UNIVERSIDADE FEEVALE

MARCEL KLEIN PASSOS

SPARQL NA IMPLEMENTAÇÃO DO QUIZ ONTOMÚSICA

Novo Hamburgo
2010

MARCEL KLEIN PASSOS

SPARQL NA IMPLEMENTAÇÃO DO QUIZ ONTOMÚSICA

Trabalho de Conclusão de Curso
apresentado como requisito parcial
à obtenção do grau de Bacharel em
Sistemas de Informação pela
Universidade Feevale

Professor orientador: Rodrigo Rafael Villarreal Goulart

Novo Hamburgo

2010

AGRADECIMENTOS

Agradeço a meu professor orientador, minha família, esposa, amigos e colegas. Todos aqueles que de alguma maneira contribuíram para que fosse possível fazer este trabalho.

RESUMO

Com o grande crescimento da web ao longo dos anos, uma das áreas que mais tem se beneficiado com isso é a educação, com o surgimento de diversas ferramentas voltadas para a mesma, como é o caso dos sistemas de educação à distância. Uma das alternativas mais interessantes para uso na construção destes sistemas são as ontologias, amplamente discutidas atualmente. Este trabalho utiliza como base o Quiz Ontomúsica, um sistema de educação musical à distância que utiliza uma ontologia e que possui diversas limitações. Sendo assim, o trabalho busca fazer a análise da ontologia e do sistema, além do desenvolvimento de uma nova versão do mesmo, visando assim atingir grandes melhorias que permitam que o mesmo cumpra sua função utilizando todo o conteúdo expresso na ontologia.

Palavras-chave: Ontologia. Sistema de educação à distância.

ABSTRACT

With the huge growth of the web over the years, one area that has most benefited with this is education, with the creation of several tools made for her, just like the case of the distance education systems. One of the most interesting alternatives to use in the construction of these systems are the ontologies, widely discussed today. This paper focus on the Quiz Ontomúsica, a musical distance education system that is based on an ontology and has several limitations. Thus, this paper aims to make an analysis of the ontology and the entire system, besides developing a new version of it, thus aiming to achieve major improvements to make the system meet his real goal, using all of the data stored in the ontology.

Key words: Ontology. Distance Education System.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
EAD	Educação à Distância
eRDF	Embedded Resource Description Framework
HTML	Hyper text Markup Language
IA	Inteligência Artificial
N3	Notation 3 Serialisation format for RDF
OWL	Web Ontology Language
PHP	PHP Hypertext Preprocessor
RDF	Resource Description Framework
SBC	Sistema Baseados em Conhecimento
SE	Sistema Especialista
SPARQL	SPARQL Protocol and RFF Query Language
SQL	Structured Query Language
W3C	World Wide Web Consortium
XML	Extensible Markup Language
XSL	Extensible Stylesheet Language

SUMÁRIO

INTRODUÇÃO

A internet, ao longo dos anos, deixou de ser apenas um recurso facilitador para tornar-se uma ferramenta indispensável na execução de diversas tarefas. Seu grande crescimento e importância fizeram com que a busca por soluções para torná-la ainda mais funcional tenha se tornado um grande foco de estudos.

Por consequência disto, diversas áreas começaram a colher frutos pelo grande crescimento da web. Uma delas é a educação à distância, que cada vez mais se torna extremamente importante para alunos e professores, superando barreiras como a distância entre as pessoas, que antes trazia muitas dificuldades por conta das poucas opções para contornar este problema.

De acordo com a ABED[1] (Associação Brasileira de Educação a Distância), a educação à distância é a modalidade que mais cresce no ensino superior brasileiro. Em 2008 o número de estudantes de graduação foi de 760.599, aumento de 91% em relação a 2007. Comparando-se desde o ano de 2004, o crescimento foi de incríveis 1.175%, sendo que o aumento de matrículas presenciais neste mesmo período foi de apenas 17%.

Considerando-se os dados apontados anteriormente e também o fato de que a educação a distância não é um recurso utilizado apenas por instituições de ensino, mas também por empresas que buscam soluções para treinamento de funcionários, conclui-se que a área em questão é uma ótima opção de investimento, já que a busca por soluções neste segmento deve ser cada vez mais frequente.

Assim volta-se a atenção para a educação musical, uma das áreas que talvez mais se beneficiasse com soluções à distância, pois se sabe que esta área é ampla e costuma exigir muita atenção e dedicação das pessoas que a estudam, não apenas das que buscam aprendizado musical prático, mas também das que buscam conhecimento sobre a história da música. Assim, o desenvolvimento de ambientes na web que permitam as pessoas adquirirem conhecimento e estimularem sua curiosidade seriam provavelmente muito bem aceitos na área.

---

[1] http://www2.abed.org.br/

Com base na ideia de permitir o ensino musical à distância, surgiu o Quiz Ontomúsica, sistema que criado por Rogério Eduardo Boff, que funciona por meio de uma ontologia. Seu objetivo foi de criar um ambiente na internet que tornasse possível testar os conhecimentos do usuário sobre a história da música e principalmente fazê-lo aprender sobre o assunto, permitindo interação entre o usuário e o conhecimento disponibilizado no sistema.

O trabalho de Boff foi desenvolvido até certo ponto e manteve diversas questões em aberto para desenvolvimento futuro. Novos meios de inferência, a busca por outros motores de inferência e a utilização de linguagem natural são alguns dos melhoramentos vistos como necessários pelo desenvolvedor da aplicação. Consequentemente, analisando a ontologia do sistema, percebe-se que a mesma possui diversas limitações surgidas de seu processo de elaboração, que a tornam estática e limitam muito a maneira como o sistema funciona.

Assim, faltam-lhe melhorias que envolvam novas inferências, motivo pela qual este trabalho busca promover a análise da ontologia utilizada pelo Quiz Ontomúsica e a reconstrução do mesmo, com o objetivo de torná-lo mais consistente e dinâmico, utilizando todo o conhecimento expresso na ontologia.

Sendo assim, este trabalho foi dividido da seguinte maneira: No capítulo 1 são apresentados diversos conceitos básicos que envolvem o assunto deste trabalho, como a educação à distância, os sistemas inteligentes e principalmente as ontologias, tratando-se de seus principais tipos, ferramentas e linguagens utilizadas no seu desenvolvimento, além de como é o processo de criação de uma ontologia.

O capítulo 2 trata exclusivamente do Quiz Ontomúsica, onde é apresentado o processo de criação da ontologia do sistema, uma análise do sistema, abordando sua construção e funcionamento, além de suas limitações, que estimularam o desenvolvimento deste trabalho.

No capítulo 3 serão apresentadas algumas das principais linguagens e sistemas existentes para consultas a ontologias, abordando diversos aspectos de seu uso, como suas vantagens, desvantagens, particularidades referentes à implementação e funcionamento, além de exemplos de uso.

No capítulo 4 é feita a descrição do ambiente de desenvolvimento da nova versão do Quiz Ontomúsica. Nele são apresentadas as ferramentas e recursos utilizados em sua construção, além de pesquisa, comparações e testes de APIs – que ofereçam suporte a linguagem SparQL - para a linguagem escolhida para o desenvolvimento do novo Quiz. Já no capítulo 5 é descrito todo o processo de criação do novo Quiz Ontomúsica em etapas.

Por fim, no capítulo 6 são feitas as considerações finais sobre os resultados do trabalho, problemas encontrados e possíveis propostas para o futuro deste trabalho.

1 CONCEITOS BÁSICOS

Tendo-se em vista que o trabalho a ser desenvolvido envolve a análise e reconstrução de um sistema de educação musical à distância, torna-se necessário abordar alguns conceitos gerais relacionados a assuntos que envolvem o trabalho.

1.1. Educação à distância

Devido aos diversos problemas encontrados atualmente na sociedade, a educação torna-se cada vez mais importante à nível mundial, não apenas em território brasileiro. Questões como o desemprego e a grande concorrência por vagas fazem com que cada vez mais seja importante para o indivíduo ter um bom nível de preparação e conhecimento. Assim, novos investimentos na área de educação são feitos com frequência, com o objetivo de oferecer a alunos e professores novas opções e modalidades de ensino.

Inegavelmente a expansão da internet abriu muitas opções neste sentido, já que o desenvolvimento de ferramentas de comunicação ajudou a diminuir a distância existente entre pontos remotos que antes sofriam com a falta de acesso às instituições de educação. Esta evolução torna-se ainda um grande benefício para a sociedade pelo intercâmbio cultural que proporciona a todos.

Assim, com o uso cada vez mais frequente dos ambientes de ensino à distância (EAD), áreas antes acostumadas aos tradicionais métodos de ensino presencial, que envolvem grupos de alunos dentro de uma mesma sala, começam a receber opções novas de ensino, como é o caso da educação musical.

Até então era bastante rara a existência de ambientes voltados ao ensino musical à distância. As opções encontradas eram websites dedicados a conteúdo musical, que disponibilizavam material dedicado ao usuário que já possuía conhecimento teórico e/ou prático. Tais websites possuem conteúdo teórico rico, mas nenhum tipo de interação com o usuário, que busca sempre encontrar opções atrativas e que ofereçam fácil aprendizado.

O desenvolvimento de ferramentas de ensino musical à distância que tenham a capacidade de atrair a atenção do usuário, ao mesmo tempo em que o faz adquirir conhecimento, torna-se então uma excelente opção para auxiliar o crescimento da área de educação musical.

Para melhor entendimento, na próxima seção serão discutidos os sistemas inteligentes, diretamente envolvidos com o objetivo deste trabalho.

1.2. Sistemas inteligentes

A inteligência artificial (IA) é uma área de pesquisa dentro da ciência da computação em constante evolução, cujo objetivo é criar métodos ou dispositivos com a capacidade de simular a inteligência humana. Criar máquinas que tenham essa habilidade é do interesse da humanidade há muito tempo, pois assim elas teriam capacidade para auxiliar a resolver inúmeras questões difíceis para o ser humano, mas que para elas seriam facilitadas pelo poder de analisar e tomar decisões encontradas na mente humana.

Existem diversas áreas onde problemas podem ser resolvidos somente por especialistas, ou seja, pessoas que possuem o conhecimento específico, adquirido ao longo do tempo dedicado ao estudo da mesma. Entre essas áreas encontram-se a computação, diagnose médica, eletrônica, etc.

Assim, para que problemas de grande complexidade nestas áreas sejam resolvidos, depende-se de conhecimento específico sobre o problema em questão. Consequentemente, para que se torne possível às maquinas auxiliarem na resolução destes problemas complexos, estas devem compreender o conhecimento específico que o ser humano possui sobre as mais diversas áreas. É justamente aí que se insere o sistema inteligente.

Sistemas inteligentes têm sido tentativas de representar o conhecimento humano na forma de sistemas computacionais. Seus principais objetivos são eliminar as consultas aos especialistas e auxiliar a tomada de decisão, além de permitir que outras pessoas tenham acesso a esse conhecimento também.

Existem basicamente dois tipos de sistemas inteligentes: Sistemas Baseados em Conhecimento e Sistemas Especialistas. De forma geral, pode-se dizer que os primeiros são sistemas capazes de resolver problemas utilizando sistema específico sobre o domínio da aplicação, enquanto os Sistemas Especialistas são SBC’s que resolvem problemas ordinariamente resolvidos por um especialista humano (REZENDE, 2005).

De acordo com Boff (2005), um sistema inteligente pode ter uma arquitetura formada por três módulos: uma memória de trabalho, que pode conter qualquer tipo de estrutura de dados; uma base de regras, que contém as condições que representam perguntas à

representação de conhecimento da memória de trabalho e juntamente com a mesma forma a base de conhecimento do sistema; e um motor de inferência, que controla a atividade do sistema.

Sistemas inteligentes atualmente podem ser construídos com o emprego de linguagens como XML e OWL, utilizadas para criação de ontologias, assunto que será discutido na próxima seção.

1.3. Ontologias

Na utilização de sistemas especialistas, o foco foi por muito tempo o problema, e não o conhecimento, algo que tornava o aproveitamento de conhecimento já formalizado praticamente impossível, pois:

> [...] toda vez que um sistema especialista tivesse que ser construído em um mesmo domínio, mas com o objetivo de realizar uma diferente tarefa, todo o processo de elicitação e codificação do conhecimento deveria ser refeito, expondo o processo a erros e inconsistências que já poderiam ter sido resolvidas, além de provocar perda de tempo, esforço e consequentemente recursos (GUIZZARDI, 2000, p. 37).

Assim, durante os anos 80, a busca por métodos para a construção de soluções que fossem capazes de fazer a reutilização de conhecimento já disponível tornou-se um dos objetivos dos pesquisadores da computação. As pesquisas começaram em uma área antes explorada apenas pela filosofia, criada inicialmente por Aristóteles, através de seu sistema de classificação, taxonomização e de representação do conhecimento de forma geral, também conhecida como Ontologias (GUIZZARDI, 2000).

Segundo Breitman (2005), O vocábulo ontologia foi introduzido no estudo da filosofia de modo a fazer uma distinção entre o estudo do ser e o estudo dos vários tipos de seres vivos existentes no mundo natural. Enquanto disciplina da área de filosofia, o objetivo da Ontologia é o fornecimento de sistemas de categorização para organizar a realidade.

Já Gruber (2007), afirma que o termo “Ontologia” vem da área da filosofia que se preocupa com o estudo do ser e da existência. Conforme o autor, na filosofia, pode-se falar de uma ontologia como uma teoria da natureza da existência, enquanto na ciência da computação, uma ontologia é um termo técnico que indica um artefato que é projetado para

uma finalidade, que é permitir a modelagem do conhecimento sobre um domínio, seja ele real ou fictício.

Existe uma grande quantidade de definições a respeito do termo ontologia, em virtude de diversas áreas utilizarem ontologias, como engenharia de software, representação de conhecimento, processamento de linguagem natural e principalmente inteligência artificial. Algumas vezes estas definições tornam-se um pouco contraditórias, devido a muitos autores discutirem este assunto em busca de um consenso, o que dificilmente acontece, assim como em (GUARINO, 1997).

Assim, a definição mais apropriada encontrada e que será utilizada como base neste trabalho, conceitua ontologia como:

> [...] uma ferramenta computacional composta de um vocabulário de conceitos, suas definições e propriedades, um modelo contendo todas as possíveis relações entre os conceitos, bem como de um conjunto de axiomas formais que restringem a interpretação dos conceitos e relações, representando de maneira clara e não ambígua o conhecimento de um domínio específico (GUARINO, apud BECKER, 2006, p. 32).

Assim, com a definição do assunto devidamente discutida, são apresentados na próxima seção os elementos que formam uma ontologia.

1.3.1. Elementos de uma ontologia

Conforme Uschold e Gruninger (1996), uma ontologia necessariamente implica em uma visão global a respeito de um determinado domínio. Esta visão é concebida por um conjunto de conceitos (entidades, atributos, processos), suas definições e relacionamentos.

Como acontece no caso das definições de ontologias, muitos autores citam elementos por vezes diferentes em suas obras. Assim, alguns dos elementos mais comuns de uma ontologia encontrados nas mais diversas obras são:

Conceito e definição de conceito: É o significado semântico de um conceito específico relacionado a um domínio específico. Exemplo: O termo “violão”, relacionado ao domínio “instrumentos”, pode ser definido como “instrumento de seis cordas que possui corpo oco”.

Propriedade: É um atributo de um conceito, que o caracteriza e diferencia de outras instâncias do mesmo dentro de um domínio. Cada conceito pode possuir diversas propriedades, e cada propriedade pode possuir uma lista de possíveis ou prováveis valores. Exemplo: para que o conceito violão seja caracterizado, podem ser atribuídas a ele as propriedades modelo, fabricante, tipo de corda, data de fabricação, etc.

Relação: Determina as relações entre os conceitos. Exemplo: o conceito "corda" é usado pelo conceito "violão".

Restrição: Expressa como os diversos conceitos de um domínio relacionam-se. Exemplo: O conceito "corda" poderá ser relacionado ao conceito "violão", mas não poderá ser relacionado ao conceito "teclado".

Estes elementos apresentados são bastante comuns no uso de ontologias, embora não sejam obrigatórios. As ontologias podem ter diversas formas, mas talvez a única restrição para seu funcionamento seja que as mesmas devem possuir necessariamente um vocabulário de conceitos e suas definições (USCHOLD E GRUNINGER, 1996).

1.3.2. Tipos de ontologias

Segundo Guizzardi (GUARINO, apud GUIZZARDI, 2000), com base em seu conteúdo as ontologias podem ser classificadas nas seguintes categorias:

- *Ontologias genéricas*: descrevem conceitos bastante gerais, tais como, espaço, tempo, matéria, objeto, evento, ação, etc., que são independentes de um problema ou domínio particular, com o intuito de serem especializados na definição de conceitos em uma ontologia de domínio. A pesquisa enfocando ontologias genéricas procura construir teorias básicas do mundo, de caráter bastante abstrato, aplicáveis a qualquer domínio (conhecimento de senso comum);
- *Ontologias de domínio*: expressam conceituações de domínios particulares, descrevendo o vocabulário relacionado a um domínio genérico, tal como Medicina e Direito. É o tipo mais comumente desenvolvido, sendo que diversos trabalhos são encontrados na literatura, enfocando áreas como química, modelagem de empreendimento, design, modelagem de processos de software, biologia molecular, bioquímica, ciência dos materiais, entre outros;

- *Ontologias de tarefas*: expressam conceituações sobre a resolução de problemas, independentemente do domínio em que ocorram, isto é, descrevem o vocabulário relacionado a uma atividade ou tarefa genérica, tal como, diagnose ou vendas. O estudo de ontologias de tarefas é a vertente mais recente do estudo de ontologias. Sua principal motivação é facilitar a integração dos conhecimentos de tarefa e domínio em uma abordagem mais uniforme e consistente, tendo por base o uso de ontologias;
- *Ontologias de aplicação*: descrevem conceitos dependentes do domínio e da tarefa particulares. Estes conceitos frequentemente correspondem a papéis desempenhados por entidades do domínio quando da realização de certa atividade;
- *Ontologias de representação*: explicam as conceituações que fundamentam os formalismos de representação de conhecimento.

Ainda de acordo com Guarino (1998), é proposto que as ontologias sejam construídas de acordo com seu nível de generalidade. Assim, os conceitos de uma ontologia de aplicação devem ser especializações dos termos das ontologias de domínio e/ou de tarefa, sendo que os destas devem ser especializações dos termos introduzidos por uma ontologia genérica. A Figura 1 mostra um esquema com os tipos de ontologias.

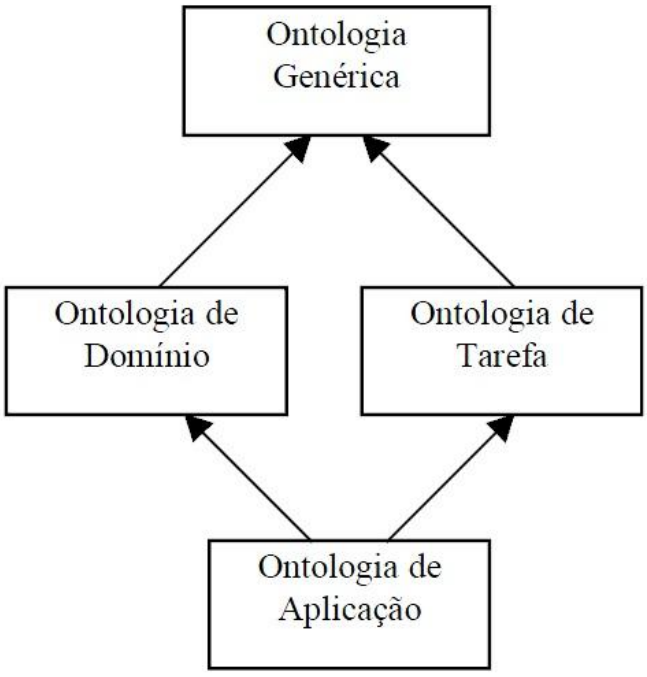


Figura 1 – Tipos de ontologias, de acordo com seu nível de generalidade.
Fonte: Guizzardi (2000, p. 40)

Conforme Uschold e Gruninger (1996), as ontologias também podem ser classificadas de acordo com seu grau de formalidade:

- Altamente informal: Quando é expressa livremente em linguagem natural;
- Estruturada informal: Quando é expressa em linguagem natural, de forma restrita e estruturada;
- Semiformal: Quando é expressa em uma linguagem artificial, definida formalmente;
- Rigorosamente formal: Quando é expressa com semântica formal, teoremas e provas.

1.3.3. Ferramentas e linguagens

Talvez um dos mais importantes aspectos envolvidos no desenvolvimento de uma ontologia seja a escolha de uma linguagem ou ferramenta apropriada para tal tarefa. Sendo que o foco deste trabalho envolve o correto entendimento do processo de criação de uma ontologia, torna-se necessária a correta avaliação das opções de ferramentas e linguagens disponíveis para este processo.

Entre as diversas linguagens de representação de ontologias encontramos as seguintes opções:

- *XML*: É um conjunto de regras para definir etiquetas semânticas, que divide um documento em partes e identifica as diferentes partes de um documento (HAROLD, apud CHAVES, 2001). XML é uma linguagem de marcação apropriada à representação de dados, documentos e entidades cujo foco baseia-se na capacidade de agregar informações (CHAVES, 2001);
- *OWL*: É utilizada para representar explicitamente o conjunto de termos de um vocabulário e os relacionamentos entre estes termos. É específica para escrita de ontologias e desenvolvida para aplicações que precisam processar o conteúdo de informações, ao invés de apenas apresenta-la (BOFF, 2005). Atualmente é padrão da W3C e possui grande quantidade de operadores;
- *RDF*: Fornece uma maneira de agregar informações semânticas a um documento sem referir-se a sua estrutura (FENSEL apud BECKER, 2005). Tudo o que é implementado em RDF também pode ser implementado em XML, porém RDF provê um padrão de representação de metadados, sendo que através de XML é possível ter diversas representações distintas (MUNÕZ apud BECKER, 2005).

Muitas opções existem quando o problema em questão é definir a linguagem a ser utilizada para a representação de ontologias. Mas para auxiliar na criação de uma ontologia deve-se optar por uma ferramenta que facilite esta tarefa.

Entre as ferramentas encontradas para a criação e edição de Ontologias encontramos a Protégé[2] e a Ontostudio[3], que é um produto mais recente no mercado, substituindo a ferramenta Ontoedit. Esta última apresentava-se na forma de um ambiente de engenharia de ontologias em que as fases de desenvolvimento eram divididas da seguinte maneira: Uma fase de especificação de requisitos, uma frase de refinamento e uma fase de avaliação (BOFF, 2005).

Já a ferramenta Protégé começou a ser desenvolvida no final dos anos 80. A ferramenta original era uma aplicação que tinha como objetivo construir ferramentas de aquisição de conhecimento para alguns programas especializados em planejamento médico. Atualmente, depois de muitos anos de desenvolvimento, é um software compatível com diversas plataformas (LICHTENSEIN, 2008). A Figura 2 mostra um exemplo de tela do Protégé.

O Protégé não é um sistema especialista, assim como não pode ser considerado um programa que constrói sistemas especialistas. Ele é uma ferramenta que auxilia usuários a desenvolverem suas próprias ferramentas que são adaptadas para auxiliar a aquisição de conhecimento para sistemas especialistas em áreas de aplicações específicas (MUSEN, apud LICHTENSTEIN, 2008).

---

[2] http://protege.stanford.edu

[3] http://www.ontoprise.de/en/home/products/ontostudio/

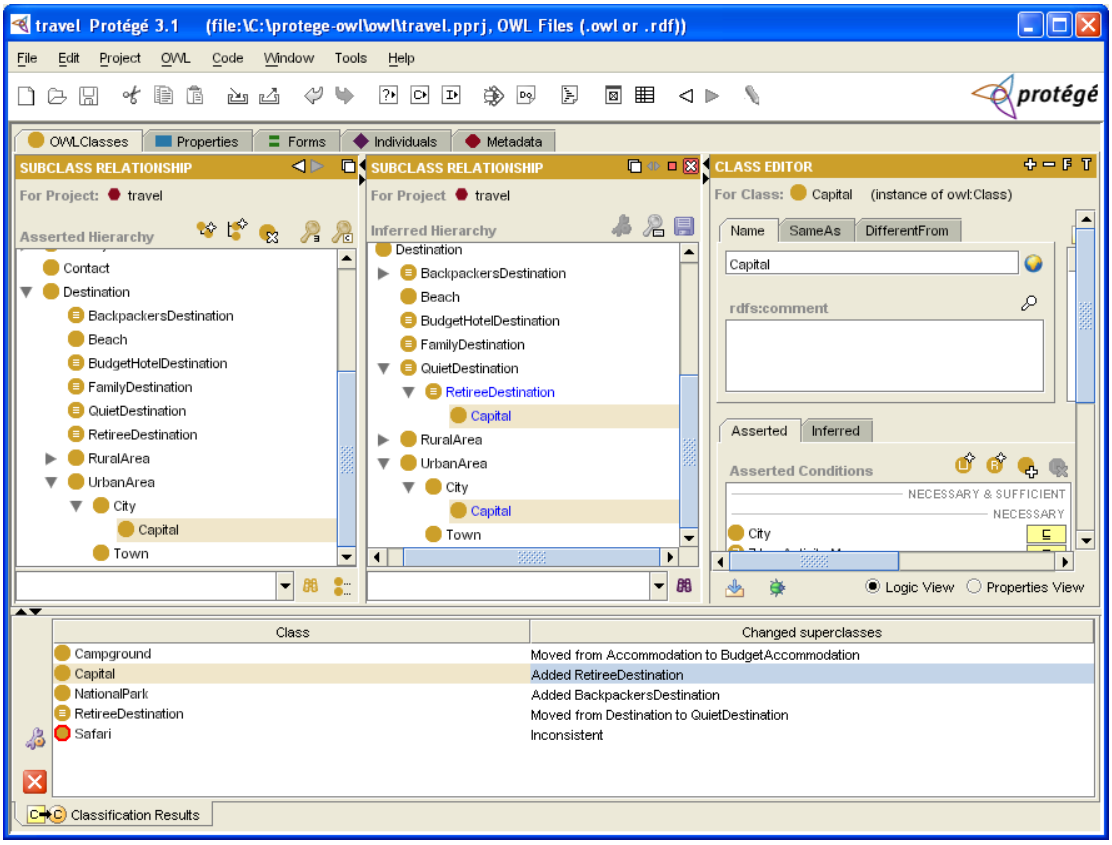


Figura 2 – Exemplo de tela do Software Protégé.

Entre suas características estão a facilidade de uso e a disponibilidade muitos recursos, como o uso de *plug-ins* para interação e compatibilidade com diversas ferramentas de programação e linguagens, como XML, RDF, OWL. Proporciona um ambiente de desenvolvimento operacional, flexível e robusto, sendo utilizado por diversos grupos de pesquisa, além de pessoas nas áreas de pesquisa e comercial.

Todas as ferramentas apontadas são capazes de criar e editar ontologias dos mais diversos tipos, utilizando as linguagens mais comuns de representação. Deu-se ênfase na descrição da ferramenta Protégé pelo fato da mesma ter sido utilizada por Boff na criação do Quiz Ontomúsica, que é o foco deste trabalho. Além disso, soma-se o fato de que a Ontoedit já não é mais disponibilizada no website da empresa desenvolvedora e a Ontostudio é muito recente, enquanto a ferramenta Protégé já é amplamente utilizada para seu propósito e existe bastante documentação disponível sobre seu uso.

1.3.4. Construindo ontologias

De acordo com Boff (2005), não existe caminho ou metodologia correta para a construção de uma ontologia, sempre sendo possível encontrar alternativas viáveis, dependendo do caso em questão. Assim, toma-se como base neste trabalho a ontologia criada por Boff para utilização no Quiz Ontomúsica, desenvolvida utilizando o Protégé e cuja metodologia será descrita no próximo capítulo.

2 O QUIZ ONTOMÚSICA

O sistema foi desenvolvido por Boff no ano de 2005 e apresentado em seu trabalho de conclusão do curso de Ciência da Computação, pela Universidade Feevale. O trabalho, que aborda todo o processo de desenvolvimento do Quiz Ontomúsica, serviu de motivação para o desenvolvimento deste trabalho, visando o aperfeiçoamento de uma ideia bastante original e muito relevante para a área.

O grande objetivo do Quiz Ontomúsica certamente foi o de permitir que qualquer pessoa, por meio do sistema, adquirisse conhecimento a respeito da história da música. Por isso, seu funcionamento é bastante simples e direto. Ele mostra ao usuário um questionamento criado pelo sistema e oferece três alternativas para resposta, sendo que apenas uma é a correta. Ainda recebe-se uma dica, caso a resposta dada seja uma alternativa errada. A Figura 3 mostra um exemplo de pergunta no Quiz Ontomúsica.

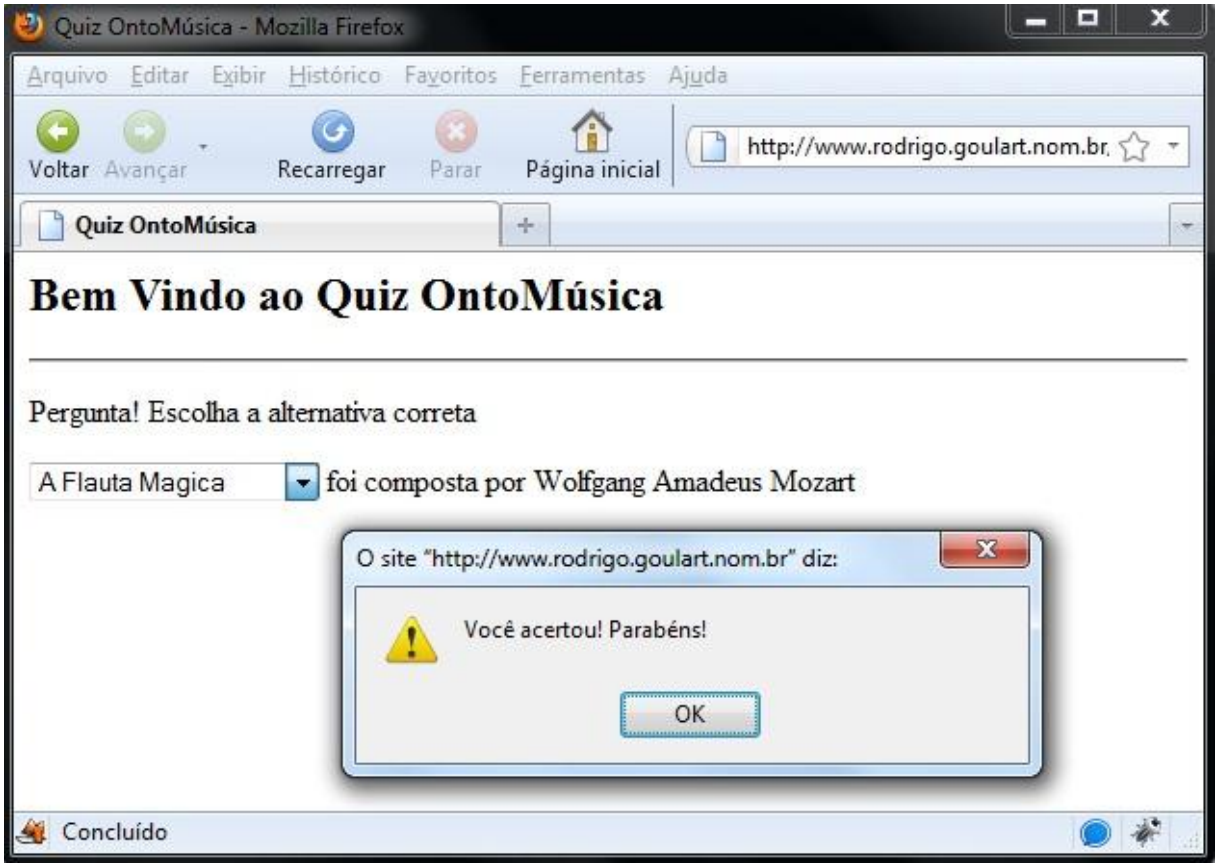


Figura 3 - Exemplo de pergunta no Quiz Ontomúsica.

Naturalmente, por ser um sistema cujo funcionamento ocorre por meio da web, o Quiz Ontomúsica envolveu diversas etapas em seu processo de desenvolvimento. Para superar todas estas etapas, que incluíram desde a criação da ontologia até o desenvolvimento do ambiente web, utilizaram-se diversas ferramentas e linguagens. O sistema foi inicialmente construído utilizando HTML, Javascript, XSL e o software Protégé. Este último usado para construção da ontologia, a base de funcionamento do mesmo, já que é a responsável por todo o conhecimento disponibilizado pelo Quiz.

É importante frisar que antes de criar uma nova ontologia deve-se pesquisar por ontologias existentes, pois a quantidade de ontologias criadas disponíveis na web é considerável, sendo assim possível encontrar alguma que seja apropriada para o uso desejado. No caso do Quiz Ontomúsica, o autor buscou opções que pudessem ser úteis, mas não encontrou nada que fosse realmente adequado, considerando-se o assunto do trabalho.

2.1. Criação da ontologia

Nesta etapa deve-se levar em consideração que ontologias baseadas em OWL têm componentes muito parecidos com a forma básica de se utilizar o Protégé, mas com uma terminologia levemente diferente. Uma ontologia OWL é formada por indivíduos, propriedades e classes, que correspondem a instâncias, slots e classes no Protégé (LICHTENSTEIN, 2008). Assim, não há risco de falta de entendimento a respeito de algumas terminologias que podem ser citadas no decorrer deste trabalho.

A ontologia utilizada pelo Quiz Ontomúsica também foi criada em etapas. A primeira etapa consistiu em definir os termos e conceitos relacionados ao domínio história da música, que consistia em compositores, gêneros musicais, instrumentos, obras, período, etc.

Em seguida foram definidas as classes, utilizando o método de combinação, para maior facilidade na criação da ontologia. Um exemplo de termo definido como classe é “Instrumento musical”, com as subclasses “Teclado”, “Sopro”, “Percussão” e “Corda”, este último com subclasses “Friccionada”, “Dedilhada” e “Percutida”. A Figura 4 mostra um exemplo de classes da ontologia do Quiz Ontomúsica.

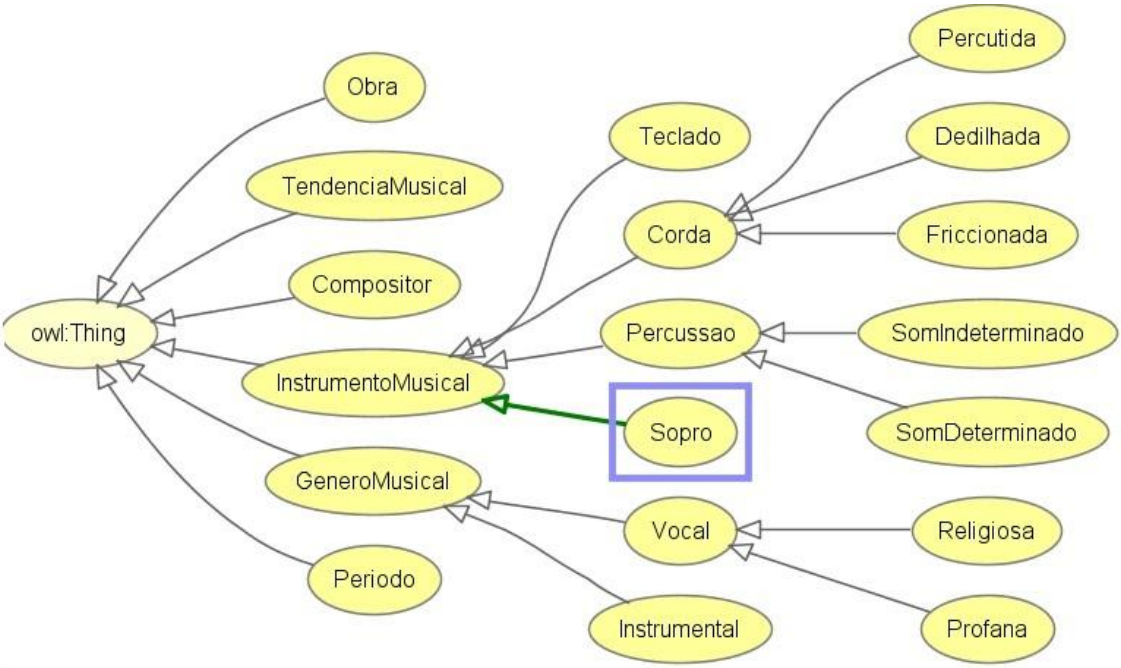


Figura 4 – Exemplo de classes da ontologia do Quiz Ontomúsica.

Depois foram definidos os atributos de cada classe, como por exemplo, a classe "Obra", que possui os atributos "nome", "caracteristicasdaobra", "anocomposicao", "foicompostapor (relacionada com a classe compositor)", e "generodaobra (relacionada com a classe GeneroMusical)". A Figura 5 mostra um exemplo de relacionamentos entre classes.

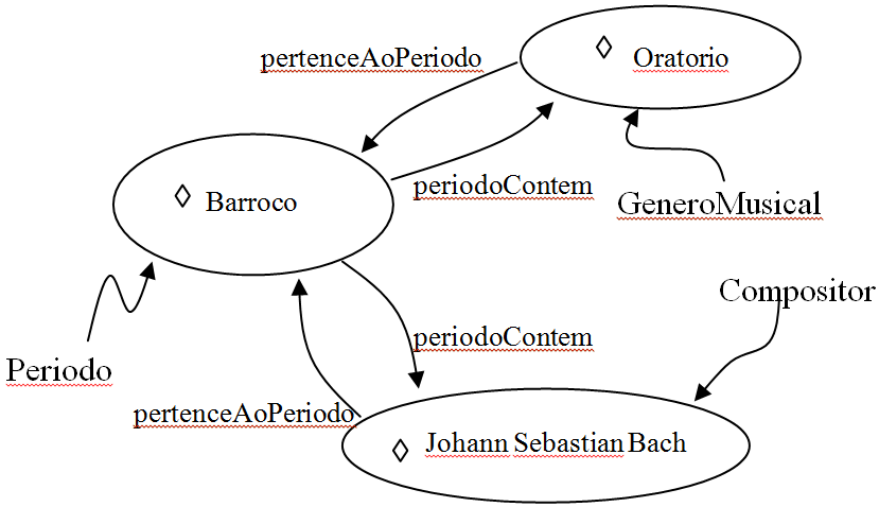


Figura 5 – Exemplo de relacionamentos entre classes.
Fonte: Boff (2005, p. 56).

Após essa etapa foram definidas as restrições dos atributos, sendo elas de cardinalidade e tipo. As restrições são importantes para permitir que, por exemplo, a classe período possa conter diversos compositores, gêneros e tendências. E por fim foram criadas as

instâncias com todas as informações referentes à história da música, todas retiradas de sites da internet.

2.2. Desenvolvimento da interface web

Com a criação da ontologia foi elaborada a interface e o mecanismo do Quiz Ontomúsica. A primeira etapa consistiu em converter a ontologia criada no protege para o formato OWL, utilizando-se uma função de conversão que é nativa do software, mas que nas versões antigas era um *plug-in*. A conversão da ontologia para o formato OWL inclui todos os elementos da mesma, como classes, atributos, instâncias e relacionamentos.

Após isso foram elaboradas folhas de estilo utilizando XSL. Essa etapa é muito importante no funcionamento do sistema, pois é ela que permite a criação de regras para que sejam feitas buscas de informações contidas no documento OWL e formatá-las em HTML. Utilizando-se o Java script foi possível gerar uma interface web interativa, que permite a visualização de toda a informação contida na ontologia criada.

O funcionamento do Quiz Ontomúsica acontece principalmente pela criação de regras de pesquisa feitas com Java script, que fazem toda a parte de pesquisa dos conteúdos. Existe uma regra, por exemplo, que busca três instâncias da classe obra e exibe-as ao usuário em ordem aleatória, o que evita que a resposta certa apareça sempre na mesma posição. Essa é uma das principais funções, diretamente relacionada com seu objetivo principal.

Assim, conclui-se que o Quiz Ontomúsica funciona de maneira bastante simples, mas cumprindo seu propósito de desafiar o usuário com questionamentos relacionados à história da música.

2.3. Limitações

O sistema apresenta-se como uma solução bastante simples e direta em seu propósito de interagir com o usuário sobre a história da música. Imediatamente notam-se alguns pontos que poderiam ser mais bem explorados no Quiz Ontomúsica, como por exemplo, o fato de a ontologia do mesmo possuir bastante informação, com muitas instâncias criadas, mas o sistema apenas ter questionamentos referentes à obra e o artista criador da mesma. Na dica que o usuário recebe caso erre a resposta percebe-se mais uma prova de como o sistema é estático, pois sempre é informado o gênero a qual a obra pertence.

Outro problema que se percebe é a falta de acentuação e caracteres especiais nas instâncias. De acordo com Boff (2005), isso foi necessário, pois quando o sistema era executado aconteciam erros ou os sinais não eram reconhecidos.

Assim, na sequência do trabalho serão estudados alguns métodos atuais de consultas a ontologias RDF/OWL, para que assim se estabeleça fundamentação suficiente para a elaboração de uma nova versão do Quiz Ontomúsica, que assim possa ser mais dinâmica, explorando todo o conteúdo da ontologia do sistema e tornando-o ainda mais interessante e intuitivo.

3 MÉTODOS DE CONSULTA A ONTOLOGIAS RDF/OWL

Ao desenvolver sistemas que utilizem ontologias e funcionem pela web, por meio do browser de internet, torna-se necessário utilizar um método que faça a comunicação entre o sistema e a ontologia, realizando consultas à mesma para retornar os dados solicitados pelo sistema. No caso do Quiz Ontomúsica, conforme visto anteriormente, são utilizadas as folhas de estilo com XSL para fazer a busca de informações no documento OWL.

Existem muitas opções atualmente para o uso nestas situações, onde se busca criar um ambiente que faça a interação do usuário com as informações da ontologia. Estas linguagens de programação são conhecidas como linguagens de consulta à RDF (RDF Query Languages). A maioria destas soluções são linguagens do tipo "SQL-Like", similares à linguagem SQL, fator que se torna importante por quase sempre dispensar a necessidade de total aprendizado de uma nova linguagem.

Entre as linguagens de consulta à RDF encontrados temos, por exemplo, as linguagens RuleML, Versa, RDFQ, SeRQL, RDQL e SPARQL, sendo estas três últimas linguagens SQL-Like. Como a intenção principal deste trabalho é disponibilizar uma nova versão do Quiz Ontomúsica, com melhorias que o tornem mais interativo e dinâmico, uma pesquisa foi feita a respeito de algumas linguagens. O objetivo desta foi avaliar diversos aspectos relacionados ao uso de tais linguagens, obtendo melhor entendimento a respeito das mesmas a fim de atingir o objetivo do trabalho.

3.1. SPARQL

SPARQL[4] é uma linguagem de consulta à RDF do tipo SQL-Like, que se tornou padrão e recomendação da W3C para consulta à RDF em 15 de janeiro de 2008. É uma linguagem totalmente orientada a dados, que recupera informações contidas em arquivos RDF, possibilitando inclusive a opção de combinar dados de arquivos de diferentes fontes.

[4] http://www.w3.org/TR/rdf-sparql-query/

A linguagem SPARQL segue a mesma estrutura de construção de arquivos RDF e é construída sobre *Triple Pattern* (Triplas), ou seja: Subject (Sujeito), Predicate (Predicado) e Object (Objeto). Algumas das principais cláusulas da linguagem SPARQL são:

- SELECT [DISTINCT]
- FROM (opcional)
- WHERE (opcional)
- ORDER BY (opcional)
- UNION (opcional – funcionamento diferente do SQL)

Existem também algumas cláusulas específicas da SPARQL:

- BASE: Define a URI base de um recurso;
- FILTER: Aplica um filtro sobre as linhas recuperadas pela consulta;
- LIMIT: Limita a quantidade de linhas recuperadas da consulta;
- OFFSET: Permite que seja aplicado um deslocamento sobre o conjunto de linhas recuperadas pela consulta;
- OPTIONAL: Permite que uma linha seja recuperada mesmo que não exista o valor de uma propriedade do RDF;
- PREFIX: Cria um "apelido" para a URI de um arquivo RDF/OWL.

Quando são criadas consultas em SPARQL, deve-se identificar as variáveis com os símbolos '?' e/ou '$'.

3.1.1. Ferramentas para consulta utilizando SPARQL

O Twinkle[5] é uma ferramenta bastante fácil de utilizar, mas limitadas em funcionalidades, permitindo ao usuário criar e salvar suas próprias consultas à RDF, além de oferecer algumas facilidades para definir a localização de recursos. A Figura 6 mostra como é a tela de consultas do Twinkle.

[5] http://www.ldodds.com/projects/twinkle/

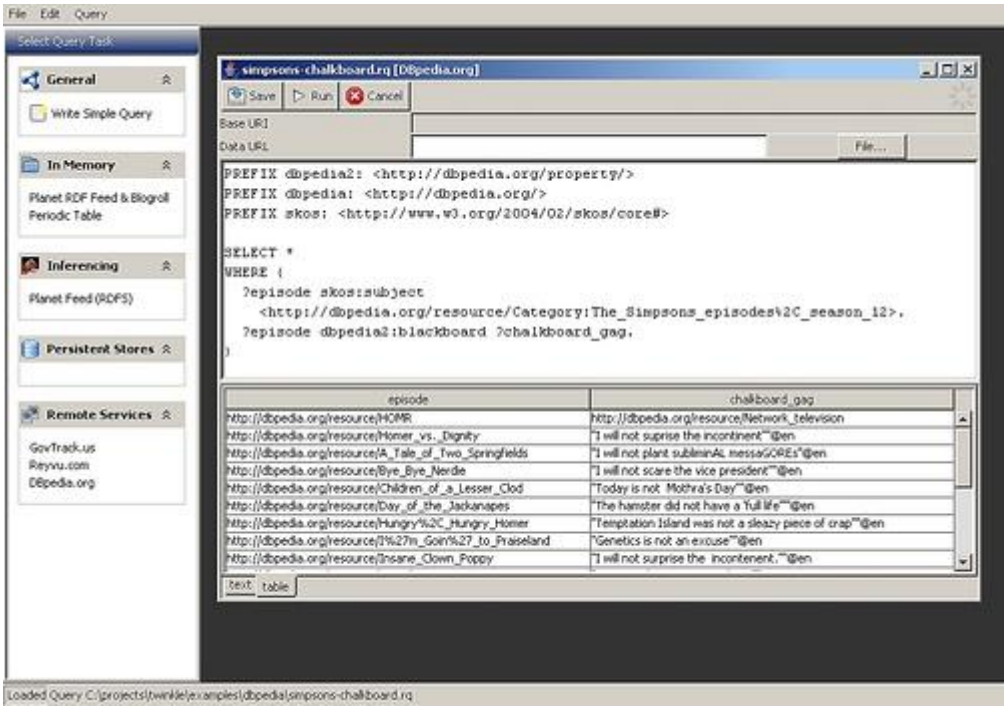


Figura 6 – Tela de consultas do Twinkle.

Outra ferramenta que pode ser utilizada para consultas com SPARQL é a Protégé[6]. Na versão 3.2 o próprio software possui um painel onde pode ser montada a pesquisa desejada utilizando a sintaxe do SPARQL apropriada. A Figura 7 mostra o painel de consultas SPARQL do Protégé. Na imagem pode ser vista o painel de visualização da ontologia na parte superior, o painel onde são inseridos os códigos da pesquisa SPARQL na parte inferior esquerda e o painel de visualização dos resultados da pesquisa na parte inferior direita.

Atualmente se encontra disponível no website do Protégé a versão 4.02 do software, que dá suporte parcial a versão 2.0 do formato OWL, mas que por outro lado ainda não possui suporte a SPARQL e outros recursos. Também está disponível a versão 4.1 Beta, que se encontra em desenvolvimento e já oferece suporte total ao formato OWL 2.0, apesar de também não suportar SPARQL.

[6] http://protege.stanford.edu/doc/sparql/

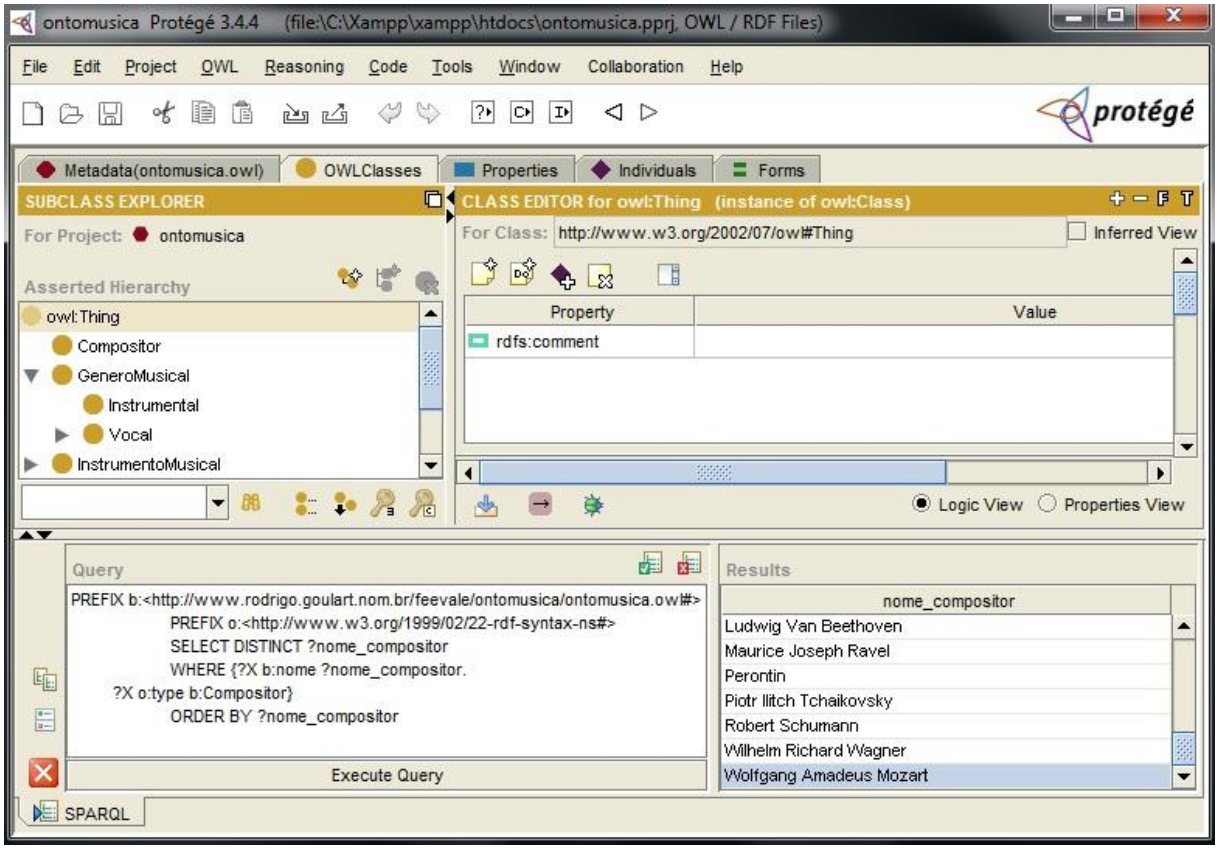


Figura 7 - Painel para consultas SPARQL no Protégé.

3.1.2. Exemplos de uso

Até agora foram descritas características diversas do SPARQL. Agora, para tornar o uso da linguagem mais clara, serão mostrados alguns exemplos de consultas utilizando SPARQL na ontologia do Quiz Ontomúsica, utilizando o próprio painel de consultas do Protégé. A Figura 8 mostra o código de uma consulta que lista todos os compositores e suas obras, ordenadas por nome:

```
PREFIX b:<http://www.rodrigo.goulart.nom.br/feevale/ontomusica/ontomusica.owl#>
SELECT ?nome_compositor ?nome_obra
WHERE { ?X b:compos ?Y.
        ?X b:nome ?nome_compositor.
        ?Y b:nome ?nome_obra}
ORDER BY ?nome_compositor
```

Figura 8 – Código de consulta de compositores e obras em SPARQL.

A Figura 9 mostra o resultado desta consulta, já aplicada no painel do Protégé. Pode-se notar a coluna de compositores com todos os nomes em ordem alfabética.

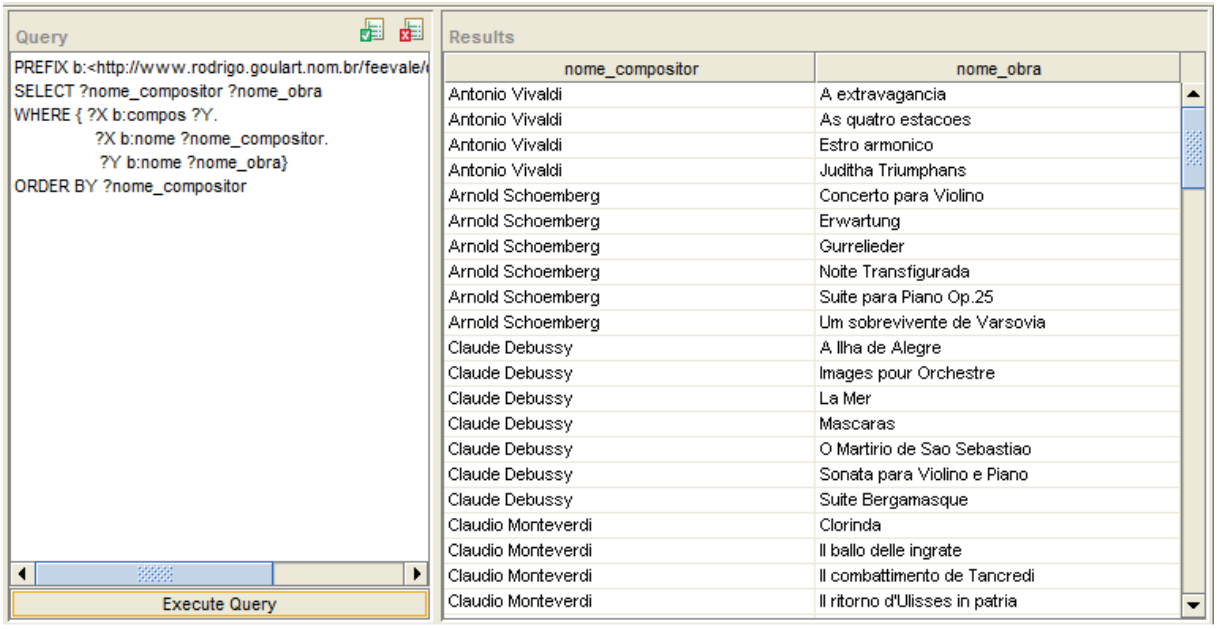


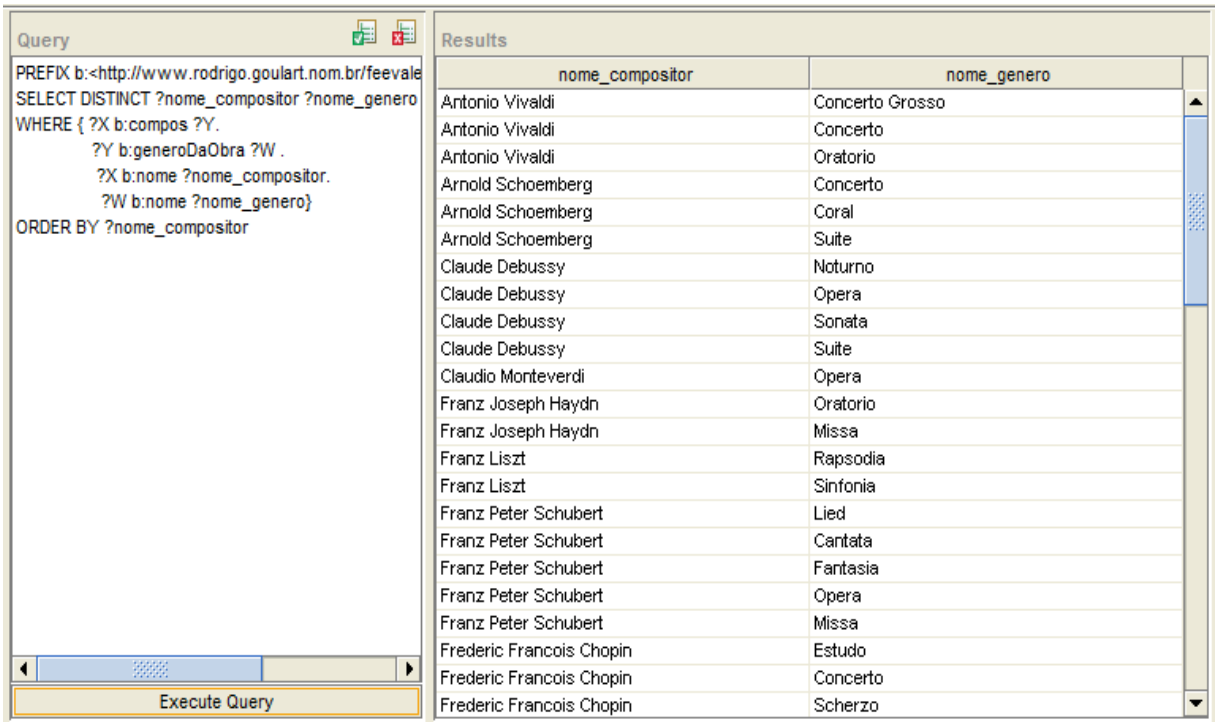
Figura 9 – Resultado da consulta de compositores e obras.

A Figura 10 mostra um exemplo de pesquisa um pouco mais complexo, onde são listados todos os gêneros em que cada compositor já criou alguma obra. No código nota-se o uso da cláusula *Distinct* após a cláusula *Select*. A função dela é fazer com que não haja repetições na listagem de gêneros, pois sem sua existência haveria diversas ocorrências de mesmos gêneros para cada compositor, em função da consulta ser feita com base em todas as obras de cada compositor.

```
PREFIX b:<http://www.rodrigo.goulart.nom.br/feevale/ontomusica/ontomusica.owl#>
SELECT DISTINCT ?nome_compositor ?nome_genero
WHERE { ?X b:compos ?Y.
        ?Y b:generoDaObra ?W.
        ?X b:nome ?nome_compositor.
        ?W b:nome ?nome_genero}
ORDER BY ?nome_compositor
```

Figura 10 – Consulta de gêneros das obras de cada compositor.

A Figura 11 mostra o resultado desta consulta, já aplicada no painel do Protégé. Pode-se notar que não há repetições de gêneros para nenhum compositor.



nome_compositor	nome_genero
Antonio Vivaldi	Concerto Grosso
Antonio Vivaldi	Concerto
Antonio Vivaldi	Oratorio
Arnold Schoemberg	Concerto
Arnold Schoemberg	Coral
Arnold Schoemberg	Suite
Claude Debussy	Noturno
Claude Debussy	Opera
Claude Debussy	Sonata
Claude Debussy	Suite
Claudio Monteverdi	Opera
Franz Joseph Haydn	Oratorio
Franz Joseph Haydn	Missa
Franz Liszt	Rapsodia
Franz Liszt	Sinfonia
Franz Peter Schubert	Lied
Franz Peter Schubert	Cantata
Franz Peter Schubert	Fantasia
Franz Peter Schubert	Opera
Franz Peter Schubert	Missa
Frederic Francois Chopin	Estudo
Frederic Francois Chopin	Concerto
Frederic Francois Chopin	Scherzo

Figura 11 – Resultado da consulta de gêneros das obras de cada compositor.

3.2. RDQL

RDQL[7] é uma linguagem de consulta à RDF do tipo SQL-Like que é totalmente orientada a dados. Suas cláusulas são muito similares a outras linguagens do mesmo tipo, mas servindo apenas para consultar, não sendo capaz de fazer inferências. Ela também opera na mesma estrutura de construção de arquivos RDF, ou seja, construída sobre *Triple Pattern*: Subject (Sujeito), Predicate (Predicado) e Object (Objeto).

A linguagem foi submetida à W3C pela HP em 2003 e sua última versão submetida é de 2004. Atualmente não é mais tão utilizada em função do uso de linguagens mais recentes como o SPARQL, que já possui a capacidade de fazer inferências. Assim, ambientes que ofereciam suporte a utilização do RDQL, como o JENA[8] – framework JAVA para construção de aplicações da web semântica – atualmente já oferecem suporte a revisões mais atuais do SPARQL.

[7] http://www.w3.org/Submission/RDQL/

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

3.2.1. Exemplos de uso

A Figura 12 mostra um exemplo de pesquisa com a linguagem RDQL. Na linha 1 do código utiliza-se *Select* para selecionar a variável "X" como o item a ser pesquisado. Na linha 2 do código utiliza-se *Where* para definir a condição da pesquisa, através da especificação de "X" como o sujeito a ser pesquisado, "http://www.w3.org/1999/02/22-rdf-syntax-ns#type" como predicado e "http://www.w3.org/2002/07/owl#Class" como objeto – presente na linha 3. Assim, o código pesquisa e retorna todas as triplas que possuam estas condições.

```
1 -     SELECT ?x
2 -     WHERE (?x,  <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>,
3 -                              <http://www.w3.org/2002/07/owl#Class>)
```

Figura 12 - Exemplo de consulta em RDQL.

Após a pesquisa realizada sobre as linguagens citadas no começo do capítulo, o SPARQL foi selecionado como a melhor opção de consulta a RDF a ser utilizada no desenvolvimento de uma nova versão do Quiz Ontomúsica. Os fatores que influenciaram nessa decisão foram a grande quantidade de APIs para as mais diversas linguagens de programação que já suportam o SPARQL – que serão descritas posteriormente, a constante pesquisa e atualização da mesma e o fato de ser a linguagem atualmente recomendada pela W3C.

As demais linguagens foram descartadas após a consideração de diversos fatores, como a falta de atualização ou até mesmo a interrupção de seu desenvolvimento, falta de documentação ou dificuldade de implementação e, no caso de algumas linguagens, o fato de terem sido diretamente substituidas pelo SPARQL em ambientes que ofereciam suporte a linguagem em questão.

No seguimento do trabalho será feita a definição do ambiente de criação do novo Quiz Ontomúsica: A linguagem de programação selecionada para o desenvolvimento da nova interface web, as ferramentas utilizadas para tal processo e também os testes realizados com APIs para permitir o uso do SPARQL com a linguagem web escolhida.

4 O AMBIENTE DE DESENVOLVIMENTO DO NOVO QUIZ ONTOMÚSICA

Na criação da nova versão do Quiz Ontomúsica optou-se pela utilização das linguagens HTML e PHP, amplamente usadas atualmente para o desenvolvimento de sites. Em algumas funções do Quiz que necessitavam de processamento instantâneo foram criadas funções em Javascript, apenas para contornar a necessidade do PHP de fazer o recarregamento da página para obter o processamento de informações.

Para a criação dos códigos de programação utilizou-se o pacote XAMPP[9] - pacote para desenvolvedores web que contém servidor Apache, Banco de dados MySQL, PHP, entre outras ferramentas e aplicações – que possui fácil instalação e dispensa configurações avançadas. Além disso, utilizou-se o editor de textos Notepad++[10] para a criação dos códigos PHP, HTML e Javascript.

Também era necessário encontrar uma API para a linguagem PHP que tornasse possível a interação com a ontologia e as pesquisas em SPARQL. Assim foi realizada uma pesquisa em busca de APIs que atendessem a esses requisitos e duas foram encontradas para a execução de testes: ARC2[11] e RAP[12].

4.1. A API ARC2:

A API ARC2 é bastante utilizada atualmente para o desenvolvimento de aplicações que utilizem RDF, além de ser frequentemente testada, atualizada e melhorada por seus desenvolvedores e comunidade. Possui interpretadores RDF com suporte a diversos formatos, como RDF/XML, Turtle, eRDF, etc., suporte a linguagem SPARQL, *plug-ins*, além de ser pequena, gratuita, código aberto e fácil de usar. É compatível com PHP4 e PHP5, dispensa o uso de variáveis globais e outras práticas que podem causar problemas de integração. Por fim, possui a detecção automática de caminho para a inclusão de componentes.

---

[9] http://www.apachefriends.org/en/xampp.html

[10] http://notepad-plus-plus.org/

[11] http://arc.semsol.org/

[12] http://www4.wiwiss.fu-berlin.de/bizer/rdfapi/index.html

Buscando-se testar a API ARC2 criou-se um código PHP cuja função era ler a ontologia do Quiz Ontomúsica, salva em um arquivo de formato OWL, e mostrar todo o seu conteúdo em formato de triplas, separadas por colunas denominadas “sujeito”, “predicado” e “objeto”. Este código pode ser visto na Figura 13.

```
10  <?php
11      include_once("C:/Xampp/xampp/php/ARC2/ARC2.php");
12      echo "<b> LISTA TODAS AS TRIPLAS: </b> ";
13      $parser = ARC2::getRDFParser();
14      $parser->parse('http://localhost/ontomusica.owl');
15      $triples = $parser->getTriples();
16      foreach($triples as $a => $b) {
17          print "<table border=#>";
18          print "<tr>";
19          print ("<td>Sujeito: ".$b['s']."</td>");
20          print ("<td>Predicado: ".$b['p']."</td>");
21          print ("<td>Objeto: ".$b['o']."</td>");
22          print "</tr>";
23          print "</table>"; }
24  ?>
```

Figura 13 - Código PHP que lista todas as triplas da ontologia.

Por ser um código PHP onde se utiliza a API ARC2 (ARC2, 2010), deve ser iniciado com a inclusão da classe estática da mesma, definida por meio da declaração “include_once()” – localizada na linha 11 do código - e que inclui o correto local para o arquivo arc2.php. Esta definição é o único requisito para o uso da API.

Na sequência do código é chamado o interpretador RDF da API, por meio do comando “$parser = ARC2::getRDFParser();” – na linha 13 do código. Neste caso é utilizado o interpretador genérico da API, mas ela também possui opções para os formatos RDF/XML, Turtle, entre outros. Este interpretador faz a leitura da ontologia do Ontomúsica, carregada a partir de arquivo .owl, através do comando “$parser->parse('http://localhost/ontomusica.owl');” – na linha 14 do código.

Por fim é feito o armazenamento das triplas analisadas pelo interpretador em um *array*, por meio da função “getTriples()”, como visto na linha de código “$triples = $parser->getTriples();” – linha 15 do código. Após a correta leitura e armazenamento das triplas, o comando “*Foreach*” – linha 16 do código - é utilizado para mover-se pelo *array* e imprimir na tela as triplas da ontologia, devidamente separadas por sujeito, predicado e objeto. O resultado deste código pode ser visto na Figura 14.

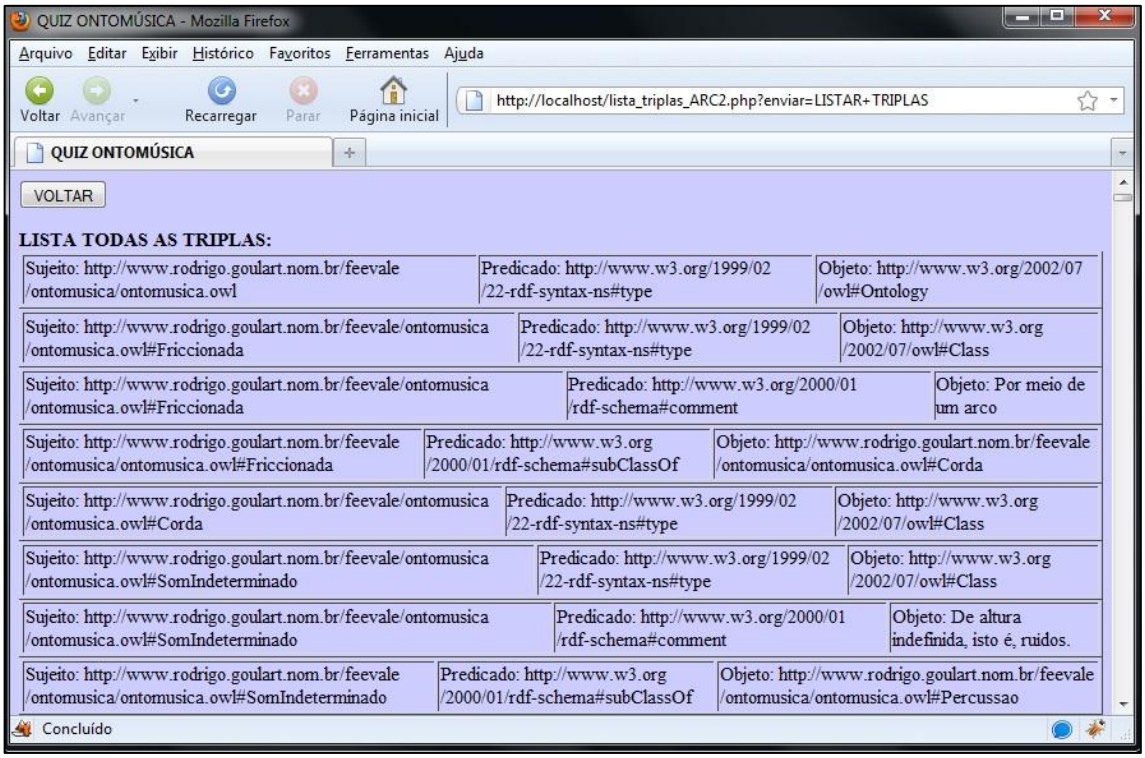


Figura 14 - Resultado do código que lista as triplas da ontologia.

Este código serve para ilustrar o funcionamento de algumas funções da API, além de ser bastante útil para o entender o funcionamento da ontologia. Ele inicialmente foi desenvolvido apenas para teste, mas decidiu-se por mantê-lo como uma função adicional do sistema por meio de um botão chamado “Listar triplas” na página inicial do Quiz Ontomúsica.

Após o teste partiu-se para a criação de um código que permitisse acessar a ontologia do Ontomúsica e fazer consultas de dados a partir de pesquisas SPARQL. O objetivo dessas pesquisas é tornar o sistema de perguntas mais dinâmico, fornecendo mais opções de questionamentos e aproveitando todo o conteúdo da ontologia para a elaboração dos mesmos.

Diferentemente da API ARC2, onde as pesquisas SPARQL podem ser realizadas diretamente “em memória”, bastando chamar uma função que cria um modelo baseado nos dados da ontologia, na API ARC2 é necessário utilizar a opção RDF Store. Basicamente esta função da API cria um banco de dados MySQL com toda informação encontrada na fonte de dados desejada – neste caso a ontologia do Ontomúsica.

É importante deixar claro que sem a utilização do RDF Store não é possível fazer as consultas em SPARQL utilizando a ARC2. A pesquisa “em memória” é um recurso que o desenvolvedor da API planeja implementar em um futuro próximo.

O código PHP para criação do RDF Store é iniciado com a inclusão da classe estática da ARC2, já descrita anteriormente. Após isso é criada a variável “config” - linha 9 do código - que recebe um *array* com a configuração do banco de dados MySQL - endereço, nome do banco, usuário, senha e prefixo utilizado para a criação das tabelas do banco - criado para a utilização da RDF Store. Essa configuração pode ser vista no código da Figura 15.

Após a configuração do banco, a linha “$store = ARC2::getStore($config);” - linha 16 do código - faz com que o RDF Store receba a configuração do banco para fazer a instanciação. Então é feita uma chamada para a função “*If*” que faz a criação das tabelas MySQL - linha 17 do código. Por fim, o RDF Store recebe todas as triplas da ontologia do Ontomúsica através do código “$store->query('LOAD <http://localhost/ontomusica.owl>');” - linha 20 do código. Esse processo é realizado pelo parâmetro “*load*” utilizado no método “*Query*”.

```
 8   include_once("C:/Xampp/xampp/php/ARC2/ARC2.php");
 9   $config = array(
10   'db_host' => 'localhost',
11   'db_name' => 'my_db',
12   'db_user' => 'ontomusica',
13   'db_pwd' => 'senha',
14   'store_name' => 'ontomusica',
15   );
16   $store = ARC2::getStore($config);
17   if (!$store->isSetUp()) {
18   $store->setUp();
19   }
20   $store->query('LOAD <http://localhost/ontomusica.owl>');
```

Figura 15 - Código PHP que cria o banco de dados MySQL para as pesquisas SPARQL.

Com as triplas da ontologia devidamente carregados no RDF Store, tornou-se possível realizar a pesquisa de informações utilizando SPARQL. Para testar a eficiência da pesquisa SPARQL no RDF Store, elaborou-se uma pesquisa simples, cujo objetivo é exibir todos os compositores e suas obras, ordenados por ordem alfabética. Essa pesquisa pode ser vista no código da Figura 16.

```
22    $q = '
23    PREFIX b:<http://www.rodrigo.goulart.nom.br/feevale/ontomusica/ontomusica.owl#>
24    SELECT ?nome_compositor ?nome_obra
25    WHERE {?X b:compos ?Y.
26        ?X b:nome ?nome_compositor.
27        ?Y b:nome ?nome_obra}
28    ORDER BY ?nome_compositor ?nome_obra
29    ';
30    $r=$store->query ($q, 'rows');
31    print_r ($r);
32    foreach ($r as $a) {
33    print ("<br />Compositor: ".$a['nome_compositor']." - Obra: ".$a['nome_obra']);
34    }
```

Figura 16 - Código PHP que faz a pesquisa SPARQL e a exibição dos resultados.

Assim, criou-se uma variável “q” - linha 22 do código - cuja função é armazenar a pesquisa a ser realizada na base de dados e um *array* “r” - linha 30 do código - que recebe o resultado da pesquisa armazenada em “q”- por meio do método “*query*”. Por fim o comando “*Foreach*” - linha 32 do código - é utilizado para deslocar-se pelo *array* e exibir os resultados na tela. O resultado deste código pode ser visto na Figura 17.

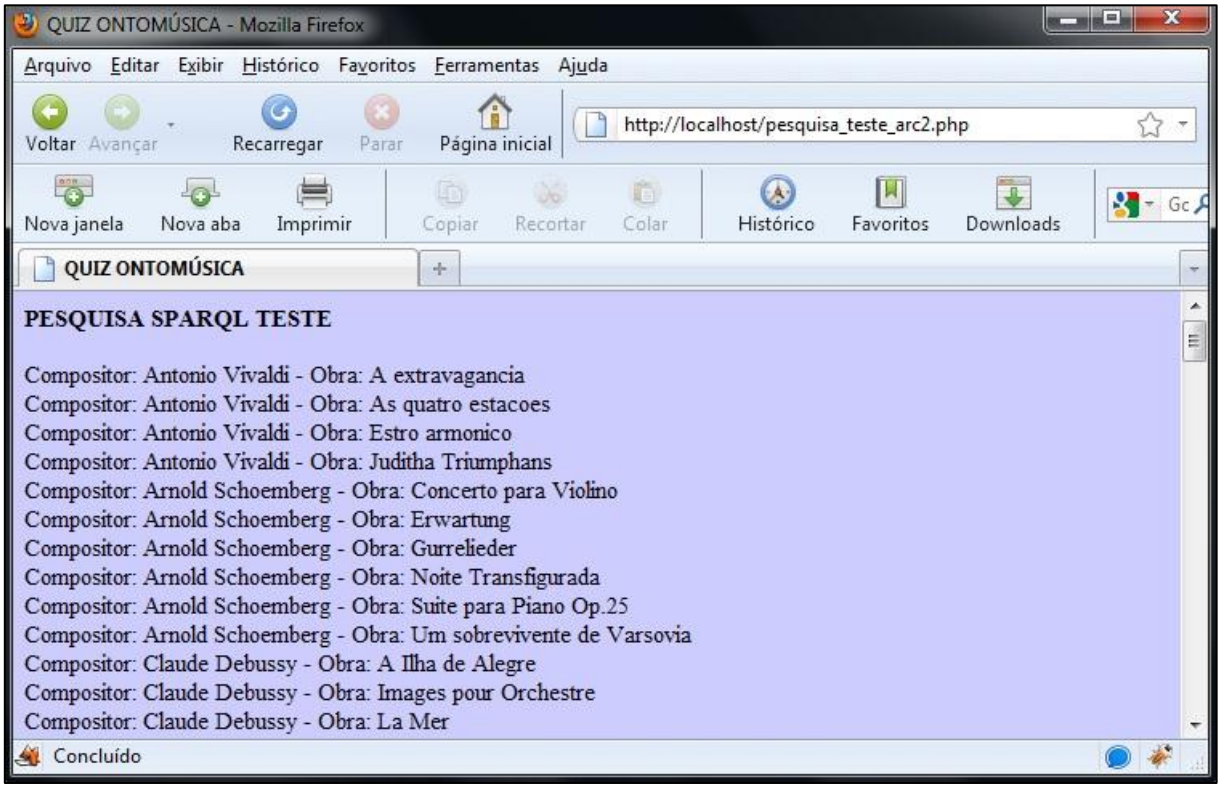


Figura 17 - Resultados da pesquisa SPARQL com uso da API ARC2.

4.2. A API RAP

A API ARC2 para PHP é útil para a manipulação e pesquisas em modelos RDF e possui interpretadores integrados para os formatos RDF/XML, N3, N-Triple, entre outros. Tem suporte a *plug-ins*, métodos de visualização de grafos e motor de pesquisas SPARQL e RDQL. Também é importante ressaltar que o website da API possui diversos tutoriais, exemplos de uso e documentação geral para dar suporte à mesma.

Uma de suas maiores vantagens é a sua capacidade de armazenar a fonte de dados para pesquisa em bancos de dados ou "em memória". Para isso, a API cria modelos baseados na origem dos dados que o usuário deseja utilizar, e carrega este modelo em memória ou em um banco de dados especificado pelo usuário.

A API teve sua primeira versão lançada em 2002, desde então passando por muitas revisões e atualizações. Inicialmente com poucos recursos, passou a oferecer suporte à linguagem RDQL em meados de 2003 e suporte à linguagem SPARQL no começo de 2006. Desde a implementação deste recurso, a API teve sua última versão lançada no começo de 2008, sem atualizações desde então. É relevante considerar que neste período a Linguagem SPARQL já passou por algumas mudanças que incluem a implementação de novos paramentos, certamente não encontrados na API ARC2.

Para testar a API ARC2, foi reproduzido o mesmo teste da API ARC2: Criar um código de programação que faça a listagem de todas as triplas da ontologia e exiba-as na tela, além de uma pesquisa simples em SPARQL. Na Figura 18 pode-se ver o código PHP que faz a listagem das triplas.

Inicialmente, para a utilização da API, existem dois requisitos: A definição do diretório onde foram extraídos os arquivos da API, depois do seu download – conforme visto na linha 2 do código, e a inclusão do arquivo "rdfapi.php" – localizada na linha 3 do código, para a inclusão das classes.

```
1  <?php
2     define("RDFAPI_INCLUDE_DIR", "C:/Xampp/xampp/php/rdfapi-php/api/");
3     include(RDFAPI_INCLUDE_DIR . "RDFAPI.php");
4     require_once 'C:/Xampp/xampp/php/rdfapi-php/test/config.php';
5     $ontomusica = ModelFactory::getDefaultModel();
6     $ontomusica->load(RDFAPI_INCLUDE_DIR . 'ontomusica.owl');
7     $ontomusica->writeAsHtmlTable();
8  ?>
```

Figura 18 - Código PHP que faz a listagem das triplas.

Para que seja possível a pesquisa por dados utilizando a API RAP (RAP, 2010), primeiramente deve-se criar um modelo de dados. A API cria modelos de dados em banco de dados ou em memória. Como a intenção deste teste é verificar se as triplas são corretamente importadas para então visualizá-las, o modelo de dados em memória foi selecionado pela sua facilidade de criação e praticidade.

Algumas operações requerem a definição do diretório onde se localiza o arquivo de configuração "config.php" – localizado na linha 4 do código – e isto inclui a criação de modelos de dados. Este passo pode ser visto em alguns exemplos de uso na documentação da API, e alguns desenvolvedores recomendam também para que outras operações funcionem corretamente sem retornar erros, já que este arquivo contém diversas configurações da API.

Assim é feita a criação do modelo de dados em memória com o nome "ontomusica" – localizado na linha 5 do código - para que então este modelo receba as triplas extraídas do arquivo .owl da ontologia do Ontomúsica – código pode ser visto na linha 6 do código – através do comando "*load*". Por fim é feita a visualização do conteúdo do modelo de dados através do comando "writeAsHtmlTable". Este comando da API já cria automaticamente uma tabela em HTML com toda a informação em formato de triplas, informando inclusive o número de triplas do modelo de dados. Parte desta tabela pode ser vista na Figura 19.

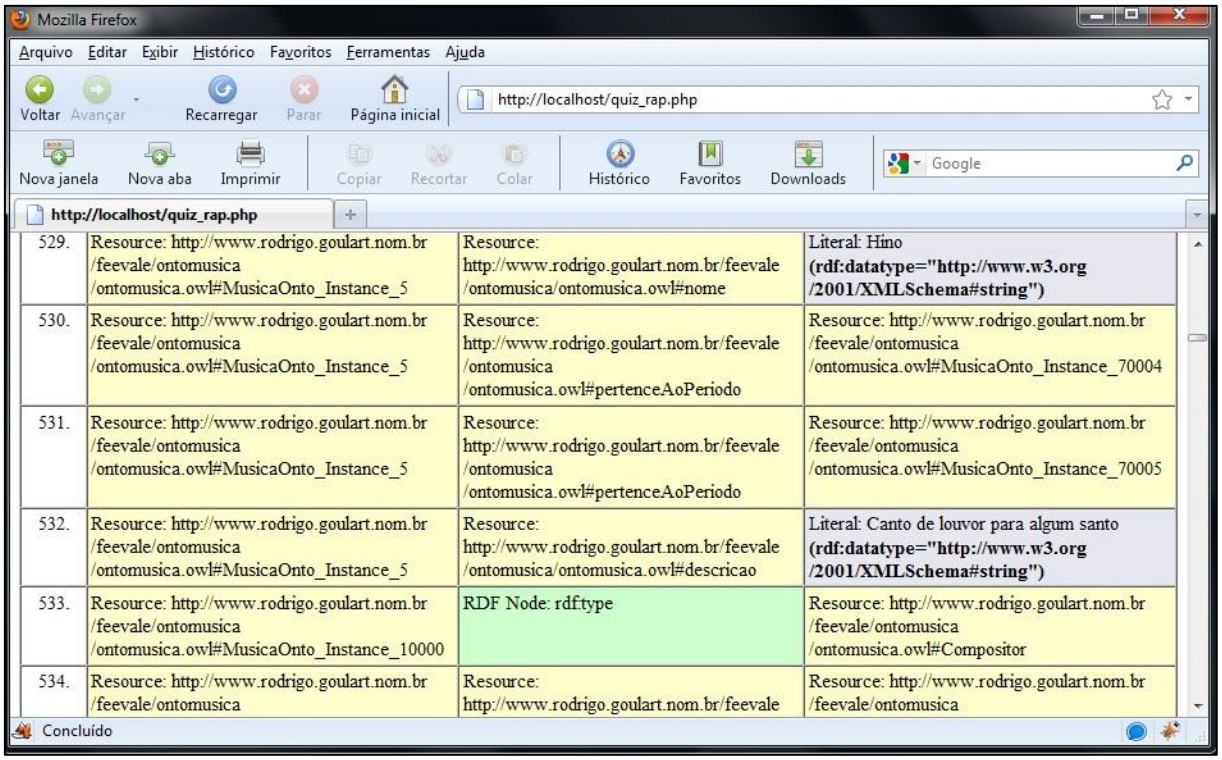


Figura 19 - Visualização das triplas com o uso da API RAP.

Após o teste de visualização das triplas ser concluído com sucesso, partiu-se para a elaboração de um código que faça a pesquisa por triplas no modelo de dados da API utilizando a linguagem SPARQL. Este código pode ser visto na Figura 20. Ele é iniciado da mesma maneira que o código utilizado no teste anterior, com as linhas necessárias para a inclusão das classes da API e também com os códigos que fazem a criação do modelo de dados.

```
2   define("RDFAPI_INCLUDE_DIR", "C:/Xampp/xampp/php/rdfapi-php/api/");
3   include(RDFAPI_INCLUDE_DIR . "RDFAPI.php");
4   require_once 'C:/Xampp/xampp/php/rdfapi-php/test/config.php';
5   $ontomusica = ModelFactory::getDefaultModel();
6   $ontomusica->load(RDFAPI_INCLUDE_DIR . 'ontomusica.owl');
7   $querystring = '
8   SELECT ?nome_compositor ?nome_obra
9   WHERE
10  {?s <http://www.rodrigo.goulart.nom.br/feevale/ontomusica/ontomusica.owl#compos>
11  ?o.
12  ?s <http://www.rodrigo.goulart.nom.br/feevale/ontomusica/ontomusica.owl#nome>
13  ?nome_compositor.
14  ?o <http://www.rodrigo.goulart.nom.br/feevale/ontomusica/ontomusica.owl#nome>
15  ?nome_obra.}';
16  echo $ontomusica->sparqlQuery($querystring, 'HTML');
```

Figura 20 - Código PHP criado para pesquisa SPARQL com a API RAP.

Na sequência é criada a variável “$querystring” – localizada na linha 7 do código – que recebe a pesquisa SPARQL criada para pesquisar no modelo de dados. A pesquisa vista no código é criada para buscar os nomes de compositores e suas respectivas obras. Em sequência o comando echo faz a impressão do modelo de dados “$ontomusica”, que recebe a pesquisa SPARQL definida anteriormente no código, seguida do parâmetro “HTML”. Este é o responsável por definir o modo de exibição dos resultados, neste caso uma tabela formato HTML. Este resultado pode ser visto na Figura 21.

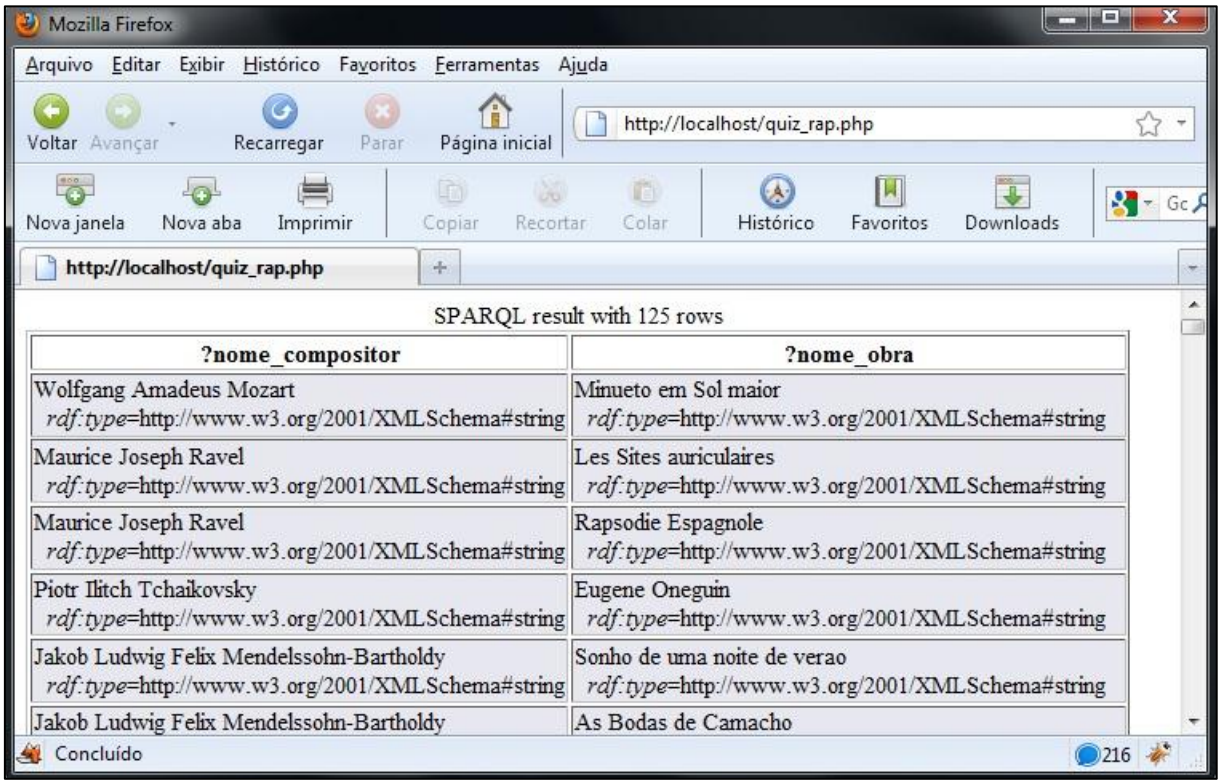


Figura 21 - Resultado da pesquisa SPARQL com uso da API RAP

Durante a criação dos testes com a API ARC2, algumas situações ocorridas fazem com que alguns fatores mereçam menção: Primeiramente, devido ao tempo de desenvolvimento desta ferramenta, já que sua primeira versão foi lançada em 2002, seu nível de maturidade é bastante alto e possui diversos recursos criados que facilitam a vida do desenvolvedor - como é o caso das funções para exibição de resultados em tabelas HTML.

Por outro lado, existem fatores que são destacados pelo seu lado negativo. Como pode ser visto no código da Figura 20, a cláusula PREFIX não foi utilizada, pois nos testes ela não surtia efeito ao ser incluída a não ser que recebesse o prefixo “vcard”. E mesmo quando utilizada desta maneira as pesquisas retornavam resultados vazios. Assim, este problema é contornado alterando esta constante nos arquivos de configuração da API para que funcione

da maneira como que o desenvolvedor deseja, embora seja algo que acabe tornando o desenvolvimento mais lento em diversos momentos.

4.3. Conclusão dos testes

Assim, conclui-se que as pesquisas SPARQL ocorrem sem maiores problemas com o uso da API ARC2, de acordo com os passos encontrados na documentação do desenvolvedor e aplicados corretamente conforme os testes acima. Já nos testes com a API ARC2 alguns problemas – que não existem na API ARC2 - tornaram seu uso menos recomendável, como a necessidade de declarar e utilizar variáveis globais e constantes, além de sua recente falta de atualizações, que torna o uso da linguagem SPARQL um pouco mais limitada.

A principal vantagem do uso da API ARC2 neste caso seria o fato de ela ter a capacidade de fazer as pesquisas em memória, criando um modelo no momento da pesquisa, sem a necessidade de utilização de um banco de dados. Mas considerando-se que o recurso RDF Store da API ARC2 é fácil de ser utilizado e totalmente funcional, este aspecto não foi tão relevante nesta avaliação.

Diante dos resultados destes testes, a API ARC2 foi escolhida para o desenvolvimento do novo Quiz Ontomúsica por causa de sua fácil implementação e utilização.

5 O NOVO QUIZ ONTOMÚSICA

Na análise feita anteriormente no trabalho sobre o Quiz Ontomúsica de Boff, foram citadas algumas limitações do sistema como ideias a serem exploradas em busca de soluções. A falta de utilização da totalidade de informações da ontologia talvez seja a maior destas limitações, já que o uso de todo este conteúdo certamente tornaria o Quiz Ontomúsica muito mais interessante.

Buscando-se atingir este objetivo, a linguagem SPARQL foi selecionada para ser utilizada como forma de criar pesquisas mais dinâmicas na ontologia do Quiz Ontomúsica. Como se sabe, as pesquisas em SPARQL são feitas utilizando o formato de triplas, o que torna possível criar pesquisas que busquem não apenas instâncias, mas que retornem as relações entre informações.

A possibilidade de criar pesquisas que busquem informações baseadas nas relações entre dados oferece muitas alternativas a um desenvolvedor, já que os resultados destas pesquisas são muito mais amplos que pesquisas que busquem apenas informações específicas. Pode-se assim trabalhar com dados que inicialmente nem eram considerados pelo desenvolvedor, mas que vendo suas relações com outras informações passa a considerá-los relevantes.

Para exemplificar este caso, são selecionadas algumas triplas da ontologia do Quiz Ontomúsica, que podem ser vistas na Figura 22. Neste exemplo é possível observar a relação de instâncias com suas classes – Instância#01 (sujeito) de tipo (predicado) compositor (objeto), propriedades – Instância#02 (sujeito) de propriedade “nome” (predicado) A Extravagância (predicado), ou mesmo relação entre instâncias – Instância#01 (sujeito) compôs (predicado) Instância#02 (objeto).

SUJEITO	PREDICADO	OBJETO
INSTÂNCIA#01	TIPO	COMPOSITOR
INSTÂNCIA#01	NOME	ANTÔNIO VIVALDI
INSTÂNCIA#01	COMPÔS	INSTÂNCIA#02
INSTÂNCIA#02	TIPO	OBRA
INSTÂNCIA#02	NOME	A EXTRAVAGÂNCIA
INSTÂNCIA#02	FOICOMPOSTAPOR	INSTÂNCIA#01

Figura 22 - Exemplo de triplas da ontologia do Quiz Ontomúsica

Mas as triplas não representam apenas estas relações entre instâncias e suas propriedades, pois as verificando se podem encontrar relações de outros tipos, que determinam os tipos das informações encontradas na ontologia, definições de classes e subclasses, etc. Na Figura 23 encontram-se alguns exemplos destas relações. Neste mesmo exemplo pode ser visto também uma tripla que especifica uma relação de inversão: “Compôs” – compositor compôs obra - e “Foicompostapor”- obra foi composta por compositor - que são relações inversas.

SUJEITO	PREDICADO	OBJETO
GENEROMUSICAL	TIPO	CLASSE
VOCAL	TIPO	CLASSE
VOCAL	SUBCLASSEDE	GENEROMUSICAL
FOICOMPOSTAPOR	INVERSODE	COMPÔS

Figura 23 - Outro exemplo de triplas da Ontologia do Quiz Ontomúsica

Tendo conhecimento destas relações novas opções surgem para a elaboração de pesquisas com o uso do SPARQL. Pode-se procurar na ontologia do Ontomúsica não apenas

por inferências específicas, como em pesquisas por nomes de compositores e obras, mas também pelos relacionamentos destas classes e inferências.

Suponha-se um cenário onde o usuário informe ao Quiz Ontomúsica que gostaria de responder perguntas sobre um assunto específico, como por exemplo, gêneros musicais. Assim, o sistema faria pesquisas que buscariam as relações da classe gênero musical e formularia questionamentos sobre as informações que encontraria. Em um exemplo disso, um questionamento poderia ser feito sobre os gêneros das obras de determinado compositor. Isso seria possível pelo fato da classe “compositor” ser relacionada com a classe “obra”, que por vez relaciona-se com a classe “generomusical”.

Na Figura 24 pode-se ver um diagrama das classes e relações da ontologia do Quiz Ontomúsica, para melhor ilustração do exemplo. A imagem foi criada com o uso do Protégé, utilizando o *plug-in Jambalaya*, que serve para a visualização de ontologias. No exemplo, pode-se observar que entre as classes existem setas que representam suas relações, como no caso das classes “compositor” e “período”, onde as setas representam as relações “periodocontem” – Período contém compositor – e “pertenceaoperiodo” – Compositor pertence ao período.

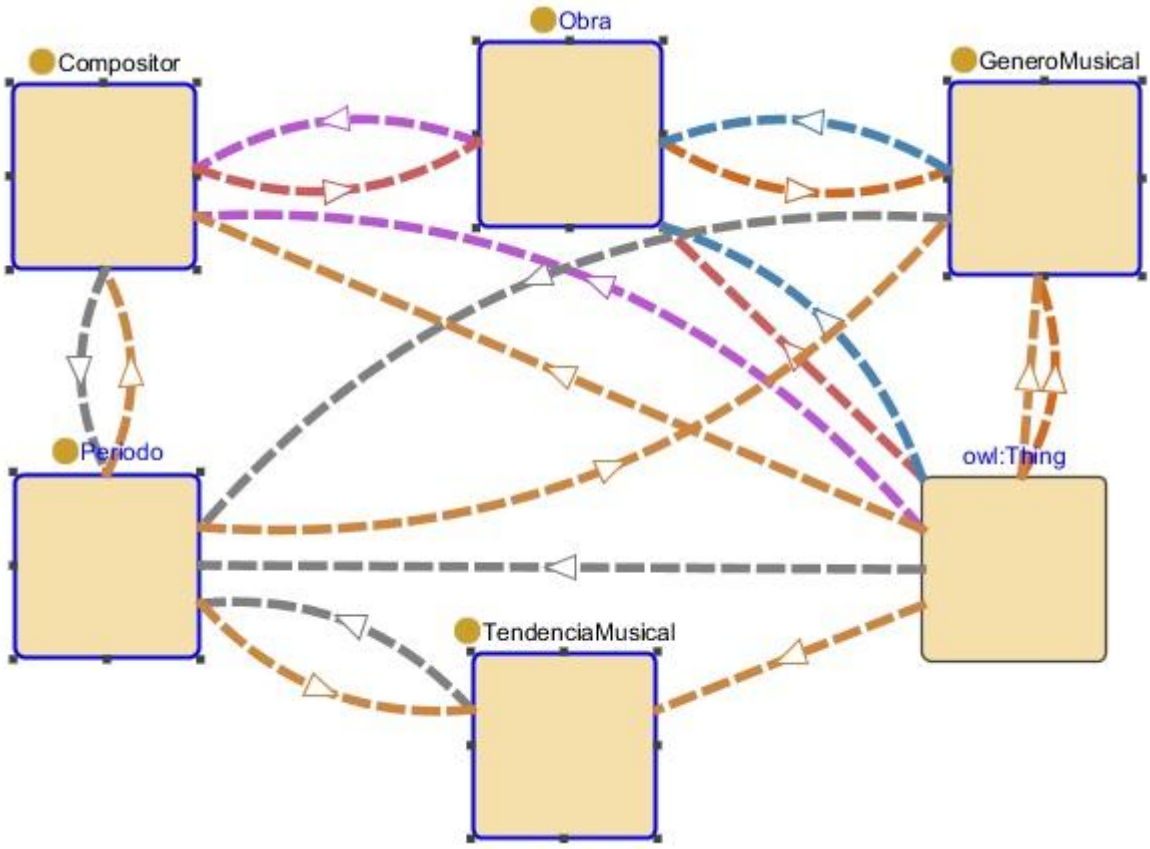


Figura 24 - Relações entre as classes da ontologia do Quiz Ontomúsica.

Em outro cenário, suponha-se que o sistema pergunte ao usuário quais os assuntos que ele gostaria de relacionar para elaborar seus questionamentos. Assim, o sistema buscaria as relações entre essas classes e, caso não encontrasse uma relação direta, buscaria relações dos assuntos selecionados com outras classes indiretas para chegar a um ponto onde seja possível elaborar um questionamento.

5.1. Modificações na ontologia

É importante registrar que a ontologia original do Quiz Ontomúsica possuía alguns pequenos problemas que tiveram que ser resolvidos, ou os resultados das pesquisas SPARQL feitas poderiam ser imprecisos. Estes problemas envolviam a criação de quase todas as propriedades da ontologia, como “compos” e “foicompostapor” que estavam criadas, mas não possuíam definição de sujeito e objeto, o que faz com que classes como compositor, obra e gênero musical não possuam relações diretas entre si.

Na Figura 25 pode-se ver o diagrama de classes da Ontologia com problemas. Neste exemplo, as classes não possuem relações entre si, mas estas relações estão todas em forma de círculo, perdidas ao lado da classe “owl:thing” do protege, pois não estão ligadas a nada.

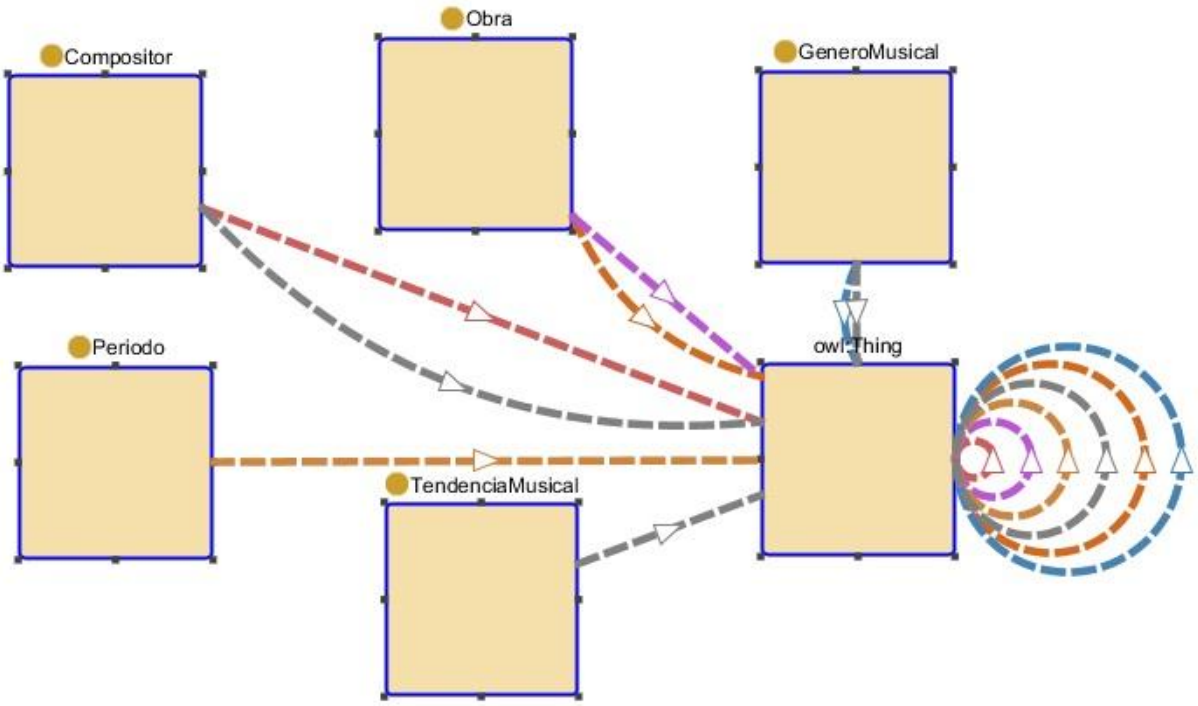


Figura 25 - Classes da Ontologia com propriedades perdidas

5.2. A criação do novo Quiz Ontomúsica

O novo Quiz Ontomúsica foi criado com duas projeções a serem atingidas em etapas: A primeira era conseguir reproduzir o funcionamento do antigo Ontomúsica, criando questionamentos sobre a obra correta de um determinado compositor, exibindo o gênero da obra correta como dica em caso o usuário escolha a alternativa errada. Já a segunda projeção envolvia desenvolver o novo Ontomúsica com a capacidade de elaborar pesquisas genéricas e mais dinâmicas, buscando as relações entre os dados da ontologia e formulando questionamentos mais diversificados.

O primeiro objetivo foi alcançado e será descrito na sequência deste trabalho em detalhes, para que haja fácil entendimento de todo o seu processo de criação. Já a segunda

projeção foi mantida como possibilidade de melhoria deixada para trabalhos futuros. Os motivos são descritos nas considerações finais deste trabalho.

Para a criação do Quiz Ontomúsica foram utilizadas as mesmas ferramentas utilizadas para os testes com as APIs para PHP testadas anteriormente: O pacote XAMPP e o editor Notepad++, além da API ARC2, escolhida após os testes realizados anteriormente neste trabalho.

Até agora foram descritos os procedimentos e regras gerais relacionados ao uso das ferramentas e linguagens selecionadas para o desenvolvimento do Quiz Ontomúsica. Então no seguimento deste serão mostradas telas do sistema e partes dos códigos do trabalho com suas devidas explicações. O código completo está disponível nos anexos deste trabalho.

Na página inicial do sistema buscou-se criar uma interface simples, que mostrasse ao usuário um menu com as opções de questionamentos e um botão que fizesse a listagem das triplas da ontologia. Esta opção foi mantida para consultas e esclarecimentos a respeito do funcionamento do Ontomúsica. A Figura 26 mostra a interface do Quiz Ontomúsica.

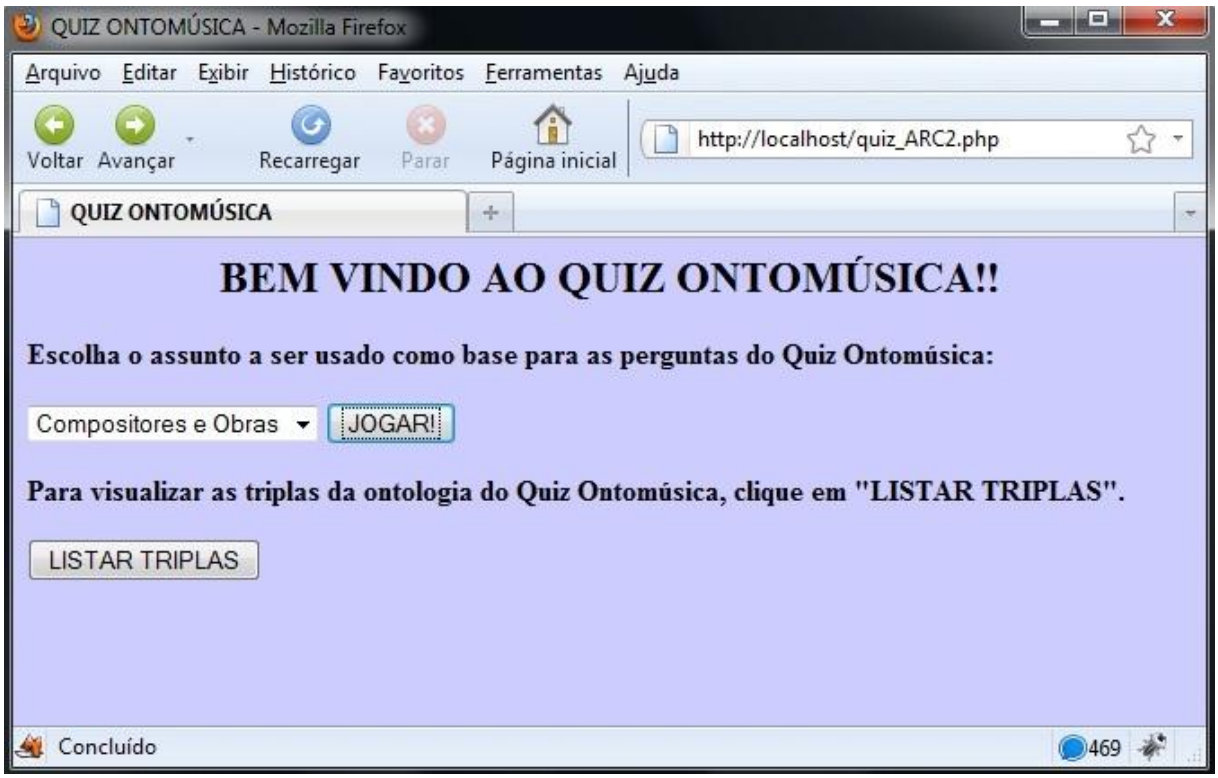


Figura 26 - Página inicial do Quiz Ontomúsica.

A Figura 27 mostra o código que faz as pesquisas SPARQL na ontologia do Ontomúsica. Pode-se ver na linha 22 do código o comando “IF” que seleciona o assunto das

perguntas do sistema, selecionados pelo usuário na tela principal, e então executa as pesquisas na ontologia. Como nesta versão do Ontomúsica apenas questionamentos relacionados a compositores e obras estão sendo feitos, esta é a única opção encontrada no código. A estrutura foi mantida, pois caso novas opções fossem introduzidas no futuro, seria apenas necessário introduzir novas funções "IF".

Na linha 27 do código é criada a variável "q1", que recebe a primeira pesquisa SPARQL, que busca na ontologia nomes de compositores e obras, baseado na relação "Compositor(sujeito) compos (predicado) Obra (objeto)". Apesar de a pesquisa procurar por estes dois valores, apenas os nomes de compositores são retornados como resultado da pesquisa, com o uso da cláusula *Select*, pois é a única informação desejada. Esta pesquisa é feita utilizando-se também a cláusula *Distinct*, que evita repetições, não existindo assim dois nomes iguais de compositores no resultado final da pesquisa.

Assim, é criada a variável "a" – localizada na linha 35 do código, que recebe os resultados da pesquisa SPARQL "q1", através da função "*Query*", executada no RDF Store da API ARC2 – já com os dados da ontologia do Ontomúsica devidamente importados anteriormente. Em seguida é utilizada a função "Shuffle" no *array* "a" – na linha 36 do código - que faz com que o *array* tenha toda a sua ordem sorteada de maneira aleatória. Essa função é bastante importante, pois ela vai fazer com que os compositores selecionados para construir a pergunta do Quiz e para buscar as respostas erradas não sejam sempre os mesmos.

```
22    if ($_POST['assunto'] == "compositores e obras") {
23    echo "<h2 align='center'>";
24    echo "<b> QUIZ SOBRE COMPOSITORES E OBRAS </b>";
25    echo "</h2>";
26
27    $q1 = '
28    PREFIX b:<http://www.rodrigo.goulart.nom.br/feevale/ontomusica/ontomusica.owl#>
29    SELECT DISTINCT ?nome_compositor
30    WHERE {?X b:compos ?Y.
31          ?X b:nome ?nome_compositor.
32          ?Y b:nome ?nome_obra}
33    ';
34
35    $a = $store->query($q1, 'rows');
36    Shuffle ($a);
37
38    for($i=0; $i<3; $i++) {
39    $b = $a[$i]['nome_compositor'];
40    $q2 = '
41    PREFIX b:<http://www.rodrigo.goulart.nom.br/feevale/ontomusica/ontomusica.owl#>
42    SELECT ?nome_compositor ?nome_obra ?nome_genero
43    WHERE {?X b:compos ?Y.
44          ?Y b:generoDaObra ?W.
45          ?X b:nome ?nome_compositor.
46          ?Y b:nome ?nome_obra.
47          ?W b:nome ?nome_genero.
48    FILTER regex(?nome_compositor, "^'.$b.'").}
49    LIMIT 1
50    ';
51    $c[$i] = $store->query($q2, 'rows');
52    }
```

Figura 27 - Código PHP das pesquisas SPARQL feitas na ontologia

Na linha 38 do código é executada uma função "*For*" que percorre as três primeiras opções do *array* "a", armazena seu valor (o nome do compositor) na mesma posição do *array* "b" e então roda a pesquisa SPARQL "q2". Esta pesquisa busca no RDF Store o nome de um compositor, obra e o gênero da mesma. O nome do compositor sempre é especificado na cláusula "FILTER" – linha 48 do código, que nesta pesquisa está programada para receber os nomes armazenados previamente no *array* "b". Também é utilizada a cláusula "LIMIT" – linha 49 do código, que limita a pesquisa à apenas um resultado por pesquisa. Por fim, o resultado desta pesquisa SPARQL é armazenada em uma posição do *array* "c", cada vez que o comando FOR muda de posição.

Depois que as duas pesquisas SPARQL são feitas, temos como resultado o *array* "c", que possui 3 posições e em cada uma delas possui 3 dados: Um nome de compositor, obra e gênero. Entre estas três posições, uma delas é selecionada aleatoriamente para ser a resposta

certa. Então seus valores são armazenados em 3 variáveis diferentes: “resposta_certa”, “compositor” e “dica” – visto entre as linhas 78 e 80 do código da Figura 28.

```
77    $temp = rand(0,2);
78    $resposta_certa = $c[$temp][0]['nome_obra'];
79    $compositor = $c[$temp][0]['nome_compositor'];
80    $dica = $c[$temp][0]['nome_genero'];
81    $opcao0 = $c[0][0]['nome_obra'];
82    $opcao1 = $c[1][0]['nome_obra'];
83    $opcao2 = $c[2][0]['nome_obra'];
84
85    $dropDownMusica = "<select name='musica' id='musica' size='1' onchange='resposta()'>";
86    $dropDownMusica .= "<option value=''>Selecione</option>";
87    $dropDownMusica .= "<option value='" . $opcao0 . "'>" . $c[0][0]['nome_obra'] . "</option>";
88    $dropDownMusica .= "<option value='" . $opcao1 . "'>" . $c[1][0]['nome_obra'] . "</option>";
89    $dropDownMusica .= "<option value='" . $opcao2 . "'>" . $c[2][0]['nome_obra'] . "</option>";
90    $dropDownMusica .= "</select>";
```

Figura 28 - Código PHP que define as respostas e cria o menu de respostas

Entre as linhas 85 e 90 do código encontra-se a parte que faz a criação do menu de respostas da pergunta elaborada pelo sistema. Nota-se que apesar dos valores que são exibidos pelo menu serem posições estáticas do *array* “c”, não existe o problema de a resposta certa aparecer sempre na mesma posição do menu, pois conforme visto anteriormente, a opção correta sempre é selecionada aleatoriamente entre as 3 posições do *array*.

Na Figura 29 encontra-se o código Javascript que faz a verificação da opção selecionada pelo usuário no menu de respostas. Ele compara o valor da opção selecionada com as variáveis criadas que armazenam a resposta correta e a dica. Caso o usuário acerte, o código mostra uma mensagem de acerto. Caso selecione a opção errada, o código mostra uma mensagem de erro e também uma dica, composta pelo gênero da obra. Por fim, após a seleção da opção certa, ele recarrega a página, gerando uma nova pergunta.

```
54    $funcaoJavaScript = " <script>
55        function resposta(){
56        var respostaCerta = document.getElementById('respostaCerta').value;
57        var valorSelecionado = document.getElementById('musica').value;
58
59        if(valorSelecionado == respostaCerta)
60        {
61            alert('Voce acertou!! Parabéns!!');
62            document.location.reload();
63        }
64        else
65        {
66            var gDaObra = document.getElementById('generoDaObra').value;
67            alert('Resposta errada! DICA: O gênero da obra correta é ' + gDaObra);
68        }
69
70        }
71        </script>";
```

Figura 29 - Código Javascript que verifica a resposta selecionada

A Figura 30 mostra a tela onde se pode ver a questão a ser respondida e suas opções de respostas.

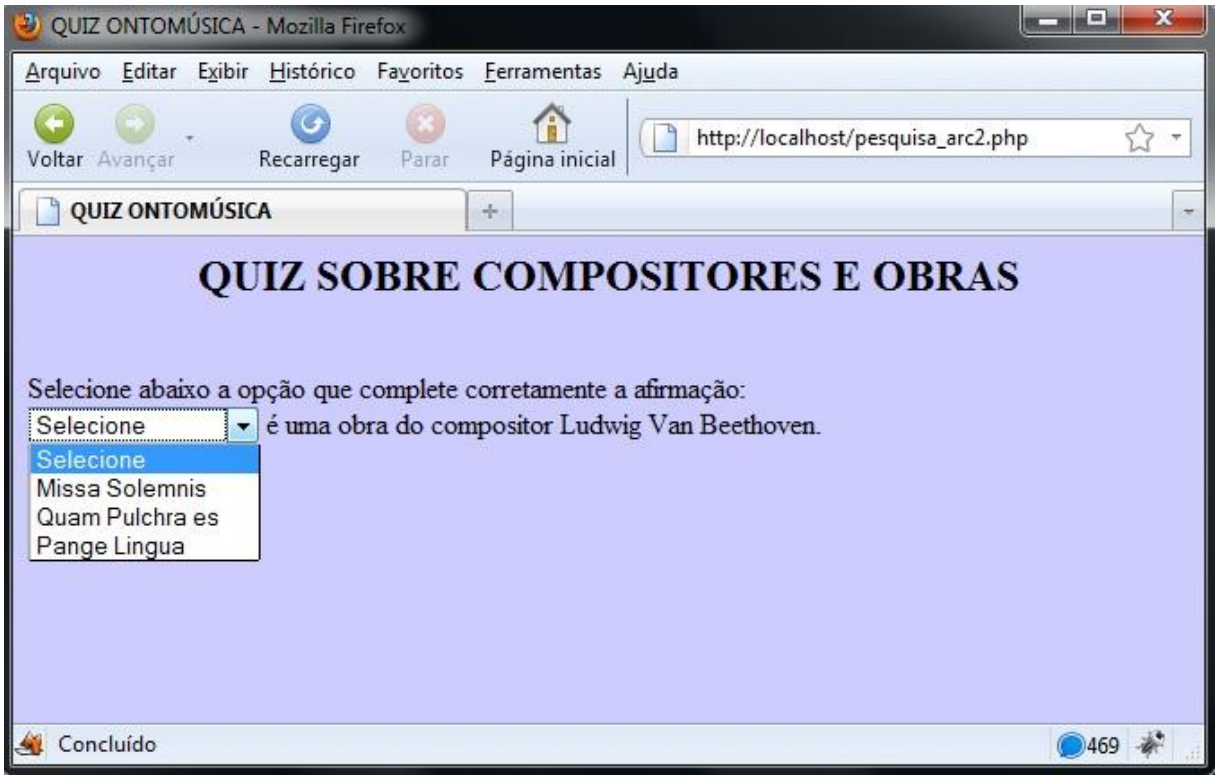


Figura 30 – Tela do Quiz Ontomúsica com pergunta e menu de respostas.

A Figura 31 mostra uma tela com a mensagem de acerto para o usuário.

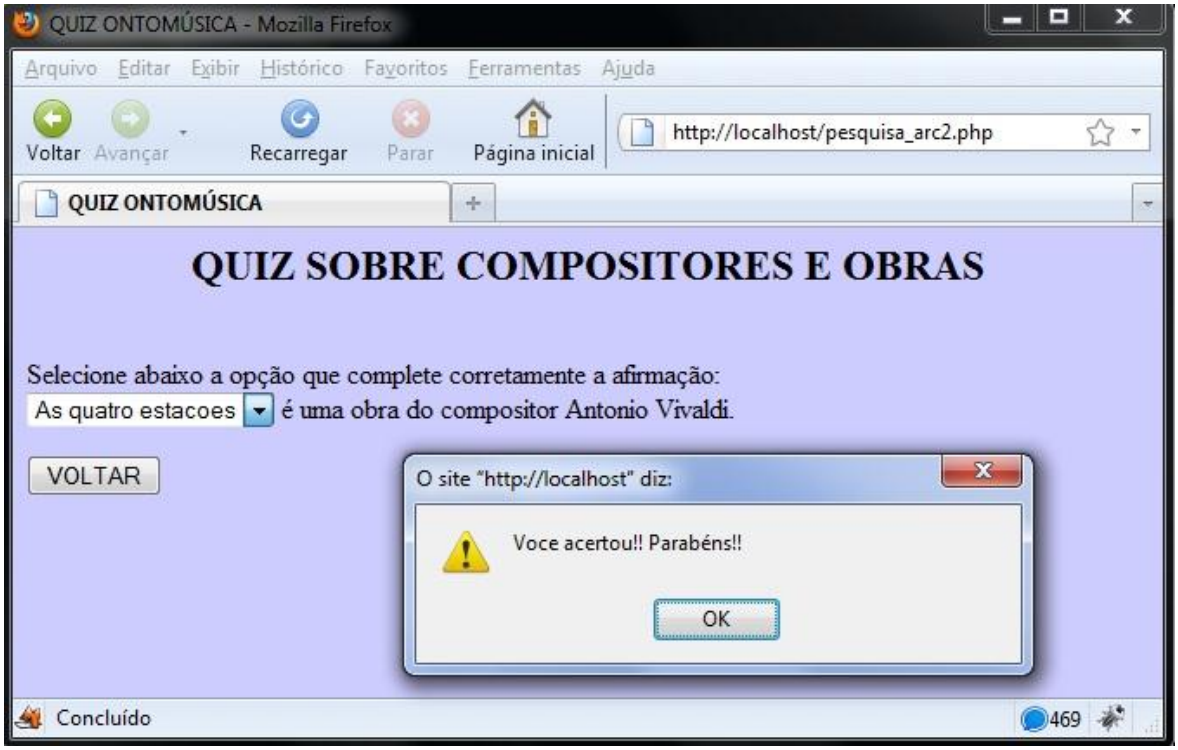


Figura 31 - Tela do Quiz Ontomúsica com mensagem de resposta certa.

Na Figura 32 pode-se ver uma tela do Quiz Ontomúsica com a mensagem de erro que aparece quando o usuário escolhe a opção errada.

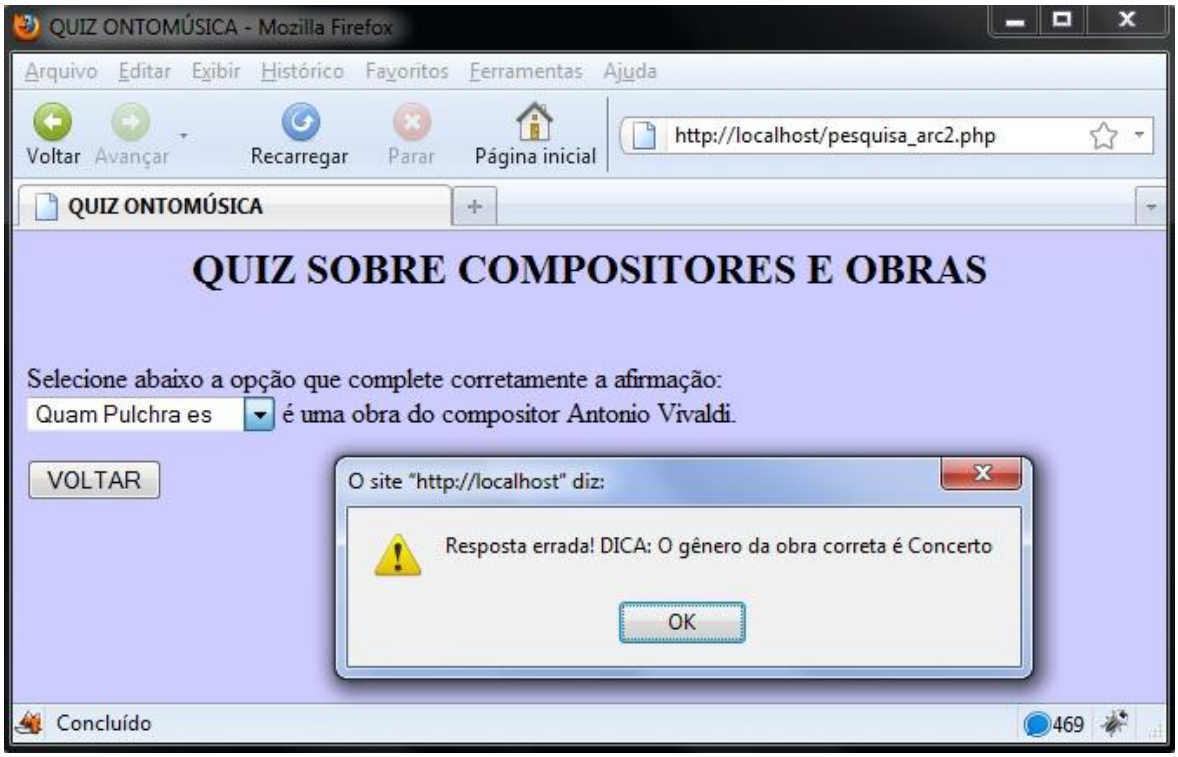


Figura 32 - Tela do Quiz Ontomúsica com mensagem de resposta errada e dica.

Como pode se ver, a nova versão do Quiz Ontomúsica é bastante simples e funcional e o uso do SPARQL abre muitas possibilidades a serem exploradas para torná-lo ainda mais interessante e intuitivo. Fica evidente que ainda há muito mais a ser feito neste sentido, conforme será relatado nas dificuldades encontradas e propostas para o futuro.

5.3. Dificuldades encontradas

Muitas dificuldades foram encontradas durante este trabalho relacionadas a testes com as APIs para a linguagem PHP. Por serem ferramentas elaboradas e mantidas por programadores interessados no uso da tecnologia e pela comunidade relacionada, muitas vezes seu desenvolvimento é mais demorado do que o encontrado em ferramentas elaboradas por empresas, que mantém suporte específico para produtos comerciais.

Assim, muitos erros e problemas encontrados aqui eram relacionados a erros de programação do sistema, bugs das APIs ou mesmo das linguagens usadas para o desenvolvimento do Ontomúsica. Certamente muito tempo foi utilizado no desenvolvimento até conseguir uma integração estável entre PHP, SPARQL e a API ARC2.

Alguns problemas relacionados ao uso da ontologia criada por Boff e utilizada no Quiz Ontomúsica original também poderiam ter ocorrido neste trabalho. A falta de algumas definições que foram resolvidas e relatadas no capítulo 5.2 teriam causado problemas caso pesquisas mais complexas com o SPARQL tivessem sido feitas no trabalho.

5.4. Propostas para o futuro

Boff apontou em seu trabalho a falta de acentuação e caracteres especiais nas instâncias da ontologia como uma necessidade para contornar problemas encontrados durante o desenvolvimento de seu trabalho. Este problema não chegou a ser abordado aqui por que a proposta de fazer o Quiz Ontomúsica funcionar corretamente com o uso da linguagem SPARQL acabou sendo tratada como prioridade. Então a solução deste problema mantém-se como uma dica de solução para o futuro.

Outra proposta interessante para ser implementada no sistema é a criação de perfis individuais para os usuários do Quiz Ontomúsica. Estes perfis seriam utilizados para armazenar e posteriormente consultar informações relacionadas aos tipos de questionamentos que são preferência dos usuários, assim como estatísticas relacionadas ao desempenho dos mesmos durante o uso do Quiz. Estas informações poderiam ser armazenadas em uma ontologia, que também poderia ser acessada com pesquisas em SPARQL.

Mas a maior proposta deixada aqui como desafio é o de tornar o Quiz Ontomúsica um sistema de educação realmente funcional utilizando pesquisas mais complexas com o SPARQL. Utilizar mecanismos criados sobre pesquisas que buscassem relações entre todo tipo de informação da ontologia podem ser a solução para tornar o sistema totalmente dinâmico, tornando-o muito mais interessante, divertido e desafiador.

CONCLUSÃO

O maior objetivo e desafio deste trabalho foi o de selecionar uma linguagem de pesquisa para ontologias como o SPARQL e aplicar seu uso diretamente na ontologia com eficiência, para comprovar que a mesma é uma alternativa viável e perfeitamente funcional na hora de criar sistemas baseados em ontologias.

O processo de elaboração deste trabalho foi bastante difícil em sua totalidade. Desde a pesquisa sobre sistemas inteligentes e ontologias; a busca por linguagens de pesquisa em ontologias e principalmente o desenvolvimento do sistema com a utilização de linguagens de programação Web foram todas muito desafiadoras. Mas também foram muito recompensantes do ponto de vista pessoal, visto que nenhuma destas áreas eram áreas de domínio do desenvolvedor, mas que desde o início teve o interesse despertado pelos assuntos.

Ao fim deste trabalho fica a conclusão de que a proposta inicial foi atingida com sucesso, mas que muito ainda pode ser feito. Fica a satisfação pelo esforço realizado e os resultados obtidos, mas também o desejo de que no futuro estes resultados possam ser retomados e novos avanços sejam feitos, para que o Quiz Ontomúsica possa evoluir e talvez até um dia seja disponibilizado na internet para todos.

REFERÊNCIAS BIBLIOGRÁFICAS

ABED - **ASSOCIAÇÃO BRASILEIRA DE EDUCAÇÃO A DISTÂNCIA**. Disponível em <http://www2.abed.org.br/>. Acessado em 03 de Maio de 2010.

Apache friends – Xampp. Disponível em <http://www.apachefriends.org/en/xampp.html> Acessado em 23 de Agosto de 2010.

ARC2 - Easy RDF and SPARQL for LAMP systems – ARC RDF Classes for PHP: Disponível em: <http://arc.semsol.org/home> Acessado em 10 de Agosto de 2010.

BECKER, Júnior. **Ontologia Terminológica para Apoio a Ferramentas de Recuperação de Informações e de Text Mining.** Monografia (Bacharelado em Ciência da Computação) – Instituto de Ciências exatas e Tecnológicas, Feevale. Novo Hamburgo, 2006.

BOFF, Rogério Eduardo. **Educação musical à distância utilizando ontologias.** Monografia (Bacharelado em Ciência da Computação) – Instituto de Ciências exatas e Tecnológicas, Feevale. Novo Hamburgo, RS, 2005.

BREITMAN, Karin. **Web Semântica: A internet do futuro.** Rio de Janeiro: LTC, 2005.

CHAVES, Marcirio Silveira. **Um estudo sobre XML, Ontologias e RDF(S).** 2001. Disponível em: <http://www.inf.pucrs.br/~mchaves/pg_portugues/tc/paperxml.pdf>. Acessado em 28 de Maio de 2010.

EDC – Educação à distância 2007.2 – Disponível em: <http://www.moodle.ufba.br/mod/book/view.php?id=10902>. Acessado em 02 de abril de 2010.

GRUBER, Tom. **Ontology (Computer Science) – Definition in Encyclopedia of Database Systems.** 2007. Disponível em: <http://tomgruber.org/writing/ontology-definition-2007.htm>. Acessado em 25 de Maio de 2010.

GUARINO, Nicola. **Formal Ontology and Information Systems.** 1998. Disponível em: <http://www.loa-cnr.it/Papers/FOIS98.pdf>. Acessado em 08 de Maio de 2010.

GUARINO, Nicola. **Understanding, Building, and Using Ontologies**. 1997. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.29.2610>. Acessado em 27 de Abril de 2010.

GUIZZARDI, Giancarlo. **Uma abordagem metodológica de desenvolvimento para e com reuso, baseada em ontologias formais de domínio.** Vitória, 2000. Disponível em: <http://www.loa-cnr.it/Guizzardi/MSc.htm>. Acessado em 05 de Maio 2010.

Jena Semantic Web Framework. Disponível em: <http://jena.sourceforge.net/> Acessado em 10 de Setembro de 2010.

LICHTENSTEIN, Flávio. **Criando uma Ontologia em Saúde com a ferramenta Protégé no padrão OWL.** 2008. Disponível em: <http://www.sbis.org.br/cbis11/arquivos/992.pdf>. Acesso em 15 de Abril de 2009. 6p.

Notepad++. Disponível em: <http://notepad-plus-plus.org/: Acessado em 3 de Agosto de 2010.

Ontoprise: Ontostudio. Disponível em: <http://www.ontoprise.de/en/home/products/ontostudio/>. Acessado em 08 de Junho de 2010.

Protégé: Using SPARQL in Protégé-OWL. Disponível em: <http://protege.stanford.edu/doc/sparql/>. Acessado em 10 de Junho de 2010.

RAP – Rdf API for PHP. Disponível em: <http://www4.wiwiss.fu-berlin.de/bizer/rdfapi/index.html> Acessado em 15 de Junho de 2010.

REZENDE, Solange Oliveira. Sistemas inteligentes: Fundamentos e aplicações. Barueri – SP: Manole, 2005.

RDQL – A query language for RDF. Disponível em <http://www.w3.org/Submission/RDQL/>. Acessado em 09 de Junho de 2010.

SPARQL Query Language for RDF. Disponível em: <http://www.w3.org/TR/rdf-sparql-query/>. Acessado em 09 de Junho de 2010.

The Protégé Ontology Editor and Knowledge Acquisition System – Disponível em: <http://protege.stanford.edu>. Acessado em 25 de Março de 2010.

Twinkle: A SPARQL Query Tool. Disponível em: <http://www.ldodds.com/projects/twinkle/>. Acessado em 09 de Junho de 2010.

USCHOLD, Mike; GRUNINGER, Michael. **Ontologies: Principles, Methods and Applications.** 1996. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.48.5917>. Acesso em 04 de junho de 2010.

ANEXO A – CÓDIGO DA TELA INICIAL DO QUIZ ONTOMÚSICA

```
<html>
      <head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
      <title> QUIZ ONTOMÚSICA</title>
      </head>
      <body style="background-color:#CCCCFF;">
      <?php
      echo "<h2 align='center'>";
      echo "<b> BEM VINDO AO QUIZ ONTOMÚSICA!! </b>";
      echo "</h2>";
      ?>
      <form action = "http://localhost/pesquisa_arc2.php"method=POST>
            <p><strong>Escolha o assunto a ser usado como base para as
                        perguntas do Quiz Ontomúsica:</strong>
            <p>
            <select name=assunto size="1">
            <option value="compositores e obras">Compositores e
                     Obras</option>
            </select>
            <input TYPE=SUBMIT VALUE="JOGAR!">
            </form>
            <P> </P>
            <form action = "lista_triplas_ARC2.php">
            <p><strong>Para visualizar as triplas da ontologia do Quiz
                        Ontomúsica, clique em "LISTAR TRIPLAS".</strong>
            <p>
            <input type="submit" value="LISTAR TRIPLAS" name = "enviar">
      </form>
      </body>
</html>
```

ANEXO B – CÓDIGO DA TELA “LISTAR TRIPLAS”

```
<html>
	<head>
	<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
	<title> QUIZ ONTOMÚSICA</title>
	</head>
	<body style="background-color:#CCCCFF;">
	<form action = "quiz_ARC2.php">
		<input type="submit" value="VOLTAR" name = "enviar">
	</form>
	<?php
	include_once("C:/Xampp/xampp/php/ARC2/ARC2.php");
	echo "<b> LISTA TODAS AS TRIPLAS: </b> ";
	$parser = ARC2::getRDFParser();
	$parser->parse('http://localhost/ontomusica.owl');
	$triples = $parser->getTriples();
	foreach($triples as $a => $b) {
		print "<table border=#>";
		print "<tr>";
		print ("<td>Sujeito: ".$b['s']."</td>");
		print ("<td>Predicado: ".$b['p']."</td>");
		print ("<td>Objeto: ".$b['o']."</td>");
		print "</tr>";
		print "</table>"; }
?>
</body>
</html>
```

ANEXO C – CÓDIGO DO QUIZ ONTOMÚSICA

```
<html>
<head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
      <title> QUIZ ONTOMÚSICA</title>
      </head>
      <body style="background-color:#CCCCFF;">
      <?php
      include_once("C:/Xampp/xampp/php/ARC2/ARC2.php");
      $config = array(
      'db_host' => 'localhost',
      'db_name' => 'my_db',
      'db_user' => 'ontomusica',
      'db_pwd' => 'senha',
      'store_name' => 'ontomusica',
      );
      $store = ARC2::getStore($config);
      if (!$store->isSetUp()) {
      $store->setUp();
      }
      $store->query('LOAD <http://localhost/ontomusica.owl>');

      if ($_POST['assunto'] == "compositores e obras") {
            echo "<h2 align='center'>";
            echo "<b> QUIZ SOBRE COMPOSITORES E OBRAS </b>";
            echo "</h2>";

            $q1 = '
            PREFIX
b:<http://www.rodrigo.goulart.nom.br/feevale/ontomusica/ontomusica.owl#>
            SELECT DISTINCT ?nome_compositor
            WHERE {     ?X b:compos ?Y.
                        ?X b:nome ?nome_compositor.
                        ?Y b:nome ?nome_obra}
            ';

            $a = $store->query($q1, 'rows');
            Shuffle ($a);

            for($i=0; $i<3; $i++) {
            $b = $a[$i]['nome_compositor'];
            $q2 = '
            PREFIX
b:<http://www.rodrigo.goulart.nom.br/feevale/ontomusica/ontomusica.owl#>
            SELECT ?nome_compositor ?nome_obra ?nome_genero
            WHERE {     ?X b:compos ?Y.
                        ?Y b:generoDaObra ?W.
                        ?X b:nome ?nome_compositor.
                        ?Y b:nome ?nome_obra.
                        ?W b:nome ?nome_genero.
            FILTER regex(?nome_compositor, "^'.$b.'").}
            LIMIT 1
            ';
            $c[$i] = $store->query($q2, 'rows');
            }

            $funcaoJavaScript = "   <script>
                  function resposta(){
```

```
            var respostaCerta = document.getElementById
            ('respostaCerta').value;
            var valorSelecionado = document.getElementById
            ('musica').value;

            if(valorSelecionado == respostaCerta)
            {
                  alert('Voce acertou!! Parabéns!!');
                  document.location.reload();
            }
            else
            {
            var gDaObra = document.getElementById
            ('generoDaObra').value;
            alert('Resposta errada! DICA: O gênero da obra correta é
                  ' + gDaObra);
            }
            }
            </script>";

      echo $funcaoJavaScript;
      $inicioForm = "<form action = '' method='POST'>";

      $temp = rand(0,2);
      $resposta_certa = $c[$temp][0]['nome_obra'];
      $compositor = $c[$temp][0]['nome_compositor'];
      $dica = $c[$temp][0]['nome_genero'];
      $opcao0 = $c[0][0]['nome_obra'];
      $opcao1 = $c[1][0]['nome_obra'];
      $opcao2 = $c[2][0]['nome_obra'];

      $dropDownMusica = "<select name='musica' id='musica' size='1'
                         onchange='resposta()'>";
      $dropDownMusica .= "<option value=''>Selecione</option>";
      $dropDownMusica .= "<option value='" . $opcao0 . "'>" .
                          $c[0][0]['nome_obra'] . "</option>";
      $dropDownMusica .= "<option value='" . $opcao1 . "'>" .
                          $c[1][0]['nome_obra'] . "</option>";
      $dropDownMusica .= "<option value='" . $opcao2 . "'>" .
                          $c[2][0]['nome_obra'] . "</option>";
      $dropDownMusica .= "</select>";

      $respostaCerta = "<input type='hidden' name='respostaCerta'
                        id='respostaCerta' value='" .
                        $resposta_certa . "'>";
      $generoDaObra = "<input type='hidden' name='generoDaObra'
                       id='generoDaObra' value='" . $dica . "'>";

      $fimForm = "</form>";
      echo "<br />Selecione abaixo a opção que complete corretamente
                  a afirmação:";
      echo $inicioForm . $dropDownMusica . " é uma obra do compositor
           " . $compositor . "." . $respostaCerta . $generoDaObra .
           $fimForm;
      }
      ?>
      <form action = "quiz_ARC2.php">
      <input type="submit" value="VOLTAR" name = "enviar">
      </form>
</body>
</html>
```