UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

RENATO MARTINS BARBIERI NUNES

Validação do framework FPOA através da implementação de um repositório de objetos de aprendizagem no modelo UMBRELO

Trabalho de Graduação.

Prof. Dr. Leandro Krug Wives
Orientador

Profa. Dra. Clevi Elena Rapkiewicz
Coorientadora

Porto Alegre, julho de 2013.

AGRADECIMENTOS

Agradeço primeiramente à minha família, principalmente aos meus pais e a minha noiva, que nunca deixaram de acreditar em mim e de me incentivar, oferecendo suporte e compreensão nos momentos difíceis e compartilhando da alegria nos momentos de felicidade.

A todos os meus amigos, por estarem sempre ao meu lado, dispostos a ajudar e a dividir não só os momentos bons. Além de compreenderem a ausência devido aos trabalhos.

Ao professor Leandro e a professora Clevi pela orientação, suporte e ajuda que permitiram chegar ao objetivo deste trabalho.

À Núbia e ao Rogério pela colaboração neste trabalho.

E a todos que contribuíram direta ou indiretamente na conclusão de mais esta etapa da minha vida.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

API	Application Program Interface
BIOE	Banco Internacional de Objetos Educacionais
CESTA	Coletânea de Entidades de Suporte ao uso de Tecnologia na Aprendizagem
DAO	Data Acess Object
FPOA	Framework de Persistência de Objetos de Aprendizagem
HTTP	HyperText Transfer Protocol
IDE	Integrated Development Environment
LabVirt	Laboratório Didático Virtual
LOM	Learning Object Metadata
MVC	Model-View-Controller
OA	Objeto de Aprendizagem
ROA	Repositório de Objetos de Aprendizagem
SBIE	Simpósio Brasileiro de Informática na Educação
SGBD	Sistema de Gerenciamento de Banco de Dados
SQL	Structured Query Language
UMBRELO	Ubiquitous Metadata based Reusable Learning Objects
WIE	Workshop de Informática na Educação
XML	eXtensible Markup Language
XHTML	eXtensible Hypertext Markup Language

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Diversos recursos digitais vêm sendo utilizados no processo de ensino. Nesse contexto, destacam-se os objetos de aprendizagem (OA). Diferentes grupos de pessoas desenvolvem OA e, para facilitar sua localização e reuso, eles são disponibilizados em repositórios de objetos de aprendizagem (ROA). Após realizar uma revisão na literatura, verificou-se que os principais ROA do país seguem um procedimento de catálogo manual, sujeita a erros, além de não incluir diversos aspectos relevantes dos objetos, dificultando, assim, o processo de localização dos mesmos. Um dos trabalhos que aborda esse problema propõe uma ferramenta chamada de Framework de Persistência de Objetos de Aprendizagem (FPOA), que é capaz de catalogar OA levando em conta qualquer conteúdo textual ou metadado que esses possuam. O propósito deste trabalho consiste em adaptar o referido framework de maneira a que ele também inclua o UMBRELO, que é um modelo para descrever objetos educacionais desenvolvidos em Flash proposto pelo grupo de pesquisa onde este trabalho está inserido. Além disso, criou-se um ROA que utiliza essa adaptação visando sua validação.

Palavras-Chave: repositório digital, Objetos de Aprendizagem, catalogação, persistência.

Validation the FPOA framework through the implementation of a learning objects repository using the UMBRELO model

ABSTRACT

Several digital resources are being used in the teaching process. In this context, stand out the learning objects (LO). Different groups of people develop LO. Thus, in order to facilitate their localization and reuse, LO are made available in learning object repositories (LOR). After performing a literature review, it was verified that the main LOR in Brazil follow a manual cataloguing procedure, which may be subject to errors, and does not include many relevant aspects of the LO, hardening their localization process. One of the works that adresses this problem proposes a tool called Persistency Framework of Learning Objects (FPOA, in Portuguese), which is capable of cataloging LO in a way that takes into account any textual content or metadata that they may possess. The purpose of this work is to adapt the FPOA framework such that it also includes UMBRELO, which is a model to describe educational objects developed in Flash proposed by the research group where this work is inserted. In addition, a LOR who uses this adaptation was developed in order to validate the proposed adaption.

Keywords: digital repository, Learning Object, catalog, persistence.

1 INTRODUÇÃO

As Tecnologias de Informação e Comunicação (TIC) são elementos fundamentais para a difusão de informação, tanto para fins de entretenimento como educacionais. Na educação, essa difusão pode ser alcançada por meio do uso e do desenvolvimento de objetos de aprendizagem (OA). Esse termo refere-se a materiais educacionais projetados e construídos a partir de pequenos componentes visando a sua reutilização em diferentes contextos de aprendizagem (WILEY, 2002). Cabe salientar que os OA são compostos de recursos da própria plataforma de desenvolvimento e de demais mídias, como: som, imagem, vídeo, etc.

Para facilitar sua localização e reuso, os objetos de aprendizagem são disponibilizados em repositórios de objetos de aprendizagem, também denominados de ROA (NUNES, 2004). Os ROA são bancos que armazenam e catalogam objetos educacionais e seus metadados. Porém, para que os conteúdos dos OA possam ser adequadamente recuperados quando necessário, eles devem ser corretamente catalogados. No entanto, Cordeiro (2009) verificou que os principais ROA do Brasil seguem um procedimento de catálogo manual, sujeito a erros, além de não incluir diversos aspectos relevantes dos objetos (como, por exemplo, seuconteúdo textual), dificultando, assim, seu processo de localização. Com o propósito de minimizar esse problema, Cordeiro desenvolveu uma ferramenta, nomeada de Framework de Persistência de Objetos de Aprendizagem (FPOA), que realiza a catalogação de OA de forma a permitir a sua recuperação por meio de todo e qualquer conteúdo, mesmo na forma textual ou metadado que um OA possua.

No entanto, após 2009 foram desenvolvidos novos modelos para a estruturação e o armazenamento de informações sobre OA. Um desses modelos é o UMBRELO. Tal modelo é capaz de descrever a estrutura interna de um OA, incluindo seus elementos textuais e suas relações (SANTOS et al., 2012).

Os objetos que utilizam esse modelo são desenvolvidos com a tecnologia Flash. Sendo assim, tais objetos são organizados de forma que a estrutura visual e o controle de interação são compilados em um arquivo separado de metadados, que descreve os elementos do arquivo Flash, estando em formato XML. Contudo, o arquivo compilado é dependente dos arquivos de descrição, pois todas as informações que o objeto deve exibir estão contidas no segundo arquivo, cujos detalhes são abordados na seção 3.2.

Neste trabalho é proposta a implementação de um ROA utilizando o FPOA como base (i.e., como ferramenta de catalogação e persistência), mas adaptando-o de maneira a suportar objetos desenvolvidos no modelo UMBRELO. Essa adaptação será realizada pois o modelo referido contempla os requisitos necessários para se obter o potencial máximo do FPOA.

O trabalho está organizado em cinco capítulos. No segundo capítulo é descrita uma análise dos trabalhos correlatos, uma análise da catalogação e dos mecanismos de busca dos principais repositórios de OA nacionais e uma breve descrição da implementação do FPOA. O terceiro capítulo aborda as questões relativas ao projeto do sistema, tais como: análise de requisitos, definição de componentes e da arquitetura utilizada. Em seguida, no quarto capítulo, a implementação do sistema é detalhada, contemplando a modelagem do banco de dados, a organização das classes dentro da solução e análise de navegação nas interfaces desenvolvidas. Por fim, no quinto e último capítulo, são avaliados os resultados obtidos, bem como as dificuldades encontradas durante o trabalho e as sugestões para trabalhos futuros.

2 TRABALHOS RELACIONADOS

Este capítulo aborda os trabalhos relacionados ao estudo, realizando uma análise das tendências dos mesmos através de técnicas de mineração de dados. Além disso, são analisados os processos de catalogação e de busca dos repositórios nacionais com o objetivo de verificar se os problemas apontados persistem e, por fim, é apresentada uma breve descrição do framework FPOA, contemplando seus requisitos, funcionalidades e arquitetura.

2.1 Metodologia

A pesquisa por trabalhos relacionados seguiu os seguintes passos:

1. Levantamento bibliográfico de artigos publicados nos últimos quatro anos mencionando o problema de localização e reuso de OA em repositórios. Para tal, foram escolhidos os dois principais anais de eventos que envolvem informática na educação: o Simpósio Brasileiro de Informática na Educação (SBIE) e o Workshop de Informática na Escola (WIE). O resultado pode ser observado na Tabela a seguir:

Tabela 2.1: Artigos relacionados ao trabalho nos últimos 4 anos.

ANO	Congresso	Total de artigos	Artigos sobre OA	Relacionados ao Trabalho
2009	SBIE	84	11	1
	WIE	37	7	0
2010	SBIE	135	21	4
	WIE	44	5	1
2011	SBIE	138	20	0
	WIE	37	4	0
2012	SBIE	131	26	3
	WIE	48	6	0

Fonte: Elaborada pelo autor.

2. Elaboração de um corpus para análise contendo o conteúdo de todos os artigos selecionados.

3. Mineração desse corpus utilizando o software WordCounter.

O software WordCounter é uma ferramenta online gratuita que apresenta a relação

das palavras mais utilizadas em um texto. Para os autores de textos ele é útil, pois mostra as palavras repetidas e/ou redundantes numa lista, juntamente com a frequência com que cada uma aparece no texto. Entretanto, esse programa não dispõe de ferramentas gráficas mais complexas para visualização das informações (KLEMANN et al., 2011).

A mineração de textos permite recuperar informações, extrair dados, resumir documentos, descobrir padrões, dentre outras análises possíveis de se realizar em documentos de texto. Pode ser utilizada com muitos propósitos, como por exemplo identificar documentos similares entre si, buscar dados relevantes dentro do documento, entre outras. (KLEMANN et al., 2011). Neste trabalho, a mineração de textos foi realizada com o objetivo de identificar quais os principais direcionamentos das pesquisas e soluções relacionadas ao problema de localização de OA em repositórios.

2.2 Resultado da mineração de artigos

O resultado da análise dos artigos utilizando a ferramenta de mineração online WordCounter pode ser acompanhado na tabela a seguir, onde as palavras foram agrupadas por similaridade:

Tabela 2.2: Termos recorrentes na mineração usando o WordCounter.

Palavras	**Frequência**
metadados	346
oa(s)	246
objeto(s)	228
ontologia(s)	219
aprendizagem	157
conhecimento(s)	147
padrão(ões)	117
repositório(s)	115
processo	100
recurso(s)	94
dados	86
busca	81
elemento(s)	73
informações	73
alinhamento	62

Fonte: Elaborada pelo autor.

Analisando-se a tabela decorrente do uso do WordCounter (Tabela 2.2), pode-se perceber o destaque para a palavra metadados. Isso ocorre pois os trabalhos analisados não buscam soluções para o problema de localização em algum repositório específico e sim para o problema de localização de OA em conjuntos de repositórios, propondo

soluções que envolvem somente os metadados dos objetos. Nenhum dos trabalhos encontrados foca no problema de localização de OA através de qualquer conteúdo que esse possua, sendo esse o principal diferencial da solução proposta neste trabalho.

2.3 Análise dos Repositórios de OA

A análise dos repositórios consistiu em verificar a eficiência de seus mecanismos de busca em pesquisas simples que envolvessem conteúdos textuais dos OA bem como a sua forma de catalogação (manual ou automatizada), a fim de averiguar se os problemas constatados por Cordeiro (2009) foram solucionados ou ainda persistem.

Foram analisados os repositórios de OA que mais se destacam no país: o Banco Internacional de Objetos Educacionais (BIOE), o Laboratório Didático Virtual (LabVirt) e o Coletânea de Entidades de Suporte ao uso de Tecnologia na Aprendizagem (CESTA). A Tabela 2.3 demonstra as principais informações sobre cada um.

Tabela 2.3: Repositórios de OA nacionais.

Nome	Link	Instituição Mantenedora	Descrição
BIOE	http://objetoseducacionais2.mec.gov.br	MEC – Ministério da Educação	Repositório com diversos tipos de recursos educacionais, tal como vídeo, som, animação, simulação, etc.
LabVirt	http://www.labvirtq.fe.usp.br/	USP – Universidade de São Paulo	Repositório com diversos tipos de recursos educacionais incluindo artigos, discussões, fórum, etc.
CESTA	http://cesta.cinted.ufrgs.br/	UFRGS – Universidade Federal do Rio Grande do Sul	Repositório com diversos recursos multimídia mas necessita de autenticação (login e senha) para a navegação.

Fonte: Adaptado de Cordeiro (2009).

2.3.1 Banco Internacional de Objetos Educacionais

O Banco Internacional de Objetos Educacionais (BIOE) é um repositório criado em 2008 pelo Ministério da Educação em parceria com diversas organizações educacionais. Ele tem o propósito de manter e compartilhar recursos educacionais digitais de livre acesso, mas elaborados em diferentes formatos - como áudio, vídeo, animação, simulação, software educacional - além de imagem, mapa, hipertexto considerados relevantes e adequados à realidade da comunidade educacional local, respeitando-se as diferenças de língua e culturas regionais (BIOE, 2013).

Para verificar a eficiência dos mecanismos de busca desse repositório, coletou-se um objeto de forma aleatória dentro de seu acervo, identificando algum conteúdo textual

que pudesse ser utilizado para testar tais mecanismos. As Figuras 2.1 e 2.2 demonstram respectivamente uma imagem do objeto selecionado, denominado Usina dos Gases, e o resultado de uma busca pela palavra “vinhoto” contida no objeto no site do BIOE.

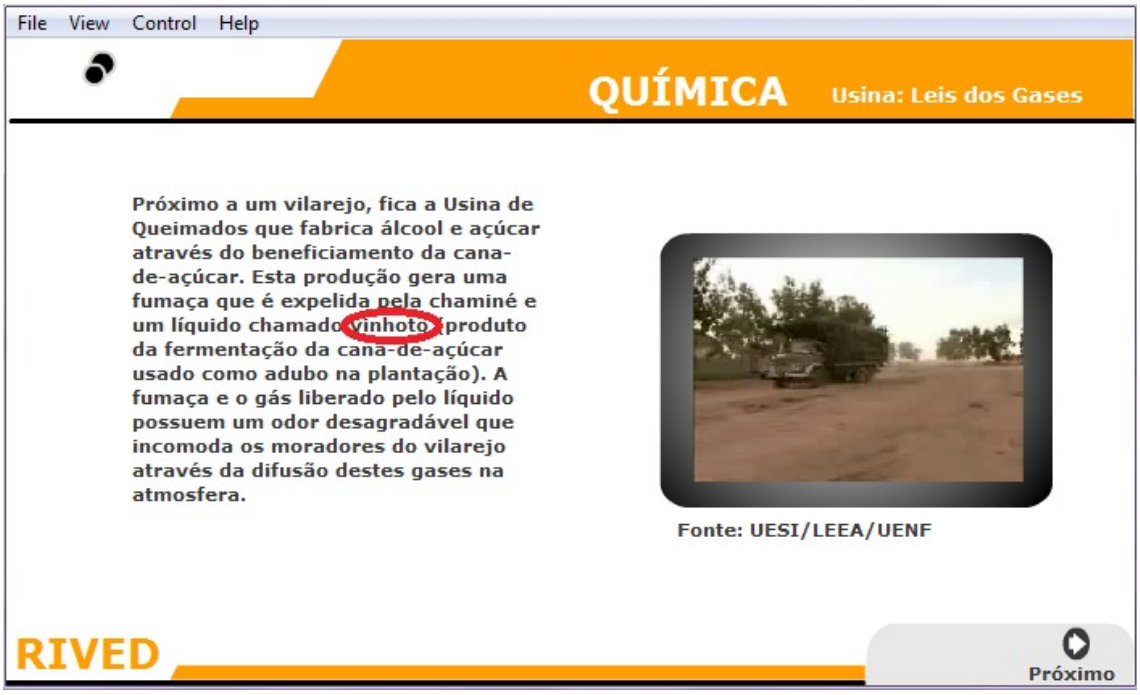

Figura 2.1: Objeto Usina dos Gases. Fonte: Elaborada pelo autor ao utilizar o OA.

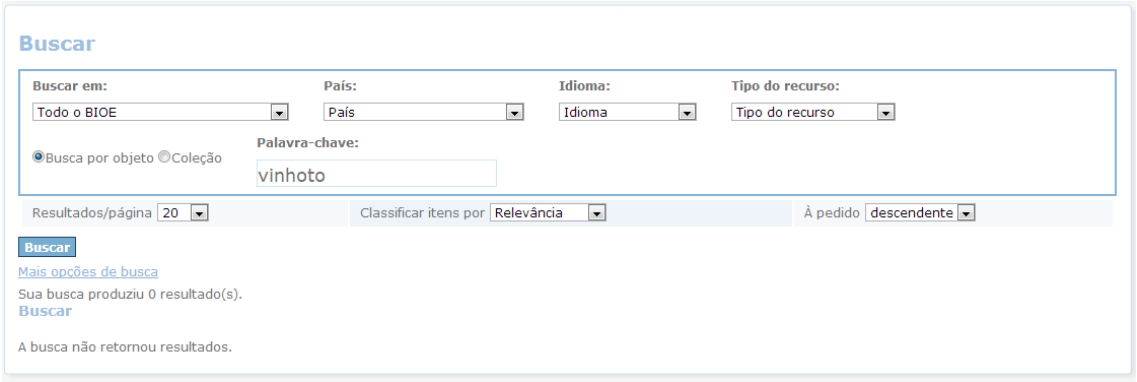

Figura 2.2: Resultado da pesquisa pela palavra “vinhoto” no BIOE. Fonte: Elaborada pelo autor ao utilizar o site.

O resultado da busca não foi satisfatório pois o processo de catalogação desse repositório continua sendo de forma manual e leva em conta somente os metadados de cada objeto, não sendo possível recuperá-los por qualquer palavra que não tenha sido catalogada no momento de seu cadastro. É possível afirmar isso pois, apesar do cadastramento de novos objetos ser somente para usuários autenticados, existe um guia disponível no site que demostra como é feita a catalogação além de quais e como os metadados devem ser preenchidos.

2.3.2 Laboratório Didático Virtual

O Laboratório Didático Virtual (LabVirt) é uma iniciativa da Escola do Futuro da

Universidade de São Paulo cuja meta é aprimorar o aprendizado dos alunos por meio do esforço coletivo de escolas e universidades. Nesse contexto, a construção do conhecimento de cada aluno é feita de uma maneira mais contextualizada, menos fragmentada e mais significativa (LABVIRT, 2013).

Da mesma forma que na análise dos mecanismos de busca do BIOE, coletou-se um objeto de forma aleatório dentre os disponíveis, identificando algum conteúdo textual que pudesse ser utilizado na análise. As Figuras 2.3 e 2.4 demonstram respectivamente uma imagem do objeto Acidez do Vinagre na Salada, coletado do repositório, e o resultado de uma busca pela palavra "acético" contida no objeto no site do LabVirt.

Figura 2.3: Objeto Acidez do Vinagre na Salada. Fonte: Elaborada pelo autor ao utilizar o referido objeto.

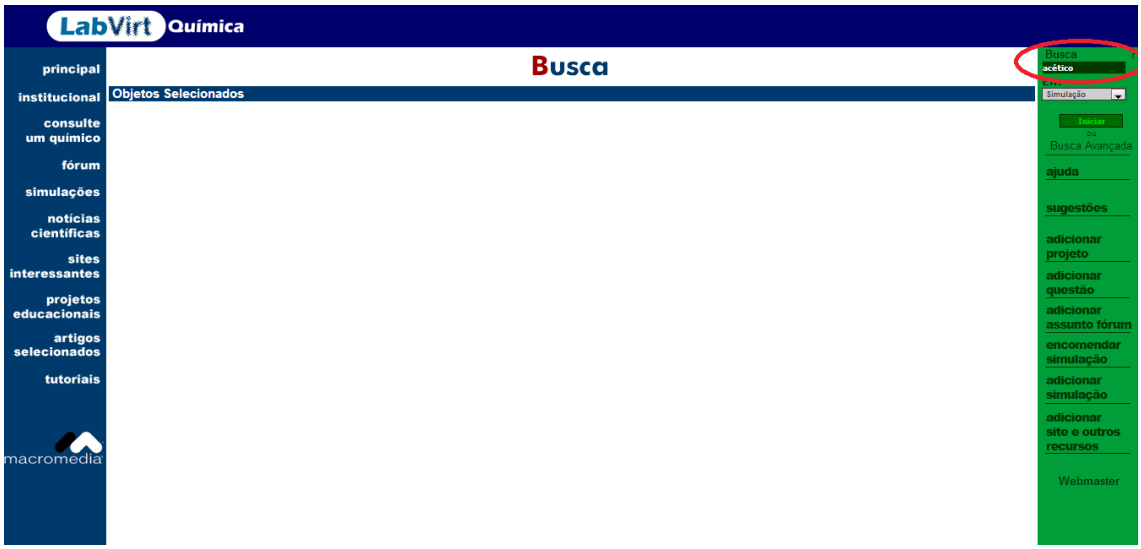

Figura 2.4: Resultado da pesquisa pela palavra "acético" no LabVirt. Fonte: Elaborado pelo autor utilizando o sistema.

Os resultados obtidos pelos mecanismos de busca deste repositório também não se mostraram eficientes quando são efetuadas pesquisas por palavras contidas em OA e não em seus metadados. Isso se deve ao fato da catalogação de OA no LabVirt ser disponibilizada de forma manual e apresentando campos para inserção somente dos metadados dos objetos, conforme mostra a Figura 2.5.

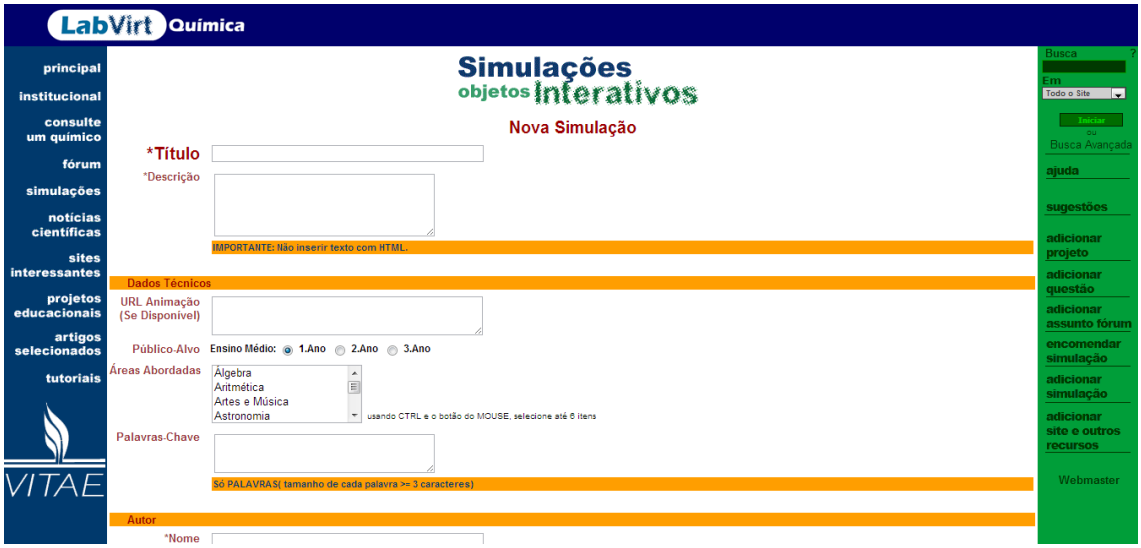

Figura 2.5: Parte do formulário para cadastramento de novo OA no LabVirt. Fonte: Elaborada pelo autor utilizando o sistema.

2.3.3 Coletânea de Entidades de Suporte ao uso de Tecnologia na Aprendizagem

O projeto CESTA (Coletânea de Entidades de Suporte ao uso de Tecnologia na Aprendizagem) foi idealizado com vistas a sistematizar e organizar o registro de objetos educacionais (CESTA, 2013). Esse repositório é uma iniciativa da Pós-Graduação Informática na Educação e do CINTED (Centro Interdisciplinar de Novas Tecnologias na Educação) da UFRGS (Universidade Federal do Rio Grande do Sul).

Dentre os objetos disponibilizados nesse repositório, foi escolhido o OA Bandeja do Cateter de Oxigênio para realizar os testes com o mecanismo de busca do CESTA a fim

de verificar sua eficiência em pesquisas por palavras que não estejam presentes nos metadados dos objetos. As Figuras 2.6 e 2.7 ilustram respectivamente uma tela do objeto com a palavra "oxigenoterapia" destacada e o resultado da pesquisa por essa palavra no repositório.

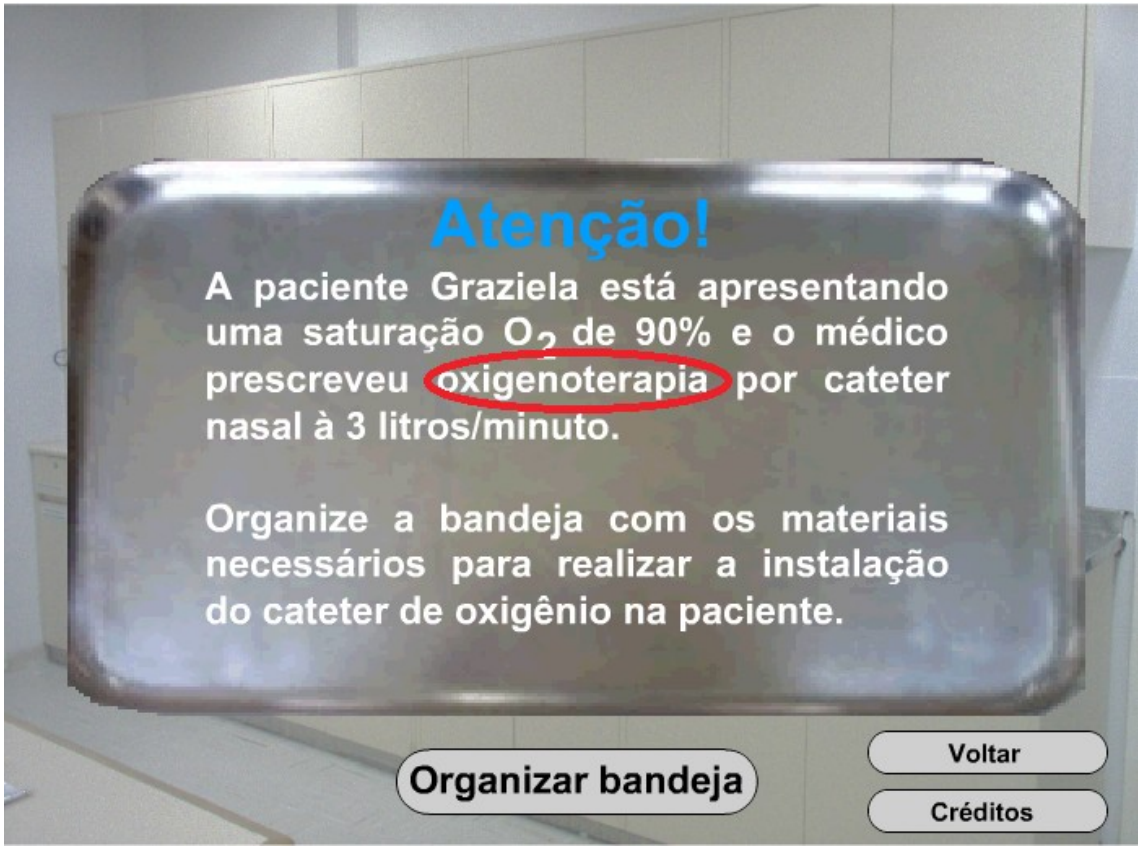

Figura 2.6: Tela do objeto Bandeja do Cateter de Oxigênio. Fonte: Elaborada pelo autor utilizando o objeto.

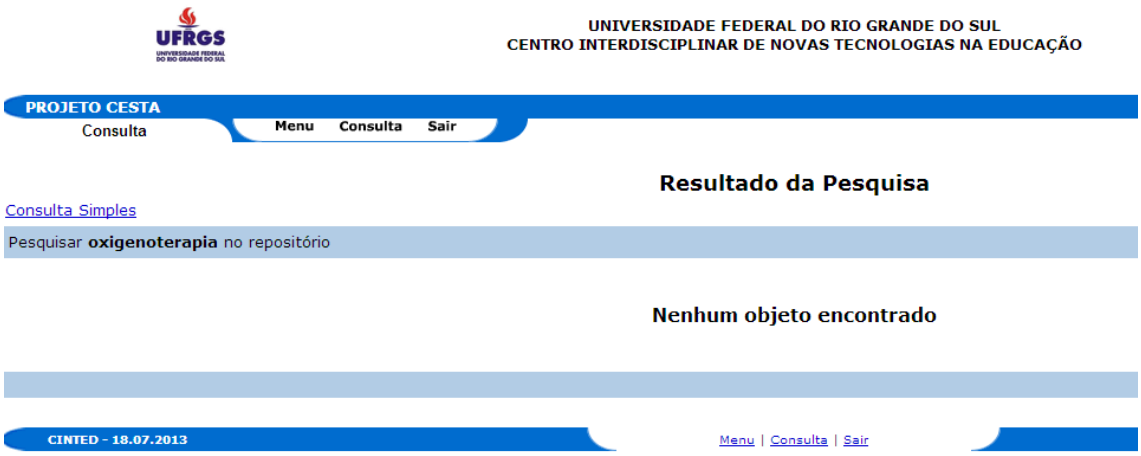

Figura 2.7: Resultado da pesquisa pela palavra "oxigenoterapia" no LabVirt.
Fonte: Elaborada pelo autor utilizando o sistema.

Da mesma forma que nos demais repositórios analisados, o resultado da pesquisa por palavra contida no OA não foi eficiente em virtude desse repositório não demonstrar

alguma diferença na forma de catalogação de seus objetos para os demais analisados. O registro de novos objetos só pode ser efetuado por usuários cadastrados no sistema, porém como mostra a Figura 2.8 tal processo ainda é executado de forma manual e leva em consideração somente os metadados do objeto.

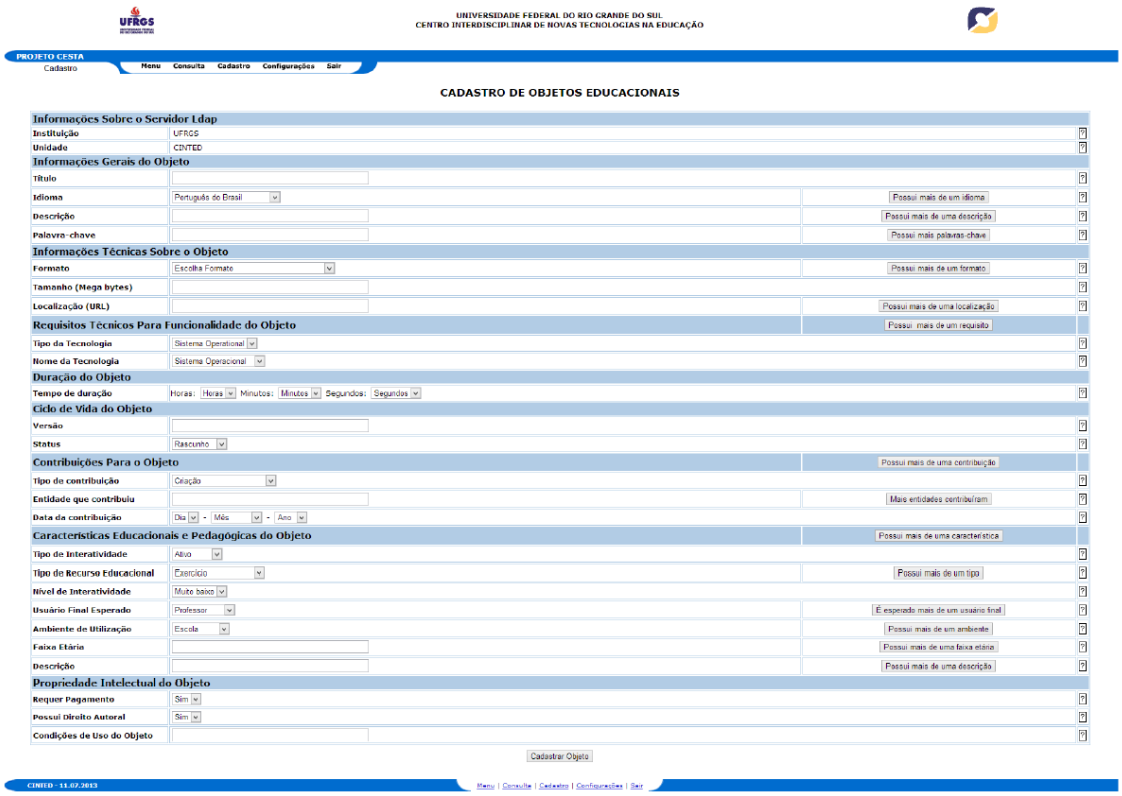

Figura 2.8: Formulário para cadastramento de novo objeto no CESTA.
Fonte: Elaborada pelo autor utilizando o sistema.

2.3.4 Resumo da Análise

Considerando a análise realizada, conclui-se que a forma de catalogação dos principais repositórios de OA nacionais continua a ser de forma manual, levando em conta somente os metadados dos objetos, mantendo o problema na localização de OA através de qualquer conteúdo textual que esse possua. A Tabela 2.4 resume os resultados obtidos na análise.

Tabela 2.4: Resumo da análise dos repositórios.

Repositório	Busca considera qualquer conteúdo textual dos OA	Forma de Catalogação
BIOE	Não	Manual
LabVirt	Não	Manual
CESTA	Não	Manual

Fonte: Elaborada pelo autor.

2.4 FPOA

O framework de persistência FPOA realiza a catalogação de objetos de aprendizagem em repositórios de forma a permitir a recuperação dos mesmos por meio de todo e qualquer conteúdo na forma textual ou metadado que este possua. Essa ferramenta extrai metadados e conteúdo dos objetos, persistindo-os em um Sistema de Gerenciamento de Banco de Dados (SGBD) e permitindo a elaboração de consultas sobre eles (CORDEIRO, 2009).

A motivação que levou a criação dessa ferramenta foi a hipótese de que automatizando o processo de catalogação de OA nos repositórios, por meio de um framework, aumentaria o índice de OA localizados e, consequentemente, seu reuso. Cordeiro (2009) levantou essa hipótese a partir da análise do processo de catalogação e recuperação de OA em diferentes repositórios nacionais, constatando que um dos principais motivos da ineficiência nos seus mecanismos de busca era devido ao fato de todos possuírem um processo de catalogação de forma manual.

As subseções que seguem apresentam os requisitos apontados no desenvolvimento do FPOA e uma breve descrição de sua arquitetura.

2.4.1 Requisitos

Segundo Cordeiro (2009), o FPOA foi desenvolvido visando contemplar os seguintes requisitos:

- persistir e consultar OA independente das tecnologias de SGBD (Sistema de Gerenciamento de Banco de Dados) utilizados;
- validar se o conteúdo textual do OA possui uma estrutura de informação que possibilite a persistência;
- possibilitar a criação/configuração de um banco de dados que suporte a persistência e consulta de OA;
- identificar o recurso no qual a palavra pesquisada é apresentado no objeto: botão, texto, crédito, cena, cenário, etc.

Com base nos requisitos supracitados, concluiu-se que o framework deveria desempenhar 2 atividades básicas: efetuar consultas de OA e persistir OA. O Diagrama de Atividades da Figura 2.1 demonstra os fluxos de controle desempenhado pelo FPOA para cada uma das funcionalidades citadas.

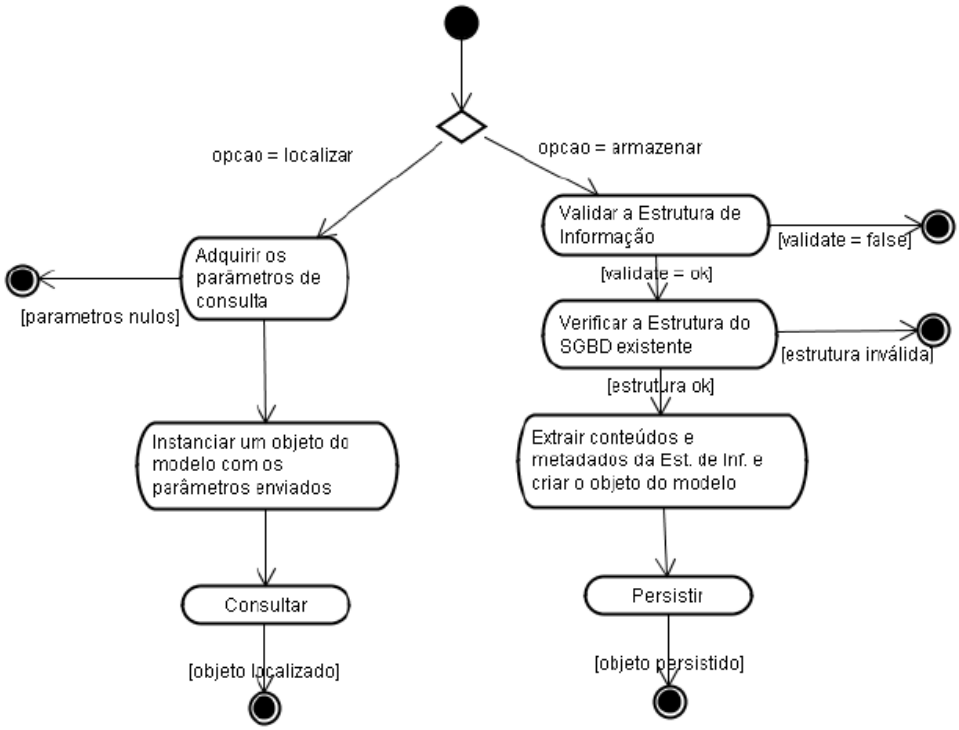

Figura 2.9: Diagrama de atividades do FPOA. Fonte: Cordeiro (2009, pág. 102)

2.4.2 Arquitetura

A arquitetura do framework desenvolvido é constituída de três componentes básicos: controller, domainobject e persistence. O controller é responsável por controlar todas as solicitações feitas e encaminhar, às entidades responsáveis, o tratamento das informações. No componente domainobject estão todos os recursos localizados na estrutura da informação, encapsulando os dados para tratamento dentro da ferramenta. O componente persistence é o componente mais relacionado ao SGBD, sendo responsável por executar tanto as transações de persistência como as de consulta de OA.

A Figura 2.2 mostra, dentro do Diagrama de Atividades do FPOA, a separação das responsabilidades entre os três componentes de sua arquitetura.

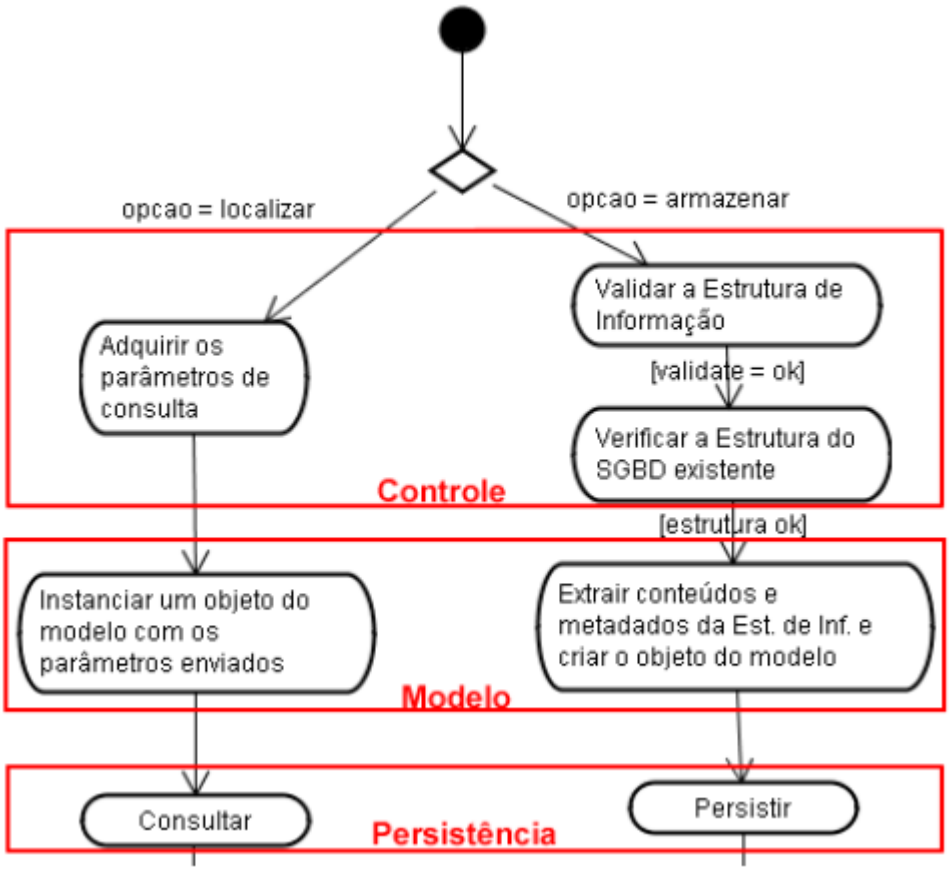

Figura 2.10: Identificação dos componentes responsáveis para cada fluxo de informação. Fonte: Cordeiro (2009, pág. 103).

3 PROJETO DO SISTEMA

Neste capítulo é descrita a implementação de um sistema para a validação do framework FPOA, armazenando e catalogando objetos de aprendizagem objetivando aumentar a reutilização dos mesmos. Primeiramente serão introduzidas as funcionalidades necessárias para que esse objetivo seja alcançado. Em seguida serão apresentados e justificados os componentes utilizados na implementação da aplicação e, por último, será explorada a sua arquitetura.

3.1 Requisitos do Sistema

A análise de requisitos tem por objetivo auxiliar os engenheiros de software a compreender melhor o problema que eles irão trabalhar para solucionar, gerando um entendimento escrito do mesmo. Isso pode ser atingido através de cenários de uso, listas de funções e de características, modelos de análise ou uma especificação. (PRESSMAN, 2006).

Nas subseções seguintes são especificados os requisitos funcionais e não-funcionais do sistema proposto para que, além de ser um repositório para objetos de aprendizagem, possa suprir os problemas já abordados nos repositórios existentes.

3.1.1 Requisitos Funcionais

Abaixo seguem os requisitos funcionais do sistema, ou seja, as funcionalidades e serviços que devem estar presentes no mesmo.

- A busca por objetos cadastradas no sistema deve ser livre, sem a necessidade de ser um usuário registrado;
- O sistema deve possuir dois tipos de busca: a simples e a avançada;
- A busca simples deve percorrer todos os conteúdos textuais de todos os objetos de aprendizagem cadastrados no sistema em busca de uma palavra-chave;
- A busca avançada deve permitir ao usuário escolher em que tipo de conteúdo textual e em qual coleção de objetos cadastrados no sistema deseja-se realizar a pesquisa pela palavra-chave;
- Usuários sem cadastro devem possuir acesso a todas as informações de um objeto buscado no sistema e poder executar o download tanto de seus metadados quanto de seu arquivo executável;
- O sistema deve permitir o cadastro de novos usuários;

- O sistema deve permitir aos usuários executarem login para acessar seus dados pessoais;
- Usuários cadastrados devem possuir permissão para adicionar novos objetos ao sistema;
- O sistema deve garantir que qualquer objeto de aprendizagem adicionado possua seus conteúdos textuais no padrão UMBRELO;
- Usuários cadastrados devem ter acesso às informações dos seus objetos adicionados para poder excluí-los.

3.1.2 Requisitos Não-Funcionais

Abaixo seguem os requisitos não-funcionais do sistema, ou seja, restrições do projeto como plataformas e tecnologias escolhidas.

- O sistema deve ser disponibilizado livremente na web;
- O sistema deve funcionar corretamente independente do sistema operacional utilizado pelo usuário, assim como independente do navegador;
- O sistema não deve utilizar nenhuma ferramenta de desenvolvimento que envolva custos;
- O sistema deve utilizar o banco de dados Firebird, por ser gratuito;
- O sistema deve utilizar o servidor web Apache;
- O sistema deve possuir interfaces de fácil utilização, onde a curva de aprendizagem do usuário será pequena.

3.2 Componentes do Sistema

Nesta seção são apresentados os componentes que se fizeram necessários para que o sistema atingisse os requisitos abordados na seção anterior. Primeiramente, é mostrado como o UMBRELO e o FPOA foram incorporados e adaptados ao projeto e depois será realizada uma descrição da plataforma de desenvolvimento e banco de dados utilizados.

3.2.1 UMBRELO

O modelo de metadados UMBRELO estende o padrão Learning Object Metadata (LOM) ao permitir que, além das informações genéricos de um OA, características particulares também sejam abrangidas tais como os recursos que compõem um OA: imagens, botões, textos, tabelas, exercícios, etc. Além de fornecer os metadados para descrição do OA e seus demais elementos textuais, descreve as relações entre eles, e guarda todos os elementos permitindo a reutilização dos mesmos (SANTOS et al., 2012).

O padrão LOM, utilizado no modelo UMBRELO, é um padrão que especifica metadados para Objetos de Aprendizagem (IEEE, 2002). Ele descreve características que podem ser agrupadas em General (Geral) - descreve características gerais do OA; Life Cycle (Ciclo de vida) - informações relacionadas à evolução do OA; Meta MetaData (Metadados) - metadados para descrever os metadados usados para o OA;

Technical (Técnica) – características técnincas e requisitos do OA; Educational (Educacional) - características educacionais e pedagógicas do OA; Rights (Direitos) - direitos de propriedade intelectual e condições de uso do OA; Relation (Relações) - relacionamentos entre OA; Annotation (Comentários) - anotações adicionais sobre um OA e Classification (Classificação) - classificação do OA. No entanto, apesar do padrão LOM permitir a descrição de metadados de OA, ele não permite a descrição de elementos específicos do conteúdo de um objeto de aprendizagem. Para os usuários, o que mais interessa é reutilizar o conteúdo de um OA, ou seja, usá-lo em outros contextos, customizando-o se necessário. Usualmente, os OAs são formados por diversos elementos (vídeo, texto, animação, áudio, etc.) que interagem entre si e o usuário e, para reutilizá-los, é preciso descrevê-los. Sendo assim, é necessário ampliar o alcance dos metadados para tratamento desses elementos, não restringindo apenas a descrição geral do OA (SANTOS et al., 2012). A Figura 3.1 apresenta a outra parte do modelo, a qual descreve os elementos de um OA.

Os metadados de conteúdo de um OA são representados pelas classes:

- Cenas (Scene): representam várias situações, sendo que uma cena deve ter no mínimo um cenário.
- Cenários (Scenary): representam as telas de cada cena. Os cenários possuem textos, imagens, botões, tabelas e exercícios.
- Textos (Texts): representam todos os textos do OA. Existem cinco tipos de textos: texto de cenário, texto explicativo, texto de Ajuda, texto de fala de personagem e texto de Help.
- Imagens (Images): representam todas as imagens/figuras que compõem o OA. Podem ser importadas através das informações contidas nos metadados que as descrevem.
- Tabelas (Tables): representam os conteúdos que estão em formato de tabela.
- Botões (Buttons): representam todos os botões e suas respectivas ações. Os tipos de botões são: navegacional, de cenas, de exercício, de glossário, mapa conceitual, de créditos, explicativo, de controle de animação e customizado.
- Exercícios (Exercises): representam os exercícios existentes no OA. Contém questões, opções de resposta, as respostas corretas e feedback.
- Créditos (Credits): representam os membros da equipe, suas funções e outros colaboradores do OA.

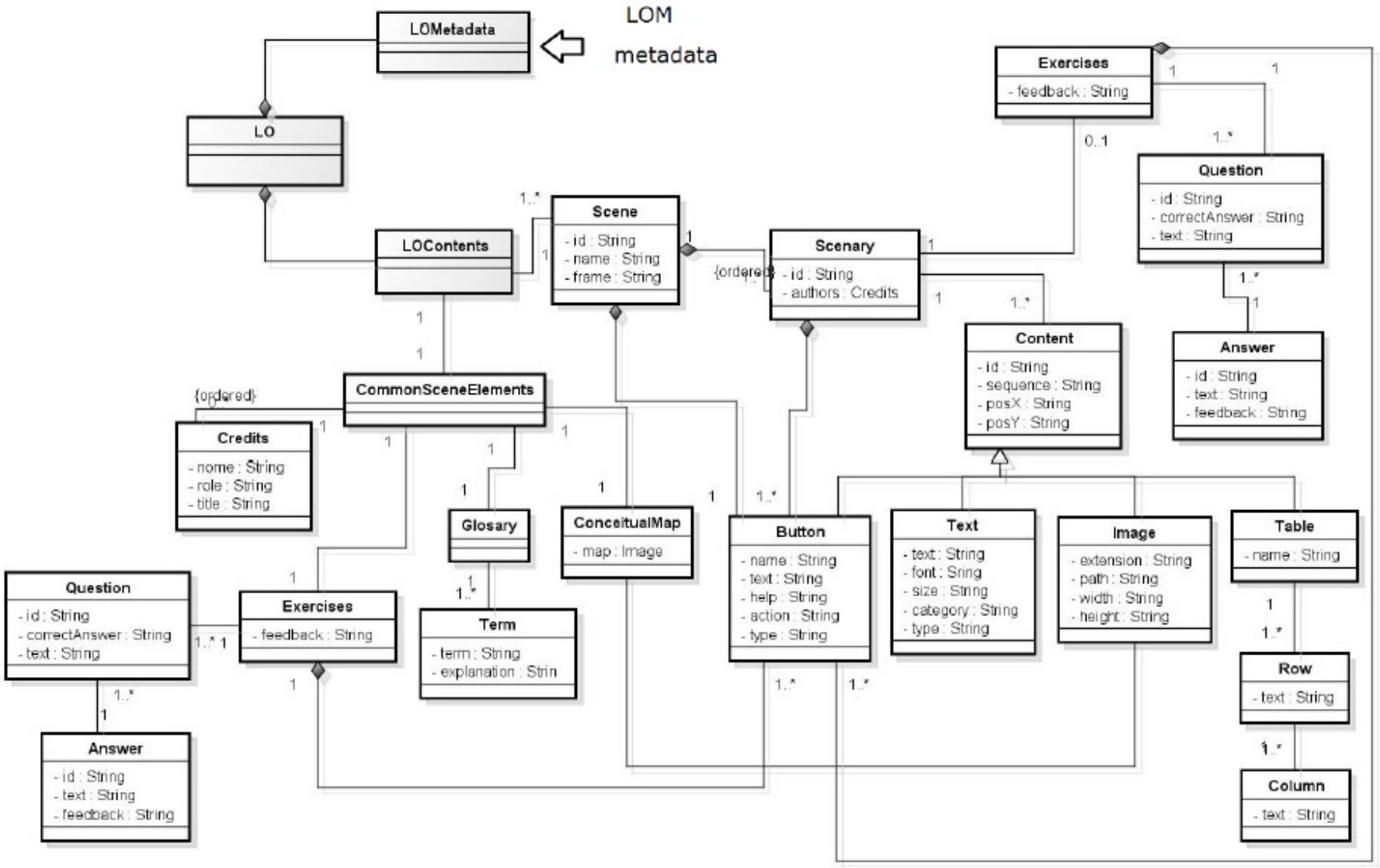

Figura 3.1: Diagrama de Classes – metadados de conteúdo do modelo UMBRELO. Fonte: Santos et al. (2012, pág. 4).

Para garantir que o repositório tenha todos seus objetos com as informações textuais nesse modelo, e tendo em vista que a estrutura de informação do mesmo é representada em XML, foi necessário gerar um XML Schema para validar todos os objetos antes de serem adicionados ao repositório.

O XML Schema é a recomendação oficial do W3C para validar um documento no formato XML, tendo como propósito definir os blocos de construção permitidos no documento, descrevendo exatamente as marcações válidas, a ordem em que devem aparecer, as marcas que podem aparecer dentro de outras, os atributos necessários e os valores que podem ter. O XML Schema utilizado pelo repositório desenvolvido está disponível no Apêndice deste trabalho.

3.2.2 Novo FPOA

Para a utilização no repositório proposto foram executadas algumas mudanças no framework FPOA, descrito no capítulo 2 deste estudo. Essas alterações visam facilitar manutenções futuras e buscar o maior proveito do fato de que todos os conteúdos textuais dos OA no repositório são representados no formato XML. Uma descrição dessas alterações e ainda como configurar o FPOA são abordados a seguir.

3.2.2.1 Estratégia de Persistência

Buscando maior proveito do fato de todos os objetos no padrão UMBRELO utilizarem a tecnologia XML, foram feitas modificações no estilo de persistência do framework alterando o banco de dados utilizado de relacional para um baseado em XML.

Um banco de dados XML nativo é um sistema de persistência de dados que permite o armazenamento destes dados em formato XML, onde podem ser consultados, exportados e serializados no formato que se desejar (BOURRET, 2013). Muitos desses bancos oferecem um modelo lógico para agrupar documentos, chamado de “coleções” onde, em algumas implementações, uma hierarquia de coleções pode existir da mesma forma que uma estrutura de diretórios em um sistema operacional. Atualmente todas as implementações desses bancos de dados suportam ao menos uma sintaxe de consulta. A sintaxe para consulta em documentos e coleções de documentos XML mais comumente suportada é a XPath, a qual permite aos usuários identificar nodos que correspondem a um determinado conjunto de critérios.

Foi realizada uma análise comparativa sobre os principais bancos de dados XML nativos gratuitos e mais completos da atualidade para escolher o mais indicado a ser utilizado no framework. A próxima subseção introduz uma breve descrição de cada um e o resultado obtido.

3.2.2.1.1 Bancos de Dados Analisados

A análise comparativa foi desenvolvida sobre os mais populares e completos bancos de dados XML nativos gratuitos, que são: BaseX, Berkeley DB XML, Exist-DB e Sedna. Além desses, ainda existe outro banco de dados que se destaca: o Xindice, mas que não será avaliado neste documento, pois está descontinuado desde 2011.

- BaseX é um banco de dados XML nativo leve e de alto desempenho desenvolvido como um projeto comunitário no GitHub, independente de

plataforma e distribuído sob uma licença de software livre. É uma ferramenta especializada em armazenamento, consulta e visualização de grandes documentos XML e coleções. Possui um processador de XQuery/XPath baseado em Java, permitindo manipulação de dados seguindo a última especificação da World Wide Web Consortium (W3C) e busca completa de textos. Ele possui uma GUI que permite aos usuários de forma interativa pesquisar, explorar e analisar os dados e avaliar expressões XPath / XQuery em tempo real.

- O Berkeley DB XML foi desenvolvido pela Oracle, na linguagem C, e distribuído sob licença LGPL 2. Ele foi construído a partir do Berkeley DB, com a adição de um parser XML, índices XML e um mecanismo XQuery. Do Berkeley DB ele herda o mecanismo de armazenamento, suporte a transações, recuperação automática, sistema de backup, transações ACID, entre outras características. Possui suporte para diversas linguagens de programação, incluindo Java.
- EXist-DB é um sistema de gerenciamento de banco de dados desenvolvido por Wolfgang Méier, inteiramente construído sobre a tecnologia XML e lançado sob a licença GPL. Esse banco de dados utiliza um sistema próprio para armazenamento de dados (B+ trees e arquivos paginados). Ele pode ser executado como um servidor de dados standalone, como uma biblioteca Java incorporada ao software, ou como um servlet para uma aplicação web. Os documentos podem ser armazenados em coleções ou em hierarquias de coleções.
- Sedna é um banco de dados XML nativo desenvolvido em C++ e distribuído sob a licença Apache. Ele implementa uma gama completa de serviços de banco de dados, como persistência de armazenamento, transações ACID, segurança, indexações e backup. Inclui suporte a XQuery e drivers para utilização em diversas linguagens de programação.

3.2.2.1.2 Análise Comparativa

A Tabela 3.1 apresenta os critérios analisados em cada banco de dados utilizado neste estudo. Esses critérios e sua importância para o framework são explicados a seguir:

- Licença: tipo de licença para a utilização do banco de dados. Por ser um projeto acadêmico, é importante que a licença gratuita;
- Compressão: implementa mecanismos para redução do espaço ocupado pelos dados. A importância deste critério é dada pelo fato de que o repositório pode crescer indefinidamente, portanto, o banco de dados deve suportar um grande volume de dados;
- Busca completa de textos: refere-se às técnicas de busca em toda uma coleção de textos de um banco de dados. Neste tipo de busca, todas as palavras em todos documentos armazenados são examinadas buscando satisfazer certos critérios. Trata-se de um critério muito importante na análise, pois o projeto que será desenvolvido conterá muitos mecanismos de busca, entre eles, a busca completa de texto;
- Sistema operacional: sistemas operacionais que o banco de dados possui compatibilidade. Como o projeto desenvolvido será alojado em um servidor

Linux, o banco de dados escolhido deve dar suporte a esse sistema operacional;

- Sistema multiusuário: permite acesso simultâneo de múltiplos usuários ao banco de dados. Característica necessária a qualquer repositório de acesso online;
- Unicode: possui suporte ao padrão de caracteres Unicode, que permite representar e manipular, de forma consistente, textos de qualquer sistema de escrita existente. Característica necessária para adicionar robustez ao sistema;
- Backup: possui suporte a cópias de segurança dos dados para que possam ser restaurados em caso de perda dos dados originais. Característica necessária para adicionar robustez ao sistema;
- Cache de consulta: cache para as consultas mais frequentes ao banco de dados. É importante para diminuir o acesso a disco e aumentar o desempenho do sistema;
- Coleções: banco de dados implementa coleções para manipular arquivos XML. Como o sistema que será desenvolvido prevê a organização dos objetos por categorias, este critério é importante para facilitar a organização dos arquivos;
- API XQuery para Java: suporta API para programação de XQuery em Java. É importante o banco de dados possuir este recurso pois o projeto será desenvolvido em Java;
- Documentação: análise da completude e simplicidade da documentação do banco de dados do ponto de vista do autor. É uma característica muito importante devido ao fato do mesmo nunca ter trabalhado com este tipo de tecnologia.

Tabela 3.1: Comparação dos principais XML-DB.

Critério	**BaseX**	**Berkeley**	**eXist-db**	**Sedna**
Licença	BSD-License	LGPL License	LGPL License	Apache License
Compressão	Sim	Sim	Sim	Sim
Busca de textos	Sim	Sim	Sim	Sim
Sistema operacional	Independente	Independente	Independente	Independente
Sistema multiusuário	Sim	Sim	Sim	Sim
Unicode	Sim	Sim	Sim	Sim
Backup	Sim	Sim	Sim	Sim
Cache de consulta	Sim	Sim	Sim	Sim
Coleções	Sim	Possui equivalente	Sim	Sim
API Xquery para Java (XQJ API)	Sim	Sim	Sim	Sim
Documentação	Boa	Ruim	Ruim	Média

Fonte: Elaborada pelo autor.

Analisando as informações da tabela, conclui-se que, em termos de funcionalidades

necessárias, todos os bancos de dados XML nativos analisados possuem as características necessárias para serem utilizados no framework, porém, analisando a documentação de cada um, o mais indicado para um usuário inexperiente é o BaseX e, portanto, é o que será utilizado.

3.2.2.2 Estrutura

A estrutura do FPOA foi alterada para se adequar à nova abordagem na persistência dos dados e melhorar a sua organização visando eventuais manutenções futuras.

Primeiramente, foi criada uma interface de acesso aos dois módulos que o framework já possuía: o de validação e o de persistência. Com a utilização dessa interface, o desenvolvedor que for utilizar o FPOA em sua aplicação não necessita de qualquer conhecimento de como os módulos funcionam.

O módulo de validação dos conteúdos textuais dos OA não sofreu nenhuma alteração, mantendo sua lógica em cima da utilização de um arquivo XML Schema para a verificação da validade dos arquivos analisados.

O módulo de persistência foi totalmente alterado devido a mudança na abordagem de armazenamento dos dados. Um motor de busca foi criado e o tipo de busca nele utilizado é definido dinamicamente a partir da interface do framework. Tendo em vista a possível utilização com outros modelos ou, até mesmo a criação de novos tipos de pesquisa, esse motor de busca utiliza uma classe derivada do padrão de projeto strategy para decidir qual tipo de busca na base de dados deverá efetuar.

O padrão de projeto strategy tem por objetivo definir uma família de algoritmos, encapsular cada um, e fazê-los intercambiáveis, permitindo que o algoritmo varie independentemente dos clientes que o utilizam (GAMMA et al., 1995). Sendo assim, para adicionar uma nova estratégia de busca ao framework, basta criar uma classe que implemente a interface strategy citada e acrescentá-la aos possíveis tipos de consulta à base de dados do FPOA. Lembrando que, ao implementar a interface strategy, os métodos devem retornar uma string contendo a consulta na linguagem Xquery e Xpath, que são os padrões dos bancos de dados XML. Um exemplo de implementação de estratégia de busca é apresentado no código seguinte.

```
public class TitleSearch implements SearchStrategy
{
   public String searchQuery( String where, String word )
   {
      String query = " for $i in fn:collection('" + where + "')" +
            " where ( $i/object/Rived/LoMetadata/General/title[" +
            "           contains( lower-case(.), lower-case('" + word +
"') )] )" +
            " return $i";
      return query;
   }

   public String searchQueryInSpecificObject( String where, String word )
   {
      String query = " for $i in fn:doc('" + where + "')" +
            " where ( $i/object/Rived/LoMetadata/General/title[" +
            "           contains( lower-case(.), lower-case('" + word +
"') )] )" +
            " return $i";
```

```
        return query;
    }
}
```

O diagrama de classes apresentado na Figura 3.2 demonstra como ficou organizada a estrutura da ferramenta após todas as modificações.

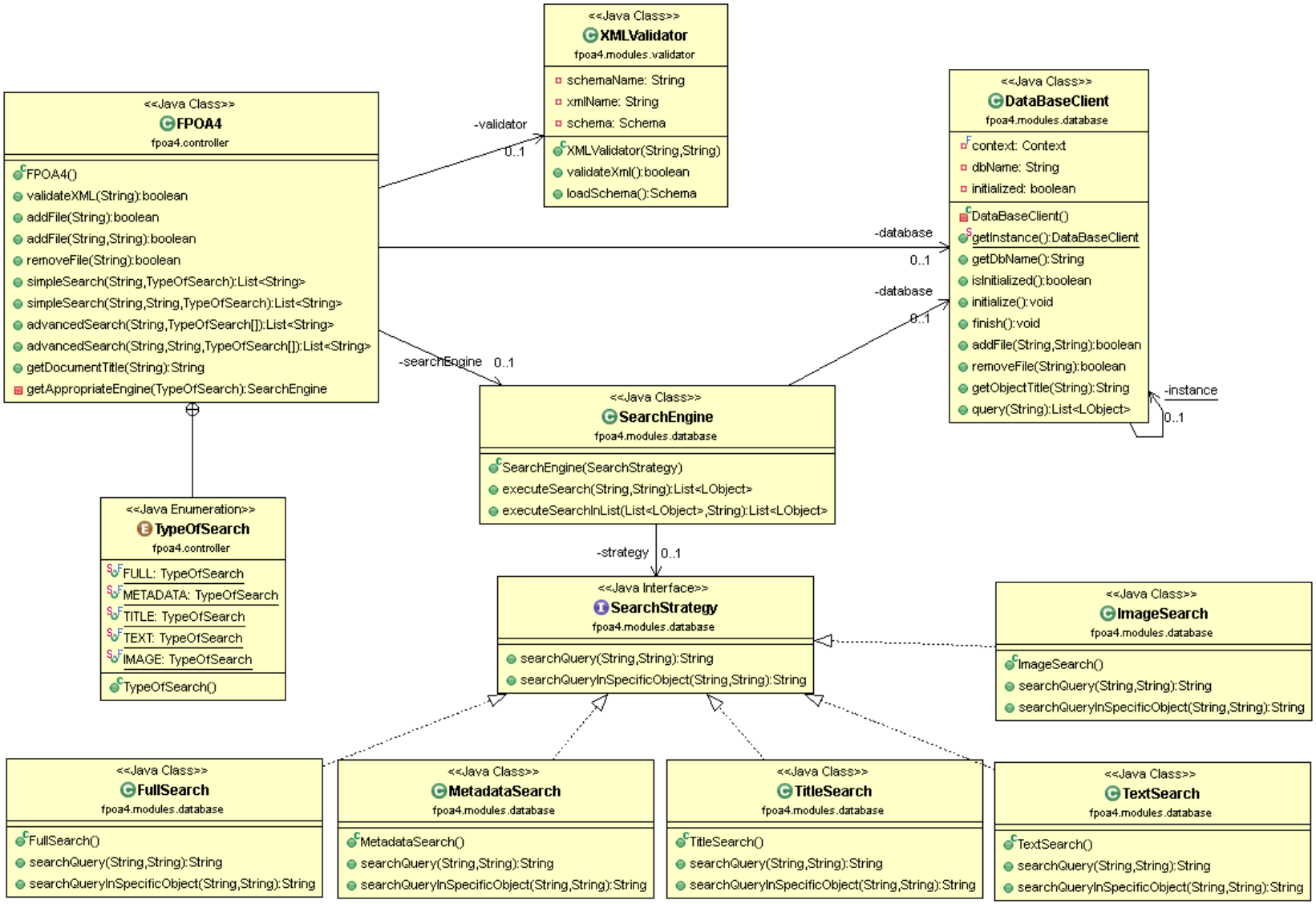

Figura 3.2: Diagram de classes do FPOA. Fonte: Elaborado pelo autor.

A classe FPOA4 executa o papel de interface da ferramenta, contendo uma função para cada operação possível do framework. Cada uma dessas funções recebe os argumentos, repassa-os para o módulo responsável e, ao final da transição, retorna a resposta do módulo. Por exemplo, quando a função "validateXML" dessa classe é chamada, ela repassa seus argumentos para a função de mesmo nome na classe XMLValidator, que executa a lógica da operação, e retorna seu valor de resposta.

O módulo de validação é representado pela classe XMLValidator, a qual é bem simples e possui apenas uma função para validar um arquivo XML através de um XML Schema configurado previamente. A configuração desse parâmetro será abordada na próxima subseção.

O módulo de persistência é composto pelas demais classes do diagrama apresentado na Figura 3.1, onde destacam-se 2 classes: DataBaseClient e SearchEngine. A primeira é a responsável por executar todas as transações com a base de dados do framework, como adição, busca, etc. Já a segunda, implementa o motor de busca da ferramenta, utilizando a estratégia definida no momento de sua instanciação. As estratégias de busca disponíveis na ferramenta são as definidas pelas seguintes classes, que são implementações da interface SearchStrategy:

- FullSearch: implementa a busca completa em todo o arquivo de conteúdos textuais de um objeto;
- MetadataSearch: implementa a busca somente nos campos de metadados no arquivo XML de um objeto;
- TitleSearch: implementa a busca somente no campo do arquivo XML que possui o título do objeto;
- TextSearch: implementa a busca em todos os campos do arquivo XML menos nos que contém os metadados;
- ImageSearch: implementa a busca somente nos campos de imagens do arquivo XML.

3.2.2.3 Configuração

O componente possui um arquivo de configuração onde é possível alterar o nome do banco de dados utilizado e o XML Schema de validação de seus objetos. Ele é escrito no formato XML e segue o seguinte modelo:

```
<Config>
<Database>Teste</Database>
<Schema>C:/Temp/umbrello.xsd</Schema>
</Config>
```

Como pode ser acompanhado no código acima, o campo Database define o nome do banco de dados onde serão armazenados os objetos do framework, não necessitando apontar para uma base de dados válida pois, caso ela não exista, será criada na primeira vez em que for executado o framework. Já o campo Schema define o caminho absoluto para o arquivo de validação dos conteúdos textuais dos OA dessa base de dados.

3.2.3 Java EE

A tecnologia Java pode referenciar tanto à linguagem de programação de alto nível

orientada à objetos com seu estilo e sintaxe particulares, quanto a plataforma em que aplicações desenvolvidas nesta linguagem rodam. Ela é conhecida por facilitar o desenvolvimento de aplicações e ser independente de plataforma (ORACLE, 2012).

Atualmente existem quatro plataformas de programação Java: Standard Edition (Java SE), Enterprise Edition (Java EE), Micro Edition (Java ME) e Java FX. Cada uma delas consiste em uma API que representa uma coleção de componentes que possibilitam a criação de aplicações e uma máquina virtual capaz de rodar essas aplicações.

A API Java SE fornece o núcleo de funcionalidades da linguagem de programação Java. Ela define desde os tipos e objetos básicos da linguagem até as classes mais complexas responsáveis pela segurança, acesso à bases de dados, interface de desenvolvimento gráfico entre outras.

A plataforma Java EE foi desenvolvida sobre a plataforma Java SE, provendo uma API para desenvolvimento de aplicações distribuídas. O propósito dessa plataforma é reduzir a complexidade do desenvolvimento de aplicações Web, através de sua API que permite aos desenvolvedores se concentrar nas funcionalidades de sua aplicação e não nos detalhes de rede, segurança, confiabilidade e etc.

Seis componentes da especificação Java EE foram empregados neste trabalho e estão brevemente descritos na Tabela 3.2.

Tabela 3.2: Componentes da API Java Enterprise Edition aproveitados neste trabalho.

Componente	**Descrição**
Servlets	Classes desenvolvidas na linguagem de programação Java que dinamicamente processam requisições, gerando respostas que, geralmente, são páginas HTML.
Tecnologia JavaServer Faces	Um framework para aplicações web que permite incluir componentes de interface de usuário em uma página, converter e validar os dados dos mesmos, salvar esses dados em estruturas do lado servidor e manter o estado do componente.
Tecnologia JavaServer Faces Facelets	São um tipo de aplicação JavaServer Faces que utilizam páginas XHTML ao invés de JavaServer Pages (JSP).
Expression Language	Um conjunto de marcas utilizadas em páginas JSP e Facelets para se referir a componentes do Java EE.
Componentes JavaBeans	Objetos que provêm armazenamento de dados temporários para as páginas de uma aplicação.
Java Database Connectivity API (JDBC)	API para acesso e obtenção de informações de bancos de dados. Um de seus usos comuns é para executar consultas SQL em

	determinados bancos de dados.

Fonte: Traduzido pelo autor com base em Oracle (2012).

Adicionalmente aos componentes citados na tabela, foram utilizadas duas bibliotecas de componentes para JavaServer Faces: PrimeFaces e Tomahawk. A primeira é uma suíte contendo mais de 100 componentes JSF de código aberto com diversas extensões e com suporte a temas, facilitando a construção de uma interface com um bom visual sem a necessidade da definição de muitos estilos nas páginas da aplicação. A segunda biblioteca, Tomahawk, também possui diversos componentes JSF e se fez necessária por possuir alguns componentes importantes para o desenvolvimento do sistema proposto que o PrimeFaces não possuía.

O autor optou por utilizar Java pelo fato de ter bastante experiência com esta linguagem de programação, conhecendo muito de seu potencial e de problemas que poderia enfrentar durante a implementação. Além disso, a escolha dessa tecnologia facilitou a integração com o framework FPOA, desenvolvido utilizando essa mesma tecnologia.

O ambiente de desenvolvimento, ou IDE, escolhido foi o Eclipse versão Juno (4.2) com suporte à linguagem de programação Java. O principal motivo dessa escolha foi o fato de tal IDE reunir diversas ferramentas que auxiliam e agilizam o processo de desenvolvimento de aplicações, além de possuir uma interface simples, uma boa documentação e de ser disponível gratuitamente na Internet.

3.2.4 Firebird

Firebird é um banco de dados relacional que oferece muitas características do padrão SQL e que funciona em Linux, Windows, e uma variedade de plataformas Unix. Firebird oferece excelente concorrência, alta performance e suporte a poderosa linguagem de procedimentos armazenados e gatilhos. O projeto Firebird é um projeto comercialmente independente, onde conta com desenvolvedores C e C + +, consultores técnicos e de suporte, baseado no código fonte liberado pela Inprise Corp (agora conhecido como Borland Software Corp) em 25 de julho de 2000.

O autor optou por utilizar esta ferramenta de banco de dados para o repositório pela sua experiência com a mesma, onde sempre observou uma boa documentação e uma facilidade de configuração e manutenção das aplicações que desenvolveu utilizando-o.

3.3 Arquitetura do Sistema

Nesta seção será abordada a arquitetura do sistema proposto. A aplicação desenvolvida é baseada na arquitetura Cliente-Servidor e segue o padrão MVC como modelo estrutural, ambos detalhados nas subseções que seguem.

3.3.1 Cliente-Servidor

O modelo Cliente-Servidor é uma estrutura de aplicação distribuída composta por dois componentes chamados Cliente e Servidor que se comunicam através de uma rede de computadores. A função do Servidor é fornecer recursos ou serviços ao Cliente e, para tanto, deve receber requisições de uma ou mais instâncias de um Cliente, processá-las e enviar uma resposta baseada na requisição. Os Clientes apenas geram requisições

e, após enviá-las, aguardam pela resposta (TANEMBAUM, 2003).

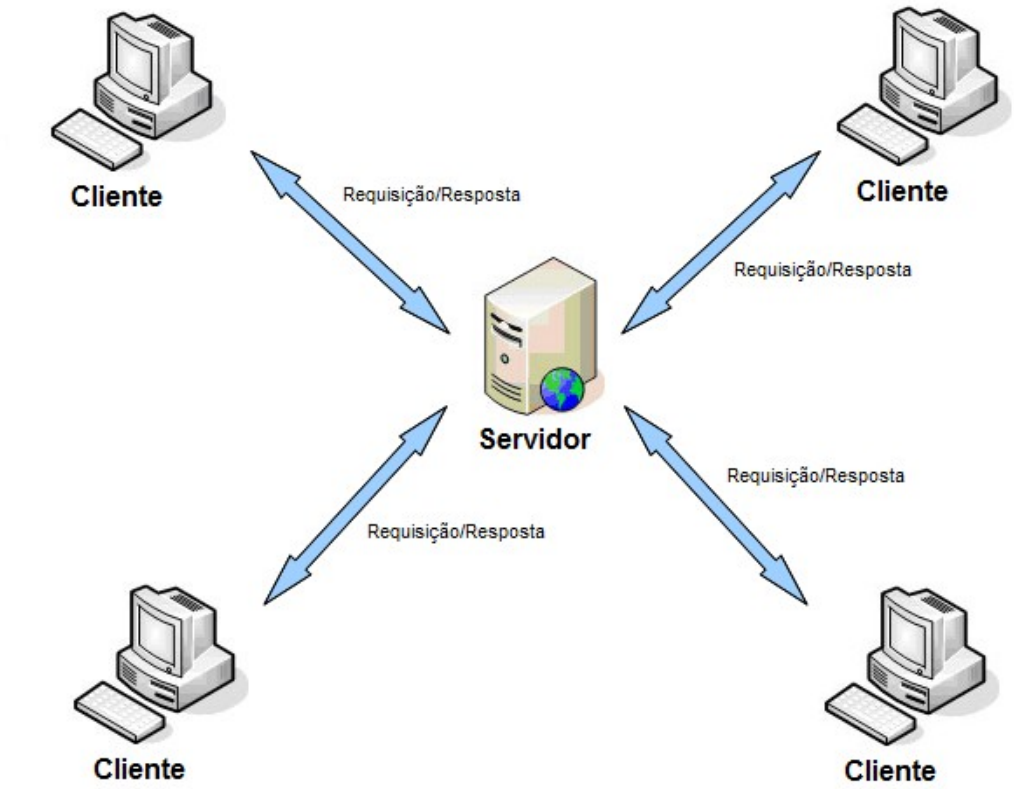

Figura 3.3: Ilustração da arquitetura Cliente-Servidor. Fonte: Elaborado pelo autor.

No sistema proposto neste trabalho, o navegador web pelo qual o usuário irá acessar o repositório faz o papel de Cliente que interagirá com o Servidor através do protocolo HTTP, requisitando serviços disponíveis na aplicação como efetuar uma busca ou adicionar um novo objeto ao repositório.

3.3.2 MVC

O MVC (Model-View-Controller) é um padrão arquitetural que consiste em dividir as responsabilidades da aplicação em três tipos de componentes. O Modelo representa o objeto da aplicação, a Visualização representa as telas do sistema e o Controlador define o caminho que a interface irá seguir conforme uma ação do usuário. Esta abordagem reduz a dependência entre o Modelo e as Visualizações, aumentando a flexibilidade e o reuso do código (GAMMA, HELM, JOHNSON, VLISSIDES, 1995).

A Figura 3.4 ilustra as interações entre os componentes do padrão MVC.

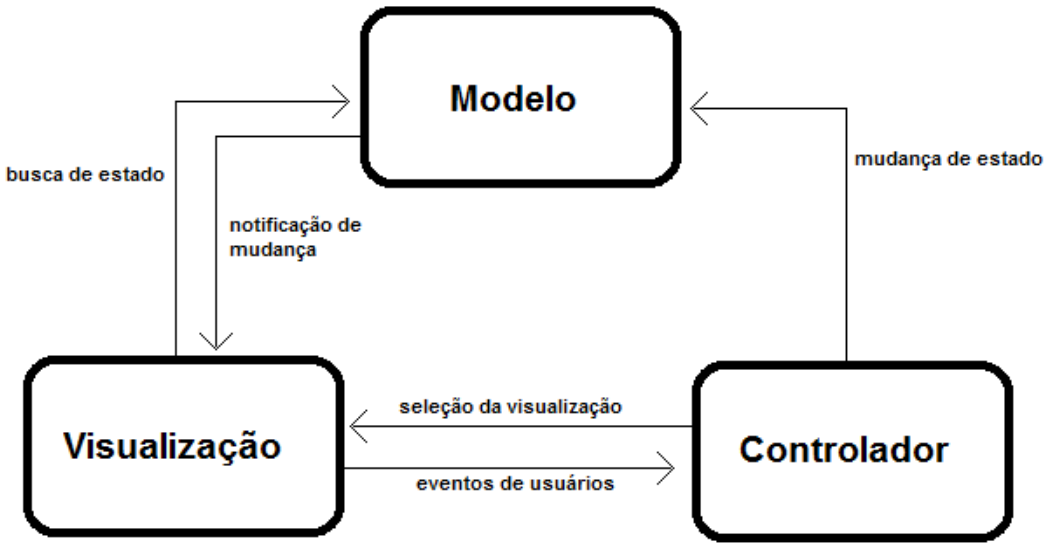

Figura 3.4: Interações entre os componentes MVC. Fonte: Elaborado pelo autor.

4 DESENVOLVIMENTO DO SISTEMA PROPOSTO

No capítulo anterior foram apontados os principais requisitos para o sistema proposto validar o framework de persistência FPOA, implementando um repositório que armazene e catalogue objetos de aprendizagem. Também foram vistos os componentes utilizados no desenvolvimento deste sistema e a sua arquitetura.

Este capítulo detalha a solução do sistema proposto. Primeiramente é apresentada a estrutura da base de dados utilizada pela aplicação. Na segunda seção é abordada a estrutura de classes resultante desse modelo. A terceira seção mostra imagens do sistema desenvolvido com base nessa modelagem e apresenta uma análise navegacional de toda a aplicação.

4.1 Modelagem do Banco de Dados

O FPOA possui um sistema próprio de armazenamento para os conteúdos textuais dos objetos. Nesse caso, tendo em vista o armazenamento das informações de usuários cadastrados e de informações dos objetos, a criação de um banco de dados complexo para a aplicação tornou-se desnecessária. Assim, a simplicidade do banco de dados é visível.

O diagrama ER (Entidade-Relacionamento) da base de dados modelado para o sistema proposto está ilustrado na Figura 4.1. Cada entidade e seus respectivos atributos estão descritos nas subseções seguintes.

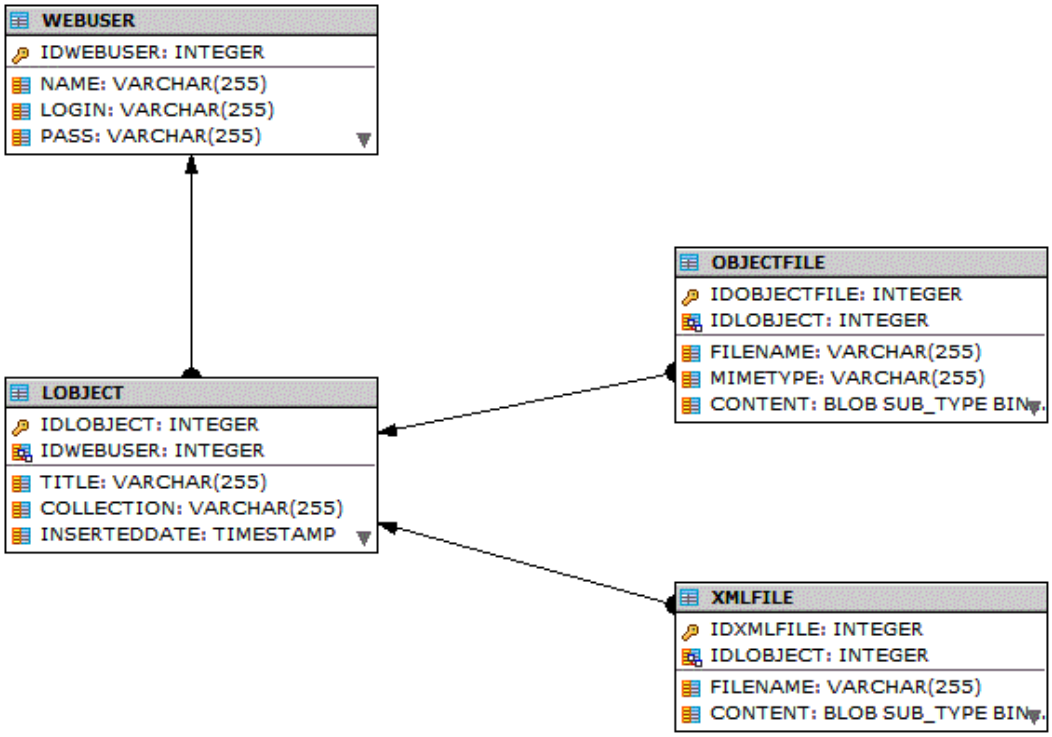

Figura 4.1: Diagrama ER do banco de dados do sistema. Fonte: Elaborado pelo autor.

4.1.1 Entidade WebUser

Esta entidade é responsável por armazenar todas as informações dos usuários cadastrados no sistema. Os dados necessários e obrigatórios para o registro de um usuário são nome, login e senha, representados, respectivamente, por “name”, “login” e “pass”, onde o campo “login” é único no tabela. Além desses campos, existe um identificador único de cada usuário no banco de dados para controle do sistema.

4.1.2 Entidade LObject

A tabela LObject é a principal do sistema, mantendo as informações de todos os objetos adicionados no repositório. Ela possui, além do identificador único de cada objeto no sistema, o título do objeto, a coleção em que o mesmo se encontra e a data em que foi adicionado ao repositório.

Visto que para adicionar um objeto ao sistema o usuário precisa possuir um registro, esta tabela mantêm para cada objeto o identificador único do usuário que o adicionou, criando uma relação com a tabela de usuários. Um usuário pode estar relacionado com n objetos, já que pode adicionar quantos objetos desejar, porém cada objeto terá relação com apenas um usuário.

4.1.3 Entidade XMLFile

A entidade XMLFile representa os arquivos de conteúdos textuais dos objetos adicionados ao repositório. Ela possui um campo para o nome do arquivo com sua extensão (no caso sempre será .xml) e um campo do tipo binário para armazenar o conteúdo do arquivo em si. Cada objeto possui um único arquivo contendo seus

metadados e, por isso, esta tabela armazena em uma chave única o identificador do objeto a que se refere.

4.1.4 Entidade ObjectFile

Enquanto a entidade XMLFile se refere aos arquivos de conteúdos textuais dos objetos adicionados ao repositório, a entidade ObjectFile representa os arquivos contendo os conteúdos propriamente ditos dos objetos. Ela possui os mesmos campos da tabela XMLFile acrescido de um campo para manter o tipo de arquivo do registro. Cada objeto deve possuir somente um arquivo executável e, portanto, essa tabela possui uma relação de 1 para 1 com a tabela de objetos.

4.2 Organização das Classes

A estrutura do sistema proposto seguiu o padrão MVC com a utilização do framework JavaServer Faces, ambos especificados no capítulo anterior. As seções subsequentes apresentam as classes e páginas criadas na implementação desta aplicação e a sua relação com os componentes do MVC.

4.2.1 Modelo

A Figura 4.2 mostra as classes que representam o componente Modelo do padrão MVC na aplicação desenvolvida.

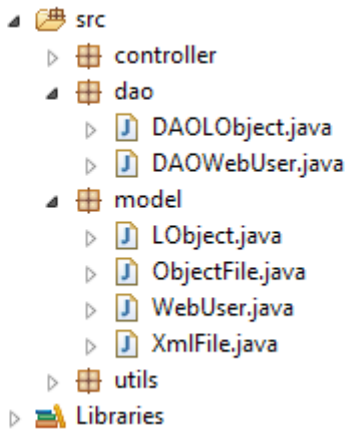

Figura 4.2: Classes do componente Modelo da aplicação. Fonte: Elaborado pelo autor.

Como pode ser visto na Figura 4.2, os pacotes “dao” e “model” representam o componente Modelo no sistema proposto. O pacote “dao” contém as classes que seguem o padrão DAO (Data Acess Object) para o acesso à base de dados, onde cada uma possui o nome da tabela acessada por seus métodos. Não foi criada uma classe DAO para as tabelas XmlFile e ObjectFile pois elas sempre são acessadas no mesmo contexto que a tabela LObject, utilizando assim a mesma interface dela para acessar seus dados.

O outro pacote, denominado “model”, possui as classes responsáveis por encapsular e operar sobre os dados do sistema. Cada classe deste pacote representa uma tabela existente no banco de dados com todos os seus atributos. Sempre que as informações sobre um registro de alguma tabela são trazidas do banco de dados, uma instância de sua

classe é criada para encapsular as suas informações. Esta instância é manipulada pelo Controlador e mapeada para uma Visualização permitindo a exibição dessas informações ao usuário.

4.2.2 Visualização

A Figura 4.3 mostra as páginas que representam o componente Visualização do padrão MVC na aplicação desenvolvida.

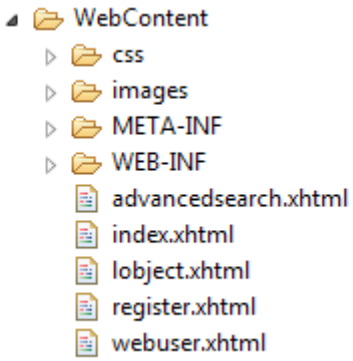

Figura 4.3: Páginas do componente Visualização da aplicação. Fonte: Elaborado pelo autor.

A pasta WebContent é o diretório raiz da aplicação no servidor e contém, além das imagens e estilos da aplicação, as páginas em XHTML que representam o componente Visualização neste sistema. Cada uma dessas páginas representa uma tela diferente da aplicação e estão ligadas aos Controladores do sistema. A página lobject.xhtml, por exemplo, representa a tela com as informações de um objeto de aprendizagem selecionado pelo usuário. Esta página está associada ao Controlador denominado LObjectController de forma que, quando o usuário interage com a página, é ele que define a ação que deve ser tomada no sistema.

4.2.3 Controlador

A Figura 4.4 ilustra as classes que representam o componente Controlador do padrão MVC na aplicação desenvolvida.

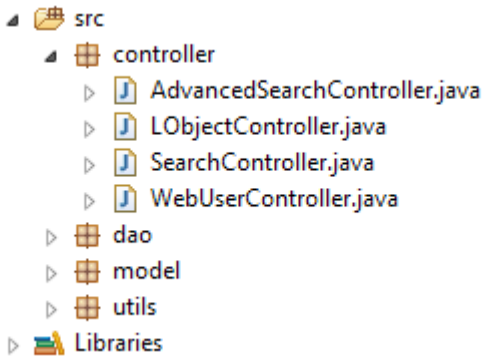

Figura 4.4: Classes do componente Controlador da aplicação. Fonte: Elaborado pelo autor.

O pacote “controller” contém as classes que representam o componente Controlador nesta aplicação. Como mencionado anteriormente, cada uma delas está associada a uma

tela do sistema e são acionadas a partir de ações do usuário. O trabalho de uma classe controladora é identificar as mudanças que devem ser executadas nas instâncias do Modelo de acordo com a ação efetuada pelo usuário, chamar as funções responsáveis por tais mudanças e identificar a Visualização que deve ser apresentada ao usuário.

4.3 Interface Navegacional

Todas as telas do sistema possuem uma estrutura padrão. Na parte superior da página existe a imagem de cabeçalho, contendo o título do sistema e o nome de seu autor. Na mais inferior de cada tela está a imagem de rodapé, com os logos da UFRGS e do Instituto de Informática. Por fim, entre as imagens de cabeçalho e de rodapé encontram-se os conteúdos de cada página.

Nas subseções seguintes são exibidas as telas da aplicação desenvolvida esclarecendo sua navegação e funcionamento.

4.3.1 Tela Inicial

A página inicial é a primeira visão do sistema exibida para o usuário ao acessar a sua URL e todos os usuários tem acesso a ela, sem a necessidade de um cadastro prévio.

Figura 4.5: Tela inicial do sistema. Fonte: Imagem salva pelo autor utilizando o sistema.

Como pode ser acompanhando na Figura 4.5, a tela apresenta no seu topo direito, abaixo da imagem de cabeçalho, um botão para o usuário poder se logar no sistema e

acessar sua área restrita. Esse botão não encaminha o usuário a nenhuma outra página, apenas dá acesso ao formulário de login e ao link para se cadastrar no sistema como pode ser visto na Figura 4.6.

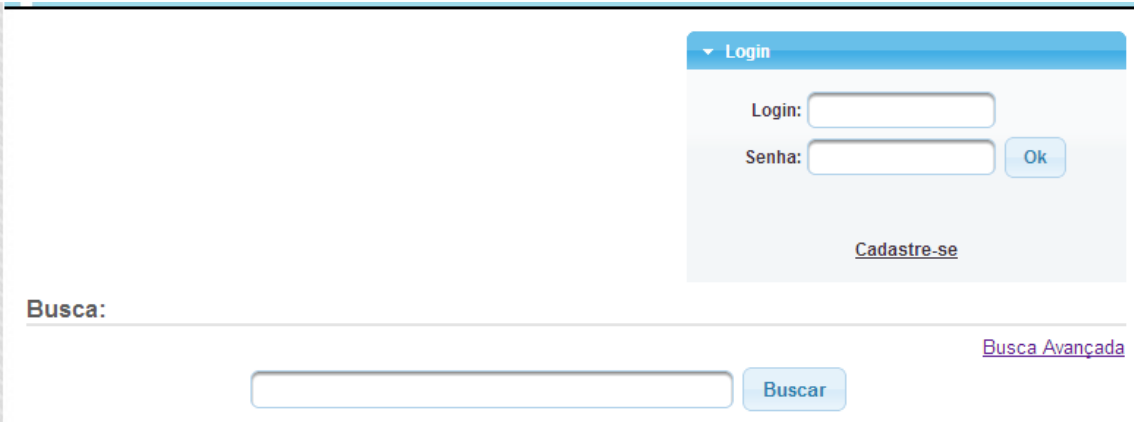

Figura 4.6: Trecho da tela inicial com formulário de login habilitado. Fonte: Imagem salva pelo autor utilizando o sistema.

Caso o usuário esteja logado no sistema, ao invés do botão de login, serão apresentados dois botões: um para ir para a página pessoal do usuário e outro para efetuar o logout. A Figura 4.7 ilustra esse caso.

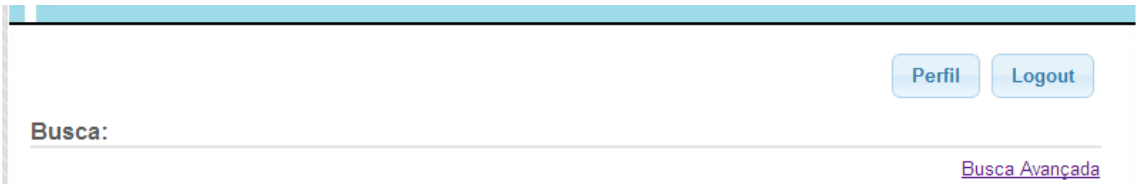

Figura 4.7: Trecho da tela contendo o menu de usuário logado. Fonte: Imagem salva pelo autor utilizando o sistema.

No centro da página inicial existe um pequeno formulário para busca rápida por objetos de aprendizagem no repositório. Ele contém um campo para colocar a palavra-chave que se deseja buscar e um botão para realizar a pesquisa que, ao ser pressionado, exibe ao usuário uma tabela com os resultados obtidos na busca. A tabela possui três colunas: o título do objeto, sua coleção e a sua data de inserção no repositório. Para visualizar mais informações de um objeto e poder baixar os seus arquivos, o usuário deve clicar em seu título, que é um link para a página de detalhes de um objeto, explicada nas próximas subseções. Vale ressaltar que essa pesquisa pela palavra-chave é executada em todos os conteúdos textuais de todos os objetos do repositório. A Figura 4.8 simula uma busca pela palavra “química” no repositório e as Figuras 4.9 e 4.10 mostram, respectivamente, um cenário do objeto “Efeito Estufa” e a busca por uma palavra contida no texto desse cenário.

Acima deste formulário existe um link que redireciona o usuário para uma página onde pode efetuar buscas avançadas no sistema. Essa tela será o foco da próxima subseção.

Figura 4.8: Tela inicial com resultados de uma busca. Fonte: Imagem salva pelo autor utilizando o sistema.

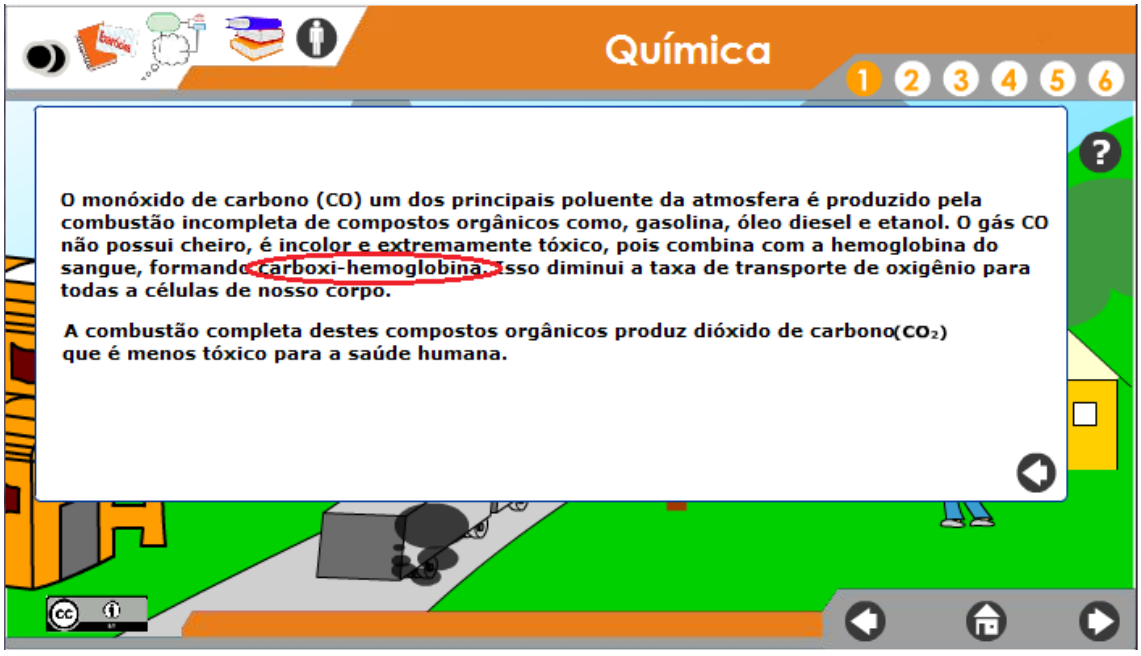

Figura 4.9: OA Efeito Estufa. Fonte: Imagem salva pelo autor utilizando o objeto.

Figura 4.10: Exemplo de busca por uma palavra contida no OA Efeito Estufa. Fonte: Imagem salva pelo autor utilizando o sistema.

4.3.2 Tela de Busca Avançada

A página de busca avançada é semelhante a tela inicial, a diferença encontra-se no fato de o formulário de busca possuir mais campos, permitindo uma pesquisa refinada. Neste estilo de busca, o usuário deve inserir a palavra-chave que deseja pesquisar tendo como opção informar a coleção e os locais no arquivo de conteúdos textuais onde essa palavra deve ser pesquisada. As opções de conteúdos textuais para busca são as estratégias de busca implementados no FPOA, explicados no capítulo anterior. Os resultados da pesquisa são exibidos da mesma forma que na busca simples, como pode ser acompanhado na Figura 4.12.

Figura 4.11: Tela de busca avançada. Fonte: Imagem salva pelo autor.

Figura 4.12: Simulação de uma busca avançada. Fonte: Imagem salva pelo autor.

4.3.3 Tela de Cadastro

A tela de cadastro apresenta um formulário com todos os campos necessários para o

registro do usuário no sistema. Ao término de seu preenchimento, o usuário já é encaminhando automaticamente a sua página pessoal e está apto a efetuar o login no momento que desejar, inserindo os dados de login e senha preenchidos no cadastro.

Figura 4.13: Tela de cadastro. Fonte: Imagem salva pelo autor utilizando o sistema.

4.3.4 Tela Pessoal de Usuário

A tela pessoal é a página mostrada ao usuário quando ele se loga no sistema. Ela se divide em 3 partes: o menu de navegação, a tabela de objetos adicionados e o formulário para adição de um novo OA.

O menu de navegação é bem simples e está posicionado do canto superior direito da tela, abaixo da imagem de cabeçalho. Ele contém 2 botões: um para ir para a página inicial (de busca) do sistema e outro para o usuário efetuar o logout.

A tabela de objetos adicionados, como o próprio nome já diz, possui todos os objetos que o usuário já inseriu no repositório desde o momento de seu cadastro. Essa tabela contém as mesmas informações que as tabelas de resultados de pesquisa, acrescida de uma coluna com um botão para excluir o OA do sistema, privilégio único do usuário que o adicionou. Ao clicar nesse botão é exigida um confirmação como ilustrado na Figura 4.15.

O formulário mais abaixo na página permite ao usuário inserir um novo objeto no repositório. Tendo em vista que todas as informações textuais serão providas do arquivo XML do objeto, para adicionar um novo objeto ao sistema basta inserir seu arquivo de conteúdos textuais e seu arquivo de execução. Opcionalmente é possível inserir uma

coleção para o OA. É importante ressaltar que o arquivo contendo os conteúdos textuais deve estar no formato XML e seguir o modelo UMBRELO, pois é efetuada uma validação através FPOA nesse arquivo antes de adicioná-lo ao repositório. A Figura 4.16 exibe a resposta do sistema para algumas tentativas de adição de um OA.

Figura 4.14: Tela pessoal do usuário no sistema. Fonte: Imagem salva pelo autor utilizando o sistema.

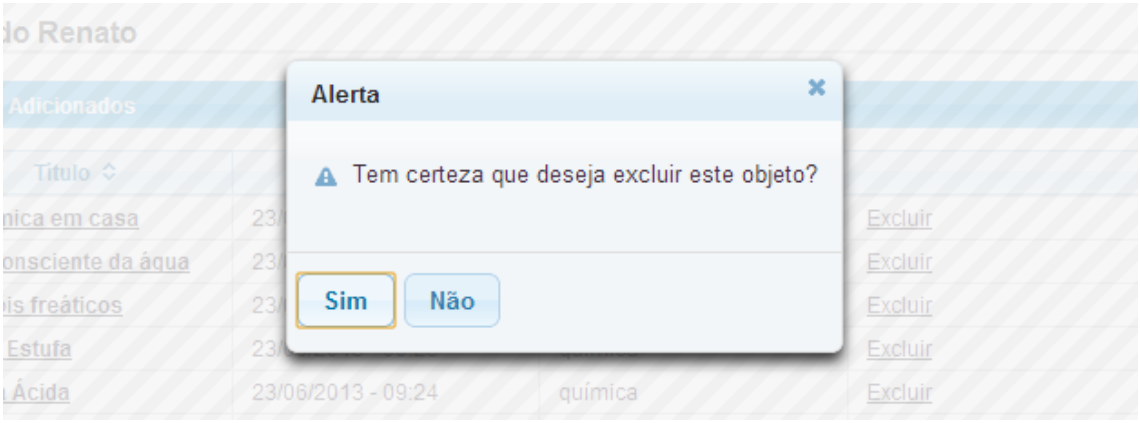

Figura 4.15: Pedido de confirmação de exclusão. Fonte: Imagem salva pelo autor utilizando o sistema.

Figura 4.16: Exemplos de resposta do sistema para o formulário de adição de OA. Fonte: Imagem salva pelo autor utilizando o sistema.

A partir do momento em que o arquivo XML é validado e aceito, o mesmo é

adicionado à base de dados do FPOA e do repositório, podendo ser pesquisado e baixado por outros usuários do sistema.

4.3.5 Tela de Objeto

Nesta página são exibidas todas as informações relevantes de um objeto de aprendizagem selecionado nos resultados de uma pesquisa ou na tabela de objetos adicionado pelo usuário. Os dados amostrados são retirados direto do arquivo de conteúdos textuais do objeto e são formatados para exibição através de uma transformação utilizando XSLT, uma ferramenta potente que permite a criação de código HTML a partir de um arquivo XML.

Bem abaixo, acima da imagem de rodapé, existem dois botões: um para o usuário baixar o arquivo XML do objeto e outro para baixar o objeto propriamente dito, lembrando que ambos arquivos são necessários para o funcionamento do objeto nos exemplos utilizados neste trabalho.

Figura 4.17: Tela de informações de um OA. Fonte: Imagem salva pelo autor ao utilizar o sistema.

5 CONCLUSÕES

Neste trabalho foi apresentado o projeto e desenvolvimento de um repositório web de objetos de aprendizagem visando a validação da ferramenta FPOA permitindo, assim, a localização e recuperação de OA através de qualquer conteúdo textual que esse possua. Para conseguir tal objetivo, se fez necessária a adaptação da referida ferramenta ao modelo UMBRELO, visto que esse descreve todos os conteúdos textuais de um OA, possibilitando a sua catalogação e recuperação.

Durante o desenvolvimento do trabalho foram realizadas análises referentes aos estudos relacionados, aos estados atuais dos repositórios de OA e ao framework FPOA desenvolvido em trabalhos anteriores. Após, foram descritos os requisitos necessários para o sistema contemplar os objetivos propostos, seus componentes e sua arquitetura. Dentro da descrição do FPOA, foram sugeridas e implementadas mudanças em sua solução para adicionar maior robustez e versatilidade com relação ao modelo de dados utilizado por ele. Definidas questões de projeto, a implementação foi realizada resultando na aplicação apresentada no capítulo anterior.

5.1 Limitações

A integração entre a base de dados do FPOA e do repositório apresentou algumas incompatibilidades, principalmente no que diz respeito ao conjunto de caracteres configurados em cada uma. Na primeira foi utilizado o UTF-8 (8-bit Unicode Transformation Format), o qual pode representar qualquer carácter universal do padrão Unicode. Ele foi utilizado, pois todos os arquivos de descrição dos objetos estavam com seus caracteres formatados nesse padrão. Em contra partida, a base de dados do repositório foi configurada com o conjunto de caracteres ISO-8859-1, sendo essa a codificação de caracteres do alfabeto latino. A sua utilização foi necessária visto o repositório ter sido desenvolvido com a tecnologia JavaServer Faces, necessitando de um servidor Apache para funcionar, o qual codifica todas as requisições e respostas HTTP nesse formato.

Apesar das limitações encontradas, nenhuma funcionalidade do sistema foi prejudicada, pois foram implementadas conversões nas codificações dos caracteres, tornando possível a comunicação entre as bases de dados sem ocorrer erros.

5.2 Resultados

Os resultados obtidos com a realização desse trabalho foram:

- Verificação de pesquisas relacionadas ao problema de localização de OA em

repositórios mostrando a escassez de estudos sobre o assunto e que os existentes não propõem soluções que vão além da utilização dos metadados para indexação dos objetos nos repositórios;

- Verificação dos mecanismos de catalogação e busca dos principais repositórios de OA nacionais, constatando que eles ainda utilizam catalogação manual e mecanismos de busca pouco eficazes quando realizadas pesquisas por palavras que não estão presentes nos metadados dos objetos;
- Implementação de um ROA com funcionalidades que auxiliam o reuso dos OA no modelo UMBRELO e que não são encontradas nos demais repositórios do país, tais como: realizar a catalogação de objetos de aprendizagem somente utilizando um arquivo de descrição do OA e permitir a localização e recuperação de OA através de qualquer conteúdo textual ou metadado que o mesmo possua;
- Validação do framework FPOA, pois foi o componente do sistema implementado que possibilitou a catalogação automática dos OA e buscas eficazes por qualquer conteúdo textual dos objetos;
- Implementação de um motor de busca para o framework FPOA, permitindo que ele seja utilizado para outros padrões de metadados através da implementação de novas estratégias de busca;
- Validação do modelo UMBRELO em um sistema real, descrevendo os objetos de forma a possibilitar o funcionamento pleno do FPOA.

Conforme os itens listados acima, pode-se afirmar que os principais objetivos deste trabalho, validar o FPOA e implementar um repositório que o utilizasse, não só foram alcançados como permitiram a obtenção de outros resultados importantes nesta linha de pesquisa.

5.3 Trabalhos Futuros

Durante a implementação do sistema surgiram algumas ideias interessantes de melhorias para o repositórios que não foram implementadas por não serem prioridade no momento de seu desenvolvimento mas que futuramente podem ser realizadas para torná-lo mais robusto. Dentre essas ideias destacam-se:

- Expandir o sistema desenvolvido de forma a possibilitar a catalogação e busca de objetos que utilizam outro padrão diferente do UMBRELO. Para tal, modificar o XML Schema configurado no FPOA e adicionar novas estratégias de busca;
- Buscar no próprio arquivo de descrição a coleção em que o objeto deve ser inserido na base de dados, retirando o único campo manual que o repositório permite a edição no momento da catalogação;
- Possibilitar o download de um arquivo compactado contendo todos os arquivos necessários para o funcionamento do OA;
- Coletar sugestões de melhorias na interface e usabilidade do sistema através de pesquisas com usuários reais do repositório.

REFERÊNCIAS

BASEX. **BaseX Documentation.** Disponível em: <http://docs.basex.org/wiki/Main_Page>. Acesso entre março de 2013 e junho de 2013.

BERKELEY DB XML. **Oracle Berkeley DB XML.** Disponível em: <http://www.oracle.com/technetwork/products/berkeleydb/index-083851.html>. Acesso entre março de 2013 e junho de 2013.

BIOE. **Banco Internacional de Objetos Educacionais.** Disponível em: <http://objetoseducacionais2.mec.gov.br/>. Acesso entre março de 2013 e junho de 2013.

BOURRET, Ronald. **XML and Databases.** Disponível em: <http://www.rpbourret.com>. Acesso entre março de 2013 e junho de 2013.

CESTA. **Coletânea de Entidades de Suporte ao uso de Tecnologia na Aprendizagem.** Disponível em: <http://cesta.cinted.ufrgs.br>. Acesso entre março de 2013 e junho de 2013.

CORDEIRO, R. A. **Um framework de persistência de objetos de aprendizagem para catalogação em repositórios.** 2009. Dissertação (Mestrado em Engenharia de Produção) - UENF, Rio de Janeiro.

EXISTDB. **Exist-DB Open Source Native XML Database.** Disponível em: <http://www.exist-db.org/exist/apps/homepage/index.html> Acesso entre março de 2013 e junho de 2013.

FIREBIRD. **Firebird Database.** Disponível em: <http://www.firebirdsql.org/>. Acesso entre março de 2013 e junho de 2013.

GAMMA, E.; HELM, R.; JOHNSON, R.; VLISSIDES, J. **Design Patterns: Elements os Reusable Object-Oriented Software.** Ed. Addison Wesley, 1995.

IEEE - LTSC. **IEEE Standard 1484.12.1**. Standard for Learning Object Metadata. Nova York, 2002.

KLEMANN, M.; REATEGUI, E.; RAPKIEWICZ, C. Análise de Ferramentas de Mineração de Textos para Apoio à Produção Textual. **Anais do XXII SBIE – XVII – WIE**. Aracaju, Nov 2011.

LABVIRT. **Laboratório Didático Virtual.** Disponível em: <http://www.labvirtq.fe.usp.br/indice.asp >. Acesso entre março de 2013 e junho de

2013.

NUNES, César. **Objetos de Aprendizagem em Ação.** Cadernos de Pesquisa Reflexões. NEA/FEA/USP, Vol 1, nº1, 2004.

ORACLE. **Your First Cup: An Introduction to the Java EE Platform.** Abril 2012. Redwood City, California. Disponível em: <http://docs.oracle.com/javaee/6/firstcup/doc/gkhoy.html>. Acesso entre março de 2013 e junho de 2013.

PRESSMAN, R. **Engenharia de Software.** 6ª ed. Editora: McGraw-Hill, 2006.

SANTOS, N. S. R. S.; CORDEIRO, R. A.; NUNES, R. M. B.; RAPKIEWICZ C. E.; WIVES, L. K. Metadados para Objetos de Aprendizagem: prova de conceito do modelo UMBRELO. **SBIE 2012 – Trilha 7: Modelagem, representação, armazenamento e recuperação de conteúdos educacionais.**

SEDNA. **Sedna XML Database.** Disponível em: <http://www.sedna.org>. Acesso entre março de 2013 e junho de 2013.

TANEMBAUM, A. S. **Redes de Computadores.** 4ª ed. Rio de Janeiro: Campus, 2003.

WILEY, D. A. **Connecting learning objects to instructional design theory: A definition, a metaphor, and a taxonomy.** 2002. Disponível em: <http://www.reusability.org/read/chapters/wiley.doc>. Acesso entre março de 2013 e junho de 2013.

APÊNDICE SCHEMA DE VALIDAÇÃO

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified"
xmlns:xsi="http://www.w3.org/2001/XMLSchemainstance">
  <xs:import namespace="http://www.w3.org/2001/XMLSchema-instance"
    schemaLocation="xsi.xsd" />
  <xs:element name="object">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Rived" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Rived">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="LoMetadata" />
        <xs:element ref="LOContents" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="LoMetadata">
    <xs:complexType mixed="true">
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element ref="Anotation" />
        <xs:element ref="Classification" />
        <xs:element ref="Educational" />
        <xs:element ref="General" />
        <xs:element ref="LifeCycle" />
        <xs:element ref="MetaMetadata" />
        <xs:element ref="Relation" />
        <xs:element ref="Rights" />
        <xs:element ref="Technical" />
      </xs:choice>
    </xs:complexType>
  </xs:element>
  <xs:element name="Anotation">
    <xs:complexType>
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element name="entity" type="xs:string" />
        <xs:element name="date" type="xs:string" />
        <xs:element name="description" type="xs:string" />
      </xs:choice>
    </xs:complexType>
```

```
</xs:element>
<xs:element name="Classification">
  <xs:complexType>
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element name="purpose" type="xs:string" />
      <xs:element name="keyword" type="xs:string" />
      <xs:element name="description" type="xs:string" />
    </xs:choice>
  </xs:complexType>
</xs:element>
<xs:element name="Educational">
  <xs:complexType>
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element name="interactivityType" type="xs:string" />
      <xs:element name="description" type="xs:string" />
      <xs:element name="language" type="xs:string" />
      <xs:element name="context" type="xs:string" />
      <xs:element name="interactivityLevel" type="xs:string" />
      <xs:element name="resource_type" type="xs:string" />
      <xs:element name="typicalAgeRange" type="xs:string" />
    </xs:choice>
  </xs:complexType>
</xs:element>
<xs:element name="General">
  <xs:complexType>
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element name="identifierCatalog" type="xs:string"
        minOccurs="0" />
      <xs:element name="identifierEntry" type="xs:string" />
      <xs:element name="title" type="xs:string" />
      <xs:element name="language" type="xs:string" />
      <xs:element name="description" type="xs:string" />
      <xs:element name="keyword" type="xs:string" />
      <xs:element name="discipline" type="xs:string" />
      <xs:element name="module" type="xs:string" />
    </xs:choice>
  </xs:complexType>
</xs:element>
<xs:element name="LifeCycle">
  <xs:complexType>
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element name="version" type="xs:string" />
      <xs:element name="status" type="xs:string" />
      <xs:element name="contributes">
        <xs:complexType>
          <xs:choice minOccurs="0" maxOccurs="unbounded">
            <xs:element name="role" type="xs:string" />
            <xs:element name="entity" type="xs:string" />
            <xs:element name="date" type="xs:string" />
          </xs:choice>
        </xs:complexType>
      </xs:element>
    </xs:choice>
  </xs:complexType>
</xs:element>
<xs:element name="MetaMetadata">
  <xs:complexType>
    <xs:choice minOccurs="0" maxOccurs="unbounded">
```

```
      <xs:element name="identifierCatalog" type="xs:string" />
      <xs:element name="identifierEntry" type="xs:string" />
      <xs:element name="schema" type="xs:string" />
    </xs:choice>
  </xs:complexType>
</xs:element>
<xs:element name="Relation">
  <xs:complexType>
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element name="resource">
        <xs:complexType>
          <xs:choice minOccurs="0" maxOccurs="unbounded">
            <xs:element name="kind" type="xs:string" />
            <xs:element name="identifier_catalog" type="xs:string" />
            <xs:element name="identifier_entry" type="xs:string" />
            <xs:element name="description" type="xs:string" />
          </xs:choice>
        </xs:complexType>
      </xs:element>
    </xs:choice>
  </xs:complexType>
</xs:element>
<xs:element name="Rights">
  <xs:complexType>
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element name="cost" type="xs:string" />
      <xs:element name="copyright" type="xs:string" />
      <xs:element name="description" type="xs:string" />
    </xs:choice>
  </xs:complexType>
</xs:element>
<xs:element name="Technical">
  <xs:complexType>
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element name="format" type="xs:string" />
      <xs:element name="size" type="xs:string" />
      <xs:element name="location" type="xs:string" />
      <xs:element name="requirementType" type="xs:string" />
      <xs:element name="requirementName" type="xs:string" />
      <xs:element name="font" type="xs:string" />
      <xs:element name="stage" type="xs:string" />
    </xs:choice>
  </xs:complexType>
</xs:element>

<xs:element name="LOContents">
  <xs:complexType mixed="true">
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element ref="CommonSceneElements" />
      <xs:element ref="Scene" />
    </xs:choice>
  </xs:complexType>
</xs:element>

<xs:element name="CommonSceneElements">
  <xs:complexType mixed="true">
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element ref="Text" />
```

```
      <xs:element ref="ConceitualMap" />
      <xs:element ref="Glosary" />
      <xs:element ref="Exercises" />
      <xs:element ref="Credits" />
      <xs:element ref="Button" />
    </xs:choice>
  </xs:complexType>
</xs:element>
<xs:element name="ConceitualMap">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="src" type="xs:string" />
      <xs:element ref="Button" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="Credits">
  <xs:complexType>
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element name="src" type="xs:string" />
      <xs:element ref="Contribution" />
      <xs:element ref="Text" />
      <xs:element ref="Button" />
    </xs:choice>
  </xs:complexType>
</xs:element>
<xs:element name="Contribution">
  <xs:complexType>
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element name="name" type="xs:string" />
      <xs:element name="role" type="xs:string" />
      <xs:element name="title" type="xs:string" />
      <xs:element name="entity" type="xs:string" />
    </xs:choice>
  </xs:complexType>
</xs:element>
<xs:element name="Glosary">
  <xs:complexType>
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element name="title" type="xs:string" />
      <xs:element name="term">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="concept" type="xs:string" />
            <xs:element name="definition" type="xs:string" />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element ref="Button" />
    </xs:choice>
  </xs:complexType>
</xs:element>

<xs:element name="Scene">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Scenary" maxOccurs="unbounded" />
    </xs:sequence>
```

```
    <xs:attribute name="frame" use="required" />
    <xs:attribute name="id" use="required" />
    <xs:attribute name="name" use="required" />
  </xs:complexType>
</xs:element>
<xs:element name="Scenary">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Content" />
    </xs:sequence>
    <xs:attribute name="id" use="required" type="xs:integer" />
  </xs:complexType>
</xs:element>
<xs:element name="Content">
  <xs:complexType>
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element ref="Text" />
      <xs:element ref="image" />
      <xs:element ref="Button" />
      <xs:element ref="Table" />
      <xs:element ref="Exercises" />
      <xs:element ref="Credits" />
    </xs:choice>
  </xs:complexType>
</xs:element>
<xs:element name="Text">
  <xs:complexType>
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element name="text" type="xs:string" />
      <xs:element name="font" type="xs:string" />
      <xs:element name="size" type="xs:string" />
      <xs:element name="category" type="xs:string" />
    </xs:choice>
    <xs:attribute name="id" use="required" />
    <xs:attribute name="posX" use="required" />
    <xs:attribute name="posY" use="required" />
    <xs:attribute name="sequence" use="required" />
  </xs:complexType>
</xs:element>
<xs:element name="image">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="extension" type="xs:string" />
      <xs:element name="path" type="xs:string" />
      <xs:element name="width" type="xs:string" />
      <xs:element name="height" type="xs:string" />
    </xs:sequence>
    <xs:attribute name="id" use="required" />
    <xs:attribute name="posX" use="required" />
    <xs:attribute name="posY" use="required" />
    <xs:attribute name="sequence" use="required" />
  </xs:complexType>
</xs:element>
<xs:element name="Button">
  <xs:complexType mixed="true">
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element name="name" type="xs:string" />
      <xs:element name="text" type="xs:string" />
```

```
        <xs:element name="action" type="xs:string" />
        <xs:element name="help" type="xs:string" />
        <xs:element name="type" type="xs:string" />
      </xs:choice>
      <xs:attribute name="id" use="required" />
      <xs:attribute name="posX" use="required" />
      <xs:attribute name="posY" use="required" />
      <xs:attribute name="sequence" use="required" />
    </xs:complexType>
  </xs:element>
  <xs:element name="Table">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="tr" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="td" maxOccurs="unbounded" />
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="id" use="required" />
    </xs:complexType>
  </xs:element>
  <xs:element name="Exercises">
    <xs:complexType>
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element ref="Question" />
        <xs:element ref="Button" />
        <xs:element name="feedback" type="xs:string" />
      </xs:choice>
    </xs:complexType>
  </xs:element>
  <xs:element name="Question">
    <xs:complexType>
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element name="text" type="xs:string" />
        <xs:element name="correctAnswer" type="xs:string" />
        <xs:element name="answer">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="text" type="xs:string" />
              <xs:element name="feedback" type="xs:string" />
            </xs:sequence>
            <xs:attribute name="id" use="required" />
          </xs:complexType>
        </xs:element>
      </xs:choice>
      <xs:attribute name="id" use="required" />
    </xs:complexType>
  </xs:element>

</xs:schema>
```