CODI4In + MovieShow: em busca de personalização de consultas baseadas no contexto do usuário

Marcelo Freitas[1], Livio Ribeiro[2], Thiago Moura[3], Damires Souza[4]

[1] marcello.dudk@gmail.com. [2] livioribeiro@outlook.com. [3] tmoura@gmail.com. [4] damires@ifpb.edu.br . Instituto Federal de Educação, Ciência e Tecnologia da Paraíba – Campus João Pessoa

RESUMO

Realizar consultas em sistemas computacionais pode ser uma tarefa desafiadora, visto que, quando uma consulta é submetida por vários usuários, normalmente, as mesmas respostas são retornadas, independentemente de suas preferências e do contexto no qual a consulta ocorreu. Para facilitar esse processo, uma abordagem centrada no usuário pode ser usada visando prover a personalização da consulta. Neste trabalho, essa personalização é realizada considerando o contexto do usuário. Para tal, foi desenvolvida uma primeira versão de um plugin chamado CODI4In, que provê a persistência e recuperação das informações contextuais do usuário. Essas são representadas por meio de uma ontologia e armazenadas em um banco de dados baseado em grafos. Neste trabalho, apresentamos os resultados obtidos com a implementação e experimentos realizados.

Palavras-chave: Contexto do usuário. Ontologia. Personalização de consultas. Plugin.

ABSTRACT

Submitting queries on computational systems can be a challenging task. This is due to the fact that when a query is submitted by diverse users, usually the same answers are returned, regardless of their preferences and of the context in which the query was submitted. To help matters, a user-centered approach can be used in order to provide query personalization. In this work, query personalization is accomplished by considering the user's context. To this end, a plugin named CODI4In has been developed, which provides the persistence and recovery of the user's contextual information. The user's contextual elements are represented by means of an ontology and stored in a graph based database. In this work, we present the results obtained with the implementation and accomplished experiments.

Keywords: *User context. Ontology. Query personalization. Plugin.*

1 Introdução

A busca por informações nos dias de hoje, estejam elas em um banco de dados local ou distribuídas em fontes diversas, tornou-se uma tarefa cada vez mais comum. Entretanto, o aglomerado de dados existentes dificulta essa busca e maximiza as chances de inexatidão diante das consultas que são normalmente executadas sem considerar o perfil do usuário. Para minimizar esse problema, a estratégia de personalização da consulta vem sendo estudada e implementada em diferentes ambientes e soluções (ARRUDA *et al.*, 2010; KOSTADINOV *et al.*,2007). A ideia por trás da personalização da consulta é que diversos usuários, ao realizar uma consulta, possam obter resultados diferentes como resposta de acordo com um *modelo de usuário* identificado (KOUTRIKA; IOANNIDIS, 2005).

Segundo Dey (2001), *contexto* se refere a qualquer informação que caracteriza a situação de uma entidade, em que uma entidade é uma pessoa, lugar ou objeto considerado relevante para a interação entre um usuário e uma aplicação. O *contexto* pode ser utilizado para ampliar o conhecimento que se tem sobre uma determinada situação, desempenhando um papel importante em domínios que envolvam requisitos como compreensão, raciocínio, resolução de problemas ou aprendizado (SANTORO *et al.*, 2006). A utilização de informações contextuais facilita a comunicação entre usuários e aplicações, além de ajudar a diminuir ambiguidades, facilitando também a identificação de perfis de usuários.

Neste trabalho, consideramos que o *contexto do usuário* pode ajudar a construir o *modelo do usuário* (perfil, interesses, histórico) e, consequentemente, pode contribuir na personalização de consultas, ajudando a prover usuários com respostas mais significativas. O desafio nesse sentido é como automaticamente adquirir, gerenciar e usar essa informação de contexto de usuário. Nesse panorama, este trabalho apresenta o *CODI4In*, um *plugin* que trata do armazenamento e recuperação de informações contextuais do usuário. Por se tratar de um *plugin*, é possível acoplá-lo a diferentes aplicações computacionais que envolvam o processamento de consultas de usuários.

Com o intuito de testar a abordagem definida e validar uma primeira versão do *CODI4In*, foi desenvolvida uma aplicação web de consultas a filmes chamada MovieShow. Por meio do acoplamento do *CODI4In* ao MovieShow foram realizados testes e experimentos com usuários reais que demonstraram diferenças significativas em termos de satisfação em relação às respostas obtidas (personalizadas) quando seu contexto era considerado.

Este trabalho está organizado da seguinte forma: a Seção 2 introduz conceitos, apresenta a ontologia CODI-User e a arquitetura do *CODI4In,* além de prover uma discussão sobre trabalhos relacionados; a Seção 3 descreve a implementação do *CODI4In*, apresenta a aplicação MovieShow e os resultados obtidos; por fim, a Seção 5 tece conclusões e indica trabalhos futuros.

2 Material e métodos

Contexto pode ser entendido como o que está por trás da habilidade de definir o que é ou não relevante em um dado momento (SOUZA *et al.*, 2008; DEY, 2001). Dessa maneira, todo o processo de personalização de consultas, incluindo a obtenção e tratamento de preferências, histórico de utilização, captura de informações dinâmicas, como localização do usuário, pode ser facilitado através do uso de contexto. Entretanto, gerenciar o contexto implica na implementação de tarefas como aquisição, representação, armazenamento, recuperação e uso do contexto. Nesse sentido, para podermos utilizar contexto, precisamos inicialmente de um modelo de representação que proveja uma mesma compreensão semântica tanto para agentes humanos quanto de *software*. Diante dessa premissa, optamos por utilizar uma ontologia como meio de representar e armazenar as informações contextuais dos usuários. A escolha de ontologias como modelo de representação de contexto possui algumas vantagens como: a ontologia é um modelo portável; é um modelo formal; permite reusabilidade; e provê mecanismos de inferência. Para este trabalho, realizamos uma extensão da ontologia de contexto para integração de dados denominada CODI (SOUZA *et al.*, 2008), com foco na entidade *User* (Usuário), como descrita a seguir. Nesse panorama, introduzimos a arquitetura do *CODI4In* e descrevemos alguns trabalhos relacionados a ele.

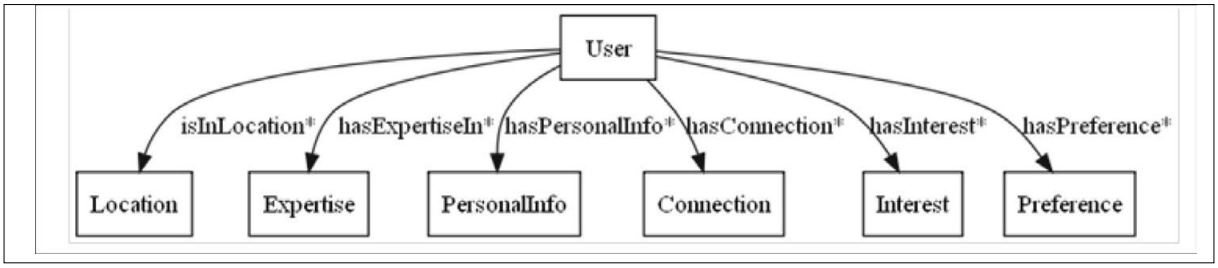
2.1 A ontologia de contexto CODI-User

A CODI (*Context Ontology for Data Integration*) é uma ontologia desenvolvida com base em requisitos da área de integração de dados e de gerenciamento de dados em ambientes P2P (SOUZA *et al.*, 2008). Na CODI, são definidos dois conceitos principais: enti-

dade de domínio (*Domain Entity*) e elemento contextual (*Contextual Element*). A ideia é que os elementos contextuais sejam construídos e/ou capturados sobre seis entidades de domínio: *user*, *environment*, *data*, *procedure*, *association* e *application*.

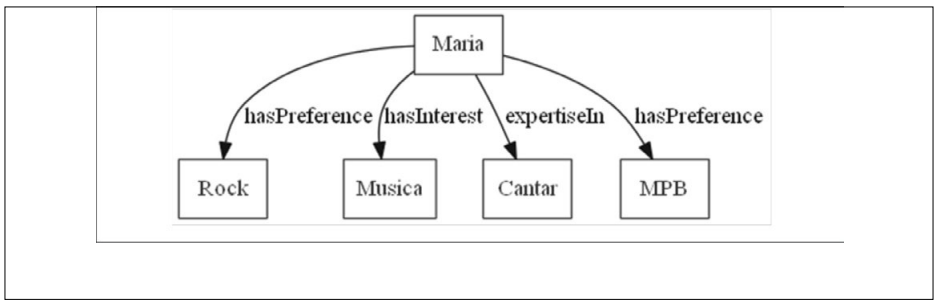
Com o propósito de criar uma ontologia de contexto de usuário independente de aplicações específicas, foi realizada uma extensão da CODI com foco na entidade de domínio *User* denominada CODI-User. Tendo em vista que a ontologia de contexto de usuário não deveria se ater apenas a preferências ou interesses, foram incorporados à CODI-User, além desses, elementos contextuais diversos como, por exemplo, expertise, habilidades, fatores físicos. Elementos comuns a ambientes de consulta como localização, dispositivo, interface e tarefa também foram incluídos. O conjunto de elementos contextuais vinculados a um determinado usuário forma o *modelo do usuário*. Uma visão da entidade de domínio *User* juntamente com alguns de seus elementos contextuais é apresentada na Figura 1. Neste fragmento, são mostrados apenas metadados referentes a relacionamentos da entidade *User* e alguns elementos contextuais (Location, *Expertise*, *PersonalInfo*, *Conection*, *Interest* e *Preference*).

Figura 1 – Fragmento da ontologia de contexto CODI-User.



A Figura 2 ilustra um exemplo de uma instância da ontologia de contexto CODI-User. Neste exemplo, utilizamos alguns elementos contextuais da nossa ontologia (*Interest*, *Preference* e *Expertise*) para descrever algumas informações sobre o usuário, criando, assim, uma estrutura inicial para o seu *modelo do usuário*. Essa estrutura poderá ser aprimorada a partir da identificação de novas informações relevantes para o usuário em questão ou da atualização das informações já identificadas. Assim, o conjunto de relacionamentos entre a entidade de domínio (*User*) e os elementos contextuais identificados nos permite inferir que Maria tem interesse em Música com preferência nos estilos Rock e MPB, além de ser expert em Cantar.

Figura 2 – Exemplo de instanciação da ontologia de contexto CODI-User.



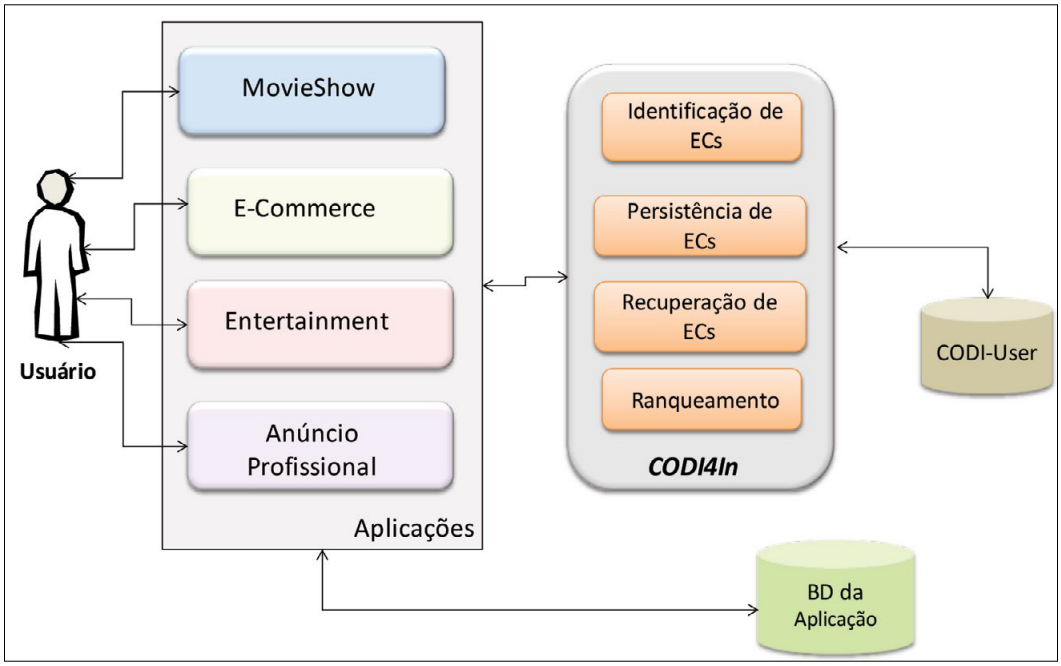
2.2 Arquitetura do CODI4In

O *CODI4In* foi especificado como um *plugin* que pode ser acoplado a diversos tipos de aplicações que envolvam consultas a dados. Ele tem como objetivo prover a obtenção e persistência de informações (elementos) contextuais, de forma dinâmica e incremental, a partir das interações dos usuários e de suas consultas. Em outras palavras, trabalha como um serviço *back-end,* ampliando funções de aplicações *front-end* a partir do uso das informações contextuais armazenadas na CODI-User. A Figura 3 apresenta

uma visão da arquitetura do *CODI4In* e de sua relação com aplicações *front-end*. Assim, qualquer aplicação que envolva consultas, como o MovieShow (descrita na Seção 3) e/ou o módulo de consultas de um sistema PDMS (*Peer Data Management System*) (ARRUDA *et al.*, 2010), são exemplos de aplicações *front-end* que poderão usar o serviço do *CODI4In*.

Dessa maneira, no momento em que o usuário realiza uma consulta, o serviço permite a coleta das informações (elementos contextuais) da base da CODI-User (modelo do usuário) e efetua a personalização dos dados (por exemplo, através de um ranqueamento dos resultados de uma consulta), valendo-se não só dos parâmetros da consulta, mas também dos elementos contextuais identificados e capturados pelo *CODI4In*. Como ilustração, suponha que um usuário submeta a seguinte consulta: "Mostre os pontos turísticos da cidade". No momento da submissão, o *CODI4In* identifica que o usuário se encontra localizado no Rio de Janeiro. Considerando esse elemento contextual (*localização*), o *CODI4In* efetua a personalização dos resultados, por meio de uma estratégia de ranqueamento, e exibe como primeiros resultados "visita ao Cristo Redentor" ou "Bondinho do Pão de Açúcar".

Figura 3 – Arquitetura do CODI4In.



2.3 Trabalhos relacionados

Os sistemas web são o maior foco de utilização de personalização na atualidade. Exemplos de serviços que utilizam personalização são o Google e o Amazon. O primeiro utiliza um histórico de cada usuário através de suas buscas e das páginas que foram mais visitadas para cada busca. O segundo, por sua vez, armazena o histórico de compras dos usuários e trabalha com sugestão de itens relacionados às compras efetuadas por eles.

Em termos de ambientes de bancos de dados, o trabalho de Koutrika e Ioannidis (2005) desenvolveu um framework de personalização para sistemas de bancos de dados baseado nos perfis dos usuários. Os perfis são criados e armazenados, podendo ser definidos de forma explícita, quando o usuário informa suas preferências, ou implicitamente pelo sistema, através da análise da interação do usuário com o banco. A cada preferência é designado um escore entre 0 e 1 para auxiliar na execução da consulta e na classificação das respostas obtidas.

Kostadinov e seu grupo (KOSTADINOV *et al.*, 2007) propuseram uma solução baseada em perfis de usuários para sistemas baseados em mediadores. Para tal, eles utilizam três metamodelos: (i) um modelo de perfis, composto por cinco dimensões (domínio

de interesse, dados pessoais, qualidade, segurança e apresentação), (ii) metamodelo de contexto que trata de informações espaço-temporais, e (iii) um metamodelo de preferências que organiza as preferências dos usuários.

Stefanidis *et al.* (2009) criaram um sistema de recomendação que expande resultados de consultas com base nas preferências dos usuários. Assim, tuplas que originalmente não seriam retornadas, são identificadas como sugestões. Por exemplo, ao se consultar por filmes dirigidos por Woody Allen, o sistema pode recomendar também sua biografia. Esse sistema pode computar os resultados das consultas utilizando o histórico do usuário, o estado corrente da consulta e da base de dados.

Comparando os trabalhos citados a este, percebe-se que, apesar da maioria deles também fazer uso das preferências do usuário para prover personalização, em geral, eles não o fazem através da identificação de informações contextuais específicas do usuário que submete às consultas. Este trabalho, por sua vez, permite construir um modelo de usuário (perfil, preferências, localização), através do contexto identificado (incluindo não somente elementos pessoais, mas também aqueles relacionados ao ambiente, às interações e à tarefa em questão). Além disso, provê um *plugin* que realiza a persistência dessas informações (representadas por meio de uma ontologia) através de um banco baseado em grafos. Esse *plugin* trabalha como um serviço back-end de gerenciamento do contexto, deixando aplicações front-end focadas apenas nos seus requisitos funcionais específicos.

3 Resultados e discussão

Nesta seção, a implementação do *CODI4In* e da aplicação MovieShow é descrita juntamente com os resultados obtidos ao realizar o acoplamento de ambos. São apresentados também os experimentos realizados.

3.1 O CODI4In

O *CODI4In* foi desenvolvido e implementado utilizando a linguagem Java e pode ser facilmente acoplado a qualquer aplicação sendo adicionado como uma biblioteca Java. Ontologias como a CODI-User podem ser interpretadas e tratadas como grafos (AN; BORGIDA, 2006), em que conceitos são identificados como nós e os relacionamentos semânticos (como especialização, parte de) são identificados como as arestas do grafo. Considerando essa premissa, foi adotado o Neo4J para persistência da CODI-User. O Neo4J é um banco de dados baseado em grafo que armazena dados como nós e arestas e provê algoritmos de indexação dos nós. A escolha do Neo4J se justificou devido às facilidades de utilização de um banco de dados que fizesse uso do mesmo paradigma de representação das informações contextuais adotado neste trabalho. Assim, tornou-se mais apropriado e natural mapear a ontologia de contexto (um grafo) para uma estrutura lógica e física também em termos de um grafo. Além disso, o Neo4J suporta o uso de OWL, linguagem utilizada para codificação da ontologia CODI-User.

3.2 O MovieShow

O MovieShow é uma aplicação web de consulta sobre filmes, também desenvolvida em Java, que utiliza um banco de dados relacional para armazenar seus dados. O esquema do banco é composto por tabelas que envolvem usuários, atores, diretores, filmes e gêneros de filmes, e foi populado com dados importados do IMDb.

O MovieShow foi especialmente projetado e implementado para este trabalho. Por meio dele, o usuário pode realizar consultas sobre filmes utilizando parâmetros como nome do diretor, do ator/atriz, entre outros e especificando os gêneros de sua preferência. À medida que o usuário faz as consultas, o *CODI4In* permite a captura das informações contextuais do usuário através de uma interface de gerenciamento do modelo do usuário e as armazena na CODI-User. Posteriormente, essas informações são recuperadas com o intuito de prover a personalização dos resultados. A estratégia de personalização adotada foi a de ranquear os resultados de acordo com os elementos contextuais do usuário relacionados às suas preferências. Dessa forma, a aplicação (junto com o *CODI4In*) retorna os resultados com o que mais interessa ao usuário no topo do ranking.

3.3 O CODI4In e o MovieShow na prática

Para que o MovieShow possa utilizar os serviços do *CODI4In*, o usuário precisa, em primeiro lugar, efetuar seu cadastro. Para isso, o MovieShow disponibiliza um formulário com as requisições de informações necessárias para realizar essa ação. Uma vez cadastrado, o usuário pode, então, realizar login no sistema. Quando essa ação ocorre, o MovieShow verifica se o

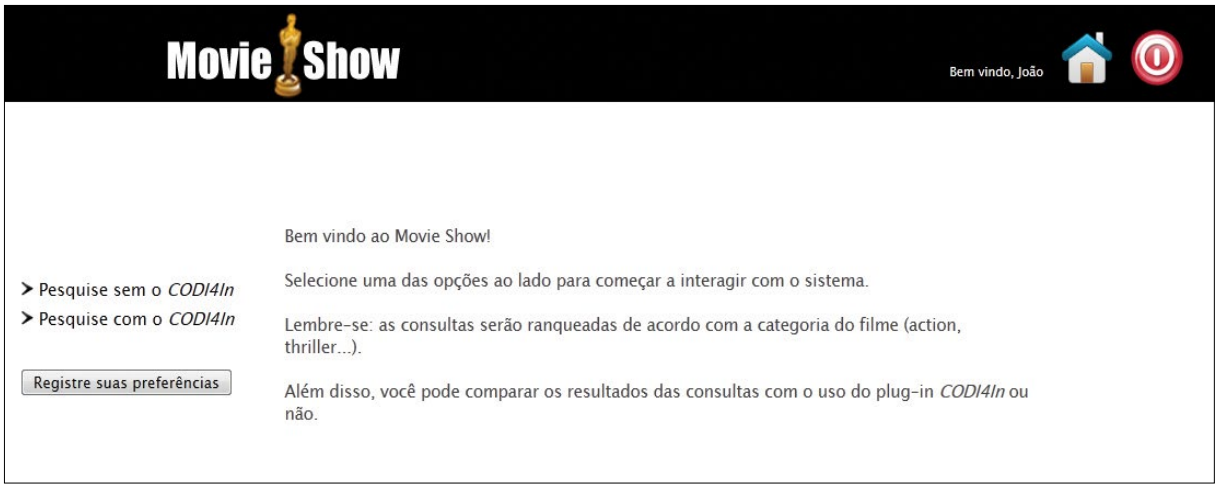
usuário está registrado em seu banco de dados. Em seguida, o *CODI4In* identifica esse usuário e verifica se a estrutura básica de nós da CODI-User para ele já está armazenada na base de dados do Neo4J. Caso negativo, o grafo contextual para o referido usuário (com os elementos contextuais) é montado, e os elementos (nós) são indexados e armazenados na base do Neo4J. O usuário, então, é direcionado à página inicial da aplicação (Figura 4), podendo, a partir deste ponto, utilizar as três funcionalidades básicas desenvolvidas nessa versão do MovieShow:

Registro de preferências: o usuário pode informar de forma explícita os gêneros de filme (por exemplo, suspense, comédia) de que mais gosta. Para isso, ele identifica uma lista de gêneros e cola no topo dela aqueles que mais lhe interessa, formando, assim, um *ranking* de preferências. O objetivo é que o *CODI4In* tome por base esse *ranking* de preferências e personalize as consultas de acordo com os gêneros listados pelo usuário, dando prioridade àqueles que estão no topo.

Submissão de consultas sem o uso do *CODI4In*: o MovieShow apenas efetua uma consulta padrão no banco de dados relacional utilizando os parâmetros submetidos pelo usuário. O objetivo dessa funcionalidade é mostrar os resultados das consultas sem a utilização das informações contextuais do usuário, ou seja, sem os serviços do *CODI4In*.

Submissão de consultas com o uso do *CODI4In:* o MovieShow efetua uma consulta no banco de dados, mas os resultados são reorganizados (ranqueados) tomando por base as preferências identificadas do usuário. Nessa versão, as preferências dizem respeito aos gêneros de filmes. O objetivo dessa funcionalidade, a principal no escopo deste trabalho, é identificar os benefícios obtidos quando considerado o contexto do usuário na realização da consulta.

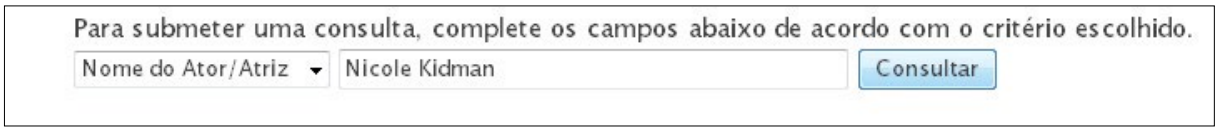
Figura 4 – Interface do MovieShow.



Caso haja alguma atualização dos elementos contextuais do usuário, essa atualização é aplicada aos dados e o *CODI4In* as armazena em sua base de dados (Neo4J), atualizando, também, a estrutura montada na memória em tempo de execução.

Para melhor entendimento do uso e funcionamento da ferramenta, ilustraremos um exemplo com dois usuários distintos. Neste exemplo, os usuários "Ben" e "Ana" tem um interesse em comum: filmes, entretanto possuem preferências diferentes com relação aos gêneros dos filmes. Ben gosta de filmes do tipo Documentário, Drama e Comédia, nessa ordem de prioridade. Ana, por sua vez, gosta de Drama, Comédia e Documentário, seguindo essa ordem de interesse. Na Figura 5, tela de consulta da aplicação, os usuários Ben e Ana submetem uma mesma consulta, com uso do *CODI4In*, sobre filmes estrelados pela atriz "Nicole Kidman". A consulta SQL é gerada tendo como condição de filtro o nome do ator/atriz selecionado e os elementos contextuais identificados pelo *CODI4In*. Os resultados da consulta são, então, reorganizados pelo *CODI4In* e exibidos de forma ranqueada pelo MovieShow.

Figura 5 – Consulta submetida pelos Usuários Ben e Ana.



Assim, as listas de filmes relacionados à atriz "Nicole Kidman" exibidas para os usuários Ben e Ana são diferentes, isto é, são personalizadas, como mostra a Figura 6. Na lista exibida para o usuário Ben, os filmes do gênero Documentário são mostrados primeiro, depois os do gênero Drama e por fim os do gênero Comédia, seguindo a ordem de suas preferências anteriormente cadastradas. O mesmo ocorre com o usuário Ana, que tem a sua lista de filmes exibida de acordo com suas preferências cadastradas.

Por meio desse exemplo, pode-se perceber que usuários com modelos diferentes, construídos a partir do uso de informações contextuais, que utilizem uma aplicação de consulta a dados em um domínio comum a ambos, ao submeterem uma mesma consulta, poderão receber resultados diferentes. Esses resultados podem estar personalizados segundo estratégias diferentes, como o ranqueamento efetuado nesse exemplo, de acordo com elementos contextuais específicos daquele usuário, como as preferências em relação aos gêneros de filmes.

Figura 6 – Resultados personalizados (ranqueados) da consulta submetida pelos usuários Ben e Ana, respectivamente.

Título	Gênero
101 Most Starlicious Makeovers	Documentary
76th Annual Academy Awards, The	Documentary
Climbing 'Cold Mountain'	Documentary
American Darlings	Drama, Musical
Birth	Drama, Mystery
Emma's War	Drama
Interpreter, The	Drama, Thriller
Bewitched	Comedy, Fantasy
58th Annual Tony Awards	Music

Resultados para Ben

Título	Gênero
American Darlings	Drama, Musical
Birth	Drama, Mystery
Emma's War	Drama
Interpreter, The	Drama, Thriller
Bewitched	Comedy, Fantasy
101 Most Starlicious Makeovers	Documentary
76th Annual Academy Awards, The	Documentary
Climbing 'Cold Mountain'	Documentary
58th Annual Tony Awards	Music

Resultados para Ana

3.4 Experimentos

Com o intuito de validar a eficácia do *CODI4In* quando acoplado ao MovieShow, foram realizados alguns experimentos com usuários reais. A ideia era medir o grau de satisfação do usuário em relação à relevância dos resultados obtidos através das consultas realizadas sem a consideração do contexto (isto é, *CODI4In* desativado) e com a consideração do contexto (isto é, *CODI4In* ativado). Para isso, convidamos um grupo de usuários composto, em sua maioria, por alunos de graduação em Sistemas para Internet do IFPB.

Os experimentos foram realizados em etapas. Primeiro, foi feita uma breve explanação sobre o que é o *CODI4In* e o MovieShow e explicados os objetivos dos testes. Os usuários, então, receberam um questionário com perguntas divididas em três tópicos: (i) dados pessoais (idade, profissão), (ii) consulta sem contexto, e (iii) consulta com o contexto. Em seguida, foram realizadas as atividades:

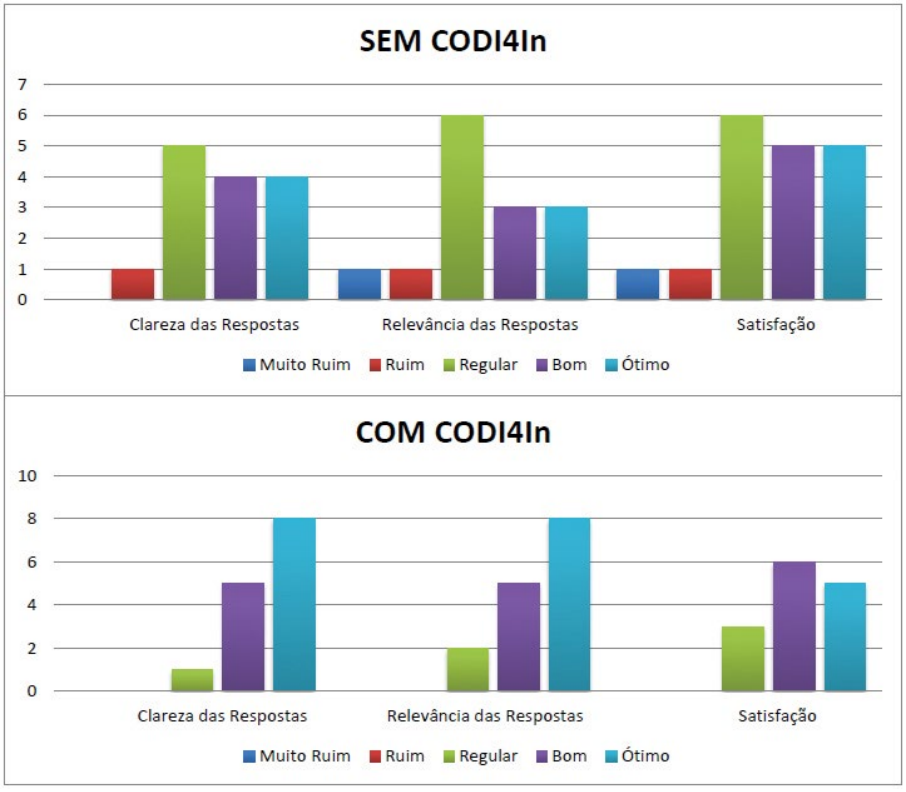
1. Os usuários foram convidados a definir suas preferências relacionadas aos gêneros dos filmes numa ordem de prioridade de interesse (registro de preferências).

2. Eles experimentaram a submissão de consultas com o plugin desabilitado. Registraram as consultas submetidas e verificaram os resultados obtidos para cada uma delas. Depois, responderam às perguntas apresentadas no questionário para o tópico (ii).

3. Eles habilitaram o *CODI4In* e submeteram o mesmo conjunto de consultas já realizadas na etapa 2. Depois de analisar os novos resultados produzidos, eles responderam as perguntas do questionário para o tópico (iii).

A Figura 7 mostra um resumo dos resultados obtidos com o *feedback* dos usuários quando deparados com as respostas das consultas realizadas sem e com consideração do contexto (sem e com ativação do *CODI4In*). Percebe-se que o grau de clareza, relevância e satisfação geral com as respostas obtidas cresceu significativamente quando o contexto foi considerado. Observar preferências (no tocante aos gêneros de filmes) como informação contextual foi um primeiro passo na comprovação da eficácia da abordagem e de quão promissora ela é para futuras extensões.

Figura 7 – Comparativo de *feedback* dos usuários em relação ao uso do CODI4In.



4 Conclusões

Este trabalho apresentou o *CODI4In*, um *plugin* para gerenciamento de informações (elementos) contextuais do usuário que permite a identificação, persistência e recuperação desses elementos. Para representação e armazenamento dos elementos contextuais, foi desenvolvida uma ontologia de contexto (CODI-User). Essa ontologia é persistida através de um banco de dados baseado em grafos.

O *CODI4In* foi desenvolvido como um *plugin* que pode ser acoplado a qualquer sistema que envolva consulta a dados. Para testar essa versão do *CODI4In*, realizamos um estudo de caso onde o mesmo foi acoplado à aplicação MovieShow. Os experimentos realizados com os usuários mostraram que as respostas das consultas tornaram-se mais relevantes e claras quando o contexto foi considerado.

Atualmente, estamos interessados em agregar outros elementos contextuais ao *plugin*, tais como a localização do usuário, o dispositivo utilizado para efetuar a consulta, entre outros, e avaliar os benefícios obtidos com a combinação desses novos elementos. Como trabalho futuro, será acrescentado ao *CODI4In* um mecanismo de raciocínio, como forma de prover resultados ainda mais personalizados e relevantes ao usuário.

REFERÊNCIAS

AN, Y.; MYLOPOULOS, J.; BORGIDA A. Building Semantic Mappings from Databases to Ontologies, In: PROCEEDINGS OF THE TWENTY-FIRST NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE (AAAI-06) Nectar Track, Boston, MA, 2006.

ARRUDA, T.; SOUZA, D.; SALGADO, A. C. PSemRef: Personalized Query Reformulation based on User Preferences. In: 12th INTERNATIONAL CONFERENCE ON INFORMATION INTEGRATION AND WEB-BASED APPLICATIONS & SERVICES, 2010, Paris. **Anais...** doiiWAS2010. New York: ACM, 2010. p. 681-684.

DEY, A. Understanding and Using Context. Personal and Ubiquitous Computing, Londres, v. 5, p. 4-7, fev. 2001.

KOSTADINOV, D.; BOUZEGHOUB, M.; LOPES S. Query Rewriting based on User's Profile Knowledge. In: ACTES DES 23 EMES JOURNÉES BASES DE DONÉES AVANCÉES. **Anais...** BDA 2007, France.

KOUTRIKA, G.; IOANNIDIS, Y. Personalized Queries under a Generalized Preference Model. In: 21st INTL. CONF. ON DATA ENGINEERING (ICDE). **Anais...** ICDE 2005, Tokyo, p. 841-852.

SANTORO, F. M.; BRÉZILLON, P.; ARAÚJO, R. M. Context Dynamics in Software Engineering Process. **Anais..** CSCWD 2006, p. 377-388.

SOUZA, D.; BELIAN, R.; SALGADO, A. C; TEDESCO P. Towards a Context Ontology to Enhance Data Integration Processes. In: 4TH WORKSHOP ON ONTOLOGIES-BASED TECHNIQUES FOR DATABASES IN INFORMATION SYSTEMS AND KNOWLEDGE SYSTEMS (ODBIS), Auckland. New Zealand: **Anais... ODBIS 2008**. p. 24-30.

STEFANIDIS, K.; DROSOU, M.; PITOURA, E. You May Also Like Results in Relational Databases. In: 3RD INTERNATIONAL WORKSHOP ON PERSONALIZED ACCESS, PROFILE MANAGEMENT AND CONTEXT AWARENESS IN DATABASES (PersDB), 2009, in conjunction with the VLDB 2009. **Anais... PersDB**, Lyon, ago. 2009.