Revista da Graduação

Vol. 4 No. 1 2011 23

Seção: FACULDADE DE INFORMÁTICA

Título: MAÎTRE: Motor de Recomendação para Dispositivos Móveis

Autores: Guilherme Borges Colombo e Pablo Meira Costa

Este trabalho está publicado na Revista da Graduação.
ISSN 1983-1374
http://revistaseletronicas.pucrs.br/ojs/index.php/graduacao/article/view/8809/6173

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
FACULDADE DE INFORMÁTICA
CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO

GUILHERME BORGES COLOMBO
PABLO MEIRA COSTA

MAÎTRE

Motor de Recomendação para Dispositivos Móveis

Porto Alegre
2010

GUILHERME BORGES COLOMBO
PABLO MEIRA COSTA

MAÎTRE: MOTOR DE RECOMENDAÇÃO PARA DISPOSITIVOS MÓVEIS

Trabalho de conclusão de curso de graduação apresentado à Faculdade de Informática da Pontifícia Universidade Católica do Rio Grande do Sul, como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação.

Orientador: Professor Dr. Michael da Costa Móra

Porto Alegre

2010

RESUMO

O crescimento constante da população mundial e a quantidade de informações disponíveis hoje configuram o que se pode chamar de uma revolução contemporânea da comunicação, agravada pelo fato de que a produção de informações existentes na Internet excede a capacidade de sua absorção pelas pessoas, demandando a exploração de ferramentas capazes de abarcar essa problemática, como a discussão aqui proposta. Esse Trabalho de Conclusão de Graduação dispõe-se a discutir e implementar um Motor de Recomendação para aplicações voltadas a Dispositivos Móveis, denominado *Maître,* que disponibiliza uma plataforma capaz de fornecer recomendações a seus usuários. Metodologicamente, o trabalho pesquisa técnicas de classificação e filtragem de informações, tomada de decisão e dispositivos móveis. Como resultado esperado desejou-se projetar uma arquitetura flexível, disponível, distribuída e provida de um motor de classificação e busca englobados em um Motor de recomendação para as aplicações desenvolvidas para dispositivos móveis. O resultado obtido demonstrou a aplicabilidade da arquitetura proposta através da implementação e capacidade operacional do projeto piloto, chamado *Apprenti Maître*.

Palavras-Chave: Motor de Recomendação. Sistemas de Recomendação. Dispositivos Móveis. Motor de Busca. Motor de Classificação.

ABSTRACT

The global population increasing and the range of information available today configure what we can call a Contemporary Revolution of Communication, aggravated by the fact that the quantity of information produced on the internet, exceed the capacity of the absorption of it, demanding inquire to find tools capable of enclose this problem, as the argument proposed at this time. This Coursework of Graduation, intend to discuss and implement an Engine of Recommendation to application turned to Mobile Devices, titled Maître, which provides a platform capable to give recommendations to who use it. Methodologically, the work search techniques of information's rating and filtering, take of decisions and mobile devices. As expected result, the wanted was to design a well-built architecture, available, distributed and provided of an arrangement and search engine, comprise in a Recommendation Engine to developed applications to mobile devices. The results show us the applicability of the architecture proposed by the implementation and operational capacity of the pilot project, named Apprenti Maître.

Keywords: Recommender Systems. User Profile. Decisions Making. Mobiles Gadgets. Search Engines. Recommendation engines. Social Network.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE SIGLAS

ADO	*ActiveX Data Objects*
API	*Application Programming Interface*
DTO	*Data Transport Objects*
ED	Estrutura de Dados
Linq	*Learn about language-integrated query*
MR	Motor de Recomendação
SR	Sistema de Recomendação
TSQL	*Transact Structured Query Language*
URL	*Universal Resource Locator* ADO
XML	*eXtensible Markup Language*

SUMÁRIO

1. INTRODUÇÃO

A população mundial[1] vem crescendo constantemente, o que permite inferir que o processo de criação e circulação de informações também acompanhe esse crescimento. Essas informações compõem o que Pierre Lévy (1999) chama de *a revolução contemporânea da comunicação*, que permitiu o enriquecimento da humanidade através de ferramentas como a *Internet* e a digitalização da informação.

Convém ressaltar-se que hoje, a produção de informações existentes na *Internet,* entre outros meios, segundo Zaiane (2000), excede o que a pessoa é capaz de absorver. Esse grande número informações acarreta numa sobrecarga. Segundo Belkin (2000), esse *excesso* pode causar ao usuário o sentimento de confusão e impossibilidade de obter realmente as informações que deseja. Some-se a isso o aspecto da *dispersão* das informações que, segundo Braga (2005) causaria confusões e decisões equivocadas, ao invés de embasar o conhecimento e auxiliar a tomada de decisão, gerando, inclusive, ansiedade.

Um meio para tentar amenizar esse problema poderia ser a utilização de ferramentas como os *motores de busca*. Entre estes, poder-se-ia citar os mais conhecidos, como o *Google (www.google.com),* o *Bing (www.bing.com),* o Terra (www.terra.com.br), o UOL (www.uol.com.br), dentre outros. Através deles, o usuário tem a possibilidade de buscar a informação que deseja através da inserção de *palavras-chave* no que é conhecido como *sistemas de classificação de informações*, localizados nesses motores de busca. Isso implica em facilitar o difícil processo de tomada de decisão do usuário que, inseguro em escolher, fica com a sensação de que poderia obter mais algumas informações que lhe dariam mais embasamento, conforme salienta Braga (2005). Além disso, o tempo para reflexão é cada vez mais escasso, cedendo lugar ao tempo gasto na absorção de mais e mais informações.

Se por um lado se verificam essas dificuldades, por outro, surge um aspecto positivo, como, por exemplo, a modificação e ampliação da memória e da percepção do mundo, que, segundo Lévy (2000) é realizada através das informações disponíveis na *Internet,* que possibilitaram com que os instrumentos se

[1] Relatório sobre a situação da população mundial 2010 – UNFPA - http://www.ufpa.org.br/swop2010. Acesso em 28/10/2010 às 18:10.

transformassem em *coletivos*. Esse autor traz a perspectiva, inclusive, do surgimento de uma *Inteligência coletiva*.

Em que pesem esses aspectos positivos, aparentemente poucos usuários parecem ter condições ou experiência para fazer o correto uso dos motores de busca. Estes parecem não ser capazes de exibir informações direcionadas às necessidades específicas dos usuários. Essas, quando exibidas, também não consideram o perfil do usuário, ignorando o que poderia influenciar os resultados recebidos durante a busca de informações. Outro fator relevante é o histórico de navegação anterior do usuário, que poderia inferir determinados resultados não previstos em resultados tradicionais. Esse fator parece não prever o compartilhamento da informação, contrapondo o que é proposto por Lueg (1998).

Assim, um Sistema de Recomendação (SR) poderia ser considerado um dos elos que complementa e consolida o compartilhamento e a classificação de informações, influenciando e focando os itens recomendados pelos Sistemas com intuito de objetivar as recomendações, abrandando os problemas citados anteriormente (REATEGUI E CAZELLA, 2005).

Em confluência com o aumento da disponibilização das informações, as tecnologias inseridas em Dispositivos Móveis, como celulares, *smartphones*, *PDAs*, dentre outros, também se encontram em evolução e parecem destinadas a transformar-se no novo paradigma dominante da computação conforme anotam Myers & Beigl (2003).

É cada vez maior a queda dos custos de aquisição e crescente disponibilidade no mercado de novos modelos de dispositivos móveis[2]. Contudo, esse mercado ainda está em expansão.

Segundo Silva (2008), a maioria dos dispositivos móveis não possui um hardware com grande capacidade de processamento, sendo um dos maiores desafios dos profissionais envolvidos no desenvolvimento de aplicações voltadas a estas tecnologias. Com isso, o potencial dos dispositivos móveis ainda não possibilita grandes aplicativos capazes de satisfazer a voracidade com que os consumidores de tecnologia estão habituados a usufruir em seus computadores pessoais. Baseado nisso, seria inviável desenvolvedores criarem um bom Sistema de Recomendação que pudesse ser instalado fisicamente em um Dispositivo Móvel.

---

[2] Conforme site IBGE (Instituto Brasileiro de Geografia e Estatística) http://www.ibge.gov.br/home/presidencia/noticias/noticia_impressao.php?id_noticia=1745

A necessidade da construção de plataformas que suportem este grande processamento necessário à qualificação de informações disponíveis, emerge meio a ampla capacidade de utilização destes Dispositivos Móveis, sustentando a ampliação da comunicação social.

Essa comunicação, segundo Lévy (2002), não é aleatória e só progride se houver a partilha de funções cognitivas quando aumentadas e transformadas por sistemas técnicos e externos ao organismo humano, como a *Internet*, capaz de compor o que este autor chama de inteligência coletiva. Essa se desenvolveu na medida em que a linguagem evolui e permite a interconexão e a ubiqüidade que são características de uma era de informação.

O autor Rheingold (2000) refere-se a esse estágio de comunicação pessoal como o início de uma nova revolução social. Então, para que os usuários de Dispositivos Móveis possam se beneficiar de todo o potencial dos Sistemas de Recomendação, é necessário criar um novo elo entre ambos. Esse elo pode ser um Motor de Recomendação (MR), contendo um conjunto de técnicas que recomendam um item às aplicações desenvolvidas voltadas a Dispositivos Móveis.

1.1 OBJETIVOS

1.1.1. Objetivo Geral

O principal objetivo deste trabalho de conclusão de curso é disponibilizar uma plataforma provida de um conjunto de técnicas capazes de fornecer recomendações de conteúdo para aplicações desenvolvidas e focadas em Dispositivos Móveis constituindo assim um Motor de Recomendação.

1.1.2. Objetivos Específicos

Os objetivos específicos deste trabalho são:

a) Pesquisar técnicas de classificação de informações, filtragem de informações, tomadas de decisão e dispositivos móveis;

b) desenhar uma arquitetura flexível provida de algoritmos que componham um Motor de Recomendação, algoritmos estes baseados em: perfil de usuário, histórico de utilização, redes sociais e conhecimento;

c) o Motor de Recomendação irá prover uma *API* para desenvolvedores que necessitem recomendar qualquer tipo de item através de palavras-chave, seja um filme, um livro, um lugar, um vinho ou qualquer outro item em seu sistema, através dos algoritmos do Motor de Recomendação; e

d) provar que conceito do Motor de Recomendação pode ser aplicável através do desenvolvimento de uma aplicação simples instalada em um Dispositivo Móvel.

2. FUNDAMENTAÇÃO TEÓRICA

Baseando-se na necessidade da criação de uma plataforma que viabilize o desenvolvimento de aplicações voltadas a Dispositivos Móveis que necessitem de recomendações, neste capítulo são apresentadas informações sobre: o processo decisório, conceitos sobre Sistemas de Recomendação e suas abordagens, técnicas de classificação, exemplos e casos práticos e, por fim, conceitos sobre Dispositivos Móveis.

2.1. TOMADA DE DECISÃO

A Tomada de Decisão é o processo de definição, dentre um determinado número de opções, daquela que se adequa melhor às necessidades do usuário, segundo Chiavenato (1997). É a análise de uma gama de opções e escolha de qual delas a pessoa irá selecionar. No Motor de Recomendação, tomar decisões é um fator de influência para a abordagem de Filtragem de Informação. Para Gomes et al. (2006), esse processo é um posicionamento em relação ao futuro dentro do contexto atual.

Na visão de um Motor de Recomendação, a Tomada de Decisão está diretamente ligada às classificações feitas a partir dos dados históricos de tomadas de decisão anteriores, para que, então, se busque uma lista com as informações pertinentes aquele usuário. Segundo Oliveira (2004), o processo de decisão faz a conversão das informações em ações. Sendo assim, a cada decisão tomada pelo usuário frente aos itens recomendados pelo Motor de Recomendação, maior será o aprendizado sobre este.

2.1.1. Elementos da Decisão

Segundo Chiavenato (1997), para que o processo de decisão fique claro para quem o está utilizando, são necessários alguns elementos. São eles:

1. **Tomador de Decisão**: é a pessoa que faz a seleção dentre várias alternativas de atuação;

2. **Objetivos**: propósito ou finalidade que o Tomador de Decisão almeja alcançar com sua ação;
3. **Preferências**: critérios com juízo de valor do Tomador de Decisão que distinguirá a escolha;
4. **Estratégia**: direção ou caminho que o Tomador de Decisão sugere para melhor atingir os objetivos e que depende dos recursos que se dispõe;
5. **Situação**: são os aspectos ambientais fora do controle, conhecimento ou compreensão dos quais se vale o Tomador de Decisão que afetam a opção; e
6. **Resultado**: é a decorrência ou resultante de uma dada estratégia definida pelo decisor.

2.1.2. Processo de Tomada de Decisão

O processo de Tomada de Decisão está dividido em sete fases sugeridas, pois, sendo um processo, busca as características do usuário, neste caso, do Tomador de Decisão (Chiavenato, 1997). O algoritmo de tomada de decisões possui suas etapas (modificáveis) dependentes umas das outras. São as seguintes:

1. Percepção da situação que abrange algum problema;
2. Diagnóstico e definição do problema;
3. Definição dos objetivos;
4. Busca de alternativas de solução ou de cursos de ação;
5. Escolha da alternativa mais apropriada ao alcance dos objetivos;
6. Avaliação e comparação dessas alternativas; e
7. Implementação da alternativa escolhida.

Para simplificação do modelo, esse autor sugere o corte e supressão de três etapas quando a necessidade da decisão obtida for imediata, formando um modelo como prescrito abaixo.

1. Percepção da situação que abrange algum problema;
2. Diagnóstico e definição do problema;
3. Busca de alternativas de solução ou de cursos de ação; e
4. Avaliação e comparação dessas alternativas.

2.2. SISTEMAS DE RECOMENDAÇÃO

Os Sistemas de Recomendação servem para filtrar informações de um determinado usuário através de dados previamente conhecidos.

Para a criação de um vínculo entre a informação e o usuário existem algumas abordagens focadas na obtenção e manipulação das informações. Basicamente, essas abordagens possuem uma inteligência para a criação deste vínculo.

Existem princípios básicos para recomendar algo a alguém, podendo-se classificar as informações em diversos níveis, como: grandeza, quantidade de acesso, ou, ainda, perfil de usuário. Para qualquer uma das abordagens, o Motor de Recomendação visa estabelecer alguma ligação lógica para indicar ao usuário alguns itens. Essa ligação baseia-se em alguma característica cadastrada previamente pelo usuário ou em uma inferência feita pelo sistema a seu respeito, inferência essa baseada em um perfil ou em um histórico de acesso (SCHAFER, 1999 apud CAZELLA, 2006).

A inteligência dessas técnicas é descrita por Montaner (2003 apud CAZELLA, 2006) como um processo de identificação e localização de informações sobre itens, pessoas e fontes de informação relacionadas a um usuário ou a um grupo ao qual ele pertença. A partir desta coleta, é possível que o Sistema de Recomendação antecipe suas indicações ao usuário, ou seja, o Sistema pode predizer itens a respeito do utilizador. Discorrendo sobre esse tema do ponto de vista do usuário, predizer informações e aprender com o uso do sistema é montar um Perfil do Usuário de forma implícita.

O Motor de Recomendação tem como ponto central os algoritmos de recomendação, que servirão de fonte para a busca de dados. Dessa forma, o processo de classificação é fundamentado pela forma como é aplicado o conceito de filtragem da informação. Em decorrência, essa classificação surge como alternativa para solucionar uma questão que, de acordo com Resnick & Varian (1997 apud CAZELLA, 2006), faz com que os usuários consigam dicernir o que é informação útil, ou seja, auxilia a interpretação de tudo o que lhes é entregue.

Dentre os métodos de filtragem da informação, iremos tratar os que atendam às necessidades de classificação buscando a diferenciação da informação por Perfil, Rede Social, Histórico e Conhecimento.

Uma das aplicações do Sistema de Recomendação é o auxílio da Tomada de Decisão, pois este incrementa o processo decisório no dia-a-dia. Mesmo possuindo diferentes abordagens, um Sistema de Recomendação almeja enquanto conceito a recepção de dados e direcionamento dos resultados para o cliente (RESNICK & VARIAN, 1997 apud CAZELLA, 2006).

As abordagens utilizadas para compor um Sistema de Recomendação podem ser compreendidas conforme a figura 1, onde a Rede Social, representada por perfis de outros usuários, é vinculada às técnicas baseadas em conteúdo e filtragem colaborativa. Já o Perfil do Usuário utiliza adicionalmente a Rede Social e a filtragem baseada em conhecimento. Como o objetivo do trabalho não utiliza técnicas que considerem a demografia, os dados sobre as mesmas não serão considerados.

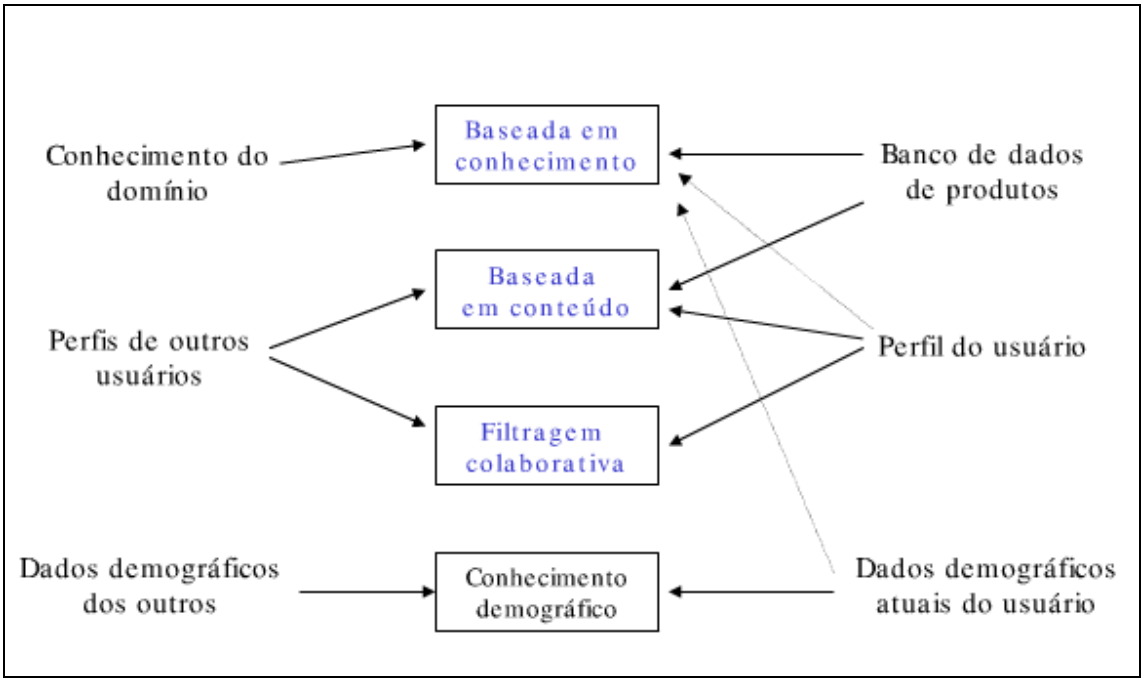


Figura 1: Sistema de Recomendação

2.2.1. Recuperação da Informação

Belvin (1992 apud CAZELLA, 2006) descreve um Sistema de Recuperação como um sistema que conduz o usuário diretamente para aquilo que lhe é interessante. O autor cita que os exemplos mais encontrados são: os motores de busca na *Web* e os sites de pesquisa.

Juntamente com o termo Filtragem de Informação, essas abordagens servem para que o usuário tenha acesso a informações mais precisas, diferentemente de

quando ele busca alguma informação na *Internet* sem ter uma *interface* entre o seu objetivo e o motor de busca.

O ponto chave no processo de recuperação da informação é a descoberta do que é importante para o usuário, ou seja, processo esse que, de alguma maneira, está a ele ligado e diretamente relacionado com o que está buscando. Após o término da obtenção do que é relevante, para que o processo seja concluído, é necessário ainda reordenar os itens para que os indesejáveis tenham menor probabilidade de utilização. Em alguns casos, estas informações são armazenadas, evitando assim a recorrência do aparecimento desses itens.

Normalmente, os Sistemas de Recomendação servem como elo entre o motor de busca e o usuário, pois, mesmo que os motores de busca sejam bons recuperadores de informação, para utilizar seus recursos, o mesmo necessita dominar amplamente seus interesses, além de saber utilizar o motor de busca.

Durante o processo de recuperação de informação, as ferramentas ditas boas agentes de recomendação estão, na verdade, cumprindo um papel equivocado enquanto Sistemas de Recomendação sendo, de fato, apenas motores de busca (HERLOCKER, 2000 apud CAZELLA, 2006).

2.2.2. Filtragem de Informação

Para Foltz (1992 apud CAZELLA, 2006), a Filtragem de Informação está ligada à Recuperação da Informação, porém, é estruturada e possui processamento dinâmico. Esta última tende a ser estática, pois, é feita através da indexação dos documentos armazenados, ao contrário da Filtragem de Informação que busca o armazenamento de informações sobre o usuário montando um perfil de como o usuário a utiliza.

Segundo o autor (idem, *ibidem*), a grande diferença reside no fato de que, enquanto a recuperação faz seu processamento de acordo com as necessidades momentâneas do usuário, a Filtragem de Informação visa uma especificação clara de um perfil de usuário, perfil este cadastrado na medida em que a pessoa utiliza o sistema. Na Filtragem de Informação, o processo tende a ser mais longo, pois, as informações a respeito desta são mantidas.

Muito antes de se tornarem populares, as práticas de Filtragem de Informação já eram descritas por Loeb (1992 apud REATEGUI e CAZELLA, 2005). De acordo

com esse autor, os sistemas geravam muita demanda por enviarem cada vez mais informações aos usuários. Isto ocorria porque o foco estava na geração de todas as informações necessárias para que essas pudessem suprir as necessidades dos usuários. Porém, este conceito se contrapõe a idealização inicial da Filtragem de Informação. Por isso, foram melhorados os sistemas de coleta de informação e desmembrados em diversas técnicas.

A partir dessas divergências na utilização da Filtragem de Informação, surgem novas idéias para esclarecer esse conceito. Por exemplo, o ideal de Belvin (1992 apud CAZELLA, 2006), mostrando que o leque de processos envolvidos na entrega de informação a um usuário que as necessita, é Filtragem de Informação.

2.2.2.1. Abordagens de Filtragem de Informação

A seguir, são apresentados alguns exemplos de classificações de técnicas de filtragem da informação aplicadas, conforme o Quadro de Classificação de Recursos de Sistemas de Recomendação:

Abordagem de Recomendação	**Técnica de Recomendação**	
	Baseado em heurística	**Baseado em Modelo**
Baseado em conteúdo	Técnicas normalmente usadas: • TF-IDF (recuperação da informação) • Clusterização Exemplos representativos de pesquisa: • Lang 1995 • Balabanovic & Shoham 1997 • Pazzani & Billsus 1997	Técnicas normalmente usadas: • Classificações Beyesianas • Clusterização • Árvores de decisão • Redes artificiais neurais Exemplos representativos de pesquisa: • Pazzani & Billsus 1997 • Mooney et al. 1998 • Mooney & Roy 1999 • Billsus & Pazzani 1999, 2000 • Zhang et al. 2002
Colaborativo	Técnicas normalmente usadas: • Vizinho mais próximo • Clusterização • Teoria dos Grafos Exemplos representativos de pesquisa: • Resnick et al. 1994 • Hill et al. 1995 • Shardanand & Maes 1995 • Breese et al. 1998 • Nakamura & Abe 1998 • Aggarwal et al. 1999 • Delgado e Ishii 1999 • Pennock & Horwitz 1999 • Sarwar et al. 2001	Técnicas normalmente usadas: • Classificações Beyesianas • Clusterização • Redes artificiais neurais • Regressão linear • Modelos probabilísticos Exemplos representativos de pesquisa: • Billsus & Pazzani 1998 • Breese et al. 1998 • Ungar & Foster 1998 • Chien & George 1998 • Getoor & Sahami 1999 • Kumar et al. 2001 • Pavlov e Pennok 2002 • Shani et al. 2002

		• Yu et al. 2002, 2004 • Hofmann 2003, 2004 • Marlin 2003 • Si & Jin 2003
Híbrido	Combinação de componentes baseado em conteúdo e colaborativo usando: • Combinação linear e ranqueamento preditivo • Vários esquemas de votação • Incorporando um componente ou parte dele da heurística para outro Exemplos representativos de pesquisa: • Balabanovic & Shoham 1997 • Claypool at. 1999 • Good et al. 1999 • Pazzani 1999 • Billius & Pazzani 2000 • Tran & Cohen 2000 • Melville et al. 2002	Combinação de componentes baseado em conteúdo e colaborativo por: • Incorporando um componente ou parte dele da heurística para outro • Construingo um modelo unificado Exemplos representativos de pesquisa: • Basu et al. 1998 • Condliff et al. 1999 • Soboroff & Nicholas 1999 • Ansari et al. 2000 • Popescul et al. 2001 • Schein et al. 2002

Quadro 1: Classificação de Recursos de Sistemas de Recomendação (CAZELLA, 2006). Fonte: (ADOMAVICIUS & TUZHILIN, 2005, p. 742).

2.2.2.1.1 Filtragem Colaborativa

Inicialmente, os Sistemas de Recomendação foram chamados de processos de “Filtragem Colaborativa”, nome dado pelos criadores desta tendência (GOLDBERG, 1992 apud CAZELLA, 2006). Nesses, apelidados de *Tapestry*, toda a recomendação era manual, sendo necessária a participação de diversos interessados. Nessa abordagem, acreditava-se que para recomendação funcionar, o “cliente” deveria ter o mesmo perfil do recomendador ou estar em um grupo similar.

Segundo Gomes (2001 apud OLIVEIRA & REIS, 2004), a filtragem colaborativa representa um processo de recomendação baseado na análise da similaridade entre a opinião de um determinado usuário e um grupo de usuários presentes em um sistema. Para que o usuário faça esta análise da similaridade devem ser levados em consideração o peso e classificação dos documentos já recomendados. Para executar esse processo, devem ser identificados três passos descritos abaixo:

1. O sistema deve identificar quem são os usuários que buscam os resultados e criar uma ligação entre aqueles que desejam as mesmas informações. Estes usuários são chamados de vizinhos;
2. O segundo passo é processar o que foi informado pelo usuário e, baseado em algum algoritmo de recomendação, ligar estas informações com o que seus vizinhos desejam receber, para abrir um leque maior de opções; e
3. Ao final da coleta de dados e de ligação entre os vizinhos, o algoritmo irá indicar as recomendações ao usuário.

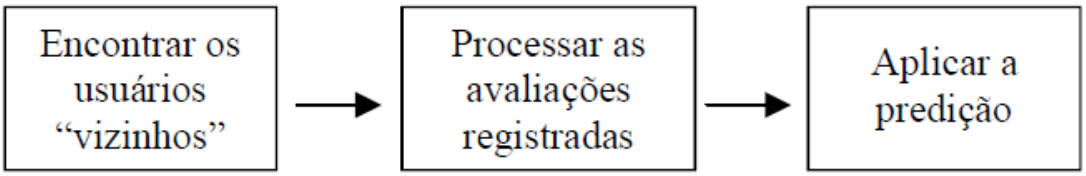


Figura 2: Passos para o processo de filtragem colaborativa (SOBOROFF, 2000 apud OLIVEIRA & REIS, 2004)

Rêgo (2006), para ilustrar uma filtragem colaborativa, criou uma tabela de Exemplo de Filtragem Colaborativa.

Nesse caso para a recomendação de um produto ao usuário Mauro, procuram-se outros usuários com hábitos de consumo semelhantes. No caso, Paulo e João já compraram produtos que Mauro também comprou (Prod2). Em seguida, são recomendados a Mauro produtos que estes dois usuários já possuem, mas, que Mauro ainda não, como: Prod1 e Prod5 (Rêgo, 2006).

Usuário	Prod1	Prod2	Prod3	Prod4	Prod5	Prod6
Paulo	-	x	-	-	x	-
João	x	x	-	-	-	-
Marcia	-	-	x	x	x	-
Carlos	-	-	x	-	-	-
Ana	x	-	-	x	-	-
Mauro	-	x	-	-	-	-

Tabela 1: Exemplo de Filtragem Colaborativa (Rêgo, 2006)

2.2.2.1.2 Filtragem Baseada em Conteúdo

De acordo com Rêgo (2006), no processo de Filtragem de Informação baseado em conteúdo são feitas comparações entre as descrições dos itens de informação e os modelos de usuário. Podem-se ter ainda modelos de usuário sendo atualizados com base na avaliação que o usuário faz de elementos de informação recebidos. Desta maneira, são filtrados itens similares a outros que o usuário tenha avaliado positivamente no passado (Baseada em Conhecimento).

Considerando que para usuários domésticos a sobrecarga de informações já é um problema, em um ambiente mais robusto de pesquisa, esta pode ocasionar um problema muito maior. Isso porque uma informação inválida, na medida em que depende exclusivamente de dados confiáveis, tem um efeito em cascata. Baseado nisso, Herlocker (2000 apud CAZELLA, 2006) demonstra que cientistas utilizam-se de meios que unam o reconhecimento da boa informação e sua categorização.

Balabanovic (1997 apud CAZELLA, 2006) afirma que a Filtragem Baseada em Conteúdo objetiva a geração automática de descrições dos conteúdos dos itens. Após esta geração, é feita uma comparação entre a descrição dos itens e o que realmente interessa ao usuário.

Esta abordagem ocorre por meio de Filtragem Baseada em Conteúdo porque as informações do conteúdo de interesse do usuário são conhecidas e extraídas de um perfil criado de maneira implícita ou explícita para, então, serem comparadas. Estas informações são obtidas por um histórico de consumo, de forma implícita ou explícita, por consulta.

A seguir são descritos alguns dos algoritmos utilizados para a filtragem de conteúdo:

- **Filtragem por Frequência de Termo**

Um dos algoritmos bastante utilizados na Filtragem Baseada em Conteúdo é o de identificar e indexar termos freqüentes dentro do conteúdo gerado sobre os itens. A execução deste algoritmo acontece através da ocorrência na descrição das palavras na base de dados por vetores, onde uma dimensão do vetor corresponde a uma palavra. Ainda na descrição, cada componente do vetor corresponde a uma freqüência baseada na ocorrência das dimensões dos vetores (palavras) dentro dos

conteúdos do conteúdo dos itens. O próximo passo é a classificação dos vetores, identificando os mais próximos do vetor de interesses do usuário, criando índices de relevância para cada um (HERLOCKER, 2000 apud CAZELLA, 2006).

- **Filtragem por Avaliação do Usuário**

Para a utilização da Filtragem Baseada em Conteúdo, pode ser feita uma solicitação de análise para o usuário consumidor, fazendo com que ele classifique/indique os itens de seu interesse. Após esta indicação, o sistema classifica os itens como *de interesse* e *não interesse*, considerando para a filtragem apenas os de interesse. Este processo pode ser agregado com o armazenamento tanto dos interessantes como dos não interessantes, para que, da próxima vez, os itens já venham mais selecionados para a análise do usuário.

O Sistema de Recomendação Baseado em Conteúdo, utilizando a avaliação do usuário, tem como premissa pontuações. A pontuação *R* de um item *i* por um usuário *u (R(u, i))* é baseada na pontuação fornecida pelo mesmo usuário (*R(u, i')*) para outros itens *i' ∈ Itens* similares ao conteúdo do item *i* (ADOMAVICIUS, 2005 apud CAZELLA, 2006). Este algoritmo pode ser descrito da seguinte forma: *Conteúdo (i)* é um conjunto de atributos que caracterizam o item *i*, que por sua vez é computado pela extração de um conjunto de características do item *i*. Geralmente tais sistemas são aplicados na recomendação de itens quando estes pertencem a documentos de texto tendo por base palavras-chave. Este conceito contrapõe-se ao de *algoritmo de freqüência de termos* (CAZELLA, 2006).

2.2.2.1.3 Híbrido

Pelo fato das diferentes técnicas de recomendação não garantirem total precisão para a recomendação de itens, a união dos pontos fortes de cada técnica traz uma garantia e um grau de confiabilidade maiores. A filtragem híbrida une as duas filtragens, a Colaborativa e a Filtragem por Conteúdo, buscando os pontos mais fortes e positivos destas.

De acordo com Adomavicius (2005, apud CAZELLA, 2006), para evitar a limitação de informações, o que pode ocorrer quando se tem dois caminhos a serem

escolhidos (colaborativa ou por conteúdo), os sistemas atuais utilizam muito a abordagem de filtragem híbrida. Para aperfeiçoar este processo, Cazella (2006) identificou os pontos mais fortes de ambas, conforme figura abaixo:

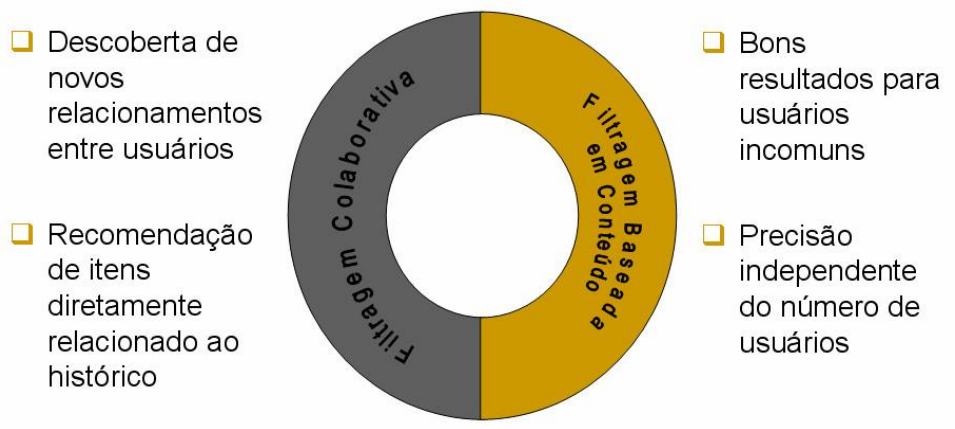


Figura 3: Pontos fortes da filtragem híbrida (Cazella, 2006).

Para a aplicação de técnicas para recomendação baseadas em uma abordagem híbrida, surgem técnicas, como as descritas a seguir.

- **Filtragem Colaborativa baseada em Conteúdo**

Um dos pontos fracos dos Sistemas de Recomendação é o utilizador inicial, que ainda não possui um grupo nem tampouco um perfil. Nesse caso, aplica-se uma técnica Híbrida baseada em conceitos de Filtragem Colaborativa e Conteúdo (BURKE, 2002 *apud* CASACA, 2008). Para que esta funcione, Li e Kim (2003 apud CASACA, 2008), criaram a ICHM (*Item-based Clustering Hybrid Method*), que integra a informação do item e as avaliações dos usuários para identificar o grau de similaridade, comparando um item com os demais. Li & Kim *(idem, ibidem)* descreveram seu funcionamento nos seguintes passos:

1. Aplicação do algoritmo *clustering* para agrupar os itens, utilizando os resultados representados por um conjunto difuso para criar uma matriz *grupo-avaliação*;
2. Cálculo da similaridade;
3. Cálculo da sub-similaridade da matriz *grupo-avaliação*;
4. Cálculo da sub-similaridade da matriz *item-avaliação*;
5. Combinação linear dos dois itens acima; e

6. Predição de um item fazendo a média ponderada do desvio das médias dos seus vizinhos.

Burke (2002, apud CASACA, 2008) mostra sete tipos de sistemas híbridos na combinação da Filtragem Baseada na Filtragem Colaborativa e na Filtragem Baseada em Conteúdo:

1. **Ponderado**: os resultados de várias técnicas de recomendação são agrupados para produzir uma única recomendação;
2. **Comutada**: o Sistema de Recomendação comuta várias técnicas, dependendo da situação, para produzir a recomendação;
3. **Mista**: várias técnicas utilizadas simultaneamente;
4. **Combinação de Características**: combina características dos dados obtidos da aplicação de diferentes técnicas como entrada para uma única técnica;
5. **Cascata**: uma técnica gera a recomendação e outra a classifica;
6. **Aumento de Características**: Uma técnica gera a recomendação como entrada para outra técnica; e
7. **Meta-nível:** uma técnica gera um modelo que servirá de entrada para outra técnica.

Adomavicius & Tuzhilin (2005 apud CASACA, 2008), combinam as técnicas de Filtragem Colaborativa e Filtragem Baseada em Conteúdo para garantir uma recomendação mais precisa. Estas combinações podem ser:

1. A utilização das duas técnicas e no final combinar seus resultados (predições)
2. Agregação da Filtragem Colaborativa com características de Filtragem Baseada em Conteúdo e vice-versa.
3. Construção de um modelo central formalizado incorporando características de ambas as técnicas.

2.2.2.1.4 Conhecimento Demográfico

Tanto a Filtragem Colaborativa quanto a Filtragem Baseada em Conteúdo têm como objetivo chegar a uma recomendação para seu usuário. Porém, existe outra abordagem que é a Demográfica (MONTANER ET al., 2003).

Na abordagem demográfica são utilizadas descrições de indivíduos, tais como: idade, ocupação, sexo etc. para determinar relações entre um determinado item e o tipo de indivíduo a que pertencem.

De acordo com Da Silva (2007), Perfis são criados a partir da classificação de utilizadores usando descrições convencionais e generalistas (estereótipos). Itens semelhantes são sugeridos para indivíduos com perfis demográficos semelhantes. Se as preferências ou interesses dos utilizadores mudarem constante ou rapidamente, os perfis demográficos não se adaptam a estes contextos. Os filtros demográficos raramente são utilizados independentes de outras técnicas de filtragem.

Na opinião de Montaner (MONTANER et al., 2003), a filtragem demográfica utiliza a descrição de um indivíduo para aprender sobre o relacionamento entre um item em particular e o tipo de indivíduo que poderia vir a se interessar. Essa abordagem utiliza descrições de pessoas para aprender sobre o relacionamento entre um item e o tipo de pessoa que o apreciaria. Uma classificação de usuários em classe de usuários, representada por um estereótipo, cria um Perfil do Usuário. Dados pessoais lhe são requisitados, geralmente em formulários de registro e utilizados posteriormente como caracterização dos usuários e seus interesses.

2.2.3. Técnicas de Classificação

O processo de classificação é fundamentado pelas maneiras de aplicação do conceito de Filtragem da Informação. A classificação, aplicada junto à Filtragem da Informação soluciona um ponto que, de acordo com Resnick & Varian (1997 apud CAZELLA, 2006), faz com que os usuários não consigam distinguir o que é uma informação útil, ou seja, não conseguem interpretar tudo o que lhes é entregue.

2.2.3.1 Perfil de Usuário

Os Sistemas de Recomendação, de modo geral, são baseados em informações referentes a quem os utiliza, muitas vezes buscando informações apenas deste. O problema ocorre quando o usuário ainda não possui informações e quer utilizar o sistema, o que Cazella (2006) assinala como um empecilho, mostrando que os Sistemas de Recomendação possuem dois problemas: a super especialização do usuário e os usuários novos. No caso de um usuário novo, o sistema não dispõe de informações a seu respeito, o que gera duas perguntas: o

que e onde buscar informação? O segundo ponto focaliza-se no Perfil do Usuário, o que para o autor representa:

> Os usuários não possuem a chance de explorar novos itens que não sejam similares aos itens incluídos em seu perfil, isto é, o sistema não recomenda itens fora do domínio de atuação do usuário. Esta situação pode criar um descontentamento no usuário, uma vez que os interesses dos usuários tendem a se modificar e se aperfeiçoar com o tempo.

O Perfil do Usuário é uma das peças-chave do Motor de Recomendação proposto, pois, se combinado às técnicas de recuperação e filtragem da informação, a chance de um usuário necessitar de uma informação e obter algo fora de escopo é reduzida de acordo com o uso do sistema. Como na maioria dos sistemas, os perfis são agregados na medida em que o usuário utiliza suas funcionalidades e clica sobre os itens.

Existem técnicas para construção de perfis a partir das semelhanças dentro de um grupo formado pelas necessidades dos integrantes, mas esta não está implementada na arquitetura. Os perfis de usuário são criados mediante um cadastro direto, e não por inferência.

2.2.3.2 Rede Social

Para Resnick & Varian (1997 apud CAZELLA, 2006), Sistemas de Recomendação também podem ser definidos como aqueles que utilizam as opiniões de uma comunidade de usuários (Rede Social) para auxiliar indivíduos desta comunidade a identificar interesses comuns em buscas que trazem por padrão uma quantidade grande de informação descartável.

Dentre as técnicas mais utilizadas para classificação baseada em redes sociais, a filtragem colaborativa é a que une a criação dos perfis dos usuários com a identificação de quem compõe uma rede. Esta técnica, de acordo com Soboroff (2000 apud OLIVEIRA & REIS, 2004) é uma junção eficaz que traz mais precisão na busca, ocorrendo através de uma análise que envolva avaliações anteriores feitas por outros usuários da Rede.

A filtragem colaborativa possui duas formas de realizar seu processamento: baseado em modelo ou em memória, sendo este o mais utilizado. Dentro os

algoritmos baseados em memória, aparecem dois indicadores para medição de similaridade: o coeficiente de correlação e o vetor da similaridade (PENNOCK, 2000 apud OLIVEIRA & REIS, 2004).

Já os algoritmos baseados em modelo têm como objetivo montar o modelo de busca de modo desligado, ou seja, gastando bastante tempo na montagem. Esse algoritmo deve ser utilizado quando as informações sobre os usuários têm características estáticas, sendo bastante pesado quando as informações são dinâmicas (YU, 2001 apud OLIVEIRA & REIS, 2004). Estas duas formas são máquinas de inferência, porém, cada uma distribuindo o peso da montagem da rede em momentos distintos.

2.2.3.3 Histórico

Quando a classificação ocorre em função das interações do usuário com o sistema, a técnica de Filtragem Colaborativa é bastante recomendada, pois o Histórico é uma das maneiras para criar um Perfil. Ele é um dos componentes do Sistema de Recomendação que, sempre que utilizado armazena ações do usuário (RESNICK & VARIAN, 1997 apud CAZELLA, 2006).

Baseados nesses Históricos os Sistemas de Recomendação mineram dados para montar pesquisas de acordo com cada usuário e ainda podem ser obtidos de forma persistente, quando a informação é obtida pela demografia ou consumo do item, ou efêmera, baseada nas ações do usuário, ou, ainda, pela união das duas formas (SCHAFER, 2005 apud CASACA, 2008).

Um modelo chamado Pelep (LEVIS 2008 apud DA SILVA et al, 2005) foi idealizado para estar conectado a um sistema ubíquo e utilizar o Histórico de interações dos usuários. Este agrega mais informações ao Perfil do Usuário ou, até mesmo, concretiza o que está descrito no seu perfil.

De acordo com Levene (2002 apud DA SILVA et al, 2005), uma Trilha é uma coleção de locais com informações de contexto associadas, composta por uma *ordem de visitação*. A Trilha é o caminho que um usuário percorre até alcançar um item. Seu conceito na *WEB* indica a sessão do usuário, ou seja, uma sessão permanente, pois o histórico é mantido e alterado constantemente na medida em que o usuário trabalha com o sistema. Além disso, a Trilha é o que melhor demonstra o Perfil de um usuário em um sistema, pois suas ações demonstram o

que o usuário realmente utiliza do sistema e não o que apenas informa em seu cadastro (LEVENE, 2002 apud DA SILVA et al 2005).

2.2.3.4 Conhecimento

Felfernig (2005 apud CASACA, 2008) traz a recomendação Baseada em Conteúdo como uma das abordagens básicas em um Sistema de Recomendação. Na Filtragem Baseada em Conteúdo, o sistema explora um conhecimento profundo no domínio com o objetivo de indicar as soluções mais adequadas às preferências e às necessidades do usuário.

Já Burke (2000 apud CASACA, 2008) assinala que os Sistemas de Recomendação Baseados em Conhecimento desempenham uma função indispensável em um universo onde as informações assumem proporções cada vez maiores.

2.3. FERRAMENTAS EXISTENTES

Atualmente, existem diversas ferramentas que utilizam Sistema de Recomendação em suas aplicações e predições a respeito dos interesses dos usuários. Abaixo estão descritas algumas destas, bastante utilizadas e conhecidas.

2.3.1. Amazon.com

Amazon é um portal de vendas *on-line*, que, segundo Cazella (2006), utiliza alguns dos recursos de um Sistema de Recomendação do seu *site*, cujas funcionalidades são analisadas a seguir:

- **Clientes que Compraram**: o *site* disponibiliza uma lista de clientes que compraram o produto e outra com produtos da mesma marca/autor. Através de uma simples relação têm-se duas dimensões para realizar recomendações;
- **Entregas**: o usuário pode solicitar o envio por *e-mail* das últimas recomendações periodicamente, além de novidades de categorias selecionadas em seu Perfil; e

- **Classificação de Produtos**: os usuários podem fornecer sua opinião com valores em uma escala entre “Detestar” e “Gostar” sobre os produtos comprados. Esse conceito corresponde ao que o Sistema de Recomendação propõe.

2.3.2. eBay

O *site* de leilões *on-line* *eBay.com* possui diversas estratégias de recomendação, das quais Cazella (2006) descreve duas:

- **Direito de Resposta**: permite que os usuários (compradores e vendedores) avaliem seus negociantes a partir do grau de satisfação da transação, o que gera uma pontuação que demonstra a confiabilidade no negociante, ou seja, sua reputação é construída por seus negociantes.
- **Comprador Pessoal**: o usuário pode escolher uma categoria de itens para receber recomendações.

2.3.3. Grupolens

É um sistema que auxilia pessoas a encontrar artigos acadêmicos, relacionando as avaliações sobre os artigos para determinar quais possuem vínculo, ou seja, uma *vizinhança*. Isso indica uma predição de interesses nos itens recomendados ao usuário em questão. Nesse processo ocorre o inverso, pois, a partir de um produto indicado a um usuário se descobre os outros usuários interessados, formando uma nova maneira de identificar a vizinhança (KONSTAN, 1997 apud CAZELLA, 2006).

2.4. DISPOSITIVOS MÓVEIS

A implementação do Motor de Recomendação pressupõe a facilidade de acesso às informações já direcionadas ao usuário, de forma simples e direta. Isso justifica seu foco em tecnologia móvel.

Os conceitos apresentados até aqui abordaram com bastante ênfase a facilitação do acesso às informações, possível em locais onde seja acessível a *Internet.*

Em uma publicação da *ComputerWorld*/EUA (2011), o tema *mobilidade* é apresentado como “Mobilidade Explode”, o que ratifica o objetivo proposto para este trabalho. Nessa publicação, os Dispositivos Móveis são considerados substitutos dos computadores de mesa em mídia, armazenamento de dados e acesso à informação.

Loureiro (2003) apresenta a computação móvel como uma área madura e possível candidata ao paradigma do futuro da computação. O autor afirma que a explosão da computação móvel faz parte de uma evolução natural, abaixo comprovada, conforme figura 4 dos paradigmas computacionais.

	Grupo	Tempo Compartilhado	Computador de Mesa	Rede
Década	1960	1970	1980	1990
Tecnologia	Escala Média de Integração	Escala Alta de Integração	Escala muito Alta de Integração	Escala Altíssima de Integração
Local	Sala do Computador	Sala do Terminal	Computador de Mesa	Móvel
Usuários	*Experts*	Especialistas	Individual	Grupo
Dados	Alfanumérico	Textos	Fontes e gráficos	*Script* e Voz
Objetivo	Calcular	Acesso	Exibir	Comunicar
Interconexão	Periféricos	Terminais	Computadores de Mesa	*Palmtops*
Aplicações	Customizada	Padrão	Genérico	Componente
Linguagens	Cobol, Fortran	PL/I, Basic	C, Pascal	Orientação ao Objeto

Figura 4: Evolução dos paradigmas computacionais (LOUREIRO, 2003 apud TESLER, 1991)

De acordo com os atuais conceitos de computação móvel e acesso à rede sem fio (*wireless*), os termos dominantes são Dispositivos Portáteis ou Móveis, *PDA's* (*Palm* e *Pocket PC*), celulares e *smartphones* em geral (PALUDO, 2003).

Os dispositivos móveis, antes artigos de luxo utilizados para realizar chamadas, enviar mensagens textuais ou consultar as horas, constituíram-se cada vez mais como recursos indispensáveis no cotidiano das pessoas, menores e mais portáveis.

Assim, as tecnologias móveis estão sendo incorporadas de forma ubíqua e em rede, permitindo interações sociais relevantes, sensíveis ao contexto e possibilitando conectividade com a *Internet* (NAISMITH, 2004 apud MEIRELLES et al, 2005). Porém, o usuário quer operar seu *PDA* da mesma forma que seu computador pessoal, sem levar em consideração suas diferenças.

Segundo Loureiro et al (2003) existe uma diferença entre o *PDA*, sempre ligado e junto ao usuário e o *notebook,* utilizado apenas quando necessário demandando mais tempo e não sendo prático.

O principal objetivo do desenvolvimento de aplicações voltadas a Dispositivos Móveis é o investimento em usabilidade, que permite as mesmas funcionalidades de um *notebook*, agora de forma reduzida e mais prática.

Loureiro (LOUREIRO et al., 2003) justifica o investimento em usabilidade afirmando que os pontos fracos dos dispositivos móveis são principalmente o processamento simultâneo e o espaço em disco. A essas restrições, soma-se o tamanho da tela e a grande quantidade de informações geradas em uma pesquisa, o que demanda um descarte das informações menos relevantes. Esse autor ainda afirma que os *notebooks* são tratados hoje em dia da mesma forma que os computadores pessoais, permanentemente conectados a uma rede e à energia. Essa troca de categoria dos notebooks altera a abrangência dos dispositivos móveis que passam a englobar *PDAs* e semelhantes. Por isso, os *softwares* também evoluíram para essas plataformas.

Para acompanhar essa evolução tecnológica e o crescimento da necessidade de acesso às informações a partir de qualquer lugar, os Dispositivos Móveis têm sido considerados boas fontes de informação.

A união entre Sistemas de Recomendação e Dispositivos Móveis é uma necessidade ascendente, pois, cada vez mais as pessoas dependem de uma informação instantânea. Através dessa união, o usuário pode acessar não apenas o

que é interessante, mas o que é potencialmente interessante, sendo considerado um meio interferente entre a informação e o usuário. Dessa forma, o descarte inicial do potencial que é menos relevante é evitado possibilitando o uso de informações complementares.

3. O MOTOR DE RECOMENDAÇÃO *MAÎTRE*

Conforme enunciado, o número de dispositivos móveis está em uma linha crescente e, cada vez mais, participam do cotidiano das pessoas. Além disso, o uso da *Internet* tornou-se comum nesses dispositivos móveis, consequentemente influenciando o número de fontes de informações disponíveis.

Todavia, o crescimento do número de fontes de informação não tem uma relação direta com a qualidade da informação encontrada. Pelo contrário, é provável que esse crescimento acarrete em sua dispersão. Por esse motivo, um Sistema de Recomendação pode auxiliar na qualificação das informações que são mais relevantes ao usuário.

Esses fatores contribuem para a necessidade de criação de um ambiente no qual Dispositivos Móveis possam utilizar um mecanismo de obtenção da informação desejada, ou seja, uma recomendação. Porém, apesar de sua facilidade de portabilidade e do aumento de suas capacidades, tais dispositivos ainda não possuem o mesmo poder de processamento e capacidade de armazenamento necessários a um sistema complexo de cálculos exigidos durante uma recomendação.

Entretanto, existem maneiras de suprir essa dificuldade dos dispositivos móveis em recomendar. Uma delas é prover um conjunto de técnicas que compõem um Motor de Recomendação capaz de prover um serviço a ser acessado diretamente pelas aplicações consumidoras desenvolvidas para dispositivos móveis, como é o caso do *Maître*.

É importante esclarecer que este Motor de Recomendação não caracteriza um Sistema de Recomendação em si, e sim, uma plataforma que provê às aplicações desenvolvidas para dispositivos móveis um meio para que possam usufruir de recomendações.

3.1 CARACTERÍSTICAS DO *MAÎTRE*

Esse Motor de Recomendação tem como sua principal característica permitir que aplicações consumidoras desenvolvidas para Dispositivos Móveis usufruam dos

benefícios de recomendações sem que possuam todos os recursos necessários para qualificação desta informação.

O *Maître* possui como referência a busca de informações através da entrada de palavras-chave que permitem visualizar todos os itens encontrados simultaneamente. Existe uma classificação previa das informações conforme os algoritmos de classificação do *Maître*. Nativamente, o *Maître* possui os algoritmos de Conhecimento, Rede Social, Perfil de Usuário e Histórico do Usuário. Adicionalmente, existe a funcionalidade para a visualização de um item da busca, sendo possível acessá-lo e/ou classificá-lo como informação útil.

Na medida em que o *Maître* é utilizado por diversas aplicações desenvolvidas para plataforma móvel, estas passam a ser denominadas *aplicações consumidoras*. Essas aplicações podem estabelecer um canal de comunicação com o Motor de Recomendação através de uma *Application Programming Interface* (*API*) que possui nativamente implementado um canal de comunicação baseado em serviços (*Web Services*).

Essa *API* possui um meio de comunicação pré-estabelecido, porém, se necessário, pode ser acrescida de novos meios de comunicação. Para isso é necessário que as implementações respeitem as *interfaces* e contratos pré-estabelecidos também disponíveis pela própria *API*. Esta implementação poderá ser acoplada na *API* e passará a funcionar assim que configurada.

Todos os serviços disponibilizados pela *API* possuem um método correspondente em uma fachada, que constitui a porta de entrada de todas as solicitações feitas. Além disso, essa fachada também inicia todo o processo de construção do motor de recomendação e garante o acesso somente a usuários autorizados às recomendações solicitadas.

A responsabilidade de receber essas solicitações é de um controlador (*controller*) que inicia o processo de recomendação através de uma ignição interna do Motor de Recomendação, a *Ignition*. Esta possui dois elementos fundamentais: um *motor de classificação* e um *motor de busca*. Ambos são acoplados à ignição através de adaptadores (*adapters*).

O motor de classificação pode ser configurado e possui a capacidade de construir os algoritmos de classificação, além de fornecer uma listagem completa dos algoritmos do próprio motor de classificação.

Com a evolução do Motor de Recomendação, novos algoritmos podem surgir, ou ainda, algoritmos existentes podem passar a necessários, como o demográfico, por exemplo. Para isso, o Maître conta com a possibilidade do incremento de novos algoritmos. No entanto, a implementação de qualquer algoritmo, independentemente de como for feita, deve respeitar os contratos pré-estabelecidos. Assim, o motor de classificação pode ser configurado novamente e atualizado passando a estender a gama de classificações a serem aplicadas.

Durante a aplicação dos algoritmos, também é possível determinar o grau de influência que cada algoritmo terá nos itens a serem classificados. Isso possibilita uma maior acuracidade nas informações obtidas.

Todos os algoritmos trabalham sobre uma *matriz de classificação*, construída com base nos algoritmos configurados e, depois de criada, o motor de classificação passará a ter a capacidade de fornecer as palavras-chave necessárias ao motor de busca.

O motor de busca, por sua vez, é responsável por obter informações através de palavras-chave. Nativamente, o *Maître* possui já implementado o motor de buscas do *Google* que é acessado através da *API* pública disponível no *site* da empresa. Esse motor de buscas do *Google* é o mesmo utilizado na interface disponível em seu site de buscas.

O *Maître* também possibilita que outro motor de busca, além do nativo, seja utilizado. A implementação desse novo motor deve respeitar os padrões estabelecidos e este deverá ser configurado para que a comunicação entre os motores de recomendação e busca obtenha sucesso.

Para garantir a segurança, está disponível na estrutura do *Maître* um sistema de registros e consultas de *log* e uso de autorizações para a concretização de operações junto à base de dados.

3.2 ARQUITETURA

A figura a seguir mostra a arquitetura lógica do *Maître* disposta em camadas:

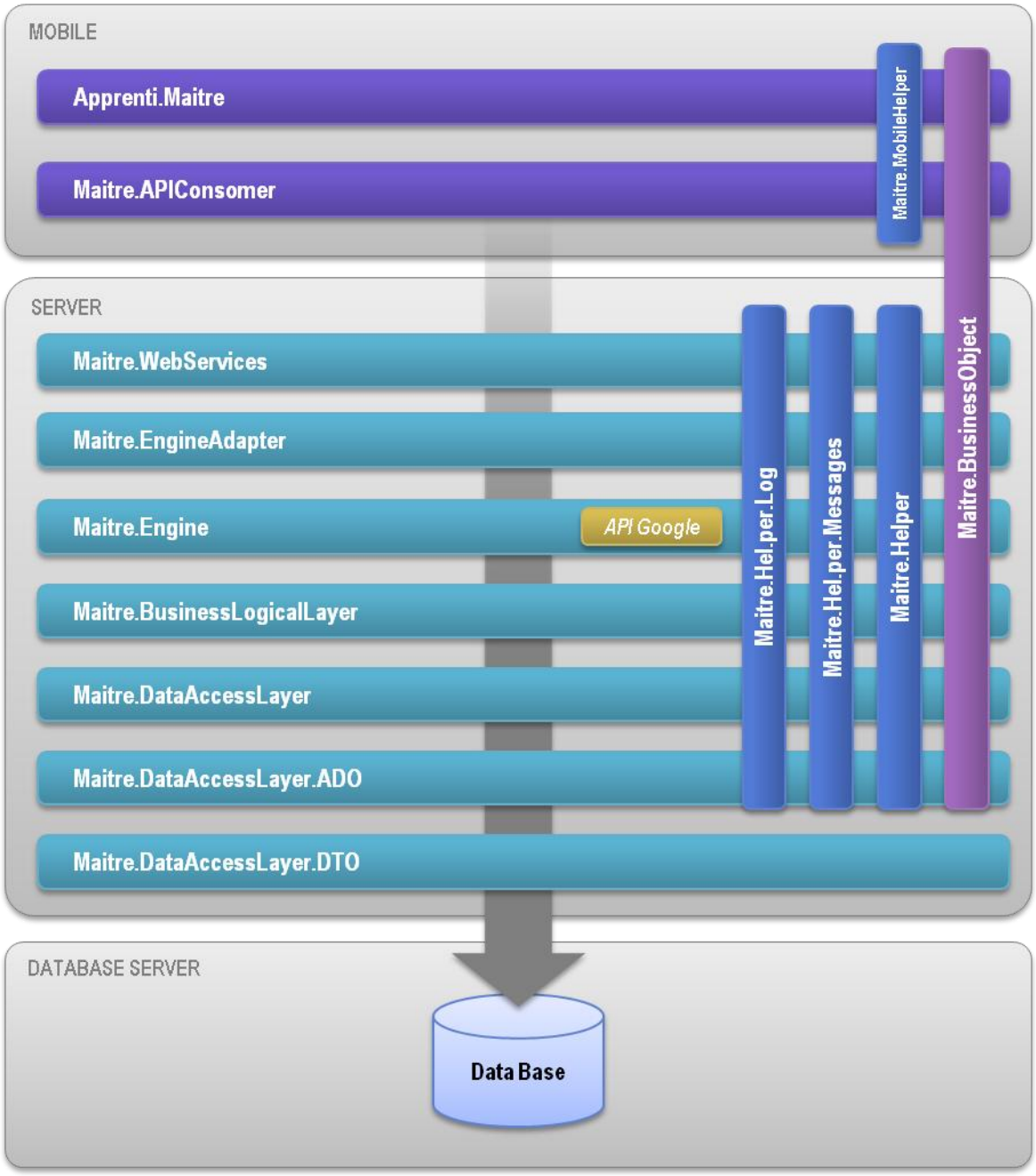


Figura 5: Arquitetura *Maître*

A arquitetura é composta por camadas interligadas, onde todas elas possibilitam a redução do acoplamento e um desenvolvimento totalmente independente. Estas são agrupadas conforme sua distribuição entre Dispositivo Móvel, Servidor e Servidor de Base de Dados.

As camadas lógicas referentes ao aplicativo do Dispositivo Móvel estão fisicamente instaladas no próprio dispositivo, sendo essa implementação feita sobre a *API* do *Maître* e de inteira responsabilidade dos desenvolvedores que utilização o *Maître* como plataforma de desenvolvimento.

Já a camada do Servidor contém o Motor de Recomendação e todos os componentes necessários para que ela funcione corretamente. Esta camada é responsável também pelas camadas de persistência, auxiliares, objetos de negócio e transversais.

Diferentemente dos conceitos apresentados até agora sobre Sistema de Recomendação, a representatividade deste para a arquitetura do *Maître* é diferente. Internamente, um Sistema de Recomendação representa um objeto que conterá uma estrutura de dados e um agrupamento para as configurações feitas. No caso em que as configurações iniciais não forem alteradas, o Sistema de Recomendação terá atrelado a ele um motor de busca do *Google* e os algoritmos de classificação de Rede Social, Perfil, Histórico e Conhecimento. Adicionalmente, o Sistema de Recomendação possui um objeto para auxílio na filtragem de dados e retorno do contexto de busca, ambos explicados posteriormente no capítulo Projeto.

- **Camadas do Dispositivo Móvel**

As camadas do Dispositivo Móvel estão localizadas fisicamente no Dispositivo Móvel e utilizam plataformas específicas para implementação voltada aos Dispositivos Móveis.

Esta camada está divida em: *Apprenti*.*Maitre* e *APIConsumer*, sendo *Apprenti.Maitre* o pacote de responsabilidade do consumidor, ou seja, utiliza o pacote *APIConsumer* para efetuar as operações possíveis através da chamada da *API* dessa camada.

  - **Camada da Aplicação Consumidora (Apprenti.Maitre)**

Conjunto de classes responsável pela apresentação de todo o sistema para o usuário consumindo diretamente a *API* Consumidora para utilização do Motor de Recomendação.

Mesmo que o objetivo da aplicação seja utilizar os recursos internos do motor, esta camada é uma das fronteiras de todo o sistema, portanto irá receber informações do usuário e por meio de outros recursos de processamento, terá que transformar estas entradas em dados de saída interessantes e esperados pelo

usuário. Para efetivação desse transporte, a camada envia uma pesquisa configurada e recebe uma lista de recomendações.

- **Camada da API Consumidora (Maitre.APIConsomer)**

Conjunto de classes que possibilita a aplicação móvel utilizar os recursos do Motor de Recomendação. Nesta camada são feitas as chamadas para o Motor de Recomendação via *WebService*. Assim, todas as operações efetuadas nessa camada refletem a maneira de trabalhar e as características dos objetos que contém os resultados obtidos.

Esta camada utiliza a camada de objetos de negócio (*BOs - Business Objects*) para enviar de maneira padronizada as requisições ao motor e receber os dados para a aplicação final. Nesta camada ocorre a transformação de um dado local em um dado compartilhado, ou seja, o tipo do dado que está no *WebService* pode ser diferente, então as classes controladoras fazem a transformação e também a chamada dos métodos dispostos no Servidor.

- **Camada de Utilidades *Mobile***

Nesta camada estão disponibilizadas as classes de ajuda (*Help*) semelhantes às dispostas no Servidor, porém, com uma implementação voltada à característica móvel, utilizando as bibliotecas disponíveis para fazer o mesmo feito no Servidor e permitindo que seja mantido um padrão de transferência.

- **Camadas do Servidor**

As camadas localizadas no Servidor abrangem a lógica do Motor de Recomendação que é composta principalmente por dois pacotes: o adaptador do motor, responsável por adaptar a *API* consumidora e o motor em si, que contém a lógica de negócio das recomendações.

Além destas camadas, existe uma camada de conexão (Maitre.WebServices) e outras complementares: a camada de negócio, de persistência, de transporte de dados e de utilidades.

- **Camada de Serviços *Web* (Maitre.WebServices)**

É a camada de conexão entre o Dispositivo Móvel e o Servidor através da exposição dos serviços da camada de adaptação do motor (Maitre.EngineAdapter). Essa camada possui métodos equivalentes aos métodos da camada de adaptação do motor.

- **Camada do Adaptação do Motor (Maitre.EngineAdapter)**

É a camada responsável por prover um canal único de comunicação de todas as solicitações da *API* (*Maitre.WebServices*) para a camada do motor (*Engine*). Esse processo ocorre através da exibição de uma fachada (*ConsomerFacade*) em que todas as solicitações devem ser obrigatoriamente autorizadas.

- **Camada do Motor (Maitre.Engine)**

Todas as solicitações feitas pelo adaptador ao motor tem um e apenas um método correspondente no controlador de solicitações, chamado *SolicitationsController*. Este controlador possui um agrupamento que representa um Sistema de Recomendação específico. Além disso, possui as funcionalidades correspondentes a *API* e uma ignição chamada *Ignition*.

Para prover as funcionalidades de marcação de um item como já visitado ou determinado ranqueamento (“Gostei” ou “Não gostei”), a ignição utiliza diretamente a *camada de lógica de negócio* (*Maitre.BusinessLogicalLayer*).

Já para executar as operações de busca de informações, são necessários os motores de classificação e busca. O motor de classificação é chamado de *ClassificationEngine* e o de busca *SearchEngine*. Ambos os motores são acoplados à ignição através de adaptadores (*adapters*) que criam novas instâncias dos motores e as adaptam à *Ignição*.

O processo de funcionamento do adaptador do motor de classificação (*ClassificationEngineAdapter*), quando solicitada a adaptação, busca na base de dados, através da camada de lógica de negócio, os algoritmos que aquele Sistema de Recomendação possui. Após, são criadas instâncias desses algoritmos através

de uma fábrica (*ClassificationEngineFactory*) que utiliza reflexão para fazer a instanciação. Isso só é possível porque todos os algoritmos implementam a mesma classe abstrata *ClassificationBase*. Todos os algoritmos criados são inseridos no motor de classificação (*ClassificationEngine*) que será capaz de processar cada um deles.

Além disso, o motor de classificação possui uma matriz (*ClassificationMatrix*) onde são processadas todas as classificações feitas pelos algoritmos.

O motor de busca (*SearchEngine*) é criado por um processo semelhante. Porém, o adaptador (*SearchEngineAdapter*) busca na base de dados um único algoritmo para busca de informações referente aquele Sistema de Recomendação. Após buscar a informação na base de dados, o adaptador solicitará a uma fábrica de criação de motores de busca (*SearchEngineFactory*) a instância do motor em questão. A fábrica, por sua vez, retornará a instância solicitada através de reflexão. O algoritmo deve implementar a classe abstrata base *SearchEngineBase*.

Após a ignição ter seus motores de busca e classificação inicializados, ela está apta a classificar informações.

O diagrama apresentado na figura 6 demonstra um exemplo de uso do motor, no qual o usuário necessita uma recomendação.

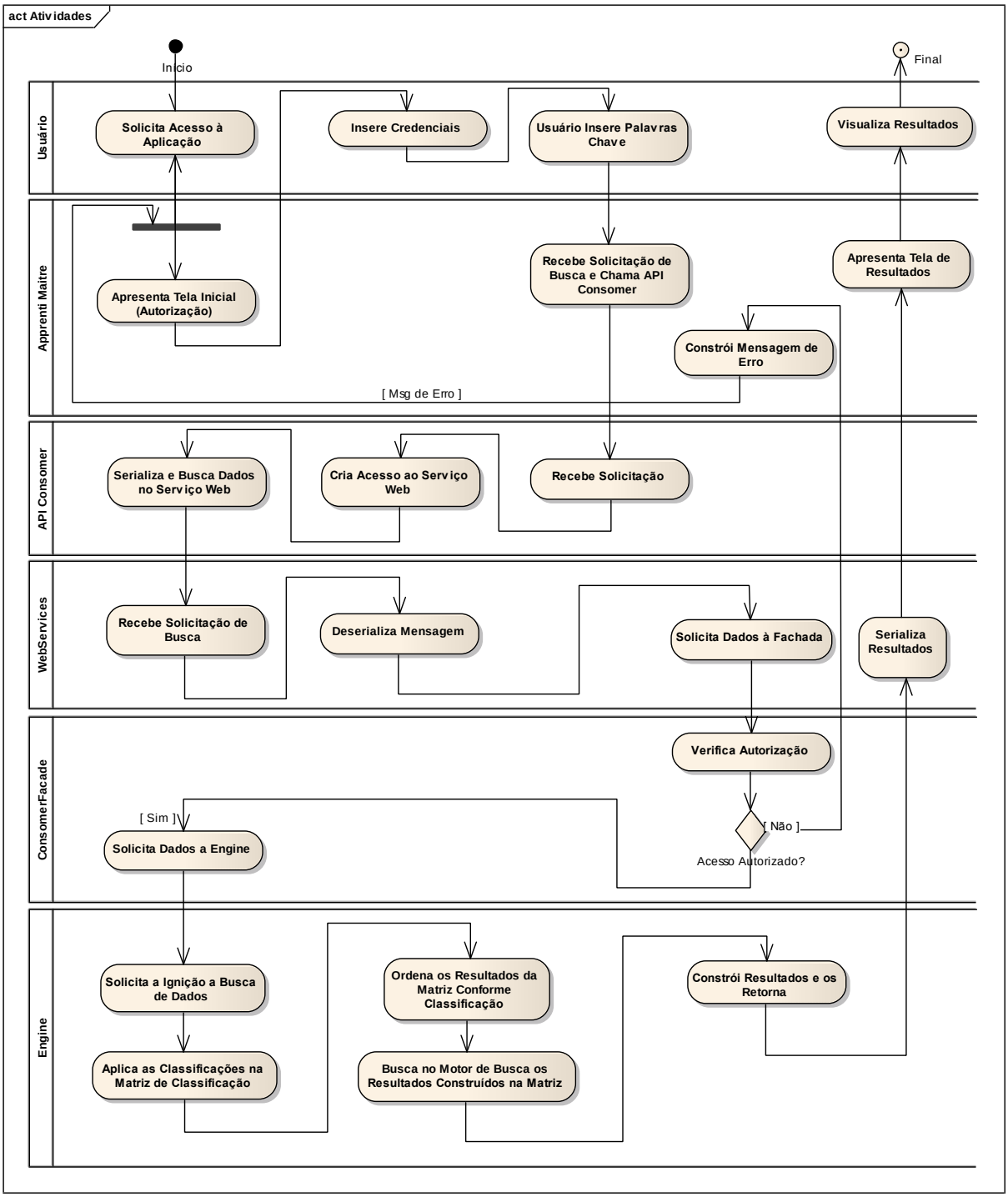


Figura 6: Exemplo de utilização do motor

- **Camada de Lógica de Negócio (*Maitre.BusinessLogicalLayer*)**

Essa camada tem como responsabilidade manter a lógica de negócio a ser implementada. Além disso, deve garantir as chamadas das *interfaces* da camada de persistência.

- **Camada de Persistência (*Maitre.DataAccessLayer, Maitre.DataAccessLayer.ADO e Maitre.DataAccessLayer.DTO*)**

A persistência do *Maître* é garantida pelos componentes da camada de persistência, cujo papel é gerenciar os dados e conexões para que o motor mantenha uma boa *performance* e não comprometa o processamento ao realizar uma chamada ao banco de dados.

A responsabilidade dessa camada é retornar dados através de requisições e persistir os dados enviados pelas demais camadas através de um *Data Transport Object* (*DTO)*, ou seja, um objeto de transporte de dados. A classe que for persistir um dado fica responsável por tratar uma inserção ou uma alteração, mapeando os dados passados por parâmetros. Toda a classe trabalha com um *DTO* recebe por parâmetro ou retorna um objeto de negócio (*BOs – BusinessObject*). Estes trafegam verticalmente pela arquitetura do *Maître*. O acesso ao banco é feito pelo gerenciador de conexões do *Maître* e *DTOs* que permitem ao banco de dados ter suas tabelas mapeadas. Porém, a leitura de um dado por um *DTO* deve gerar uma instância de um *BO* ou um tipo primitivo (*string*, *int*, *Double*, etc). Para cada *DTO*, existe um *BO* correspondente para que todas as camadas possam enxergá-lo.

Essa criação deve ocorrer sempre em uma coleção ou objeto tipado, ou seja, nomenclatura e características dos dados para garantir que, em qualquer camada onde sejam utilizadas, possam receber o mesmo tratamento.

- **Camada de Transporte de Dados**

Os *BOs* têm como responsabilidade viabilizar o tráfego dos dados utilizados pelo sistema através das camadas onde há a padronização da utilização de métodos e a garantia da segurança dos dados.

Um *BO* facilita a comunicação entre as classes, pois um objeto criado na *interface* pode ser persistido no banco de dados mantendo as mesmas características sem a necessidade de transformação de um objeto em outro, apenas mapeando suas propriedades.

Nesta camada, também são utilizadas classes do tipo fachada para expor apenas algumas propriedades dos *DTOs (estes são os BOs)*, sendo chamadas de estrutura de dados (ED). Estas EDs são classes que permitem a exibição de dados

aos consumidores sem regra alguma e servem para ocultar algumas informações que os *DTOs* possuem, como, por exemplo os relacionamentos.

Nos *DTOs* é permitido navegar entre os relacionamentos do modelo de dados por uma propriedade que o *DTO* guarda, fazendo referência às chaves estrangeiras da tabela que o mesmo representa.

Os *DTOs* são as classes que permitem a leitura e escrita no banco de dados, ou seja, quando se está fazendo um comando *select* através da camada de persistência, no caso *Linq*, o acesso é feito via *DTO*. Na persistencia, um *DTO* deve manter-se conectado ao banco de dados para ser atualizado, ou seja, caso um objeto seja exposto às camadas superiores, deverá ser desconectado do banco de dados, pois apenas o que foi selecionado pode ser lido.

Para garantir a *performance*, assim que o *DTO* ou uma lista dele é instanciada, o gerenciador de conexões deve fechar a conexão. Sendo assim, quando o objetivo da persistência for uma alteração, o *DTO* deve ser consultado antes para atualizar os dados e, por fim, enviá-los para o banco de dados, pois o mesmo objeto já é conhecido pelo banco de dados e possui um endereço nele.

Na persistência, o *Maître* trabalha da seguinte maneira: existe um projeto onde está colocada a estrutura de classes que faz o contato com o banco de dados, independente de qual banco de dados, sendo que nele, só existem interfaces. Há um outro projeto que conhece a linguagem do banco de dados e possui as chamadas ao mesmo e implementa as interfaces citadas anteriormente. Sendo assim, para efetuar a troca de banco de dados é necessário apenas reimplementar um projeto e revisar as chamadas que o consumidor destes dois projetos faz. Resumindo, temos um projeto que é conhecido pelas camadas de cima e cria objetos referenciando *interfaces* que acessam o banco de dados (sempre na mesma estrutura).

- **Camada de Utilidades**

A camada de utilidades possui camadas internas: a de *log*, para que sejam guardados registros, a camada de mensagens, para transporte de mensagens, uma camada de tratamento de exceções, para tratar erros e uma camada auxiliar para auxílios em transformação de dados.

- **Camada de *Log***

Essa camada tem como objetivo armazenar, de maneira transparente para todas as camadas, os registros de *log*. Todos os registros devem ser inseridos sem a necessidade de qualquer configuração ou implementação.

- **Camada de Mensagens**

Para que haja uma coerência entre todas as mensagens apresentadas pelo *Maître*, é disponibilizado um conjunto de mensagens padrão para garantir, inclusive, a possibilidade futura de internacionalização ou globalização do *Maître*.

- **Camada de Tratamento de Exceções**

Todos os erros devem ser tratados por essa camada, que fará com que, em um único ponto central, os erros tenham mensagens padronizadas e tratamentos igualmente padronizados.

- **Camada de Transformação de Dados**

Essa camada é responsável por serializar, deserializar, compactar e descompactar as *strings* utilizadas pelo *Maître*.

- **Camada de Objetos de Negócio (*Maitre.BusinessObject*)**

Todos os objetos de negócio são trafegados verticalmente entre as camadas da arquitetura do *Maître*. Os objetos refletem, em grande maioria, os objetos da base de dados.

4. O PROJETO

O projeto de construção do *Maître* está dividido conforme a distribuição de suas camadas: distribuídas no Dispositivo Móvel e distribuídas no Servidor. Para facilitar a compreensão, cada camada possui uma apresentação descritiva da implementação do projeto e, nas camadas mais relevantes, diagramas de classes, sequência e/ou atividade.

- **Camadas do Dispositivo Móvel**

As camadas localizadas no Dispositivo Móvel têm como principal objetivo demonstrar o uso do Motor de Recomendação através da implementação de um piloto que utiliza efetivamente a camada da *API* consumidora, responsável pela comunicação com o *Maître*.

- **Camada da Aplicação Consumidora (*Apprenti.Maitre)***

A camada da aplicação consumidora, no contexto do projeto *Maître*, programa o *Apprenti Maître*. Esta aplicação é piloto para provar o conceito da arquitetura e está desenvolvida utilizando a plataforma *Microsoft .NET Compact Framework*, na linguagem *C#* para Dispositivos Móveis.

O sistema consumidor possui entre suas configurações um código identificador que é verificado junto ao código de identificação cadastrado no banco de dados do *Maître* para aquele sistema de recomendação, buscando garantir a segurança de acesso aos métodos.

- **Camada da API Consumidora (*Maitre.APIConsomer)***

O principal objetivo desta camada é tornar viável o desenvolvimento de aplicações móveis utilizando os serviços existentes no *Maître* independentemente de onde eles estejam.

Há um canal de comunicação (Fachada) responsável por receber todas as requisições e um controlador de mensagens (*Broker*) que faz a comunicação com o Motor de Recomendação. Sendo assim, a aplicação consumidora faz a chamada dos métodos da *API* através de uma Fachada (*MobileConsomerFacade*) que irá delegar para o *Broker* (*MobileConsumerBroker*) a chamada e será responsável pela comunicação com o *Maître*, ou seja, apenas a fachada deve estar visível aos consumidores, com isso todas as classes restantes são de uso interno da camada.

A conexão com o motor, feita pelo *Broker,* é transparente para a aplicação. Dessa forma é indiferente o meio de acesso, podendo ser local, uma chamada a *WebService*, uma conexão remota, uma requisição *Web*, dentre outros. Essa conexão é feita através de um *WebService* que possui sua *URL* registrada nas configurações da *API*.

Como meio de garantir a segurança desta comunicação, os métodos de comunicação possuem uma chave que é conhecida no servidor e na *API*, portanto, faz parte de todas as chamadas.

As classes desta camada são exibidas no diagrama apresentado na figura 7.

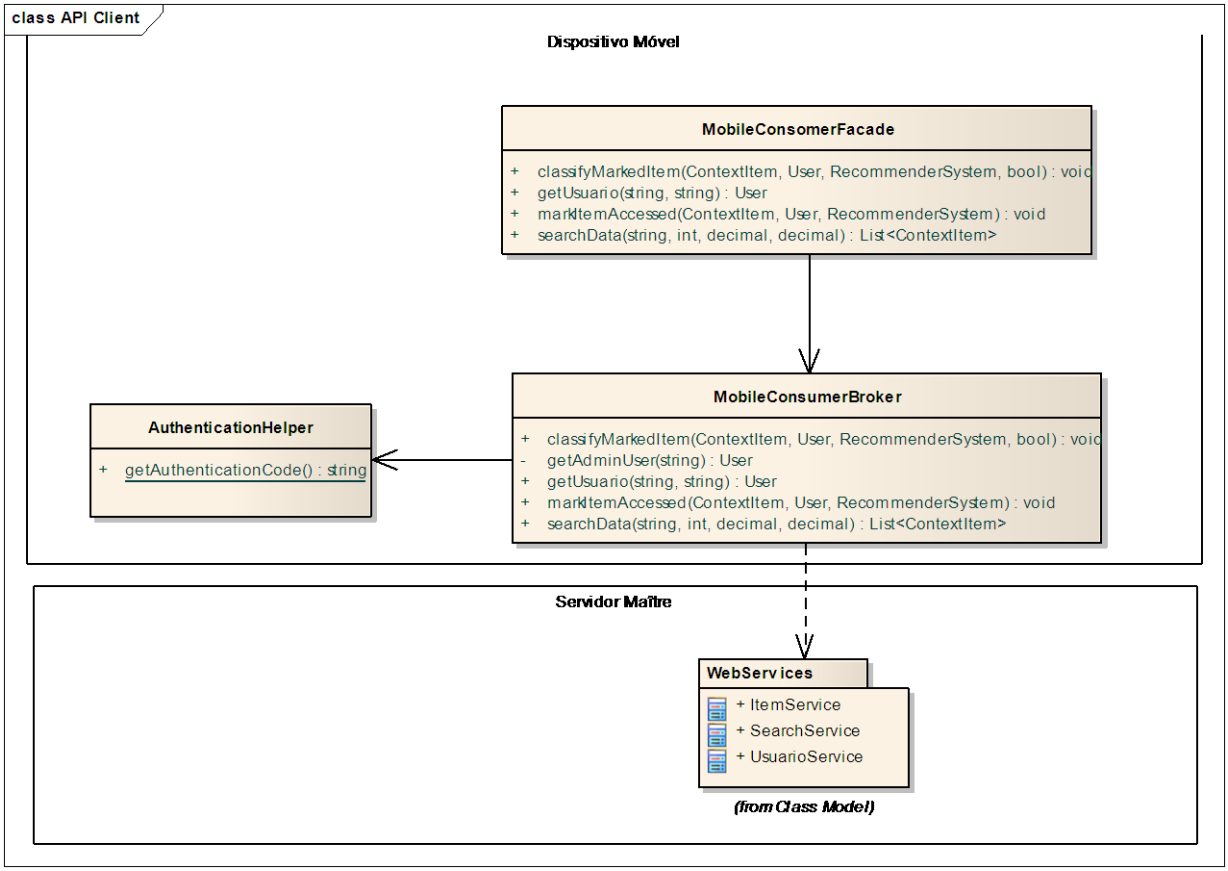


Figura 7: Diagrama de Classes da *API* Consumidora

- **Camada utilitária *Mobile* (*Maitre.MobileHelper)***

Esta camada tem como objetivo auxiliar as demais camadas localizadas no Dispositivo Móvel a executar diversas operações genéricas e utilitárias. Dentre essas operações está a serialização e deserialização de objetos e a compactação e descompactação de *strings*. A camada *Helper* não tem vínculo com o restante do sistema por ser acessada estaticamente em qualquer camada do dispositivo móvel.

Essa camada possui uma classe chamada *MobileTextHelper* que é responsável pela criptografia e serialização de objetos, contando também com os processos inversos. Esta classe é fundamental para o recebimento de dados trafegados pelas camadas entre Cliente e Servidor do *Maître*.

A utilização destas classes é estática. A camada de utilidades do Dispositivo Móvel é uma camada auxiliar, portanto não conhece nenhuma outra camada do sistema, apenas fornece serviços.

A figura 8 define a classe *MobileTextHelper.*

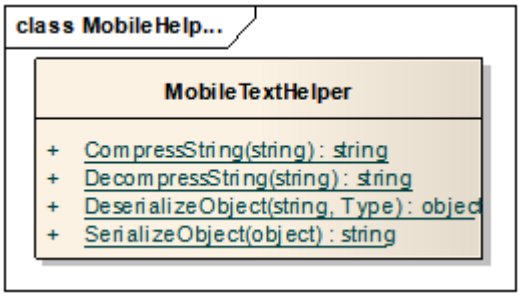


Figura 8: Diagrama da Classe *MobileTextHelper*

A figura 9 ilustra a camada de Dispositivos Móveis.

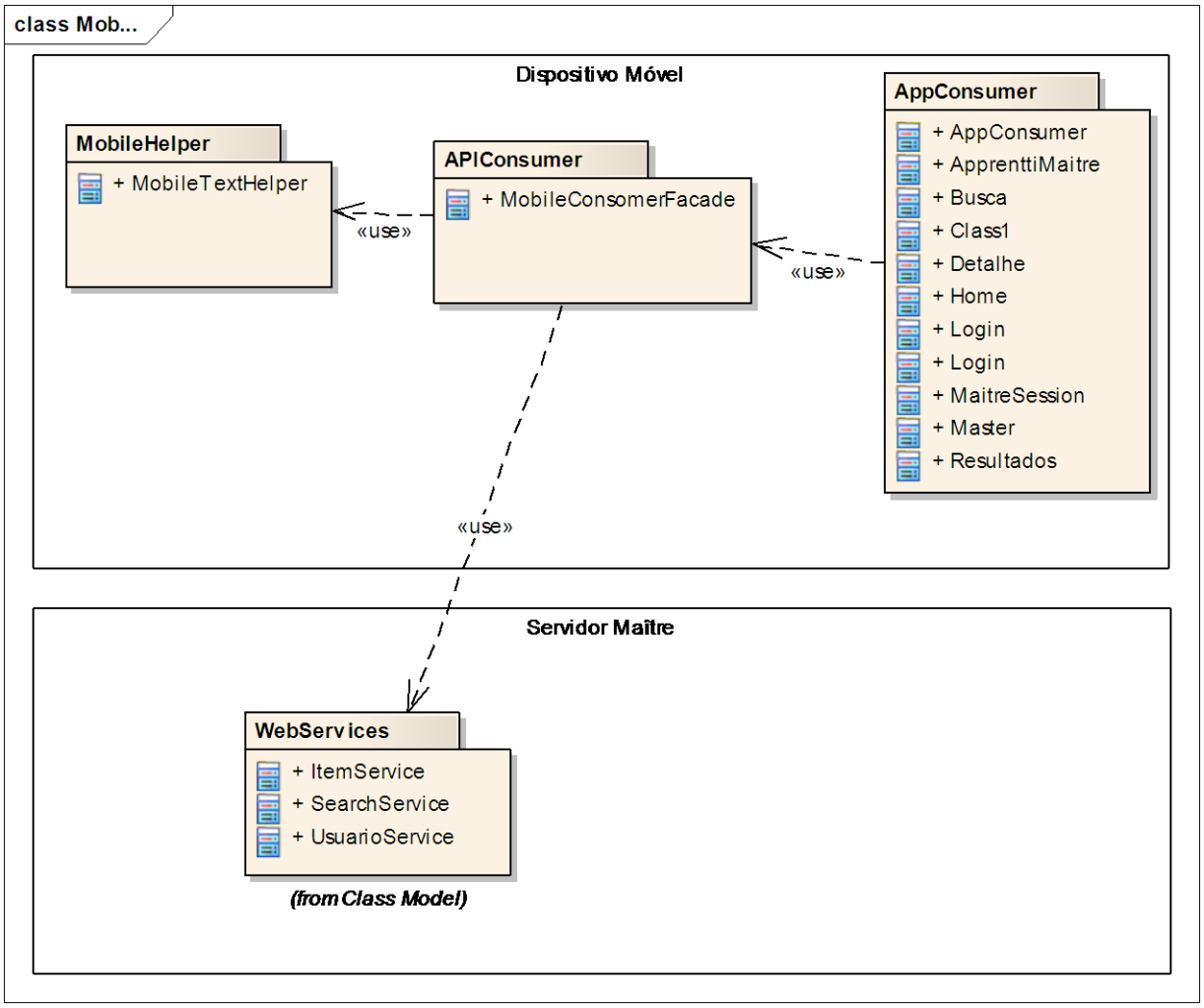


Figura 9: Diagrama de Classes da camada móvel

A figura 10 ilustra o processo de Acesso a um item através do Dispositivo Móvel, pontuando o mesmo no motor de recomendação localizado no Servidor *Maître*, em um diagrama de sequência.

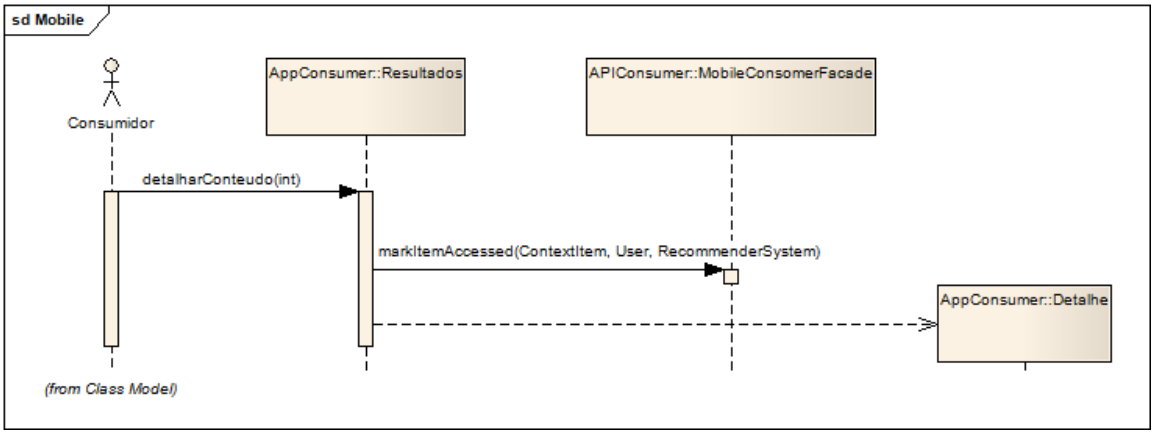


Figura 10: Diagrama de sequência do *ApprentiMaître* para acessar um item.

- **Camadas do Servidor**

O conjunto de camadas que possui toda a lógica do Motor de Recomendação está localizado em um Servidor com capacidade suficiente para prover um processamento e armazenamento de dados necessários ao Sistema de Recomendação. As seguintes camadas estão englobadas na camada do servidor:

- **Camada de Serviços *Web* – *Maitre.WebServices***

Os serviços *Web* do projeto *Maître* servem como *interface* entre o motor e o dispositivo móvel. Esses serviços são métodos e estão organizados conforme o negócio, nesse caso referentes a itens (*ItemService*), busca (*SearchService*) e usuário (*UserService*).

- **Camada do Adaptador do Motor – *Maitre.EngineAdapter***

A camada adaptadora deve fornecer um único canal de comunicação ao Motor de Recomendação. Através dela, todas as solicitações são recebidas pela *façade ConsomerFacade* e são processadas na medida em que correspondem a um determinado Sistema de Recomendação.

Cada solicitação é feita ao controlador *SolicitationsController e*, se autorizada, retorna o resultado da operação solicitada.

Para garantir uma maior segurança, as solicitações devem ser autorizadas por uma chamada estática ao método *CheckAuthorization* da classe *Authorization* que verifica a autorização de um determinado usuário a um Sistema de Recomendação.

Ao detalhar melhor o diagrama de classes (vide figura 11 - diagrama de classes da camada *Maitre.EngineAdapter*), percebe-se as classes estruturadas.

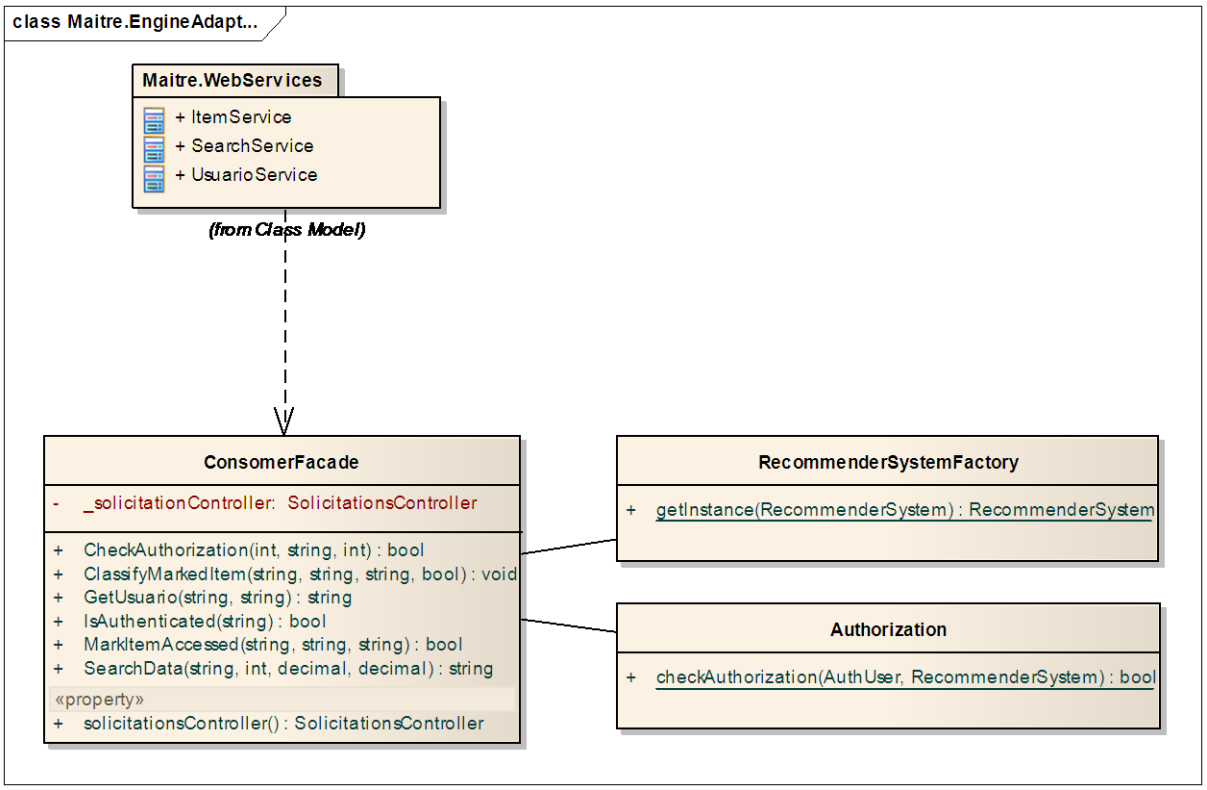


Figura 11: Diagrama de Classes Camada *Maitre.EngineAdapter*

***ConsumerFacade*:** é uma fachada, responsável por garantir o recebimento de todas as solicitações da camada *API Consomer* e sua principal função é autenticar os usuários que solicitam operações. Todas as solicitações devem ser confirmadas somente se os usuários estão autorizados. Isso ocorre através do método *CheckAuthorization* que garante a autenticidade de um usuário através de seu nome e sua senha e o identificador do Sistema de Recomendação da solicitação.

Adicionalmente à verificação da autenticidade, a fachada também é responsável por codificar/serializar e decodificar/deserializar todas as *strings* que trafegam entre a fachada e a *API*, incrementando a segurança das operações.

Essa classe possui também uma instância do controlador *SolicitationsController* da camada *Engine* estabelecendo um canal direto de comunicação entre essas camadas. Esta instância é responsável por processar as solicitações sendo que todos os métodos existentes na fachada têm um método correspondente no controlador.

***RecommenderSystemFactory*:** para que uma operação seja processada, é necessário que o Sistema de Recomendação, seja criado pelo código identificador

recebido. Assim, o Sistema de Recomendação deverá ser criado através da fábrica *RecommenderSystemFactory*, após a busca na base de dados criando uma instância única desse motor.

***Authorization*:** responsável por garantir a autenticação somente de usuários autorizados aos Sistemas de Recomendação. O controle de autenticação é feito pelo método *CheckAuthorization* e mantido em sessão evitando novas autorizações a cada transação.

- **Motor – *Maitre.Engine***

Todas as solicitações feitas ao Motor de Recomendação passam, necessariamente pelo controlador de solicitações *SolicitationsController.* Ele possui todos os métodos que o motor é capaz de processar funcionando como porta de entrada.

O controlador é capaz de marcar ou classificar itens acessados, determinar se um usuário é válido, buscar um Sistema de Recomendação e buscar uma recomendação, sendo esta sua principal funcionalidade.

Para todos os casos, exceto a solicitação de uma recomendação, o controlador irá diretamente à camada de lógica de negócio (*Maitre.BusinessLogicalLayer*) trazendo as informações solicitadas.

Já no caso de uma solicitação de recomendação, o controlador possui uma *ignição* (*Ignition*) para que os dados sejam controlados de maneira correta. Essa ignição somente será inicializada quando seu *status* estiver inicializado (*started)*.

A ignição possui dois motores, um de busca e um de classificação de informações, um Sistema de Recomendação ao qual serão buscadas as informações e um *status* para controle de seu estado.

Caso estado da ignição esteja já inicializado, a ignição é capaz de efetuar a busca. Porém, primeiro deve ocorrer a atribuição de um Sistema de Recomendação à ignição para que ela tenha condições de definir sobre qual sistema será o conhecimento obtido.

Além disso, a ignição deve fazer a inicialização dos motores através de adaptadores e ocorre em dois momentos. Em um primeiro momento, no caso do motor de busca, o adaptador será a classe *SearchEngineAdapter* que possui um

método estático que será executado para fazer a adaptação da ignição com o motor de busca.

O adaptador do motor de busca pesquisará, na camada de lógica de negócio (*Maitre.BusinessLogicLayer)* as informações de qual o motor de busca será adaptado conforme o Sistema de Recomendação determinado na ignição. Após buscar as informações na camada de negócio, a criação será feita por uma fábrica chamada *SearchEngineFactory* que cria a classe buscada na base de dados através de reflexão. Após isso, a ignição já está capacitada para buscar informações.

E um segundo momento, é adaptado um motor de classificação à ignição utilizando o adaptador *ClassificationEngineAdapter* que instancia um novo motor de classificação *ClassificationEngine* internamente na ignição e busca na base de dados, através da camada de lógica de negócio, quais algoritmos o Sistema de Recomendação utiliza. Após, é iniciado o processo de criação de cada um dos algoritmos através da fábrica de algoritmos *ClassificationAlghoritmFactory*. Internamente essa classe é capaz de transformar uma classe *ClassificationBase* em um algoritmo de classificação efetivamente.

Todos esse algoritmos criados herdam a mesma classe abstrata que é base para os algoritmo de classificação, chamada *ClassificationBase* que possui os métodos corretos para o processamento de um algoritmo de classificação.

Após a criação de todos os algoritmos o processo de inicialização a ignição está completo.

Nativamente, o *Maître* possui como motor de busca a *API* do *Google* e os algoritmos de Conhecimento, Perfil, Rede Social e Histórico de Usuário já implantados e configurados.

Posteriormente à criação do motor de classificação e busca, a ignição possui capacidade operacional e pode executar os processos para qualificação de informações.

O processo efetivo de qualificação da informação começa quando é acionado o método *SearchData* da ignição pela classe controladora *SolicitationsController*. Para isso, é necessário receber o usuário como meio de garantir a autenticidade. Além disso, é utilizado um objeto de negócio que representa o Sistema de Recomendação, a classe *RecommenderSystem*, para fazer todo o transporte de informações entre a ignição e os motores de busca e classificação.

O primeiro elemento acionado é o método *ApplyClassifications* que constrói uma matriz de classificação capaz de indicar mais palavras chave para a futura busca. Depois de aplicados todos os processamentos dos algoritmos e indicadas novas palavras chave, a ignição envia ao motor de busca *SearchEngine.* Este já objeto *RecommenderSystem* que já possui as novas palavras-chave indicadas pelo motor de classificação e serão pesquisadas pelo motor de busca. Os resultados são inseridos em uma listagem de resultados do tipo *ContextItem* que também está dentro do objeto *RecommenderSystem*.

Após o processo, a ignição retorna esse objeto que representa o Sistema de Recomendação para camada superior.

As classes do motor são as seguintes:

SolicitationsController*:* classe que controla todas as solicitações à camada *Maitre.Engine*. Ela possui acesso à camada de lógica do negócio para busca de informações e uma ignição para qualificação de informações.

Ignition**:** é uma ignição que sincroniza os motores de classificação e busca. Possui um estado (*IgnitionStatus)* para controle do estado e execução de tarefas. Além disso, a ignição ser inicializada para ser usada e buscar informações. Para garantir a sincronia, o método de busca de dados executa a classificação para obtenção da palavra-chave e, posteriormente a busca no motor de busca. Essa classe utiliza dois adaptadores utilizados de forma estática que Server para adaptar os motores de busca e classificação, chamados *adapters*.

IgnitionStatus**:** enumerador que representa os estados da ignição sendo: instanciada (0), inicializada (1), parada (2) e cancelada (3).

SearchEngineAdapter**:** é o adaptador que acopla a ignição ao motor de busca, pesquisando na base de dados, através da camada de negócio, que motor de busca o Sistema de Recomendação deve utilizar, enviando esse objeto para fábrica *SearchEngineFactory* que cria o objeto.

SearchEngineFactory**:** fábrica para criação do motor de busca. Durante a criação é instanciada a classe recebida com base em um objeto do tipo

SearchEngineBase através de reflexão. Nativamente, a classe que implementa a classe abstrata *SearchEngineBase* é a *SearchEngineGoogle*.

SearchEngineGoogle: implementa a classe abstrata *SearchEngineBase* sobrescrevendo o método *SearchData,* responsável por buscar as informações conforme as palavras-chave recebidas. Todos os resultados obtidos são inseridos em uma lista de objetos do tipo *SearchContext* que é retornada pelo próprio método.

***ClassificationEngineAdapter*:** adaptador do motor de classificação (*ClassificationEngine*) à ignição. Durante este processo de acoplamento, o adaptador busca na camada de negócio todos os algoritmos de classificação que devem ser aplicados ao Sistema de Recomendação e aciona a fábrica *ClassificationAlgorithmFactory,* que cria todos os algoritmos fisicamente.

ClassificationAlgorithmFactory: fábrica que instancia todas as classes de algoritmo a serem utilizadas. Estas representam os algoritmos, e devem ser herdadas da classe abstrata *ClassificationBase*. A fábrica instancia os algoritmos através de reflexão conforme atributo *className* recebido.

***ClassificationEngine*:** é a principal classe para qualificação de informações. Possui uma matriz de classificação do tipo *DataTable* e métodos para criação e ordenação da matriz e a aplicação dos algoritmos de classificação.

A construção da matriz de classificação ocorre acionando o método *GetKeyWordsFromAlgorithm* de cada um dos algoritmos para obter as palavras-chave que compõem a coluna zero da matriz, ou seja, une todas as palavras-chave de todos os algoritmos.

Já a aplicação dos algoritmos começa com a construção da matriz, onde é criada uma nova coluna para cada algoritmo. Após, é acionado o método *Process* de cada algoritmo que classifica as informações na coluna recém criada. Por fim, há uma reclassificação dos itens onde os itens melhores classificados são posicionados nas primeiras colocações. O número de palavras-chave utilizado é fornecido conforme a granularidade estipulada no Sistema de Recomendação.

Por fim, são inseridas as palavras-chave que obtiveram pontuação negativa, ou seja, serão excluídas pelo motor de busca.

Assim, o motor de busca está pronto para utilizar as palavras-chave já reclassificadas e reordenadas.

Complementando as informações é apresentado o diagrama de classes do motor conforme figura 12.

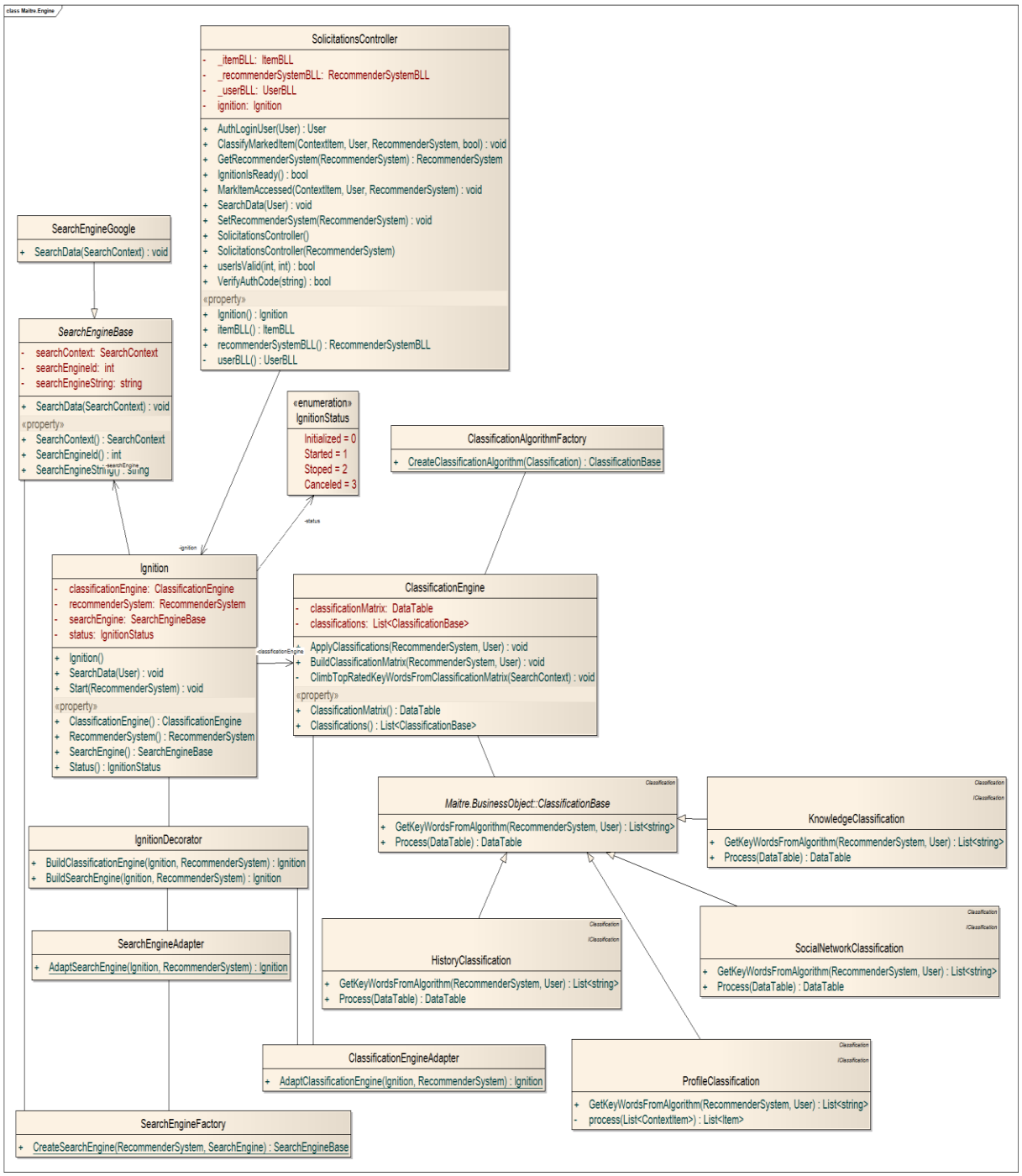


Figura 12: Diagrama de Classes do Motor

Figura 13:

- **Camada de Negócio – *Maitre.BusinessLogicalLayer***

A camada de negócios do Motor de Recomendação possibilita o desenvolvimento de um conjunto de regras de negócio no sistema. É de responsabilidade dessa direcionar o negócio. Na camada de negócios ocorre a chamada à camada de persistência.

A lógica não necessariamente deve ser vinculada ao banco de dados, podendo ser a chamada de uma biblioteca de serviços, a resolução um cálculo básico ou até mesmo a invocação de outro sistema. No Motor de Recomendação *Maître*, a principal funcionalidade da camada de negócios é atender as necessidades do motor e transformá-las em requisições ao banco de dados. Estas requisições estão melhor detalhadas na camada de persistência.

As lógicas direcionadas para o banco de dados devem ser feitas utilizando uma interface, pois a lógica de negócio é independente de qualquer tecnologia de banco de dados.

Apenas a camada do motor deve consumir serviços dessa camada, garantindo a centralização das rotinas e serviços. Este conceito pode ser visualizado no diagrama de classes da camada de negócios apresentado 13.

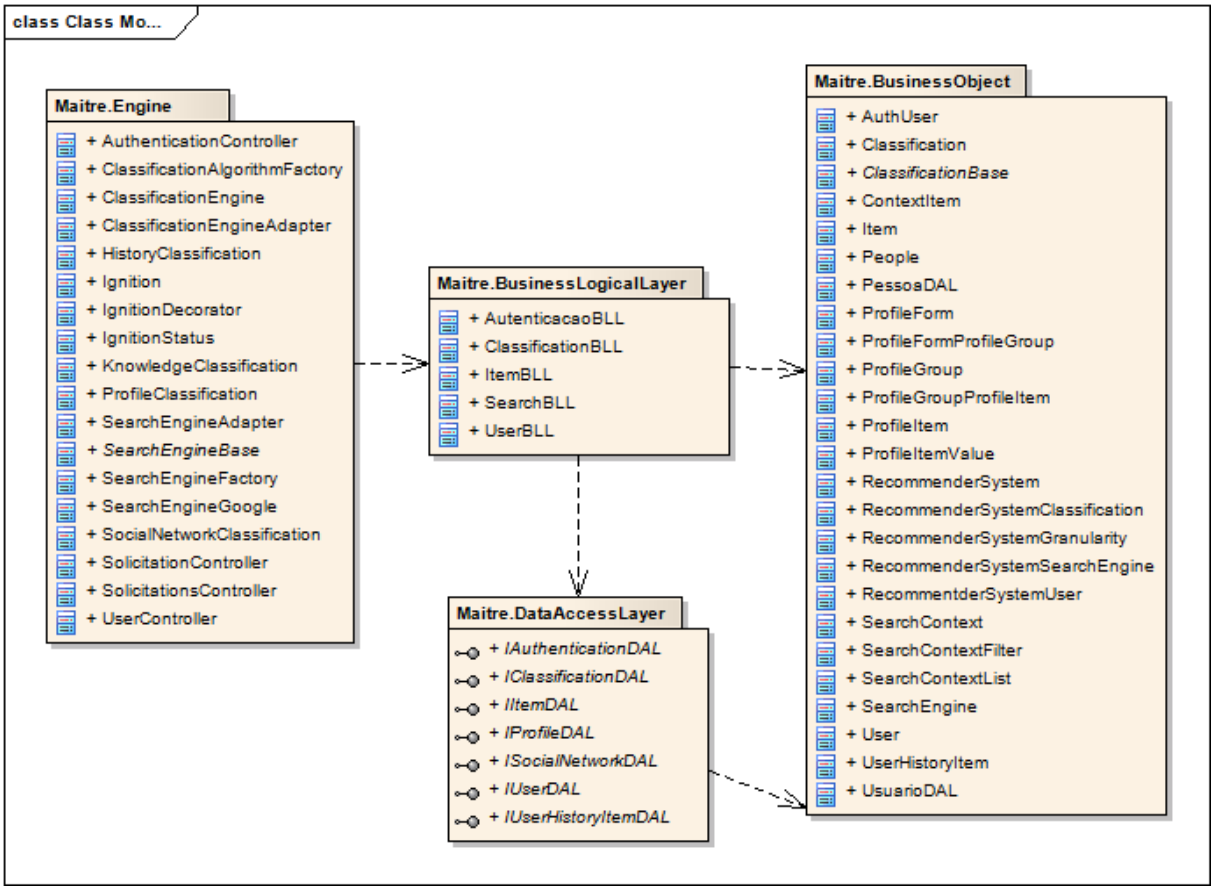


Figura 14: Diagrama de classes da camada de Negócios com suas fronteiras

- **Camada de Persistência**

Essa trata os dados e gerencia as conexões para que o sistema mantenha um bom desempenho de acesso a dados. A persistência do *Maître* é responsável por todas as camadas existentes entre a lógica de negócio e o banco de dados. Nessa camada ocorre a leitura e extração de dados a serem transportados via DTOs a serem transformados em objetos de negócio (*Maitre.BusinessObjects*).

A persistência do *Maître* possui uma série de interfaces para garantir a operabilidade do sistema, independentemente de qual plataforma de banco de dados for implementada. Esse objetos para acessarem um destes bancos de dados devem implementar obrigatoriamente as interfaces do Motor (*Maitre.DataAccessLayer*). Estes são criados como interfaces e depois instanciados como classes da camada que as implementa (*Maitre.DataAccessLayer.ADO*).

A responsabilidade desta camada é retornar dados através de requisições feitas pela camada superior e as persistir, focando principalmente na transformação

de dados em objetos de transporte. A camada está organizada e dividida da seguinte maneira:

- **Camada de Interfaces de Acesso a Dados (*Maitre.DataAccessLayer)***

Esta camada disponibiliza interfaces de métodos obrigatórios para o funcionamento do Motor de Recomendação, independentemente da camada que implementa o acesso ao banco de dados. Ao analisar a camada de persistência, esta faz a comunicação com o consumidor do banco de dados, a camada de negócios.

Para manter um padrão de comunicação entre as camadas de acesso a dados e a camada de negócio, toda requisição feita ao banco de dados deve ser feita através das *interfaces*.

- **Camada de Transporte de Dados (*Maitre.DataAccessLayer.DTO)***

A camada de transporte de dados faz a conexão direta com o banco de dados. Dentro desta estão mapeadas as classes que fazem esta conexão, sendo cada uma delas o espelho de uma tabela do modelo de dados. Apenas a camada *Maitre.DataAccessLayer.ADO* tem acesso aos objetos *DTO*, que necessitam estar conectados ao banco para permitir que diversas funcionalidades propostas pelo *framework* de persistência aqui utilizado (*LINQtoSQL*) sejam executadas.

A leitura de um dado da base é feita por uma instância de um dos *Contexts* (*MaitreDataContext* ou *MaitreSQLContext*) que dispõe os dados de determinada tabela em um dos objetos de transporte.

A transformação do dado em classe de transporte é feita por um procedimento armazenado ou por um *framework* de persistência como o *LinqToSql*, ambos previstos nesta arquitetura. Independentemente do processo escolhido pelo objeto consumidor desta camada, é criado um objeto de transporte e o preenchimento do mesmo para ser retornado e trafegado.

- **Camada de Lógica de Acesso a Dados (*Maitre.DataAccessLayer.ADO)***

Na camada de acesso a dados está localizada a persistência utilizada para trazê-los da base de dados e seguir o fluxo de transporte e transformação destes dados.

No Motor de Recomendações *Maître*, as classes da camada de acesso a dados implementam *interfaces* da camada *Maitre.DataAccessLayer* para garantir a padronização.

O banco de dados utilizado neste motor é SQL Server versão 2008 e o acesso é feito através de um *framework* de persistência chamado *LINQtoSQL*.

O *framework* acessa o banco de dados através de um gerenciador de conexões *MaitreDataContext,* responsável por abrir e fechar as conexões com o banco de dados garantindo que não existam bloqueios. Como o gerenciador implementa uma interface que permite a sua própria destruição, seu uso ocorre juntamente com a inicialização do bloco de código, sendo destruído ao final da execução deste bloco.

Para evitar o acesso das camadas à *string* de conexão do sistema, ela é disponibilizada apenas no projeto que dispara a execução, neste caso, está localizada no projeto dos serviços *Maître*. A *string* de conexão é um parâmetro lido pelo gerenciador de conexões através de uma reflexão direta apontando para o nome deste parâmetro. Portanto, para ser lido, o parâmetro deve respeitar o nome estabelecido no *Maître*.

O objetivo desta camada é receber requisições para o banco de dados e retornar os dados lidos em objetos de negócio. Cada classe dentro desta camada deve ser responsável pelo tratamento dos dados envolvidos, buscando o tipo de operação quando for uma gravação e a lógica na seleção dos dados. Grande parte dos métodos das classes desta camada trafega objetos de negócio e recebe por parâmetro ou cria outros objetos a partir da leitura de um *DTO.*

A figura 14 ilustra uma herança de classes da camada de lógica de acesso a dados utilizando a camada de interface. Na figura 15, observa-se a estrutura de conexão com o banco.

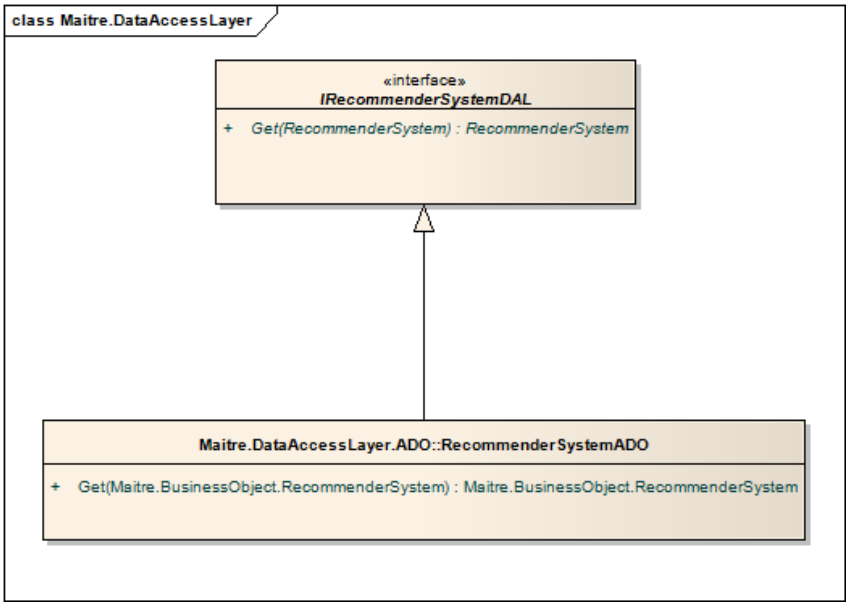


Figura 15: Herança entre classes das camadas de acesso a dados.

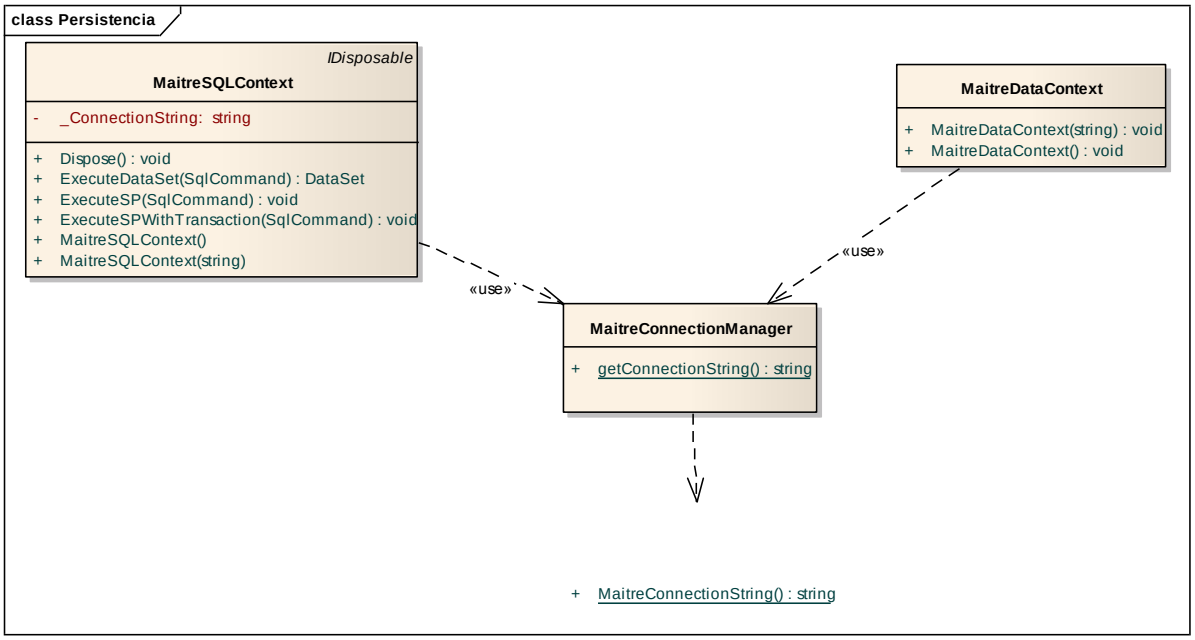


Figura 16: Diagrama de classes demonstrando o processo de conexão

A utilização da camada de persistencia pode ser melhor entendida no diagrama de sequência da camada de persistência conforme apresentado na figura 16.

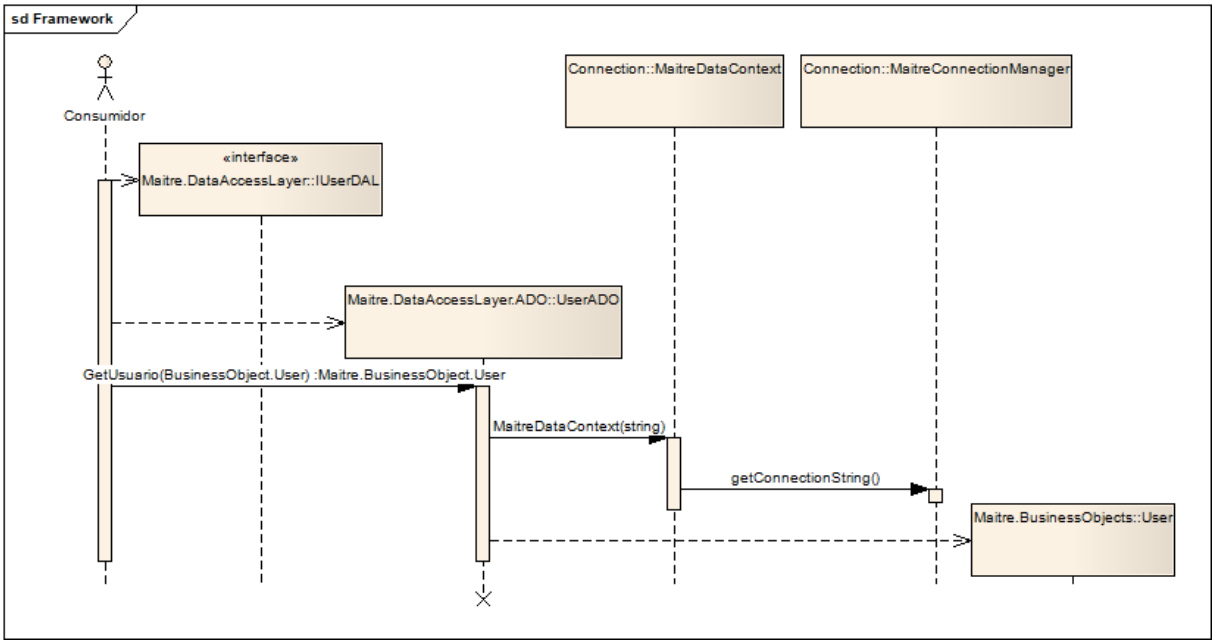


Figura 17: Diagrama de Sequência da camada de Persistência

As classes envolvidas no acesso a dados estão modeladas no diagrama de classes da camada de persistência conforme mostra a figura 17.

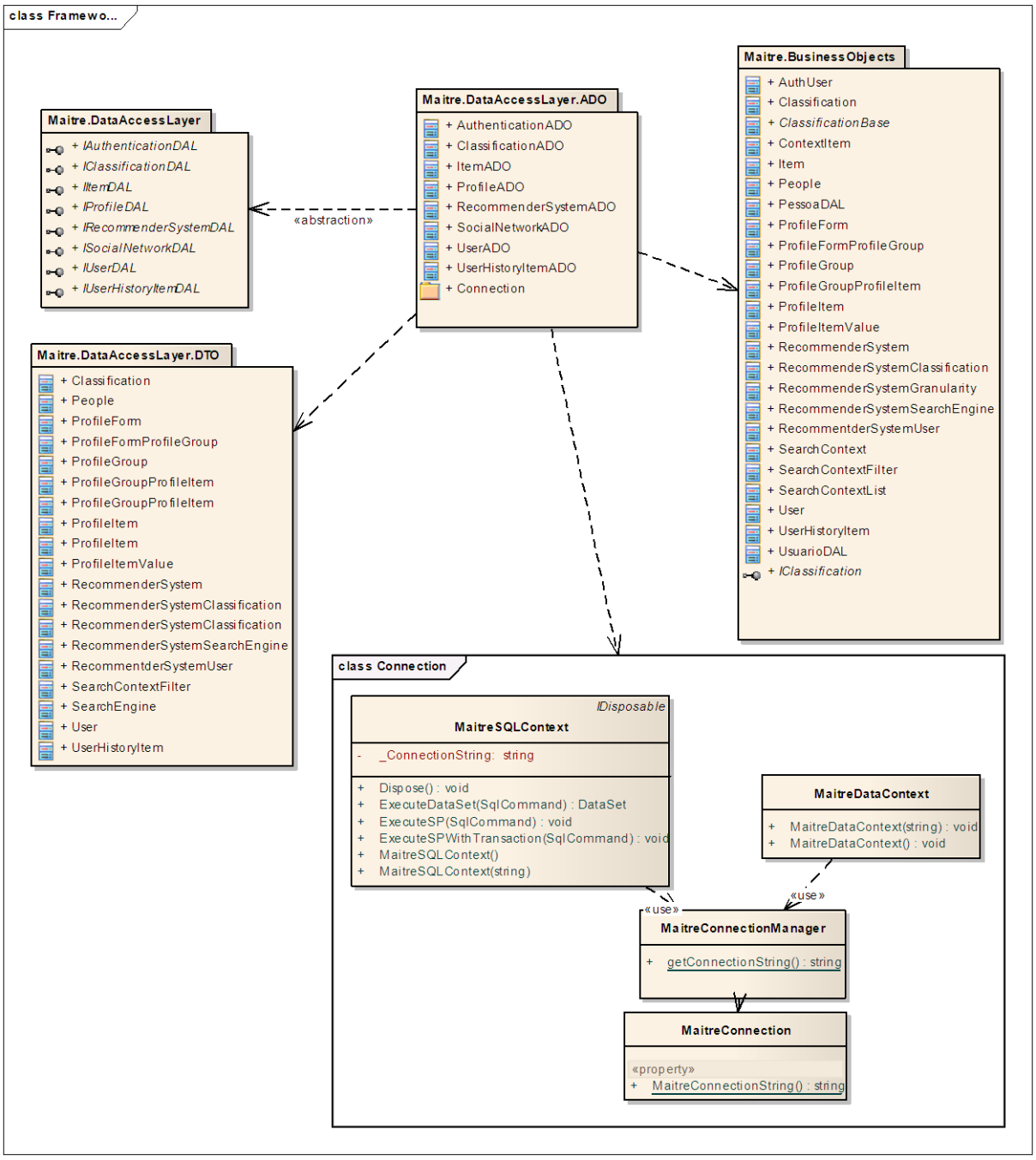


Figura 18: Diagrama de Classes da Camada de Persistência

Independentemente da lógica de busca dos dados de acordo com uma regra estabelecida no serviço, a conexão com a fonte de dados é feita nesta camada através do objeto *MaitreConnection*. Entretanto o serviço não tem visibilidade direta deste objeto de conexão e para estabelecer uma conexão, utiliza os conectores disponibilizados de acordo com o tipo de conexão utilizado para executar sua função.

Caso o serviço seja baseado em *TSQL* (linguagem *SQL Server*), deve utilizar uma instância da classe *MaitreSQLContext*, para conversar com o gerenciador de conexões, garantindo o controle das conexões.

As chamadas ao banco de dados devem, obrigatoriamente, passar pelo *framework*, pois, além das conexões, os serviços as ligam diretamente a ele para executar suas funções.

As classes envolvidas diretamente na conexão com o banco de dados podem ser melhor visualizadas analisando-se o diagrama de classes do *framework* de persistência apresentado na figura 18.

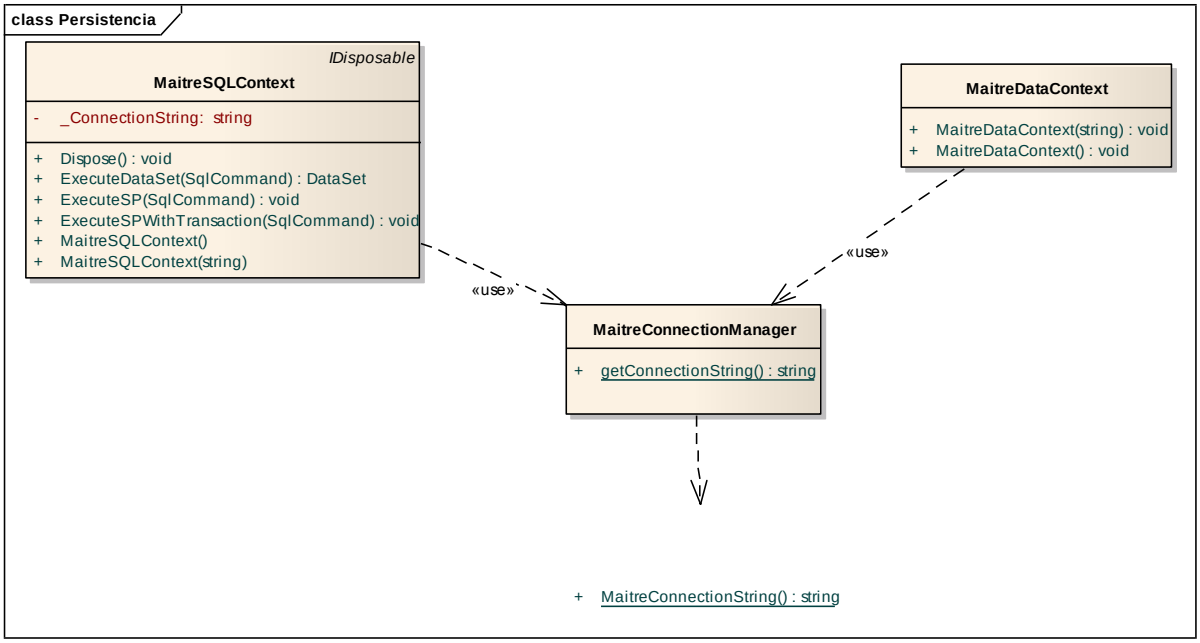


Figura 19: Diagrama de Classes da Camada de Acesso a Dados

Além do diagrama de classes, as classes correspondem às seguintes descrições:

***MaitreConnectionManager*:** é o gerenciador de conexões que exerce o papel de uma fábrica (*factory*), buscando dados para o estabelecimento de uma conexão. Todos os contextos (*contexts*) que forem estabelecer uma conexão com o banco de dados devem, obrigatoriamente, utilizar o gerenciador de conexões, pois somente ele as conhece.

Este gerenciador de conexões exerce também o papel de mediador entre os *contexts* e a *MaîtreConnection* permitindo as mudanças do banco de dados, ou seja,

se o tipo de conexão for alterado, basta adicionar uma propriedade na classe e configurar para que as classes *contexts* utilizem esta ao invés de outra.

***MaitreSQLContext*:** classe de acesso a dados baseada em *ADO.NET* que trabalha com *TSql* puro permitindo a manipulação de dados com o retorno genérico de uma *StoredProcedure*. O *MaitreSQLContext* transforma classes lógicas em dados sendo o contrário também verdadeiro através da linguagem pura de banco de dados. É de responsabilidade de suas instâncias fechar as conexões e garantir que os dados sejam trafegados entre banco de dados e cliente e não tratar erros e mascarar exceções que devem ser enviadas para que seus consumidores façam o correto tratamento.

***MaitreConnection*:** representa a *string* de conexão da camada de persistência. Toda o acesso a dados deverá ser através da *MaitreConnection* que é instanciada pelo gerenciador de conexões (*MaitreConnectionManager*). Esta conexão é uma fachada que conhece o meio de extrair a *string* de conexão definida na aplicação e que iniciou um serviço, necessitando um acesso ao banco de dados.

MaitreDataContext: a classe de DataContext do Motor de Recomendação, é uma especialização da classe DataContext da plataforma .NET. Esta transformação deve ser feita para utilizar os outros componentes do *framework* de persistência, como o gerenciador de conexão (*ConnectionManager*) e o *MaitreConnection*. O *MaitreDataContext* transforma classes lógicas em dados e o contrário, utilizando o *framework* de persistência utilizado neste projeto.

- **Camada de Utilidades**

A camada de utilidade organiza todas as funcionalidades que não fazem parte da arquitetura base e podem ser chamadas por qualquer camada do *Maître*. Todos os métodos estão dispostos de maneira estática.

Os principais métodos são referentes a formatações de datas e números e manipulação de strings.

- **Camada de Transformação de Dados (Maitre.Helpers)**

Possui as classes para ajuda na manipulação de listas e *strings*, sendo:

***ListHelper*:** possui um transformador de enumeradores em objetos de dados.

***TextHelper*:** possui os métodos para serializar e deserializar e os métodos para compactar e descompactar *strings*.

- **Camada de Log (*Maitre.Util.Log*)**

Registra todas as ações a partir de qualquer camada, sendo que os registros criados são gerados a partir da chamada de métodos estáticos. Essas chamadas não devem gerar erro para as camadas consumidoras, para que a gravação de um *log* não acarrete em um erro aninhado.

- **Camada de Mensagens (*Maitre.Util.Messages*)**

É responsável pela catalogação de todas as mensagens que devem ser exibidas no *Maître*. Através da centralização destas mensagens é possível, inclusive, utilizar o recurso de internacionalização da aplicação somente por meio de arquivos de *resources*. Nativamente, o *Maître* está disponível somente no idioma Português brasileiro.

- **Camada de Tratamento de Erros (*Maitre.Util.ErrorMessages*)**

É possível tratar todos os erros da aplicação através dela. Torna-se muito prática a implantação de funcionalidades de suporte ao usuário como o envio de *e-mails* contendo a tela com erro, por exemplo.

- **Camada Vertical**

O Maître possui uma camada vertical para transporte de dados entre as camadas fisicamente distribuídas.

- **Camada de Objetos de Negócio (*Maitre.BusinessObject*)**

Os objetos de negócio são trafegados verticalmente entre as camadas da arquitetura do *Maître* e refletem, em grande maioria, os objetos da base de dados. Esses objetos são populados conforme dados obtidos através da leitura de um *DTO*.

4.1 O ALGORITMO DE CLASSIFICAÇÃO

A proposta de implementação do algoritmo de classificação de itens do *Maître* compreende uma simplificação da proposta publicada por Cazela (2006). Em seus algoritmos este os aplica separadamente, ao contrário *Maître* que, através de uma matriz central construída no início do algoritmo, contém todos os itens a serem classificados e inseridos conforme as dimensões configuradas.

No *Maître*, nativamente, são configurados os algoritmos de: Perfil de Usuário, Histórico, Conhecimento e Rede Social que funcionarão sempre através da inserção de palavras-chave.

Esse algoritmo também possui uma configuração de granularidade onde será feito o corte dos itens conforme o tamanho do grão, ou seja, 1, 3, 5, 7 ou 9 itens serão direcionados para o motor de busca. Essa granularidade afeta a todos os algoritmos, porém, não interfere nas palavras-chave inseridas pelo usuário.

4.1.1 A MATRIZ DE CLASSIFICAÇÃO

A matriz de classificação é fundamental para aplicação das classificações dos itens por ser responsável pela interligação entre: itens, algoritmos, classificações e totalizações.

Além disso, a matriz de classificação serve como base para o motor de busca efetuar sua pesquisa, o que ocorre inserindo uma classificação positiva para os itens mais relevantes e uma negativa para itens menos relevantes.

Durante o processo de classificação, a matriz passa pelos seguintes processos:

a) criada – quando ainda não possui nenhuma linha e contem somente a coluna onde os itens serão inseridos;

b) populada – estado em que a matriz já contém os itens a serem classificados na primeira coluna;
c) construída – assim que aplicados os algoritmos, cada um insere seus itens, formando “n” linhas conforme cada algoritmo de classificação;
d) classificada – após construída a matriz e aplicados os algoritmos, ela passa por um processo de cálculo de cada item x cada algoritmo, formando uma classificação;
e) totalizada – quando todos os itens já estiverem classificados, é feita uma totalização somando todas as classificações para cada item; e
f) reordenada – aspecto final da matriz após ordenada conforme cada classificação pela coluna Total, sendo a maior classificação a primeira posição e a última a menor.

Assim, a matriz terá as seguintes apresentações:

- Apresentação da matriz **criada**:

Itens/Dimensões

Tabela 2: Criação da matriz de classificação

- Apresentação da matriz **populada**:

Itens/Dimensões
churrascaria
carne
japonesa
chinesa
gaúcho
porto alegre
masculino

Tabela 3: Matriz de Classificação Populada

- Apresentação da matriz **construída**:

Itens/Dimensões	Profile	History	Knowledge	SocialNetwork
churrascaria				
carne				
japonesa				
chinesa				
gaúcho				
porto alegre				
masculino				

Tabela 4: Matriz de Classificação Construída

- Apresentação da matriz **classificada**:

Itens/Dimensões	Profile	History	Knowledge	SocialNetwork
churrascaria	0	10	0	10
carne	0	0	10	0
japonesa	0	0	0	8
chinesa	-10	0	0	-6
gaúcho	10	0	0	0
porto alegre	4	0	0	0
masculino	8	0	0	0

Tabela 5: Matriz de Classificação Classificada

- Apresentação da matriz **totalizada**:

Itens/Dimensões	Profile	History	Knowledge	SocialNetwork	Total
churrascaria	0	7	0	10	17
carne	0	0	10	0	10
japonesa	0	0	0	8	8
chinesa	-10	0	0	-6	-16
gaúcho	10	0	0	0	10
porto alegre	4	0	0	0	4
masculino	8	0	0	0	8

Tabela 6: Matriz de Classificação Totalizada

- Apresentação da matriz **reordenada**:

Itens/Dimensões	Profile	History	Knowledge	SocialNetwork	Total
churrascaria	0	10	0	10	20
carne	0	0	10	0	10
gaúcho	10	0	0	0	10
japonesa	0	0	0	10	10
Masculino	10	0	0	0	10
porto alegre	0	0	0	0	0
chinesa	-10	0	0	-10	-20

Tabela 7: Matriz de Classificação Reordenada

4.1.2 AS DIMENSÕES DE CLASSIFICAÇÃO

O algoritmo principal do *Maître* trabalha com dimensões de classificação que podem ser considerados algoritmos menores. Nativamente, o *Maître* possui quatro (4) dimensões que fazem a classificação dos itens: algoritmo de perfil de usuário, algoritmo de histórico, algoritmo de conhecimento e algoritmos de redes sociais.

Cada algoritmo (dimensão) possui um grau de influência em um número decimal (em porcentagem), sendo que o somatório de todas as influências dos algoritmos deve ser 100%, nesse caso, um (1,00).

Os algoritmos estão todos cadastrados na base de dados do Maître, mas devem conter uma implementação de código seguindo um padrão estipulado para que funcionem, garantindo a aplicação sempre do mesmo padrão.

Assim que criado um algoritmo, ou seja, buscado na base de dados, ele passa a trabalhar sobre a matriz de classificação criando uma nova coluna com seu nome para que os itens possam, futuramente, serem classificados.

Durante o processamento do algoritmo a coluna passa a conter valores (considerando a influência do algoritmo) para cada item (linha) da coluna. Esse processo influencia como a matriz faz a contagem para classificar que itens serão incluídos na busca e os que devem ser excluídos.

Algoritmo de Perfil de Usuário:

O Perfil de Usuário é composto por diversas características, cada uma delas considerada quando houver a classificação aplicando esse algoritmo. O funcionamento ocorre através de itens de perfil de um grupo relevante de itens de perfil que deverá fazer parte do algoritmo. Esses itens são inseridos como mais itens (linhas) da matriz de classificação.

A classificação desse algoritmo funciona pontuando os itens da matriz de classificação presentes no perfil desse usuário e não pontua os itens que não estiverem marcados.

Algoritmo de Histórico:

O usuário quando acessa um item qualquer gera um histórico desse acesso, ou seja, um registro de que esse item já foi acessado pelo usuário. Quando for solicitada uma classificação a esse algoritmo, o algoritmo principal buscará em todos os itens já acessados anteriormente e o inserirá na matriz de classificação.

A pontuação positiva é dada para todos os itens que já tiverem sido acessados.

Algoritmo de Conhecimento:

Assim que busca e acessa um item, o usuário pode fornecer sua opinião quanto a esse item. Essa opinião é a base do algoritmo de conhecimento. Todos os itens retirados das opiniões proferidas pelo usuário fazem parte da matriz de classificação, sendo esta opinião um novo item (linha).

Durante a classificação, um item é classificado como positivo ou negativo conforme a opinião do usuário.

Algoritmo de Rede Social:

Para simplificar a utilização da Rede Social, o *Maître* utiliza uma Rede Social cadastrada na base de dados onde um usuário está relacionado a outro estabelecendo um vínculo bidirecional entre eles, nesse caso uma “amizade”. Um

usuário pode estar ligado a vários outros, compondo um conjunto de amizades. Através dessa relação, os itens anteriormente classificados pelos amigos da Rede Social são inseridos na matriz de classificação, sendo classificados e pontuando os itens que fazem parte da opinião dos amigos.

4.1.3 Funcionamento do Algoritmo *Maître*

O funcionamento do algoritmo principal, em pseudocódigo, consiste nos seguintes passos:

1. Busca de todos os algoritmos de classificação configurados na base de dados;
2. Criação da matriz de classificação contendo a estrutura básica vazia, ou seja, sem número de linhas e somente com a coluna que conterá os itens;
3. Construção da matriz de classificação inserindo os itens em novas linhas;
4. Criação das colunas da matriz de classificação onde cada algoritmo de classificação popula uma nova coluna;
5. Processamento dos algoritmos de classificação classificando todos os itens da matriz de classificação considerando o grau de influência de cada algoritmo; e
6. Reordenação dos itens conforme as classificações totalizadas na coluna Total, sendo o primeiro item o melhor classificado (com pontuação maior) e o último item o pior classificado.
7. Corte dos itens que estiverem fora da granularidade configurada para o conjunto de algoritmos. Além disso, são inseridos os itens com classificação negativa com o sinal de menos (-) para serem excluídos pelo motor de busca.

Para exemplificar melhor a aplicação do algoritmo principal, o processo demonstrativo correspondente ao pseudocódigo é o seguinte:

Passo 1) busca de algoritmos: são buscados os algoritmos de classificação que farão parte do algoritmo principal, sendo:
Algoritmos encontrados:

- ProfileAlgorithm (algoritmo de Perfil de Usuário);

- HistoryAlgorithm (algoritmo de Histórico);
- KnowledgeAlgorithm (algoritmo de Conhecimento); e
- SocialNetworkAlgorithm (algoritmo de Rede Social).

Passo 2) criação da matriz: é criada a matriz de classificação contendo uma coluna com os itens de classificação e nenhuma linha, conforme a apresentada na tabela Criação da Matriz de classificação.

Passo 3) construção da matriz: são inseridos todos os itens a serem classificados conforme os 4 (quatro) algoritmos. Sempre que houver qualquer item já inserido por algum algoritmo anterior, é desconsiderado, porque durante o ranqueamento obtém um valor mais alto por haver dois índices influentes, o que isenta a necessidade de duplicação dos itens. Além disso, os itens devem ser únicos, para que possam ser transformados em palavras-chave futuramente. Assim, teremos uma matriz com um número de linhas conforme os algoritmos, criadas conforme a tabela Exemplo de Matriz de Classificação Populada na tabela 8.

Itens/Dimensões
churrascaria
carne
japonesa
chinesa
gaúcho
porto alegre
masculino

Tabela 8: Exemplo de Matriz de Classificação Populada

Conforme a tabela, o item *churrascaria* foi derivado do algoritmo de histórico e os itens *japonesa, chinesa e churrascaria*, do algoritmo de Rede Social. Porém, o item *churrascaria* aplicado pela segunda vez foi ignorado por estar repetido. Além disso, foram gerados o item *carne* derivado algoritmo de conhecimento e os itens *gaúcho, porto alegre e masculino* do Perfil do Usuário.

Passo 4) criação das colunas da matriz: durante a primeira iteração, é inserida uma nova coluna na matriz referente ao primeiro algoritmo configurado, neste caso, o algoritmo de Perfil, chamado *Profile*. Durante as próximas iterações, são criadas as outras colunas conforme os outros algoritmos: Conhecimento, Histórico e Rede Social. A matriz de classificação corresponde à tabela 9.

Itens/Dimensões	Profile	History	Knowledge	SocialNetwork
churrascaria				
carne				
japonesa				
chinesa				
gaúcho				
porto alegre				
masculino				

Tabela 9: Matriz de Classificação Construída

Passo 5) Processamento dos algoritmos: momento onde são processadas todas as linhas e classificados conforme as colunas, gerando um resultado através de multiplicação, considerando o grau de influência. O resultado é inserido no cruzamento entre linha e coluna correspondentes.

A tabela final corresponde à tabela 10.

Itens/Dimensões	Profile	History	Knowledge	SocialNetwork
churrascaria	0	10	0	10
carne	0	0	10	0
japonesa	0	0	-10	10
chinesa	0	0	-10	-10
gaúcho	10	0	10	0
porto alegre	10	0	0	0
masculino	10	0	0	10

Tabela 10: Exemplo de Matriz de Classificação Classificada

Passo 6) reordenação da matriz de classificação: após a classificação de todos os itens, a matriz está pronta para ser reordenada, conforme exemplo apresentado na tabela 11 está representada a nova matriz de classificação.

Itens/Dimensões	Profile	History	Knowledge	SocialNetwork	Total
churrascaria	0	10	0	10	20
gaúcho	10	0	10	0	20
masculino	10	0	0	10	20
porto alegre	10	0	0	0	10
carne	0	0	10	0	10
japonesa	0	0	-10	10	0
chinesa	0	0	-10	-10	-20

Tabela 11: Exemplo de Matriz de Classificação Reordenada

Passo 7) após a reordenação, o algoritmo leva em consideração a granularidade (a inicial é 3) e efetua o corte dos itens melhores. Nesse caso, levando em consideração a granularidade inicial, os itens classificados são: *churrascaria, masculino e porto alegre*. Adicionalmente, é inserido o item *chinesa* com indicação negativa na frente, o que significa ao motor de busca a exclusão desse termo.

4.2 MODELO DE DADOS

Para atender as funcionalidades e características existentes no Motor de Recomendação *Maître*, a figura 19 ilustra o modelo de dados.

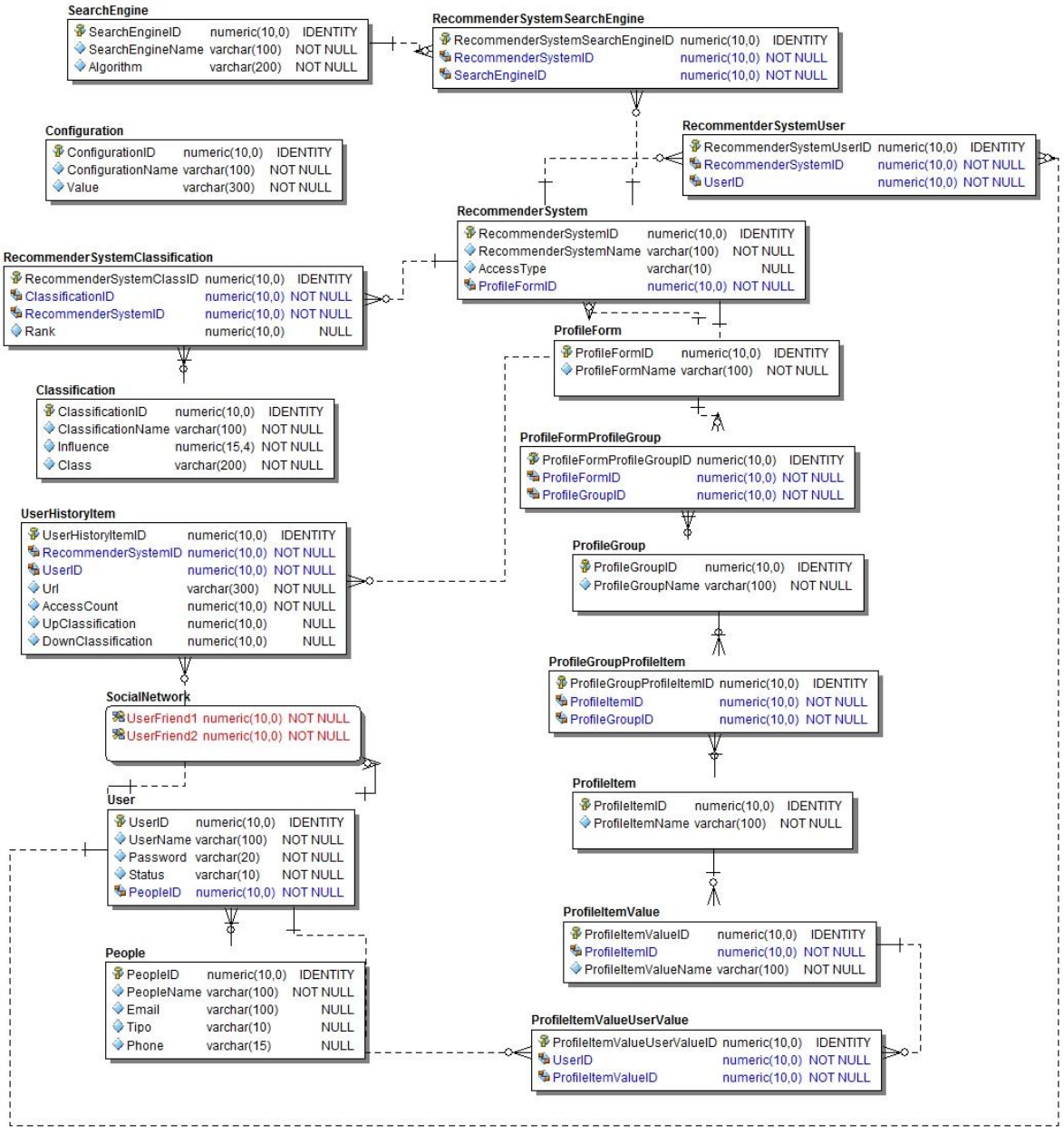


Figura 20: Modelo de dados *Maître*

As tabelas especificadas no diagrama do modelo de dados *Maître* estão descritas a seguir:

- ***Classification***

Armazena as classificações que podem ser utilizadas pelo Motor de Recomendações. Nesta tabela ficam o nome da classificação, o código, o grau de

influencia no algoritmo e qual a classe dentro do motor que executa o algoritmo de classificação.

- ***Configuration***

Tabela de parâmetros de configuração do Motor de Recomendação, possui um código e uma descrição para armazenar qualquer tipo de configuração.

- ***People***

Na tabela *People* são armazenados os dados básicos de uma pessoa, como nome, código, telefone, *e-mail* e tipo.

- ***User***

A tabela de usuários é uma especialização da tabela de pessoas, contendo os dados extra necessários para efetuar um login junto ao Motor de Recomendações.

- ***SearchEngine***

No cadastro de um motor de busca, o sistema pode ter as informações de nome, código e qual o algoritmo de busca utilizado. Um exemplo de motor de busca é o *Google*.

- ***ProfileForm***

A tabela *ProfileForm* guarda as informações de formulários de perfil. Como por exemplo um formulário chamado "*Maître*", que conterá todas as informações de perfis vinculados ao Motor de Recomendação.

- ***ProfileGroup***

Na tabela de grupos de perfil, são armazenados os dados referentes aos tipos de cadastro, como por exemplo “Dados de Usuário”. São os cadastros disponíveis dentro do formulário que existe no Sistema de Recomendação.

- ***ProfileItem***

Um item de perfil é uma característica possíveis dentro de um cadastro. O item de perfil é geral no banco de dados, ou seja, depois ele será vinculado a um grupo. Um exemplo de item de perfil pode ser “Cidades”, “Bares”, “Universidade”, e etc.

- ***ProfileFormProfileGroup***

Nesta tabela é identificado quais grupos estão presentes em um formulário de perfil, como por exemplo, para o formulário “*Maître”*, podem ser cadastrados os grupos “Dados de Usuário” e “Preferencias de Usuário”.

- ***ProfileGroupProfileItem***

Nesta tabela são vinculados os grupos de perfil aos itens de perfil. Colocar um grupo “Dados de Usuário” com os itens “Cidade”, “Comida” e “Time de Futebol” seria um exemplo de dados contidos nesta tabela.

- ***ProfileItemValue***

Aqui são cadastrados os valores que podem ser escolhidos por um usuário ao efetuar um cadastro. Então se tiver um item chamado “Cidade” com os valores “Capão da Canoa”, “Viamão” e “Bagé” seria um exemplo de registros presentes nesta tabela.

- ***ProfileItemValueUserValue***

Na estrutura de itens de perfil, aqui é o nodo folha, pois é onde ocorre o vínculo de um valor (Form>Grupo>Item>Valor) com um usuário.

- ***RecommenderSystemClassification***

Mapeia os sistemas de recomendação e as classificações, podendo ranquear esta ligação, ou seja, uma classificação pode ter uma ordem diferente para cada sistema que utiliza de seus recursos.

- ***RecommenderSystem***

Tabela que armazena os dados de um sistema de recomendação, como nome, código e também o form que contém todos os dados de perfis de usuário. O sistema de recomendação é a aplicação que consome os serviços do Motor de Recomendações, manipulada pelo usuário final.

- ***RecommenderSystemSearchEngine***

Liga o sistema de recomendação aos motores de busca, indicando quais motores são utilizados por um sistema.

- ***RecommenderSystemUser***

Uma das tabelas núcleo do modelo de dados é o vínculo de usuário com sistema de recomendação, pois, uma vez cadastrado no banco de dados do *Maître*, um usuário apenas precisa de permissão para acessar um sistema.

- ***SocialNetwork***

A tabela de redes sociais guarda o registro de “amizade” entre dois usuários, ou seja, aqui é mapeado o vínculo de usuários formando uma rede social.

- ***UserHistoryItem***

Nesta tabela é montado o histórico de um usuário, suas classificações, acessos, urls favoritas, palavras chave e etc, e todas essas informações realizadas dentro de um sistema de recomendação.

5. A APLICAÇÃO APPRENTI MAÎTRE

A camada *mobile* compreende todos os pacotes, classes, arquivos e dados utilizados no Dispositivo Móvel. Divide-se em: *AppConsomer* e *APIConsomer*, sendo que a primeira, a *AppConsomer,* é de responsabilidade do consumidor, ou seja, este utiliza o pacote para efetuar as operações possíveis através da chamada da *API* dessa camada.

Nesta camada está o *Apprenti Maître*, projeto piloto de testes do Motor de Recomendações *Maître*. O *Apprenti Maître* utiliza os recursos do Motor para auxiliar seu usuário na busca por itens. As requisições feitas ao Motor são formatadas no padrão de comunicação esperado, que é estabelecido em contrato.

Através dessa comunicação o Motor fornece ao consumidor uma lista de recomendações para que as utilize da forma que julgar necessária, restringindo portanto seu papel ao abastecimento do motor com suas necessidades.

O *ApprenttiMaître* é um sistema de recomendação cadastrado no banco de dados do Motor de Recomendações *Maître* que se utiliza um código de autenticação para utilizar os serviços disponíveis no motor.

O algoritmo de Perfil de Usuário e Rede Social utilizados nesse projeto piloto compreende um perfil de usuário e rede social cadastrados na base de dados através de *scripts*.

5.1. CASOS DE USO

A descrição dos casos de uso encontram-se nos anexos desse documento. A figura 20 mostra o diagrama de casos de uso do *Apprenti Maître*:

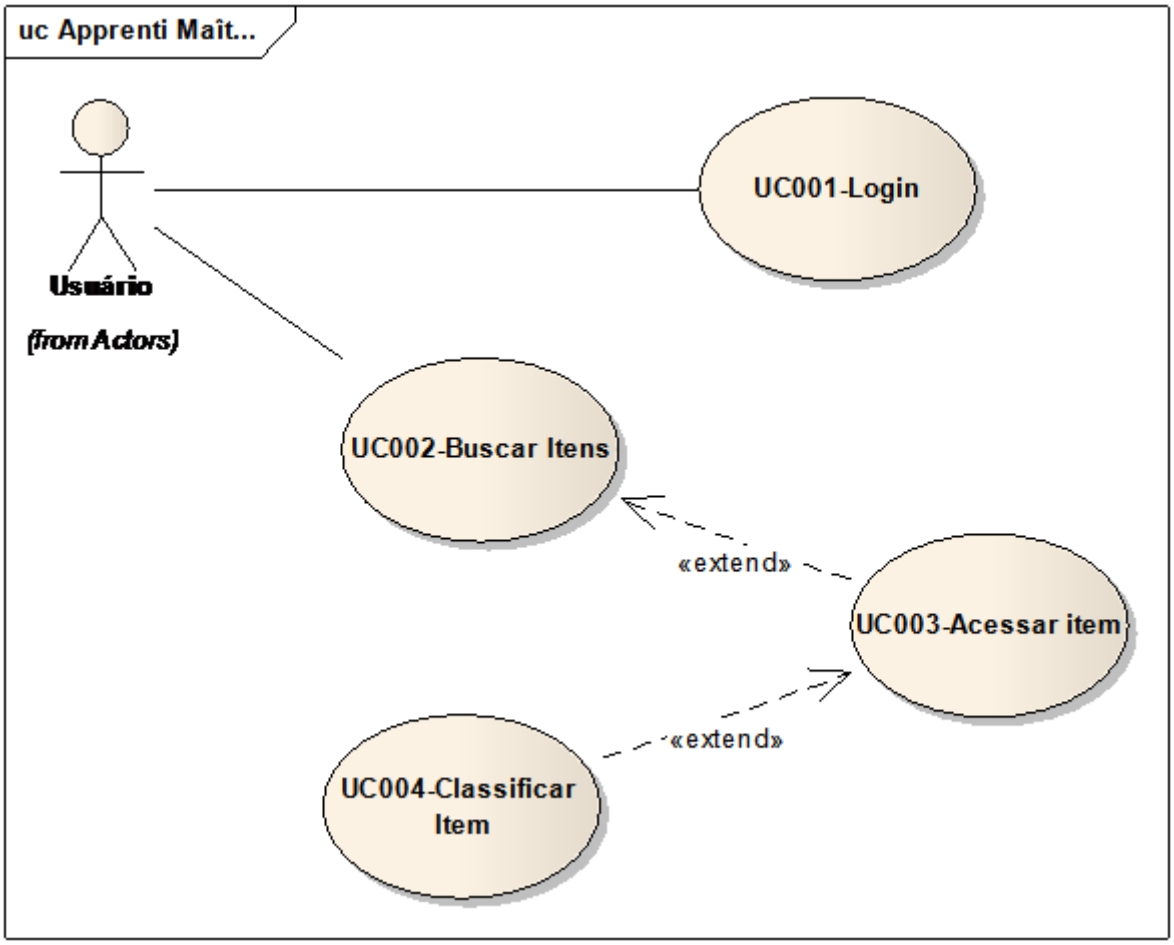


Figura 21: Diagrama de Casos de Uso da aplicação *Apprenti Maître*

O *ApprentiMaitre* tem seu funcionamento iniciado com a autenticação de um usuário pelo caso de uso UC001 informando os dados de usuário e senha. Esta autenticação é feita diretamente no servidor *Maître* através de uma chamada ao *WebService UserService*, disparada pela API consumidora localizada no dispositivo móvel.

Esse é o primeiro canal de comunicação estabelecido entre o Motor e o Consumidor e comprova a disposição de dados somente no servidor, ou seja, a aplicação consumidora não visualiza diretament os dados no banco de dados.

Para que o *Maître* possa recomendar palavras-chave de acordo com o uso do sistema pelo usuário, é necessário estabelecer um vínculo entre ambos. Este vínculo é criado através de um retorno por uma busca de informações que pode ser acessado e classificado. Detalhando este processo, a aplicação dos algoritmos de histórico, rede social e conhecimento, necessita que o motor de recomendação saiba informações sobre o usuário, que não estão explicitamente colocadas em seu perfil.

As informações citadas devem ser armazenadas com o uso do sistema pelo usuário, ou seja, quando o usuário acessa um item que foi retornado pela recomendação de palavras-chave, estas palavras ganham um ponto de acesso. Caso este item seja classificado (tanto para positivo quanto para negativo) ele recebe um grau de classificação, no campo adequado, ou seja, as classificações não são interlgadas, existe um campo para classificações positivas e um campo para as negativas.

Para que inicie o processo de conhecimento (apresentação), o usuário informa na tela de busca (UC002), as palavras-chave que ele deseja pesquisar, no caso deste protótipo, os retornos são links da web. Neste mesmo campo de busca, após o motor de recomendação ser executado, são informadas todas as palavras-chave utilizadas para que o retorno exibido fosse obtido. Essas palavras-chave são recomendadas através da aplicação dos algoritmos do motor.

Com as palavras que o usuário informou e as palavras que o motor encontrou para o usuário são obtidos resultados e exibidos nesta mesma tela em uma grade, com uma breve descrição do item. A partir destes itens, o usuário pode iniciar o processo de vínculo com as palavras-chave recomendadas de forma implícita. O vínculo é estabelecido com a execução do UC003 onde há o acesso a um item retornado, fazendo com que a *URL* deste item seja marcada como acessada juntamente com as palavras-chave.

No momento em que o item é acessado, o usuário visualiza a tela de detalhes, onde há um *minibrowser* que apresenta o acesso ao item escolhido. A partir desse ponto, é estabelecido uma ligação entre o usuário e o item. Essa ligação faz com que o item seja visualizado nas primeiras posições no caso de uma classificação positiva, ou nas últimas posições no caso da classificação negativa, realizando o caso de uso UC004.

Após a classificação positiva do item, quando este for encontrado novamente, será priorizado, pois está vinculado ao usuário. No caso em que o usuário executa a mesma busca, as palavras-chave tendem mudar, pois há itens no histórico do usuário.

Os itens classificados pelo usuário compõem o conhecimento que o motor tem sobre o usuário, recomendando seus itens aos demais usuários da rede social quando esse usuário executar uma busca.

O motor de recomendação mantém os históricos de acesso do usuário e quando este acessar novamente e efetuar a mesma busca, mesmo que em um dispositivo móvel diferente, suas recomendações serão as mesmas.

5.2. PROTÓTIPO DE INTERFACE

O *Apprenti* possui um conjunto de telas conforme suas funcionalidades: login, pesquisa, lista de itens e item.

5.2.1. Login

A tela de *login* possibilita ao usuário acessar o sistema e iniciar uma navegação utilizando dados já conhecidos sobre si. Essa tela, também pode cancelar a entrada no sistema.

Figura 22: Tela de *Login*

5.2.2. Pesquisa

A tela de pesquisa é visualizada pelo usuário quando acessa o sistema. Nela o usuário poderá pesquisar e obter os itens recomendados.

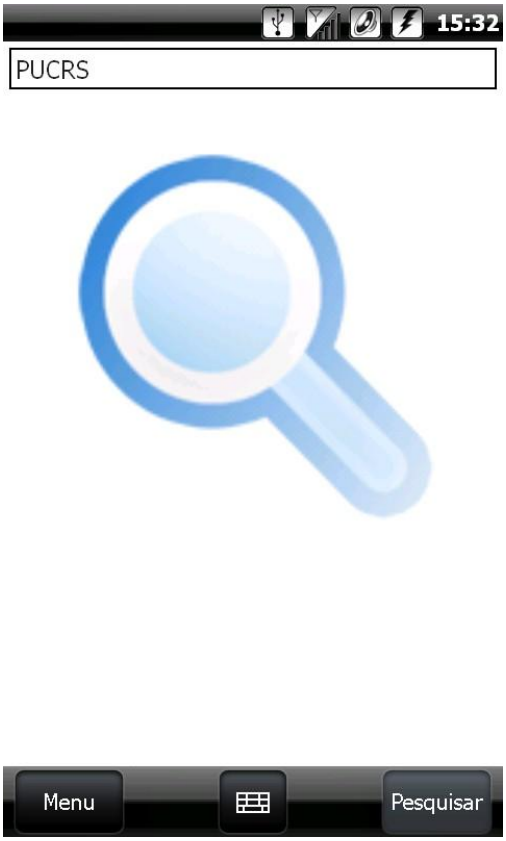


Figura 23: Tela de Pesquisa

5.2.3. Lista de Itens

A tela que lista os itens recomendadas ao usuário traz em formato de lista os itens obtidos por recomendação através do Motor de Recomendações *Maître*. Nesta tela o usuário tem acesso às palavras-chave utilizadas na pesquisa e é possível seguir navegando entre as páginas disponíveis, acessando um item e efetuando uma nova busca.

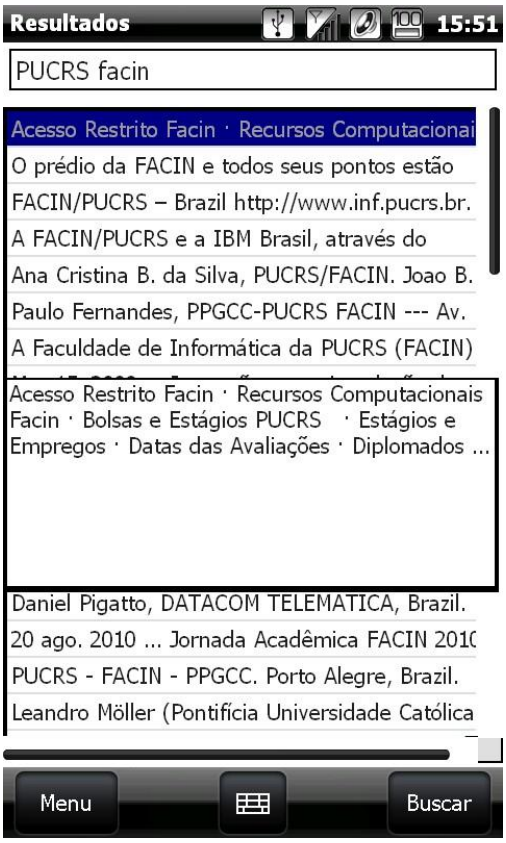


Figura 24: Tela Lista de Itens

5.2.4. Item

Ao clicar em um dos itens da lista, o sistema direciona o usuário para uma tela com maiores informações sobre o item escolhido. Nela é possível classificar o Item, realizar uma nova busca ou voltar para a lista. Na tela de Item, o usuário tem acesso à *URL* que fornece acesso ao item recomendado.

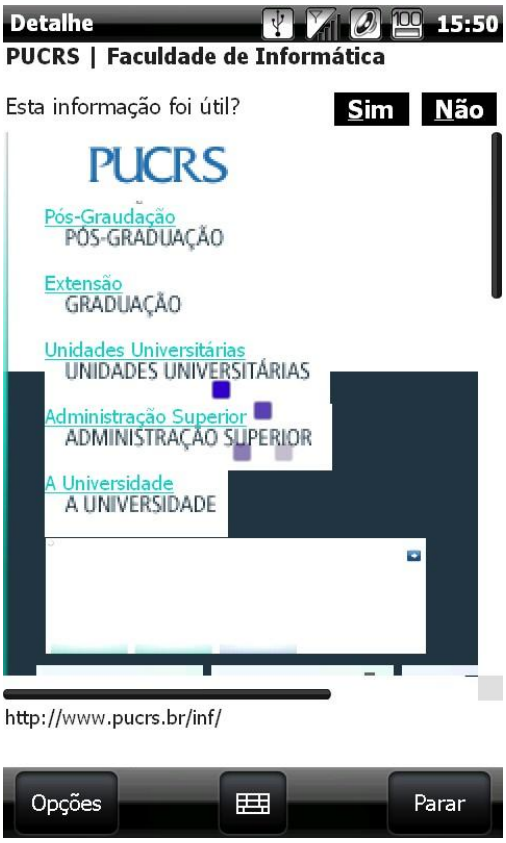


Figura 25: Tela de Consulta de Item

6. CONSIDERAÇÕES FINAIS

Esse trabalho expôs uma pesquisa teórica, contendo fundamentos sobre Sistemas de Recomendação, Dispositivos Móveis e Tomada de Decisão. A partir do estudo dos conceitos nela presente, foi projetada uma arquitetura que teve como ponto central a viabilidade de técnicas para que as aplicações desenvolvidas para Dispositivos Móveis possam usufruir de recomendações.

A implementação do Motor de Recomendação foi feita através de um piloto, o *Apprenti Maître*, que corroborou a flexibilidade da arquitetura e possibilitou um importante complemento ao desenvolvimento de aplicações voltadas a Dispositivos Móveis.

Dessa forma, os objetivos desse trabalho foram alcançados, na medida em que sua conclusão permitiu com que a comunidade de desenvolvedores voltados a Dispositivos Móveis tivesse à sua disposição uma plataforma completa e flexível composta por um motor de busca e classificação.

Durante o trabalho, o uso dos algoritmos possibilitou uma personalização das informações, além de um ganho de tempo em pesquisas, inteligência e uma maior satisfação dos resultados. Isso foi possível na medida em que os resultados iniciais foram acrescidos de informações históricas, através do algoritmo de Histórico, de informações de opiniões anteriores, através do algoritmo de Conhecimento, de informações extraídas de seu Perfil, através do algoritmo de Perfil de Usuário e de informações de seus amigos, através do algoritmo de Rede Social. Além disso, destaca-se a distribuição e disponibilidade que a arquitetura possui, fornecida através de acesso a serviços *web* (*WebServices*), o que possibilitou uma grande escalabilidade dessa aplicação introduzindo conceitos como aplicação em nuvens.

Entretanto, cabe aqui realizar uma ressalva, que diz da necessidade uma conexão de rede permanente para que o *Maître* cumpra seu propósito. No caso desse trabalho, foi utilizada a *Internet*. Todavia, se examinarmos melhor essa assertiva, podemos perceber que de fato isso não denota exatamente um problema, em função de que as conexões à *Internet* estão, cada vez mais, fazendo parte do cotidiano das pessoas.

Cabe assinalar que a principal dificuldade encontrada durante a elaboração desse trabalho foi o desenvolvimento voltado para Dispositivos Móveis, justificada pelo fato de que ainda não possui uma gama extensa de documentação. Porém,

após alguns ensaios e erros, foi possível desenvolver a aplicação do projeto piloto, o *Apprenti Maître*.

Alguns aspectos poderiam ser observados como forma de enriquecer o campo de pesquisas nessa área, como contribuição a trabalhos futuros, oferecendo outros enfoques através de novas pesquisas de aprimoramento dos temas aqui abordados, como:

- Aplicação *web* para gestão completa do *Maître*;
- Extração de métricas da eficiência e eficácia do uso do *Maître* e análise destas métricas obtidas;
- Implementação de múltiplos motores de busca para maior refinamento da buscas;
- Implementação de novos algoritmos de classificação de informações, como algoritmo Demográfico;
- Implementação de novas plataformas tornando o *Maître* multiplataforma, como plataforma voltada para dispositivos da *Apple*, por exemplo;
- Estudo da viabilidade de comercialização do *Maître* no mercado brasileiro;
- Implementação de algoritmos de criptografia e inserção de redes seguras para comunicação de dados; e
- Estudos de usabilidade do *Apprenti Maître*, garantindo ainda mais eficiência durante as buscas.

Por fim, é importante que se ressalte a probabilidade de que os aplicativos desenvolvidos para Dispositivos Móveis cresçam, ganhem novas funcionalidades e capacidades, confluindo o curso atual com Sistemas mais inteligentes e capazes de interpretar ainda melhor os desejos de seus usuários.

7. REFERÊNCIAS BIBLIOGRÁFICAS

ADOMAVICIUS, G. et al. **Incorporating contextual information in recommender systems using a multidimensional approach**. ACM Transactions on Information Systems, New York v.23, n. 1, p. 103- 45. Jan. 2005.

ADOMAVICIUS, G. TUZHILIN, A. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. **IEEE Transactions on Knowledge and Data Engineering**, VOL. 17, NO. 6, JUN 2005.

BALABANOVIC, M.; SHOHAM, Y. Fab: Content-Based, Collaborative Recommendation. **Communications of the ACM**, New York, v.40, n.3, p. 66-72, Mar. 1997.

BELKIN, N. J. Helping people find what they don't know.**Communications of the ACM**, New York, v.43, n.8, p. 58-61, Aug.2000.

BELVIN, N. J. CROFT, W. B. Information Filtering and Information Retrieval: two sides of the same coin? **Communications of the ACM**, New York, v.35, n.12, p. 29, Dez. 1992.

BRAGA, Ryon. **O Excesso de Informação. A Neurose do Século XXI.** In: HSM Management Update nº16 - Janeiro 2005 .

BURKE, R. A Case-Based Reasoning Approach to Collaborative Filtering. **Advances in Case-Based Reasoning, 5th European Workshop, EWCBR, September 6-9**, **Proceedings.** Trento, Italy, Lecture Notes in Computer Science 1898 Springer. Set 2000.

_____. **Hybrid Recommender Systems: Survey and Experiments. User Modeling and User – Adapted Interaction**. 2002

CASACA, Cristiane Borges. **Recommender Systems: Um sistema viável para organizações públicas de caráter inspetivo. Disponível** em:<http://repositorioaberto.univ-ab.pt/bitstream/10400.2/591/4/RRHT_final.pdf> Acessado em: 19 Jun. 2010.

CAZELA, S. **Aplicando a Relevância da Opinião de Usuários em Sistema de Recomendação para Pesquisadores**. 180 f. Tese (Doutorado em Ciencia da Computação) UFRGS. 2006 Disponível em:

<http://www.lume.ufrgs.br/bitstream/handle/10183/8424/000575704.pdf?sequence=1> Acessado em: 21 Nov. 2010.

CHIAVENATO, Idalberto. **Introdução à Teoria da Administração**. 5 ed. São Paulo: Makron Books, 1997.

COMPUTERWORLD/EUA (2011). **Cinco tendências para observar em 2011: lista é baseada nos planos de investimentos nas companhias no próximo ano. 2011** Disponível em:< http://computerworld.uol.com.br/negocios/2010/10/11/cinco-tendencias-para-observar-em-2011/> Acessado em: 15 Nov. 2011.

DA SILVA, Douglas. BARBOSA, Jorge. VIEIRA, Renata. **OCtoPUS: Um Modelo para Classificação de Perfil de Usuários usando Trilhas em Sistemas Ubíquos.** Programa de Pós-Graduação em Computação Aplicada (PIPCA). Universidade do Vale do Rio dos Sinos (UNISINOS) – São Leopoldo – RS – Brasil Faculdade de Informática – Pontifícia Universidade Católica do Rio Grande do Sul (PUC-RS) – Porto Alegre – RS – Brasil, 2005. Disponível em: <http://www.sbc.org.br/bibliotecadigital/download.php?paper=1430> Acessado em: 14 Nov. 2011.

DA SILVA, Manuel Fernando Rodrigues. **Proposta de uma Framework para Desenvolvimento de Sistemas Tutores Inteligentes.** Universidade do Minho, Departamento de Sistemas de Informação. Disponível em: <http://repositorium.sdum.uminho.pt/bitstream/1822/7656/1/Tese%20mestrado%20final.pdf>. Acessado em: 21 Nov. 2010.

FELFERNIG, A. **Koba4MS: Selling Complex Products and Services using Knowledge-based Recommender Technologies.** Proceedings of the Seventh IEEE International conference on E-Commerce Technology (CEC'05). Technisce Universität München, Germany, 2005.

FOLTZ, P. DUMAIS, S. **Personalized Information Delivery: An Analysis of Information Filtering Methods.** Communications of the ACM, New York, v.35, n.12, p. 51-60, Dez. 1992.

GOLDBERG, D. **Using collaborative filtering to weave an information Tapestry.** Communications of the ACM, New York, v.35, n.12, p. 61-70, Dez, 1992.

GOMES, Edeyson Andrade. **Fidus; Uma Ferramenta para buscar informações personalizadas na Web**. Dissertação de Mestrado em Banco de Dados – Universidade Federal da Paraíba, Campina Grande, 2001. Disponível em: < http://www.dsc.ufcg.edu.br/~copin/pesquisa/bancodissertacoes/2001/> Acessado em 08 de abril de 2010.

GOMES, Luiz Flavio; GOMES, Carlos Francisco Simões. ALMEIDA, Adiel Teixeira. **Tomada de Decisão Gerencial: um enfoque multicritério**. 2 ed. São Paulo: Atlas, 2006.

HERLOCKER, J. L. **Understanding and Improving Automated Collaborative Filtering Systems.** 220 f. Tese (Doutorado em Ciência da Computação) - University of Minnesota, Minnesota, 2000.

KONSTAN, J. A., KONSTAN, J. A. et al. **Grouplens: Applying Collaborative Filtering to Usenet News.** Communications of the ACM, New York, v.40, n.3, p. 77-87, Mar. 1997.

LAUDON, Kenneth C.; LAUDON, Jane Price. **Sistemas de informação**. Rio de Janeiro: LTC, 1999.

LEVENE, M., Peterson, D. **Trail Record and Ampliative Learning**, Research Report BBKCS-02-01, School of Computer Science and Information Systems. University of London, 2002.

LEVIS, D., BARBOSA, J., PINTO, S., BARBOSA, D. **Aperfeiçoamento Automático do Perfil do Aprendiz em Ambientes de Educação Ubíqua**, Revista Brasileira de Informática na Educação, v. 16, p. 29-41, 2008.

LEVY, Pierre. **A Revolução contemporânea em matéria de comunicação.** Porto Alegre: Edi*pucrs/Sulina, 1999.*

_____. *Um "Chat" com \pierre \lévy.**Entrevista a Eduardo Veras***. Agência RBS.23/05/2000. Em: http://www.urisan.tche.br/~dfrancis/pierrelv.htm. Acesso em: 20.10.2010,18:00h.

_____. *A Inteligência Coletiva Segundo Pierre Lévy. **Entrevista a Raphael Perret***. In: http://webinsider.uol.com.br/2002/09/09/a-inteligencia-coletiva-segundo-pierre-levy/ Acesso em: 20.10.2010,18:00h.

LI, Q. KIM, B. M. **An Approach for Combining Content-Based and Collaborative Filters.** Proceedings of the Sixth International Workshop on Information Retrieval with Asian Languages. 2003.

LOEB, S.; TERRY, D.. **Information Filtering.** Comumunications of ACM, New York, v.35, n.12, p.26, Dez, 1992.

LOUREIRO, Antonio A.F. SADOK, Djamel. Mateus, Geraldo R,. Nogueira, José Marcos S. Kelner, Judith. Comunicação Sem Fio e Computação Móvel: Tecnologias, Desafios e Oportunidades Disponível em: <http://homepages.dcc.ufmg.br/~loureiro/cm-rssf.html> Acessado em: 15 Nov. 2011.

LUEG, C. Considering Collaborative Filtering as Groupware:Experiences and Lessons Learned. In: INTERNATIONAL CONFERENCE ON PRACTICAL ASPECTS OF KNOWLEDGE MANAGEMENT, 1998. **Proceedings...** [S.l.: s.n.], 1998.

MONTANER, M et al.. **A Taxonomy of recommender Agents on the Internet.** Artificial Intelligence Review. Netherlands : Kluwer Academic Publishers, pp. 285-330, Ago 2003.

MONTANER, M.; LÓPEZ,B.; LA ROSA, J. L. A Taxonomy of Recommender Agents on the Internet. **Artificial Intelligence Review**,[S.l.], v.19, n. 4, p. 285–330, 2003.

MEIRELLES, Luiz Fernando. TAROUCO, Liane. **Framework para Aprendizagem com Mobilidade**. Artigo Completo. PGIE – Universidade Federal do Rio Grande do Sul (UFRGS) Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brasil, 2005.

NAISMITH, L., LONSDALE P., VAVOULA, G., SHET SHARPLES, M. **REPORT 11: Literature Review in Mobile Technologies and Learning**, NESTA FUTURELAB SERIES, University of Birmingham, 2004.

OLIVEIRA, D. P. R. **Sistemas de informações gerenciais: estratégias, táticas, operacionais.** 9 ed. São Paulo: Atlas, 2004.

OLIVEIRA, Carlos Augusto F.; REIS, Josiene Fabíola. **Filtragem colaborativa - uma forma de personalizar informações**. CienteFico. Ano IV, v. I, Salvador, janeiro - junho, 2004. Disponível em: <http://www.frb.br/ciente/Impressa/Info/I.3.Oliveira,CAF.%20FiltragemColaborativa.pdf> Acessado em: 15 Jun. 2010.

PENNOCK, Davis M. et al. **Collaborative Filtering by Personality Diagnosis: A Hybrid Memory- and Model-Based Approach**. NEC Research Institute, Inc. In: CONFERENCE ON UNCERTAINTY IN ARTIFICIAL INTELLIGENCE, 16, Proceedings. Princeton, Estados Unidos, 2000.

PALUDO, Lauriana. **Um estudo sobre as tecnologias java de desenvolvimento de aplicações movies..** UNIVERSIDADE FEDERAL DE SANTA CATARINA

DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA 2003. Disponível em: <http://pitagoras.usach.cl/~eflores/lcc/cd_redes/java-aplicaciones-moviles.pdf> Acessado em: 14 Nov. 2010.

REATEGUI, Eliseo Berni. CAZELLA, Silvio. **Sistemas de Recomendação**. XXV Congresso da Sociedade brasileira de Computação a universidade da computação: um agente de inovação de onhecimento. 22 A 29 DE JULHO, UNISINOS, 2005.

RÊGO, Lucas Drumond; NERES, Alisson Lindoso; GIRARDI, Rosario. **InfoNorma: Um Sistema de Recomendação baseado em Tecnologias da Web Semântica** UFMA - Universidade Federal do Maranhão DEINF - Departamento de Informática, 2006. Em: <http://www.dcc.ufla.br/infocomp/artigos/v5.4/art11.pdf > Acessado em: 09/04/2010.

RESNICK, P. and H. VARIAN (1997). **Recommender Systems (introduction to special section).** Communications of the ACM 40(3): 56-58. 1997.

RHEINGOLD, Howard; **The virtual community: homesteading on the electronic frontier**. Massachusetts, USA: MIT Press, 2000.

SCHAFER, J.B.; KONSTAN, J.; RIEDL, J. **Recommender Systems in ECommerce.** In: ACM CONFERENCE ON E-COMMERCE, 2., 1999. Disponível em: <http://portal.acm.org/citation.cfm?id=336992.337035> Acessado em 17 Jun. 2010.

SCHAFER, J. Bem. **The Application of Data-Mining to Recommender Systems.** University of Northern Iowa, 2005 Disp. em: <http://www.cs.uni.edu/~schafer/publications/dmChapter.pdf>Acessado em: 16.06. 2010.

SILVA, A.L. **Modelos de IDs para usuários de Dispositivos Móveis**.1998. **Dissertação de Mestrado (**Mestrado em Engenharia)- Programa de Pós-Graduação em Engenharia de Eletricidade. Universidade Federal do Maranhão, Maranhão.

SOBOROFF, Ian; NICHOLAS, Charles. **Collaborative filtering and the generalized vector space model.** In: ANNUAL ACM CONFERENCE ON RESEARCH AND DEVELOPMENT IN INFORMATION RETRIEVAL, 2000. Atenas, Grécia. Disponível em:http://portal.acm.org/citation.cfm?id=345646&coll=portal&dl=ACM&CFID=10096708&CFTOKEN=12334363&ret=1#Fulltext> Acessado em: 17 Jun 2010.

TESLER, Larry. **Networked computing in the 1990s**. Scientific American, 265(3):54–61. Set, 1991

YU, KAI ET AL.**Selecting relevant instances for efficient and accurate collaborative filtering.** ACM. in: international conference on information and knowledge management, 10, proceedings. Atlanta, E.U.. Disponível em:<http://delivery.acm.org/10.1145/510000/502626/p239-.pdf?key1=502626&key2=1765401501&coll=portal&dl=ACM&FCID=10001540&FCTOKEN=4096727> Acesso em: 12 abr. 2003.

ANEXO A – DESCRIÇÃO DOS CASOS DE USO APPRENTI MAÎTRE

Abaixo estão as descrições funcionais dos casos de uso a serem implementados:

- **UC001 – Login**

Fluxo Principal	
Identificação	UC001
Nome	Login
Atores	Usuário
Tipo	Secundario
Descrição	Usuário ao abrir o sistema, deve informar seu usuário e senha para ter acesso ao conteúdo do sistema de acordo com o seu nível de permissão.
Pré-condições	Usuário deve estar cadastrado no sistema.
Pós-condições	O usuário deve alterar seu estado para logado.
Seqüência de Eventos	
Seqüência Típica de Eventos	
Ação do Ator	Resposta do Sistema
1. O usuário digita a usuário e senha	
2. Usuário clica no botão Entrar	3. O sistema valida os dados digitados pelo usuário.
	4. O sistema redireciona o usuário para a página inicial de acordo com seu perfil.

Seqüências alternativas	
Seqüência alternativa: 001-Dados de Login Incorretos	
Ação do Ator	Resposta do Sistema
	3.1. Sistema responde com a mensagem de dados invalidos.
Requisitos Não-Funcionais	
001 – O tempo de resposta do login deve ser inferior a 10 segundos.	

- **UC002 – Buscar Itens**

Fluxo Principal	
Identificação	UC002

Nome	Buscar Itens
Atores	Usuário
Tipo	Primário
Descrição	O usuário mediante as opções disponibilizadas pela interface do Sistema irá escolher suas opções de busca e solicitar ao sistema uma busca.
Pré-condições	O usuário deve estar logado no sistema.
Pós-condições	O usuário deve ter acesso a uma lista de itens de acordo com suas configurações.
Seqüência de Eventos	
Seqüência Típica de Eventos	
Ação do Ator	Resposta do Sistema
1. Monta uma pesquisa utilizando os filtros disponíveis.	
2. Solicita uma busca ao sistema clicando no botão Buscar.	3. De acordo com a busca selecionada faz uma requisição ao Sistema Maître (motor de recomendações) para que este processe os dados.
	4. O Maître retorna os dados e estes serão formatados e exibidos ao usuário para que o mesmo possa acessá-los, ou efetuar uma nova busca.

Seqüências alternativas	
Seqüência alternativa: 001-Dados não encontrados	
Ação do Ator	Resposta do Sistema
	3.1. Informa ao usuário que não foi possível encontrar dados para o mesmo.

- **UC003 – Acessar Item**

Fluxo Principal	
Identificação	UC003
Nome	Acessar Item
Atores	Usuário
Tipo	Secundário
Descrição	O usuário escolhe um item mediante a lista apresentada baseada na consulta feita.
Pré-condições	O sistema deve ter disponibilizado uma lista de itens ao usuário.

Pós-condições	O usuário terá acesso a mais informações do item escolhido.
Seqüência de Eventos	
Seqüência Típica de Eventos	
Ação do Ator	Resposta do Sistema
1. Escolhe um item para obter mais informações	2. Exibe em uma nova tela as informações do item escolhido dando a opção de voltar para a lista e de classificar o item.
	3. Pontua no servidor que o usuário acessou o item selecionado, fazendo com que o mesmo já seja diferenciado.

- **UC004 – Classificar Item**

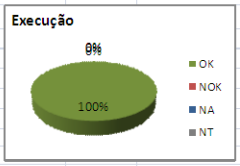
Fluxo Principal	
Identificação	UC004
Nome	Classificar Item
Atores	Usuário
Tipo	Secundario
Descrição	O usuário pode classificar um item acessado para que isto influencie em suas novas pesquisas.
Pré-condições	O usuário deve estar vendo os detalhes de um item escolhido.
Pós-condições	O item se torna classificado.
Seqüência de Eventos	
Seqüência Típica de Eventos	
Ação do Ator	Resposta do Sistema
1. Clica em Classificar Item	
2. Informa a classificação mediante as opções disponíveis	3. Atribui a classificação feita ao Perfil do Usuário junto ao servidor Maître.
	4. Direciona o usuário para os detalhes do item novamente.

ANEXO B – CASOS DE TESTE APPRENTI MAÎTRE

- **UC001 – Login**

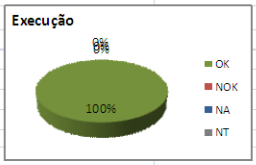
Caso de Teste: UC001 - Login

Andamento - Passos



RESULTADO EXECUÇÃO - PASSOS	
OK	6
NOK	0
NA	0
NT	0

Andamento - Cenários



RESULTADO EXECUÇÃO - CENÁRIOS	
OK	3
NOK	0
NA	0
NT	0

Descrição: Cenário Feliz - *Usuário ao abrir o sistema, deve informar seu usuário e senha para ter acesso ao conteúdo do sistema de acordo com o seu nível de permissão.*

Pré-Requisitos:

Usuário deve estar cadastrado no sistema.

O usuário deve alterar seu estado para logado.

CENÁRIO	PASSO	DESCRIÇÃO	DADOS	RESULTADOS ESPERADOS	OK/NOK/NA/NT	RESULTADO
1	1	O usuário digita a usuário e senha	Login: maitre Senha: 1234	Validar os dados digitados pelo usuário.	OK	OK
	2	Usuário clica no botão Acesar		O sistema apresentar a mensagem "Bem Vindo" e redirecionar o usuário para a página inicial de acordo com seu perfil.	OK	

Descrição: Dados de Login Incorretos - *Usuário ao abrir o sistema, deve informar seu usuário e senha para ter acesso ao conteúdo do sistema de acordo com o seu nível de permissão.*

Pré-Requisitos:

Usuário deve estar cadastrado no sistema.

O usuário deve alterar seu estado para logado.

CENÁRIO	PASSO	DESCRIÇÃO	DADOS	RESULTADOS ESPERADOS	OK/NOK/NA/NT	RESULTADO
2	1	O usuário digita a usuário e senha			OK	OK
	2	Usuário clica no botão Entrar	Login: joaodasilva Senha: 1234	Sistema responde com a mensagem de dados invalidos "Usuário inexistente".	OK	

Descrição: Dados de Login Incorretos - *Usuário ao abrir o sistema, deve informar seu usuário e senha para ter acesso ao conteúdo do sistema de acordo com o seu nível de permissão.*

Pré-Requisitos:

Usuário deve estar cadastrado no sistema.

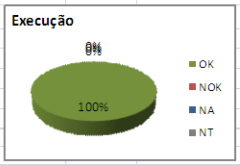
O usuário deve alterar seu estado para logado.

CENÁRIO	PASSO	DESCRIÇÃO	DADOS	RESULTADOS ESPERADOS	OK/NOK/NA/NT	RESULTADO
3	1	O usuário digita a usuário e senha			Ok	OK
	2	Usuário clica no botão Entrar	Login: maitre Senha: abcd	Sistema responde com a mensagem de dados invalidos "Senha incorreta".	Ok	

- **UC002 – Buscar Itens**

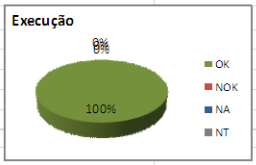
Caso de Teste: UC002 - Buscar Itens

Andamento - Passos



RESULTADO EXECUÇÃO - PASSOS	
OK	4
NOK	0
NA	0
NT	0

Andamento - Cenários



RESULTADO EXECUÇÃO - CENÁRIOS	
OK	2
NOK	0
NA	0
NT	0

Descrição: Cenário Feliz - *O usuário mediante as opções disponibilizadas pela interface do Sistema irá escolher suas opções de busca e solicitar ao sistema uma busca.*

Pré-Requisitos:

Usuário deve estar logado no sistema.

O usuário deve ter acesso a uma lista de itens de acordo com suas configurações.

CENÁRIO	PASSO	DESCRIÇÃO	DADOS	RESULTADOS ESPERADOS	OK/NOK/NA/NT	RESULTADO
1	1	Digitar palavras-chave	Palavra-Chave: Filé	Palavra(s)-chave no campo desejado	OK	OK
	2	Clica no botão Entrar		Retorna os dados e estes serão formatados e exibidos ao usuário para que o mesmo possa acessá-los, ou efetuar uma nova busca.	OK	

Descrição: Dados não encontrados - *O usuário mediante as opções disponibilizadas pela interface do Sistema irá escolher suas opções de busca e solicitar ao sistema uma busca.*

Pré-Requisitos:

Usuário deve estar logado no sistema.

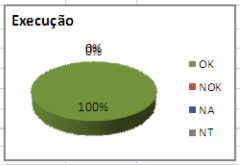
O usuário deve ter acesso a uma lista de itens de acordo com suas configurações.

CENÁRIO	PASSO	DESCRIÇÃO	DADOS	RESULTADOS ESPERADOS	OK/NOK/NA/NT	RESULTADO
2	1	Digitar palavras-chave	Palavra-Chave: Filé futebol kaigang inglaterra violão mandela	Palavra(s)-chave no campo desejado	OK	OK
	2	Clica no botão Entrar		Apresentação da Mensagem "Dados não encontrados"	OK	

- **UC003 – Acessar Item**

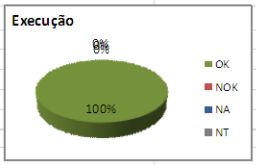
Caso de Teste: UC003 - Acessar Item

Andamento - Passos



RESULTADO EXECUÇÃO - PASSOS	
OK	4
NOK	0
NA	0
NT	0

Andamento - Cenários



RESULTADO EXECUÇÃO - CENÁRIOS	
OK	2
NOK	0
NA	0
NT	0

Descrição: Cenário Feliz - *O usuário mediante as opções disponibilizadas pela interface do Sistema irá escolher suas opções de busca e solicitar ao sistema uma busca.*

Pré-Requisitos:

Usuário deve estar logado no sistema e ter feito uma busca.

O usuário terá acesso a mais informações do item escolhido.

CENÁRIO	PASSO	DESCRIÇÃO	DADOS	RESULTADOS ESPERADOS	OK/NOK/NA/NT	RESULTADO
1	1	Escolher um item para obter mais informações		Exibe em uma nova tela as informações do item	Ok	OK
	2	Visualização da possibilidade de Classificação		Botões "Sim" e "Não" junto a mensagem "Essa informação foi útil"	Ok	

Descrição: Dados não encontrados - *O usuário mediante as opções disponibilizadas pela interface do Sistema irá escolher suas opções de busca e solicitar ao sistema uma busca.*

Pré-Requisitos:

Usuário deve estar logado no sistema.

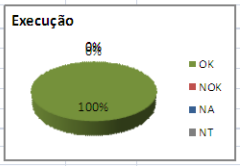
O usuário deve ter acesso a uma lista de itens de acordo com suas configurações.

CENÁRIO	PASSO	DESCRIÇÃO	DADOS	RESULTADOS ESPERADOS	OK/NOK/NA/NT	RESULTADO
2	1	Digitar palavras-chave	Palavra-Chave: Filé futebol kaigang inglaterra violão mandela	Palavra(s)-chave no campo desejado	Ok	OK
	2	Clica no botão Entrar		Apresentação da Mensagem "Dados não encontrados"	Ok	

- **UC003 – Acessar Item**

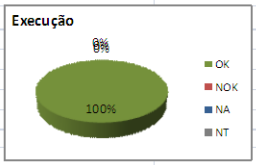
Caso de Teste: UC004 - Classificar Item

Andamento - Passos



RESULTADO EXECUÇÃO - PASSOS	
OK	2
NOK	0
NA	0
NT	0

Andamento - Cenários



RESULTADO EXECUÇÃO - CENÁRIOS	
OK	2
NOK	0
NA	0
NT	0

Descrição: Cenário Feliz - *O usuário pode classificar um item acessado para que isto influencie em suas novas pesquisas.*

Pré-Requisitos:

Usuário deve estar logado no sistema e deve estar vendo os detalhes de um item escolhido.

O item se torna classificado.

CENÁRIO	PASSO	DESCRIÇÃO	DADOS	RESULTADOS ESPERADOS	OK/NOK/NA/NT	RESULTADO
1	1	Clica em Classificar Item (botão "Sim")		Exibe mensagem "Classificação realizada com sucesso"	Ok	OK

Descrição: Cenário Feliz - *O usuário pode classificar um item acessado para que isto influencie em suas novas pesquisas.*

Pré-Requisitos:

Usuário deve estar logado no sistema e deve estar vendo os detalhes de um item escolhido.

O item se torna classificado.

CENÁRIO	PASSO	DESCRIÇÃO	DADOS	RESULTADOS ESPERADOS	OK/NOK/NA/NT	RESULTADO
3	1	Clica em Classificar Item (botão "Não")		Exibe mensagem "Classificação realizada com sucesso"	Ok	OK