ESCOLA
POLITÉCNICA DE
PERNAMBUCO

Trabalho de Conclusão de Curso

Engenharia da Computação

UM SERVIÇO DE RECOMENDAÇÃO INTER-APLICAÇÕES BASEADO EM FILTRAGEM COLABORATIVA

Jefferson Silva de Amorim

Orientador: Prof. Dr. Byron Leite Dantas Bezerra

UNIVERSIDADE
DE PERNAMBUCO

ESCOLA
POLITÉCNICA DE
PERNAMBUCO

JEFFERSON SILVA DE AMORIM

UM SERVIÇO DE RECOMENDAÇÃO INTER-APLICAÇÕES BASEADO EM FILTRAGEM COLABORATIVA

Monografia apresentada como requisito parcial para obtenção do diploma de Bacharel em Engenharia da Computação pela Escola Politécnica de Pernambuco – Universidade de Pernambuco.

Recife, Dezembro de 2010.

Aos meus pais e irmão, por todo apoio e carinho.

Agradecimentos

Primeiramente, agradeço aos meus pais, Gabriel Alves de Amorim e Maria Madalena Silva Amorim, por todo o apoio, dedicação e amor concedidos a mim durante todo o meu percurso.

Agradeço ao meu irmão, Nathanael Alves Amorim, por ter sempre me apoiado, incentivado e servido de exemplo, tanto para mim, quanto para as pessoas ao seu redor.

Agradeço a minha namorada, Mitzi Mychelle Tavares dos Santos Pereira, por ser minha amiga, companheira e por estar sempre ao meu lado, seja nos momentos felizes ou tristes.

Agradeço a todos meus amigos, que estiveram comigo nessa árdua jornada, com os quais posso dizer que aprendi bastante.

Agradeço ao Prof. Dr. Fernando José Castor de Lima Filho por ter me dado a oportunidade de ser seu aluno de iniciação científica e assim participar de pesquisas ao seu lado.

Agradeço ao Prof. Dr. Byron Leite Dantas Bezerra, por toda sua disponibilidade, dedicação e boa vontade durante toda orientação desse trabalho.

Por fim, agradeço a todos os professores que contribuíram com a minha formação, desde o colegial, passando pelo nível técnico até a graduação.

Resumo

A qualidade das recomendações geradas pelos sistemas de recomendação está fortemente relacionada à quantidade de informações previamente obtida do usuário para o qual está se sugerindo algo. O grande problema ocorre quando existe uma baixa freqüência de interação do usuário com a aplicação que faz uso desses sistemas. Isso, por exemplo, ocorre com freqüência em lojas virtuais. Sendo assim, este trabalho propõe uma metodologia de recomendação inter-aplicações utilizando técnicas de filtragem colaborativa. O sistema proposto busca realizar recomendações aos usuários a partir de seus perfis obtidos em diferentes aplicações. Na metodologia proposta, foram utilizadas técnicas de *Service-Oriented Architecture* (SOA) e um algoritmo de filtragem colaborativa baseada em *k Nearest Neighbor* (kNN) e no coeficiente de correlação de Pearson. Desta maneira, esta dissertação possui o objetivo primário de analisar as recomendações obtidas através da metodologia proposta e a mesma técnica de filtragem colaborativa utilizada por ela, porém utilizando somente o perfil do usuário obtido em uma única aplicação. As análises foram conduzidas com o propósito de demonstrar que houve o reaproveitamento das informações do usuário, obtidas em diferentes aplicações, ao se realizar recomendações. Os resultados obtidos foram satisfatórios e indicam que o método proposto traz diversos benefícios, tais como baixo acoplamento e um bom reaproveitamento dos perfis dos usuários obtidos em diferentes aplicações, além de se utilizar de uma técnica de recomendação clássica.

Palavras-chave: Sistemas de Recomendação, Filtragem Colaborativa e Arquitetura Orientada a Serviços.

Abstract

The quality of the suggestions generated by recommender systems is strongly related to the amount of information previously obtained the user for which you are suggesting something. The big problem occurs when there is a low frequency of user interaction with the application that makes use of recommendation systems. This, for example, occurs frequently in virtual stores. Thus, this paper proposes a methodology for inter-applications recommendation using collaborative filtering techniques. The proposed system makes recommendations to users based on their profiles obtained in different applications. In the proposed methodology, were used techniques of Service-Oriented Architecture (SOA) and a collaborative filtering algorithm based k nearest neighbor (kNN) and the Pearson's Correlation Coefficient. Thus, this work has the primary objective of analyzing the recommendations obtained by the proposed methodology and the same technique of collaborative filtering used by her, but using only the user profile obtained in a single application. The analyses were conducted with the aim of demonstrating that there was the reuse of user information, obtained in different applications, when performing recommendations. The results were satisfactory and indicate that the proposed method has several benefits, such as loose coupling and a good reuse of user profiles obtained in different applications, in addition to using a classic technique of recommendation.

Keywords: Recommender Systems, Collaborative Filtering and Service-Oriented Architecture.

Sumário

Índice de Figuras

Índice de Tabelas

Tabela de Símbolos e Siglas

(Dispostos em ordem alfabética)

- CMBF – *Content Modal Based Filtering*
- HMBF – *Hybrid Modal Based Filtering*
- HTTP – *HyperText Transfer Protocol*
- IARS – *Inter-Applications Recommendation Service*
- JSF – *JavaServer Faces*
- kNN – *k Nearest Neighbor*
- REST – *Representational State Transfer*
- RPC – *Remote Procedure Call Protocol*
- SMBF – *Social Modal Based Filtering*
- SOA – *Service-Oriented Architecture*
- SOAP – *Simple Object Access Protocol*
- UDDI – *Universal Description, Discovery and Integration*
- XML – *Extensible Markup Language*
- WSDL – *Web Service Definition Language*

Capítulo 1

Introdução

Este capítulo se inicia com a apresentação dos problemas e da motivação que levaram a construção desse trabalho de conclusão de curso. Logo em seguida, são descritos os seus principais objetivos. E por fim, ele é encerrado com uma breve descrição do que será abordado nos próximos capítulos.

1.1 Motivação e problema

Atualmente, diversas aplicações têm se utilizado de sistemas de recomendação para minimizar o problema de sobrecarga da informação e melhorar o relacionamento com os usuários [1]. Esse fato ocorre devido aos sistemas de recomendação se utilizarem de técnicas de personalização para tratar os usuários de forma personalizada, ou seja, tratá-los de acordo com suas necessidades e preferências. Ao se utilizarem dessas técnicas, as aplicações passam a se adequar à estratégia de marketing *one-to-one*, a qual possui o foco no cliente e tem sido cada vez mais utilizada [2]. Esta estratégia além de conceder ao cliente um maior grau de satisfação, aumenta a probabilidade de lucro por parte das empresas.

Os sistemas de recomendação são sistemas que realizam sugestões de itens (livros, filmes, comunidades, artigos, etc.) para um usuário baseadas em seu perfil [3]. O perfil do usuário é uma estrutura que contém as informações do usuário que são relevantes ao sistema de recomendação. Essas informações por sua vez podem ser obtidas de forma explícita ou implícita a cada interação do usuário com a aplicação [4]. Na forma explícita, o usuário fornece informações ao sistema à medida que ele avalia alguns itens encontrados na aplicação, indicando o seu grau de satisfação com os mesmos, ou preenche alguns formulários com seus dados demográficos, tais como endereço residencial, de nascença ou de trabalho. Entretanto, na forma implícita, o comportamento e as ações realizadas pelo usuário ao interagir com a aplicação são mapeadas implicitamente para preferências,

utilizando para isso funções definidas de forma coerente e representativa de acordo com o contexto da aplicação [3].

Dentre os métodos de personalização utilizados pelos sistemas de recomendação se encontram os de personalização de conteúdo [5]. Esses métodos podem ser agrupados em três técnicas: a baseada em conteúdo, a colaborativa, e a demográfica [3]. A técnica baseada em conteúdo sugere novos itens ao usuário de acordo com o nível de semelhança entre os itens sugeridos e seus itens preferidos. Já na técnica colaborativa ou de filtragem colaborativa, a recomendação de itens a um usuário é realizada a partir dos itens preferidos por um grupo de usuários semelhantes a ele. E na técnica demográfica se recomenda novos itens a partir dos usuários demograficamente semelhantes.

A grande maioria das aplicações atuais, que possuem sistemas de recomendação embutidos, utiliza técnicas de filtragem colaborativa [6]. Isto ocorre por que essas técnicas apresentam alta velocidade de resposta ao recomendar itens, nível satisfatório de qualidade das recomendações na maioria dos cenários, facilidade de extensão e alto grau de reuso dos métodos implementados. Além disso, elas possibilitam a realização de recomendações inovadoras e surpreendentes [3].

Apesar das vantagens apresentadas pelas técnicas de filtragem colaborativa, elas possuem alguns pontos fracos, tais como: o problema do novo usuário, do novo item, da esparsidade e da baixa qualidade de recomendação para usuários conhecidos como ovelhas negras [7]. Além desses problemas, a qualidade de recomendação dessa técnica é fortemente influenciada pela quantidade de informações do usuário obtidas ao longo do tempo. Contudo em algumas aplicações, por exemplo, as lojas virtuais, para se coletar informações suficientes dos usuários levam-se semanas ou até mesmo meses [3]. Isto se caracteriza devido à baixa freqüência de compra dos produtos por parte dos usuários e por não existir um reaproveitamento dos perfis deles obtidos por outras aplicações, seja de domínio semelhante ou não.

Sistemas de recomendação têm sido aplicados em diversos domínios e mostraram ser de grande valia aos mesmos [8]. Porém as implementações atuais são fortemente acopladas ou ao domínio da aplicação, por definirem uma estrutura

para o perfil do usuário fortemente ligada a ele, ou à própria aplicação, devido às implementações serem muitas vezes realizadas diretamente no código da mesma [9].

Com o objetivo de diminuir o acoplamento entre as aplicações e os sistemas de recomendação, alguns trabalhos realizados consideram uma implementação que se utilize dos recursos de SOA (do inglês *Service-Oriented Architecture*) [9]. A idéia de utilizar SOA na construção de sistemas de recomendação traz diversos benefícios, tais como o reuso, a interoperabilidade e o baixo acoplamento com a aplicação [10]. Todavia os sistemas de recomendação ainda permanecem fortemente acoplados ao domínio da aplicação.

O presente trabalho se dedicou a solucionar o problema do reaproveitamento de informações do usuário por parte dos sistemas de recomendação. Para isso foi criado um serviço de recomendação que se utiliza de recursos de SOA e de uma técnica de filtragem colaborativa baseada em kNN [11] (do inglês *k Nearest Neighbor*) e no coeficiente de correlação de Pearson [12].

1.2 Objetivos

Neste trabalho criaremos um serviço de recomendação baseado em filtragem colaborativa que realize recomendações a partir dos perfis dos os usuários obtidos em diversas aplicações. Por exemplo, ao se utilizar desse serviço, sistemas como lojas virtuais poderiam recomendar itens a usuários a partir dos seus perfis obtidos em redes sociais, outras lojas virtuais, etc. Desta forma, essa abordagem traz diversos benefícios ao se utilizar de perfis do usuário provenientes de diversas aplicações, sendo o mais importante, a minimização do problema do novo usuário e da baixa qualidade de recomendação para usuários conhecidos como ovelhas negras.

1.2.1 Objetivos específicos

- Realizar uma revisão da literatura em torno de Sistemas de Recomendação e, sobretudo, levantar propostas de sistemas e técnicas de recomendação inter-aplicações.

- Adaptar uma implementação de uma técnica de filtragem colaborativa baseada em kNN e no coeficiente de correlação de Pearson para recomendar itens a partir dos perfis dos usuários obtidos em mais de uma aplicação.
- Desenvolver um *Web Service* que expõe as principais funcionalidades da técnica de filtragem colaborativa adaptada.
- Desenvolver uma ferramenta web para coleta de dados e suporte à análise de recomendações.
- Analisar as recomendações realizadas pelo serviço de recomendação desenvolvido.

1.3 Estrutura da monografia

Este trabalho está divido em cinco capítulos. No primeiro capítulo foi dada uma introdução ao tema Um serviço de recomendação inter-aplicações baseado em filtragem colaborativa, assim como a motivação e a caracterização dos problemas.

Capítulo 2: Revisão Bibliográfica

Neste capítulo, será abordado o conteúdo teórico necessário para o desenvolvimento do tema proposto: Sistemas de recomendação e Arquitetura Orientada a Serviços.

Capítulo 3: IARS - Inter-Applications Recommendation Service

Neste capítulo, apresentaremos o IARS (do inglês *Inter-Applications Recommendation Service*), um serviço de recomendação inter-aplicações que utiliza uma técnica de filtragem colaborativa baseada em kNN e no coeficiente de correlação de Pearson. Sendo assim, apresentaremos como ele foi projetado e como ele consegue recomendar itens a partir dos perfis dos usuários obtidos em diferentes aplicações.

Capítulo 4: Avaliação do IARS

Neste capítulo, discorreremos sobre a ferramenta desenvolvida para realizar a coleta de preferências dos usuários em diferentes domínios e dar suporte à análise

de recomendações, a base de dados coletada e a análise realizada das recomendações geradas pelo IARS utilizando tal base.

Capítulo 5: Conclusão

Neste capítulo, apresentaremos uma visão geral do trabalho, dificuldades encontradas, discussões e as conclusões obtidas, além de trabalhos futuros e melhorias.

Capítulo 2

Revisão Bibliográfica

Neste capítulo abordaremos todo o conteúdo teórico necessário para solucionar o problema descrito no Capítulo 1 deste trabalho. A seção 2.1 mostra como funcionam os sistemas de recomendação, um sistema capaz de realizar sugestões de itens que possam ser de interesse do usuário a partir do seu perfil. Logo em seguida, a seção 2.2 fala sobre Arquitetura Orientada a Serviços (SOA), termo esse que representa um estilo arquitetural cuja idéia é disponibilizar funcionalidades de sistemas em forma de serviços.

2.1 Sistemas de Recomendação

São sistemas que se utilizam de tecnologias de personalização para oferecer uma experiência personalizada aos usuários deste sistema a partir das informações do seu perfil. Estes sistemas possuem normalmente duas funcionalidades principais: Gerar Lista Personalizada e Predizer Nota. Como estas funcionalidades são objetos de estudos para a maioria dos trabalhos realizados no contexto de sistemas de recomendação, elas serão as únicas abordadas neste trabalho.

A funcionalidade de Gerar Lista Personalizada, também conhecida como *Find Good Items* [13] e *Preference-Based Filtering* [6], apresenta ao usuário uma lista dos itens mais relevantes, onde a relevância do item é calculada a partir do perfil do usuário. Esta lista de itens pode ser gerada utilizando todos os itens contidos no repositório de dados ou somente um subconjunto dos mesmos.

A geração de lista personalizada a partir de um subconjunto de itens, geralmente, é utilizada com a finalidade de executar determinadas decisões estratégicas. Por exemplo, em sistemas de comércio eletrônico muitas vezes, utiliza-se tal funcionalidade quando se deseja impulsionar as vendas de uma dada categoria de produtos. Sendo assim, esta funcionalidade fornece os mecanismos necessários às empresas para seguir os preceitos da abordagem *Marketing One-to-One* [1], uma das estratégias de marketing mais popular no meio empresarial.

A funcionalidade de Predizer Nota, também conhecida como *Annotation in Context* [13], tem por objetivo estimar a relevância de um dado item para um usuário utilizando as informações do seu perfil. Essa abordagem é semelhante à descrita na funcionalidade de Gerar Lista Personalizada.

Conforme dito anteriormente, os Sistemas de Recomendação se utilizam das Tecnologias de Personalização para tratar os usuários de forma customizada. Os métodos e algoritmos dessas tecnologias são implementados em um subsistema denominado Engenho de Personalização, que por sua vez servem de núcleo para as funcionalidades de Gerar Lista Personalizada e Predizer Nota dos Sistemas de Recomendação.

2.1.1 Engenho de Personalização

Ele é o responsável por fornecer ao usuário uma experiência diferenciada ao interagir com aplicações que fazem uso de Sistemas de Recomendação. Conforme a **Figura 1**, o Engenho de Personalização é constituído pelo módulo de aquisição, o módulo de recomendação e o repositório de perfis [3].

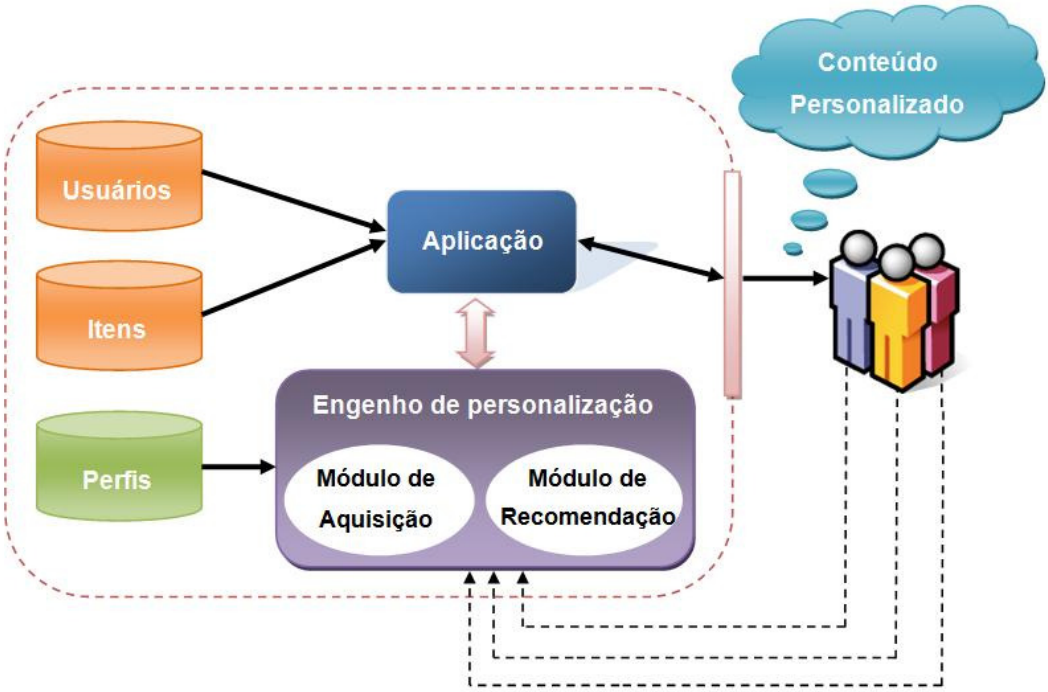


Figura 1. Integração de uma Aplicação com o Engenho de Personalização.

O módulo de aquisição é o responsável por obter os dados relevantes do usuário e registrá-los no repositório de perfis. Primeiramente, ele recebe os dados

resultantes das interações do usuário com a aplicação. Logo em seguida, os dados coletados passam por um processo de transformação. A complexidade desse processo depende da forma como o perfil do usuário foi estruturado. Quanto menos organizado for o perfil, mais complexo será o processo de transformação. Por último, é realizada a atualização do perfil no repositório.

O módulo de recomendação é responsável por fornecer recomendações personalizadas ao usuário a partir dos dados que se encontram no repositório de perfis. Esse módulo inicia sua operação carregando o perfil do usuário para o qual a aplicação deseja recomendar algo. Logo em seguida, os itens são carregados e caso seja necessário, passam por um processo que os restringe a partir do contexto da aplicação em que o usuário se encontra. Com os itens carregados, calcula-se a relevância ou a utilidade destes para o usuário. Após essas etapas, o módulo de recomendação formata a saída de acordo com a operação solicitada, ou seja, prediz a nota, seleciona itens com maior utilidade ou gera lista em ordem decrescente de utilidade. Contudo, para fornecer recomendações personalizadas a um usuário, o módulo de aquisição deve ter sido executado ao menos uma vez para o dado indivíduo.

O repositório de perfis, também conhecido como base de conhecimento, é o local utilizado pelo Engenho de Personalização para armazenar os perfis dos usuários. Estes perfis, por sua vez, são estruturados de acordo com os algoritmos de recomendação implementados pelo engenho. A idéia do repositório de perfis foi concebida com o objetivo de permitir o desenvolvimento de um engenho de recomendação menos acoplado à aplicação e mais modular.

Os algoritmos e métodos implementados pelo Engenho de Personalização são técnicas de personalização de conteúdo, ou seja, técnicas que possuem como entrada o perfil do usuário [3].

2.1.2 Personalização de conteúdo

A Tecnologia de Personalização pode ser utilizada para realizar personalização em nível de interface ou de conteúdo [5]. A personalização de interface se restringe a proporcionar, por exemplo, uma navegação customizada em uma aplicação ou a possibilidade de modificação do layout do site por parte do

usuário, enquanto a personalização de conteúdo se utiliza de informações do usuário para recomendar músicas, enviar e-mail oferecendo livros de acordo com o gosto do usuário, etc.

O problema de personalização de conteúdo é definido da seguinte forma. Seja $\boldsymbol{U}$ o conjunto de todos os usuários e $\boldsymbol{I}$ o conjunto de todos os itens, existe uma função $\boldsymbol{\rho}$ responsável por medir a utilidade de um item $\boldsymbol{i}$ para um usuário $\boldsymbol{v}$, tal como apresentado na **Equação 2.1**. Onde Γ é uma relação no conjunto dos números reais positivos incluindo o elemento neutro. Entretanto, a idéia básica do problema de personalização de conteúdo, conforme a **Equação 2.2**, é encontrar um item $\boldsymbol{i'} \in \boldsymbol{I}$ para cada usuário $\boldsymbol{v} \in \boldsymbol{U}$, tal que maximize a utilidade para o usuário [3].

$$\rho: U \; x \; I \; \longrightarrow \; \Gamma \tag{2.1}$$

$$\forall v \in U, i'_v = \arg\max_{i \in I} \rho(v, i) \tag{2.2}$$

A função utilidade é o principal componente do módulo de recomendação de um Engenho de Personalização. Como visto anteriormente, ela é utilizada no momento de calcular a relevância de um item para um determinado usuário. Porém, pode-se observar que ela corresponde diretamente à função de Predizer Nota. Para realizar a funcionalidade de Gerar Lista Personalizada, precisaria adaptar a **Equação 2.2** para recuperar um conjunto de itens $\boldsymbol{N}$, tal que a relevância desses itens seja máxima [3]. O pseudocódigo desta adaptação pode ser observado na **Tabela 1**. Como entrada para a função ***ObterRecomendacoes*** temos o usuário alvo $\boldsymbol{v}$, o conjunto de usuários do repositório $\boldsymbol{U}$, o conjunto de itens do repositório $\boldsymbol{I}$ e o número de itens a serem recomendados $\boldsymbol{N}$. O retorno desta função é uma lista contendo os itens recomendados $\boldsymbol{S_v}$, ou seja, os itens de maior relevância.

O perfil do usuário é uma estrutura que contêm as informações do usuário que sejam relevantes ao sistema de recomendação. As técnicas de personalização de conteúdo recebem essa estrutura como entrada. O perfil do usuário pode ser estruturado de forma a conter informações que vão desde uma simples associação de um item a um indicador de relevância, como por exemplo, uma nota, até a descrição de itens e dados pessoais do usuário [6]. Tais informações podem ser obtidas implícita ou explicitamente [4].

Tabela 1. Pseudocódigo da funcionalidade de Gerar Lista Personalizada

ObterRecomendacoes(v, U, I, N) {

1. $S_v = \emptyset$
2. Enquanto $tamanho(S_v) \neq N$ faça {
3. $Q = I - S_v$
4. $i'_v = \arg\max_{i \in Q} \rho(v, i)$
5. $S_v = S_v \cup \{i'_v\}$
6. }
7. Ordenar S_v, caso seja necessário

}

As técnicas de personalização de conteúdo podem ser divididas em três categorias: a baseada em conteúdo, a colaborativa, e a demográfica [3]. Atualmente, a maioria das aplicações que fazem uso de sistemas de recomendação, se utiliza de técnicas de filtragem colaborativa para realizar as funcionalidades de Gerar Lista Personalizada e Predizer Nota [6]. Neste trabalho nos concentraremos em técnicas de filtragem colaborativa, logo as demais não serão detalhadas.

2.1.3 Filtragem Colaborativa

Os métodos de filtragem colaborativa realizam a predição da nota de um item para um determinado usuário a partir dos itens que foram previamente avaliados por outros usuários [6]. Formalmente esse método pode ser definido como o responsável por predizer a nota de um item $\boldsymbol{i'}$ para um dado usuário $\boldsymbol{v}$ tal que esta nota seja calculada a partir das notas dos itens $\boldsymbol{i} \in \boldsymbol{I}$ dadas pelos usuários $\boldsymbol{u} \in \boldsymbol{U}$ que são similares ao usuário $\boldsymbol{v}$.

O fundamento da filtragem colaborativa é basicamente identificar os usuários que possuem preferências semelhantes. Estes usuários são comumente chamados de vizinhos [3]. Após a identificação dos vizinhos, analisam-se quais dos itens avaliados por estes tiveram maior relevância, ou seja, os itens que obtiveram maior nota. A partir destes itens estima-se a relevância de um item ou gera-se uma lista com os itens mais relevantes para um dado usuário.

A função utilidade no contexto de filtragem colaborativa depende fundamentalmente da forma como será realizado o cálculo da semelhança entre os usuários. Porém, existem duas abordagens que podem ser utilizadas para fazer este cálculo: a *memory-based* e a *model-based* [7].

Na abordagem *memory-based*, heurísticas são definidas a fim de realizar a predição de nota a partir de todo conjunto de avaliações de itens feitas pelos usuários. Em contrapartida, na abordagem *model-based*, o conjunto de avaliações é utilizado na definição de um modelo, que será utilizado na predição de nota. Neste trabalho será utilizada uma técnica de filtragem colaborativa que segue a abordagem *memory-based*, portanto não entraremos em detalhes sobre a abordagem *model-based*.

Atualmente, existem várias medidas de similaridade que podem ser utilizadas por técnicas de filtragem colaborativa que se baseiam na abordagem *memory-based* para calcular a similaridade entre os usuários [14]. Uma das medidas clássicas usadas por tais técnicas é o coeficiente de correlação de Pearson [12], definido na **Equação 2.3**. Onde $\vec{v}$ e $\vec{u}$ são, respectivamente, os vetores de avaliações dos usuários v e u; $\overline{\vec{v}}$ e $\overline{\vec{u}}$ representam a média aritmética das avaliações realizadas pelos usuários v e u, respectivamente; $\vec{v}(i)$ é a avaliação do item i realizada pelo usuário v; e $\vec{u}(i)$ é a avaliação do item i realizada pelo usuário u.

$$\Psi(v,u) = \frac{\sum_{i \in I}\left(\vec{v}(i) - \overline{\vec{v}}\right) * \left(\vec{u}(i) - \overline{\vec{u}}\right)}{\sqrt{\sum_{i \in I}\left(\vec{v}(i) - \overline{\vec{v}}\right)^2 * \sum_{i \in I}\left(\vec{u}(i) - \overline{\vec{u}}\right)^2}} \tag{2.3}$$

Como visto anteriormente, após realizar a identificação dos usuários semelhantes, ainda é preciso predizer as nota de um item ou gerar uma lista com os itens mais relevantes a um dado usuário. Sendo assim, a técnica de filtragem colaborativa deve definir uma função utilidade que realize tais procedimentos. Para tal, pode-se utilizar uma função utilidade baseada em kNN [11], como apresentado na **Equação 2.4**, tendo em vista que esta é uma das mais utilizadas no contexto de filtragem colaborativa [15].

$$\rho(v,i) = \overline{\vec{v}} + \frac{\sum_{k=1}^{h}\left(\vec{u}_k(i) - \overline{\vec{u}}_k\right) * \Psi(v,u_k)}{\sum_{k=1}^{h}\Psi(v,u_k)} \quad (2.4)$$

Na **Equação 2.4**, $\boldsymbol{v}$ é o usuário para o qual será medida a relevância do item $\boldsymbol{i}$; $\boldsymbol{h}$ é o número de vizinhos a ser considerado; $\boldsymbol{u_k}$ representa o $\boldsymbol{k}$-ézimo vizinho do usuário $\boldsymbol{v}$; $\overline{\vec{v}}$ e $\overline{\vec{u}}_{\boldsymbol{k}}$ representam as médias aritméticas das avaliações dos usuários $\boldsymbol{v}$ e $\boldsymbol{u_k}$, respectivamente; e $\vec{\boldsymbol{u}}_{\boldsymbol{k}}(\boldsymbol{i})$ é a avaliação do usuário $\boldsymbol{u_k}$ para o item $\boldsymbol{i}$.

Os métodos de filtragem colaborativa possuem uma característica bastante peculiar: só precisam saber dos identificadores dos itens e dos usuários no repositório, além dos relacionamentos entre eles, para realizar recomendações. O relacionamento dos itens com os usuários é dado através das avaliações ou interações do usuário. Nesse sentido, esses métodos alcançam um nível de reusabilidade maior que os demais, além de ter a capacidade de ser aplicado em diferentes domínios [3]. Apesar das vantagens apresentadas pelos métodos de filtragem colaborativa, estes possuem algumas limitações, tais como o problema do novo usuário, novo item, esparsidade e das ovelhas negras [7].

O problema do novo usuário se refere aos usuários que nunca avaliaram nenhum item, ou seja, não há associações de itens com estes usuários. Enquanto o problema do novo item ocorre quando se cadastra um novo item e este ainda não foi avaliado por nenhum usuário ou foi avaliado por poucos. O problema da esparsidade está associado à quantidade de itens no repositório que nunca foram avaliados. Por fim, o problema das ovelhas negras se refere à baixa qualidade das recomendações dadas para usuários incomuns. Todos esses problemas influenciam diretamente na qualidade da recomendação fornecida pelo sistema.

Apesar das desvantagens apresentadas pelos métodos de filtragem colaborativa, eles ainda apresentam um nível satisfatório de qualidade das recomendações para a maioria dos cenários, facilidade de extensão e possibilitam a realização de recomendações inovadoras e surpreendentes [3].

2.2 Arquitetura Orientada a Serviços

Arquitetura Orientada a Serviços (SOA) é um modelo arquitetural que tem por objetivo aumentar a agilidade e a rentabilidade das empresas, utilizando para isso o paradigma de orientação a serviços. Este paradigma representa a lógica da solução como pequenas unidades, formadas individualmente, que podem ser utilizadas em conjunto a outras unidades ou individualmente distribuídas e reutilizadas por outros domínios [16]. Sendo assim, pode-se definir SOA como um modelo em que a lógica do negócio é decomposta em pequenas e distintas unidades, que podem vir a ser utilizadas coletivamente, formando um fluxo de negócio, ou individualmente distribuídas, permitindo um maior nível de reuso [17].

Atualmente, o termo “SOA” tem sido motivo de diversos debates realizados no meio corporativo. Alguns chegam acreditar que SOA tornou-se sinônimo para uma nova plataforma de computação devido à forma como ela é utilizada [17].

Como já foi dito anteriormente, SOA se utiliza de mecanismos de orientação a serviços para alcançar seus objetivos, entretanto existem quatro características que dão suporte aos princípios de orientação a serviços e que podem ser identificadas em qualquer forma de SOA. A primeira é a que diz que SOA é direcionada ao negócio, ou seja, a arquitetura tecnológica deve estar alinhada à arquitetura de negócio atual. A segunda está relacionada à independência de plataforma, ou seja, o modelo arquitetural da solução não depende de plataformas proprietárias. A terceira diz que SOA é centrada na empresa, ou seja, o escopo da arquitetura representa um segmento significativo da empresa. Por fim, SOA é direcionado a composição, ou seja, permite a agregação entre os serviços, tornando-os bastante adaptáveis [18].

2.2.1 Serviço

Serviços representam uma unidade lógica da solução na qual foi utilizado o paradigma de orientação a serviços. A aplicação deste paradigma é que distingue a representação de uma unidade lógica como um serviço da representação dada por um objeto ou componente [17].

Nos cenários onde se observa a presença de serviços podemos identificar diversos elementos importantes, tais como o consumidor do serviço, o descritor de

serviço, o serviço em si mais suas funcionalidades e as mensagens recebidas e enviadas pelo serviço. Um exemplo de cenário pode ser visto na **Figura 2**.

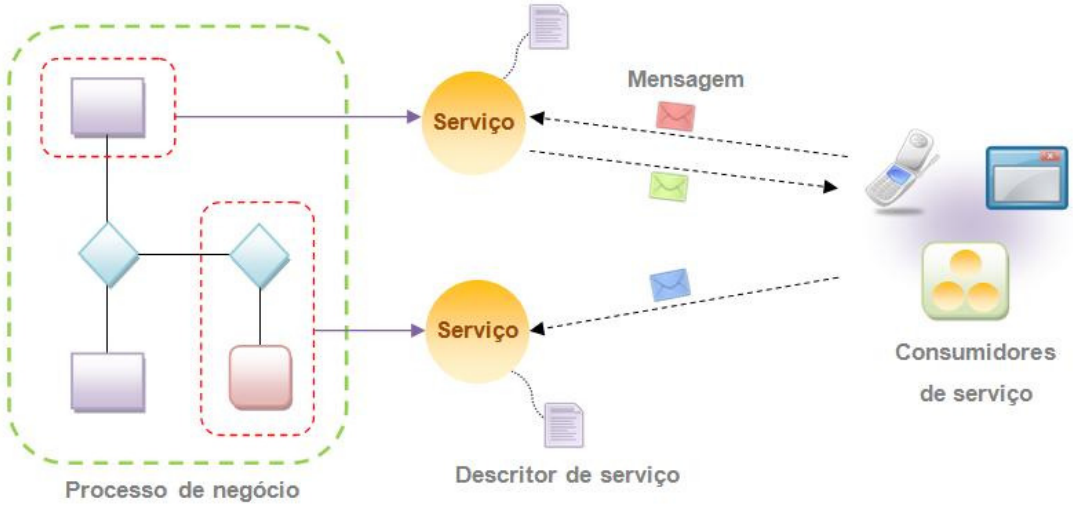


Figura 2. Principais elementos de um serviço.

O consumidor do serviço é um papel dado a um programa ou um serviço que interage ou invoca um determinado serviço. Este papel só pode ser observado em tempo de execução, quando está havendo uma troca de dados, ou mensagens, entre o serviço e o elemento que interage com ele. O descritor de serviço é o elemento responsável por expor os dados relevantes de um serviço. Estes dados são necessários para que outros serviços ou aplicações possam interagir com ele. O serviço é o elemento que encapsula um pequeno pedaço da lógica da solução dentro de um contexto distinto. Este contexto pode ser específico para uma tarefa de negócio, uma entidade de negócio ou qualquer outro agrupamento lógico. As mensagens são elementos responsáveis por encapsular os dados trocados na interação entre o consumidor do serviço e o próprio serviço. Sendo assim, as mensagens podem ser de dois tipos: Solicitação e Resposta. As mensagens de solicitação são enviadas do consumidor para o serviço, enquanto as de resposta seguem o fluxo inverso, ou seja, do serviço para o consumidor [16].

Um serviço pode ser implementado através de Componentes, REST (do inglês *Representational State Transfer*) ou *Web Services* [17]. Neste trabalho nos concentraremos em *Web Services*, devido ao fato de ser a tecnologia de implementação mais utilizada, portanto, as demais não serão detalhadas.

2.2.2 Web Services

Os *Web Services* utilizam e definem um conjunto de padrões para implementar os serviços e garantir um alto nível de interoperabilidade [18]. Dentre os padrões utilizados pelos *Web Services* temos o HTTP (do inglês *HyperText Transfer Protocol*) e o XML (do inglês *Extensible Markup Language*). Estes são definidos em [19] e [20], respectivamente. Enquanto os padrões fundamentais definidos pelos *Web Services* são o WSDL (do inglês *Web Service Definition Language*), o SOAP (do inglês *Simple Object Access Protocol*) e o UDDI (do inglês *Universal Description, Discovery and Integration*), definidos respectivamente em [21], [22] e [23]. Podemos ver na **Figura 3** onde alguns destes padrões são aplicados.

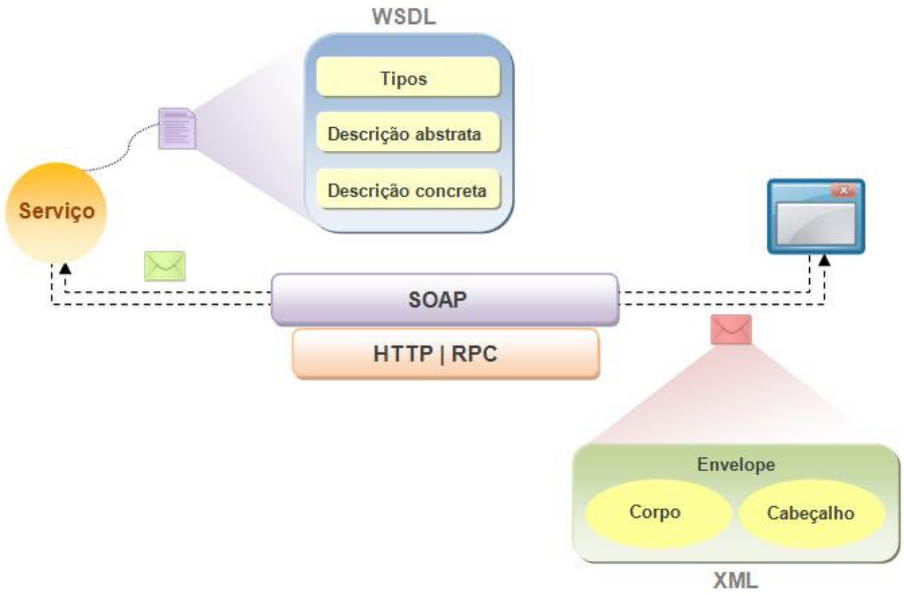


Figura 3. Padrões utilizados em *Web Services*.

WSDL é uma linguagem baseada em XML utilizada para descrever os *Web Services* [21]. Ou seja, ao implementar serviços através de *Web Services* utiliza-se WSDL para criar o descritor de serviço. Basicamente, um descritor de serviço feito utilizando-se WSDL é dividido em três seções: definição de tipos, descrição abstrata e descrição concreta. A seção de definição de tipos contém a declaração de todos os tipos utilizados no serviço. Essa declaração é feita utilizando-se XML *Schema* [24]. A seção de descrição abstrata define a interface do serviço sem nenhuma referência às tecnologias utilizadas. Por fim, a seção de descrição concreta realiza a ligação entre a interface do serviço e sua implementação, descrevendo assim as tecnologias utilizadas.

SOAP é um protocolo utilizado para trocar informações estruturadas em um meio descentralizado e distribuído [22]. As suas mensagens têm o formato baseado em XML e ele se baseia em outros protocolos da camada de aplicação, tais como o RPC [25] (do inglês *Remote Procedure Call Protocol*) e o HTTP. As mensagens SOAP são empacotadas em uma estrutura chamada *envelope*, que se subdivide em *header* e *body*. Sendo estas responsáveis por encapsular as informações de cabeçalho e de corpo da mensagem, respectivamente.

UDDI oferece aos *Web Services* um recurso que permite a eles serem registrados em um local central, de modo que possam ser descoberto por terceiros facilmente. Esta especificação permite a criação de registros de descrição de serviços padronizados, tanto dentro como fora das fronteiras de uma organização. Diferentemente do WSDL e do SOAP, UDDI não teve grande aceitação na indústria e continua a ser uma extensão opcional para SOA [17].

Ao se utilizar dos padrões apresentados, os *Web Services* conseguem aderir a todos os princípios de orientação a serviço facilmente. Todavia, os *Web Services* e seus padrões são bastante flexíveis, portanto, dependendo da forma que eles forem desenvolvidos, poderão não atender mais a alguns princípios de orientação a serviço.

2.2.3 Princípios de orientação a serviços

O paradigma de projeto orientado a serviços possui oito princípios bem definidos, como pode ser visto na **Figura 4**. Cada um desses princípios aborda aspectos fundamentais da concepção de um serviço, portanto, ao aplicar tais princípios, se garante que uma solução esteja de acordo com o paradigma orientado a serviços [16].

O princípio do *Contrato Padronizado* está relacionado ao desenvolvimento de um serviço a partir de um contrato, tal que este esteja no padrão dos contratos dos demais serviços que se encontram em seu repositório. O princípio do *Baixo Acoplamento* se refere à minimização de dependência entre o serviço e os seus consumidores e o serviço e o ambiente em que o mesmo se encontra. Logo em seguida temos o princípio da *Reusabilidade*, este é responsável por garantir que um serviço possa ser tratado como um recurso reutilizável. O princípio da *Abstração*

minimiza a necessidade de metadados, pois ele determina que os contratos dos serviços devam possuir somente as informações essenciais do serviço. O princípio da *Autonomia* está associado ao fato do serviço ter alto nível de controle sobre seu ambiente em tempo de execução. Ainda temos o princípio da *Composição*, responsável por maximizar o nível de composição de serviços, ou seja, os serviços podem participar de uma composição de serviços independente do tamanho e da complexidade da mesma. O princípio do *Estado Não Gerenciado* se refere à ausência da necessidade de gerenciar o estado dos recursos utilizados pelo serviço. Por último temos o princípio da *Descoberta*, este permite aos serviços serem descobertos e interpretados de forma efetiva através da adição de metadados de comunicação nos mesmos.

Figura 4. Princípios de orientação a serviço.

Capítulo 3

IARS - Inter-Applications Recommendation Service

Este capítulo discorre sobre o serviço de recomendação inter-aplicações desenvolvido neste trabalho, utilizando-se as teorias apresentadas no Capítulo 2. Inicialmente, na Seção 3.1, é dada uma visão geral de sistemas de recomendação inter-aplicações e do IARS, um serviço de recomendação criado a partir dos estudos realizados nesse trabalho. Logo em seguida, a Seção 3.2 apresenta uma visão detalhada da arquitetura do sistema e descreve como funciona cada módulo dele. Por fim, a Seção 3.3 apresenta como a base de conhecimento do IARS foi estruturada.

3.1 Visão geral

Os sistemas de recomendação atuais são capazes de recomendar itens a usuários se baseando em seus perfis. Por sua vez, tais sistemas possuem uma limitação, só se baseiam no perfil do usuário obtido no domínio da aplicação que está se utilizando deles. Ou seja, o perfil deste usuário em outras aplicações não é utilizado ao recomendar algum item. Isso ocorre por que os sistemas de recomendação são fortemente ligados ao domínio da aplicação. Algumas aplicações, como dito anteriormente, são bastante afetadas por esta limitação. Todavia, um sistema de recomendação teria tal limitação exaurida caso ele fosse capaz de realizar recomendações utilizando perfis de usuários obtidos em diversas aplicações.

A partir dos estudos realizados neste trabalho em torno de sistemas de recomendação, verificou-se a inexistência de trabalhos envolvendo sistemas de recomendação capazes de recomendar itens a partir de perfis de usuários obtidos em diversas aplicações. Sendo assim, a partir deste trabalho, denominamos os sistemas de recomendação que possuem esta capacidade de sistemas de recomendação inter-aplicações e definimos um modelo genérico para este tipo de

sistema que pode ser visto na **Figura 5**. Neste modelo, podemos observar que um sistema de recomendação inter-aplicações é composto basicamente pelo Engenho de Personalização, o Repositório de Conhecimento, um Módulo de Normalização e uma Interface de Comunicação. O Engenho de Personalização é o componente principal do sistema, é nele que se encontra a implementação das técnicas de personalização utilizadas. O Repositório de Conhecimento mantém os dados necessários para gerar as recomendações, ou seja, os perfis de usuários e alguns outros dados auxiliares, tais como os dados de mapeamento. O Módulo de Normalização é responsável por normalizar as avaliações e recomendações quando necessário. Por fim, a Interface de Comunicação é o componente que expõe as operações do sistema de recomendação que serão utilizadas pelos demais sistemas.

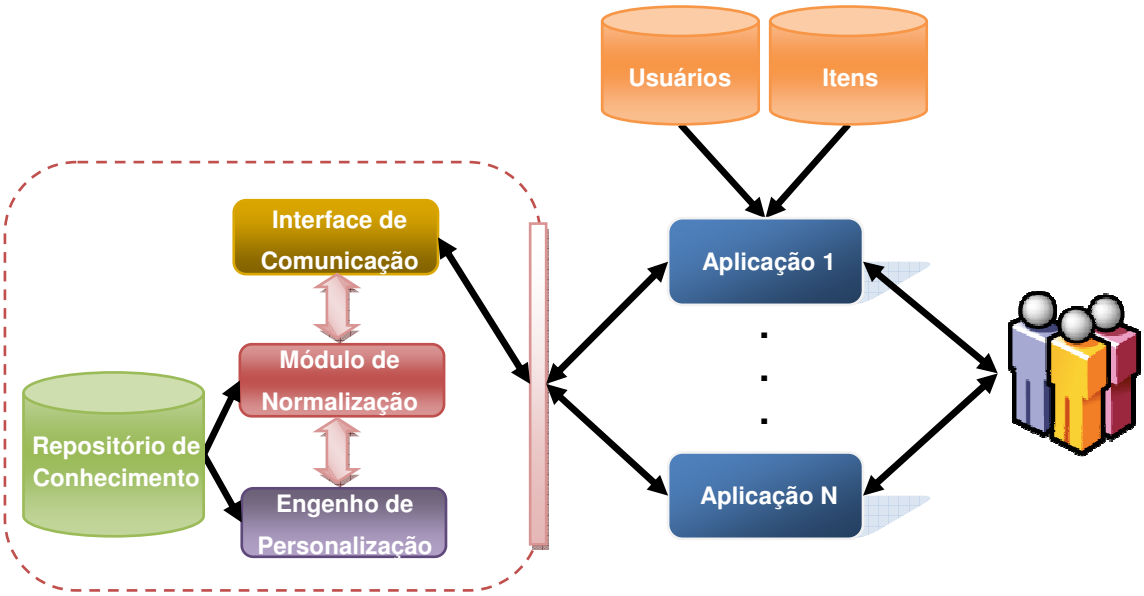


Figura 5. Modelo de sistema de recomendação inter-aplicações.

Os principais desafios encontrados no desenvolvimento de um sistema de recomendação inter-aplicações estão relacionados à forma de representar o perfil do usuário, de recomendar itens de uma determinada aplicação para um usuário, levando em consideração os perfis obtidos em todas as aplicações que se utilizam do sistema, e de estruturar o sistema de modo a diminuir o acoplamento dele com a aplicação, além de se manter interoperável. A complexidade do desenvolvimento

desse tipo de sistema está fortemente relacionada à técnica de personalização utilizada e à arquitetura adotada.

O perfil do usuário é uma estrutura que dependendo de como for modelada torna-se fortemente acoplada ao domínio da aplicação. Ele pode ser representado de diferentes formas em cada aplicação. Em uma aplicação, por exemplo, o perfil do usuário pode representar as avaliações de um usuário para itens dadas através de notas entre 0 e 10, em uma outra aplicação, as avaliações podem ser dadas através de algum indicador, ou seja, bom,ruim, gosta, não gosta e etc.

Em um sistema de recomendação inter-aplicações, sugerir um item de uma aplicação para um usuário, baseando-se nos perfis dele obtidos em todas as aplicações que se utilizam do sistema, não é algo tão trivial, devido às técnicas de personalização não terem sido criadas com esse intuito. Portanto, ao utilizar alguma das técnicas de personalização já criadas, torna-se necessária a realização de uma adaptação de modo a conceber tal funcionalidade.

Os sistemas de recomendação são comumente desenvolvidos como subsistemas de uma aplicação. Porém, um sistema de recomendação inter-aplicações precisa ir mais além devido ao fato de ser utilizado por mais de uma aplicação, que na sua maioria, são desenvolvidas sob plataformas distintas. Além disso, precisam manter um baixo nível de acoplamento com a aplicação.

Os estudos realizados nesse trabalho resultaram na criação de um sistema de recomendação inter-aplicações denominado IARS[1] (do inglês *Inter-Applications Recommendation Service*). Este sistema se utiliza de uma técnica de filtragem colaborativa baseada em kNN e no coeficiente de correlação de Pearson, ou seja, umas das técnicas mais utilizadas atualmente no mercado. Entretanto, tal técnica foi adaptada com a finalidade de sugerir um item de uma aplicação para um usuário a partir do seu perfil obtido em todas as aplicações que utilizarem o sistema. Além disso, o IARS é definido como um serviço, mais especificamente um *Web Service*, portanto, ele se utiliza dos princípios de orientação a serviço para alcançar um maior grau de reusabilidade e interoperabilidade.

[1] O código fonte do projeto está disponível em http://code.google.com/p/iars

O IARS foi desenvolvido sob a plataforma *Java*, mais especificamente a *Java Enterprise Edition*[2], da qual foram utilizados diversos recursos, tais como gerenciamento de transações, de threads, de pool de conexão, entre outros. Tal plataforma foi escolhida devido a Java ser uma linguagem orientada a objetos bem difundida, tanto no meio acadêmico quanto no mercado, e por esta plataforma conceder uma maior robustez a aplicação através do seu implícito gerenciamento de recursos. Como servidor de aplicação e de banco de dados foram utilizados o *JBoss Application Server*[3] e o *MySQL Server*[4], respectivamente, devido aos mesmos serem bastante utilizados no mercado e darem suporte aos recursos utilizados no IARS. Por fim, foi utilizado um *framework* chamado de *Mahout*[5] que contém, além de outros recursos, uma biblioteca de engenho de personalização, que foi incorporada ao *framework* e era previamente conhecida como *Taste*[6].

Nas próximas seções, será apresentado como IARS foi projetado e desenvolvido, além de como são realizadas as recomendações inter-aplicações.

3.2 Arquitetura do sistema

O IARS é composto basicamente por três módulos: o Módulo de Serviço, o Módulo de Contextualização e o Módulo de Recomendação. Ao mapear o IARS no modelo genérico de sistema de recomendação inter-aplicações apresentado na **Figura 5**, temos que o Módulo de Serviço assumiria o papel de Interface de Comunicação, o Módulo de Contextualização seria o Módulo de Normalização e o Módulo de Recomendação incorporaria o papel do Engenho de Personalização, tal mapeamento é apresentado na **Figura 6**. Ou seja, o Módulo de Serviço define a *interface* de comunicação entre as aplicações e o IARS, o Módulo de Contextualização realiza a normalização dos dados e o Módulo de Recomendação é

[2] http://www.oracle.com/technetwork/java/javaee/overview/index.html

[3] http://www.jboss.org/jbossas/

[4] http://www.mysql.com/

[5] http://mahout.apache.org/

[6] http://taste.sourceforge.net/

o responsável por recomendar itens aos usuários no contexto de uma dada aplicação.

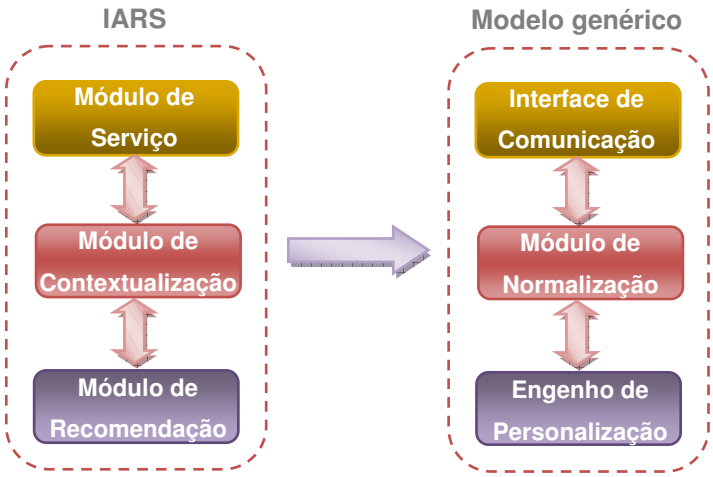


Figura 6. Mapeamento do IARS no modelo genérico de sistema de recomendação inter-aplicações.

Nas seções seguintes, os módulos do IARS serão apresentados em mais detalhes, ou seja, serão apresentadas quais são suas responsabilidades e como eles as realizam.

3.2.1 Módulo de Serviço

O Módulo de Serviço serve basicamente de ponte para as principais funcionalidades do IARS, ou seja, ele é o responsável por expor o contrato das suas operações na forma de serviço, definir os tipos utilizados no serviço e realizar as conversões necessárias entre os tipos definidos e os tipos utilizados internamente nos Módulos de Contextualização e de Recomendação.

O serviço definido neste módulo é implementado através de *Web Services*, desta forma o IARS atinge um alto nível de interoperabilidade e um baixo nível de acoplamento com as aplicações. Portanto, ele pode ser utilizado por diversas aplicações, independente da plataforma em que elas tenham sido desenvolvidas, além de manter um baixo nível de dependência com elas.

Este módulo é composto pelo o descritor de serviço, a implementação do serviço especificado no descritor e os tipos definidos por ele. O descritor de serviço especifica as cinco operações principais do IARS. Como visto na **Figura 7**, as operações especificadas são: *addContext*, *setPreference*, *setPreferences*,

recommend e *estimatePreference*. Estas operações são detalhadas logo em seguida.

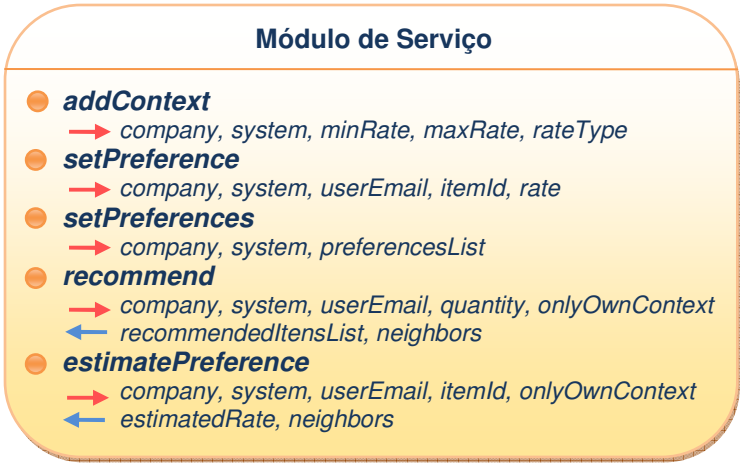


Figura 7. Contrato do Módulo de Serviço do IARS.

A operação *addContext* é utilizada pelas aplicações para fornecer alguns dados que as identifiquem unicamente, além de algumas informações referentes ao processo de avaliação pertencente a aplicação, tais como a representação da nota, ou seja, um número inteiro ou decimal, e a menor e a maior nota que podem ser dadas pelos usuários. Sendo assim, qualquer aplicação, que deseje utilizar o IARS, deverá invocar tal operação pelo menos uma vez antes de invocar as demais, pois para invocá-las será necessário que a aplicação se identifique.

A operação *setPreference* é utilizada pelas aplicações para fornecer os dados da avaliação de um item realizada por um determinado usuário em uma aplicação. Portanto, esta operação faz uso dos dados que identificam unicamente o item na aplicação, ou seja, a chave primária do item, a nota dada pelo usuário ao item e o identificador do usuário que realizou a avaliação. O usuário no IARS é identificado unicamente pelo seu e-mail, portanto, as interações deste usuário com as diversas aplicações terão seu uso melhorado caso ele (o usuário) use o mesmo e-mail nessas aplicações, caso contrário, para o IARS ele será tratado como outro usuário.

A operação *setPreferences* é utilizada pelas aplicações para informar os dados de um conjunto de avaliações realizadas em uma aplicação por diversos usuários. Esta operação se utiliza dos mesmos dados utilizados na operação *setPreference*, a única diferença é que em *setPreference* se trabalha com

informações de uma única avaliação, enquanto que em *setPreferences* tem-se uma lista de avaliações.

A operação *recommend* retorna uma lista de itens recomendados para um dado usuário em uma aplicação, além dos usuários que são identificados como seus vizinhos, ou seja, aqueles usuários que são mais semelhantes a ele. A lista de itens é composta basicamente dos identificadores dos itens recomendados, junto as suas respectivas avaliações calculadas pelo sistema. A quantidade de itens a serem retornados deve ser indicada ao invocar tal operação, além do identificador do usuário. Como pode ser observada, esta operação reflete a funcionalidade de Gerar Lista Personalizada dos Engenhos de Personalização.

Por último temos a operação *estimatePreference*, esta prediz a nota do usuário para um dado item de uma aplicação. Ou seja, expõe a funcionalidade de Predizer Nota dos Engenhos de Personalização. Com isso, ele retorna a nota estimada pelo sistema, além dos vizinhos do usuário. Ao invocar tal operação, a aplicação deve fornecer o identificador do item para o qual se deseja predizer a nota e o identificador do usuário.

Para as operações *recommend* e *estimatePreference* ainda é permitido que uma determinada aplicação especifique se deseja que a recomendação, gerada pelo Módulo de Recomendação, se baseie no perfil do usuário obtido somente na própria aplicação ou em todas as aplicações que se utilizam do IARS.

No IARS, os dados utilizados para identificar unicamente a aplicação foram o nome da empresa responsável pela aplicação e o nome dado a aplicação, ou qualquer outro identificador dado à aplicação pela empresa. Estes dados são utilizados ao invocar qualquer das operações do IARS. Todavia, existe a possibilidade do nome da empresa, por exemplo, ser obtido a partir de um Certificado Digital. Tal modificação traria alguns benefícios, tais como a possibilidade de se usar Assinatura Digital, e poderia ser realizada utilizando o WS-*Security* [26]. A partir desta especificação é possível se definir diversas políticas de segurança desde a utilização de Criptografia até a de Certificado e Assinatura Digitais. Logo, este recurso poderá ser implementado em trabalhos futuros a fim de tornar o IARS mais robusto e seguro.

3.2.2 Módulo de Contextualização

O Módulo de Contextualização é o responsável por realizar as normalizações e mapeamentos necessários dos dados. Tais procedimentos se fazem necessários devido às diferentes formas de avaliação de itens utilizadas pelas aplicações e a possibilidade de duplicação de chaves primárias para itens de diferentes aplicações. Além disso, existe a necessidade de mapear tanto os usuários, quanto os contextos em identificadores únicos dentro do IARS. Portanto, este módulo fornece mecanismos que possibilitam ao Módulo de Recomendação realizar recomendações inter-aplicações.

As operações expostas pelo Módulo de Contextualização basicamente são as mesmas que são expostas pelo Módulo de Serviço, diferenciando-se somente por alguns tipos utilizados. No entanto, estas operações são invocadas pelo Módulo de Serviço, logo após eles terem realizado as conversões de tipos necessárias.

Ao ter suas operações invocadas, o Módulo de Contextualização pode executar até três tarefas, como visto na **Figura 8**: Descontextualização, Invocação do Módulo de Recomendação e Contextualização.

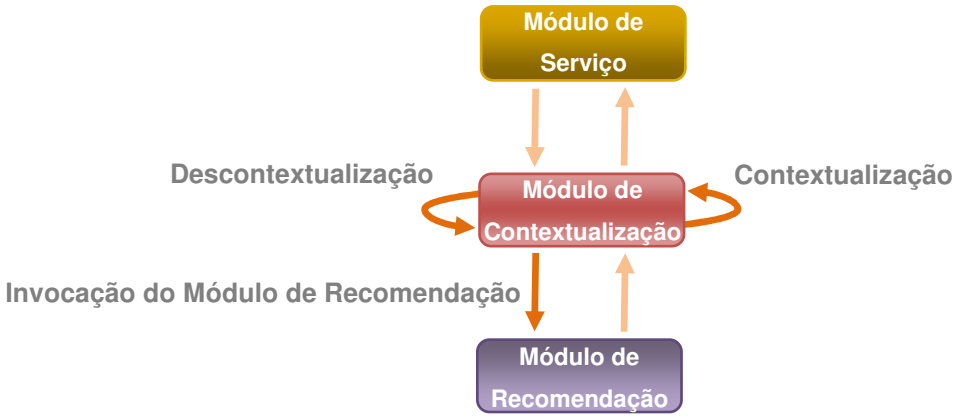


Figura 8. Tarefas realizadas pelo Módulo de Contextualização.

A Descontextualização é a tarefa responsável por realizar o mapeamento dos dados provenientes das aplicações para representações internas utilizadas pelo Módulo de Recomendação. Além disso, este tarefa também realiza a normalização das notas dadas por usuários aos itens em uma determinada aplicação para a representação de notas utilizada pelo Módulo de Recomendação.

A Invocação do Módulo de Recomendação é a tarefa em que as operações do Módulo de Recomendação são invocadas a fim de obter uma lista de itens recomendados ou predizer a nota de item a um dado usuário em uma aplicação.

Por fim, a tarefa de Contextualização é a responsável por realizar o mapeamento dos dados provenientes do Módulo de Recomendação para a aplicação que invocou o serviço. Além disso, este tarefa também realiza a normalização das notas dadas pelo Módulo de Recomendação a um ou mais itens para a representação utilizada pela aplicação que invocou o serviço.

A execução dessas tarefas depende de qual operação foi invocada, pois nem todas as operações expostas pelo Módulo de Contextualização necessitam da execução das três tarefas. Entretanto, vale à pena ressaltar que algumas dessas tarefas realizam consultas, inserções ou atualizações no Repositório de Conhecimento, ou seja, a partir do Módulo de Contextualização, o Repositório de Conhecimento passa a ser utilizado. A seção 3.3 deste capítulo apresentará o Repositório de Conhecimento em mais detalhes.

A operação *addContext* executa basicamente a tarefa de Descontextualização, pois ela só necessita que os dados do contexto sejam mapeados para um identificador a ser utilizado internamente pelo sistema. Caso o contexto a ser mapeado já possua identificador, os seus dados serão atualizados no Repositório de Conhecimento e seu identificador será mantido.

Nas operações *setPreference* e *setPreferences*, são executadas as tarefas de Descontextualização e Invocação do Módulo de Recomendação. Ou seja, primeiramente os dados do contexto, do usuário e do item são mapeados para identificadores no Repositório de Conhecimento. Vale ressaltar que para o usuário e o item, caso estes ainda não possuam identificadores, novos serão gerados, entretanto, em relação ao contexto, este já deve ter o seu identificador gerado, restando só carregá-lo. Além disso, as notas dos itens serão normalizadas para a representação de notas utilizada pelo Módulo de Recomendação a partir da **Equação 3.1**, onde temos que $\boldsymbol{\eta'(\rho)}$ é a nota normalizada, $\boldsymbol{\rho}$ é a nota não normalizada, $\boldsymbol{max(\rho)}$ e $\boldsymbol{max'(\rho)}$ são as notas máximas e $\boldsymbol{min(\rho)}$ e $\boldsymbol{min'(\rho)}$ são as notas mínimas que podem ser dadas na aplicação e no módulo de recomendação, respectivamente. Após ter sido realizada a Descontextualização, a Invocação do

Módulo de Recomendação é realizada, entretanto os parâmetros a serem passados nessa invocação serão os identificadores carregados anteriormente na Descontextualização mais a nota normalizada.

$$\eta'(\rho) = \frac{\big(\rho - min(\rho)\big)}{\big(max(\rho) - min(\rho)\big)} * \big(max'(\rho) - min'(\rho)\big) + min'(\rho) \tag{3.1}$$

Por fim, nas operações *recommend* e *estimatePreference*, as três tarefas apresentadas anteriormente são executadas. Ou seja, inicialmente os dados do contexto, do usuário e do item são mapeados para identificadores no Repositório de Conhecimento, onde o contexto já deve ter seu identificador gerado, só restando carregá-lo, e caso o usuário ou o item ainda não tenha um identificador, um será gerado para cada um deles. Logo em seguida, o Módulo de Recomendação será invocado passando como parâmetros os mesmos recebidos pelo Módulo de Contextualização, substituindo somente os dados do usuário, do item e do contexto pelos seus respectivos identificadores. Por último será executada a tarefa de Contextualização e logo em seguida retornado os dados necessários já contextualizados. Nesta tarefa, os dados de um ou mais itens, pertencentes à aplicação que invocou o serviço, serão carregados novamente a partir dos seus identificadores utilizados internamento pelo sistema que foram retornados pelo Módulo de Recomendação ao realizar as recomendações. Além disso, as notas geradas para esses itens também serão normalizadas, a partir da **Equação 3.2**, para a representação utilizada pela aplicação que invocou o serviço. Neste caso, $\boldsymbol{\eta(\rho')}$ é a nota normalizada para representação da aplicação, $\boldsymbol{\rho'}$ é a nota normalizada para a representação do Módulo de Recomendação.

$$\eta(\rho') = \frac{\big(\rho' - min'(\rho')\big)}{\big(max'(\rho') - min'(\rho')\big)} * \big(max(\rho') - min(\rho')\big) + min(\rho') \tag{3.2}$$

3.2.3 Módulo de Recomendação

O Módulo de Recomendação é o responsável por realizar as recomendações de itens aos usuários para aplicações que utilizam o IARS. Ou seja, este módulo assume o papel de Engenho de Personalização dentro do IARS. Podemos considerar esse módulo como o núcleo do sistema. Ele conterá as adaptações

necessárias em uma técnica de filtragem colaborativa baseada em kNN e no coeficiente de correlação de Pearson para realizar recomendações inter-aplicações.

Diferentemente do Módulo de Contextualização, que possuía as mesmas operações do Módulo de Serviço, distinguindo-se somente por alguns tipos utilizados, o Módulo de Recomendação só possui três operações, *setPreference*, *recommend* e *estimatePreference*, como pode ser visto no contrato apresentado na **Figura 9**. Estas operações são utilizadas basicamente para gravar uma nova preferência de um usuário para um item, gerar uma lista de itens recomendados a um usuário e predizer a nota de um item para um usuário, respectivamente. Entretanto, tais operações serão apresentadas em mais detalhes logo abaixo.

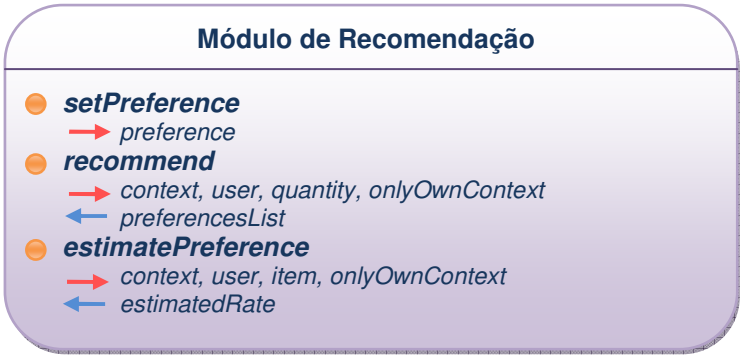


Figura 9. Contrato do Módulo de Recomendação do IARS.

A operação *setPreference* é responsável por persistir as informações relevantes da preferência de um usuário por item em uma determinada aplicação. A preferência é composta pelos identificadores para os quais o usuário e o item foram mapeados, além da nota dada pelo usuário ao item e que foi previamente normalizada

A operação *recommend* é responsável por gerar uma lista de itens recomendados para um usuário a partir das suas preferências obtidas em diferentes aplicações ou aquelas obtidas somente na aplicação que invocou o serviço. Para realizar isto, a operação precisa saber qual aplicação está invocando o serviço, ou seja, a qual aplicação pertencerão os itens da lista gerada, o usuário para o qual os itens serão recomendados, a quantidade de itens a serem listados e se a recomendação deve ser gerada a partir das preferências obtidas somente na aplicação que invocou o serviço.

Por último, a operação *estimatePreference* prediz a nota de um item para um determinado usuário em uma aplicação. A predição é feita baseando-se nas preferências do usuário obtidas em diferentes aplicações ou apenas obtidas na aplicação que invocou o serviço. No entanto, para realizar esta operação, são necessários os dados de qual aplicação está invocando o serviço, o usuário e o item para o qual se deseja predizer a nota, além do indicador que diz se a predição da nota deve ser realizada a partir das preferências obtidas somente na aplicação que invocou o serviço.

O desenvolvimento do Módulo de Recomendação se deu através da utilização do *framework Mahout*. Este *framework* disponibiliza implementações de técnicas de personalização baseada em filtragem colaborativa. No Módulo de Recomendação, a implementação fornecida pelo *framework* da técnica de filtragem colaborativa, sendo esta baseada em kNN e no coeficiente de correlação de Pearson, foi adaptada a fim de utilizar o Repositório de Conhecimento projetado para o IARS e realizar recomendações inter-aplicações.

A adaptação da técnica de filtragem colaborativa baseada em kNN e no coeficiente de correlação de Pearson fornecida pelo *framework* foi realizada em dois momentos: no cálculo do coeficiente de correlação de Pearson e no cálculo da utilidade de um item para um dado usuário, definidos pelas **Equação 2.3** e **Equação 2.4**, respectivamente.

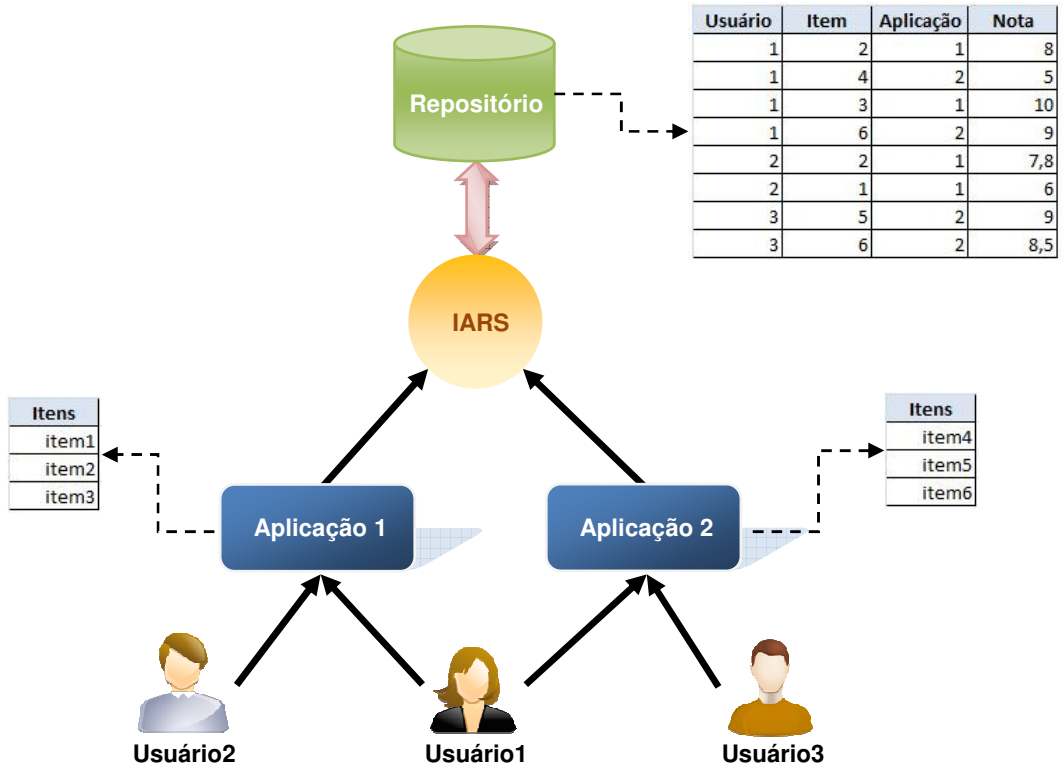
Para o cálculo do coeficiente de correlação de Pearson, duas restrições foram criadas, uma relacionada ao conjunto de itens $\boldsymbol{I}$ e a outra relacionada ao usuário $\boldsymbol{u}$. A primeira restrição, ou seja, a que está relacionada ao conjunto de itens $\boldsymbol{I}$, diz que ao calcular o coeficiente de correlação de Pearson, caso a aplicação que invocou o IARS não tenha especificado que a recomendação ou a predição da nota deva ser realizada a partir das preferências obtidas somente na aplicação que invocou o serviço, o conjunto de itens $\boldsymbol{I}$ será representado por todos os itens que estejam cadastrados no Repositório de Conhecimento, independente a qual aplicação eles pertençam. Caso contrário, somente os itens pertencentes à aplicação que invocou o serviço serão considerados. Portanto, isto permite que todas as preferências dos usuários sejam levadas em consideração ao recomendar itens ou predizer uma nota em uma aplicação. A segunda e última restrição criada para este cálculo está

relacionada ao usuário candidato a vizinho do usuário v, para o qual se deve realizar recomendações. O candidato a vizinho neste caso deve possuir ao menos uma preferência para um item da aplicação que invocou o serviço. Desta forma, somente usuários que podem ser utilizados para recomendar algum item dentro de uma determinada aplicação serão candidatos a vizinhos. Caso contrário, usuários que não pertencessem à aplicação poderiam ser identificados como vizinhos, no entanto, eles não teriam itens, dentre suas preferências, passíveis de serem recomendados na aplicação que invocou o serviço.

Para o cálculo da utilidade de um item para um dado usuário, três considerações foram realizadas. A primeira delas está associada ao usuário v, este não necessariamente precisa possuir alguma preferência para um item da aplicação que invocou o serviço, ou seja, isso pode minimizar ou eliminar o problema do novo usuário, entretanto ele deve possuir ao menos uma preferência para algum item de outra aplicação que utilize o IARS. A segunda consideração realizada está relacionada ao item i, este necessariamente deve pertencer à aplicação que invocou o serviço. Por fim, temos a consideração associada aos vizinhos u_k do usuário v, estes devem possuir ao menos uma preferência para um item da aplicação que invocou o serviço.

A fim de exemplificar como as adaptações apresentadas atuariam em tempo de execução, consideremos o cenário apresentado na **Figura 10**. Nele, há duas aplicações que se utilizam do IARS para recomendar itens aos seus usuários, são elas: *Aplicação1* e *Aplicação2*. Cada uma dessas aplicações possui três itens cadastrados. Neste cenário, ainda podem ser vistos três usuários, o *Usuário1*, o *Usuário2* e o *Usuário3*. O *Usuário1* possui avaliações de itens tanto na *Aplicação1*, quanto na *Aplicação2*. O *Usuário2* possui avaliações de itens somente na Aplicação1 e o *Usuário3* somente na *Aplicação2*. A partir deste cenário, supondo que o *Usuário2* viesse a acessar a *Aplicação1* e essa requisitasse a geração de uma lista de itens recomendados ao IARS, o cálculo do coeficiente de correlação de Pearson consideraria todas as preferências dos candidatos a vizinho para qualquer item pertencentes a *Aplicação1* e a *Aplicação2*. Além disso, o *Usuário1* seria o único usuário a ser considerado ao calcular o coeficiente de correlação de Pearson, devido ao *Usuário3* não possuir nenhuma avaliação de item na *Aplicação1*. No cálculo da

utilidade de um item para o *Usuário2*, somente os itens da *Aplicação1* seriam considerados e somente aqueles usuários que possuíssem ao menos uma avaliação de item na *Aplicação1* seriam candidatos a vizinho do *Usuário2*, ou seja, somente o *Usuário1*.



Usuário	Item	Aplicação	Nota
1	2	1	8
1	4	2	5
1	3	1	10
1	6	2	9
2	2	1	7,8
2	1	1	6
3	5	2	9
3	6	2	8,5

Figura 10. Cenário de utilização do IARS.

A partir das adaptações apresentadas tornou-se possível realizar recomendações inter-aplicações, pois a técnica de filtragem colaborativa utilizada passou a considerar todas as preferências, ou avaliações, dos usuários ao calcular a correlação entre eles e o usuário para o qual se recomendaria algum item. Além disso, tais adaptações permitiram que alguns problemas identificados em técnicas de filtragem colaborativas fossem amenizados, tais como o do novo usuário, devido ao uso do perfil dele obtido em outras aplicações, e da baixa qualidade de recomendação para usuários conhecidos como ovelhas negras, pois mesmo se um usuário não possuir preferências tão comuns em uma determinada aplicação, o perfil dele em outras aplicações será utilizado ao recomendar itens, aumentando assim a probabilidade de se encontrar vizinhos mais semelhantes a ele. No entanto, outros problemas, como o do novo item e da esparsidade, não sofreram qualquer alteração.

A próxima seção encerrará apresentando em detalhes como foi estruturado o Repositório de Conhecimento a fim de possibilitar a recomendação inter-aplicações.

3.3 Repositório de Conhecimento

O Repositório de Conhecimento criado para o IARS contém basicamente quatro tabelas fundamentais: *context*, *item*, *preference* e *user*. A estrutura dessas tabelas pode ser vista na **Figura 11**. A estrutura delas foi concebida de modo a dar total suporte as adaptações realizadas na técnica de filtragem colaborativa utilizada pelo serviço.

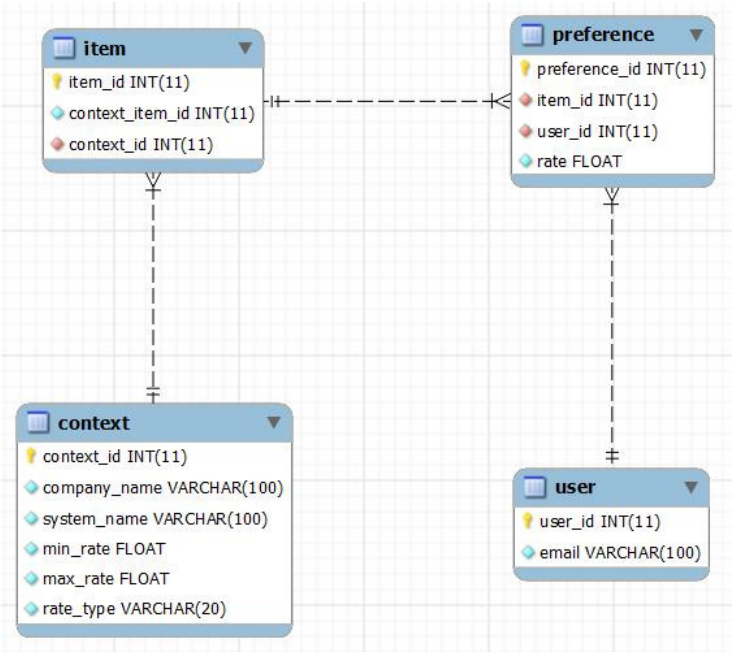


Figura 11. Modelo físico do Repositório de conhecimento.

A tabela *context* é utilizada para manter os dados associados às aplicações que utilizam o IARS. Ela possui os campos *context_id*, *company_name*, *system_name*, *min_rate*, *max_rate* e *rate_type*, eles representam o número seqüencial que identifica a aplicação unicamente dentro do IARS, o nome da empresa responsável pelo sistema, o identificador do sistema, a nota mínima e máxima que podem ser dadas na avaliação de um item por parte de um usuário na aplicação e o tipo de nota, respectivamente. Onde o tipo de nota pode ser decimal ou inteiro. Esta tabela ainda contém uma chave candidata definida pela combinação do nome da empresa com o identificador do sistema. Como vimos anteriormente, estes são os dados utilizados pelas aplicações para se identificar ao utilizar o IARS.

A tabela *item* contém os dados do item que são relevantes ao IARS. Esta tabela é definida pelos campos *item_id*, *context_item_id* e *context_id*, onde tais campos representam, respectivamente, o número seqüencial que identifica o item unicamente dentro do IARS, o identificador do item utilizado na aplicação a qual ele pertence e o seqüencial utilizado pelo IARS para identificar unicamente a aplicação. Esta tabela ainda possui uma chave candidata definida através da combinação do identificador do item utilizado na aplicação com o seqüencial que identifica a aplicação seja única dentro do IARS.

A tabela *user* é responsável por manter os dados relevantes dos usuários. Esta tabela contém os campos *user_id* e *email* que representam, respectivamente, o número seqüencial que identifica o usuário unicamente dentro do IARS e o e-mail do usuário utilizado nas aplicações. Nela, o e-mail do usuário é uma chave candidata.

A tabela de *preference* mantém as preferências, ou avaliações, realizadas pelos usuários para os itens das aplicações. Ela possui os campos *preference_id*, *user_id*, *item_id* e *rate* que definem, respectivamente, o número seqüencial que identifica a preferência unicamente dentro do IARS, os seqüenciais utilizados pelo IARS para identificar unicamente o usuário e o item, além da nota dada ao item pelo usuário. Esta tabela ainda possui uma chave candidata composta basicamente pela combinação dos seqüenciais que identificam o usuário e o item, garantindo desta forma, que uma determinada avaliação de um item realizada por um usuário em um contexto seja única no IARS.

Capítulo 4

Análise do IARS

Este capítulo apresenta uma análise das recomendações obtidas pelo serviço de recomendação inter-aplicações apresentado no Capítulo 3, o IARS. A seção 4.1 apresenta uma ferramenta desenvolvida para realizar a coleta de preferências de usuários sob diferentes domínios e dar suporte à análise das recomendações a ser realizada neste trabalho. A seção 4.2 discorre sobre as características da base de dados coletada e utilizada na análise das recomendações. Por fim, na seção 4.3, são analisadas algumas recomendações obtidas pelo IARS utilizando a base de dados coletada.

4.1 Ferramenta de coleta de dados e suporte à análise de recomendações

A ferramenta desenvolvida para dar suporte à coleta de dados e à análise das recomendações, foi concebida sob a plataforma *Java Web*, utilizando *frameworks* como JSF[7] (do inglês *JavaServer Faces*), *Facelets*[8] e *Primefaces*[9], além de algumas das tecnologias utilizadas no desenvolvimento do IARS.

Esta ferramenta é composta basicamente por três módulos: o módulo de coleta de dados, o módulo de recomendação e o módulo de usuário. O módulo de coleta de dados define uma estrutura que permite novos usuários avaliarem alguns itens de diferentes contextos previamente cadastrados. Por outro lado, o módulo de recomendação fornece uma estrutura que permite analisar as recomendações geradas para um usuário sob os diferentes contextos cadastrados. Por fim, o módulo de usuário fornece uma estrutura que permite visualizar todos os usuários

[7] https://javaserverfaces.dev.java.net/

[8] https://facelets.dev.java.net/

[9] http://www.primefaces.org/

cadastrados no sistema, além de fornecer acesso aos demais módulos. Estes módulos serão apresentados em mais detalhes a seguir.

O módulo de coleta de dados permite ao usuário avaliar 10 itens de cada contexto cadastrado. Os contextos cadastrados neste trabalho foram Filmes, Cantores e Livros. Para o contexto de Filmes foram especificados 1 e 10 como os valores para a nota mínima e a máxima, respectivamente, e como o tipo de nota a ser utilizada foi especificado *decimal*. Para o contexto de Cantores foi especificado o valor *inteiro* como o tipo de nota a ser utilizada, além dos valores 1 e 2 para a nota mínima e a máxima, respectivamente, onde o número 1 representa o "*Eu não gosto"*, enquanto o 2 representa o "*Eu gosto"*. Por último, no contexto de Livros também foi especificado o valor *inteiro* como o tipo de nota a ser utilizada e os valores 1 e 5 para a nota mínima e a máxima, respectivamente. Para esse contexto, o valor 1 mapeia o "*Ruim"*, o 2 o "*Abaixo da média"*, o 3 o "*Na média"*, o 4 o "*Muito bom"* e o 5 mapeia o "*Excelente"*. Para cada um dos contextos apresentados acima, 30 itens foram escolhidos diversificadamente e disponibilizados para avaliação por parte dos usuários. Os itens de cada contexto se encontram em sua respectiva aba, localizada logo abaixo do campo do nome do usuário, como pode ser visto na **Figura 12**.

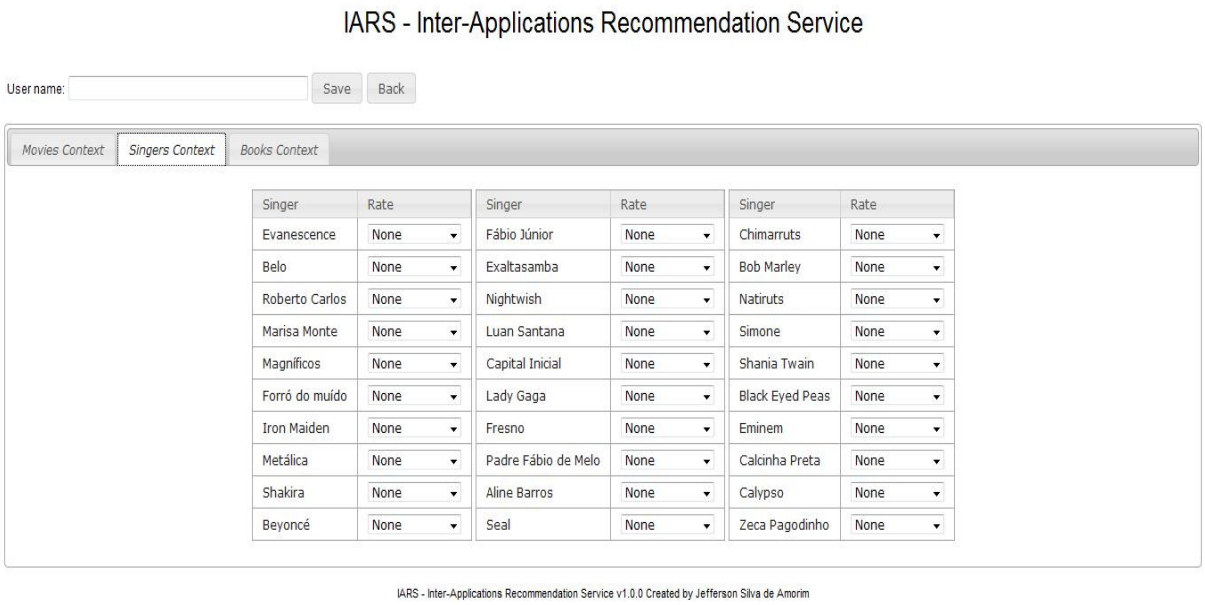


Figura 12. Módulo de coleta de dados.

O módulo de recomendação permite analisar as recomendações geradas para um usuário sob os 3 contextos citados anteriormente. Neste módulo são

apresentadas duas tabelas para cada contexto contendo no máximo 10 itens, onde a tabela mais a esquerda contém os itens recomendados a partir das preferências do usuário no contexto alvo e a tabela mais a direita contém os itens recomendados a partir das preferências do usuário nos 3 contextos cadastrados. Além das recomendações geradas, este módulo ainda fornece informações relacionadas aos vizinhos do usuário alvo sob cada recomendação apresentada. Como pode ser visto na **Figura 13**, estas informações podem ser localizadas na parte superior do módulo, enquanto as recomendações para cada contexto se localizam em diferentes abas na parte inferior do módulo.

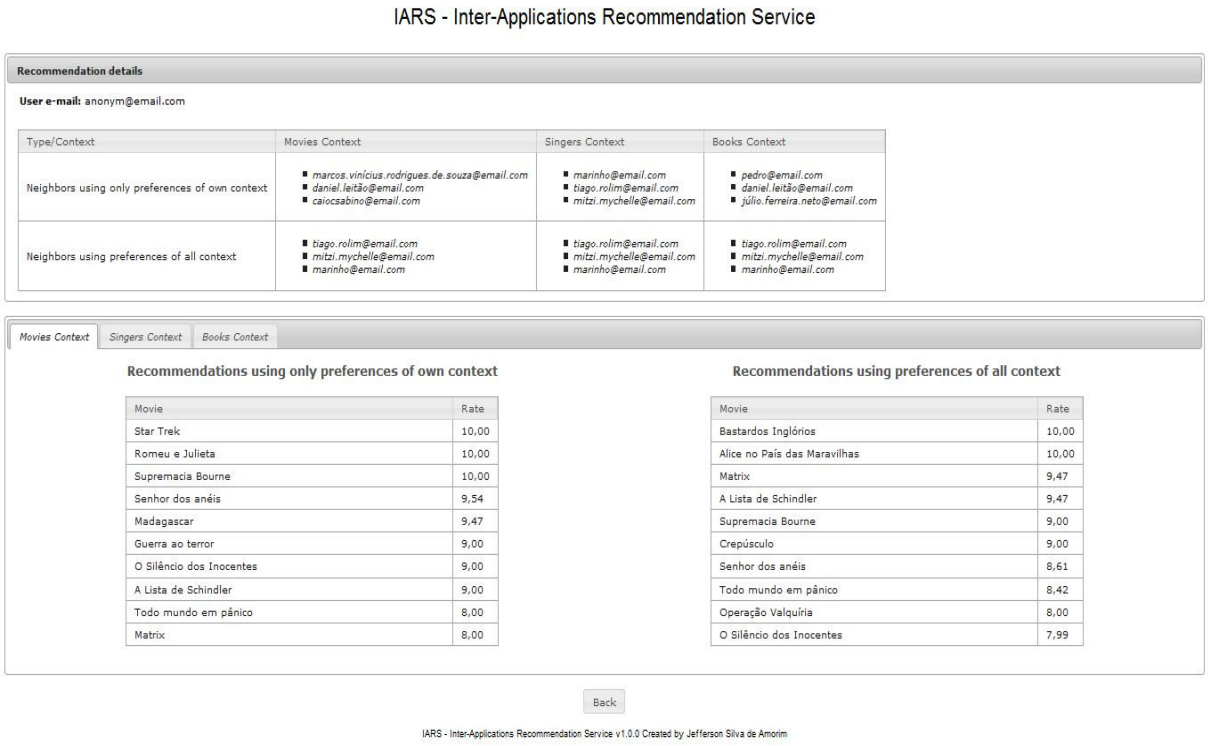


Figura 13. Módulo de recomendação

Por último temos o módulo de usuário, este fornece uma estrutura que permite visualizar os usuários cadastrados no sistema e acessar os demais módulos, ou seja, o módulo de recomendação e de coleta de dados. Como apresentado na **Figura 14**, este módulo apresenta uma lista dos usuários cadastrados, onde ao lado de cada usuário existe um botão que ao ser pressionado carrega o módulo de recomendação e apresenta as recomendações e vizinhos dele. Além disso, neste módulo ainda é possível fornecer novas avaliações de itens realizadas por usuário a partir do link *Insert Preferences*, localizado na parte superior direita do módulo.

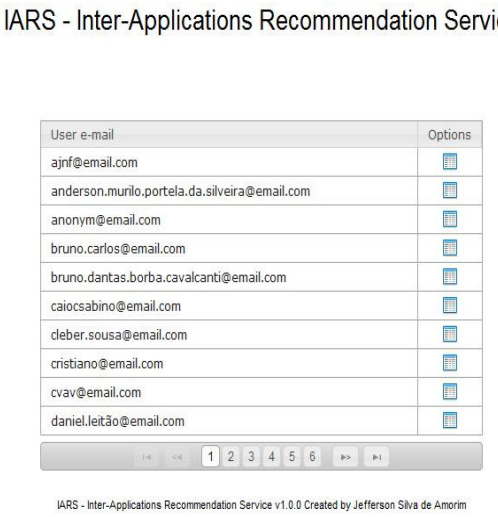


Figura 14. Módulo de usuário.

4.2 Base de dados

A partir da ferramenta apresentada na seção anterior, um processo de coleta de dados foi realizado a fim de conceber uma base de dados que possibilitasse uma análise mais realística do IARS. Essa coleta se fez necessária devido à ausência de bases disponíveis que contenham dados que reflitam as interações dos usuários com diversas aplicações. Entretanto, existem algumas bases disponíveis, tais como *Wikilens*, *MovieLens*, *Book-Crossing* e *Jester Joke*[10], que possuem uma grande quantidade de informações, porém tais informações refletem somente as interações dos usuários dentro de uma única aplicação.

A base de dados obtida após o processo de coleta de dados realizado se constitui de 1560 registros de avaliações realizadas por 52 usuários. Estas avaliações foram realizadas a partir de 3 contextos diferentes, onde para cada contexto há um total de 30 itens cadastrados, totalizando assim 90 itens. Para cada contexto, obteve-se um total de 520 avaliações. Após a coleta desses dados, algumas análises foram realizadas e mostraram que esses dados foram mais do que suficientes para o escopo das análises a que esse trabalho se propôs.

[10] Todas essas bases podem ser obtidas em http://www.grouplens.org.

4.3 Análise das recomendações

A análise realizada na base de dados coletada foi feita sob duas perspectivas, a primeira se deu sob os itens recomendados e a segunda sob os vizinhos identificados ao realizar as recomendações. A análise consistiu em realizar comparações dos resultados obtidos ao gerar recomendações utilizando informações de mais de um contexto, os seja, recomendações inter-contexto ou inter-aplicações, com recomendações geradas utilizando informações de um único contexto, ou seja, a forma como é comumente realizada.

As seções a seguir apresentam alguns cenários relevantes que demonstram que a abordagem proposta neste trabalho trouxe diferentes resultados da abordagem mais utilizada atualmente, ou seja, a abordagem que se baseia nas informações obtidas em um único contexto. Os cenários detalhados nas próximas seções foram identificados durante a análise realizada. Eles apresentam situações ou comportamentos que puderam ser observados em algumas recomendações.

Os cenários foram identificados sob as mesmas perspectivas utilizadas na análise, ou seja, itens recomendados e vizinhos do usuário. Desta forma, as seções a seguir foram definidas como Itens recomendados e Vizinhos do usuário, onde são apresentados, respectivamente, alguns cenários que descrevem situações ou comportamentos relacionados aos itens recomendados e cenários relacionados aos vizinhos do usuário utilizados ao recomendar os itens.

4.3.1 Itens recomendados

Na análise realizada, sob a perspectiva de itens recomendados, foram identificados cinco cenários. O primeiro cenário está relacionado a novos itens presentes na lista de itens recomendados. O segundo a omissão de itens. O terceiro e o quarto, a itens com notas iguais, porém com ordens de utilidade iguais e diferentes, respectivamente, onde a ordem de utilidade de um item é definida pela posição deste na lista. Por fim, o último cenário está relacionado a itens com notas diferentes.

No primeiro cenário, observa-se a presença de novos itens na lista de itens recomendados obtida a partir da abordagem de recomendação inter-aplicações. Um

exemplo deste cenário pode ser facilmente observado na **Figura 15**, onde se encontram os novos filmes *Romeu e Julieta* e *Predador* destacados em verde.

Recommendations using only preferences of own context

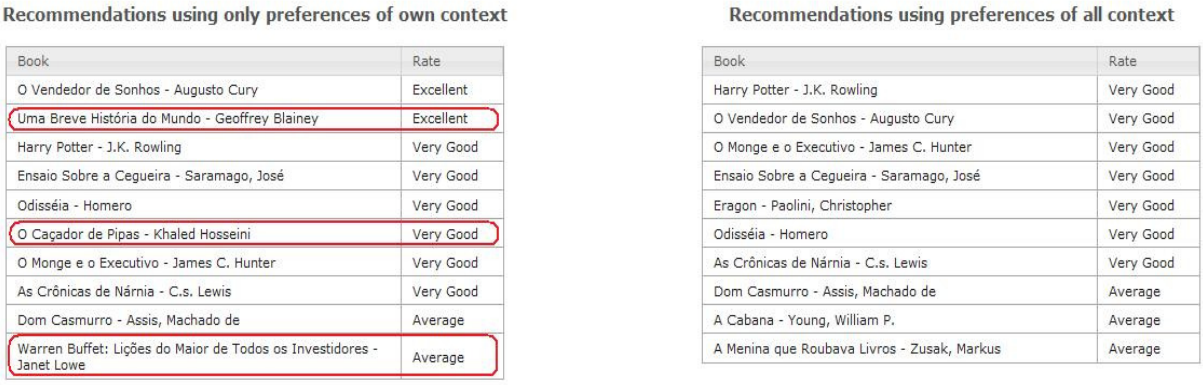
Movie	Rate
Star Wars	10,00
Supremacia Bourne	9,00
O Curioso Caso de Benjamin Button	9,00
Nemo	8,65
Avatar	8,00
Cruzada	8,00
Todo mundo em pânico	7,00
Guerra ao terror	5,00
Crepúsculo	2,53
Dragon Ball Evolution	1,00

Recommendations using preferences of all context

Movie	Rate
Star Wars	10,00
Romeu e Julieta	9,00
Nemo	9,00
Guerra ao terror	9,00
O Curioso Caso de Benjamin Button	9,00
Cruzada	8,49
Predador	8,00
Supremacia Bourne	7,50
Avatar	4,53
Dragon Ball Evolution	4,00

Figura 15. Cenário de novos itens.

O segundo cenário se caracteriza pela ausência de algum item na lista de itens recomendados obtida a partir da abordagem de recomendação inter-aplicações. Um exemplo para este cenário é visto na **Figura 16**. Pode-se observar nela, a ausência dos livros *Uma Breve História do Mundo*, *O Caçador de Pipas* e *Lições do Maior de Todos os Investidores*, estes destacados em vermelho.



Recommendations using only preferences of own context

Book	Rate
O Vendedor de Sonhos - Augusto Cury	Excellent
Uma Breve História do Mundo - Geoffrey Blainey	Excellent
Harry Potter - J.K. Rowling	Very Good
Ensaio Sobre a Cegueira - Saramago, José	Very Good
Odisséia - Homero	Very Good
O Caçador de Pipas - Khaled Hosseini	Very Good
O Monge e o Executivo - James C. Hunter	Very Good
As Crônicas de Nárnia - C.s. Lewis	Very Good
Dom Casmurro - Assis, Machado de	Average
Warren Buffet: Lições do Maior de Todos os Investidores - Janet Lowe	Average

Recommendations using preferences of all context

Book	Rate
Harry Potter - J.K. Rowling	Very Good
O Vendedor de Sonhos - Augusto Cury	Very Good
O Monge e o Executivo - James C. Hunter	Very Good
Ensaio Sobre a Cegueira - Saramago, José	Very Good
Eragon - Paolini, Christopher	Very Good
Odisséia - Homero	Very Good
As Crônicas de Nárnia - C.s. Lewis	Very Good
Dom Casmurro - Assis, Machado de	Average
A Cabana - Young, William P.	Average
A Menina que Roubava Livros - Zusak, Markus	Average

Figura 16. Cenário de omissão de itens.

No terceiro cenário, ao se comparar as duas listas de itens recomendados, ou seja, a lista de itens obtidos a partir da abordagem de recomendação inter-aplicações e a obtida pela abordagem de recomendação baseada em informações de uma única aplicação, itens com notas e ordens de utilidade iguais são observados. Como exemplo para este cenário temos o apresentado na **Figura 17**. Pode-se observar na figura citada que os filmes *Senhor dos anéis* e *Matrix* possuem as mesmas notas e ordens de utilidades.

Recommendations using only preferences of own context

Movie	Rate
Senhor dos anéis	10,00
Matrix	10,00
Duro de matar 4.0	9,00
Nemo	8,69
Madagascar	8,00
À Espera de um Milagre	8,00
Romeu e Julieta	8,00
Cruzada	8,00
Pearl Harbor	8,00
Avatar	7,49

Recommendations using preferences of all context

Movie	Rate
Senhor dos anéis	10,00
Matrix	10,00
Duro de matar 4.0	8,04
Bastardos Inglórios	8,00
Cruzada	8,00
Nemo	7,96
Madagascar	7,65
Guerra ao terror	7,00
Todo mundo em pânico	7,00
Operação Valquíria	7,00

Figura 17. Cenário de itens com notas e ordens de utilidade iguais.

O quarto cenário é semelhante ao terceiro, porém nele alguns itens observados possuem notas iguais e ordens de utilidade diferentes. Na **Figura 18**, pode-se observar um exemplo para este cenário. Observa-se que os livros *A Menina que Roubava Livros*, *Harry Potter*, *O Menino de Pijama Listrada*, *Investimentos Inteligentes* e *O Mundo é Plano: Uma Breve História do Século XXI* possuem as mesmas notas nas duas listas de itens recomendados, porém estes possuem ordens de utilidade diferentes.

Recommendations using only preferences of own context

Book	Rate
A Menina que Roubava Livros - Zusak, Markus	Excellent
O Monge e o Executivo - James C. Hunter	Very Good
O Menino do Pijama Listrado - Boyne, John	Very Good
Carrie , a Estranha - King, Stephen	Very Good
O Símbolo Perdido - Dan Brown	Very Good
Direito Administrativo Descomplicado - Alexandrino, Marcelo; Paulo, Vicente	Very Good
Harry Potter - J.K. Rowling	Very Good
O Mundo É Plano: Uma Breve História do Século XXI - Friedman, Thomas L.	Average
Investimentos Inteligentes - Gustavo Cerbasi	Average
Dom Casmurro - Assis, Machado de	Average

Recommendations using preferences of all context

Book	Rate
Ensaio Sobre a Cegueira - Saramago, José	Excellent
A Menina que Roubava Livros - Zusak, Markus	Excellent
Harry Potter - J.K. Rowling	Very Good
Carrie , a Estranha - King, Stephen	Very Good
O Menino do Pijama Listrado - Boyne, John	Very Good
Direito Administrativo Descomplicado - Alexandrino, Marcelo; Paulo, Vicente	Average
Warren Buffet: Lições do Maior de Todos os Investidores - Janet Lowe	Average
Investimentos Inteligentes - Gustavo Cerbasi	Average
O Mundo É Plano: Uma Breve História do Século XXI - Friedman, Thomas L.	Average
Dom Casmurro - Assis, Machado de	Average

Figura 18. Cenário de itens com notas iguais e ordens de utilidade diferentes.

Por fim, o último cenário apresenta alguns itens iguais, porém com notas diferentes. O exemplo deste cenário pode ser visto na **Figura 19**. Observa-se na figura a presença do cantor *Roberto Carlos* e da banda *Fresno* nas duas listas, no entanto, tais itens se encontram com notas diferentes.

Recommendations using only preferences of own context

Singer	Rate
Natiruts	I like it
Bob Marley	I like it
Evanescence	I like it
Roberto Carlos	I like it
Forró do muído	I don't like it
Lady Gaga	I don't like it
Fresno	I don't like it
Zeca Pagodinho	I don't like it
Calcinha Preta	I don't like it
Shania Twain	I don't like it

Recommendations using preferences of all context

Singer	Rate
Natiruts	I like it
Evanescence	I like it
Bob Marley	I like it
Fresno	I like it
Roberto Carlos	I don't like it
Lady Gaga	I don't like it
Calcinha Preta	I don't like it
Magníficos	I don't like it
Calypso	I don't like it
Forró do muído	I don't like it

Figura 19. Cenário de itens iguais com notas diferentes.

4.3.2 Vizinhos do usuário

Na análise realizada sob a perspectiva de vizinhos de usuário, quatro cenários foram identificados. O primeiro cenário está relacionado à presença dos mesmos vizinhos na mesma ordem de semelhança. O segundo também está relacionado à presença dos mesmos vizinhos, porém em ordem diferente de semelhança. O terceiro, a vizinhos diferentes e o último apresenta uma vizinhança híbrida. Estes cenários são apresentados em mais detalhes logo abaixo.

No primeiro cenário, as duas listas de vizinhos obtidas em um determinado contexto apresentam os mesmos vizinhos na mesma ordem de semelhança, ou seja, cada vizinho se localiza na mesma posição nas duas listas. Pode-se observar na **Figura 20** um exemplo deste cenário, onde as duas listas de vizinhos para o contexto de músicos apresentam os vizinhos *gusstavo.lyra.marques.dos.santos@email.com*, *periclesmiranda@email.com* e *jefferson.amorim@email.com* na mesma ordem de semelhança.

Type/Context	Movies Context	Singers Context	Books Context
Neighbors using only preferences of own context	▪ anonym@email.com ▪ pedro@email.com ▪ cristiano@email.com	▪ gusstavo.lyra.marques.dos.santos@email.com ▪ periclesmiranda@email.com ▪ jefferson.amorim@email.com	▪ jefferson.amorim@email.com ▪ henrique.specht@email.com ▪ pedro.frança@email.com
Neighbors using preferences of all context	▪ gusstavo.lyra.marques.dos.santos@email.com ▪ periclesmiranda@email.com ▪ jefferson.amorim@email.com	▪ gusstavo.lyra.marques.dos.santos@email.com ▪ periclesmiranda@email.com ▪ jefferson.amorim@email.com	▪ gusstavo.lyra.marques.dos.santos@email.com ▪ periclesmiranda@email.com ▪ jefferson.amorim@email.com

Figura 20. Cenário de vizinhos iguais com mesma ordem de semelhança.

O segundo cenário é semelhante ao primeiro cenário, porém nele os vizinhos são apresentados em ordem diferente de semelhança, ou seja, cada vizinho se localiza em posições diferentes das duas listas. Um exemplo deste cenário pode ser

visto na **Figura 21**. Nela os vizinhos *marinho@email.com, tiago.rolim@email.com* e *mitzi.mychelle@email.com* se encontram nas duas listas de vizinhos para o contexto de músicos, no entanto, em ordens de semelhança diferentes.

Type/Context	Movies Context	Singers Context	Books Context
Neighbors using only preferences of own context	▪ marcos.vinícius.rodrigues.de.souza@email.com ▪ daniel.leitão@email.com ▪ caiocsabino@email.com	▪ marinho@email.com ▪ tiago.rolim@email.com ▪ mitzi.mychelle@email.com	▪ pedro@email.com ▪ daniel.leitão@email.com ▪ júlio.ferreira.neto@email.com
Neighbors using preferences of all context	▪ tiago.rolim@email.com ▪ mitzi.mychelle@email.com ▪ marinho@email.com	▪ tiago.rolim@email.com ▪ mitzi.mychelle@email.com ▪ marinho@email.com	▪ tiago.rolim@email.com ▪ mitzi.mychelle@email.com ▪ marinho@email.com

Figura 21. Cenário de vizinhos iguais com ordens de semelhança diferentes.

No terceiro cenário, ao se comparar as duas listas de vizinhos obtidas para um determinado contexto, verifica-se a presença de duas listas completamente distintas, ou seja, listas compostas por vizinhos diferentes. Tal cenário pode ser visto nas listas de vizinhos no contexto de filmes apresentadas na **Figura 22**. Nesta figura, a lista de vizinhos para recomendação baseada em um contexto, ou em uma aplicação, possui os vizinhos *pedro.França@email.com, rafael.albuquerque@email.com e marcos.cardoso@email.com*, enquanto a lista de vizinhos para recomendação inter-aplicações possui *péricles.miranda@email.com, júlio.ferreira.note@email.com* e *teilson.medeiros@email.com*.

Type/Context	Movies Context	Singers Context	Books Context
Neighbors using only preferences of own context	▪ pedro.França@email.com ▪ rafael.albuquerque@email.com ▪ marcos.cardoso@email.com	▪ bruno.dantas.borba.cavalcanti@email.com ▪ júlio.ferreira.neto@email.com ▪ teilson.medeiros@email.com	▪ paulo.leite@email.com ▪ bruno.dantas.borba.cavalcanti@email.com ▪ nadile@email.com
Neighbors using preferences of all context	▪ periclesmiranda@email.com ▪ júlio.ferreira.neto@email.com ▪ teilson.medeiros@email.com	▪ periclesmiranda@email.com ▪ júlio.ferreira.neto@email.com ▪ teilson.medeiros@email.com	▪ periclesmiranda@email.com ▪ júlio.ferreira.neto@email.com ▪ teilson.medeiros@email.com

Figura 22. Cenário de vizinhos diferentes.

O último cenário é caracterizado pela presença de alguns vizinhos diferentes e outros iguais ao se comparar as listas de vizinhos utilizadas na abordagem de recomendação inter-aplicações e a na abordagem de recomendação baseada em informações de uma única aplicação ou contexto. Na **Figura 23**, um exemplo deste cenário é apresentado. Nela, as listas de vizinhos para o contexto de cantores compartilham de alguns vizinhos em comum, *diegoliber@email.com* e *jefferson.pereira@email.com*, e outros diferentes, *pedro@email.com* e *péricles.miranda@email.com*.

Type/Context	Movies Context	Singers Context	Books Context
Neighbors using only preferences of own context	■ hitalo.oliveira@email.com ■ pedro@email.com ■ lemotbr@email.com	■ diegoliber@email.com ■ jefferson.pereira@email.com ■ pedro@email.com	■ marcos.cardoso@email.com ■ marinho@email.com ■ rodrigo.castro@email.com
Neighbors using preferences of all context	■ jefferson.pereira@email.com ■ periclesmiranda@email.com ■ diegoliber@email.com	■ jefferson.pereira@email.com ■ periclesmiranda@email.com ■ diegoliber@email.com	■ jefferson.pereira@email.com ■ periclesmiranda@email.com ■ diegoliber@email.com

Figura 23. Cenário de vizinhança híbrida.

Capítulo 5

Conclusão

Este trabalho teve como finalidade desenvolver um serviço de recomendação inter-aplicações utilizando recursos de SOA e uma técnica de filtragem colaborativa baseada em kNN e no coeficiente de correlação de Pearson. A fim de analisar tal serviço, foram realizadas algumas análises das recomendações realizadas por ele a partir de uma base de dados, que teve seus dados previamente coletados através do preenchimento de um formulário on-line por parte de usuários reais.

Os resultados obtidos a partir da análise realizada se mostraram promissores, pois foi possível identificar que ao utilizar o serviço proposto, conseguiu-se realizar recomendações a um dado usuário a partir do seu perfil proveniente de diversas aplicações. Além disso, vimos que a abordagem proposta neste trabalho ameniza problemas como o do novo usuário e das ovelhas negras em uma determinada aplicação ao se utilizar do perfil do usuário obtido nas demais aplicações. No entanto, problemas como o do novo item e da esparsidade continuam a existir da mesma maneira.

Dentre as dificuldades que foram encontradas ao longo do desenvolvimento desse trabalho, vale a pena ressaltar a relacionada à base de dados a ser utilizada para análise do serviço. As bases de dados disponíveis na internet, que são utilizadas atualmente na realização de experimentos em sistema de recomendação, não puderam ser utilizadas devido ao fato delas possuírem unicamente avaliações de itens realizadas por usuários dentro do contexto de uma aplicação em específico. Portanto, houve a necessidade de se criar um sistema para coletar as preferências de usuários para itens de diferentes aplicações.

Como trabalhos futuros, experimentos baseados em análises estatísticas podem ser realizados a fim de medir a qualidade das recomendações realizadas pelo serviço de recomendação proposto. Outra possibilidade é utilizar técnicas de filtragem baseadas em classificadores simbólicos modais, tais como CMBF (do inglês *Content Modal Based Filtering*), SMBF (do inglês *Social Modal Based Filtering*) ou HMBF (do inglês *Hybrid Modal Based Filtering*), ao invés da técnica de

filtragem colaborativa utilizada neste trabalho, já que estas, atualmente, apresentam melhores resultados [3]. Por fim, ainda poderia ser realizado um estudo relacionado à forma de representação do conteúdo descritivo de itens pertencentes a aplicações e domínios distintos, utilizando-se a Análise de Dados Simbólicos.

Bibliografia

[1] LAUNDON, K. C.; TRAVER, C. G. **E-Commerce - Business, Technology, Society**. Boston: Addison-Wesley Longman Publishing Co. Inc., 2001.

[2] DRUCKER, P. F. **The New Realities**. 1ª st: Transaction Publishers, 2003.

[3] BEZERRA, B. L. D. **Soluções em personalização de conteúdo baseadas em classificadores simbólicos modais**. Universidade Federal de Pernambuco. Recife, p. 211. 2008.

[4] NICHOLS, D. M. **Implicit Rating and Filtering**. In Proceedings of the 5th DELOS Workshop on Filtering and Collaborative Filtering. Budapest: 1997. p. 31-36.

[5] RALPH, P.; PARSONS, J. **A Framework for Automatic Online Personalization**. Proceedings of the 39th Annual Hawaii International Conference on System Sciences. : . 2006. p. 137b - 137b.

[6] ADOMAVICIUS, G.; TUZHILIN, A. Toward the next generation of recommender systems: a survey of the state-of-the-art. **IEEE Transactions on Knowledge and Data Engineering**, v. 17, n. 6, p. 734-749, June 2005.

[7] BURKE, R. Knowledge-Based Recommender Systems. In: KENT, A. **Encyclopedia of Library and Information Systems**. v. 69, 2000.

[8] WEI, K.; HUANG, J.; FU, S. **A Survey of E-Commerce Recommender Systems**. International Conference on Service Systems and Service Management. 2007. p. 1-5.

[9] BIRUKOU, A. et al. **IC-service - A service-oriented approach to the development of recommendation systems**. Proceedings of the 2007 ACM symposium on Applied computing. Seoul: ACM. 2007. p. 1683 - 1688.

[10] HOHPE, G.; WOOLF, B. **Enterprise Integration Patterns - Designing, Building, and Deploying Messaging Solutions**. 1st. ed. Boston: Addison-Wesley Professional, 2003.

[11] AHA, D. W.; KIBLER, D.; ALBERT, M. K. Instance-Based Learning Algorithms. **Machine Learning**, v. 6, p. 37 - 66, June 1991.

[12] SHARDANAND, U.; MAES, P. **Social information filterin - Algorithms for automating "word of mouth"**. Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems. Denver: ACM Press/Addison-Wesley Publishing Co. 1995. p. 210-217.

[13] HERLOCKER, J. L.; KONSTAN, J. A.; TERVEEN, L. G. &. R. J. T. Evaluating collaborative filtering recommender systems. **ACM Transactions on Information Systems**, New York, 22, n. 1, January 2004. 5-53.

[14] HUANG, Z.; ZENG, D.; CHEN, H. A Comparison of Collaborative-Filtering Recommendation Algorithms for E-commerce. **IEEE Intelligent Systems**, Piscataway, v. 22, n. 5, p. 68-78, September 2007.

[15] BELL, R. M.; KOREN, Y. **Improved neighborhood-based collaborative filtering**. KDD Cup and Workshop at the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2007.

[16] ERL, T. **SOA Design Patterns**. 1st. ed. Indiana: Pearson Education, Inc, 2008.

[17] ERL, T. **Service-Oriented Architecture:** Concepts, Technology, and Design. 1st. ed. Indiana: Pearson Education, Inc, 2005.

[18] JOSUTTIS, N. M. **SOA na Prática - A Arte da Modelagem de Sistemas Distribuídos**. 1ª Edição. ed. Rio de Janeiro: Alta Books Ltda., 2008.

[19] FIELDING, R. et al. Hypertext Transfer Protocol -- HTTP/1.1, June 1999.

Disponivel em: <http://www.rfc-editor.org/rfc/rfc2616.txt>.

[20] BRAY, T.; PAOLI, J.; SPERBERG-MCQUEEN, M. **Extensible Markup Language (XML) 1.0**. 5ª Edição.: World Wide Web Consortium, 2008.

[21] CHRISTENSEN, E. et al. **Web Services Description Language (WSDL) 1.1**: World Wide Web Consortium, 2001.

[22] GUDGIN, M. et al. **SOAP Version 1.2 Part 1:** Messaging Framework. 2ª Edição.: World Wide Web Consortium, 2007.

[23] BELLWOOD, T.; CLÉMENT, L.; RIEGEN, C. V. **UDDI Spec Technical Committee Specification 3.0.1**: Organization for the Advancement of Structured Information Standards, 2003.

[24] THOMPSON, H. S. et al. **XML Schema Part 1:** Structures. 2ª Edição.: World Wide Web Consortium, 2004.

[25] THURLOW, R. RPC: Remote Procedure Call Protocol Specification Version 2, 2009. Disponivel em: <ftp://ftp.rfc-editor.org/in-notes/rfc5531.txt>.

[26] NADALIN, A. et al. **Web Services Security - SOAP Message Security 1.1**: Organization for the Advancement of Structured Information Standards, 2006.

Apêndice A

Componentes – Visão Geral

Neste apêndice, apresentaremos os principais componentes que foram desenvolvidos e utilizados ao longo desse trabalho. Podemos observar, na **Figura 24**, como se dá o relacionamento dos 5 componentes desenvolvidos neste trabalho, como eles são distribuídos e onde foi utilizado o *framework Mahout*.

O *iars.ear* é o componente que possui a implementação do sistema de recomendação inter-aplicações na forma de serviço. Para isso, ele incorpora outros 2 componentes, o *iars-service.jar* e o *iars-engine.jar*, onde o *iars-service.jar* é o responsável por fornecer a implementação do módulo de serviço e o *iars-engine.jar* a implementação do módulo de contextualização e de recomendação.

Por outro lado, o *iars-web.war* é o componente que possui a implementação da ferramenta desenvolvida para dar suporte a análise das recomendações e a coleta de dados neste trabalho. Para consumir o serviço exposto pelo componente *iars.ear*, este componente se utiliza de um outro componente, o *iars-service-client.jar*, que possui um conjunto de classes geradas a partir do descritor de serviço definido no *iars-service.jar*.

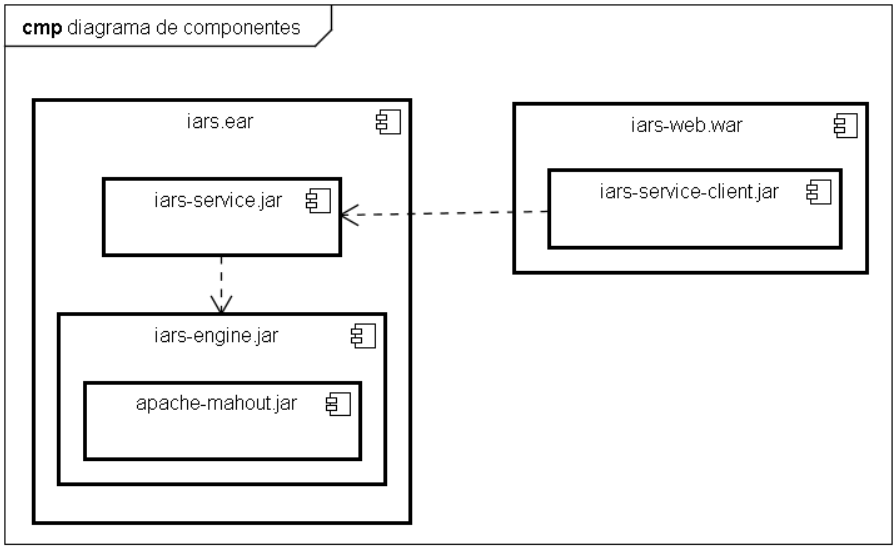


Figura 24. Componentes relacionados ao IARS.

Apêndice B

Componentes - Visão Arquitetural

Neste apêndice, apresentaremos uma visão da arquitetura adotada por cada um dos componentes citados no **Apêndice A**, ou seja, o *iars-engine.jar*, *iars-service.jar*, *iars-service-client.jar* e o *iars-web.war*. O *iars.ear* não será detalhado devido a ele só realizar o agrupamento dos componentes *iars-engine.jar* e *iars-service.jar*, permitindo desta forma que eles sejam distribuídos em conjunto.

O *iars-engine.jar* foi projetado em camadas e em alguns momentos se utilizou dos princípios do padrão de projeto *Strategy.* Uma visão da arquitetural deste componente é apresentada na **Figura 25**.

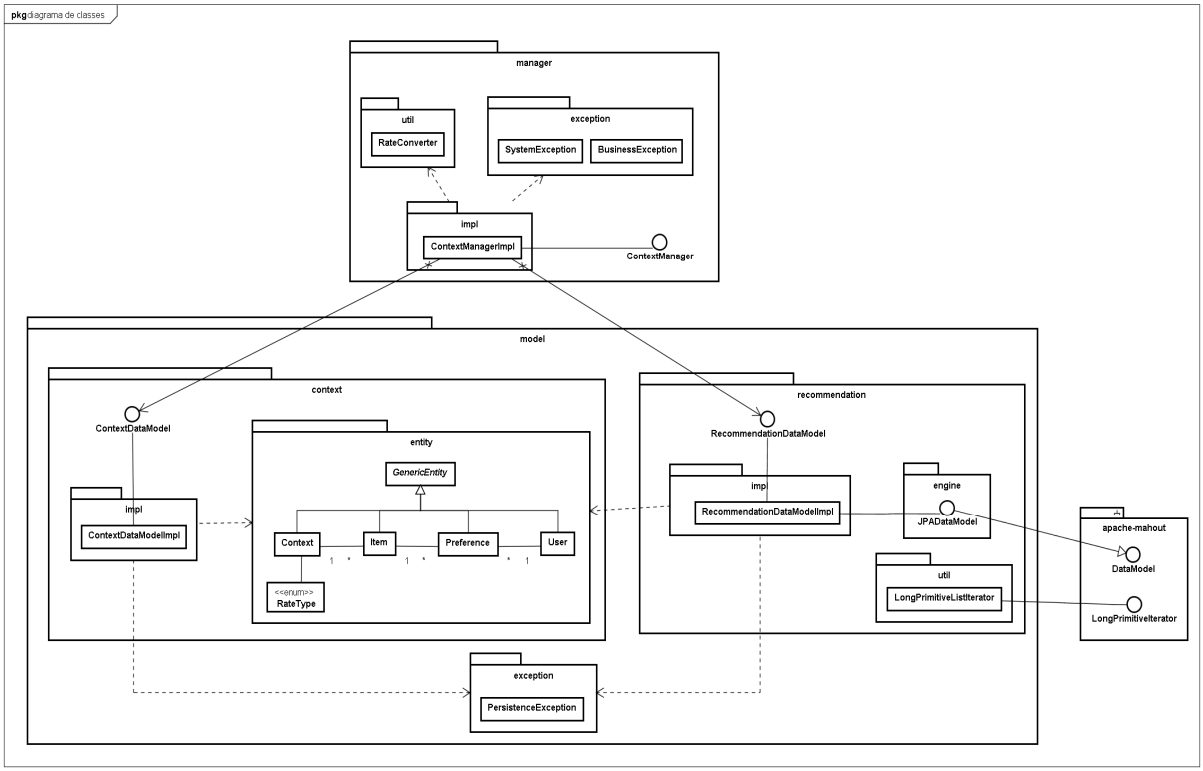


Figura 25. Visão arquitetural do *iars-engine.jar*.

O *iars-service.jar* define um conjunto de tipos e fornece a implementação das operações disponibilizadas pelo serviço. A arquitetura adotada por este componente pode ser vista na **Figura 26**. Pode-se observar que ele utiliza as funcionalidades do *iars-engine.jar* a partir da interface *ContextManager*.

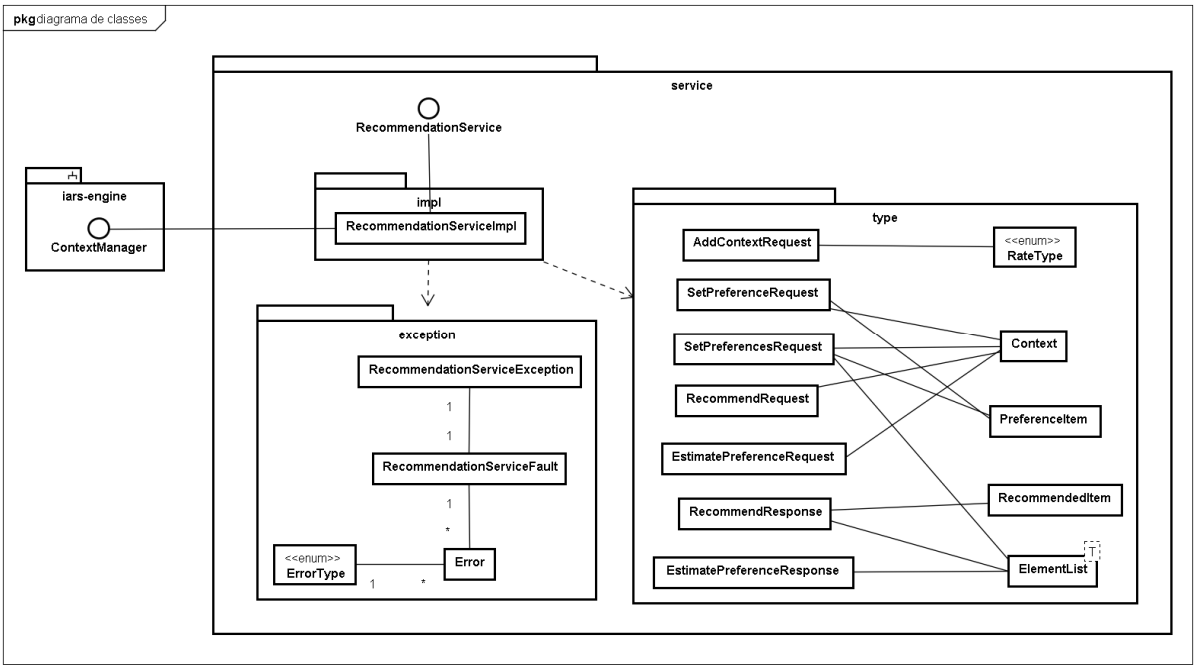


Figura 26. Visão arquitetural do *iars-service.jar*.

O *iars-service-client.jar* é composto basicamente por classes geradas a partir do descritor do serviço disponibilizado pelo componente *iars-service.jar*. A sua arquitetura é apresentada na **Figura 27**.

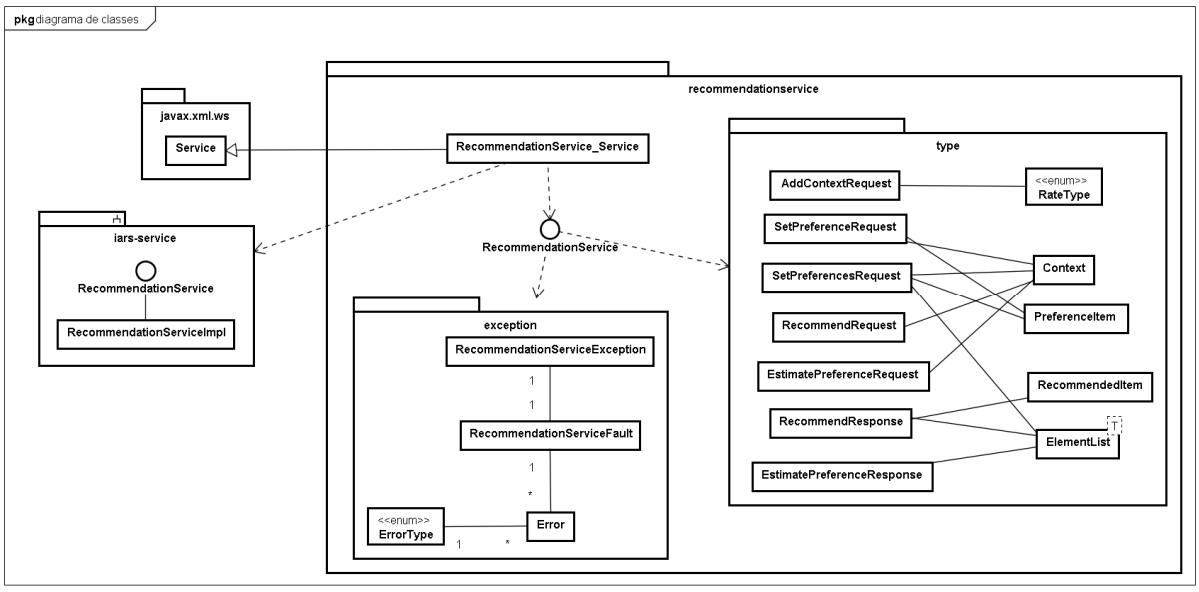


Figura 27. Visão arquitetural do *iars-service-client.jar*.

Por último, temos o *iars-web.war*, o componente que define todo o sistema de suporte a análise de recomendações e coleta de dados desenvolvido nesse trabalho. Este sistema foi projetado a partir do padrão arquitetural *Model-View-*

Controller, também conhecido como MVC. A arquitetura deste componente é apresentada na **Figura 28**.

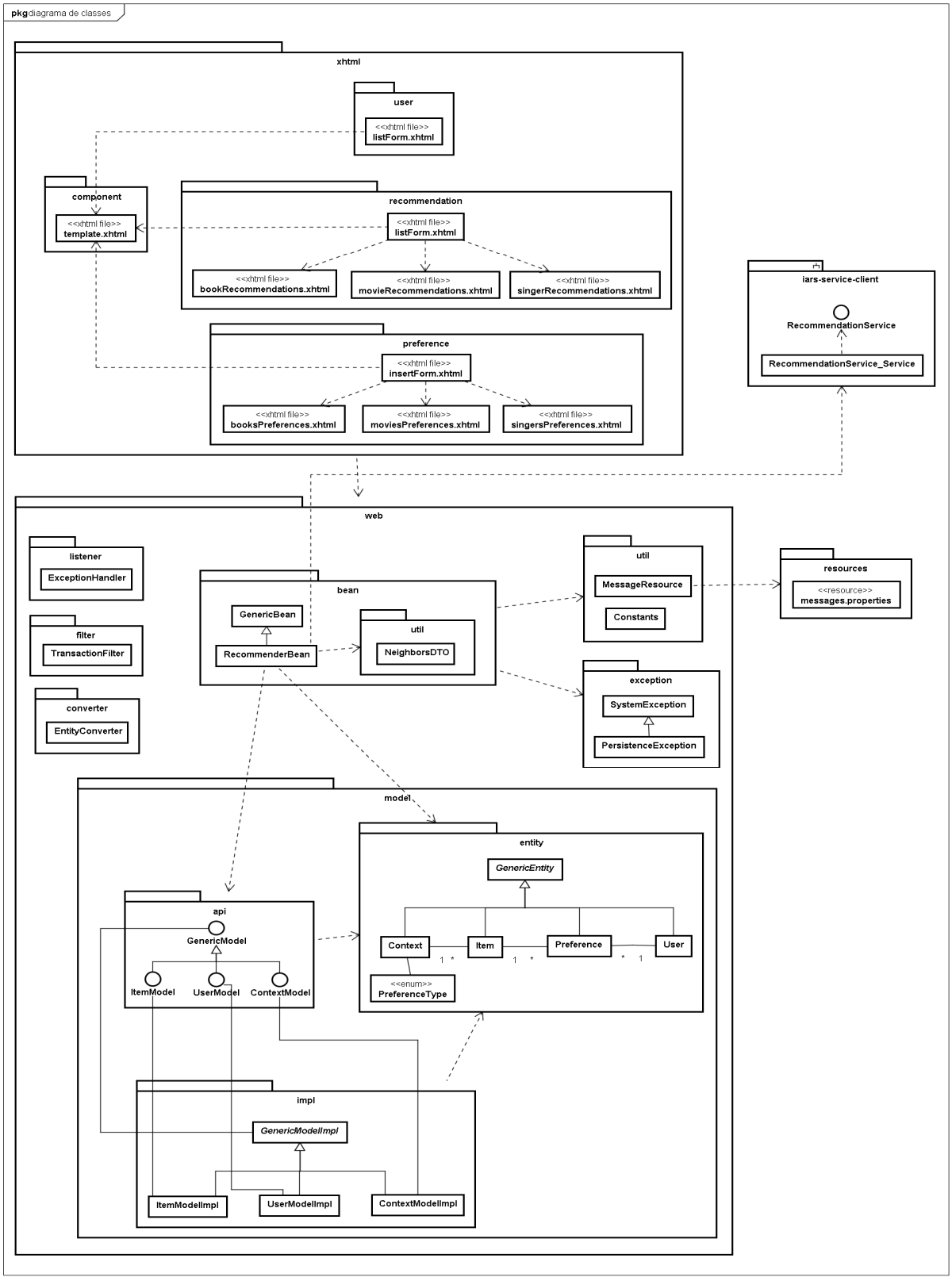


Figura 28. Visão arquitetural do *iars-web.war.*