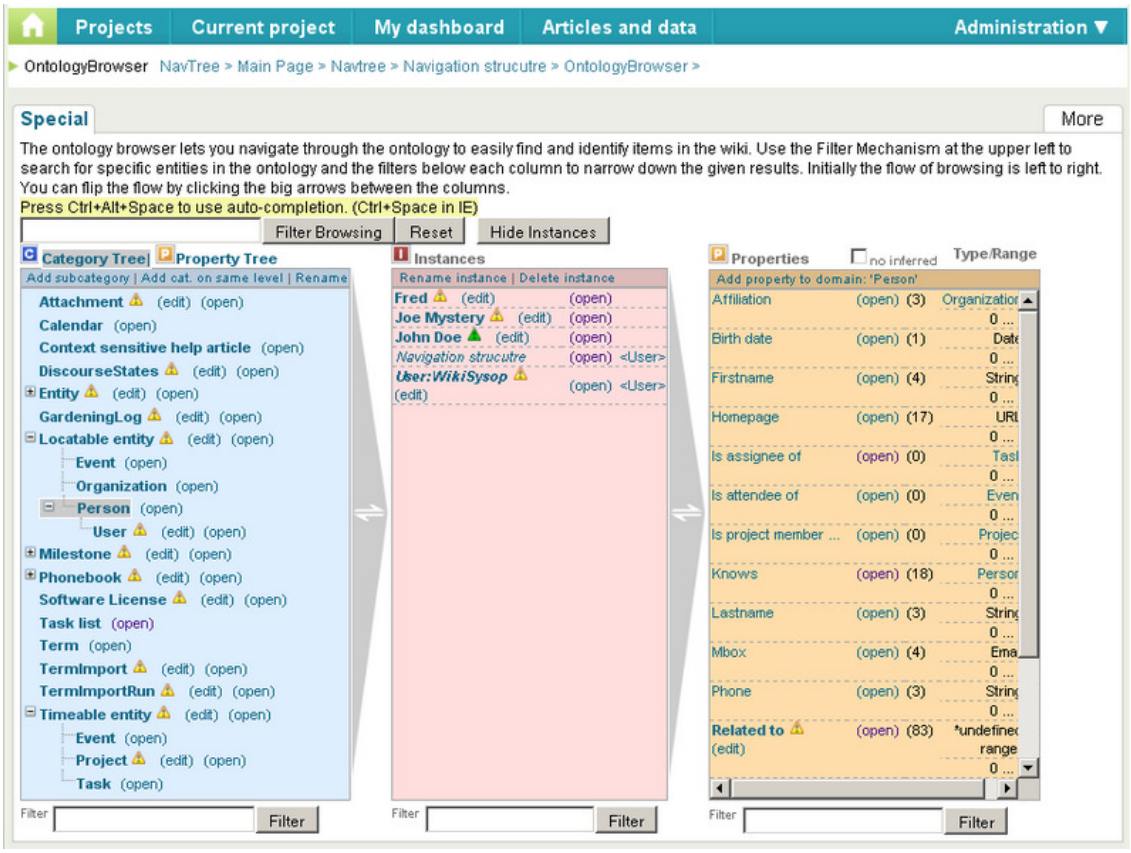


Figura 3.5 Halo. Navegação

Funcionalidades de autoria:

- Auto-completamento, sugerindo instâncias, propriedades, categorias ou *templates* ao escrever para dar possíveis correspondências (*matches*)
- Barra de ferramentas semânticas, permitindo rapidamente inspecionar, criar e alterar as anotações semânticas de um artigo wiki

- Modo Avançado de anotação, para anotar conteúdos semânticos de uma maneira WYSIWYG, sem ter que lidar com o texto fonte wiki.

Funcionalidades de recuperação:

- Interface de consulta gráfica, capacitando aos usuários para compor facilmente consultas e oferecendo uma visão prévia dos resultados em diferentes formatos de saída.

Como foi evidenciado este *framework* possibilita funcionalidades com foco na navegação, adição de informação semântica, e facilidades para compor consultas e apresentar os resultados em determinado formato. Mesmo assim, ele não está orientado ao desenho de estruturas navegacionais a um nível conceitual para tarefas específicas.

3.2.2. XIMDEX

Ximdex é um sistema de gerenciamento de conteúdo (CMS: *Content Management System*) semântico, de código aberto, que simplifica o processo de gestão da informação através do uso de tecnologias semânticas.

Ximdex específica estrutura para o site web permitindo visualmente o gerenciamento de documentos, assim como de todos os elementos constituintes (vídeos, imagens, aplicativos, áudio...). A tecnologia de *Ximdex* abrange uma ampla gama de funcionalidades complementarias para gerenciamento de documentos e conteúdo da Web, permitindo a edição visual de qualquer conteúdo ou aplicação Web, da sua estrutura e meta tags na forma de anotações semânticas.

O uso de tecnologias da Web Semântica e computação na nuvem em *Ximdex* aumenta a reutilização de informações e permite uma independência real dos canais de renderização final (web, dispositivos móveis), facilitando a adaptação automática de informações a qualquer formato atual ou futuro.

Os módulos de Ximdex estendem a sua funcionalidade. Alguns dos módulos disponíveis são: *Ximedit & Xowl, Ximfind, Ximtax, Linked Open Data* (LOD)

O módulo XIMEDIT, integrado na base, é um editor visual para XML que mostra em tempo real e aparência real, como o documento será exibido uma vez publicado no site e, mais importante, garante que o documento esteja devidamente estruturado de acordo com o esquema definido.

Ximedit em conjunto com o módulo *Xowl*, aplica tecnologias semânticas para enriquecer o conteúdo automaticamente, proporcionando imagens, textos e links relacionados com o que foi escrito.

Ximfind é um módulo que fornece indexação de conteúdo e funcionalidades de busca. *Ximfind* pode gerenciar diferentes documentos, mesmo sendo eles não estruturados (HTML, PDF, Word, etc.), estruturados (XML) ou semânticos (RDF).

Ximfind combina diferentes tecnologias de recuperação de informação e indexação (XML nativo a partir de *crawlers*, processamento de meta-informação, microformatos, etc.), proporcionando um acesso centralizado. Além disso, suas raízes na Web Semântica permitem a ligação dos resultados da pesquisa a ontologias e taxonomias.

Ximtax é um módulo que permite a criação e edição da informação semântica associada tanto a recursos internos, gerenciados por *Ximdex* quanto recursos externos.

Por meio de sua interface gráfica, o usuário pode adicionar e editar livremente meta-informação para o conteúdo gerenciado. Toda esta informação semântica é recolhida internamente com base em uma ontologia permitindo a interoperabilidade com outros elementos de sistemas padrões, através da tradução entre os diferentes esquemas de representação.

Linked Open Data (LOD) em combinação com o módulo Xlyre para o gerenciamento de *Linked Open Data*, os dados podem ser obtidos, em seguida, transformados, interligados, anotados e melhorados por meio de tecnologias semânticas e publicados em outros *datasets* ou em diferentes formatos.

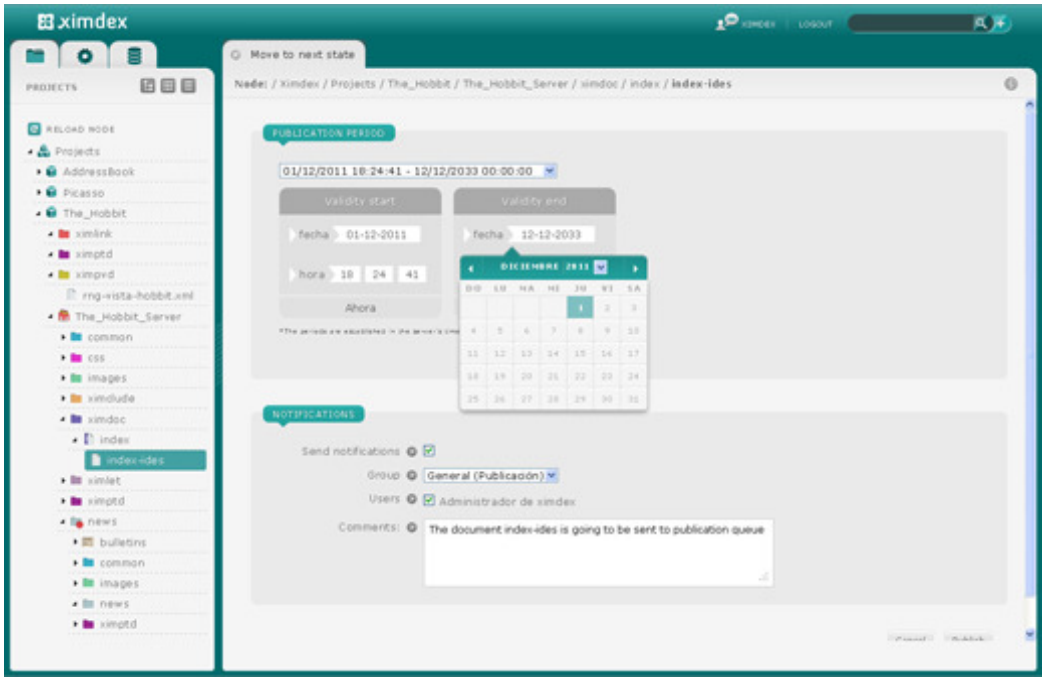


Figura 3.6 Gerenciamento fácil de grande quantidade de documentos e publicação na nuvem

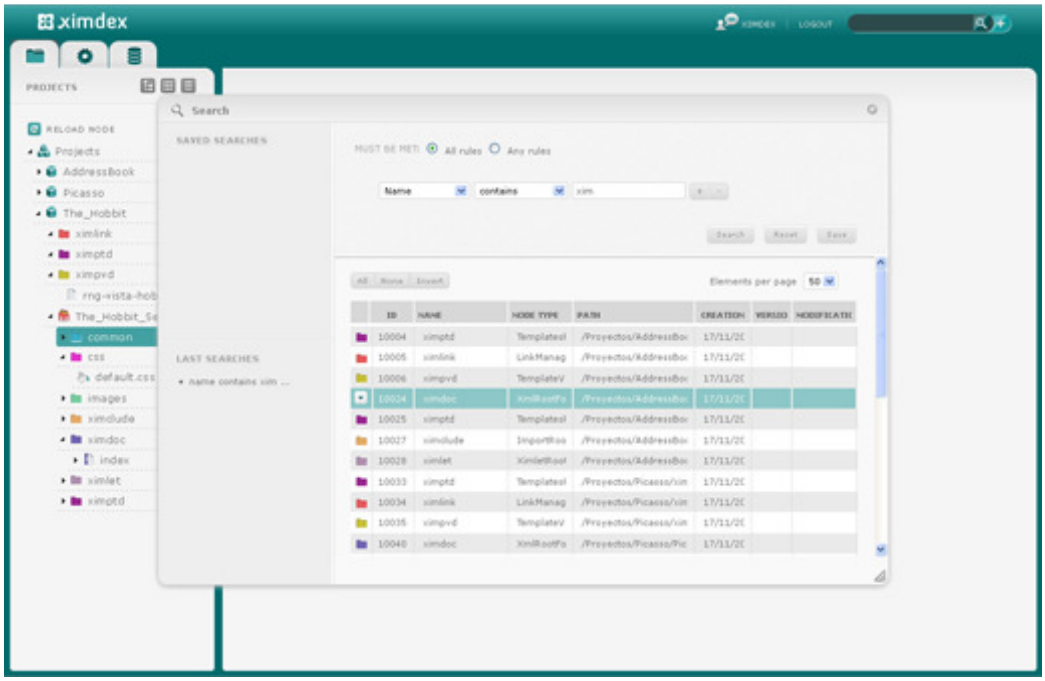


Figura 3.7 Motor de busca semântica (*search engine*) para encontrar conteúdos

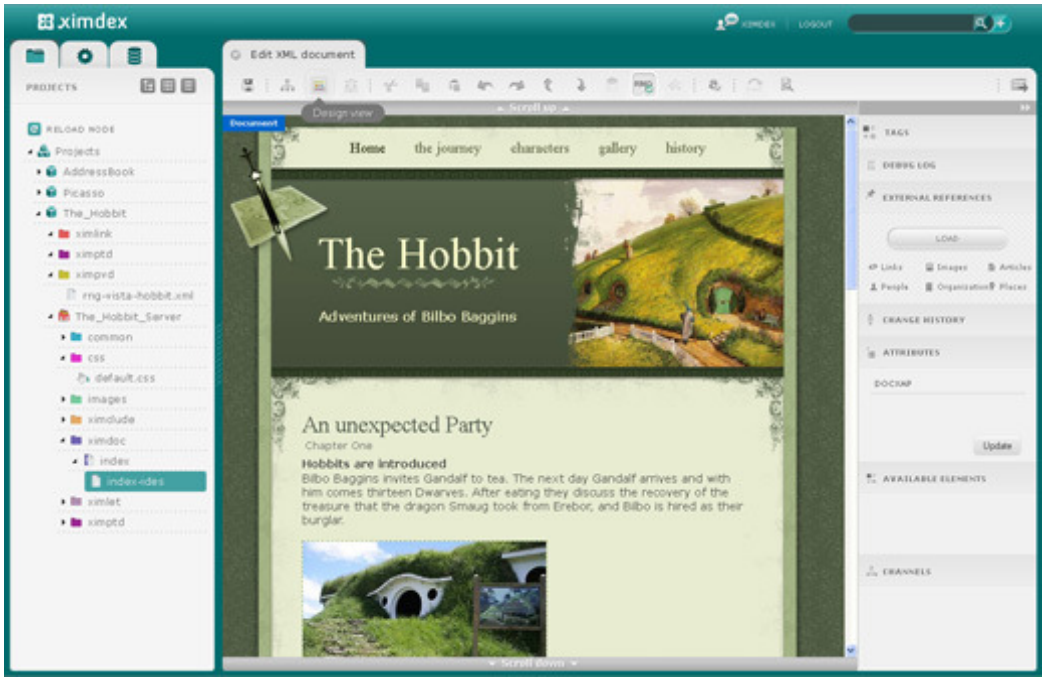


Figura 3.8 Editor visual com modelo de interação WYSIWYG para modificar os documentos XML semânticos para a web, mobile

O *Ximdex* oferece um ambiente de desenvolvimento com foco na apresentação e formatação de documentos com um estilo de manipulação WYSIWYG. Possibilita a anotação de informação semântica para enriquecer o conteúdo apresentado, mas também não comporta as funções de navegação das aplicações hipermídia.

3.2.3. RDFaCE

RDFaCE é uma implementação do conceito *What You See Is What You Mean* (WYSIWYM) para a manipulação direta de conteúdo semanticamente estruturado em modalidades convencionais. *RDFaCE* utiliza a geração de formulários on-the-fly baseado no vocabulário Schema.org para a incorporação de metadados em documentos web. Além disso, emprega os serviços externos de Processamento de Linguagem Natual (*Natural Language Processing* - NLP) para permitir a anotação automática de entidades e sugerir URIs para entidades.

RDFaCE oferece uma criação amigável de conteúdo semântico de forma manual e semiautomática. A abordagem *RDFaCE* combina a autoria de texto WYSIWYG com a criação de anotações semânticas. Fornece quatro visões diferentes para os autores de conteúdo: a visão clássica WYSIWYG, a visão WYSIWYM tornando visível as anotações semânticas, a visão de fatos e a visão

respectiva HTML / RDFa do código fonte. As visões são sincronizadas para que as alterações feitas em uma delas se atualizem automaticamente nas outras.

O objetivo de *RDFaCE* é integrar a anotação semântica diretamente na criação de conteúdo e fazer a anotação tão fácil e não-intrusiva como seja possível. Isto é conseguido através da integração das visões do código fonte e WYSIWYG clássicos com visões facilitando a anotação semântica. Concebe-se o conceito de visão WYSIWYM, que estende a visão WYSIWYG com anotações semânticas destacadas. Além disso, uma visão dos fatos ajuda os autores e engenheiros de conteúdos e conhecimentos a analisar de forma rápida e possivelmente revisar as anotações semânticas. A lógica por trás do conceito WYSIWYM é que tem que proporcionar um ambiente para o usuário, com o que ele está suficientemente familiarizado, mas ao mesmo tempo permite-lhe entender, acessar e trabalhar com anotações semânticas. Assim, WYSIWYM é uma extensão do conceito WYSIWYG, onde os usuários podem observar e gerenciar as anotações semânticas no contexto familiar desse ambiente.

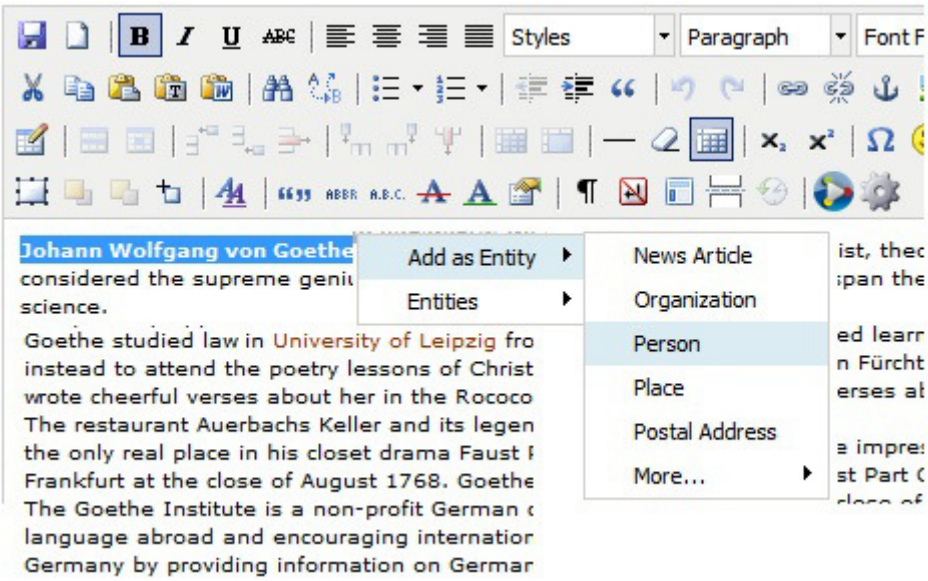


Figura 3.9 Seleção de uma parte do texto para anotar como entidade

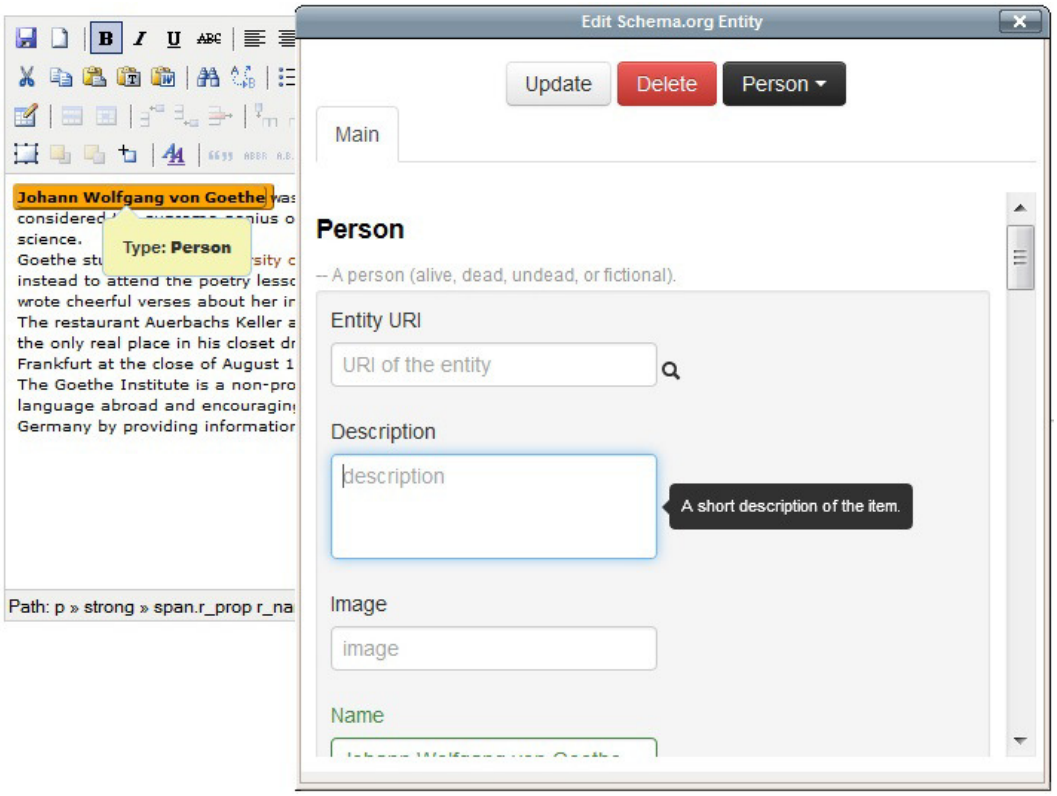


Figura 3.10 Especificação dos valores das propriedades de uma entidade selecionada

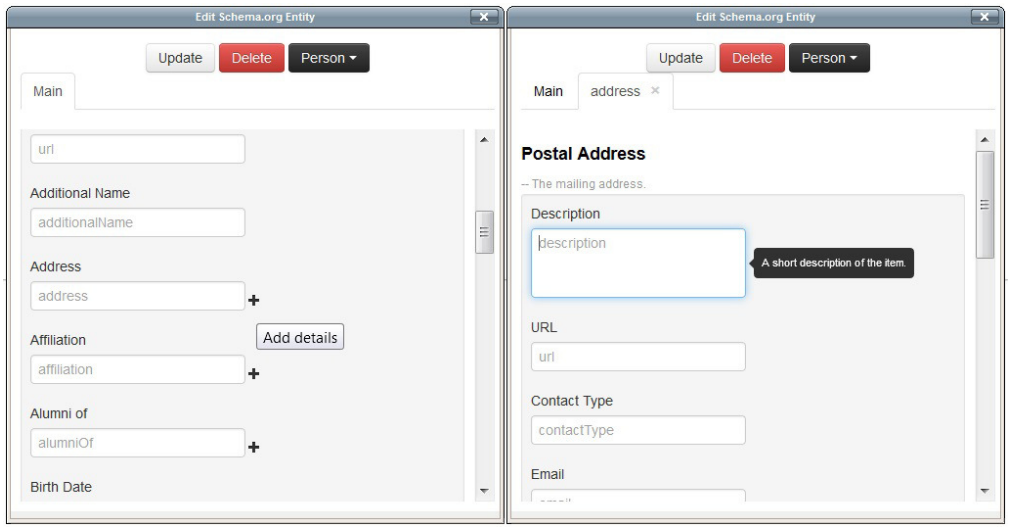


Figura 3.11 Especificação das propriedades de uma entidade associada por meio de uma *object property*

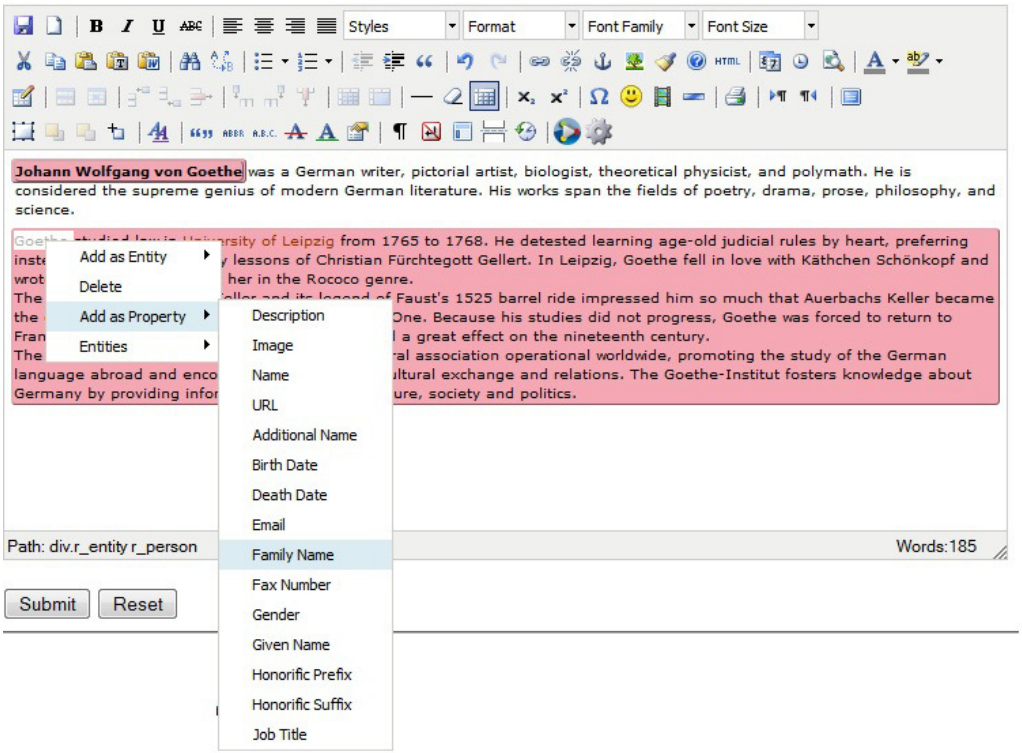


Figura 3.12 Um parágrafo pode ser especificado como entidade, assim no menu contextual só aparecem as propriedades do tipo da entidade

RDFaCE é uma ferramenta com um modelo de interação WYSWYG e WYSIWYM mas apenas para dar formato a documentos e fazer anotações semânticas.

3.2.4. Kimono

Kimono (Kimono, 2014) foi criado para permitir aos usuários transformar instantaneamente qualquer site web em uma API para informação estruturada. Esta ferramenta permite extrair dados estruturados de uma aplicação seguindo uma estratégia de programação por exemplo com um estilo WYSIWYM.

Kimono permite obter como saída dados estruturados em formato JSON ou CSV. A API de *Kimono* e os dados são hospedados na nuvem e pode ser executada no cronograma que se especificar, sem necessidade de escrever código. O extrator inteligente do *Kimono* reconhece padrões no conteúdo web, permitindo obter os dados visualmente.

3.2.5. X3S

X3S (Stegemann, Hussein, Gaulke, & Ziegler, 2012) é uma técnica e formato para especificar "*widgets* semânticos" que integram consulta e filtragem de dados semânticos com a definição de seu estilo de layout e apresentação. X3S fornece um formato aberto baseado em padrões para folhas de estilo semânticas reutilizáveis e pode ser usado com fontes de dados RDF arbitrárias. Além disso, um editor foi desenvolvido que permite que não-especialistas explorem intuitivamente os dados semânticos visualmente e definam *templates* personalizados em um estilo de manipulação direto.

X3S possibilita selecionar propriedades arbitrárias de um recurso especificado, em vez de receber e exibir todos os campos disponíveis. Além disso, a técnica permite a filtragem dos dados de uma maneira flexível e proporciona meios para aninhar várias propriedades. Isto permite representar visualmente recursos como valores de propriedades de outros recursos.

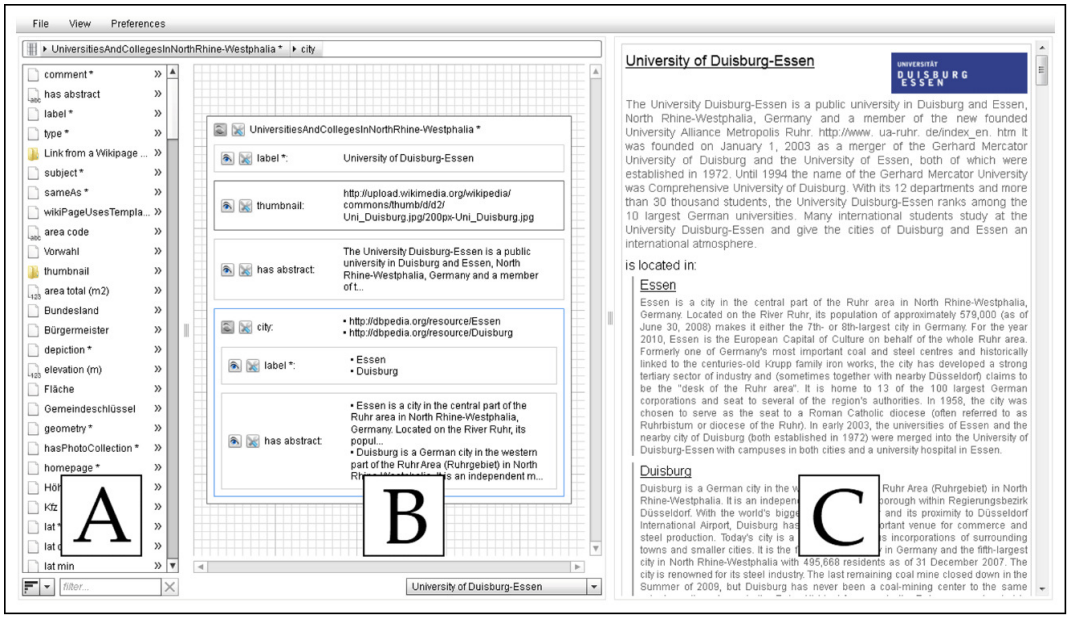


Figura 3.13 Editor ao criar uma folha de estilo semântico, trabalhando com dados de DBpedia

A abordagem *X3S* dá suporte apenas aos aspectos de interface, não sendo possível derivar um modelo de navegação generalizado como proposto no SHDM.

Todas estas abordagens têm seus benefícios nos respectivos casos de uso e tem aspectos em comum com nossa proposta. Por outro lado, nenhuma delas permite a síntese de modelos de navegação e de interface mais abstratos, que

permitem explorar o uso de modelos conforme postulado em metodologias como SHDM.

4
Assistente. Concepção e implementação.

Hoje em dia as aplicações são construídas a partir de dados publicados por outros, eventualmente adicionando-se dados específicos para o problema em questão. Portanto, é comum que se reaproveitem dados já publicados e se construa uma visão navegacional sobre esses dados estendidos.

A presente proposta define um método baseado em exemplos e manipulação direta, e uma ferramenta associada, que permite a usuários o projeto do modelo de navegação de aplicações hipermídia a partir de um modelo de domínio previamente definido. Estas aplicações são implementadas em um ambiente dirigido por modelos, que são sintetizados através do uso da ferramenta. No caso, o ambiente dirigido por modelos é o Synth (Bomfim, 2011) (Barbosa do Nascimento, 2013) que dá suporte à construção de aplicações projetadas segundo o método SHDM.

A modelagem por exemplo procura inferir um modelo para o artefato sendo projetado a partir de exemplos, que são entendidos como instâncias fornecidas pelo usuário. Assim, há uma parte que é a abordagem conceitual e outra que é a operacional. A parte conceitual é aquela que, dada uma instância, infere o modelo do qual ela é uma instância. Este modelo por sua vez é um caso particular de um metamodelo. Desta forma, têm-se o metamodelo, uma instância deste que é o modelo e uma instância deste que é o dado concreto (o exemplo dado pelo usuário).

Exemplo:

Metamodelo: classe de navegação

Modelo: classe Pessoa

Instância de Pessoa: o indivíduo João

A parte operacional diz respeito ao estilo de interação WYSIWYG e da manipulação direta utilizada para informar o exemplo e dar feedback para o usuário da interpretação feita pelo sistema daquilo que foi informado.

A ferramenta será referida como assistente ou *wizard* (anglicismo que representa um padrão de projeto de software amplamente utilizado em interfaces gráficas do usuário). Esta aplicação oferece um meio simples de realizar tarefas mais complexas, através de um esquema que decompõe a tarefa em um passo-

a-passo, i.e. um utilitário interativo que ajuda e que orienta o usuário em cada etapa de uma determinada tarefa. Além disso, o nosso assistente inclui a lógica referente ao metamodelo de navegação. Ele processa as ontologias do domínio de interesse, inclui a organização do conhecimento, modelos, metamodelos e a geração do diálogo que o projetista vai precisar para poder sintetizar os modelos. Ao final, o assistente instancia as estruturas SHDM.

4.1. Manipulação direta.

O núcleo do presente trabalho está na autoria WYSIWYG, ou seja, por prototipação rápida com manipulação direta. Trata-se de aproximar o mais possível a intenção do usuário à concretização por meio da ferramenta. Em uma solução WYSIWYG permite-se que uma tela, enquanto manipulada, tenha uma aparência similar à sua realização final.

Por meio da manipulação direta, oferece se um estilo de interação que envolve representação contínua do objeto de interesse, com feedback imediato. Ela deve também ser reversível e utilizável através de ações de incremento e retorno.

A manipulação direta se materializou através da intenção de permitir a manipulação e especificação de protótipos usando recursos próximos (ou análogos) ao resultado pretendido. Cada representação usada é a um protótipo de uma tela da aplicação final, o seja, é uma abstração da interface que vai se instanciar posteriormente na aplicação real.

Por exemplo, consideremos um índice, que é uma das primitivas do modelo navegacional de SHDM que representa uma lista de ponteiros para outros elementos (por exemplo, um menu ou uma lista de links). Observemos as seguintes telas (Figura 4.1 e Figura 4.2):

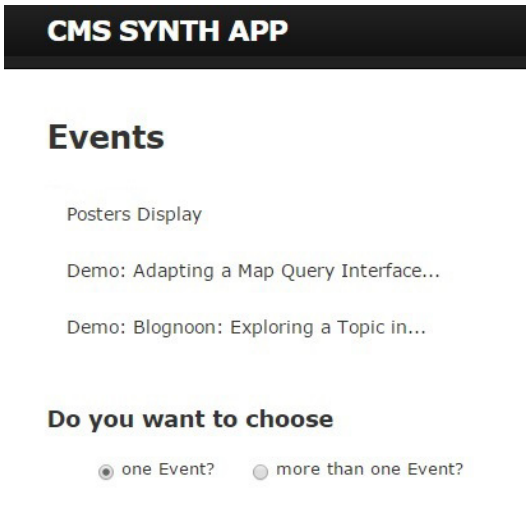


Figura 4.1 Lista de eventos de seleção simples

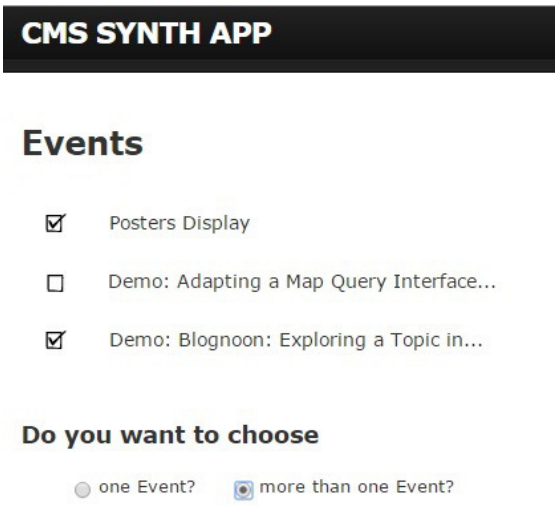


Figura 4.2 Lista de eventos de seleção múltipla

No caso, o índice é de eventos e a lista sugere que clicando-se em um dos elementos (um evento) o nó navegacional[4] correspondente ao evento selecionado será exibido para permitir a visualização dos seus detalhes. A seleção de um evento também poderia permitir o acesso a uma outra classe de navegação, por exemplo, uma lista dos apresentadores do evento.

Para a especificação de se a lista de eventos vai permitir seleção simples (Figura 4.1) ou múltipla (Figura 4.2) é exibida uma representação que varia de um caso a outro, com o uso o não de *checkboxes*, padrão conhecido para ilustrar esta situação. Esta mudança acontece de maneira interativa. Note-se que o uso de *checkboxes* na interface do *wizard* é meramente ilustrativo, e não implica que exatamente este mesmo tipo de *widget* de interface será de fato utilizado na aplicação final. O uso deste *widget* justifica-se apenas pelo fato da maioria das pessoas estarem familiarizadas com a sua semântica (selecionar uma ou mais opções de uma lista).

Outro exemplo da autoria WYSIWYG é o da Figura 4.3 e Figura 4.4. Nelas pode se ver que na escolha dos atributos desejados na primeira seção, muda-se a tabela com os exemplos da segunda seção. São exibidas as novas colunas e estas aparecem na ordem e na hora da escolha.

[4] Nó navegacional - instância do domínio (recurso RDF) acessado a partir de um contexto navegacional. As instâncias das classes navegacionais são denominadas nós

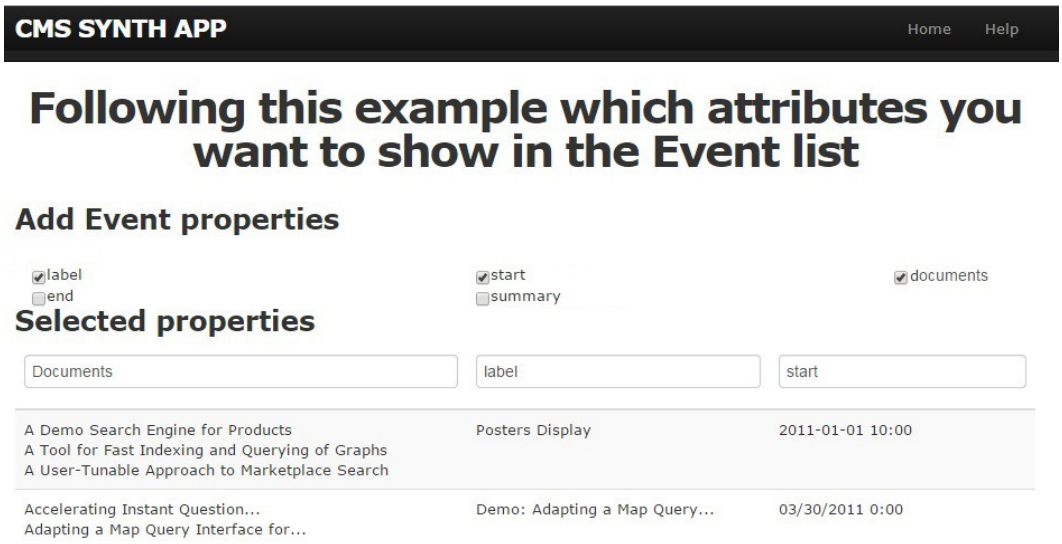


Figura 4.3 Seleção de propriedades diretas dos eventos

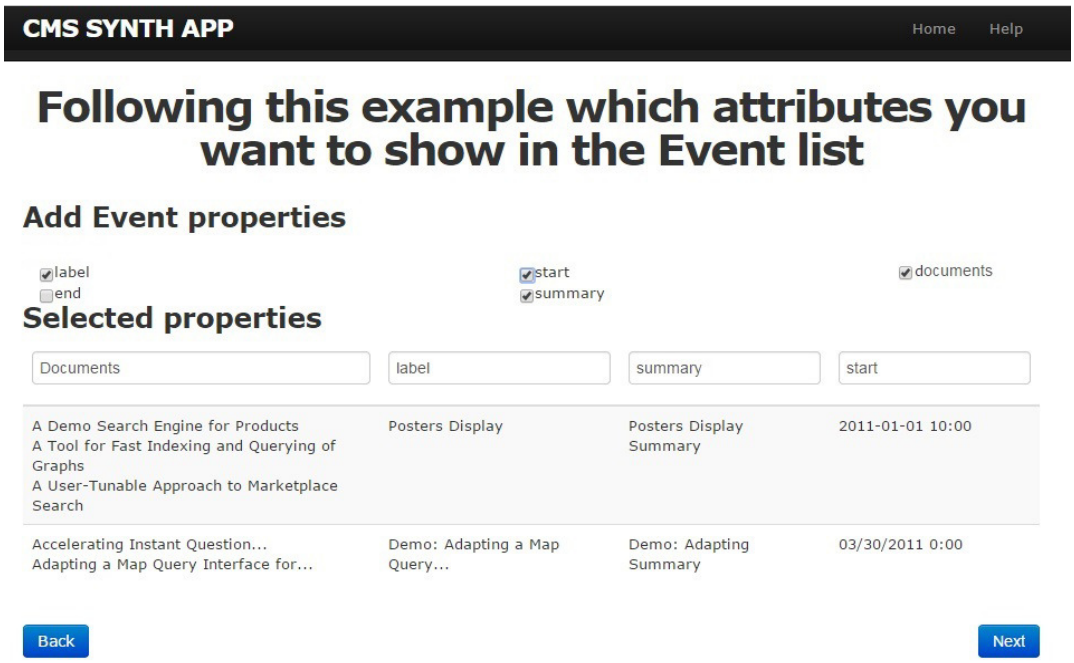


Figura 4.4 Seleção de propriedades diretas do evento. Mais atributos

Resumindo, o propósito foi desenhar ou fazer um esboço de telas nas que se usam *widgets* abstratos[5]. Esta representação não é definitiva, pois abstrai a forma, mas com uma aparência sugestiva e similar à pretendida. O mapeamento

[5] Widget abstrato - define os elementos de interface que estão previstos em um nível abstrato, sem focar em que componente concreto irá ser utilizado. Por exemplo, 'um elemento para selecionar um valor'. Esses elementos são independentes de implementação ou plataforma.

é em abstrato, mas tem um processamento posterior para gerar a parte da interface real a ser utilizada.

4.2. Programação por exemplo

Outra forma de tornar a tarefa mais intuitiva para o usuário é a estratégia de Programação através de Exemplo (PBE, por suas siglas em inglês de *Programming By Example*), também conhecida como Programação Por Demonstração. PBE é uma técnica de desenvolvimento dirigida ao usuário final para ensinar ao computador um novo comportamento, demonstrando ações através de exemplos concretos. O sistema registra as ações do usuário e infere um procedimento generalizado que pode ser usado em novos exemplos.

Em um sistema de PBE puro, é gravado um processo que repete exatamente o que o usuário faz. Ocasionalmente, a execução de tais programas invariantes é útil, mas na maior parte das vezes, fazer exatamente a mesma coisa não é o que o usuário tem em mente. O usuário gostaria de parametrizar ou generalizar o programa, para que ele possa alterar os dados que o programa usa. Analogamente, na abordagem proposta, ele deverá ser capaz de indicar ao sistema que alguns objetos de dados usados no exemplo servem apenas como variáveis, e deve ser substituído por parâmetros, o mesmo valendo para opções de navegação.

A estratégia de programação por exemplo se materializou na proposta construindo-se o modelo através dos exemplos utilizando as interfaces com dados concretos. O usuário vai compondo uma interface, e, por trás, o sistema vai construindo o modelo. Toda vez que o usuário faz alguma escolha no assistente e sempre que proceder se oferecem instâncias exemplo das classes conceituais. No caso construímos os exemplos a partir das instancias recuperadas da ontologia de domínio original informada no inicio do processo. Através das figuras (Figura 4.5 e Figura 4.6) se mostram dois passos consecutivos do assistente que ilustram isto. Quando o usuário escolhe a opção de mostrar o detalhe do evento (Figura 4.5) o feedback imediato é uma tela com exemplos de cada atributo de um evento (Figura 4.6).

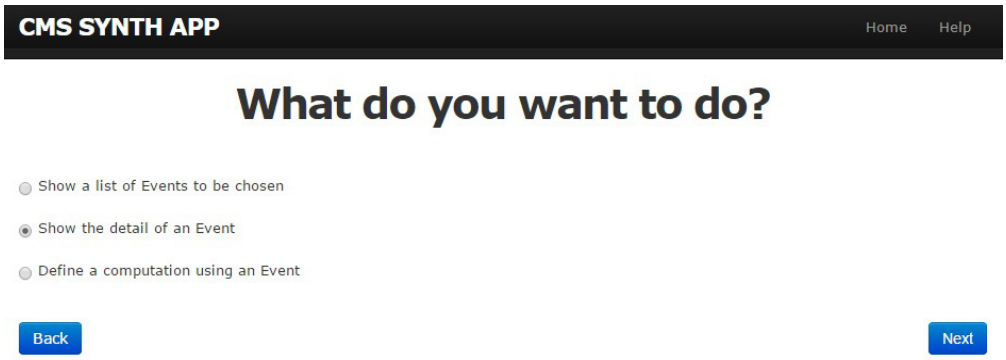


Figura 4.5 Seleção de uma ação

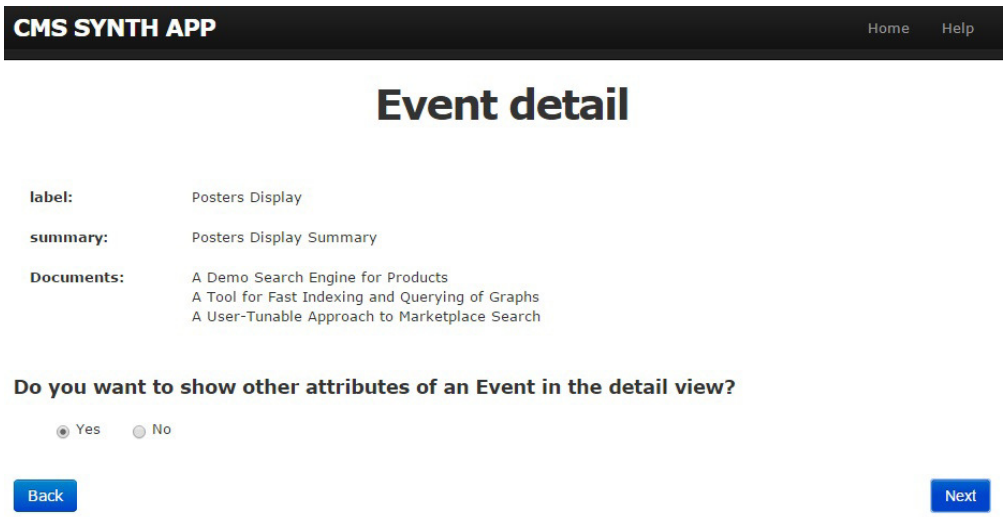


Figura 4.6 Detalhe de um evento

Na Figura 4.7 se pode ver como ao especificar novas propriedades do evento para a visão de detalhe (no caso, o atributo "*start*" que não aparecia na tela da Figura 4.6) a aplicação apresenta um exemplo para este atributo, e assim se vão mostrar ou deletar exemplos segundo sejam as escolhas das propriedades.

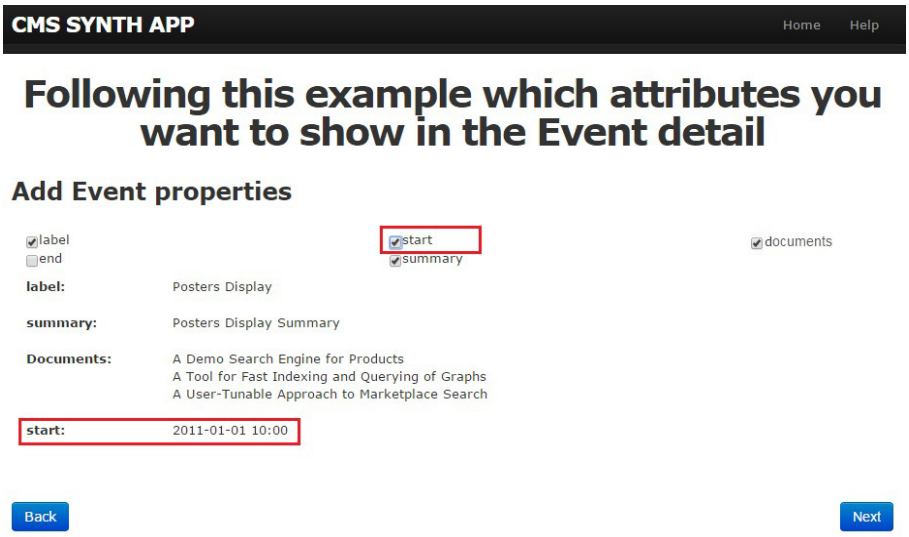


Figura 4.7 Seleção de propriedades diretas do evento

Outro exemplo da abordagem PBE é o da Figura 4.8 e Figura 4.9, que se refere ao passo onde a aplicação sugere os caminhos possíveis para relacionar um evento com uma pessoa. A partir de um evento, o projetista poderia querer mostrar, além dos atributos diretos, quais são seus apresentadores, que são do tipo de uma outra classe conceitual: *Person*. Assim a aplicação oferece os caminhos possíveis (i.e, composição de relações) entre *Event* até esta classe, que no caso são dois: "*Event > Document > Person*" e "*Event > Person*". Uma vez que o usuário tiver escolhido um caminho, se apresentam exemplos das instâncias das classes e das propriedades que as relacionam.

A Figura 4.8 indica que no caminho "*Event > Document > Person*" têm-se as instâncias exemplo *Event1* e *Event2* para *Event* e *Albert* e *Schwabe* para *Person*; como exemplos de propriedades que relacionam as classes Event e Document têm-se *hasOpeningDoc* e *hasClosingDoc;* e como exemplos das propriedades entre Document e Person estão *presenter* e *advisor*. O evento *Event1* tem um documento de abertura (*hasOpeningDoc*) com apresentador *Albert* e *Event2* tem um documento de fechamento (*hasClosingDoc*) com orientador *Schwabe*.

Figura 4.8 Seleção do primeiro caminho de evento a pessoa

A Figura 4.9 mostra o exemplo análogo ao da Figura 4.8 mas para o caminho "*Event > Person*". Aqui se tem as instâncias exemplo *Event1* e *Event2* para *Event* e *Tim Berners Lee* para *Person*; um exemplo de propriedade que relaciona diretamente as classes Event e Person é *organizer*. Assim, têm-se que os eventos *Event1* e *Event2* tem o organizador *Tim Berners Lee*.

Figura 4.9 Seleção do segundo caminho de evento a pessoa

Em resumo, o trabalho visa desenvolver um ambiente que facilite o papel do desenvolvedor, de tal forma que seja mais fácil para ele começar a entender o metamodelo que representa as funções do método SHDM. A intenção é ajudar ao projetista a visualizar como vai ser a sequência de navegação e para isso, o mais intuitivo, é permitir que ele especifique através de um exemplo conhecido e daí oferecer uma generalização. Observe-se que em geral não se pode esconder a formalização totalmente, mas pode-se chegar a um ponto aonde seja mais natural falar dela.

4.3. Especificação de Requisitos Iniciais

Os requisitos iniciais para o desenvolvimento da aplicação são resumidos em:

1. Oferecer uma ferramenta com características de assistente que permite a criação de aplicações através de exemplos, utilizando interfaces com dados concretos.
    a. Permitir a interação para o projeto da aplicação e/ou representação de dados semânticos de maneira intuitiva mediante uma interface de manipulação direta.
    b. Oferecer um mecanismo intuitivo para instrumentar a navegação
    c. Permitir a formulação das especificações seguindo a estratégia de "especificação por exemplos ou demonstração".

4.4. Casos de Uso

O diagrama de casos de uso apresentado na Figura 4.10 ilustra as possíveis funcionalidades do sistema e como estas se relacionam entre si. O ator é o projetista que vai poder criar uma aplicação. Para isto foi definido um caso de uso geral, "Criar aplicação". Este tem relações de dependência de extensão com "Especificar lista de elementos", "Especificar visão de detalhe de um elemento" e "Definir uma operação para um elemento" para indicar comportamentos do sistema que são opcionais dependendo das escolhas do projetista. Isto significa que o usuário pode definir, na aplicação que está criando, telas que apresentem uma lista de elementos, telas para mostrar o detalhe de um elemento e operações para um elemento, indistintamente. Tanto em uma lista de recursos quanto em uma tela de detalhe o projetista deve especificar os atributos a mostrar de um recurso (relações de inclusão para o caso de uso "Especificar atributos"). De maneira opcional, um atributo pode ser uma âncora para uma outra visão navegacional da aplicação (caso de uso "Definir âncoras"). A especificação de atributos pode ser de três tipos: atributos diretos de uma entidade (caso de uso "Especificar atributos diretos"), atributos resultantes da relação do recurso com uma outra entidade ("Especificar atributos relacionados")

e atributos que requerem algum tipo de processamento para serem mostrados (“Especificar atributos computados”). De igual forma, a definição de atributos pode incluir âncoras para uma lista de elementos ou para o detalhe de um elemento (relações de dependência que estendem “Definir âncoras”). Este diagrama de casos de uso é isomorfo ao metamodelo de navegação do SHDM.

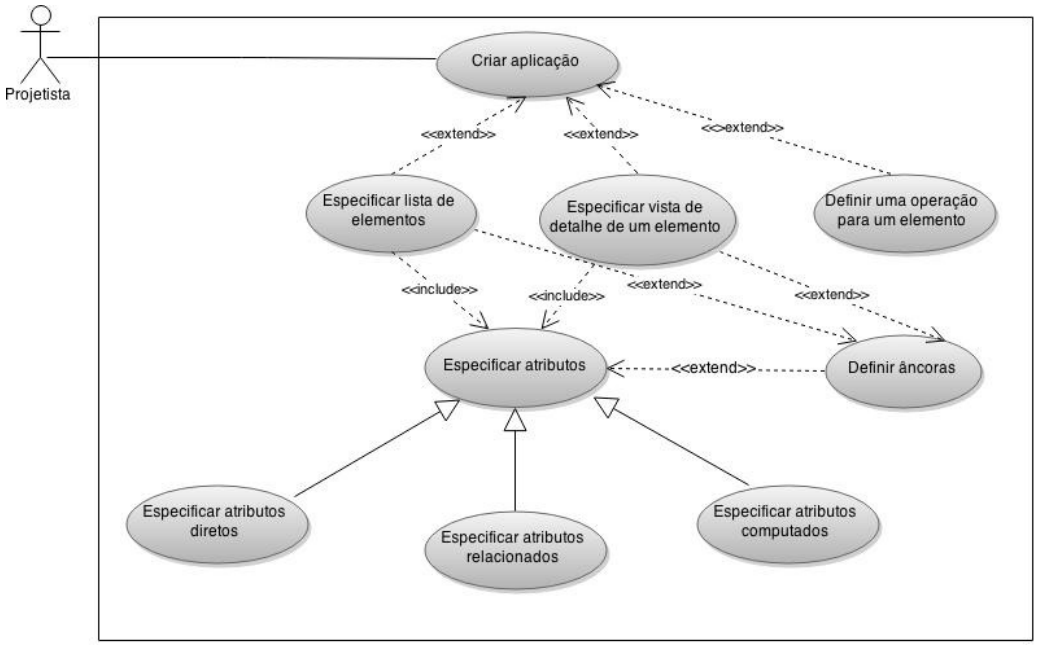


Figura 4.10 Diagrama de casos de uso

O diagrama de sequência apresentado na Figura 4.11, ilustra a interação do usuário com o sistema durante o desenvolvimento de uma aplicação, assim como a colaboração entre os diferentes módulos de implementação. As interações entre o usuário e o sistema sempre ocorrem através da interface visual, com características de assistente, disponibilizada pelo módulo de interface visual (MIV).

A seguir se descreve passo-a-passo a sequência de colaboração ilustrada na Figura 4.11:

Passo 1: criar_aplicação_para(ontologia) – Inicialmente, o projetista indica a criação de uma aplicação e escolhe uma ontologia de domínio já previamente definida no assistente (MIV). O MIV invoca a operação *generate_wizard(ontology)* do módulo processador de ontologias (MPO) por meio do envio de uma requisição do protocolo HTTP. O MPO processa a ontologia recebida e retorna um modelo (JSON) que representa a sequência de passos do assistente, obtida de acordo com as classes e recursos dessa ontologia.

Passo 2...n: definir_lista_de_elementos, definir_detalhe_elemento, ..., definir_atributos – O usuário continua interagindo com o assistente do MIV para especificar as características da aplicação que está desenvolvendo. Toda vez que ele faz uma escolha ou especificação a aplicação mostra dados concretos como exemplo, que foram obtidos da ontologia no Passo 1, ilustrando como fica a aplicação resultante das escolhas feitas naquele passo.

Passo n+1: instanciar_aplicação – No último passo do assistente o usuário aciona a opção do MIV para instanciar a aplicação descrita. O MIV invoca a operação *create_application* do módulo gerador de aplicações (MGA) com as escolhas feitas pelo usuário (*user_selections*) e o modelo da sequência de telas do assistente *(wizard_model)* como parâmetros, por meio do envio de uma requisição do protocolo HTTP. O MGA se comunica com o Synth através de uma requisição HTTP para criar a aplicação neste ambiente (*create_app(name)*). Em seguida, analisando a sequência de telas do assistente e com as escolhas do usuário, vai determinando em cada passo que elemento de modelo no SHDM se deve criar, através da API do Synth, chamando métodos como: *create_landmark* (name, position, type, att_label_expression, att_target_index), *create_context (name, title, query)*, *create_index (name, title, context)*, *create_computed_attribute_index (name, expression, position)*, entre outros. Finalmente, o usuário é informado que a aplicação foi criada no Synth.

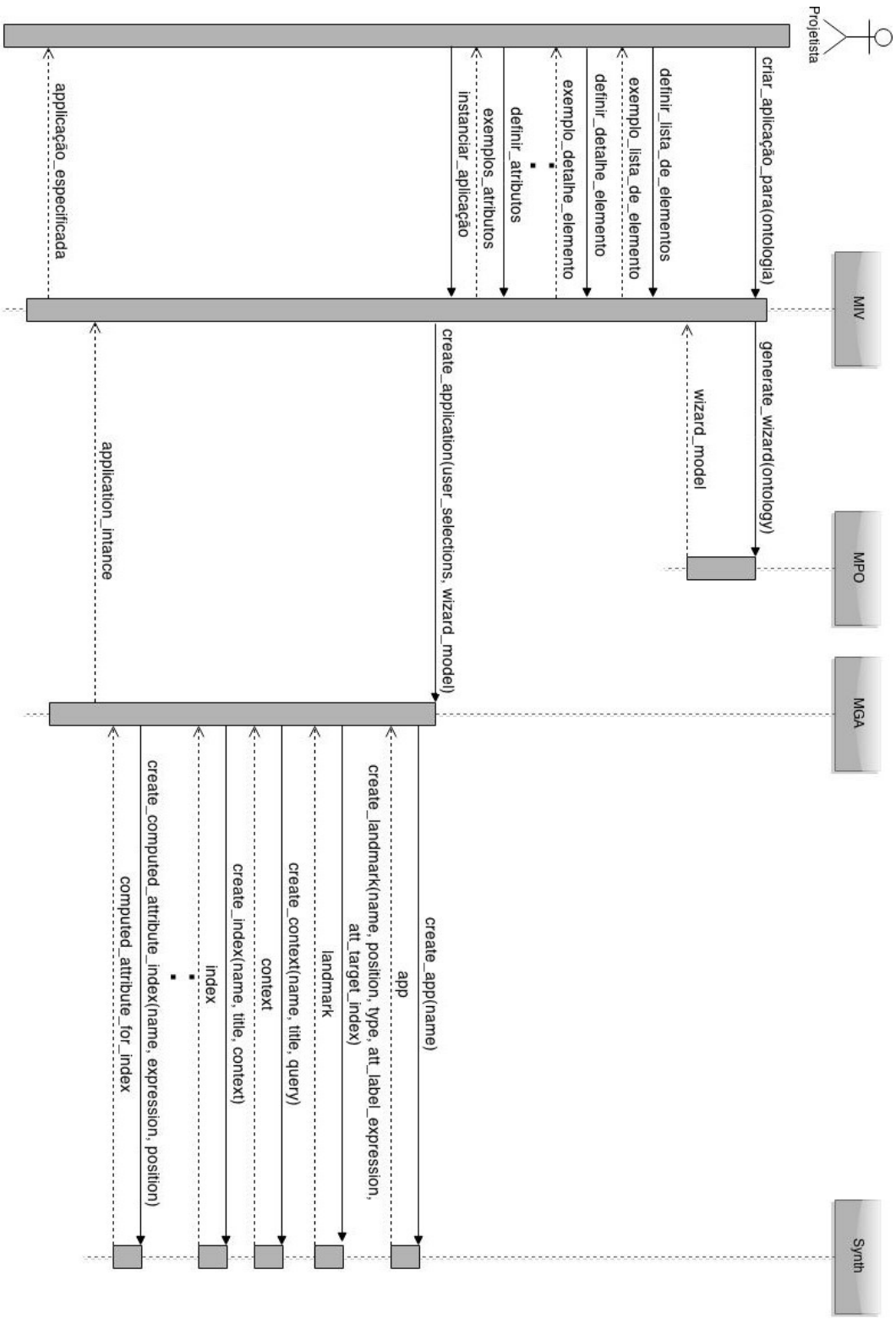


Figura 4.11 Diagrama de sequência

4.5. Arquitetura

No desenvolvimento do assistente foram definidos três módulos independentes e uma camada APIRest para o acesso às funções de criação das estruturas SHDM do Synth:

- Módulo processador de ontologias (MPO): recebe uma ontologia de domínio e gera um modelo que representa o diálogo ou a sequência de telas a renderizar do assistente. Esta sequência vai ser fixa e gerada a priori, ou seja, não depende da interação com o usuário, só da ontologia recebida.
- Módulo da interface visual (MIV): recebe como entrada o modelo da sequência de telas do MPO, gera a interface do assistente com as diferentes opções e obtém como saída as escolhas feitas pelo usuário.
- Módulo gerador de aplicações (MGA): recebe as especificações do usuário obtidas no MIV e a sequência de telas do MPO e gera as instâncias do modelo SHDM fazendo uso do ambiente Synth.

O Synth: oferece uma API para o acesso as funções de criação dos modelos SHDM e produz uma aplicação resultante da interpretação desses modelos.

Uma representação gráfica da arquitetura de software especificada para o assistente pode ser vista na Figura 4.12:

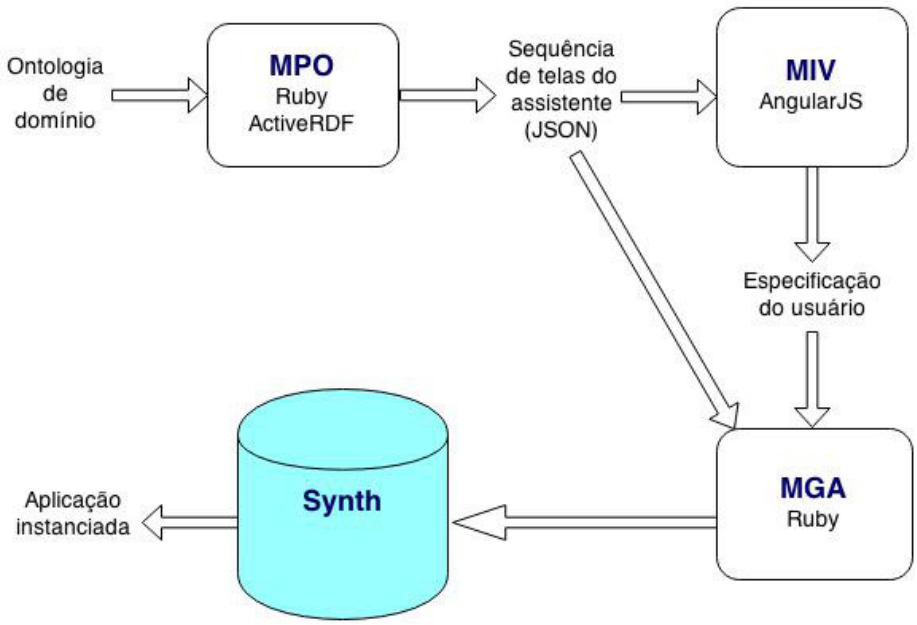


Figura 4.12 Arquitetura de implementação do assistente

4.6. Implementação

Os módulos desenvolvidos foram concebidos para gerar um diálogo, interagir com o usuário, obter a intenção deste, e gerar a aplicação desejada finalmente. Esse diálogo vai ser representado como uma máquina de estado, onde se definem estados e transições entre estes.

O MPO recebe a especificação da ontologia de domínio, definindo o esquema conceitual como é ilustrado em um exemplo simplificado na Figura 4.13. Além disso, se define na ontologia um conjunto de instâncias para as classes do domínio, como ilustra a Figura 4.14. Essas instâncias são recuperadas da ontologia para a construção dos exemplos a mostrar para o usuário. No escopo do nosso trabalho se assume que esta ontologia já é conhecida, pois terá sido definida em outros passos do processo de desenvolvimento.

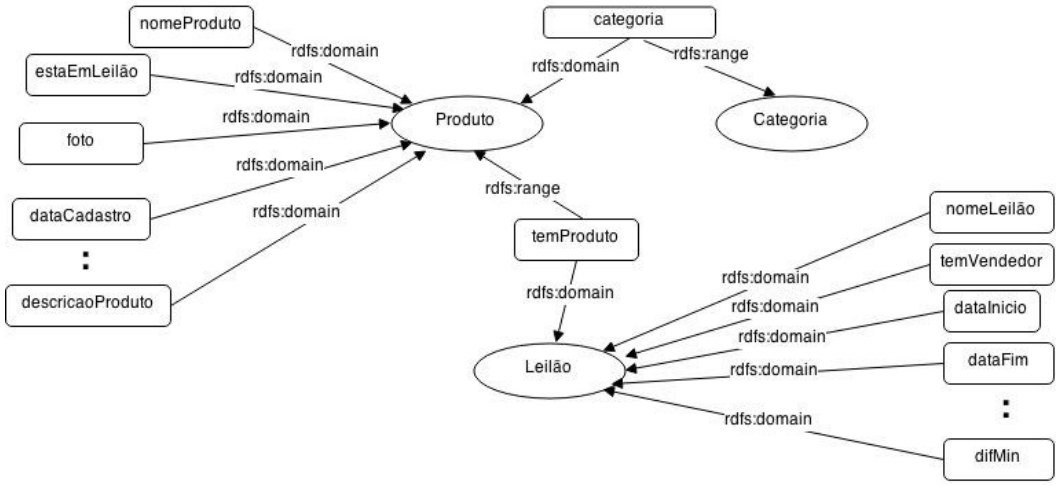


Figura 4.13 Grafo RDF do esquema de domínio

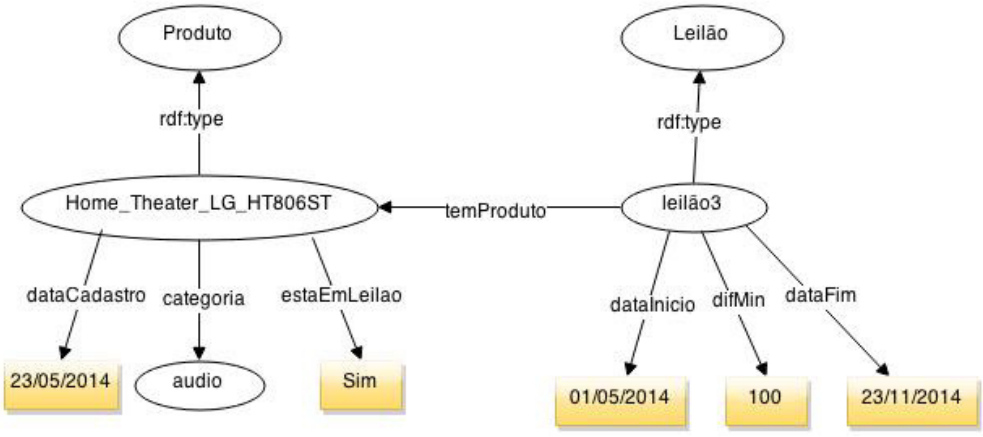


Figura 4.14 Grafo RDF para uma instância Produto e uma instância Leilão

Assim, a partir da ontologia especificada o MPO produz como saída o modelo que representa a sequência de telas do assistente. Este modelo constitui uma máquina de estados esboçada a título de exemplo, sem os dados reais, na Figura 4.15.

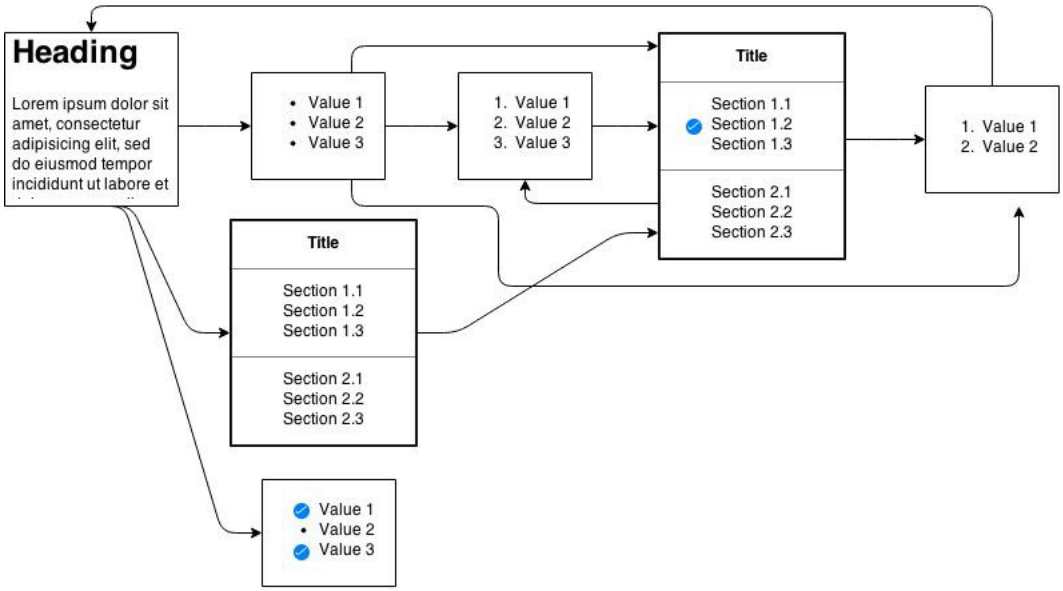


Figura 4.15 Esboço do fluxo entre telas

Um estado representa um passo no assistente e as transições indicam o próximo passo em função das escolhas do usuário no estado atual. Um estado é definido como ilustrado no Quadro 4.1

```
[{
"id": identificador do passo,
"title": título da janela, informativo para o usuário,
"type": tipo de janela,
"message": informação a mostrar ao usuário,
"messageOptions": mensagem ou pergunta nos casos com várias opções,
"mainClass": classe da ontologia que está sendo usada
"options": [opções possíveis para a seleção do usuário
      {
      "key": chave para identificar a opção,
      "text": texto da opção a mostrar para o usuário,
      "next": identificador do passo seguinte no assistente
                (transição para o próximo estado),
      "child": ("New branch", "End branch") define a existência de uma
            conjunção de possibilidades,
      "todo": ver definição embaixo
      },
      ...
```

```
      { ... }
      ],
      "details": [[exemplos a mostrar para cada opção] ...[..]],
      "todo": [funções a executar para a instanciação das estruturas
                  SHDM
      {
            "function_name": nome da função,
            "params": [parâmetros da função
                  {
                  "param_type": tipo de parâmetro (constante; tipo
                        que representa um valor obtido na execução
                        anterior de uma função; tipo que representa
                        uma escolha do usuário),
                  "name": nome do parâmetro,
                  "value": valor do parâmetro
                  },
                  ...
                  {
                        ...
                  }
            ],
            "results": [estrutura para armazenar os valores
                  resultantes da execução de uma função
                  {
                  "name": nome para identificar o resultado,
                  "global_var": nome do resultado em um escopo
                        global para as funções
                  },
                  ...
                  { ... }
                  ]
      },
      ...
      { ... }
]
```

Quadro 4.1 Modelo que representa um passo do assistente

No MIV, foram definidas várias interfaces gráficas que representam janelas ou passos do assistente. Elas permitem compor uma sequência de telas a partir do reuso de *designs* recorrentes. Isto ajuda à modularização pois se agrupam funcionalidades comuns, evitando esforço de (re)implementação em passos similares. O campo `"type"` no modelo do Quadro 4.1 foi definido para especificar o tipo de controle visual a mostrar nesse passo.

Em um fluxo qualquer é possível que a partir de um estado ou passo se definam várias operações com um fluxo específico para cada uma. Em certos casos pode ser preciso (ou útil) voltar ao estado inicial para continuar por outro fluxo alternativo. Isto corresponde a uma arvore e-ou (*and-or tree*). O campo

`"child"` tem dois valores possíveis: `"New branch"` para indicar o começo de um novo fluxo e `"End branch"` para o fim do fluxo atual.

O campo `"todo"`, tanto aquele definido na raiz da estrutura quanto os que estão definidos nos campos `"options"`, especifica as funções que devem ser executadas para criar as estruturas SHDM se o usuário transitar por esse passo ou escolher uma dessas opções, respectivamente. Nesta estrutura se define o nome da função, os valores dos parâmetros de entrada e quais dos valores resultantes da execução da função vão se conservar para futuros chamados a funções. Note-se, que a seção `"todo"` não é descrita em todos os estados, senão que é especificada só naqueles pontos da interação com o usuário em que se tem completamente a informação e a certeza para a instanciação de um dado modelo SHDM.

Os demais campos da estrutura de um estado podem ser compreendidos na leitura do Quadro 4.1.

No assistente podem-se prever todos os possíveis fluxos, incluindo ciclos, e representá-los por meio de uma máquina de estado, sem restringir nem predeterminar por isso as possibilidades de trajetórias do usuário durante o uso da ferramenta.

No MIV foi concebido um conjunto de telas que facilita a programação por exemplo e a prototipação rápida apresentados nas seções anteriores. Como resultado da interação do usuário com a interface gráfica desenvolvida, se gera uma árvore com as escolhas feitas e dados relevantes ao processo como um todo. A árvore gerada é representada pela estrutura do Quadro 4.2 e como já foi comentado é uma das entradas do módulo seguinte.

```
[{
"wizard_step": identificador do passo na sequência do assistente
"selected_options": opções selecionadas pelo usuário
"user_data": dados particulares de cada passo ou introduzidos pelo
      usuário
"children": árvores filhas
}]
```

Quadro 4.2 Estrutura com informação da interação do usuário

O MGA processa o modelo do assistente (conjunto de estados descritos anteriormente no Quadro 4.1) para uma sequência específica de passos seguidos pelo usuário. O MGA realiza o chamado a funções que determinada em tempo de execução pela combinação comentada assistente-usuário. O assistente utiliza uma árvore que representa a sequência de passos seguidos pelo usuário, e cada estado tem uma lista de funções a executar para esse estado em geral e uma lista de funções para cada uma das opções. Estas listas podem ser opcionais, ou seja, podem não estar presentes por se tratar de um passo meramente ilustrativo ou ainda ser intermediário, que não leve uma operação completamente definida porque simplesmente se estão fazendo escolhas intermediárias para etapas posteriores.

A maneira como o MGA foi implementado é genérica. O MGA mantém um escopo de execução das funções especificadas na estrutura de estados. Isto permite conservar ou disponibilizar nesse escopo valores da execução de uma sequência de passos em particular, para chamados posteriores a funções. Ou seja, a saída de uma função pode ser registrada para servir como entrada de outras funções, representando-se assim definições encadeadas, onde uma definição pode depender de ações anteriores.

O pseudo-código do procedimento geral do módulo no Quadro 4.3 ilustra isto. Têm-se duas estruturas `wizard_definition` e `user_definition` com a sequência de passos retornada no MPO e a sequência de escolhas do usuário, respectivamente. Inicialmente, se realiza o chamado para a criação da aplicação no Synth (linha 4). Seguidamente se percorre a árvore que representa cada uma das escolhas do usuário seguindo uma busca em profundidade (Depth-first search - DFS) (na linha 5) e em função destas se determina o passo onde o usuário fez essa escolha (linha 6). Na especificação do passo, se obtém tanto as funções a executar a esse nível (linha 7) quanto as que dependem da opção escolhida pelo usuário (linha 9). Estas funções são finalmente processadas

(linhas 8 e 10). No processamento, para cada função se procura o nome desta (`funct['function_name']`), se preparam os parâmetros (linha 15) e com isso se realiza a invocação da mesma (linha 16). Alguns dos valores retornados são conservados para usos posteriores (linha 17). Os parâmetros que recebem as funções podem ser de três tipos: de tipo constante, o seja, valores estáticos e atribuídos a priori na estrutura do estado. Podem ser procurados em uma variável do escopo global, porque são valores compartidos por várias funções. Ou pode ser que dependam de uma seleção feita pelo usuário, para o qual é utilizada a variável `user_definition`.

```
1: def create_app
2:  wizard_definition # sequência de passos do wizard
3:  user_definition   # escolhas do usuário
4:  create_app_wizard(app_name) # chamado ao Synth para criar aplicação

5:  foreach step in user_definition.dfs() {
6:    wizard_step <- selecionar de wizard_definition o passo
                     correspondente à escolha do usuário (em step)
7:   todo <- selecionar as funções definidas no campo 'todo' para esse
                     passo (no wizard_step)
8:   process_functions(todo, step) <- processar cada uma das funções
9:   todo_option <- selecionar as funções definidas no campo 'todo'(em
             wizard_step) da opção escolhida pelo usuário (em step)
10:  process_functions(todo_option, step) <- processar cada uma das
             funções do todo_option
11:  }
12: end

13: def process_functions(todo, step)
14:   foreach funct in todo {
15:       params = get_params(funct, step) <- obter os parâmetros de
                     entrada da função a executar do escopo
16:      result = send(funct['function_name'], params) <- executar a
                          função
17:     process_return_values(funct, result) <- conservar valores no
                                                 escopo
18:   }
19: end
```

Quadro 4.3 Pseudo código da implementação do módulo MGA

O procedimento exposto pode ser exemplificado para uma estrutura de entrada como a do Quadro 4.4 e uma sequência de escolhas como a do Quadro 4.5. Como parte das instanciações dos modelos SHDM através do API do Synth se pode criar aplicações, *landmarks*, índices, contextos, classes em contexto, atributos, etc., para a etapa de navegação. No exemplo do Quadro 4.4 se

ilustram fragmentos de três passos no modelo retornado pelo MPO onde se especificam funções para a criação de um contexto, um índice e um *landmark*, respectivamente, no processamento do MGA.

No primeiro passo com identificador `"0.1"` se especificam o campo `function_name` com valor `create_context` para indicar o chamado à função do Synth que cria um contexto. No campo `params` se define o conjunto de parâmetros de entrada onde cada um tem um tipo (dos tipos de parâmetros explicados anteriormente), um nome para se referir ao nome do parâmetro no API do Synth e um valor que é o que se vai a atribuir ao parâmetro na invocação do método. Desta forma, no passo `"0.1"` se tem dois parâmetros constantes, um com a expressão da consulta do contexto (no caso, todos os usuários) e o outro com o nome para o contexto ("Usuário"). O campo `results` define um conjunto de objetos para guardar o resultado da execução da função. No exemplo, têm-se dois objetos: o primeiro para dizer que o contexto criado e representado na variável `context` no Synth vai se conservar no escopo global do MGA na variável com nome `context_id`, assim pode se obter este valor para o chamado a outra função. O segundo, para dizer que o índice criado automaticamente e associado ao contexto atual é representado na variável `default_index` no Synth e vai se conservar no escopo global na variável `index_id`. Assim, por exemplo no passo `"1.0"` pode-se usar este índice para criar um *landmark* quando se especifica o parâmetro de tipo `global_var` com valor `index_id`. Isto indica o acesso à variável `index_id` do escopo global e seu passo ao parâmetro de mesmo nome (declarado como `"name": "index_id"`) do método `create_index_landmark`. De maneira análoga ao passo `"0.1"` podem ser descritos os passos `"0.2"` e `"1.0"`.

```
wizard_definition = [
{
     id": "0.1",
     "next": "1.0", ...
     "todo": [{
            "function_name": "create_context",
            "params": [
                  {"type":   "constant",   "name":   "query",   "value":
                                    "AUCTION::Usuario.find_all"},
                  {"type": "constant", "name": "name", "value":
                                    "Usuario"}
            ],
            "results": [
                  {"name": "context", "global_var": "context_id" },
                  {"name": "default_index", "global_var": "index_id"}
            ]
      }]
}, {
     "id": "0.2",
     "next": "1.0", ...
     "todo": [{
      "function_name": "create_index",
      "params": [
            {"type": "global_var", "name": "context_id", "value":
                                          "context_id"},
            {"type": "constant", "name": "name", "value": "Usuario"}],
      "results": [{"name": "index", "global_var": "index_id"}]
     }]
     },{
     "id": "1.0", ...
     "todo": [{
            "function_name": "create_index_landmark",
            "params": [
                  {"type":  "global_var",  "name":  "index_id",  "value":
                                                      "index_id"},
                  {"type": "constant", "name": "name", "value":
                                                      "Usuarios"}]
     }]
}]
```

Quadro 4.4 Exemplo da estrutura de um passo do assistente com definição das funções a executar no MGA

O Quadro 4.5 mostra um exemplo de escolhas do usuário, onde se especifica que ele transitou pelo estado `"0"` e selecionou a opção `2`, passou ao estado `"0.2"` e escolheu a opção `1` e terminou no passo `"1.0"` com a opção `1`. Quando o usuário transitar pelo passo `"0.2"` pode se ver no Quadro 4.4 que vai se criar um índice com nome `"Usuario"` para um contexto gerado previamente e guardado em `context_id` e este índice vai se guardar em `index_id`. Ao usuário

transitar para o passo `"1.0"` o MGA vai criar um *landmark* com uma âncora para o índice criado no passo anterior.

```
user_definition =
{
     "wizard_step": "Artificial Node",
     "children": [
             {"wizard_step": "0", "selected_options": [2] },
             {"wizard_step": "0.2", "selected_options": [1] },
             {"wizard_step": "1.0", "selected_options": [1] },
     ]
}
```

Quadro 4.5 Exemplo da estrutura onde se representam as escolhas do usuário

A concepção dos módulos na implementação e a abstração do diálogo através de uma estrutura comum faz com que estes possam ser desenvolvidos independentemente e usados em outros sistemas. As funcionalidades de cada módulo ficam isoladas, tornando-os totalmente independentes entre si, comunicando-se apenas por meio dessa estrutura comum. Por exemplo, se se se quiser ampliar o diálogo com o usuário oferecendo mais opções, só é preciso modificar o MPO. Se se quiser enriquecer a interação na interface visual para contribuir ao estilo de manipulação direta, seria só responsabilidade do MIV.

4.7. Arquitetura de implementação

O MPO e o MGA do assistente, encarregados do processamento de uma ontologia de domínio e da geração das instâncias do modelo SHDM, respectivamente, foram desenvolvidos usando a linguagem Ruby[6]. O MPO, além disto, está descrito em Ruby usando a linguagem específica de domínio (DSL pelas siglas em inglês) do ActiveRDF[7] para a manipulação dos recursos RDF da ontologia.

O MIV do assistente, encarregado da geração da interface visual, foi implementado sobre Ruby on Rails[8] que é um framework MVC (Model–View–Controller) para desenvolvimento de aplicações web. Sob esse aspecto, o MIV é uma aplicação web desenvolvida segundo o paradigma MVC. O código do lado

[6] https://www.ruby-lang.org/

[7] http://www.activerdf.org

[8] http://rubyonrails.org/

do cliente desse módulo foi descrito usando AngularJS[9] que é um framework javascript para adicionar interatividade ao HTML.

Além destes módulos foi definida uma camada APIRest para permitir o acesso às funções de criação das estruturas SHDM do Synth.

4.7.1. Ruby

Ruby foi escolhida como linguagem de programação para a construção da ferramenta por ser usada no Synth, do qual foi aproveitado parte de seu suporte. Assim, também era possível continuar usando o popular framework Ruby on Rails que facilita o desenvolvimento de aplicações web. Além de oferecer características desejáveis das linguagens de *scripting* dinâmicas, Ruby se ajusta bem aos requerimentos. Entre as razões mais importantes estão:

- Pelos seus poderosos recursos de reflexão, Ruby permite representar no MGA a execução de funções especificadas em tempo de execução.
- Por seu sistema de tipos dinâmico, os objetos não requerem um tipo no tempo de desenvolvimento e não se limita a funcionalidade do objeto a seu tipo definido. Por exemplo, o mecanismo de ruby "*mixin*" permite estender e sobrescrever objetos e classes com dados e funcionalidades específicas em tempo de execução.
- Sua capacidade de metaprogramação permite sobrescrever seu sistema de tipo interno, ou gerar uma hierarquia de classes compatível on-the-fly em tempo de execução. Assim, é possível o mapeamento das classes RDF para as classes em linguagem de programação orientado a objeto.
- Sua simplicidade, leveza e tolerância na sintaxe permite que a curva de aprendizado seja suave nas áreas onde o usuário do assistente necessita utilizar linguagem de programação (Por exemplo para definir atributos computados).

[9] https://docs.angularjs.org/

4.7.2. Ruby on Rails

O Rails é um framework de código aberto de desenvolvimento web escrito na linguagem Ruby. As aplicações criadas utilizando o framework Rails são desenvolvidas com base no padrão de arquitetura MVC. Ele foi projetado para tornar a programação de aplicações web mais fácil, fazendo várias suposições sobre o que cada desenvolvedor precisa para começar. Permite escrever código organizado favorecendo a convenção sobre a configuração. Assim, responde a uma filosofia que faz muito conveniente seu uso para o nosso desenvolvimento: tudo é possível mas o default é mais fácil. Além disso, o Ruby on Rails foi escolhido para a construção da ferramenta por ser usado no Synth, e era interesse reaproveitar parte do suporte do Synth.

4.7.3. ActiveRDF

ActiveRDF e uma biblioteca orientada a objeto para acessar dados RDF a partir de programas escritos em linguagem Ruby. É completamente dinâmica, oferece manipulação total e capacidades de consulta sobre dados RDF. Não está sujeito a um esquema e pode ser usado sobre diferentes fontes de dados. Oferece uma linguagem específica de domínio (DSL) para o modelo RDF dando acesso a recursos RDF, classes e propriedades de modo programático. Sua integração com o popular framework Rails permitiu seu uso no Synth (desenvolvido sobre Rails). O processamento da ontologia de domínio, portanto, foi realizado desde o kernel do Synth pois já tinha incluído o suporte para ActiveRDF e o trabalho com a DSL.

4.7.4. Angular JS

Angular JS é um framework JavaScript do lado de cliente. O framework adapta e estende o HTML tradicional para uma melhor experiência com conteúdo dinâmico. Oferece templates declarativas com a ligação direta e bidirecional dos dados (two-way data-binding) que permite sincronização automática de modelos e visões. Como resultado, AngularJS abstrai a manipulação do DOM e melhora os testes. Angular segue o padrão MVC e encoraja o baixo acoplamento entre apresentação, dados e componentes lógicos.

Esta biblioteca foi usada no módulo 2 na definição das telas tipo do assistente, assim como na ligação destas com o modelo, especificado em um recurso JSON dinâmico obtido no MPO.

4.7.5.Synth

O assistente após gerar as instâncias dos modelos SHDM se comunica com o Synth para a especificação e persistência destes e o Synth oferece uma aplicação resultante da interpretação desses modelos. O Synth foi implementado sobre Ruby on Rails.

> O Synth é um ambiente de desenvolvimento que dá suporte à construção de aplicações modeladas segundo o método SHDM, fornecendo um conjunto de módulos capazes de receber como entrada os modelos gerados na execução das etapas do método SHDM e produzir como saída uma aplicação hipermídia descrita por estes modelos. O Synth também dispõe de um ambiente de autoria que facilita a inserção e edição destes modelos através de uma interface gráfica de formulários que pode ser executada em qualquer navegador de internet.
>
> (Bomfim, 2011)

5
Exemplos de Uso

Para exemplificar o uso e a utilidade da abordagem proposta vai se considerar parte de um cenário para o qual foi desenvolvida uma aplicação usando o ambiente de autoria dirigido por modelos Synth. A idéia é comparar os dois ambientes de desenvolvimento e ilustrar como ficam representadas no assistente oferecido as diferentes especificações realizadas no Synth para o projeto de navegação. O uso do assistente leva a um menor esforço de parte do projetista, pois encapsula a síntese dos modelos para instanciar a aplicação final no ambiente Synth.

Para a compreensão aprofundada das etapas de modelagem utilizando o Synth, sugere-se a leitura do Capítulo 4 da dissertação de (Bomfim, 2011).

5.1.
Descrição dos cenários

O cenário que vai se tratar faz parte de uma aplicação simplificada de cunho didático para introduzir de maneira sistemática a modelagem do SHDM e o uso do Synth. Trata-se de um site para a venda e compra de produtos mediante leilões. Especificamente, o foco está na interação do vendedor para ofertar um produto em um leilão.

Cada etapa do método SHDM produz um ou mais modelos que focam em aspectos distintos de uma aplicação hipermídia. O método SHDM foi extensamente discutido em trabalhos anteriores, principalmente em (Lima, 2003), (Szundy, 2004) e (Nunes, 2005), assim como algumas outras mudanças introduzidas podem se consultar em (Bomfim, 2011).

Por isso, sô serão apresentados sucintamente os artefatos obtidos sem detalhar a notação.

5.1.1. Cenário 1: Ofertar um produto em um leilão

Ator: Vendedor

A aplicação mostra uma lista de produtos e apresenta de cada um o **nome, categoria** e **se está em oferta**. Eu posso procurar o produto pela **categoria** e espero obter **uma lista dos produtos (nome, categoria, se está em oferta)** que cumprem com esse parâmetro. Eu escolho o produto a ofertar e a aplicação mostra a **informação do produto (nome, descrição, categoria, data do cadastro, se está em oferta, quantidade)**. Além disso, a aplicação mostra uma lista das ofertas anteriores relacionadas com esse tipo de produto**.** Se eu clicar em uma oferta anterior se mostra, na tela de detalhe, **o nome do produto, preço, data, tempo ativo do leilão, preço inicial e final do leilão**. Eu escolho a opção ofertar em leilão e a aplicação mostra uma interface para programar o leilão. Eu informo **o tempo de início, de fim, o lance mínimo, a diferença mínima entre lances e os meios de pagamento**. Finalizo o leilão. Se eu desejar ofertar outro produto eu retorno à lista dos produtos e escolho outro.

Quadro 5.1 Cenário "Ofertar um produto em um leilão"

Caso de uso 1: Programar um leilão para um produto
Descrição:

1. A aplicação mostra uma lista de produtos
2. O usuário escolhe o produto a ofertar
3. A aplicação mostra a informação do produto (foto, nome, descrição, quantidade e a lista de ofertas anteriores relacionadas com esse tipo de produto)
4. O usuário escolhe a opção de ofertar em leilão
5. Quando o usuário escolhe ofertar um produto informa a data de início, de fim, o lance mínimo, a diferença mínima entre lances e os meios de pagamento
6. O usuário finaliza o leilão
7. Se ele desejar ofertar outro produto ele pode retornar à lista de seus produtos no passo 1.

FLUXOS ALTERNATIVOS

(A1) Alternativa ao Passo 2 – Filtragem de produtos por categoria

- O usuário informa a categoria
- O usuário obtém uma lista dos produtos (nome, categoria, se está em oferta) que cumprem com essa categoria, o sistema retorna ao Passo 2

(A2) Alternativa ao Passo 4 – Ver oferta anterior do produto

- O usuário escolhe ver uma oferta anterior
- A aplicação vai mostrar o nome do produto, preço, data, tempo ativo do leilão, preço inicial e final do leilão. O sistema retorna ao Passo 4

Quadro 5.2 Caso de Uso "Ofertar um produto em um leilão"

5.2. Esquema conceitual

A Figura 5.1 apresenta um diagrama de classes simplificado utilizando uma notação no estilo UML que representa a ontologia utilizada para a aplicação.

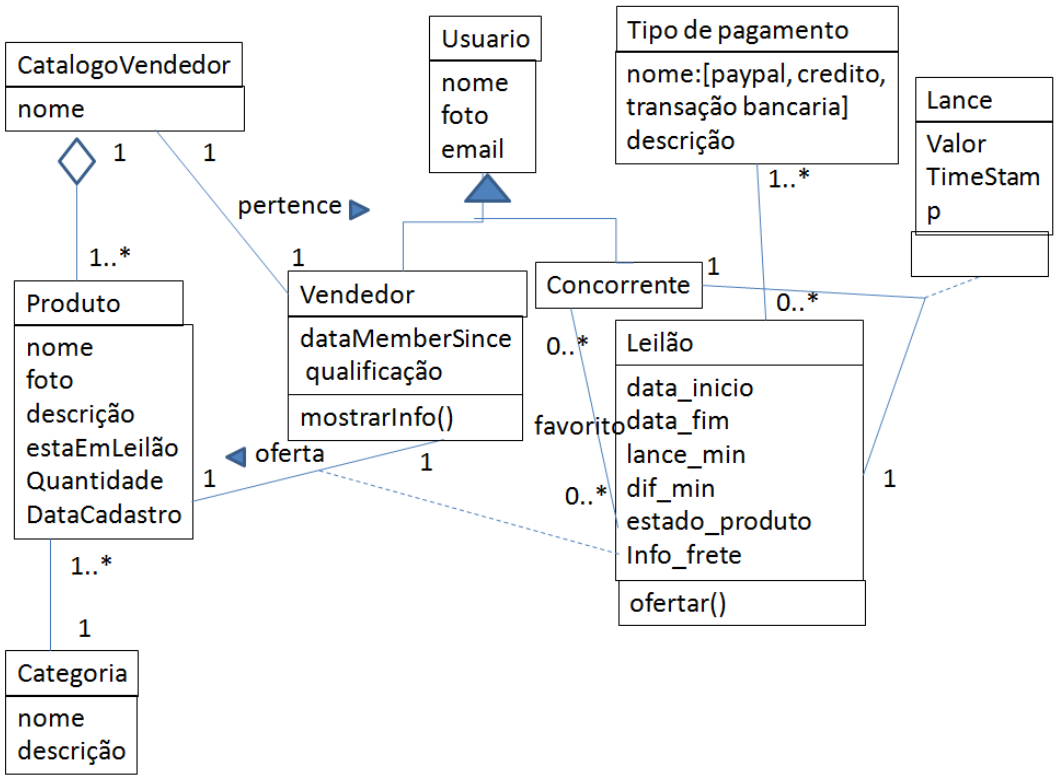


Figura 5.1 Esquema conceitual

5.2.1.
Modelagem navegacional

A Figura 5.2 apresenta a parte do modelo de navegação correspondente ao cenário “Ofertar um produto em um leilão”. A Figura 5.3 apresenta o modelo de navegação da aplicação em geral.

Para a classe “Produto”, foram especificados os contextos “Produto Alfabético”, para navegação em ordem alfabética por todos os produtos, e “Produto por Categoria” para os produtos de uma dada categoria, cujos nós são derivados da propriedade “categoria” de “Produto”. O contexto “Produto por Categoria” é parametrizado e espera receber como parâmetro o identificador de um recurso do tipo “Categoria”. Os índices “Produtos” e “Produtos por Categoria” são baseados nos contextos “Alfabético” e “por Categoria”, respectivamente. O índice “Categorias” define o conjunto de todas as categorias de produto que apontam para o contexto de navegação “Produtos por Categoria”.

Para a classe “Leilão”, foi especificado o contexto “Leilão por Produto”. Os nós deste contexto são derivados da propriedade “temProduto” de “Leilão”. Este contexto é parametrizado e espera receber como parâmetro o identificador de um recurso do tipo “Produto”.

Dos contextos “Produto Alfabético” e “Produto por Categoria” se pode navegar para o contexto “Leilão por Produto”, para por exemplo mostrar, dado um produto, os leilões anteriores em que foi ofertado. Da mesma maneira, do contexto “Leilão por Produto” da classe Leilão se pode navegar para os contextos “Produto Alfabético” e “Produto por Categoria”, para voltar ao produto do qual se partiu.

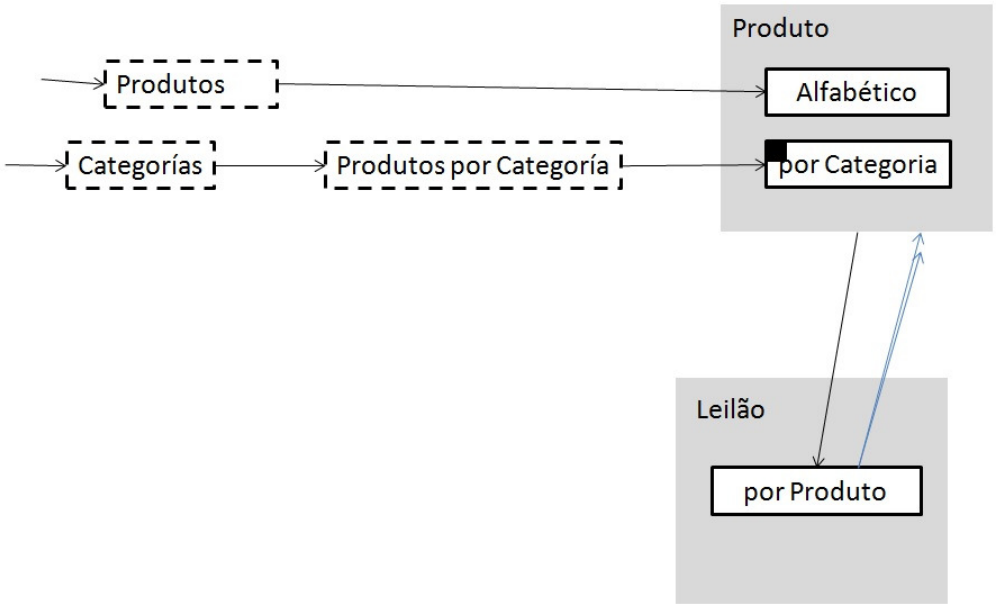


Figura 5.2 Modelo de navegação “Ofertar produto em um leilão”

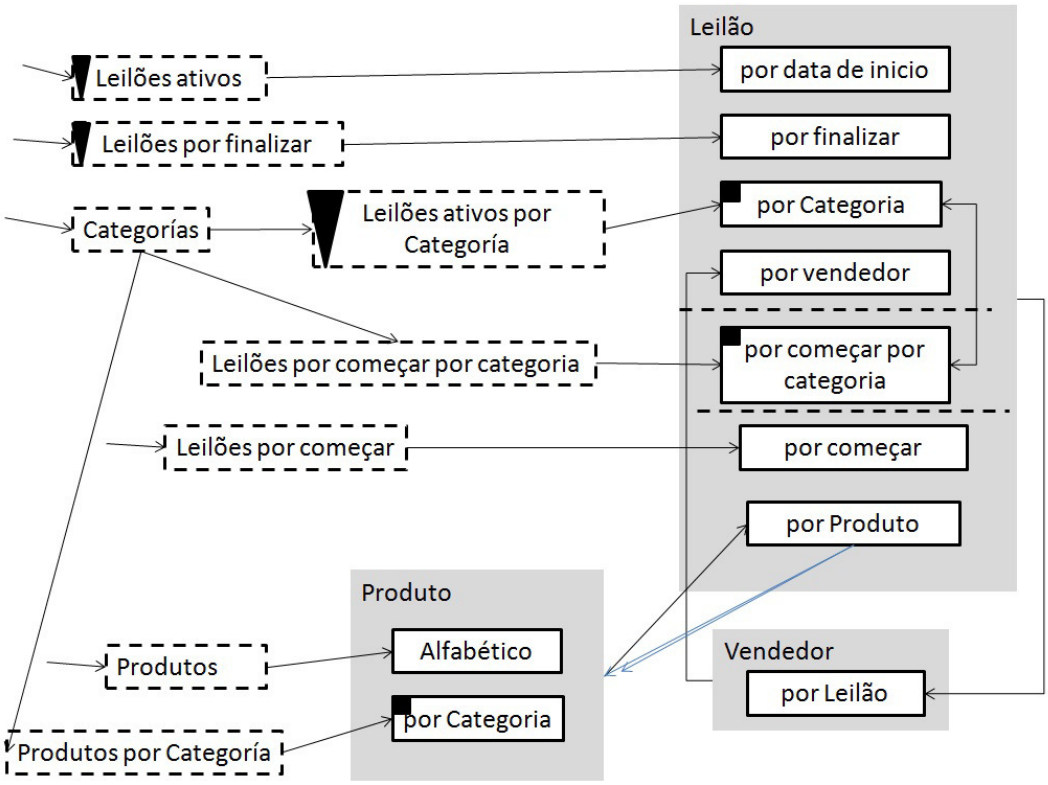


Figura 5.3 Modelo de navegação

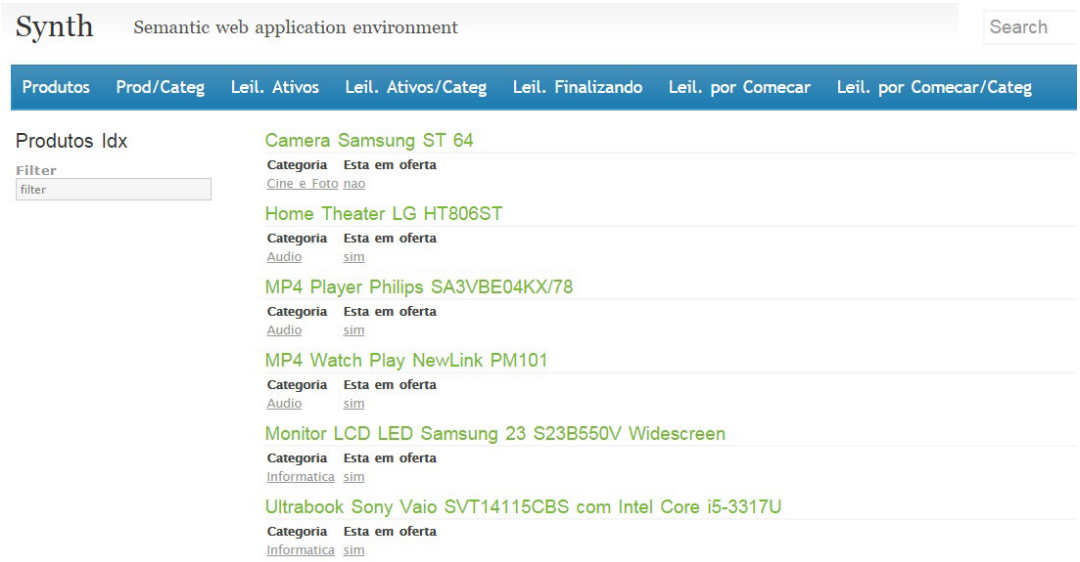


Figura 5.4 Tela do índice "Produtos"

A tela da Figura 5.4 mostra uma lista de produtos. Para a representação desta foi definido o índice “ProdutosIdx” (Quadro 5.5) com os elementos do contexto “Produtos” (Quadro 5.3) para mostrar todos os produtos. No índice, foram especificados os atributos navegacionais: um do tipo âncora para o contexto, que apresenta o nome do produto (i.e. “Camera Samsung ST 64”) e os outros dois do tipo computados que mostram a categoria (i.e. “Cine e Foto”) e se o produto está em oferta (i.e. “não”).

Imaginemos agora que o usuário tem em mente uma tela como esta. Usando o assistente, a sequência de passos seria:

1. Escolher entres as classes da ontologia, a classe “Produto” (Figura 5.5)

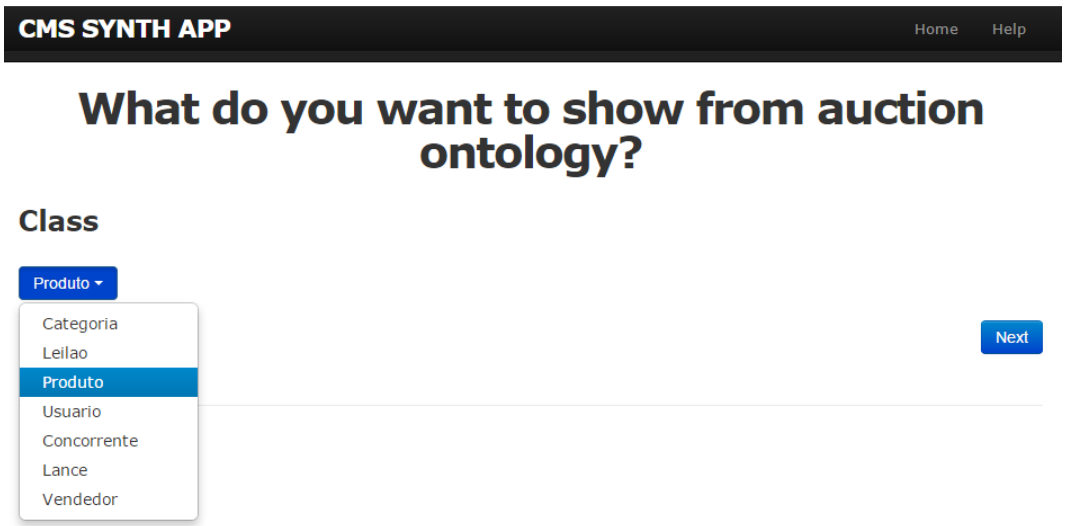


Figura 5.5 Tela para seleção da classe

2. Escolher a opção para mostrar uma lista de produtos (Figura 5.6).

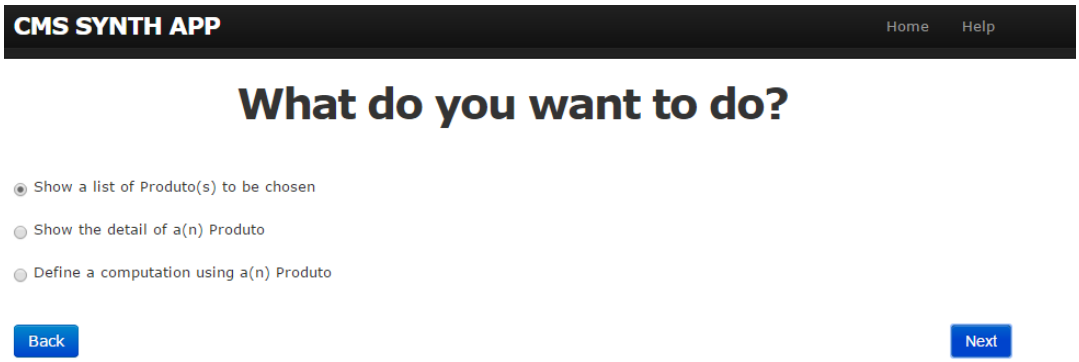


Figura 5.6 Ações a realizar com as instâncias de Produto

3. Escolher a opção para indicar a seleção de um só produto (Figura 5.7). Em uma lista de elementos o usuário pode querer que se selecione um elemento só ou vários, para posteriormente fazer uma outra operação.

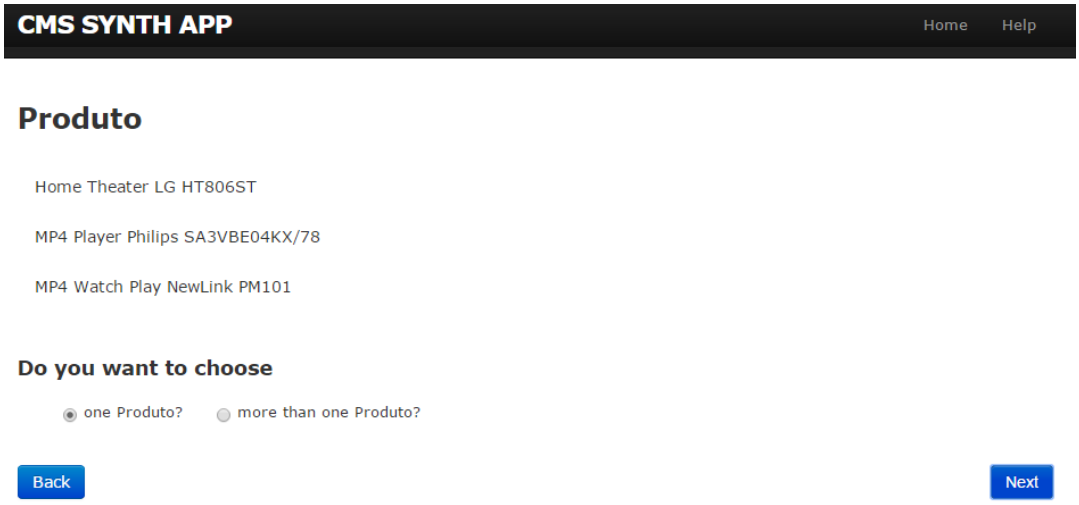


Figura 5.7 Tela para especificar a seleção simples ou múltipla dos produtos

4. Escolher a opção para mostrar atributos adicionais para cada produto (Figura 5.8)

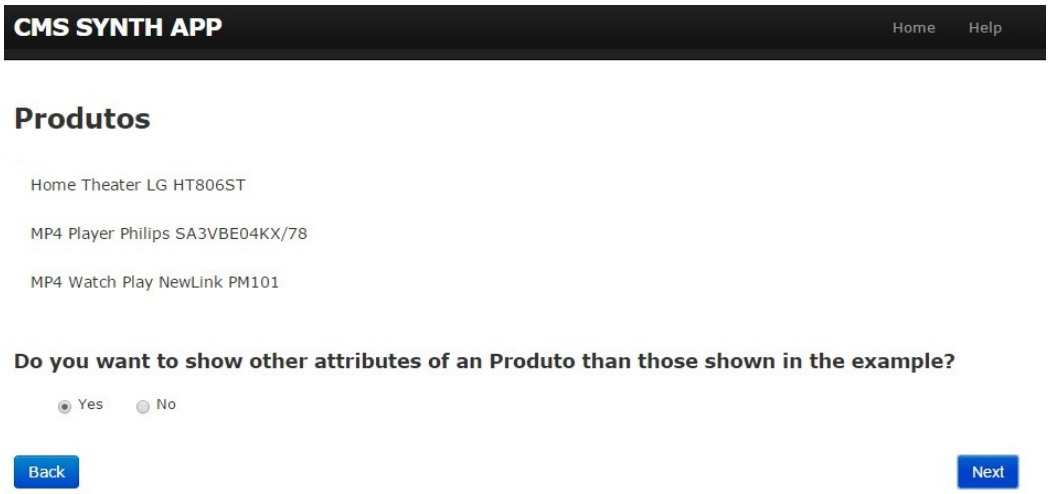


Figura 5.8 Tela para informar se se quer mostrar mais de uma propriedade para cada elemento

5. Escolher a opção para especificar um atributo computado (Figura 5.9). As propriedades de um produto podem ser dos tipos: diretas, quando são propriedades que na ontologia são declaradas para essa classe, o seja, para as que têm essa classe como domínio; pertencentes a outras classes que se relacionam com a classe de interesse através de um ou mais níveis de indireção (composição de relações); atributos computados para mostrar o resultado de uma operação sobre um valor.

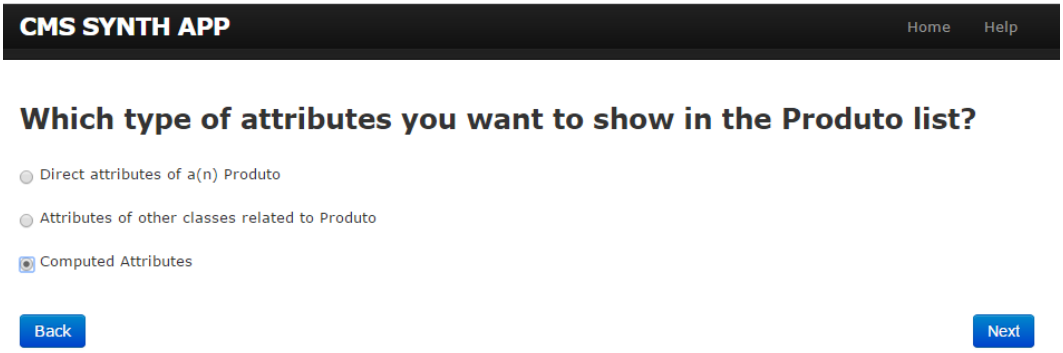


Figura 5.9 Opções para tipo de propriedades

6. Definir o nome e a operação a executar com o atributo computado "Categoria" (Figura 5.10)

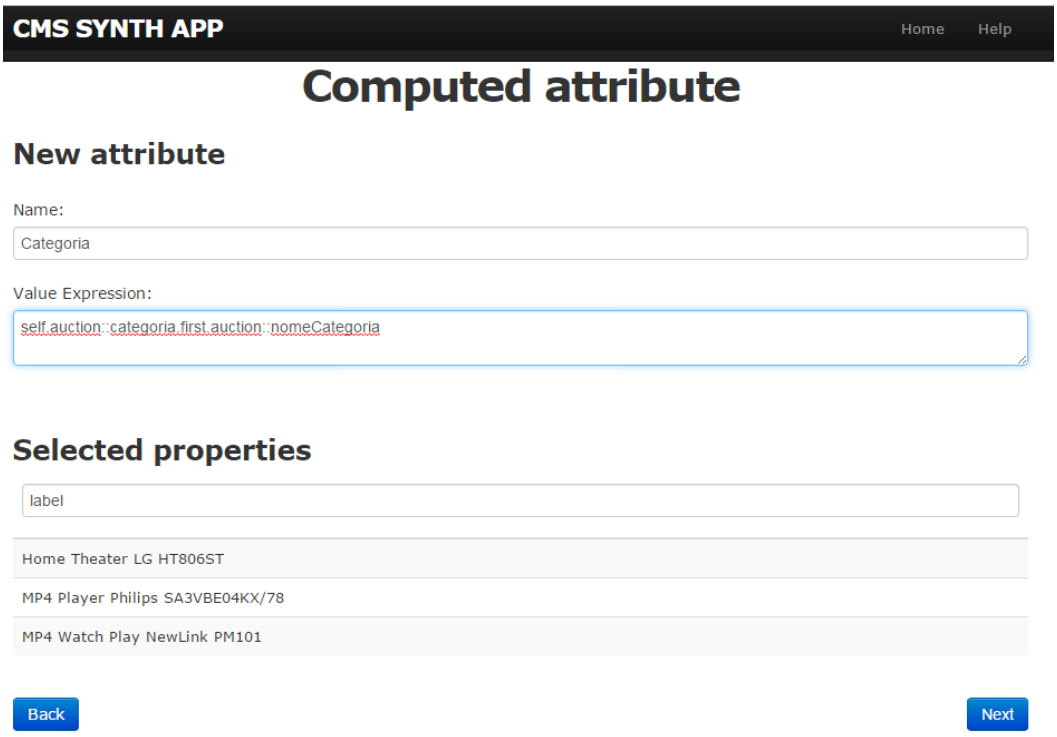


Figura 5.10 Definindo o atributo computado "Categoria"

7. Escolher a opção para especificar um atributo computado (Figura 5.11).

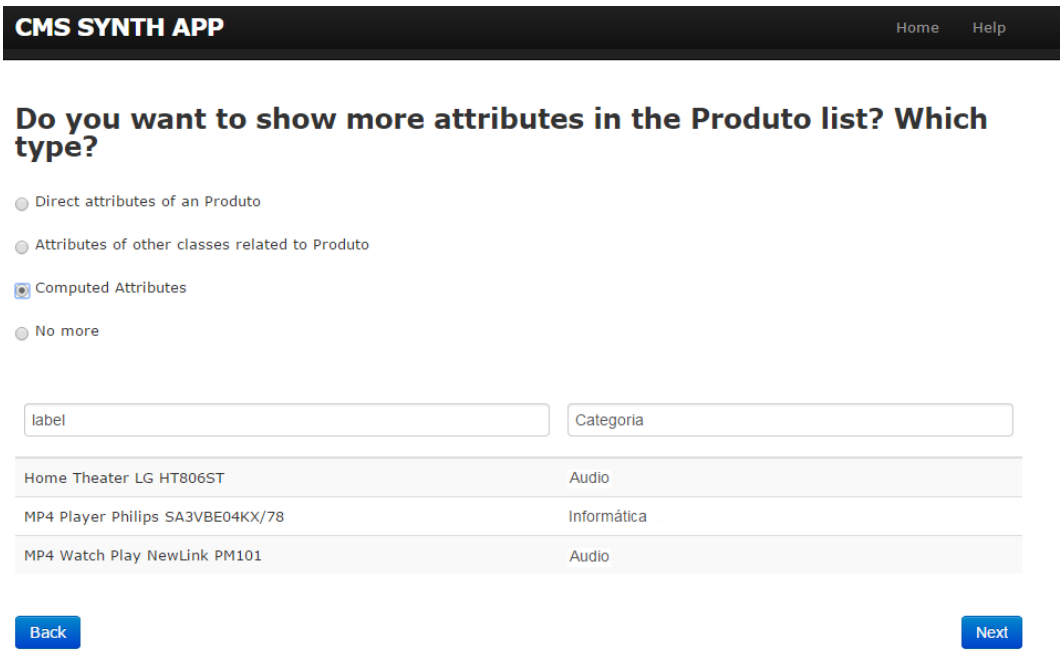


Figura 5.11 Opções para tipo de propriedades

8. Definir o nome e a operação a executar com o atributo computado “Está em oferta” (Figura 5.12).

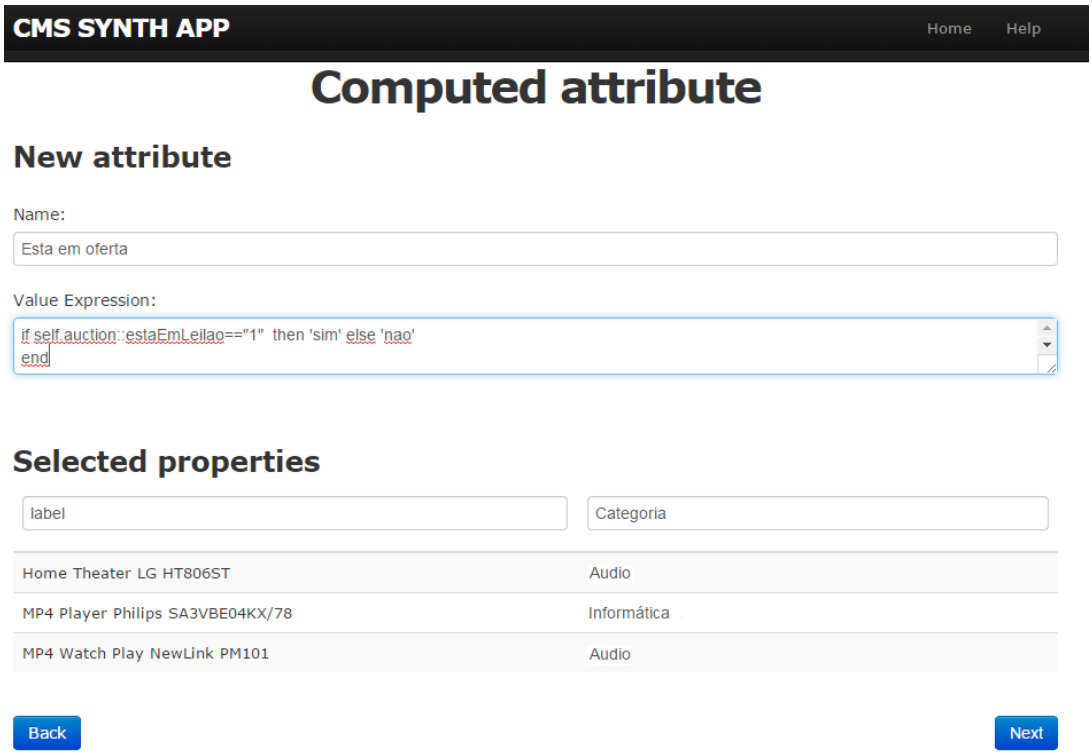


Figura 5.12 Definindo o atributo computado "Está em oferta"

9. Escolher a opção para não definir mais atributos (Figura 5.13).

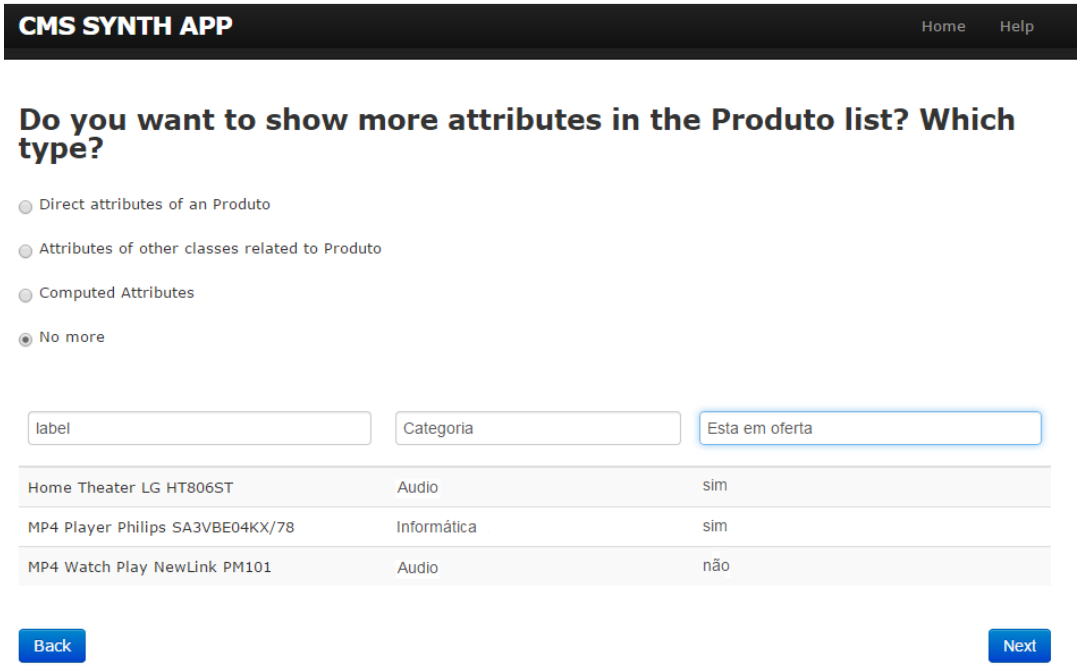


Figura 5.13 Terminar de definir atributos

Um exemplo adicional de outra tela desenvolvida com o Synth é o da Figura 5.14, que mostra os detalhes de um produto. A tela é mostrada ao clicar um elemento do índice na Figura 5.4 (No caso o elemento “MP4 Watch Play

NewLink PM101"). Ela representa um nó no contexto de todos os produtos ("Produtos" no Quadro 5.3), assim é possível navegar por todos os nós deste contexto. Para sua definição, além da especificação do contexto "Produtos", foi preciso uma classe para o contexto de todos os produtos ("ClasseEmContextoProduto" no Quadro 5.7). Assim, foram acrescidos os atributos "Categoria" (atributo computado) e "Leiloes anteriores" (atributo do tipo índice) cujos valores são derivados das propriedades "categoria" da classe "Produto" e "temProduto" da classe "Leilão" respectivamente. Para o atributo "Leiloes anteriores" então foi definido o índice "LeiloesPorProdutoIdx" com os elementos do contexto "LeiloesPorProduto" (Quadro 5.6).

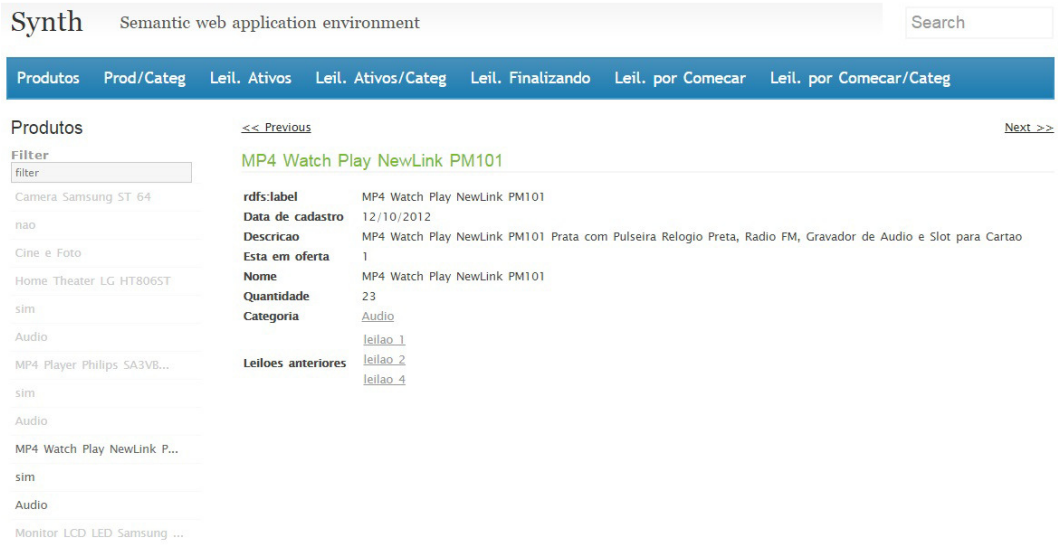


Figura 5.14 Tela de acesso a um nó no contexto de todos os produtos ("Produtos")

Com nosso assistente para uma tela como a da Figura 5.14, a sequência de passos seria continuando a partir das telas apresentadas anteriormente:

10. Escolher a opção para definir um campo âncora (Figura 5.15).

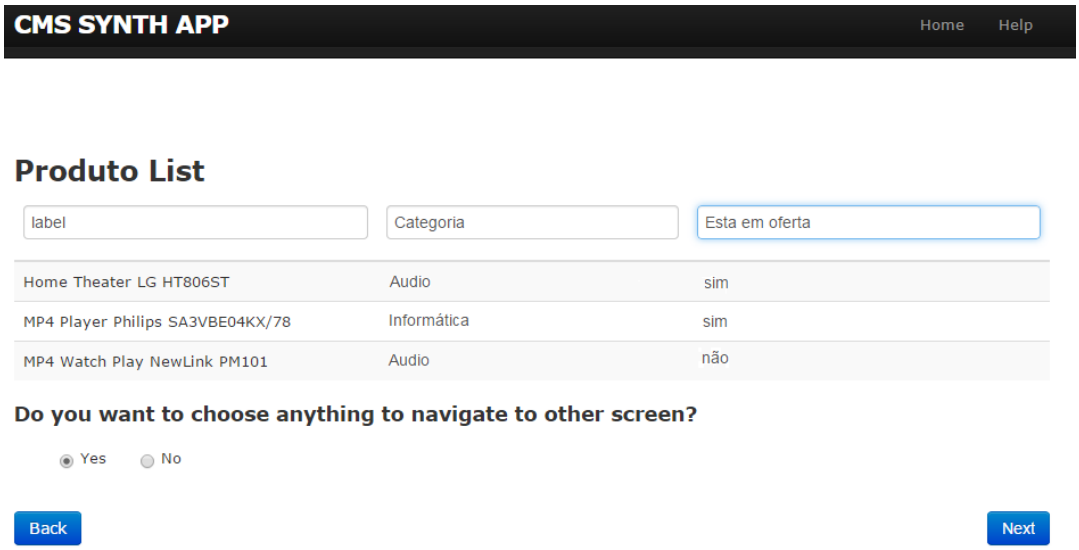


Figura 5.15 Tela para decidir se vai ter campo âncora

11. Escolher o campo “label” para definir o âncora (Figura 5.16).

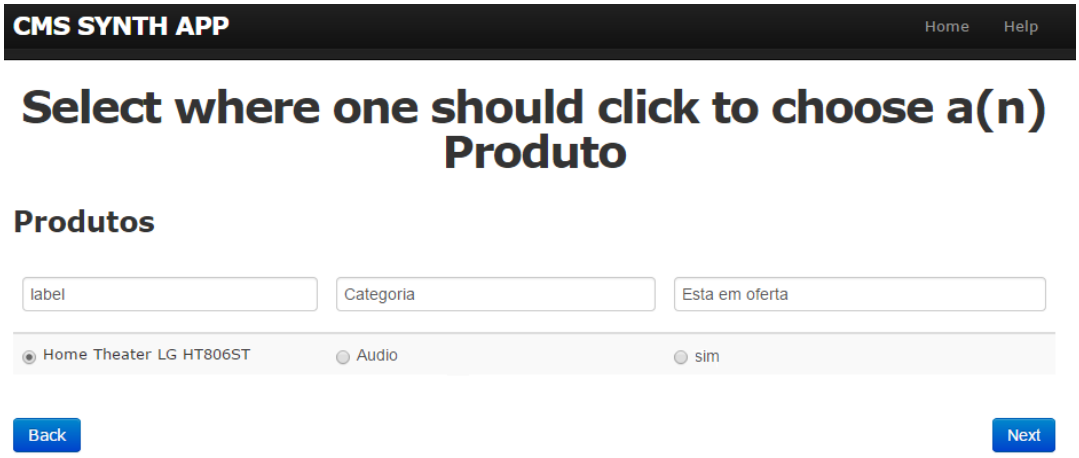


Figura 5.16 Opções para definir um campo âncora

12. Escolher a classe a mostrar após clicar em um produto (Figura 5.17).

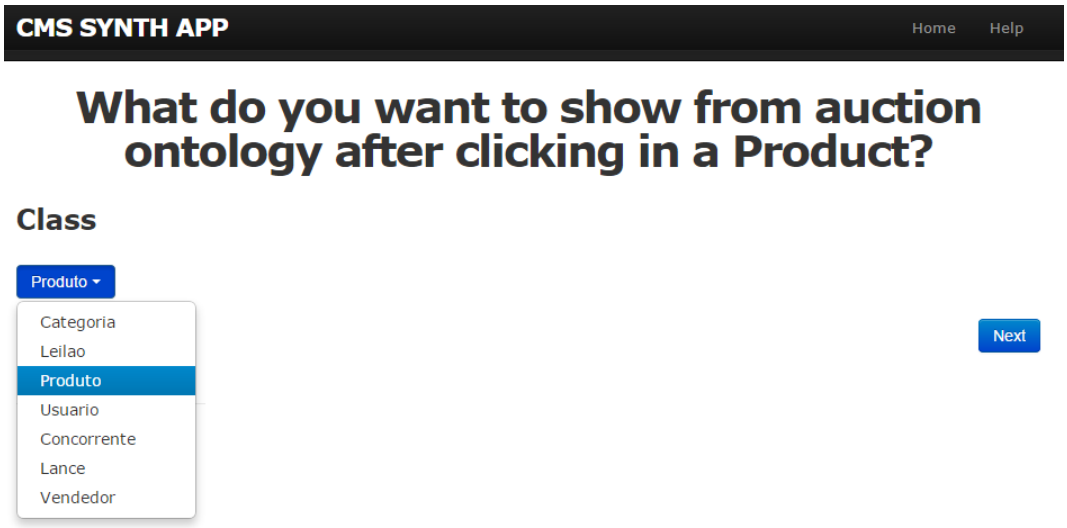


Figura 5.17 Classe a mostrar após clicar na âncora

13. Escolher a opção para mostrar o detalhe de um produto (Figura 5.18).

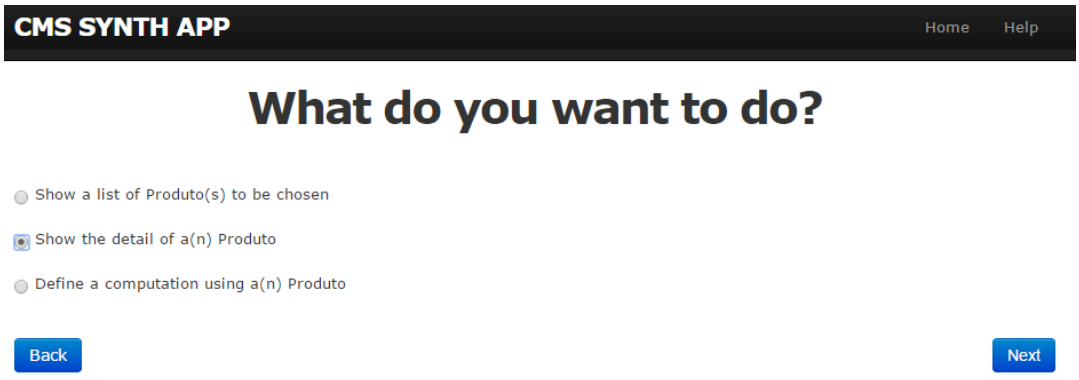


Figura 5.18 Ações a realizar com as instâncias de Produto

14. Escolher a opção para mostrar mais atributos do produto (Figura 5.19)

Figura 5.19 Tela para dizer se se quer mostrar outras propriedades na visão de detalhe do produto

15. Escolher a opção para especificar entidades relacionadas com Produto (Figura 5.20). No caso se quer mostrar os leilões anteriores do produto.

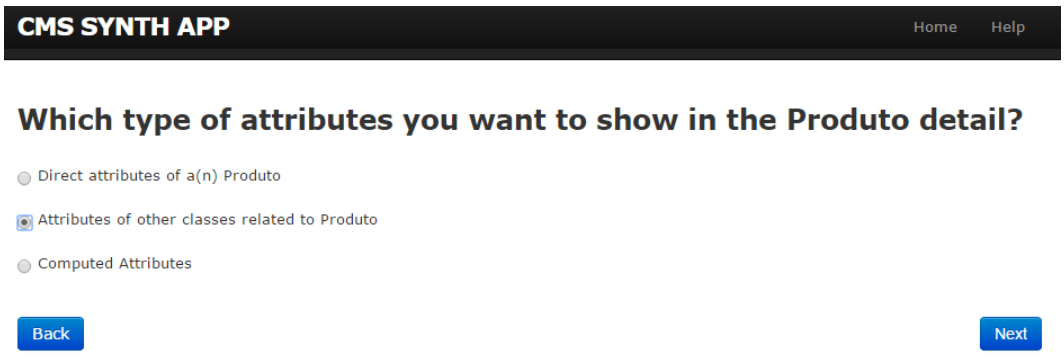


Figura 5.20 Opções para tipo de propriedades

16. Escolher a classe relacionada Leilão para mostrar os leilões anteriores de um produto (Figura 5.21).

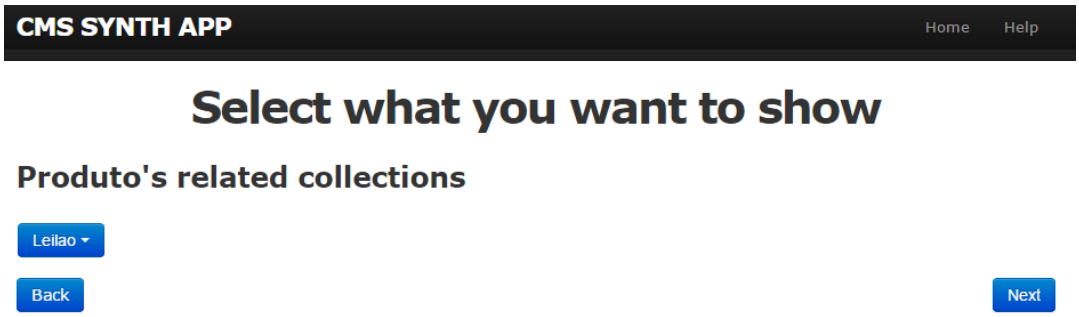


Figura 5.21 Escolha de uma classe relacionada com Produto

17. Escolher o caminho que relaciona as classes Produto e Leilão. No caso é só um caminho (Figura 5.22).

Figura 5.22 Tela que apresenta exemplos do caminho da relação de Produto com Leilão

18. Escolher a propriedade de interesse para relacionar um produto com seus leilões anteriores. No caso, só tem uma propriedade(“temProduto”) (Figura 5.23)

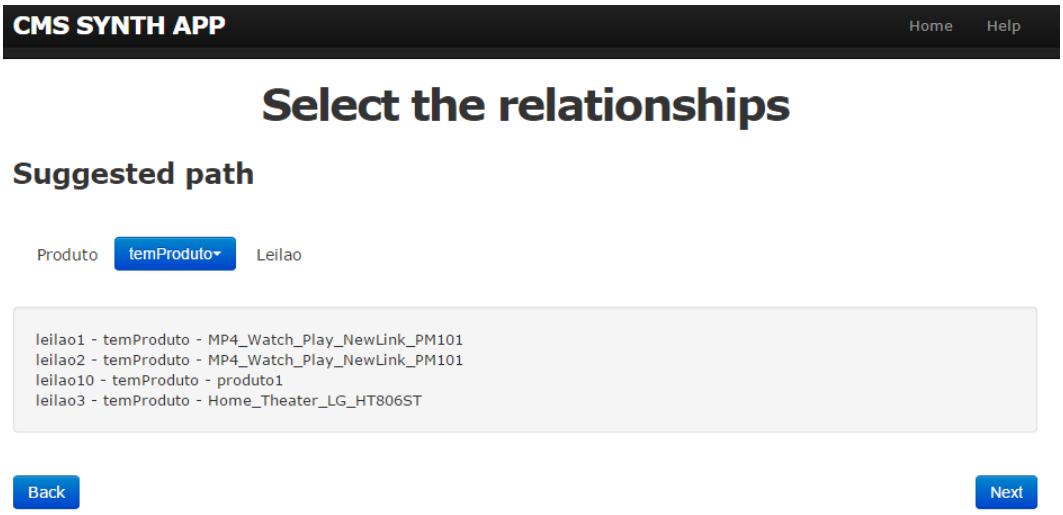


Figura 5.23 Tela para escolher a propriedade de interesse do usuário

A tela da Figura 5.24 apresenta o detalhe de um leilão e é mostrada ao clicar um elemento do índice “Leilões anteriores” na Figura 5.14 (No caso “leilão 1”). Ela representa um nó no contexto dos leilões anteriores por produto (“LeiloesPorProduto” no Quadro 5.6), assim é possível navegar por todos os nós deste contexto. Para sua definição, além da especificação do contexto “LeiloesPorProduto”, foi preciso uma classe para o contexto (“InContextClass”). Assim, foram acrescidos os atributos “Qualificador do vendedor”, “Vendedor”, “Categoria do produto” e os atributos do tipo âncora para contexto para, por exemplo, obter mais informação e outros leilões do vendedor ou para voltar ao detalhe do produto da Figura 5.14. Estes valores são derivados das propriedades “temVendedor” da classe “Leilão” e categoria da classe “Produto”. Além dos contextos já comentados foi preciso definir o contexto “VendedoresPorLeilão”.

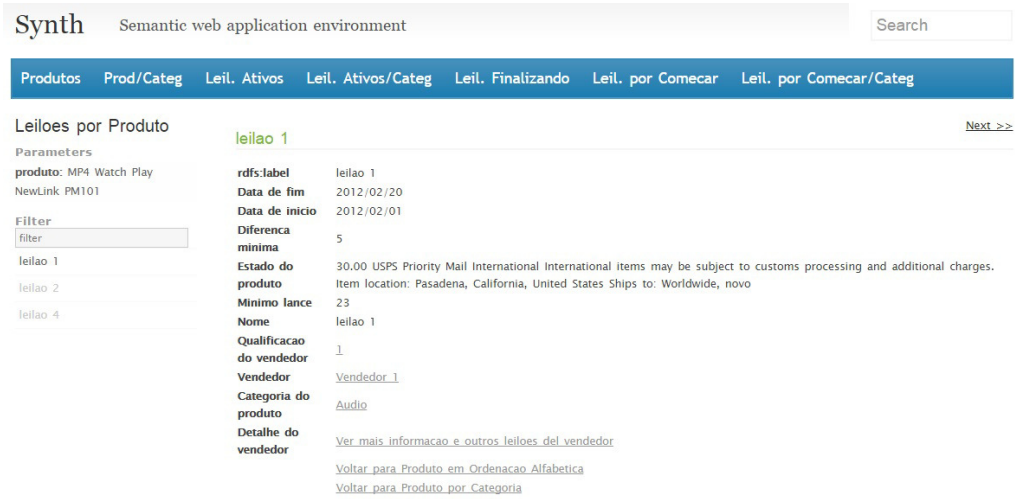


Figura 5.24 Tela de acesso a um nó no contexto “LeiloesPorProduto”

De maneira similar a apresentada para a tela de detalhe do produto (Figura 5.14) os passos a seguir seriam realizados no assistente para obter o detalhe do leilão da Figura 5.24.

A seguir serão apresentadas as especificações dos contextos e índices da aplicação para o cenário “Ofertar produto”, no formato utilizado pelo Synth. Esta especificação é gerada pelo MGA, e é idêntica àquela gerada se o desenvolvedor entrar com a especificação destes modelos diretamente na interface de autoria do Synth.

O Quadro 5.3 apresenta a especificação em RDF do contexto “Produtos” (“Alfabético” no modelo da Figura 5.3). A expressão de consulta do contexto (propriedade shdm:context_query) está descrita em Ruby usando a DSL do ActiveRDF, cujo significado é “Todos os recursos do tipo `AUCTION::Produto` que tenham quantidade maior que zero, em ordem alfabética do valor da propriedade auction::nomeProduto”.

```
:Produtos a shdm:Context;
       shdm:context_name "Produtos";
       shdm:context_title "Produtos";
       shdm:context_query "AUCTION::Produto.find_all
                .select{|x|(x.auction::quantidade.first<=>0)==1}
                .sort_by{|x| x.auction::nomeProduto.first}".
```

Quadro 5.3 Especificação em RDF do contexto “Produtos”

O Quadro 5.4 apresenta a especificação do contexto "ProdutosPorCategoria" ("PorCategoria" no modelo da Figura 5.3). Este contexto possui o parâmetro "categoria". É possível notar na expressão de consulta o uso do parâmetro "categoria". A expressão de consulta obtém os tipos de produtos de uma dada categoria (passada como parâmetro) que têm uma quantidade maior que zero.

```
:ProdutosPorCategoria a shdm:Context;
       shdm:context_name "ProdutosPorCategoria";
       shdm:context_title "Produtos por Categoria";
       shdm:context_query "AUCTION::Produto.find_all
           .select{|x| x.auction::quantidade.first<=>0)==1 and
                       x.auction::categoria==categoria}
           .sort_by{|x| x.auction::nomeProduto.first}";
       shdm:context_parameters [
           a shdm:ContextParameter;
                shdm:context_parameter_name "categoria".
        ].
```

Quadro 5.4 Especificação em RDF do contexto "ProdutosPorCategoria"

O Quadro 5.5 apresenta a especificação do índice "ProdutosIdx", que aparece como caixa tracejada com o rótulo "Produtos" na Figura 5.3. Um índice é uma lista de referências para algo, no caso, Produtos.

Nesta especificação, se observa a associação do índice com o contexto "Produtos" através da propriedade shdm:context_index_context para mostrar no índice os elementos deste contexto. Também são especificados os atributos navegacionais do índice, que são os elementos que ocorrem em cada elemento da lista. Estes podem ser do tipo: índice (para mostrar uma lista de elementos em cada atributo do índice principal), âncora para um contexto, âncora para um outro índice ou computado. No exemplo, há um atributo do tipo âncora para contexto e dois atributos do tipo computado.

Os outros índices da aplicação têm uma descrição muito similar à do índice "ProdutosIdx", variando segundo o valor do contexto associado e dos atributos navegacionais, e portanto omitimos suas descrições.

```
:ProdutosIdx a shdm:Context;
    shdm:index_name "ProdutosIdx;
    shdm:index_title "Produtos Idx";
    shdm:context_index_context :Produtos;

shdm:context_anchor_attributes [
    a shdm:ContextAnchorNavigationAttribute;
    shdm:navigation_attribute_name "produto"
    shdm:navigation_attribute_index_position 1;
    shdm:context_anchor_label_expression "self.auction::nomeProduto";
    shdm:context_anchor_target_context : Produtos;
    shdm:context_anchor_target_node_expression "self";
 ];

 shdm:computed_attributes [
     a shdm:ComputedNavigationAttribute;
     shdm:navigation_attribute_name "Esta em oferta";
     shdm:navigation_attribute_index_position 3;
     shdm:computed_attribute_value_expression
       "if self.auction::estaEmLeilao == 1  then
          'sim'
        else
          'nao'
        end".

     a shdm:ComputedNavigationAttribute;
     shdm:navigation_attribute_name "Categoria";
     shdm:navigation_attribute_index_position 2;
     shdm:computed_attribute_value_expression
         "self.auction::categoria.first.auction::nomeCategoria".
 ].
```

Quadro 5.5 Especificação em RDF do índice “ProdutosIdx”

O Quadro 5.6 apresenta a especificação do contexto “LeiloesPorProduto” (“PorProduto” no modelo da Figura 5.3). Este contexto contém o parâmetro “produto”. É possível notar na expressão de consulta o uso do parâmetro “produto”. A expressão de consulta obtém os leilões que foram realizados anteriormente para um tipo de produto dado.

```
:LeiloesPorProduto a shdm:Context;
      shdm:context_name "LeiloesPorProduto";
      shdm:context_title "Leiloes por Produto";
      shdm:context_query "AUCTION::Leilao.find_all
            .select{|x| x.auction::temProduto == produto}";
      shdm:context_parameters [
            a shdm:ContextParameter;
            shdm:context_parameter_name "produto".
      ].
```

Quadro 5.6 Especificação em RDF do contexto “LeiloesPorProduto”

O Quadro 5.7 apresenta a especificação da classe em contexto[10] “ClasseEmContextoProduto” para decorar o nó de um produto no contexto de todos os produtos. No nó, foram acrescidos dois atributos: o primeiro do tipo índice e o segundo do tipo computado. O primeiro tem associado o índice “LeiloesPorProdutoIdx” para mostrar os leilões anteriores desse tipo de produto. Este atributo índice tem também definido um parâmetro (“produto”) cujo valor “self” se refere ao próprio nó atual para a filtragem de seus leilões anteriores.

[10] Classe em Contexto – classe especial que decora os nós, permite que um mesmo nó tenha uma aparência diferente, e apresente âncoras e funcionalidades distintas, quando seus objetos são apresentados em um contexto específico

```
:ClasseEmContextoProduto a shdm:InContextClass;
      shdm:in_context_class_class auction:Produto;
      shdm:in_context_class_context :Produtos;
      shdm:in_context_class_index_attributes [
            a shdm:NavigationAttribute;
            shdm:navigation_attribute_name "Leiloes anteriores";
            shdm:index_navigation_attribute_index :LeiloesPorProdutoIdx;
            shdm:index_navigation_attribute_index_parameters [
                  a shdm:NavigationAttributeParameter;
                  shdm:navigation_attribute_parameter_name "produto";
                  shdm:navigation_attribute_parameter_value_expression
                        "self" .
            ] .
      ];
      shdm:in_context_class_computed_attributes [
            a shdm:ComputedNavigationAttribute;
            shdm:navigation_attribute_name "Categoria";
            shdm:computed_attribute_value_expression
            "self.auction::categoria.first.auction::nomeCategoria" .
      ] .
```

Quadro 5.7 Especificação em RDF da classe em contexto "ClasseEmContextoProduto"

Para o cenário exemplo abordado o modelo gerado pelo assistente é o mesmo gerado pelo Synth (ilustrado nos quadros Quadro 5.3, Quadro 5.4, Quadro 5.5, Quadro 5.6, Quadro 5.7) ao cadastrar as especificações das estruturas SHDM.

6
Avaliação

Neste capítulo se apresenta uma avaliação do Assistente.

O objetivo do trabalho é oferecer uma solução para o problema do projeto de navegação das aplicações hipermídia, através da criação do assistente. Assim, pretendemos responder à pergunta ou hipótese formulada no início, a saber: uma abordagem dirigida por modelos com caraterísticas de assistente facilita o projeto de navegação das aplicações?. Para testar esta hipótese tem-se que avaliar quanto a ferramenta auxilia de fato e para isso consideramos os tipos de usuários que vão usá-la e como medir para cada um quão amigável ela é, o grau de expressividade, o tempo requerido e se eles alcançam o resultado desejado.

A abordagem adotada é pedir para os usuários fazerem uma aplicação com a ferramenta para determinados cenários e resumir os dados coletados.

Num segundo momento compara-se o desempenho do usuário ao criar uma aplicação para esses cenários, primeiro em um ambiente que lhe seja familiar, com aquele gerado por meio do assistente. A conclusão é que a ferramenta facilita na verdade se há um melhor desempenho com o assistente que com o outro framework. Idealmente, este seria a melhor forma de avaliação mas requer tempo e que seja factível a comparação entre frameworks considerando que sejam realmente comparáveis.

Um fator complicador é que os *frameworks* disponíveis ou mais usados pelos usuários não gerenciam a parte da interface, como faz o Synth e a ferramenta proposta. Neste caso não se estaria em igualdade de condições e ter-se-ia que conceber a forma de avaliação adequada pois é evidente que nestes casos o usuário teria que trabalhar mais para obter o mesmo resultado. Dito de outra forma, tem-se que avaliar apenas o desempenho do usuário no aspecto de projeto da navegação, sem considerar os aspectos da interface sempre que for possível.

Para definir os método e artefatos usados para coletar os dados dos experimentos primeiro caracterizamos os tipos de usuários e planejamos as atividades e perguntas em função deles. Uma vez distinguidos os tipos de

usuários para os quais a ferramenta foi destinada, procuram-se indivíduos representativos da população para a qual se quer fazer a generalização.

6.1. Tipos de usuários

O assistente não é destinado a qualquer usuário, requer-se pelo menos que ele tenha conhecimento de aplicações, que entenda do domínio e que seja capaz, desde o ponto de vista intelectual, de especificar uma navegação. Este requerimento se deve ao fato que um usuário com essas caraterísticas mínimas, embora não seja desenvolvedor, já pode por exemplo fazer uma descrição de cenários, ainda que não use os diagrama de interação do usuário[11] (UID). Cabe observar que há um paralelo próximo entre a maneira como o assistente funciona e como se faz um caso de uso no UID. No fundo o que o usuário faz é informar ao assistente os passos que ele descreve em um cenário de uso. Em outras palavras, é quase como se ele estivesse especificando o UID, só que usando um sistema final, não uma notação abstrata.

Assim, os perfis de usuários identificados são:

1. O desenvolvedor que conhece SHDM
2. O desenvolvedor que não conhece SHDM
3. O usuário leigo: Um usuário que não é desenvolvedor, mas conhece um pouco do domínio e é qualificado pelo menos para especificar um caso de uso.

6.2. Variáveis a medir

O segundo aspecto analisado é como medir o desempenho do assistente, que pode ser testado definindo-se as métricas: facilidade de uso, expressividade, tempo de desenvolvimento, grau de sucesso na tarefa.

Para a facilidade de uso se definiu uma escala e se pediu ao usuário atribuir um valor nessa escala ao finalizar o teste. Além disso, se observou durante o teste se o usuário podia explorar alternativas muito mais facilmente do que se pode com o Synth.

[11] Os diagramas de interação do usuário (UIDs) representam graficamente as interações entre o usuário e a aplicação. Um UID é composto por um conjunto de estados de interação conectados através de transições.

Outro aspecto a observar é se o usuário precisou mudar escolhas, e como o fez: refazendo tudo ou mantendo escolhas intermediárias já feitas. O esforço de refazer tem que ser proporcional à mudança em si. Uma pequena mudança deve requerer um pequeno esforço.

O tempo de desenvolvimento avalia-se comparando-o com o tempo de desenvolver as mesmas tarefas no Synth. Se tem os tempos de referencia a priori obtidos a partir da execução dos cenários exemplos no Synth.

Para saber se o usuário conseguia ou não terminar os objetivos do teste definiram-se níveis independentes com tarefas que aumentam em complexidade. O grau de completude então está dado pelo nível cumprido satisfatoriamente.

A abrangência do assistente quanto ao método SHDM é garantida pelo fato de ele gerar interações para todos os elementos do metamodelo, permitindo a sua instanciação com a mesma expressividade (em relação as expressões computacionais aceitas) que se tem ao se inserir a especificação na interface de autoria (direta) do Synth. Desta forma, podemos afirmar que qualquer modelo de navegação gerado diretamente no Synth também pode ser gerado utilizando-se o assistente.

6.3. Cenário exemplo

Os cenários dos testes foram definidos a partir da ideia de criar uma aplicação para fornecer uma agenda eletrônica que exiba os eventos de uma conferência. A descrição deste domínio e o esquema da ontologia encontra-se no Apêndice. Os cenários escolhidos respondem aos critérios expostos previamente.

Para garantir uma complexidade mínima no exemplo, e permitir avaliação de tarefas mais complexas, foram incluidas sutilezas nos requisitos, tais como o fato dos nós de navegação, dependendo do contexto em que se apresentam, exibirem ou não determinados atributos. Esta questão foi considerada no Cenário 1 do Apêndice. Por exemplo, no caso de estudo, suponhamos que se tenham dois contextos, “Autor por Instituição” e outro contexto “Autor por Artigo”. No caso de “Autores por Instituição”, por exemplo da PUC, não tem sentido incluir na descrição de cada Autor um atributo “Instituição”, visto que todos os autores exibidos são da PUC. Já quando se trata de “Autor por Artigo”, torna-se necessária a inclusão do atributo “Instituição”, pois um artigo pode ter autores de instituições distintas. Esta possibilidade de especificar atributos de classe em

contexto é um conceito da abordagem que não é tão comum encontrar com outros frameworks.

6.4. Método de teste

O método de avaliação consiste em inicialmente fazer um pequeno nivelamento com o usuário escolhido. Neste nivelamento explicam-se os principais conceitos da terminologia, sobretudo para os não familiarizados com a metodologia. Isto inclui definir índices, detalhes e aqueles termos relevantes para especificar uma navegação, sem aprofundar com termos que poderiam ser desconhecidos, tais como “contexto”, “classes em contexto”, etc. Pelas próprias perguntas do diálogo da ferramenta estas estruturas SHDM são inferidas, sem que o usuário preciso conhece-los explicitamente. Estes conceitos podem ser reforçados com exemplos.

Em seguida são explicados para o usuário os cenários de interesse e pede-se para ele construir um sistema que implemente esses cenários. Durante a implementação segue-se o método *think aloud* para que a medida que a pessoa vá fazendo ela vá falando o que pensar, permitindo-se assim acompanhar e registrar suas observações e dificuldades. Eventualmente se houver tempo e o usuário conhecer outro framework comparável pede-se para ele fazer nesse outro ambiente e ter uma retroalimentação da comparação. No caso dos usuários conhecedores da metodologia SHDM pede-se para eles comparar diretamente com o uso do Synth.

6.4.1. Coleta de informação do usuário

Ao final, para coletar a opinião do usuário além da informação obtida através do método *think aloud* se faz uma entrevista com perguntas formuladas segundo o tipo de usuário e o procedimento seguido na aplicação do test. Estes dados são interpretados para em função de uma distribuição previamente definida para as variáveis consideradas determinar se esta é normal e obter uma medida do desempenho do assistente.

6.5.Execução do teste

O indivíduo selecionado para avaliar a abordagem foi um desenvolvedor que não conhece a metodologia SHDM. Este usuário tem experiência desenvolvendo aplicações web com outros *frameworks* mas nenhum deles resolvem a parte da interface quanto o Synth e a ferramenta sendo testada. Por isso, só se pediu para ele desenvolver os cenários do teste usando o assistente proposto e o Synth e comparar estes com a experiência dele com outros frameworks. Para o uso do Synth foi preciso fazer um nivelamento para introduzir ao usuário nos principais conceitos da metodologia SHDM.

O usuário manifestou que a ferramenta foi amigável, de fácil uso, na escala definida avaliou em 3, sendo um indicador favorável a nosso propósito. As tarefas do cenário definido foram finalizadas quase na totalidade, só faltando funcionalidades que o Synth não permite, e, portanto, ficam fora do escopo da ferramenta. Ou seja, tudo o que ele conseguiria fazer no Synth pôde também fazê-lo no assistente.

A opinião do usuário foi que a possibilidade de mudar escolhas anteriores não é totalmente intuitiva embora a aplicação o permita. Foi fácil mudar escolhas recentes, mas não ficou tão evidente quando teve que fazer mudanças em outra estrutura navegacional previamente definida. Nestes casos, nem sempre a tela apresentada na seleção de uma estrutura já especificada foi a que o usuário imaginou, requerendo fazer outros passos para chegar na tela desejada. Mesmo assim, se comprovou que o usuário pode fazer mudanças sem ter que refazer tudo.

Outra observação do usuário foi a limitação de não poder reusar estruturas navegacionais já definidas. Além disso, ele considera que fazer atributos computados fica tão difícil quanto no Synth, pela necessidade de conhecer a sintaxes.

A avaliação do tempo requerido foi muito positiva. O usuário levou um tempo muito menor usando o assistente que com o Synth. Além disso, ele afirmou que ambos ambientes permitem o desenvolvimento dos cenários propostos em menos tempo que com os frameworks conhecidos por ele. Com o Synth o usuário teve mais dificuldades porque mesmo com o nivelamento ele não sempre conseguia traduzir os conceitos e suas necessidade nos modelos do Synth.

As recomendações foram conseguir um desenho no visual um pouco mais intuitivo para mudar escolhas intermediarias, e oferecer algum mecanismo para edição de consultas.

6.6. Concluções da avaliação

Os resultados do teste ajudou a concluir que a abordagem dirigida por modelos com carateristicas de assistente facilita o projeto de navegação de aplicações por quanto a facilidade do uso, expressividade, tempo de desenvolvimento e grau de sucesso da tarefa foram todas melhoradas com relação ao Synth.

7
Conclusão e Trabalhos Futuros

A justificativa desta abordagem é obter uma aplicação com aspectos de programação por exemplo e manipulação direta para gerar a parte de navegação e uma visualização na interface. Assim, se concebeu uma estratégia o mais intuitiva possível para definir uma entrada e uma outra estratégia para processar o *feedback*. Ou seja, se objetivou a generalização ao traduzir a interação do usuário através de um diálogo para o modelo de navegação subjacente.

7.1.
Contribuições

As contribuições deste trabalho são listadas a seguir:

- Um assistente dirigido por modelos para desenvolver aplicações hipermídia: o ambiente permite de forma intuitiva, interativa e por manipulação direta a especificação de protótipos. Se aproxima a construção da aplicação ao domínio específico sendo abordado, o seja, ao processo de pensamento dos usuários finais.
- Visão navegacional na aplicação resultante: ao fazer uso do *framework* do Synth para especificação e persistência dos modelos do método SHDM, a aplicação final tem uma visão navegacional sobre um modelo específico de domínio.

7.2.
Trabalhos futuros

Existe um aspecto na geração do assistente que não foi abordada, mais é uma das principais conclusões após o trabalho feito. O assistente poderia utilizar um metamodelo e a partir dele gerar o diálogo e opções do assistente, como já se faz com o modelo SHDM para criar aplicações hipermídia.

A ideia é a partir das metaclasses navegacionais: índice, contexto, etc., gerar programaticamente a estrutura de diálogo correspondente para instanciar este metamodelo. Assim, se houver uma mudança nas metaclasses o assistente automaticamente geraria novas perguntas.

A estrutura do assistente é fixa no momento, o que permitiu uma versão específica para começar a analisar qual poderia ser o metamodelo do assistente para gerar as perguntas, opções, exemplos, e como estes se associam com o outro modelo a ser instanciado.

O usuário não pode fazer qualquer especificação, ele apenas segue que o sistema é capaz de fazer. Dado o metamodelo do SHDM, especificamente o de navegação, todas as perguntas que o assistente faz são determinadas por ele. Isso está embutido no código mas poderia ser feito genericamente se se tiver a descrição completa do metamodelo do assistente. E por último, dado um modelo qualquer (i.e. um modelo ER) se poderia gerar a partir do metamodelo do assistente o diálogo correspondente.

Além do assistente utilizar o metamodelo diretamente, a própria estrutura de diálogo poderia também ser dirigida por modelos. Para tanto, seria necessário definir uma ontologia de diálogos, de maneira similar a como se criam agora as aplicações hipermídia a partir do metamodelo SHDM.

Um outro uso do assistente é permitir o reuso de experiência aplicado a projetos de aplicações hipermídia. Ao longo do tempo, pode-se coletar exemplos em uma biblioteca, com exemplos que constituem tarefas frequentes em diferentes domínios. Ao iniciar um novo projeto, o desenvolvedor poderia acionar o assistente a partir de um projeto anterior já realizado, e ir acrescentando alternativas ou alterando decisões realizadas no exemplo da biblioteca.

Uma extensão interessante consiste em definir um mecanismo para a edição de frases em linguagem pseudo-natural que ajude nos casos que o usuário tem que formular expressões, escritas presentemente na linguagem de programação Ruby ou SPARQL.

Referências Bibliográficas

Araujo, S., Schwabe, D., & Barbosa, S. (2009). **Experimenting with explorator: a direct manipulation generic rdf browser and querying tool**. *In Workshop on Visual Interfaces to the Social and the Semantic Web (VISSW2009).*

Barbosa do Nascimento, V. (2013). Modelagem e geração de interfaces. PUC-Rio.

Berners-Lee, T., Hollenbach, J., Lu, K., Presbrey, J., Prud'ommeaux, E., & Schraefel, M. (2008). **Tabulator Redux: Browsing and writing Linked Data**. *In Proc. WWW 2008 Workshop: LDOW.*

Bomfim, M. (2011). **Um método e um ambiente para o desenvolvimento de aplicações na Web Semântica**. *Dissertação de mestrado.* Departamento de Informática – PUC–Rio.

Hearst, M., English, J., Sinha, R., Swearingen, K., & Yee, P. (2002). **Finding the Flow in Web Site Search**. *Communications of the ACM, 45 (9)*, (pp. 42-49).

Heath, T., & Bizer, C. (2011). ***Linked Data: Evolving the Web into a Global Data Space***. Morgan & Claypool.

Heim, P., Ertl, T., & Ziegler, J. (2010). **Facet graphs: Complex semantic querying made easy**. *In Proc. 7th Extended Semantic Web Conference (ESWC 2010)* (pp. 288-302, volume 6088 of LNCS). Berlin/Heidelberg: Springer.

Hildebrand, M., Ossenbruggen, J., & Hardman, L. (2006). /**facet: A browser for heterogeneous semantic web repositories**. *In International Semantic Web Conference*, (pp. 272-285).

Huynh, D. (2009). ***Nested Faceted Browser***. Fonte: http://people.csail.mit.edu/dfhuynh/projects/nfb/

Huynh, D., & Karger, D. (2009). **Parallax and companion: Set-based browsing for the Data Web**.

Huynh, D., Karger, D., & Miller, R. (2007). **Exhibit: lightweight structured data publishing**. *In Proc. WWW '07: Proceedings of the 16th international conference on World Wide Web* (pp. 737-746). New York, NY, USA: ACM.

Karger, D., Ostler, S., & Lee, R. (2009). **The web page as a wysiwyg end-user customizable database-backed information management application**. *In Proc. UIST '09* (pp. 257-260). New York, NY, USA: ACM.

Kimono. (2014). Fonte: www.kimonolabs.com

Kobilarov, G., & Dickinson, I. (2008). Humboldt: **Exploring Linked Data**. *In Proc. WWW 2008 Workshop: LDOW.*

Lima, F. (2003). **Modelagem semântica de aplicações na WWW**. *Tese de Doutorado.* PUC-Rio.

Longwell. (2005). ***RDF Browser, SIMILE***. Fonte: http://simile.mit.edu/longwell/

Luna, A. (2009). **Geração de Interfaces RIA Dirigida por Ontologias**. *Dissertação de Mestrado.* PUC-Rio.

Nunes, D. (2005). **HyperDE - um Framework e Ambiente de Desenvolvimento dirigido por Ontologias para Aplicações Hipermídia**. *Dissertação de Mestrado.* PUC-Rio.

Popov, I., Schraefel, M., Hall, W., & Shadbolt, N. (2011). **Connecting the Dots: A Multi-pivot Approach to Data Exploration**. *In Proc. ISWC.*

Quan, D., Huynh, D., & Karger, D. (2003). **Haystack: A Platform for Authoring End User Semantic Web Applications**. *In Proc. ISWC 2003, LNCS*, (pp. 738-753).

Rossi, G. (1996). **Um método orientado a objetos para o projeto de aplicações hipermidia**. *Tese de Doutorado.* PUC-Rio.

Schraefel, M., Smith, D., Owens, A., Russell, A., Harris, C., & Wilson, M. (2005). **The evolving mSpace platform: Leveraging the Semantic Web on the trail of the memex**. *In Proc. Hypertext 2005,* (pp. 174-183). ACM Press.

Schwabe, D., & Rossi, G. (1998). ***An Object Oriented Approach to Web-Based Application Design***.

Stegemann, T., Hussein, T., Gaulke, W., & Ziegler, J. (2012). **Interacting with Semantic Data by Using X3S**. *In Proc. 4th ACM SIGCHI symposium on Engineering interactive computing systems.*

Szundy, G. (2004). **Modelagem e Implementação de Aplicações Hipermídia Governadas por Ontologias para a Web Semântica**. *Dissertação de Mestrado.* PUC-Rio.

Apêndice

Cenário 1: Ver detalhe de uma publicação

Ator: Usuário comum

A aplicação exibe uma lista de publicações ordenada alfabeticamente e apresenta de cada uma o **título e uma lista com os nomes dos autores**. Eu escolho a publicação de interesse e a aplicação apresenta a **informação detalhada da publicação (título, resumo, mês, ano)**. Além disso, a aplicação exibe uma **lista dos autores dessa publicação.** Aqui se pode navegar à próxima ou anterior publicação do conjunto de todas as publicações. Se eu clicar em um autor na tela de detalhe ou na tela de lista de publicações se exibe a mesma tela detalhe da publicação mas sem a lista de autores com o título **Publicações do autor que foi escolhido**. Aqui se pode navegar à próxima ou anterior publicação do conjunto das publicações do autor escolhido.

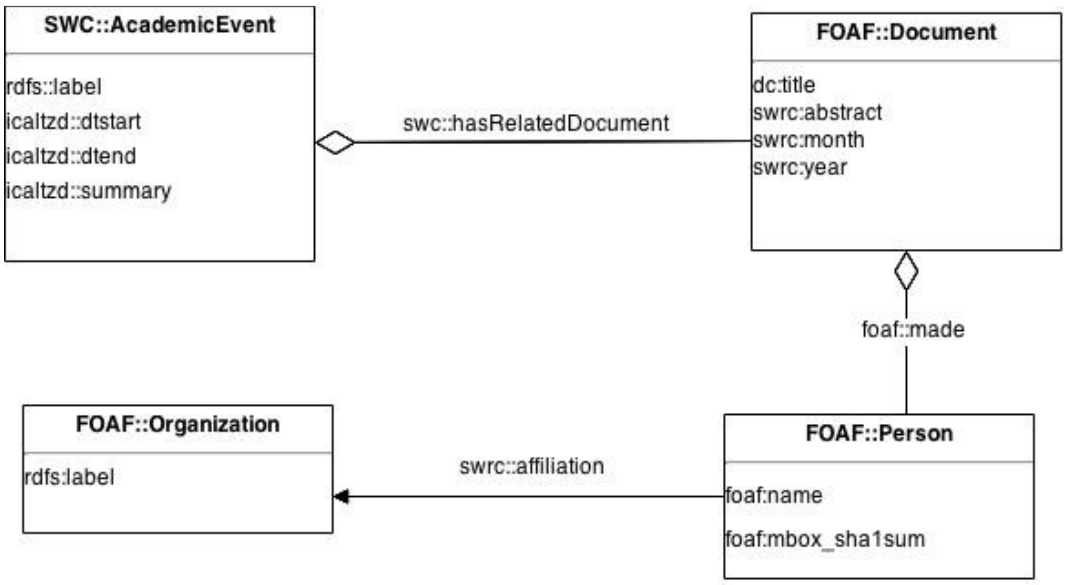
Cenário 2: Ver detalhe de um evento

Ator: Usuário comum

A aplicação exibe uma lista de eventos ordenada pela data e hora de começo e apresenta de cada um o **nome, o campo quando (data, hora de começo, hora de terminação) e local**. Eu escolho o evento de interesse e a aplicação exibe a **informação detalhada do evento (nome, o campo quando (data, hora de começo, hora de terminação), local, resumo)**. Além disso, a aplicação exibe uma **lista de apresentadores e uma lista de organizações.** Aqui se pode navegar ao próximo ou anterior evento do conjunto de todos os eventos. Se eu clicar em um apresentador se exibe a tela detalhe do apresentador com o **nome, código hash associado ao email e uma lista das publicações**. Aqui se pode navegar ao próximo ou anterior apresentador do conjunto dos apresentadores do evento escolhido. Se eu clicar em uma publicação se exibe a tela detalhe da publicação mas sem a lista de autores com

o título **Publicações do apresentador que foi escolhido**. Aqui se pode navegar à próxima ou anterior publicação do conjunto das publicações do apresentador escolhido. No detalhe de um evento, se pode também clicar em uma organização e a aplicação exibe a mesma tela detalhe do evento mas sem a lista de organizações. Aqui se pode navegar ao próximo ou anterior evento da organização escolhida.

Diagrama de classes do domínio Eventos



Perguntas para antes do teste da aplicação

1. Você tem experiência desenvolvendo aplicações?
2. Você conhece a metodologia SHDM?
   2.1. Se conhece a metodologia SHDM, tem experiência desenvolvendo no ambiente Synth?
3. Se tem experiência desenvolvendo aplicações, poderia desenvolver a aplicação do teste usando um framework de sua preferência? (No caso de conhecer o Synth, usar este ambiente)
4. Desenvolva agora usando o assistente

Perguntas para despois do teste da aplicação

1. Que você achou, foi fácil ou difícil usar a ferramenta? Diga em uma escala 1-10 sendo 1 fácil e 10 difícil.

1.1. Conseguiu fazer o que pretendia sem se perder nos detalhes do modelo, especificação de queries e detalhes da implementação do ambiente?

1.2. Quais foram as dificuldades?

2. Se desenvolveu em outro ambiente, diga qual foi mais fácil/amigável.

2.1. Se só usou a ferramenta, você acha que conseguiria fazer nesse mesmo tempo usando seus conhecimentos?

3. Se conhece o Synth ou outro ambiente dirigido por formulários, desenvolver por exemplo é melhor que por especificação o não?

4. Quais são suas recomendações?