INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Sistema de Recomendação: Share GFS Games, Films & Series

JOSÉ FRANCISCO DE JESUS DA SILVA MARTINS ADEGAS
Outubro de 2013

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Share GFS – Games, Films & Series

Sistema de Recomendação

José Francisco de Jesus da Silva Martins Adegas

Dissertação para obtenção do Grau de Mestre em Engenharia Informática, Área de Especialização em Tecnologias do Conhecimento e da Decisão

Orientador: Doutor Paulo Oliveira

Júri:

Presidente:

[Nome do Presidente, Categoria, Escola]

Vogais:

[Nome do Vogal1, Categoria, Escola]

[Nome do Vogal2, Categoria, Escola] (até 4 vogais)

Porto, Outubro 2013

Dedicatória

Esta dissertação de mestrado é dedicada à minha família pelo facto de me terem apoiado em todas as decisões académicas que tomei e por terem feito todos os respectivos sacrifícios de boa e livre vontade.

Resumo

Esta dissertação incide sobre o estudo e análise de uma solução para a criação de um sistema de recomendação para uma comunidade de consumidores de media e no consequente desenvolvimento da mesma cujo âmbito inicial engloba consumidores de jogos, filmes e/ou séries, com o intuito de lhes proporcionar a oportunidade de partilharem experiências, bem como manterem um registo das mesmas. Com a informação adquirida, o sistema reúne condições para proceder a sugestões direccionadas a cada membro da comunidade.

O sistema actualiza a sua informação mediante as acções e os dados fornecidos pelos membros, bem como pelo seu feedback às sugestões. Esta aprendizagem ao longo do tempo permite que as sugestões do sistema evoluam juntamente com a mudança de preferência dos membros ou se autocorrijam. O sistema toma iniciativa de sugerir mediante determinadas acções, mas também pode ser invocada uma sugestão directamente pelo utilizador, na medida em que este não precisa de esperar por sugestões, podendo pedir ao sistema que as forneça num determinado momento.

Nos testes realizados foi possível apurar que o sistema de recomendação desenvolvido forneceu sugestões adequadas a cada utilizador específico, tomando em linha de conta as suas acções prévias. Para além deste facto, o sistema não forneceu qualquer sugestão quando o histórico destas tinha provado incomodar o utilizador.

Palavras-chave: Sistema de recomendação, Tipificação de utilizadores, Inteligência Artificial, Manipulação e tratamento de incerteza, Redes de Bayes.

Abstract

This dissertation focuses on the study and analysis of a solution for the creation of a recommendation system of a media consumer community and the consequent development of the system whose initial scope encompasses consumers of games, movies and/or series, in order to provide them with the opportunity to share experiences as well as keep track of them. With the acquired information, the system meets the conditions to adjust suggestions to every member of the community.

The system updates its information through the actions and data provided by the members of the community, as well as from the feedback of previous suggestions. This learning over time allows the system's suggestions to evolve along with the changing preferences of members. The system takes the initiative to suggest after certain actions, but also a suggestion may be invoked directly by the user, since he does not need to wait for advice, he can ask the system to provide a suggestion anytime.

In the tests performed, it was found that the recommendation system developed provided suggestions appropriate to each specific user, taking into account its prior actions. Beyond this, the system does not provide any suggestion when the history of them proven to bother the user.

Keywords: Recommendation System, Grading of users, Artificial Intelligence, Handling and treatment of uncertainty, Bayes networks.

Agradecimentos

Agradeço ao Dr. Paulo Oliveira pela disponibilidade, atenção e motivação que me prestou sempre que necessário.

Queria ainda agradecer a todos os bons Engenheiros que tive o prazer de ter como professores durante todo o meu percurso académico.

Agradeço por fim à minha família, pois sem o seu apoio e motivação não me teria sido possível atingir os meus objectivos.

Índice

Lista de Figuras

Lista de Tabelas

Acrónimos e Símbolos

Lista de Acrónimos

GFS	Games, Films and Series
RNA	Rede neuronal artificial
Jogos FPS	*Jogos First Person Shooter*
WCF	Windows Communication Foundation

1 Introdução

Neste primeiro capítulo é pretendido apresentar o ambiente em que se insere a presente dissertação, declarar os objectivos da mesma e ainda resumir brevemente a sua estrutura.

1.1 Enquadramento

No âmbito do entretenimento caseiro, tornou-se cada vez mais vulgar e generalizado o consumo de jogos, filmes e séries. Isto aumenta, portanto, toda essa área de negócio que se vem expandido rapidamente nas últimas duas décadas.

Com este crescimento exponencial, esta área vai-se diversificando abrangendo cada vez mais necessidades e passando a ser um passatempo de cada vez mais consumidores. Assim sendo, existe uma comunidade crescente mas dispersa com vários interesses comuns.

Neste sentido, é de relevância uma plataforma que reúna esta comunidade, aprendendo com ela e que lhe forneça informação interessante em tempo útil, tomando em conta os gostos e interesses de cada utilizador.

A plataforma tem de cobrir vários tipos de dispositivos actuais, como sendo portáteis, telemóveis, tablet's, etc. e deve possuir uma interface comum às suas aplicações, para que os utilizadores se sintam confortáveis com a mesma. Desta forma, a plataforma insere-se nos dispositivos e sistemas que os potenciais "utilizadores alvo" já possuem e conhecem aumentando assim as suas probabilidades de sucesso.

Pretende-se que uma significativa percentagem dos utilizadores sinta uma necessidade diária de aceder à plataforma para receber sugestões, por exemplo a visualização de um episódio que saiu no próprio dia de uma série que o utilizador segue. Essa assiduidade permite à plataforma adquirir informação suficiente para melhorar progressivamente a sua interacção com cada utilizador.

Esta plataforma para além da componente sugestiva/informativa permite manter um registo do que o utilizador consumiu, para futura pesquisa. Permite pesquisar informação referente a outros utilizadores que tenham dado o seu consentimento. Essa informação passa pelos seus filmes, séries e jogos e permite ainda a um utilizador partilhar uma determinada informação com outro utilizador que não seja necessariamente um utilizador da plataforma em particular.

1.2 Objectivos

Com o desenvolvimento desta dissertação é suposto, depois de uma fase intensiva de análise, desenvolver um sistema de recomendação que cumpra os seguintes objectivos:

- Criar uma infra-estrutura que permita o desenvolvimento de uma comunidade de consumidores de media.
- Criar um sistema de sugestões adaptadas aos interesses dos utilizadores.
- Permitir o apoio na tomada de decisão sobre novos artefactos de media.
- Permitir que o sistema aprenda com a interacção com os utilizadores.

1.3 Estrutura do documento

Segue-se uma descrição dos diversos capítulos que compõem esta dissertação.

No Capítulo da Introdução enquadra-se o tema, descrevem-se os objectivos inerentes à elaboração deste trabalho de mestrado bem como a estrutura do presente documento.

No Capítulo do Estado da Arte são feitos os levantamentos tecnológicos e científicos necessários para apoiar a decisão sobre quais as tecnologias mais apropriadas à resolução do problema em causa. Pretende-se neste capítulo salientar essas características bem como proceder à sua comparação, possibilitando a sua escolha numa fase posterior.

No Capítulo de Análise é demonstrado o processo pelo qual se passou antes de começar o desenvolvimento do sistema de recomendação. São apresentados os requisitos que o sistema tem de possuir, as escolhas das tecnologias apresentadas no estado da arte e que serão usadas no desenvolvimento do sistema, a arquitectura que o sistema deve ter para cumprir os objectivos do trabalho, o modelo de dados que é necessário para armazenar os dados relevantes e de controlo que o sistema manipulará. São, ainda, apresentados os requisitos de hardware e software, é apresentado um estudo sobre os sistemas detentores de informação acerca de artefactos media (jogos, filmes e séries) que são relevantes para a pesquisa necessária para validar dados ou dar respostas aos utilizadores. É ainda neste capítulo que é dado a conhecer o utilizador-alvo do sistema de recomendação e neste âmbito é definida uma persona representativa, são descritos os tipos de utilizadores, a sua segmentação e tratamento de incerteza quanto à apresentação personalizada de sugestões.

No Capítulo do Desenvolvimento da componente servidora e após completa a análise sobre o problema em questão será apresentada a componente responsável pela parte de cálculo e de dar resposta aos pedidos dos diversos utilizadores, são apresentados os serviços disponibilizados e as API's utilizadas nos mesmos.

Depois de concluída a componente de cálculo, segue-se o Capítulo do Desenvolvimento da componente cliente, uma componente mais apelativa sobre a interface que se pretende que o sistema disponibilize aos utilizadores e será esta que influenciará ou não o sucesso deste sistema. Serão aqui apresentadas as suas características e o ambiente sugestivo da aplicação.

No sexto Capítulo Testes e Experiências serão apresentados alguns casos de estudo que permitem o teste de algumas funções do sistema, nomeadamente as sugestões encontradas para o utilizador em causa e a oportunidade de as apresentar ou não.

Por fim no Capítulo Conclusões vão ser abordados quais os objectivos alcançados, as principais dificuldades na realização desses objectivos, bem como as limitações da solução apresentada. Por fim, ficam algumas notas sobre possíveis desenvolvimentos futuros.

2 Estado da Arte

Após a apresentação dos objectivos da dissertação, segue este capítulo onde será abordado o estado da arte. Neste foi feita uma introdução aos sistemas de recomendação, um levantamento sobre sistemas de recomendação com semelhanças ao proposto, foi ainda realizado um levantamento tecnológico sobre as interfaces que possam colmatar os objectivos declarados e foi por fim elaborado um levantamento científico sobre manipulação de incerteza, visto que esta faz parte do problema que se pretende abordar.

2.1 Sistemas de recomendação

Os sistemas de recomendação são uma sub-área da aprendizagem máquina e debruçam-se sobre as preferências, hábitos e escolhas dos utilizadores com o intuito de lhes sugerir algo. Estas sugestões podem também advir de relacionamentos que o utilizador tenha com grupos de utilizadores ou com outros sites do género. Estes sistemas podem ser normalmente encontrados em sites de compra/venda de items, por exemplo E-Bay, como é ilustrado na Figura 1. Outros usos para esta tecnologia podem ser sites de pesquisa, visualização de e-mails ou sistemas especificos [Souza, 2012].

Figura 1 – Exemplo de um sistema de recomendação

Actualmente a quantidade de informação que pode ser relevante para os utilizadores é enorme, mas nem toda a informação tem a mesma relevância para todos os utilizadores. O desafio reside em escolher que informação sugerir a que utilizador ou grupo de utilizadores. Para este efeito existem três principais técnicas: a filtragem de informação que se baseia nas preferências do utilizador, a filtragem baseada em conteúdo que leva em conta o histórico e a tipificação do utilizador e a filtragem colaborativa que consiste na recomendação de itens que pessoas com gosto semelhante preferiram no passado [Cazella *et al*., 2005].

2.1.1 Personal Movie

Os seres humanos baseiam-se em dicas dadas por amigos, na opinião de especialistas, nas sinopses ou noutras fontes como, por exemplo, nas redes sociais para tomarem a decisão sobre qual o filme que querem ver. Um sistema de recomendação deve ser capaz de reproduzir estas recomendações que já são normais entre seres humanos, para isso este terá de conhecer de forma implícita ou explicita os interesses do utilizador. O sistema Personal Movie é um modelo de sistema de recomendação de filmes geolocalizados em eventos. Este sistema surge da necessidade de fornecer aos turistas opções de filmes nos cinemas na sua área geográfica, mas fazendo-o tomando em conta as características de filmes que agradariam o turista em questão. Consiste num sistema desenvolvido para telemóvel que usa a geolocalização e a personalidade das pessoas para melhorar a qualidade das suas

recomendações. Os testes realizados mostram resultados promissores, na medida em que as suas sugestões foram assertivas [Nunes *et al*., 2012].

2.1.2 Max da Netflix

A Netflix, uma empresa norte-americana líder na TV por internet, desenvolveu um sistema de recomendação designado Max. Este sistema tem como finalidade indicar ao utilizador filmes, quando este não sabe o que assistir. O sistema utiliza um questionário para chegar às recomendações que posteriormente apresenta. Este foca-se em questões como: os géneros ou temas preferidos, os actores ou editores favoritos, etc. Após o questionário, o Max apresenta uma lista de opões e o utilizador pode atribuir notas e preferências que ajudam o sistema a dar uma recomendação final. O Max também tem a opção de recomendar um filme sem indagar o utilizador, recorrendo ao histórico de conversação do mesmo [Souza, 2013].

O sistema que se propõe neste trabalho de mestrado, embora tenha semelhanças com os dois sistemas de recomendação apresentados, possui objectivos bem diferentes, para além de ser mais abrangente.

2.2 Interface

Em qualquer aplicação que interaja com utilizadores humanos a sua interface é de grande relevância, visto que o diálogo com a aplicação é feito através desta. Uma má interface pode levar ao declínio de uma aplicação com uma óptima componente de cálculo e representação do negócio. Neste caso a aplicação destina-se a utilizadores consumidores de media pelo que uma interface de linha de comandos ou uma interface de menus está fora de questão. A interface tem portanto de ser gráfica e ser construída numa linguagem que permita a sua utilização em diversos tipos de dispositivos. Neste sentido seguem-se duas subsecções que apresentam duas possíveis abordagens gráficas.

2.2.1 Ambiente Java Swing

As aplicações desenvolvidas em Java possuem a característica de serem executadas dentro de uma máquina virtual (JVM – Java Virtual Machine). Esta característica faz com que as aplicações nela desenvolvidas sejam multiplataforma, bastando existir a dita máquina virtual para essa plataforma, para que a aplicação ser compatível [Eric, 2012].

O Java Swing, componente gráfica da linguagem Java, permite a personalização dos objectos gráficos (botões, caixas de texto, imagens, etc). No entanto, esta personalização realizada dentro dos parâmetros normais, faz com que a aplicação tenha um aspecto um pouco convencional [Bellotti *et al*., 2010], como se pode ver na Figura 2.

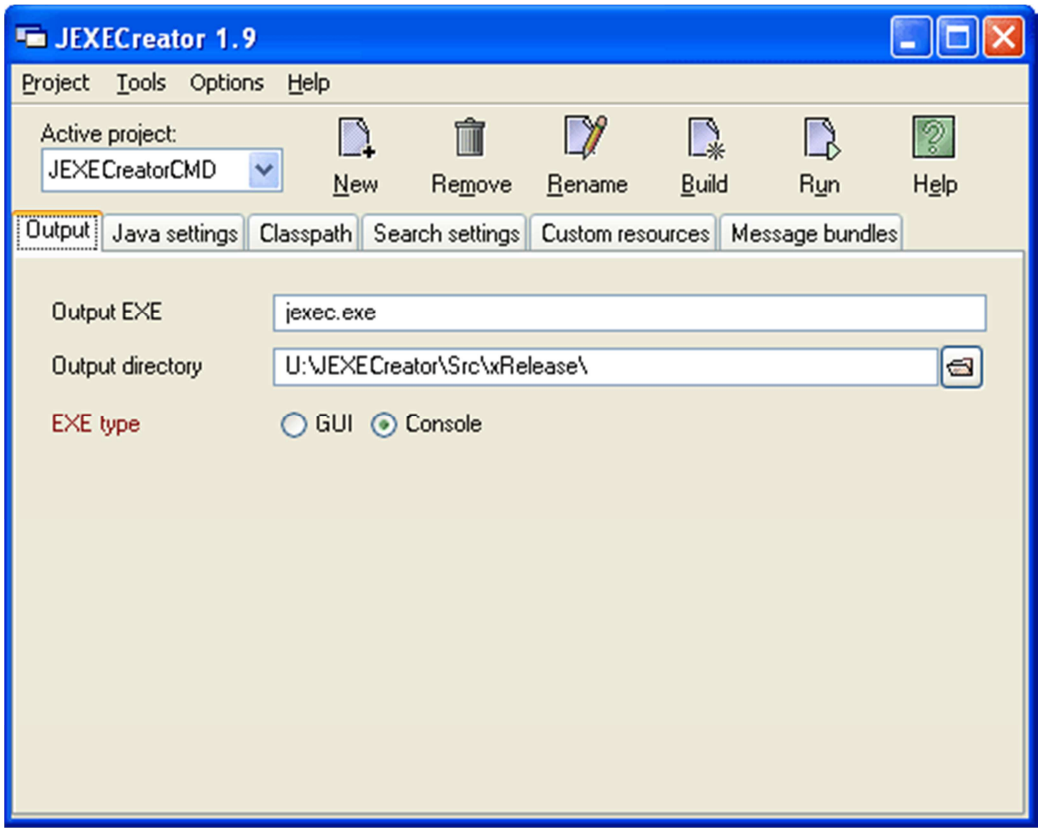

Figura 2 – Exemplo de uma aplicação Java convencional

Por outro lado se a personalização dos objectos gráficos for exagerada pode causar desconforto à maioria dos utilizadores, na medida em que estes não conseguem enquadrar o modelo usado com nenhum dos modelos mentais que o utilizador já possui. Exemplo disso é a interface da aplicação apresentada na Figura 3. Ainda que esteja bem estruturada e possua ajudas, corre o risco de não agradar a utilizadores que tenham pouca predisposição para despender tempo a entender a interface.

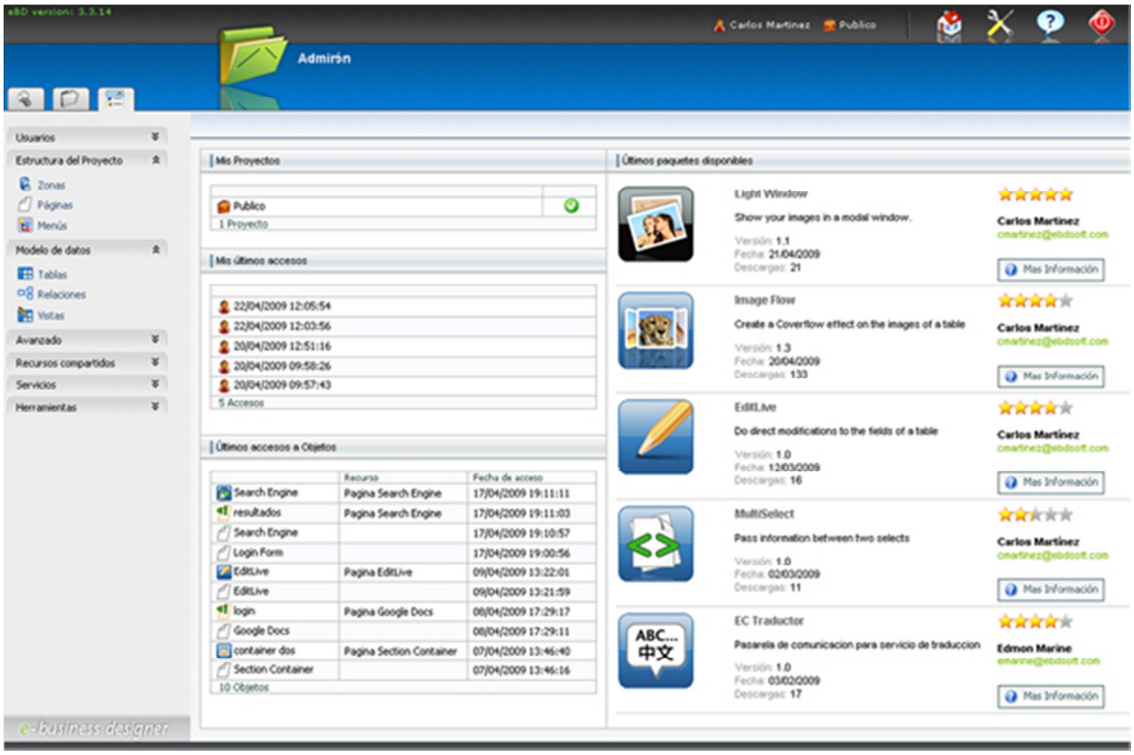

Figura 3 – Exemplo de uma aplicação Java não convencional

2.2.2 Ambiente Metro

As aplicações metro são aplicações cuja parte gráfica é desenvolvida utilizando a linguagem de construção topológica desenvolvida pela Microsoft para media *center*, mostrando inicialmente os seus primeiros traços em sistemas como o "Microsoft XP Media Center Edition". Contudo o seu vislumbre completo surgiu apenas em 2006 com o lançamento dos produtos da Zune (marca de media digital lançada pela Microsoft). Estes princípios de programação de interfaces foram usados para construir as interfaces dos telemóveis da Microsoft e mais recentemente em finais de 2012 foi utilizado na construção da componente gráfica do sistema operativo Windows 8 [Wiki, 2012].

O Windows 8 introduziu as mudanças mais radicais ao nível de interface desde o Windows 95. Quando se entra no sistema este apresenta um ambiente metro, ou seja, uma tabela de tiles (*icons* dinâmicos) que lançam as aplicações Metro ou invés do habitual desktop. Este por sua vez não passa de uma aplicação metro que uma vez lançada tem um aspecto muito similar ao do Windows 7. Este novo sistema operativo foi desenhado a pensar nos *Tablets* e logo no seu modelo *touchscreen* e embora funcione também com o rato e teclado, enquanto o utilizador não se habitua, o seu uso pode ser um pouco frustrante [Cross *et al*., 2012].

A linguagem de interface metro é uma linguagem moderna e orientada ao conteúdo, mostrando-o de uma forma topológica e hierarquizada. Esta forma de apresentar a informação pode ser um pouco difícil de adaptar para quem já possui um modelo mental de

uma outra forma mais vulgarizada, mas para quem inicia é bastante intuitiva, simples e organizada [Robbins, 2012].

A linguagem de design Metro foi desenvolvida com o propósito de rentabilizar o espaço disponível dos visores e de aumentar a produtividade na utilização das aplicações (como se pode visualizar na Figura 4), retirando-lhe barras e botões desnecessários para que todo o visor fosse preenchido com informação de conteúdo e toda a informação de controlo ficasse escondida [Oliveira, 2012]

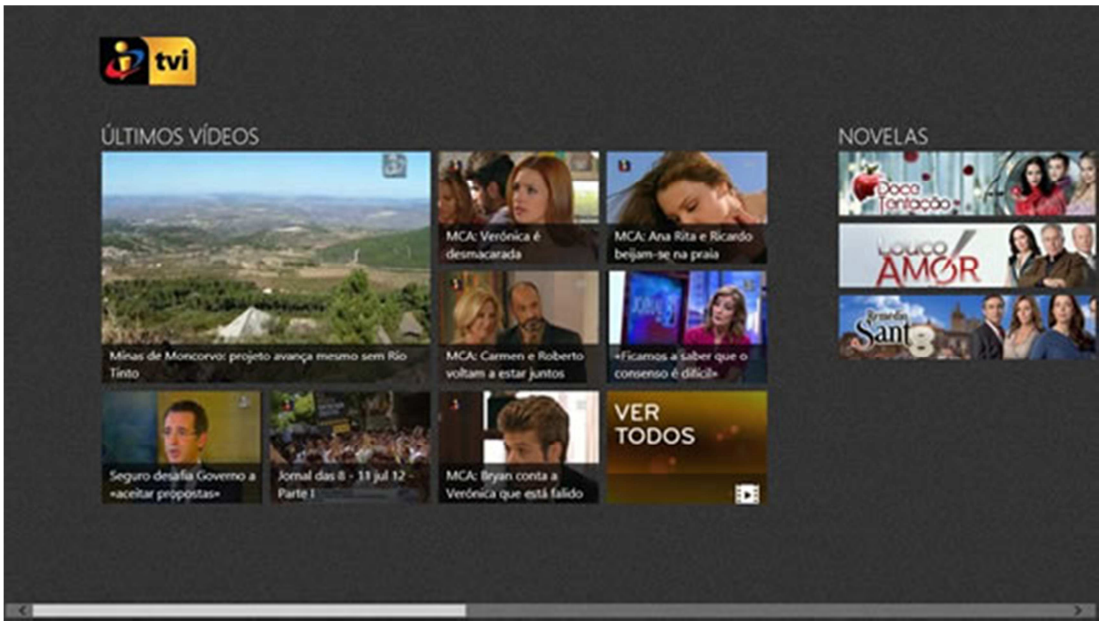

Figura 4 – Exemplo de uma aplicação metro

Como plataforma de desenvolvimento para as aplicações metro o Microsoft Visual Studio 2013 incorpora, não só a capacidade de apoio ao desenvolvimento, como também variados templates, para que programadores menos familiarizados com a linguagem possam desenvolver a partir de uma base sólida, ou pelo menos possam perceber a ideia inerente à interacção entre os componentes na linguagem Metro. O Visual Studio permite por isso que os programadores escolham a sua linguagem de programação preferida para o desenvolvimento do núcleo da aplicação e que independentemente da linguagem escolhida possam usar a linguagem metro para a interface gráfica da mesma [MSDN, 2013] .

2.2.3 Comparação entre as tecnologias de interface estudadas

Na tabela 1 que se segue são apresentadas as considerações acerca das duas abordagens às interfaces gráficas apresentadas.

	Java Swing	Metro
Multiplataforma/ Multi-dispositivo	Permite	Permite
Livre	Sim	Sim
Comum	Sim	Sim
Apelativa	Difícil	Fácil

Tabela 1 — Considerações sobre as tecnologias de ambientes gráficos

2.3 Raciocinar sobre incerteza

Por vezes os sistemas contêm dados que não oferecem certeza sobre um determinado aspecto pertinente à realização de uma tarefa. Nestes casos o sistema terá de raciocinar sobre conhecimento ausente ou incompleto e/ou relações causa-efeito não determinísticas para descobrir os dados que pretende.

Neste tipo de sistemas incertos, normalmente sistemas reais, é conveniente usar ou desenvolver uma tecnologia para lidar de forma adequada com o problema em questão. Neste sentido e como será relevante o seu estudo para o sistema em causa, passa-se a analisar várias tecnologias desenvolvidas ou adaptadas para o efeito, sendo estas a lógica difusa, as Redes de Bayes, um modelo multi-agente e os factores de certeza. [Faria, 2012a]

2.3.1 Lógica Fuzzy

Por vezes a forma como se expressa o conhecimento é confusa ou ambígua. Este facto advém de que para as pessoas que possuem conhecimento do domínio é compreensível o significado

de termos como “o prazo está muito apertado” ou “diminui mais um pouco a pressão”. Mas este tipo de especificação não é fácil de representar num sistema, pois os computadores não lidam bem com ambiguidade de valores, por exemplo “diminuir um pouco a pressão” pode representar uma diminuição de 4,5 ou 6 unidades.

A lógica difusa (fuzzy) consegue resolver este problema pois foi desenvolvida para lidar com esta imprecisão na informação, baseada na teoria dos conjuntos difusos, admite que todas as coisas podem ser representadas em graus de certeza. [Faria, 2012b]

Um sistema com interface Fuzzy consiste num conjunto de regras “se-então” definidas por cima de conjuntos Fuzzy. Os conjuntos Fuzzy generalizam o conceito dos conjuntos tradicionais permitindo que o grau de adesão de um membro seja qualquer valor entre zero e um. Isto facilita resolver problemas reais na medida em que a pertença de um membro a um grupo é maior do que a sua pertença a outro podendo assim transformar uma designação ambígua na pertença a um conjunto concreto. [Castillo e Melin, 2008]

Um modelo Fuzzy consiste em três etapas principais a *fuzzificação*, a inferência e a *deffuzificação*. A *fuzzificação* consiste em converter as variáveis linguísticas (entrada) em variáveis subjectivas (do modelo) e na definição das suas funções de pertença. Esta etapa engloba análise do problema, definição de variáveis, definição das funções de pertença e criação das regiões. Segue-se a etapa de inferência na qual são definidas e examinadas as regras paralelamente. Esta etapa engloba a definição das proposições, a análise das regras e a criação da região resultante. Por fim, segue-se a etapa de *defuzzificação* na qual se converte as regiões resultantes em variáveis de saída. É nesta etapa que se relacionam as regiões fuzzy com o valor de saída esperado do sistema. [Faria, 2012c]

A técnica de Fuzzy tem sido usada em diversas áreas, por exemplo na área do comércio. As técnicas tradicionais baseiam-se nos indicadores técnicos construídos com o preço e quantidade passados e geram indicadores (recomendações) sobre compra/venda. Contudo basear recomendações apenas no preço e quantidade passados é proceder a raciocínio sobre conhecimento incompleto e retorna valores que podem ser difíceis de interpretar. Neste sentido um modelo Fuzzy pode ser usado para determinar quando negociar e como negociar, fazendo uso de indicadores como vontade ou capacidade do mercado e devolvendo informação como “aguardar”, “venda forte” ou “venda muito forte”. O objectivo principal é reduzir a incerteza sobre as recomendações fornecidas ao utilizador. [Nikola e Ramazan, 2013]

2.3.2 Redes de Bayes

Aquando da necessidade de trabalhar com conhecimento incerto ou incompleto, uma das possíveis abordagens é fazer uso da teoria das probabilidades estudada pelo Reverendo Thomas Bayes e apresentada dois anos após a sua morte em 1763 [Bayes e Price, 1763].

As redes Bayesianas foram criadas no início da década de oitenta para resolver problemas de predição e abdução, como sendo uma maneira prática de representar a teoria das

probabilidades. Desta forma era possível os programas tomarem ilações sobre problemas que envolvessem incerteza, munindo-os de um certo raciocínio sobre problemas pré determinados [Pinto, 2005].

Nestas redes Bayesianas existem variáveis iniciais que quando relacionadas entre si possibilitam a chegada a conclusões finais ou intermédias que por sua vez podem também, estas últimas, gerar conclusões finais ou intermédias e assim sucessivamente [Azevedo, 2013].

As ligações que existem entre as variáveis iniciais levam a um dado resultado com uma determinada probabilidade. Estas probabilidades têm de ser definidas na rede de forma não aleatória, caso contrário a rede não teria grande utilidade, visto que são estas que permitem à rede retirar conclusões com algum grau de certeza [Chagas, 2007].

As redes Bayesianas são portanto representadas através de grafos acíclicos onde os nós representam as variáveis de um determinado problema (domínio) e os arcos representam as interacções condicionais entre os nós. Entre cada relação nós-filhos e nó-pai existe uma probabilidade associada, sendo a probabilidade daquela relação entre nós filhos concluir o nó pai [Matsuura, 2003].

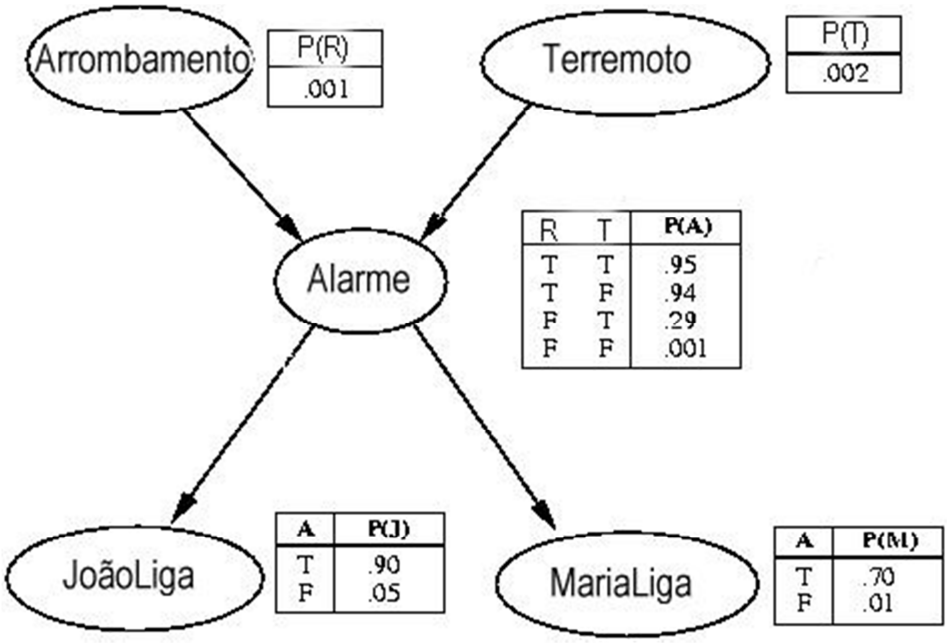

Figura 5 – Exemplo de uma rede Bayesiana

A obtenção das probabilidades a inserir na rede pode vir de diversos locais dependendo do problema que a rede pretende representar. Exemplo desses locais seria por exemplo um perito no domínio em estudo ou dados históricos previamente recolhidos [Camarinha, 2009].

Depois da rede estar definida esta ajusta-se facilmente à nova informação permitindo verificar rapidamente o impacto de uma determinada alteração na mesma. Por exemplo, como se

pode ver na Figura 5, se a probabilidade de acontecer Terramoto no nó da figura se alterar de 0.002 para 0.5 a probabilidade do João ou da Maria serem alertados sobe imediatamente.

Actualmente esta tecnologia tem sido usada em problemas relacionados com a Web como por exemplo na protecção inteligente contra SPAM [Gulyás, 2006].

Existem plataformas que possibilitam aos programadores a utilização das redes de Bayes de uma forma simplificada, bastando defini-la e usá-la, visto que a plataforma procede a todos os cálculos necessários. Um exemplo destas plataformas é a plataforma Microsoft Belief Networks. Existem bibliotecas que permitem que outras aplicações utilizem as plataformas adicionando informação às redes de Bayes e lendo os resultados produzidos por estas, permitindo às aplicações lidarem com a incerteza sem terem de proceder a cálculos [Microsoft Research, 2010].

2.3.3 Factores de Certeza

Um dos sistemas periciais mais conhecido é o MyCin [Harmond e King, 1995]. Este sistema apoia a decisão dos médicos no tratamento de infecções bacterianas. Quando os engenheiros de conhecimento deste sistema tentaram extrair o conhecimento dos peritos aperceberam-se que as suas respostas eram subjectivas e sem suporte estatístico, por isso desenvolveram a tecnologia factores de certeza. Nesta tecnologia as regras cujo resultado interessa, assim como as regras que lhe dão origem possuem um factor de certeza que não é percentual mas sim um valor decimal variante entre -1 a 1. No qual valores negativos representam descrença daquela regra ou do resultado da mesma, enquanto valores positivos representam uma crença acerca da veracidade da regra se concretizar e 1 representa a certeza quanto à veracidade do resultado. Ou seja, o factor de certeza não é probabilidade nem é um valor de certeza, mas mede a precisão/confiança do resultado mediante as premissas apresentadas (regra) [McCarthy, 1984].

Na extracção de conhecimento, por exemplo de um perito, este pode dizer que dada a premissa A (o serviço de previsão de meteorologia afirma que vai chover) e a premissa B (o céu está escuro) o factor de certeza de acontecer a acção T (chover) é de 0,9. Esta regra possui portanto uma crença positiva bastante elevada.

O Mycin foi desenvolvido no início da década de 70 tendo uma duração de desenvolvimento de vários anos e contando com o apoio de vários especialistas. Possui uma base de conhecimento bastante completa representada sob a forma de aproximadamente 450 regras. Na altura da sua criação foram feitos estudos que revelaram que com o uso de factores de certeza, a alteração de um factor numa regra causa um impacto mínimo na performance do sistema. Tornando-se por isso uma tecnologia ideal a usar visto que a quantidade de regras era elevada. Este sistema teve um desempenho tão bom que chegou a superar o desempenho de alguns especialistas [Shortliffe, 1976].

Estes sistemas de atribuição de crença tem vindo a ser usado em muitas áreas, por exemplo na produção, onde as regras podem ter factores de certeza com valor -1 que representa a impossibilidade de um escalonamento de máquinas e 1 representa a certeza absoluta acerca da possibilidade de execução completa do escalonamento. Estas regras vão sendo actualizadas nas premissas para permitir a propagação da actualização para todos os resultados que tenham sido encontrados levando em linha de conta as premissas actualizadas [Zhaa *et al*.,, 2001].

2.3.4 Rede Neuronal

As redes neuronais artificiais (RNA) são modelos simplificados do sistema neuronal do ser humano. É, portanto, uma estrutura de nós e ramos extremamente interligada. Os nós da rede são unidades de cálculo que representam os neurónios humanos e os ramos são as ligações entre esses nós que são responsáveis pelo *input* e *output* dos mesmos [Cortez e Neves, 2000].

A RNA pode ser vista então como um processador eminentemente paralelo, composto por simples unidades de processamento que possui uma propensão natural para armazenar conhecimento empírico e torná-lo acessível ao utilizador. O conhecimento é adquirido através do seu ambiente utilizando um processo de aprendizagem e é armazenado nas conexões à semelhança do cérebro humano [Cortez e Neves, 2000].

Esta tecnologia é muito usada para proceder a classificações de dados não catalogados. Um exemplo desta classificação é por exemplo a classificação de músicas, ou seja, mediante um trecho da música a rede pode classificá-la como sendo de um determinado género musical [Malheiro *et al*., 2004].

As redes neuronais são portanto uma tecnologia capaz de raciocinar o que permite trabalhar com conhecimento incerto e/ou impreciso. Para tal é necessário configurar a rede para que esta conheça o método de classificar os dados que recebe e sobre eles tirar ilações. A rede vai aprendendo com as interacções com o meio ambiente e vai auto ajustando os seus pesos para que os seus resultados estejam dentro das espectativas. Um exemplo prático pode ser o uso destas redes na análise da incerteza na representação de classes temáticas [Sabo, 2006].

2.3.5 Modelo Multi-agente

Uma arquitectura multi-agente é uma arquitectura composta por agentes que podem ser componentes de software ou hardware que possuem um conjunto de características, sendo as mais comuns a autonomia, a capacidade de interagir socialmente, a reacção e a pro-actividade [Wooldridge, 1995]. Os sistemas baseados em agentes foram-se diversificando ao longo dos anos, com um especial crescimento na década de 90 e neste momento estão inseridos em muitas áreas como sendo a robótica, o controlo de tráfego, o controlo de linhas

de produção, etc. Pode ser implementado com um sistema supervisor, ou seja, existe um sistema que controla as entradas e saídas dos agentes, podendo distribuir a carga entre estes. Em alternativa pode ser implementado sem sistema supervisor, ou seja, cada agente tem de tomar em conta não só os seus próprios objectivos como também os da comunidade. [Ramos e Silva, 2011] .

Dependendo do problema de raciocinar com incerteza, pode ser vantajoso usar a tecnologia multi-agente, por exemplo, caso faça sentido dividir as variáveis por entidades e neste caso cada entidade seria representada por um agente que geria apenas as suas variáveis. Exemplificando, esta tecnologia foi usada num sistema que pretendia dar suporte a negociações de preços. Estes preços são estipulados mediante informação informal, advinda muitas vezes de locais não fidedignos. Portanto as empresas, aqui representadas por agentes têm de raciocinar sobre a informação imprecisa que dispõem e sem terem a certeza dos custos que as empresas concorrentes vão suportar, ou que lucro estas pretendem ter. Tem de praticar um preço que embora não seja máximo tem de ser apelativo à empresa e ao mesmo tempo competitivo. Para isso, cada agente calcula os factores de certeza e socorre-se de um sistema pericial para proceder aos cálculos [Lia e Sheng, 2010] .

Embora no exemplo acima cada agente represente uma entidade idêntica, a arquitectura multiagente pode ser usada em sistemas cujas entidades pertençam a modelos destintos. No caso de problemas no âmbito do planeamento em linhas de montagem é possível ter agentes a controlar diversos factores, por exemplo para criar as várias estratégias de escalonamento que permitam satisfazer os vários departamentos (vendas, produção, recursos humanos, etc.) que muitas vezes têm objectivos que colidem com os objectivos dos outros departamentos, é conveniente ter uma de duas situações: ou existem agentes coordenadores ou agentes de negociação para além de ter agentes que representem as máquinas de produção e agentes que representem ordens de entrada. Embora numa arquitectura deste tipo nem todos os agentes possam ter de lidar com incerteza, existem agentes que necessitam, como por exemplo, a incerteza sobre a satisfação dos departamentos mediante um plano de escalonamento, ou a incerteza sobre os cumprimentos de restrições temporais [Coudert *et al*., 2010].

2.3.6 Comparação entre as tecnologias de raciocínio sobre incerteza estudadas

Na Tabela 2 que se segue são apresentadas algumas considerações, relevantes ao problema em análise, sobre as tecnologias que se debruçam no estudo da manipulação de incerteza apresentadas no presente capítulo.

	Lógica Fuzzy	Rede de Bayes	Factores Certeza	Rede Neuronal	Modelo Multi-Agente
Grandes quantidades de dados	Permite	Permite	Permite	Permite	Permite
Percepção da lógica usada	Fácil	Fácil	Fácil	Difícil	Difícil
Lida com dados imprecisos	Bem	Bem	Bem	Razoavelmente	Razoavelmente
Aprendizagem	Difícil	Fácil	Fácil	Fácil	Fácil
Definição matemática	Não precisa	Precisa	Precisa	Não precisa	Não precisa
Dados históricos	Necessita	Necessita	Necessita	Necessita	Não necessita

Tabela 2 — Considerações sobre as tecnologias de apoio à decisão

3 Análise

Dando por concluído o estudo sobre as tecnologias e ciências relevantes para a realização deste trabalho de mestrado, segue-se a análise de um sistema on-line que visa servir uma comunidade dispersa de utilizadores caseiros que requerem e enviam informação sobre artefactos de media.

3.1 Requisitos

A aplicação deve possuir uma interface familiar ao utilizador e inserida no ambiente que este usualmente utiliza, para que de uma forma fácil e rápida este possa aceder ao que necessita. Este requisito é muito importante na medida em que pode ser decisivo no sucesso ou insucesso do sistema, visto que este foca uma comunidade de consumidores caseiros de media que podem possuir um nível de conhecimento reduzido sobre tecnologia.

A componente de cálculo vai necessitar de métodos que produzam dicas para os utilizadores tendo em conta o que eles já consumiram e as suas preferências. Estas dicas terão de ser analisadas com vista a descobrir se há a possibilidade de ser uma mais valia apresentá-las como sugestões ao utilizador, ou se é preferível ignorá-las.

O sistema tem de possuir um método de feedback e um histórico de acções que lhe permita guardar informação sobre a interacção com o utilizador, possibilitando assim ao sistema inferir sobre essa informação para adequar progressivamente as suas acções a cada utilizador específico, aprendendo e tentando melhorar com cada interacção.

3.2 Opção de tecnologias estudadas no estado da arte

Após a apresentação dos requisitos do sistema é conveniente analisar a comparação das características sobre as tecnologias estudadas no estado da arte e proceder à escolha das mesmas.

Mediante uma reflexão sobre as características descritas na Tabela 1, que se pode encontra na secção “Comparação entre tecnologias de interface estudadas”, conclui-se que ambas as tecnologias (Java Swing e Metro) cumprem os requisitos necessários, ou seja, serem multiplataforma e estarem presentes como modelo mental do utilizador alvo. É objectivo deste projecto a criação de uma comunidade, pelo que a componente apelativa adquire grande relevância. Neste sentido opta-se pela tecnologia Metro como componente gráfica do sistema a desenvolver, visto que para além de apelativa é bastante uniforme entre aplicações e pode possuir ajudas do sistema integradas na própria aplicação, como sendo a função de pesquisa ou de partilha.

Mediante uma reflexão sobre as características descritas na Tabela 2, que se pode encontrar na secção “Comparação entre as tecnologias de raciocínio sobre incerteza estudadas“, e face a um estudo sobre as necessidades e características do problema em questão chegou-se à conclusão que este, idealmente, pode necessitar de manipular grandes quantidades de dados e que a percepção da lógica utilizada seria relevante para melhorar a performance do sistema no que respeita às sugestões apresentadas. Mais ainda, é notório que as tecnologias utilizadas necessitam de lidar com dados imprecisos e que necessitam de aprender ao longo do tempo.

Embora exista uma falta de dados históricos que seriam relevantes para a definição dos parâmetros de uma determinada tecnologia, usados para lidar com a informação imprecisa, esta característica foi desvalorizada, já que é suposto que a aplicação analise os dados que vai produzindo e/ou recebendo e ajuste os pesos respectivos automaticamente.

Em relação aos restantes atributos valorizados, grandes quantidades de dados, percepção da lógica usada, lidar com dados imprecisos, a aprendizagem e a definição matemática, as únicas tecnologias estudadas que contemplam todas as características analisadas são as Redes de Bayes e os Factores de Certeza.

3.3 Arquitectura

Sendo o sistema em causa divido pelos utilizadores dispersos geograficamente e por uma componente de cálculo centralizada, faz sentido que este deva ser decomposto numa aplicação cliente-servidor que faça uso de uma base de dados relacional devido à sua capacidade de acesso múltiplo, flexibilidade, integridade da informação e melhor gestão da mesma.

Tal como se pode visualizar na Figura 6, o sistema é então composto pela componente servidora (base de dados e web-services) e pela componente cliente (aplicação metro) a ser instalada em cada dispositivo cliente.

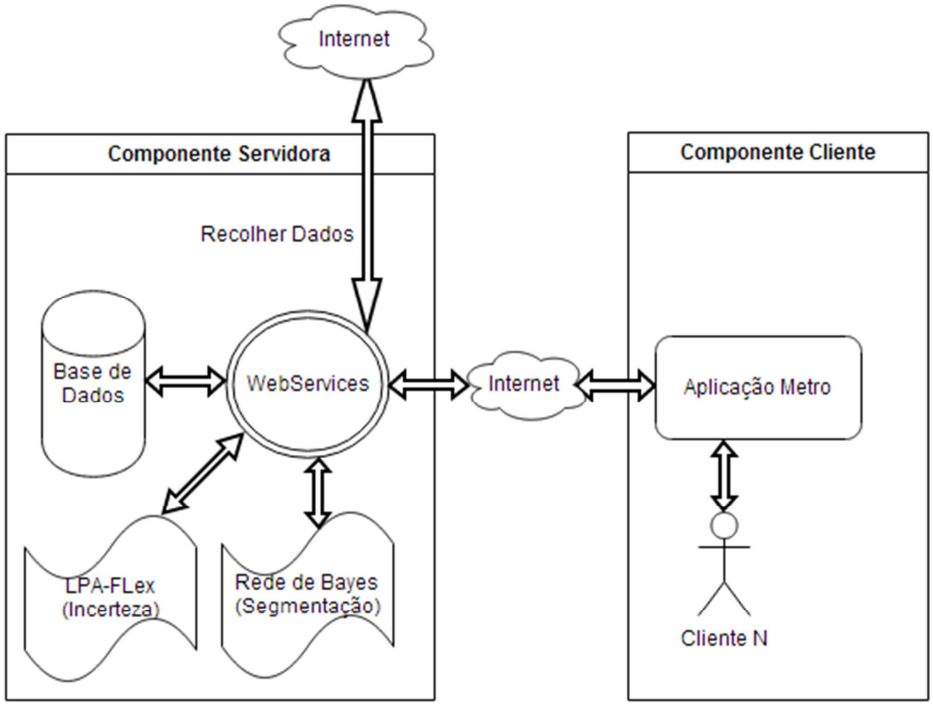

Figura 6 – Diagrama da arquitectura

O utilizador interage apenas com a sua aplicação metro e esta interage com os *webservices* da componente servidora. Estes *webservices* fazem uso de outros componentes, como sendo a base de dados para guardar toda a informação necessária, acesso a sites na Internet para recolher nova informação que possa interessar aos utilizadores e armazena-a na base de dados. Para além disso, acedem também a uma rede de Bayes sempre que um utilizador executa uma determinada acção que possa alterar o seu tipo de preferências e, por fim, acedem também à componente que possui as regras de incerteza e que é responsável pelos cálculos para executarem sugestões com um determinado grau de certeza.

3.4 Modelo de Dados

Nesta secção apresenta-se o modelo de dados a ser implementado na base de dados relacional prevista anteriormente, elaborado para servir de suporte ao sistema que se pretende desenvolver. Este modelo não é considerado final, na medida em que após a análise dos dados históricos do sistema, quando este estiver em produção, pode ser pertinente proceder a alterações na própria estrutura de dados. Para além deste facto, é intenção do autor alargar o âmbito do sistema, pelo que será necessário adicionar tabelas para abranger novos tipos de media, por exemplo, música.

De notar que as relações entre tabelas não estão representadas por uma questão de simplificação do modelo. Estas apenas estão perceptíveis pelas chaves primárias e secundárias.

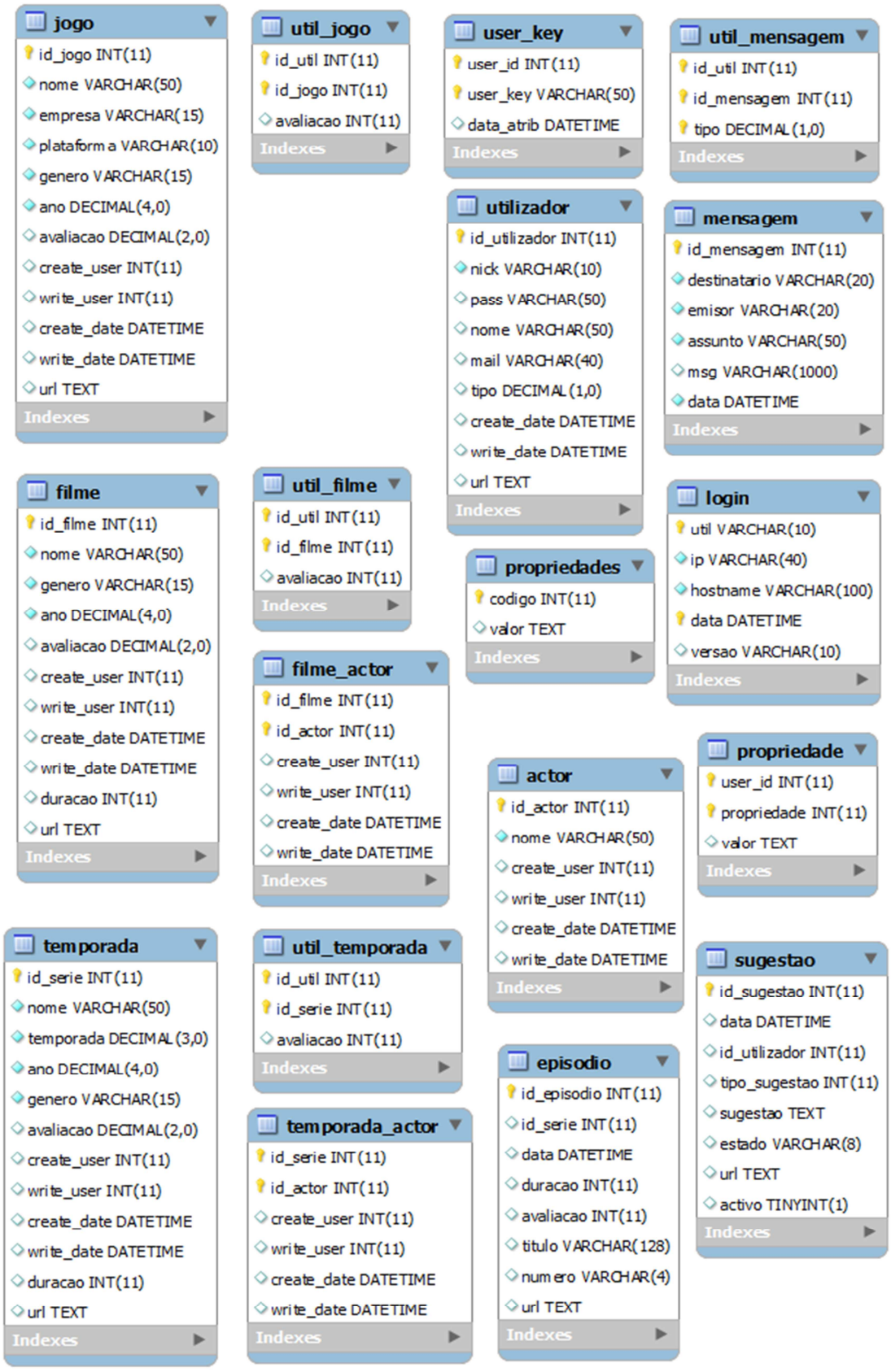

Figura 7 – Modelo de dados

No modelo de dados existem tabelas que representam objectos do sistema e outras que são tabelas de controlo. Como tabelas do modelo de negócio temos a representar os Jogos (jogo), os Filmes (filme e filme_actor), as séries (temporada, temporada_actor e episodio), os actores (actores), as mensagens (mensagem), os utilizadores (utilizador), a relação entre os utilizadores e os restantes objectos (util_jogo , util_fime, util_temporada,util_mensagem) e para o objecto sugestão (sugestao). Como tabelas utilizadas para controlo de acesso dos utilizadores (login e user_key), para controlo de acções dos utilizadores (propriedades) e para controlo e características do próprio sistema (propriedade).

3.5 Requisitos de Software e Hardware

Na componente servidora é recomendado um sistema operativo Microsoft Windows Server 2008 com a framework .Net 4.5 instalada e o motor de base de dados MySQL. Esta componente deve ter uma largura de banda adequada à quantidade de utilizadores activos pelo que tende a ser variável e crescente. Os requisitos de hardware desta componente podem começar por ser baixos, visto que enquanto o número de utilizadores for reduzido, o tráfego de informações e a realização de cálculos são também eles reduzidos, sendo no entanto esperado que os administradores de sistema analisem periodicamente os indicadores de utilização de hardware para poderem actuar em conformidade com as necessidades.

Nas componentes clientes os sistemas têm de possuir um ambiente metro como por exemplo o Microsoft Windows 8. Ao nível de requisitos de hardware estes são pouco significativos visto que o processamento relevante é feito na componente servidora.

3.6 Sistemas detentores de informação acerca de media

O sistema GFS necessita de um ou mais sistemas detentores de informação não estática, ou seja, sistemas cuja informação sobre media seja constantemente actualizada. Esta informação passa por exemplo pelo titulo, imagem, actores, tipo, duração, etc.

Se o sistema conseguir aceder autonomamente à informação sempre actualizada, este mantem-se apto a obter sugestões interessantes para os seus utilizadores, sem grande esforço por parte dos seus administradores.

Numa primeira análise foi feita uma pesquisa sobre sites que disponibilizavam Torrents1 na medida em que estes são criados com muito pouco atraso após o artefacto de media ter sido lançado para o mercado. No entanto, esta abordagem levanta problemas visto que estes sites normalmente possuem transferências de dados protegidos por direitos de autor e por essa razão são muitas vezes encerrados, pelo que se torna um sistema pouco interessante na medida em que pode deixar de existir sem possibilidade de antecipação.

Outra abordagem que se seguiu foi o uso de Blogs. Existem no universo da Internet inúmeros Blogs que se dedicam a alojar links que permitem fazer o download dos artefactos, usando estes como chamariz para atrair tráfego e depois utilizando anúncios para realizar capital. Estes Blogs não alojam nenhum artefacto, apenas disponibilizam os links para os mesmos o que os torna imunes à ilegalidade contra os direitos de autor, mas por outro lado são normalmente geridos por particulares que sem aviso prévio podem decidir encerrar o Blog e isso torna esta opção também pouco fiável.

Numa tentativa de encontrar fiabilidade foi estudada a opção de usar motores de busca como por exemplo o Google, mas neste caso foi encontrado o problema da actualização. Exemplificando, se sair um novo episódio de uma série, este irá ser compactado, enviado para o servidor de dados o qual disponibilizará um link que posteriormente será adicionado a um ou mais blogs que serão posteriormente pesquisados pelos Bots (Robôs Web) dos motores de pesquisa e só depois serão encontrados nas suas pesquisas. Ora estas operações consomem algum tempo e esta opção torna-se pouco aliciante.

Na espectativa de aceder à informação de uma forma fiável e ao mesmo tempo atempada foi analisado o sistema IMDB. Este funciona como um site e dedica-se apenas a transmitir informação sobre artefactos media como jogos, filmes e séries. Mais focado nestes dois últimos, estes sistemas procuram lançar a informação mesmo antes de o artefacto ser lançado ao público e fá-lo adicionando a data prevista de lançamento o que é óptimo para dar sugestões atempadas aos utilizadores do sistema a desenvolver. O IMDB mune ainda cada artefacto de mais informação relevante para o sistema como imagem, actores, sinopse e até avaliação dada pelos seus utilizadores.

3.7 Utilizadores

Os utilizadores deste sistema serão essencialmente consumidores finais que procuraram nele uma ou mais das seguintes funcionalidades:

- Registo de Games, Films e Series (GFS) adquiridos/experimentados

[1] Um ficheiro torrent é um ficheiro de computador que contém meta-dados sobre os ficheiros e/ou pastas a serem distribuídos. Geralmente, também possui uma lista dos locais de rede de rastreadores, que são os computadores que ajudam os participantes do sistema a encontrarem-se mutuamente.

- Pesquisa das opiniões/avaliações acerca de um GFS
- Auxílio na tomada de decisão sobre o próximo GFS a experimentar
- Partilha de experiências adquiridas sobre GFS
- Receber informação acerca do lançamento de novos GFS que lhe interessem

Estes utilizadores serão obrigatoriamente utilizadores de um sistema operativo que incorpore o ambiente metro, pelo que é suposto que a interface com o utilizador siga as normas e as boas práticas de design das aplicações metro para que o modelo mental que os utilizadores possuam seja válido no uso do programa. Assim poupa-se trabalho mental ao utilizador e minimiza-se o risco de este desistir de utilizar o sistema.

A estratégia para a intensificação e generalização da utilização deste sistema será a "palavra passa palavra" pelo que é esperado que um dado utilizador tenha amigos ou conhecidos que também usem o programa. Neste sentido é requerido que um utilizador do sistema possa ver as associações e avaliações de outro utilizador específico, desde que esse utilizador tenha dado permissões para tal.

3.7.1 Persona

Na concepção de um sistema cuja interface é construída para ser utilizada por seres humanos é importante não esquecer que cada um destes tem as suas próprias características e o que é bom para um pode não o ser para outro. Neste sentido e para não se correr o risco de desenvolver um sistema que até pode ser muito bom ao nível de arquitectura mas a interface não se adequa ao nosso público-alvo, condenando assim o projecto, é importante desenvolver tendo como base as características gerais do público-alvo.

A persona deve possuir características demográficas, psicográficas e topológicas, como sendo as suas necessidades, o risco à mudança, o conhecimento de outros produtos e/ou serviços.

Nesse sentido para a criação da interface do "Share GFS", programa cliente do sistema GFS imagina as necessidades de uma pessoa específica, embora fictícia, cujas necessidades e características se enquadram com as necessidades e características do público-alvo da aplicação.

Neste sentido foi criada a persona de nome Ricardo Ribeiro, com 26 anos e residente no Porto. Este utiliza o Windows 8 e as aplicações metro nele embutidas já com alguma destreza. O Ricardo vê alguns filmes, segue muitas séries e gosta de jogar jogos FPS – First Person Shooter[2]. Procura uma aplicação que seja gratuita e integrada no sistema que actualmente utiliza, que lhe permita de forma fácil e de acordo com o seu esquema mental de aplicações,

[2] FPS é um género de jogo de computador e consolas, no qual se vê apenas o ponto de vista do protagonista, como se o jogador e personagem do jogo fossem o mesmo observador.

gerir o seu lazer caseiro, partilhar essa informação com amigos, encontrar novos filmes, séries e jogos de seu agrado e ser alertado para novos episódios das séries que segue.

3.7.2 Tipos de utilizadores

Os utilizadores não são todos iguais e como tal não têm as mesmas preferências pelos géneros ou tipos de GFS. Nesse sentido é suposto que o programa use o histórico do utilizador bem como a informação recolhida deste para proceder a sugestões de qualquer tipo de artefacto. Outro ponto a ter em conta na selecção de sugestões a apresentar ao utilizador é o quanto este está familiarizado com o universo GFS, ou seja, por exemplo se o utilizador é um consumidor casual de séries, o programa deve sugerir que veja um novo episódio de uma série que este já consome, mas não deve sugerir outra série ainda que esta tenha imensas semelhanças com a série consumida actualmente.

Para possibilitar a tipificação das sugestões aplicadas aos utilizadores, estas são divididas de forma transparente pelos três grupos que se seguem.

- Jogos
- Filmes
- Temporadas

Os utilizadores não se apercebem desta segmentação pois esta é gerida pelo sistema e serve apenas para a tipificação das sugestões. À medida que o utilizador consome e actualiza a informação acerca dos GFS's consumidos, este vai evoluindo podendo mudar de segmento e consequentemente de tipo de sugestões.

3.7.3 Segmentação de utilizadores (Rede de Bayes)

A rede de Bayes definida para a caracterização das preferências dos utilizadores (Figura 8) usa como nós iniciais a percentagem de cada jogo, filme e série sobre o total de todos os GFS's declarados pelo utilizador. São ainda usados como nós iniciais da rede a quantidade de partilhas e pesquisas feitas em cada um dos tipos de artefactos. Estes nós levam ao cálculo da percentagem à qual o utilizador pertence aos grupos intermédios (G,F,S) que por sua vez dão origem a um resultado final sobre a pertença do utilizador ao grupo de Jogos, Filmes e Séries.

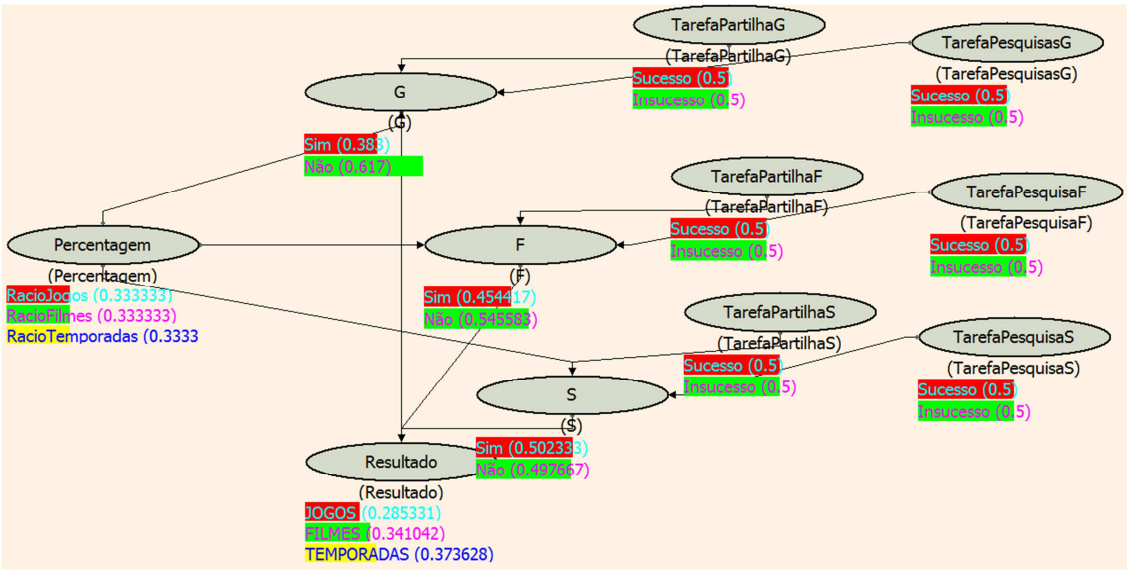

Figura 8 – Rede de Bayes – Tipo de utilizador

Na Figura 9 é possível visualizar que cada grupo de nós intermédios da rede de Bayes possui uma percentagem que corresponde a uma pertença a cada grupo final. Por falta de histórico e/ou perito acerca deste conhecimento, estes valores foram definidos segundo uma reflecção sobre essas probabilidades e estão sujeitos a rectificações caso o histórico do próprio sistema demonstre indícios de que os mesmos estão incorrectos. Para este efeito foi desenvolvido um método que calcula a percentagem de jogos, filmes e séries global e a cada tipo adiciona-lhe o respectivo rácio das tarefas partilhadas (rácio_partilha=partilhas_sucesso/(partilhas_sucesso+ partilhas_insucesso)) e o rácio das pesquisas para esse tipo. No fim deste processo são conhecidos três valores, cada um representando um tipo de artefacto, estes valores são comparados e são ajustadas as respectivas crenças na rede de Bayes.

Parent Node(s)			Resultado			
G	F	S	JOGOS	FILMES	TEMPORADAS	bar charts
Sim	Sim	Sim	0,333	0,333	0,333	
		Não	0,45	0,45	0,1	
	Não	Sim	0,45	0,1	0,45	
		Não	1,0	0,0	0,0	
Não	Sim	Sim	0,04	0,48	0,48	
		Não	0,0	1,0	0,0	
	Não	Sim	0,0	0,0	1,0	
		Não	0,333	0,333	0,333	

Figura 9 – Rede de Bayes – Percentagens

A rede de Bayes é utilizada pelos *webservices* definidos na componente servidora, sendo inicializada com as variáveis necessárias com informação vinda da base de dados relativa ao utilizador em questão. A informação advém de rácios entre os artefactos comentados pelo utilizador e de propriedades que reflectem, por exemplo, o número de pesquisas ou partilhas efectuadas como se pode visualizar no código que se encontra na Secção A1 do Anexo A. O sistema começa por inicializar as variáveis de acesso à base de dados e à rede de Bayes, depois procede à pesquisa sobre a informação pertinente àquele utilizador, e assim que a obtém, regista-a na rede de Bayes. Por fim o sistema lê as crenças da rede de Bayes retornando-as, como se pode ver no algoritmo "getUserSegmentation" que se segue.

De notar que é na tabela propriedades que são guardadas informações como, por exemplo, o total de tarefas de partilha com sucesso de um determinado utilizador e essa propriedade tem a chave 51, por isso quando no algoritmo que se segue, se escreve *bd.get_valor_da_propriedade51 (...)* é esse valor que se está a ler.

```
metodo getUserSegmentation(user_id)
inicio
        bd = iniciar_conector_base_de_dados;
        vetor resultado = novo vector[3];

        //carregar rede de bayes dos segmentos de utilizadores
        MSBN3Lib.Node n;

        n = modelAuto.ModelNodes["Percentagem"];

         int total_g = 0, total_f = 0, total_s = 0;
         total_g = bd.ler_total_jogos_do_utilizador
         total_f = bd.ler_total_filmes_do_utilizador
         total_s = bd.ler_total_temporadas_do_utilizador

        definir["RacioJogos"] = total_g / (total_g + total_f + total_s);
        definir["RacioFilmes"] = total_f / (total_g + total_f + total_s);
        definir["RacioTemporadas"] = total_s / (total_g + total_f + total_s);

        n = no_da_rede_de_bayes["TarefaPartilhaG"];
        s = bd.get_valor_da_propriedade51 (...)
        i = bd.get_valor_da_propriedade52( ...)
        n.definir["Sucesso"] = (s) / (s + i);
        n.definir["Insucesso"] = (i) / (s + i);

        n = no_da_rede_de_bayes["TarefaPartilhaF"];
        s = bd.get_valor_da_propriedade53 (...)
        i = bd.get_valor_da_propriedade54( ...)
        n.definir["Sucesso"] = (s) / (s + i);
        n.definir["Insucesso"] = (i) / (s + i);

        n = no_da_rede_de_bayes["TarefaPartilhaS"];
        s = bd.get_valor_da_propriedade55 (...)
```

```
        i = bd.get_valor_da_propriedade56( ...)
        n.definir["Sucesso"] = (s) / (s + i);
        n.definir["Insucesso"] = (i) / (s + i);

        n = no_da_rede_de_bayes["TarefaPesquisaG"];
        s = bd.get_valor_da_propriedade57 (...)
        i = bd.get_valor_da_propriedade58( ...)
        n.definir["Sucesso"] = (s) / (s + i);
        n.definir["Insucesso"] = (i) / (s + i);

        n = no_da_rede_de_bayes["TarefaPesquisaF"];
        s = bd.get_valor_da_propriedade59 (...)
        i = bd.get_valor_da_propriedade60( ...)
        n.definir["Sucesso"] = (s) / (s + i);
        n.definir["Insucesso"] = (i) / (s + i);

        n = no_da_rede_de_bayes["TarefaPesquisaS"];
        s = bd.get_valor_da_propriedade61 (...)
        i = bd.get_valor_da_propriedade62( ...)
        n.definir["Sucesso"] = (s) / (s + i);
        n.definir["Insucesso"] = (i) / (s + i);


        //ler crenças
        resultado[0] = ler_crença("Resultado", "JOGOS");
        resultado[1] = ler_crença("Resultado", "FILMES");
        resultado[2] = ler_crença("Resultado", "TEMPORADAS");

        retornar resultado;
fim
```

A informação da segmentação pode ser requisitada em várias situações como, por exemplo, no login do utilizador para determinar quais os tipos de sugestões a apresentar ou na actualização de dados (Secção A2 do Anexo A), ou seja, quando o utilizador realiza uma acção que pode influenciar o resultado da rede, podendo alterar o tipo de sugestões.

3.7.4 Tratamento de incerteza na apresentação de sugestões

Após o utilizador dar entrada no sistema é calculada a sua pertença aos diferentes segmentos de media (G,F,S). Após calculada essa pertença, o sistema tendo em conta a mesma, procura avisos/sugestões que façam sentido apresentar ao utilizador. Contudo existe uma incerteza sobre a pertinência de apresentar determinada sugestão ou aviso. Esta incerteza poderá surgir devido a vários factores. A título de exemplo, o utilizador pode previamente ter negado uma sugestão semelhante, tornando mais provável a sua falta de interesse na sugestão actual o que pode levar a que o sistema não a apresente desta vez.

As sugestões são divididas em categorias que as separam, não só pelo tipo de artefacto a que estas estão relacionadas, como também o tipo de sugestão, ou seja, se é uma sugestão de um artefacto idêntico ou uma nova versão de uma já existente. Por exemplo, o próximo filme de uma trilogia ou a nova versão de um jogo ou ainda um novo episódio de uma temporada de uma série.

Estas sugestões podem ser muito úteis ao utilizador, podendo por isso levar o sistema ao sucesso. Contudo se forem mal geridas, ou seja, se forem intrusivas ou insistentes, podem levar à fadiga do utilizador e causar ruptura no uso do sistema.

Com o intuito de tornar o sistema mais inteligente, para que este se adapte às necessidades do utilizador e não o perturbe com sugestões que não lhe interessam, foi desenvolvido em Prolog-Flex[3] um programa que incorpora um conjunto de regras responsáveis por tratar essa incerteza sobre a apresentação de sugestões através da técnica dos Factores de Certeza. Esta aplicação Prolog recebe como parâmetros de entrada variáveis (factores de certeza) que pertencem ao utilizador como, por exemplo, se este se interessa por jogos em relação a outros artefactos ou se tem aprovado sugestões sobre novos jogos em relação a todas as sugestões que tem recebido dessa categoria. Posto isto, infere qual o factor de certeza sobre a apresentação de uma determinada categoria de sugestões, como se pode visualizar no código da Figura 10, onde cada regra de incerteza relaciona várias variáveis e consoante os seus valores atribui um factor de certeza a uma opção da solução (retorno). Sendo que um exemplo de opção de retorno seria “mostrar jogos idênticos” com um factor de certeza de 0.75, com os valores a variarem entre -1 e 1. Os valores de certeza atribuídos às regras foram encontrados apenas usando ponderação e bom senso, devido à falta de dados que possibilitassem a sua definição com mais rigor. Contudo espera-se que com o decorrer do tempo a própria aplicação forneça dados históricos suficientes para ajustar estes pesos. Para isso foi desenvolvida uma rotina que verifica o rácio do somatório de cada tipo de sugestões apresentadas com o somatório por tipo das sugestões com feedback positivo. Se este rácio por cada tipo de sugestões estiver abaixo dos 50%, os pesos serão ajustados.

```
uncertainty_rule gfs_r1
if games is yes or new_identical_game is yes
then suggestion is show_identical_game
with certainty factor 0.75.

uncertainty_rule gfs_r2
if games is yes or new_version_of_game is yes
then suggestion is show_new_version_of_game
with certainty factor 0.9.

uncertainty_rule gfs_r3
if films is yes or new_identical_film is yes
then suggestion is show_identical_film
```

[3] Flex é um kit de ferramentas de sistemas muito poderosa e versátil. O Flex oferece uma solução baseada no conhecimento aberto para problemas de negócios e é implementado em Prolog, uma linguagem baseada em regras de alto nível, e tem acesso ilimitado ao poder da tecnologia subjacente.

```
with certainty factor 0.75.

uncertainty_rule gfs_r4
if films is yes or new_version_of_film  is yes
then suggestion is show_new_version_of_film
with certainty factor 0.9.

uncertainty_rule gfs_r5
if series is yes or new_identical_serie is yes
then suggestion is show_identical_serie
with certainty factor 0.75.

uncertainty_rule gfs_r6
if series is yes or new_version_of_serie is yes
then suggestion is show_new_version_of_serie
with certainty factor 0.9.

uncertainty_rule gfs_r7
if series is yes or new_episode is yes
then suggestion is show_new_episode
with certainty factor 0.99.

group gfs_group_rules gfs_r1,gfs_r2,gfs_r3,gfs_r4,gfs_r5,gfs_r6,gfs_r7.

action output do
 uncertainty_value_get( certainty_factor, suggestion,
                        show_identical_game, FC_SIG)
 and uncertainty_value_get( certainty_factor, suggestion,
                        show_new_version_of_game, FC_SNVOG)
 and uncertainty_value_get( certainty_factor, suggestion,
                        show_identical_film , FC_SIF)
 and uncertainty_value_get( certainty_factor, suggestion,
                        show_new_version_of_film , FC_SNVOF)
 and uncertainty_value_get( certainty_factor, suggestion,
                        show_identical_serie , FC_SIS)
 and uncertainty_value_get( certainty_factor, suggestion,
                        show_new_version_of_serie , FC_SNVOS)
 and uncertainty_value_get( certainty_factor, suggestion,
                        show_new_episode , FC_SNE)
 and write('FC_SIG: ') and write (FC_SIG) and write('~M')
 and write('FC_SNVOG: ') and write (FC_SNVOG) and write('~M')
 and write('FC_SIF: ') and write (FC_SIF) and write('~M')
 and write('FC_SNVOF: ') and write (FC_SNVOF) and write('~M')
 and write('FC_SIS: ') and write (FC_SIS) and write('~M')
 and write('FC_SNVOS: ') and write (FC_SNVOS) and write('~M')
 and write('FC_SNE: ') and write (FC_SNE) and write('~M').

relation run_gfs
(CF_games,CF_films,CF_series,CF_NIG,CF_NVOG,CF_NIF,CF_NVOF,CF_NVOS,
  CF_NIS,CF_NE)
if trace propagation
and reset all certainty_factor values
and the certainty_factor that the games is yes = CF_games
and the certainty_factor that the films is yes = CF_films
and the certainty_factor that the series is yes = CF_series
and the certainty_factor that the new_identical_game is yes = CF_NIG
and the certainty_factor that the new_version_of_game is yes = CF_NVOG
and the certainty_factor that the new_identical_film is yes = CF_NIF
and the certainty_factor that the new_version_of_film is yes = CF_NVOF
```

```
and the certainty_factor that the new_version_of_serie is yes= CF_NVOS
and the certainty_factor that the new_identical_serie is yes = CF_NIS
and the certainty_factor that the new_episode is yes = CF_NE
and propagate gfs_group_rules certainty_factor rules
and output.
```

Figura 10 – Determinar os factores de certeza (Prolog).

Na declaração de uma regra de incerteza na linguagem Prolog-Flex, esta começa com o token `uncertainty_rule` seguido do nome da regra. Posteriormente, define-se a estrutura condicional onde, mediante a comparação de alguma(s) condição(ções), se define o valor de uma(s) variável(eis) mediante um factor de certeza (`certainty factor).`

4 Desenvolvimento componente servidor

Após a conclusão da análise dá-se início ao desenvolvimento do sistema, começando pela componente que é responsável pelo controlo de dados e por servir de interface com as aplicações clientes, ou seja, a componente servidora.

A componente servidora destina-se a armazenar a informação na base de dados, vinda da Internet e dos clientes. Destina-se também a servir de interligação entre a base de dados e as aplicações clientes e a fazer diversificados cálculos de segurança e de operacionalidade necessários ou especificamente pedidos pelos clientes.

Para executar determinados cálculos, esta componente faz uso de aplicativos externos desenvolvidos especialmente para o efeito, retirando peso de processamento à aplicação.

4.1 Web Services

Alguns dos serviços web são responsáveis por receber os pedidos das aplicações metro clientes e por lhes dar as respectivas respostas. São ainda responsáveis pela gestão da informação que é transferida de e para a base de dados.

Os *webservices* são responsáveis por periodicamente acederem à Internet para recolher dados como, por exemplo, datas dos próximos episódios das últimas temporadas das séries ainda activas, ou recolher imagens de artefactos media (G,F,S) que ainda não as possuam.

Os serviços são ainda responsáveis por dividir os utilizadores em segmentos e fornecer-lhes sugestões mediante um grau de incerteza. Para essas sugestões são usados os dados extraídos da Internet, as preferências do utilizador em questão e o histórico da sua interacção com sugestões idênticas.

Em suma, os *webservices* são o núcleo do sistema na medida em que toda a informação passa por estes e são estes que a controlam e/ou geram.

Segue-se um descritivo dos serviços disponibilizados pela componente servidora:

- **login** – Recebe por parâmetro o utilizador, a password e a versão da aplicação cliente, procede ao início de uma sessão e devolve em caso de sucesso o identificador do utilizador, uma chave da sessão iniciada e a segmentação do utilizador.
- **logout** – Recebe por parâmetro o identificador do utilizador e a chave da sessão e procede à destruição da sessão.
- **addFilm; addFilmPartilha; addGame; addGamePartilha; addTemp; addTempPartilha**– Estes webservices recebem por parâmetro as credenciais do utilizador e os dados do media a inserir, insere-o caso ele ainda não exista e associa-o ao utilizador que o inseriu.
- **adminUpdateFilmPhoto; adminUpdateGamesPhoto; adminUpdateSeasonPhoto** – Estes webservices são de manutenção e podem ser chamados pelos administradores ou por acções agendadas e tem a funcionalidade de encontrar novas imagens para o media já existente.
- **getFilme; getGame; getTemp** – Recebe por parâmetro o nome do media e retorna os dados do mesmo.
- **getFilmeByID; getGameByID; getTempByID**– Recebe por parâmetro o ID do media e retorna os dados do mesmo.
- **getFilmes; getGames; getTemps** – Recebe por parâmetro as credenciais do utilizador e retorna uma lista id e titulo dos medias especificados.
- **updateFilme; updateGame; updateTemp** – Recebe por parâmetro as credenciais e os dados a actualizar e retorna sucesso ou insucesso.
- **getFilmeSearch; getGameSearch; getTempSearch**– Recebe por parâmetro o nome a pesquisar e pesquisa retornando os dados caso encontre. Método usado para utilizadores sem sessão iniciada.
- **getGeralData**– Quando invocado devolve propriedades gerais do sistema.
- **getMediaUsers**– Recebe um tipo de media e devolve dados sobre o respectivo media consumido pela comunidade, agrupado por utilizador.
- **getMyFilmsList; getMyGamesList; getMyTempsList**– Recebe por parâmetro as credenciais do utilizador e devolve a respectiva lista de media.
- **getTemps_Nums**– Devolve as diferentes temporadas de uma determinada série.
- **getUserSegmentation**– Recebe por parâmetro os indicadores do utilizador e calcula a sua pertença a cada um dos segmentos de media.
- **pesquisa**– Recebe por parâmetro a palavra a procurar e devolve qualquer tipo de media que contenha a respectiva palavra.
- **regUser** – Recebe por parâmetro os dados pessoais do utilizador e as suas preferências e procede ao seu registo no sistema.
- **updatePropriedade**– Recebe por parâmetro as credenciais e a propriedade a actualizar e retorna sucesso ou insucesso.
- **getDashBoardData**– Recebem por parâmetro as credenciais do utilizador e devolve a informação a apresentar na página inicial e cria previamente uma lista de sugestões tendo em conta o método **getUserSegmentation** e o método **getIncerteza** que devolve os factores de certeza que indicam ao sistema se deve ou não apresentar determinada sugestão.
- **getSuggestionList** - Devolvem as respectivas listas de sugestões geradas previamente pelo método **getDashBoardData**.

- getSuggestionByID – Devolve todo o conteúdo de uma determinada sugestão.
- setSuggestionState - Altera o estado das sugestões caso estas tenham sido lidas, ou tenham sido classificadas.

4.2 Sistema de autenticação

Quando um utilizador chega à aplicação pela primeira vez é-lhe pedido que se autentique ou que se registe, o que pode ser feito gratuitamente. No registo são pedidos opcionalmente dados sobre os interesses do utilizador que ajudam a segmentá-lo antes de se obter dados de interacção do mesmo. Caso estes dados não sejam preenchidos, as suas preferências serão as *standard* para todos os utilizadores até este começar a interagir com o sistema.

Após o registo, sempre que o utilizador tenta aceder à aplicação com os seus dados (utilizador e respectiva password) a aplicação metro invoca o *webservice* de login que procede às seguintes operações: verifica se o login é válido e em caso afirmativo regista o login para motivos administrativos; gera um token para aquele login específico e regista-a (a partir do login os pedidos são feitos indicando o identificador do utilizador e a chave gerada para o login para além de outros possíveis parâmetros); por último, é calculada a segmentação do utilizador e o método devolve o identificador do utilizador, a chave do login e a respectiva segmentação. Após o login é chamado o *webservice* DashBoard que carrega a página inicial com os artefactos de media, as possíveis sugestões e encontra-se preparado para apresentar outras informações sem necessidade de alterar a componente cliente.

4.3 Interligação entre Webservices e a Rede de Bayes

Os utilizadores podem utilizar a aplicação apenas para um dos artefactos, por exemplo, filmes, visto que podem não se sentir atraídos por séries ou jogos, mas também podem interessar-se por mais do que um tipo de media, pelo que se levanta a questão de qual destes o interessa mais e logo qual a probabilidade de este preferir sugestões de um tipo ou de outro.

Neste sentido os utilizadores podem ser classificados em segmentos, onde haverá a possibilidade de um utilizador pertencer apenas a um segmento, mas também haverá a possibilidade de este pertencer a todos os segmentos, ainda que na maior parte dos casos isto aconteça de forma desigual. Por exemplo um utilizador pode pertencer 50% ao segmento de séries, 40% ao segmento de filmes e 10% ao segmento de jogos. Ao analisar este exemplo repara-se que é provável que este utilizador valorize mais sugestões de séries ou filmes do que sugestões acerca de jogos.

Como é intuito do sistema não incomodar o utilizador com sugestões que provavelmente não lhe serão pertinentes, aquando da geração das mesmas, a segmentação do utilizador é consultada e caso o grau de pertença a um determinado grupo não seja relevante, não serão apresentadas sugestões acerca desse grupo. Como se pode ver no código que se encontra na Secção A3 do Anexo A, foi considerado que apenas se justificava apresentar sugestões de um determinado grupo se este tivesse uma pertença superior a 10% ao mesmo. Esta percentagem estará sujeita a alteração após haver dados históricos do sistema que permitam analisar se a percentagem é adequada ao seu propósito.

Embora a percentagem possa ser insuficiente para a apresentação de sugestões, desde que esta seja superior a 0%, as sugestões devem surgir ocasionalmente, para colmatar uma possível falta de precisão na própria percentagem, ou até mesmo para permitir que o utilizador possa divergir um pouco dos seus segmentos fortes. Neste sentido, após cada dez tentativas falhadas de procurar sugestões de um determinado segmento (GFS), devido à percentagem de segmentação do utilizador em causa, caso esta percentagem seja positiva a procura de sugestões é realizada. Para uma melhor interpretação sugere-se o código parcial apresentado na Secção A3 do Anexo A.

4.3.1 Segmentação inicial

Inicialmente se houver ausência de dados preferenciais, ou seja, se o utilizador não fornecer as suas preferências no momento do registo e se este ainda não tiver tecido considerações acerca de nenhum artefacto de media, o sistema terá de supor uma percentagem sobre a pertença a cada segmento que será igual para todos os utilizadores nestas circunstâncias. Esta percentagem foi atribuída de forma a espelhar o esperado dos utilizadores desta aplicação, como se pode ver na Figura 11. Contudo é aconselhável, após um determinado período de tempo, a análise de dados históricos para determinar se esta distribuição se aproxima da realidade ou se é aconselhável uma redistribuição. Esta análise e ajuste podem ser feitos através do método desenvolvido para o efeito, como descrito no capítulo da análise.

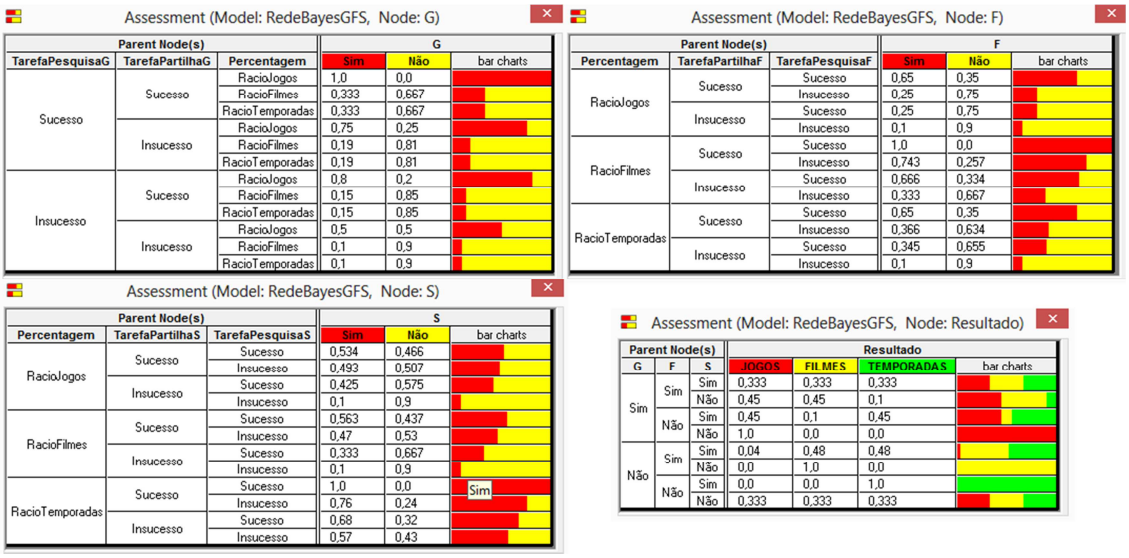

Figura 11 – Percentagens iniciais da segmentação

4.3.2 Actualização

A segmentação dos utilizadores é processada tendo em conta as seguintes variáveis de entrada: percentagem de artefactos comentados, tarefas de pesquisa de cada tipo de artefacto e tarefas de partilha dos mesmos. Como tal, a atribuição de segmentos é dinâmica e actualizada no momento, ou seja, não há necessidade de sair e voltar a entrar no sistema para uma acção ter reflexo na segmentação.

Para esta reacção imediata na segmentação, actualizam-se as respectivas propriedades aquando de cada acção e no fim da mesma recalcula-se a segmentação lendo todas as propriedades necessárias, por exemplo, quando um utilizador identificado pesquisa uma temporada/série com sucesso, ou seja com um match no sistema, este regista essa informação como se pode visualizar no código da Secção A4 do Anexo A.

No fim da apresentação do resultado da pesquisa, a aplicação cliente pede ao servidor para recalcular e atribuir as novas pertenças aos segmentos, como é visível no código da Secção A5 do Anexo A.

4.4 Incerteza

Quando um utilizador tenta aceder ao dashboard da aplicação cliente, os dados nele apresentados são resultado da chamada ao *webservice* `public string getDashBoardData(string user_id, string key)`, que retorna não só os dados relativos aos

artefactos do utilizador e da comunidade mas também pode retornar, caso existam, dados relativos a sugestões. Estas sugestões são geradas tendo em conta as preferências do utilizador em questão, a sua segmentação e o seu factor de incómodo[4], ou seja, em que medida é que sugestões idênticas no passado receberam feedback negativo em relação à sua pertinência. Ou seja, as suas preferências ou gostos declarados directamente ou indirectamente (por uma acção relativa a um tipo de artefacto) podem gerar sugestões, desde que estas pertençam a um segmento que seja relativo para esse utilizador e que esse tipo de sugestão não seja considerado uma sugestão que não desperte interesse no respectivo utilizador.

Neste sentido o sistema começa por consultar a segmentação do utilizador, chamando o método `getUserSegmentation` que usa como recurso a rede de Bayes definida para o efeito e para cada segmento verifica se este é relevante ou se este deve ser ignorado. Caso suceda essa relevância é então analisado o factor de certeza recorrendo ao método `getIncerteza` sobre a pertinência de apresentar cada tipo de sugestões dentro do segmento corrente. Se a visualização desse tipo de sugestões se mostrar adequada, então tentam-se encontrar sugestões desse tipo e registam-se para poderem ser posteriormente referenciadas e usadas como estatística, sendo então devolvidas pelo dashboard.

As sugestões geradas e guardadas pelo `geraSugestoesTipo` que é invocado, como se pode ver no código da Secção A6 do Anexo A, tem vários atributos, entre eles o visto que informa se é uma sugestão activa ou não, o que permite ao dashboard retornar todas as sugestões activas daquele utilizador. Outros atributos, como o estado, permitem mais tarde uma análise de pertinência e podem levar inclusive a uma reformulação do processo de inferência.

[4] O factor de incomodo é o uso do factor de certeza acerca da pertinência sobre a apresentação de sugestões sem que estas incomodem o utilizador.

5 Desenvolvimento componente cliente

A componente cliente é uma aplicação que estará instalada nos dispositivos de cada cliente e que comunicará, via Internet, com a componente servidora alojada num servidor que disponibilizará *webservices* como interface para esta. Esta aplicação possui apenas modo on-line e necessita de registo (gratuito) e login para ser acedida.

5.1 Ambiente gráfico

O ambiente gráfico de interacção com o utilizador alvo de qualquer aplicação da actualidade é a chave para o sucesso ou insucesso [Campos, 2011]. Os utilizadores têm por hábito avaliar as qualidades de uma aplicação pelo seu aspecto e facilidade de uso. Com isto em mente é natural que na maioria das aplicações a interface com o utilizador final seja a componente que demora mais a desenvolver, devido à dificuldade de dar todas as opções possíveis aos utilizadores e mesmo assim manter um ambiente apelativo, organizado e "limpo".

Para os utilizadores que pretendem usar a aplicação de uma forma continuada, esta tem de ser operacionalmente simples, quer isto dizer que as opções (operações) mais comuns têm de se encontrar de fácil e rápido acesso.

Os utilizadores que vão entrar em contacto com a aplicação são na sua esmagadora maioria utilizadores de outras aplicações, as quais muito provavelmente seguem um fluxo de operações relativamente similar. Isto permite aos utilizadores construírem um modelo mental que os ajuda a usar as várias aplicações, sem fazer um grande esforço ao transitar entre as mesmas.

Na medida em que esta aplicação se enquadra no domínio Microsoft, a qual apostou fortemente no ambiente e aplicações metro, as quais tem um modelo muito específico e semelhante, o ambiente desta aplicação usou as normas metro. Isto enquadra a aplicação no ambiente e facilita o seu uso e interpretação, como se pode visualizar na Figura 12.

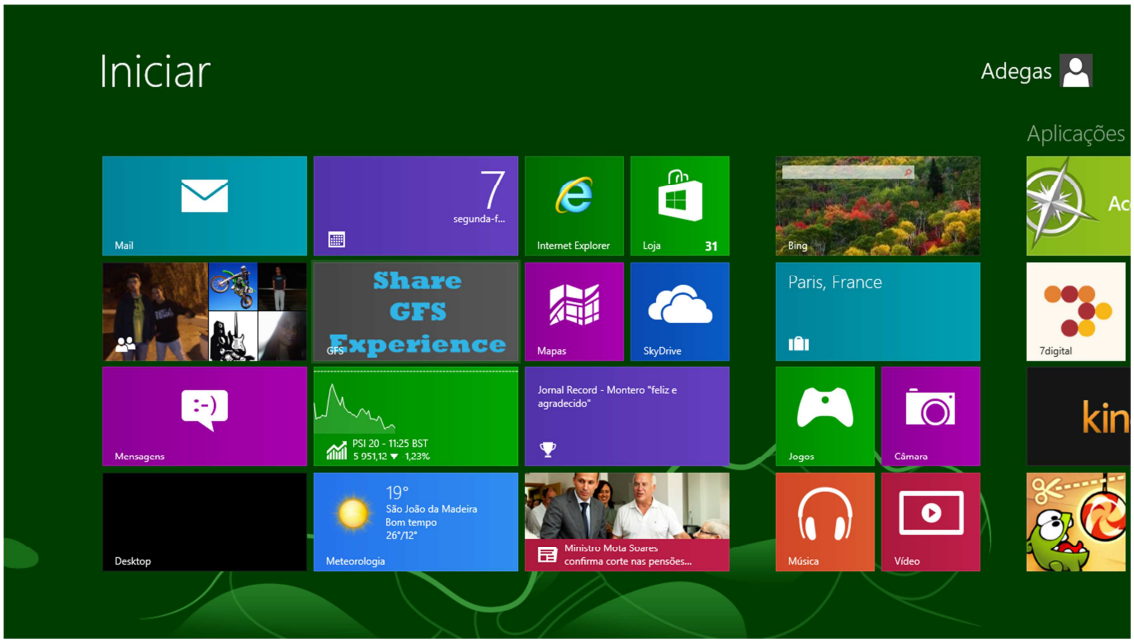

Figura 12 – Ambiente Metro

A construção da interface em linguagem metro proporciona uma outra grande vantagem na medida em que a simplicidade, organização topológica e design são indicados para um sistema de recomendação [Figueiredo, 2011].

5.2 Pesquisas

O sistema operativo com interface metro disponibiliza um serviço de pesquisa comum a todas as aplicações que implementem uma interface para o mesmo. Esta funcionalidade não é de implementação obrigatória, mas a sua implementação torna a aplicação mais user-friendly, na medida em que o utilizador pode estar fora da aplicação e com a caixa de pesquisa padrão do sistema operativo, pesquisar dentro da aplicação, abrindo-a para lhe apresentar o resultado. Com este princípio, uma simples caixa de texto permite pesquisar aplicações, configurações, ficheiros ou dados dentro de uma aplicação, seja esta proprietária da Microsoft ou não.

O contrato de SEARCH da Microsoft implica a implementação de uma interface que recebe do sistema operativo o pedido de pesquisa, realiza a pesquisa propriamente dita (dentro da aplicação) e apresenta o resultado dentro da própria aplicação, fazendo com que esta se inicie automaticamente como se pode ver na Figura 13.

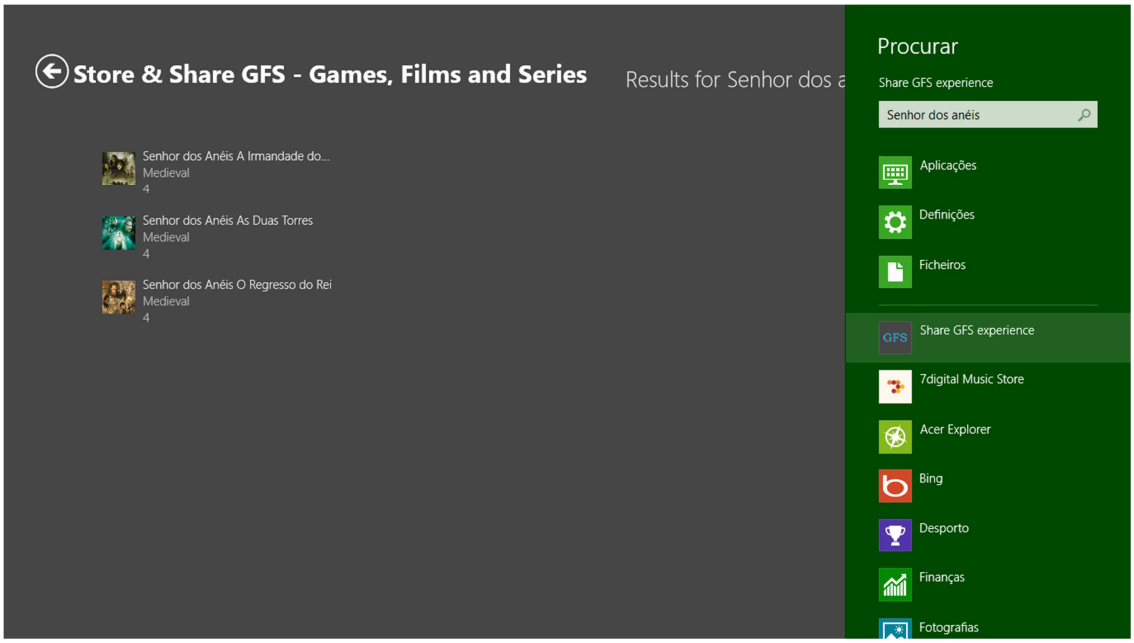

Figura 13 – Contrato de pesquisa da Microsoft

Na implementação do contrato de pesquisa seguem-se os seguintes passos: acede-se ao package manifest e no separador declarações adiciona-se a declaração search contract. De seguida cria-se uma página que dê resposta aos pedidos, como se pode ver na estrutura de código que se encontra na Secção A7 do Anexo A. É essencial que esta página tenha os métodos necessários para servir de interface de pesquisa, pelo que é aconselhável que esta herde esses métodos de uma classe que os implemente como a classe `LayoutAwarePage`. Uma possível implementação simplista desta classe pode ser consultada no código apresentado na Secção A7 do Anexo A.

5.3 Partilhas

Os sistemas metro disponibilizam a possibilidade de facilitarem a funcionalidade de partilha entre as suas aplicações. Com esta funcionalidade bem implementada, uma aplicação metro pode exportar e importar informação utilizando o standard de dados para partilha. Esta funcionalidade não é de implementação obrigatória, mas a sua implementação torna a aplicação mais user-friendly, na medida em que o utilizador pode com facilidade partilhar uma informação, sem precisar de recorrer ao antigo copy/paste.

O contrato Share da Microsoft implica a implementação de uma interface que outras aplicações possam utilizar para enviar informação mesmo que a aplicação esteja fechada, à semelhança do contrato de pesquisa. Como se pode visualizar na Figura 14, a partir de uma página que implemente a interface de partilha, pode-se escolher uma aplicação que receba partilhas e a informação da página corrente é embutida num pacote e enviada a essa aplicação. Na Figura 15 é visível a recepção de um pacote partilhado por outra aplicação.

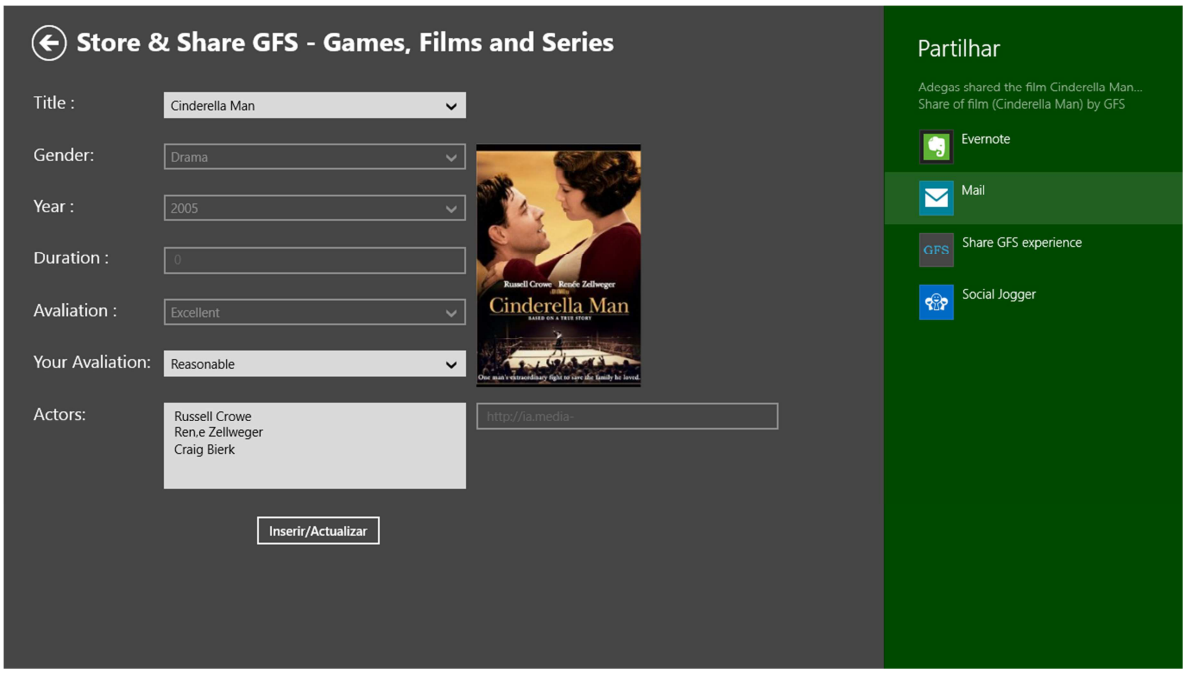

Figura 14 – Contrato de partilha da Microsoft (Partilhar informação)

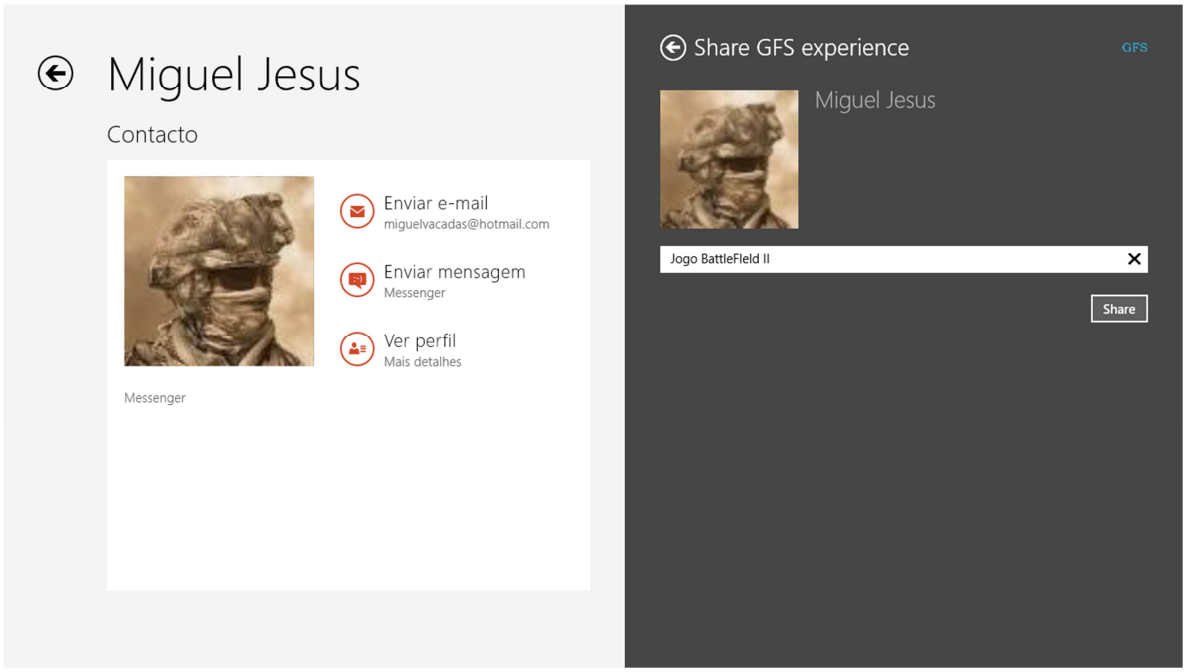

Figura 15 – Contrato de partilha da Microsoft (Receber informação)

Na implementação do contrato de partilha seguem-se os seguintes passos: acede-se ao package manifest e no separador declarações adiciona-se a declaração Share Contract. De seguida desenvolve-se uma página de recepção deste tipo de informação que é herdada da classe `LayoutAwarePage` e que se encarrega de ficar à espera que partilhem informação, recebendo-a, tratando-a e guardando-a na base de dados. A titulo de curiosidade uma possível implementação simplista desta classe pode ser consultada no código apresentado na Secção A8 do Anexo A.

Nos formulários ou páginas em que se pretender disponibilizar a possibilidade de partilhar a informação neles contida, estes herdam da classe SharePage e rescrevem o método GetShareContent, o qual devolve os valores para o título, e descrição do artefacto a partilhar. Uma possível implementação da criação deste pacote de dados para partilha entre aplicações pode ser consultado no código apresentado na Secção A9 do Anexo A.

5.4 Ambiente sugestivo

O subsistema que apresenta as dicas faz com que elas tenham um carácter meramente sugestivo, fazendo com que a sua apresentação embora seja visível, não seja intrusiva para que não incomode o utilizador, caso este não esteja interessado.

Estas sugestões podem ser visualizadas de duas formas, no dashboard onde são apresentados *tiles*[5] para sugestões de jogos, filmes ou séries. Ou seja, podem ser apresentados de 0 a 3 *tiles* e no caso de ser pressionado um tile, por exemplo o tile de jogos, será apresentada uma lista com apenas as sugestões de jogos, as quais podem ser maximizadas para a visualização de toda a informação e envio do respectivo feedback. A outra maneira de visualizar sugestões é pedindo-as especificamente, indicando o seu tipo e respectivos atributos. Por exemplo, tipo Jogo e atributos FPS da Activision[6] para PC.

As sugestões são geradas independentemente para cada utilizador, tendo em conta as suas preferências e o seu feedback anterior, o que faz com que estas se aproximem cada vez mais das necessidades do mesmo, reduzindo assim as sugestões desinteressantes para aquele utilizador específico.

As sugestões são então apresentadas, não só tendo em conta a segmentação do utilizador, como o seu feedback (Like ou Don't Like) sobre sugestões anteriormente apresentadas e o total das mesmas. Estes dados, no servidor, entram então no processo de cálculo que define se é oportuna a apresentação das sugestões.

[5] Os tiles são botões em forma de rectângulos ou quadrados que são efectivamente atalhos para lançar aplicativos, e muitas vezes são exibidos dados adicionais sobre o aplicativo que eles representam.
[6] A Activision é uma conhecida empresa americana especializada no desenvolvimento de jogos para diversas plataformas

6 Testes e Experiências

Após ter sido exposto o desenvolvimento da aplicação, seguem-se algumas experiências realizadas no sistema para verificar se a sua resposta, ou seja, as suas sugestões se enquadravam com o utilizador que as recebia e se o sistema levaria em conta o incómodo do respectivo utilizador.

Para a melhor compreensão dos *printscreens* apresentados nas experiências, segue-se uma descrição das componentes que compõem o ecrã principal do sistema. Quando este não apresenta sugestões, apresenta uma matriz 2X3 onde na linha de cima estão representados os jogos, filmes e séries do utilizador que procedeu ao *login*, onde cada grupo é representado por um quadrado. Na linha de baixo estão representados os utilizadores de jogos, filmes e séries da comunidade, também estes representados por um quadrado cada. Quando o sistema apresenta sugestões de jogos, filmes ou séries estas são também representadas por um quadrado cada, que surge à frente da matriz inicial.

6.1 Teste – Sugestões de filmes

Cenário:

Um novo utilizador, após se registar sem demonstrar preferências, realiza a sua primeira acção no sistema inserindo o seu primeiro filme, neste caso o filme "*The Expendables*" de acção, lançado em 2010. De seguida à inserção o utilizador volta para o dashboard.

Resultado esperado:

É esperado que o sistema apresente apenas sugestões de filmes de acção, ordenando-as caso estas sejam mais idênticas, através dos actores ou mesmo do ano de lançamento.

Resultado obtido:

Como se pode visualizar na Figura 16 as únicas sugestões apresentadas foram sugestões de filmes de acção, como é o caso do filme *"Death Race"*.Premindo este, aparece uma lista com todos os filmes de acção sugeridos, onde entre outros se encontra o filme *"Death Race"*. Este agrupamento de sugestões é indicado para não surpreender nem incomodar o utilizador.

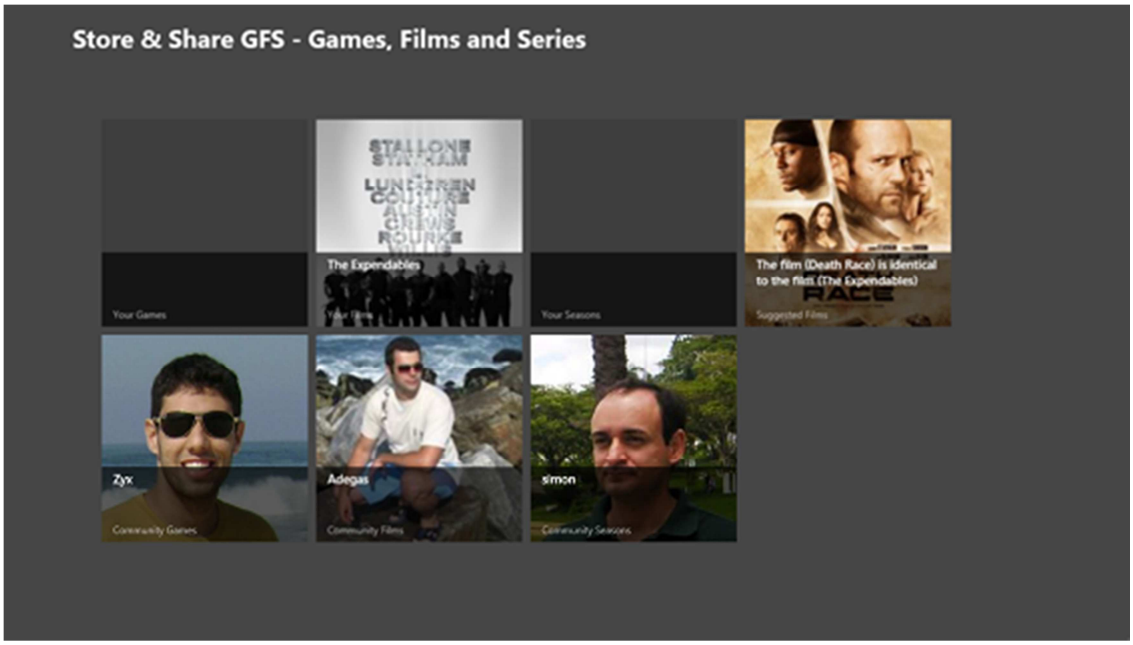

Figura 16 – *Resultado do primeiro teste*

6.2 Teste – Sugestões de filmes e séries

Cenário:

Um utilizador entra na aplicação onde registou um filme de comédia, duas séries de acção, uma série de comédia e nenhum jogo. Para além disso o utilizador ainda não realizou nenhuma acção de pesquisa ou consulta.

Resultado esperado:

Neste caso é esperado que o sistema apresente sugestões de filmes de comédia e de séries de comédia ou de acção, visto que foram os únicos tipos avaliados anteriormente pelo utilizador.

Resultado obtido:

Na Figura 17 é visível a sugestão do filme "O despertar da mente" cujo género é comédia e também a sugestão da série "FlashPoint" cujo género é acção. Qualquer um destes artefactos foi seleccionado para ser a capa do grupo de sugestões do respectivo tipo apresentadas.

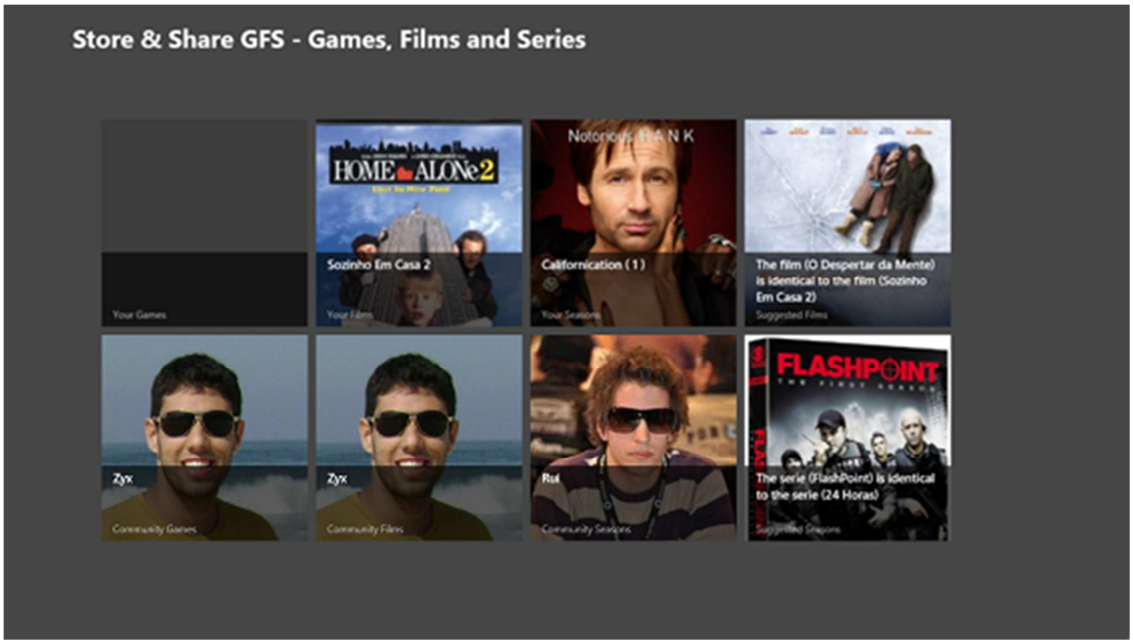

Figura 17 – *Resultado do segundo teste*

6.3 Teste – Sugestões de jogos, filmes e séries

Cenário:

Um utilizador previamente registado conta com dois registos de filmes policiais, um de uma séria de acção e quatro jogos do tipo FPS. O utilizador em questão tinha feito várias pesquisas sobre jogos e acaba de realizar uma partilha sobre um jogo também ele FPS. Seguidamente o utilizador volta ao Dashboard.

Resultado esperado:

Neste cenário é esperado que o sistema apresente sugestões dos três tipos de média desde que encontre sugestões apropriadas, visto que o utilizador avaliou artefactos dos três tipos de média e que realizou acções de pesquisa e partilha.

Resultado obtido:

Na Figura 18 são visíveis as sugestões que o sistema apresentou de jogos, filmes e séries, todas elas relacionadas com artefactos anteriormente avaliados pelo utilizador.

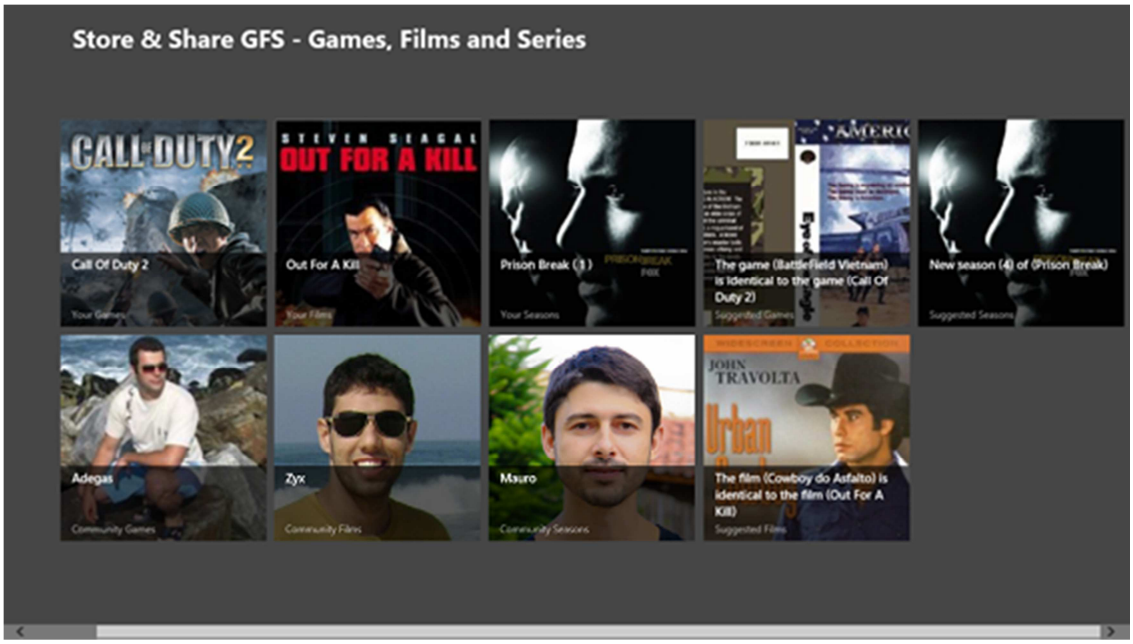

Figura 18 – *Resultado do terceiro teste*

6.4 Teste – Ausência de sugestões

Cenário:

Um utilizador entra no sistema possuindo apenas cinco filmes de acção, contudo as três únicas sugestões apresentadas (tanto de novos filmes como de novas versões do mesmo), foram apresentadas recentemente e todas elas receberam feedback negativo.

Resultado esperado:

É esperado que o sistema não apresente qualquer sugestão já que a segmentação do utilizador é predominantemente pertencente ao segmento dos filmes e as sugestões para esse segmento receberam feedback negativo recentemente.

Resultado obtido:

Como se pode visualizar na Figura 19 não foram apresentadas quaisquer sugestões embora o utilizador tenha avaliado filmes anteriormente.

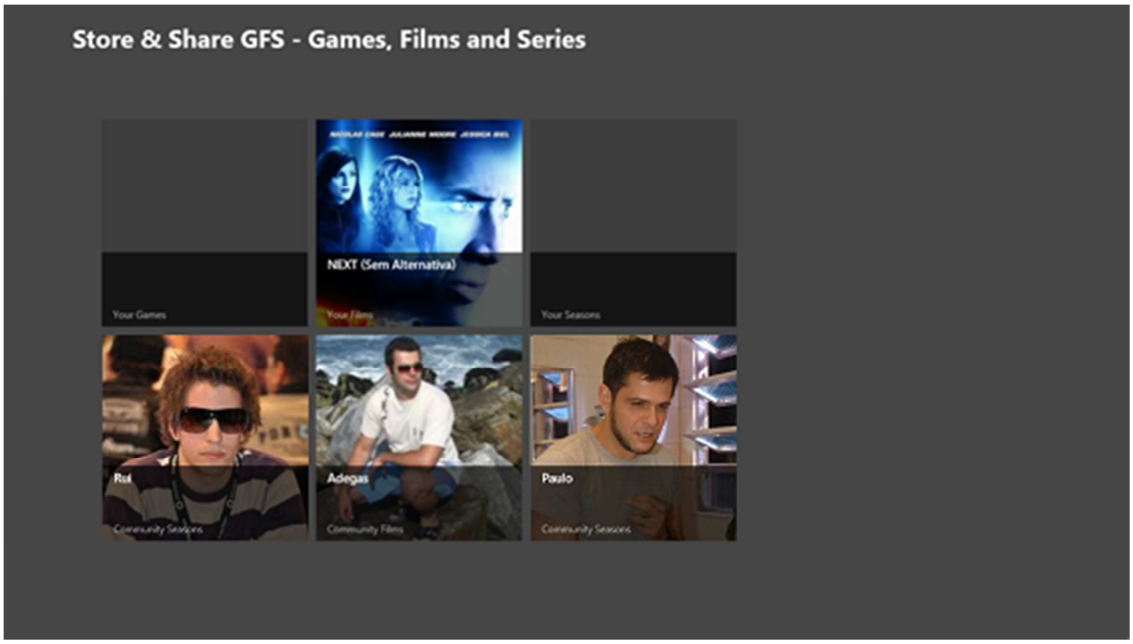

Figura 19 – *Resultado do quarto teste*

6.5 Teste – Sugestões de filmes variados

Cenário:

Um utilizador entra no sistema possuindo diversos filmes de acção e comédia e não tendo dado feedbacks negativos às sugestões anteriormente apresentadas.

Resultado esperado:

É esperado que o sistema apresente sugestões de filmes de comédia e acção visto que são os tipos de artefactos que este possui e não se regista feedback negativo.

Resultado obtido:

Como se pode visualizar na Figura 20, o utilizador recebeu sugestões tanto de filmes de comédia, como de filmes de acção.

De notar que na Figura 20 é apresentada a lista de sugestões apresentadas ao utilizador, ao invés do dashboard inicial com apenas um quadrado para as sugestões de filmes. Neste caso o filme apresentado nesse quadrado tanto pode ser um filme de comédia como de acção, visto que o artefacto que serve de capa à lista de sugestões é escolhido aleatoriamente a partir da respectiva lista.

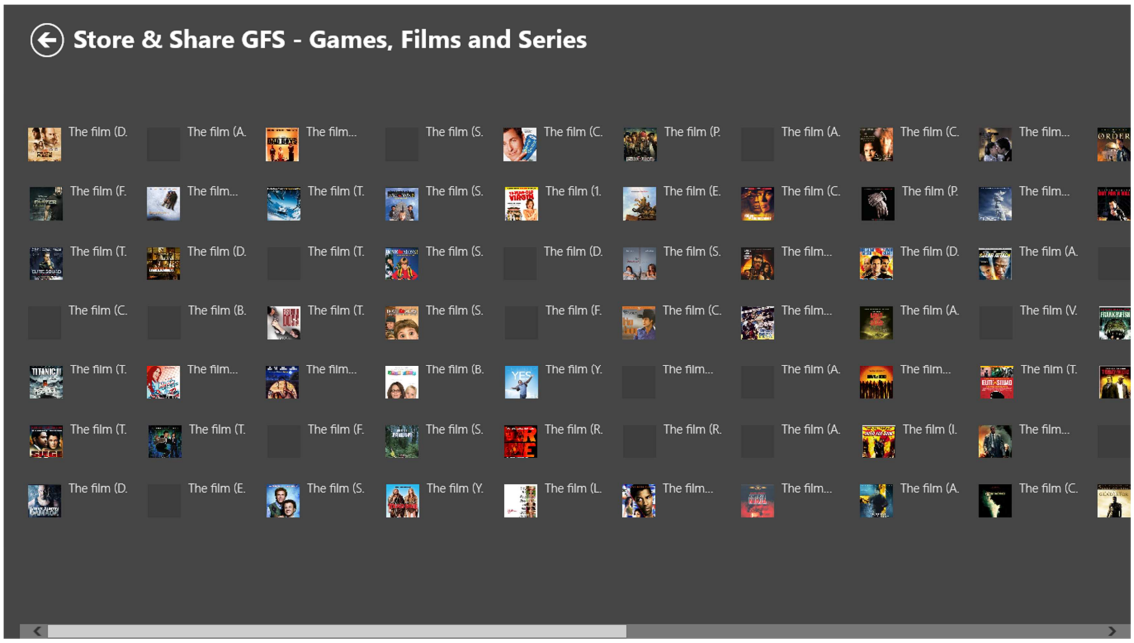

Figura 20 – *Resultado do quinto teste*

A título de exemplo, é visível na Figura 21 o formulário de uma das sugestões apresentadas na lista apresentada na Figura 20. É aqui demonstrada a possibilidade do utilizador fornecer o seu feedback, através dos botões *Like* e *Don't Like*.

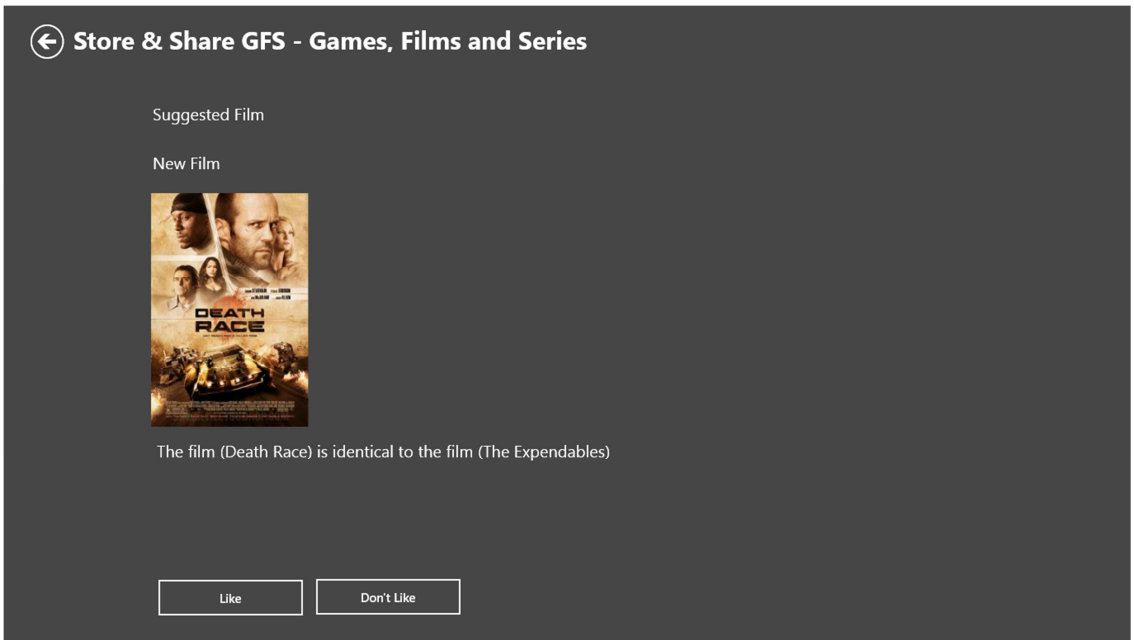

Figura 21 – *Formulário de uma sugestão*

6.6 Sugestões a pedido

Como se pode visualizar na Figura 22, o sistema possui uma barra de funções localizada no fundo do ecrã que pode ser chamada com o botão direito do rato, como é habitual nas aplicações metro. Nesta barra encontra-se a opção de pedir sugestões.

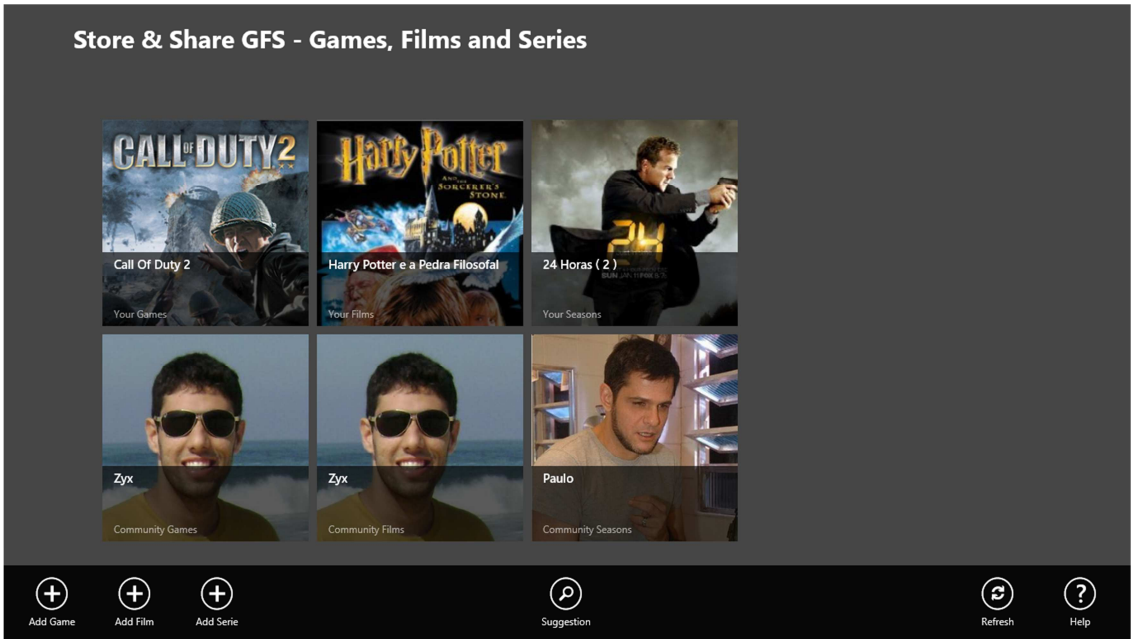

Figura 22 – *Barra de comandos*

Depois de pedir uma sugestão o utilizador deve indicar que tipo de artefacto e os seus respectivos atributos como, por exemplo, o seu género. Com estes dados o sistema realiza uma pesquisa e caso encontre sugestões, apresenta-as em forma de listagem como se pode ver na Figura 23. Neste caso foram pedidos filmes do género guerra, com uma duração normal e uma avaliação alta.

Figura 23 – *Lista de sugestões a pedido*

7 Conclusão

Após um estudo sobre as respostas obtidas do sistema de recomendação, passa-se então a uma fase de análise conclusiva. Sobre o sistema proposto e desenvolvido verifica-se que o mesmo possui as características que se consideraram relevantes para o seu sucesso no mercado. Para concretizar este ponto é necessário alojá-lo num servidor e realizar acções de marketing que proporcionem uma afluência ao seu uso, com vista à criação e futura gestão da dita comunidade.

7.1 Objectivos alcançados

Nesta Secção procura-se argumentar que todos os objectivos subjacentes à realização deste trabalho de mestrado foram realizados com sucesso.

- Com a conclusão deste trabalho é conveniente constatar que está criada uma infra-estrutura multiplataforma que permite aos diversos consumidores de media se agruparem e partilharem as suas experiências.
- A infra-estrutura está munida de um sistema de sugestões que se adaptam às preferências de cada utilizador e proporcionam uma ajuda na tomada de decisão sobre artefactos a ver/rever automaticamente pelo sistema baseado nos feedbacks do utilizador e do resto da comunidade.
- O sistema permite ao utilizador pedir um determinado tipo de sugestões, cujos critérios são definidos na hora, ajudando o utilizador na tomada de decisão sobre que artefacto escolher/consumir.
- As acções dos utilizadores são registadas o que permite ao sistema recalcular as segmentações dos utilizadores e a pertinência de lhes mostrar determinado tipo de sugestões. Estas acções são registadas gerando dados históricos que permitem ao sistema ajustar os pesos, tanto da rede de Bayes como dos factores de certeza. Estas características tornam o sistema mais inteligente na medida em que este aprende

com as interacções dos utilizadores, o que faz com que ele melhore a sua performance sem necessidade de se proceder a uma alteração do código.

7.2 Principais dificuldades

As dificuldades sentidas no desenvolvimento deste projecto podem ser subdivididas em categorias, como sendo dificuldades técnicas, falta de documentação e discrepância entre conhecimentos prévios e as novas tecnologias.

No âmbito das dificuldades por falta de documentação, realça-se que esta dissertação começou antes do lançamento do Windows 8 pelo que a informação para desenvolvedores de software ainda não era abundante, nem tão pouco clara. Destaca-se a dificuldade de desenvolver a parte gráfica segundo o layout aconselhado pela Microsoft, a implementação do contracto de pesquisa e a implementação do contracto de partilha.

Ainda na secção de falta de documentação realça-se a dificuldade de encontrar dados históricos que suportem a atribuição de valores iniciais para as percentagens da rede de Bayes acerca da segmentação dos utilizadores, bem como a atribuição de incerteza sobre a apresentação de sugestões ao utilizador.

Como dificuldades técnicas apresenta-se uma questão relativa ao ambiente de desenvolvimento, sendo que o primeiro Microsoft Visual Studio 2012 lançado, focava apenas o desenvolvimento de aplicações metro deixando em descuido, por exemplo, a criação de *Webservices* o que implicou o uso de duas versões do ambiente de desenvolvimento, uma para a aplicação cliente (metro) e outra para a componente servidora (*Webservices*). Mais tarde, aquando do lançamento da versão completa do Microsoft Visual Studio 2012, a componente servidora foi migrada para o novo ambiente de desenvolvimento, abdicando assim da necessidade de recorrer à versão antiga do mesmo.

A maior dificuldade encontrada encontra-se nas secção técnica e prende-se com a dificuldade de comunicar entre os *Webservices* e um programa Prolog (Lpa-Flex). Os programas desenvolvidos em C# (caso dos *Webservices*) podem fazer uso de uma biblioteca chamada LPA.dll que permite carregar o Prolog e executar comandos de Prolog recebendo as respectivas respostas. No entanto, ocorria uma falha no servidor web sempre que o Webservice tentava executar um comando Prolog. A dll precisava de ser executada na framework 2 , mas os Webservices utilizam outras bibliotecas que necessitam da framework 4 ou superior. Com este impedimento em mente rumou-se para o uso de uma livraria WCF (Windows Communication Foundation) que fizesse o acesso ao Prolog e devolvesse a resposta ao *Webservice*, mas surgiu um problema, o uso da biblioteca LPA obriga a que a aplicação seja de 32bits e a livraria WCF apenas funciona a 64 bits, o que colocou também esta tecnologia de parte. Por fim optou-se por uma situação cliente/servidor onde o servidor é uma aplicação standalone em C# que comunica com o Prolog recebendo os pedidos (comandos Prolog)

através de um pipe e respondendo pelo mesmo. Na outra extremidade do pipe o *webservice* faz o pedido e fica à escuta de forma assíncrona.

Por fim, na secção de discrepância entre conhecimentos prévios e as novas tecnologias situa-se o desenvolvimento dos *webservices*. Actualmente, os *webservices* utilizam palavras reservadas como sendo o `async` antes do tipo de retorno do método que invoca o *webservice* e a palavra reservada `await` imediatamente antes da invocação do *webservice* para indicar que a comunicação com o *webservice* será assíncrona. Para além das palavras reservadas, denota-se também uma mudança na invocação dos métodos. Estes passaram a usar `Async` à frente do nome do método, a resposta usa `Response` à frente do nome do método e o Resultado usa `Result`. Por exemplo, o método `getUserSegmentation` seria chamado como `getUserSegmentationAsync`, a sua resposta seria do tipo `getUserSegmentationResponse` e o resultado da resposta do tipo `getUserSegmentationResult`.

7.3 Limitações

Pensa-se que teoricamente a plataforma se encontra preparada para atingir os objectivos propostos inicialmente, contudo, na prática muitas vezes o difícil é arranjar estratégias para criar a própria comunidade, ou seja, dar a conhecer e despertar o interesse da aplicação ao público-alvo.

A falta de API's relevantes por parte do motor IMDB, faz com que a componente servidora do sistema se encontre sujeita à estrutura HTML das páginas do dito motor. Ou seja, se a estrutura de páginas do IMDB for alterada é necessária uma intervenção na componente servidora, para que esta se adapte a pesquisar nas páginas com uma nova disposição. Esta limitação pode ser facilmente colmatada se do lado IMDB forem criadas API's mais completas que as existentes que permitam a pesquisa e recolha facilitada de informação.

7.4 Trabalho futuro

Futuramente pretende-se que o sistema permita a publicação directa no *Facebook* de uma avaliação, possibilitando o acrescento de um texto à escolha do utilizador.

Pretende-se também expandir o âmbito do sistema para incluir músicas, com o mesmo intuito dos restantes artefactos, sendo estas identificadas pelo nome, artista, álbum e duração. O utilizador terá a opção de, para além de classificar a música, poder indicar se a ouviu em algum concerto e qual.

Com o decorrer do tempo de vida do sistema pretende-se fazer uma análise do seu histórico, para verificar se as actualizações à rede de Bayes e aos factores de certeza estão a ser uma mais valia, ou se será necessário alguma intervenção à componente servidora (*webservices*).

Está também incluído na lista de trabalhos e já está contemplado na base de dados a partilha de mensagens privadas entre utilizadores do sistema.

Encontra-se também em fase de análise uma possível interligação através de *webservices* entre o sistema GFS e um sistema de classificação de links (locais para download/compra) que permitiria sugerir aos utilizadores qual ou quais os melhores locais para aquisição do artefacto. Este sistema encontra-se ainda em fase de desenvolvimento, também ele inserido no âmbito de tese/dissertação do MEI (Mestrado em Engenharia Informática).

No nível administrativo encontra-se planeado um método que à semelhança do método de actualizar as imagens dos artefactos, consulte o sistema IMDB para descobrir e inserir artefactos que o sistema ainda não possua. Assim, pode-se melhorar a qualidade e quantidade de sugestões oferecidas aos utilizadores. Esta funcionalidade ficou pendente devido à falta, no momento, de uma API por parte do IMDB que possibilite o acesso aos dados de uma forma mais correcta e com uma menor possibilidade de erro. Neste momento, a inserção dos artefactos no sistema está dependente dos seus utilizadores, contudo o sistema só necessita que o utilizador introduza o título do artefacto, caso este não exista já no sistema, sendo os restantes campos relevantes preenchidos automaticamente. Esta funcionalidade socorre-se de uma API que não pertence ao IMDB, mas faz uso da sua base de dados permitindo pesquisar os atributos de um artefacto bastando fornecer-lhe o título. Pelo facto de ter de fornecer o título, esta API não é indicada para o sistema procurar novos artefactos sem a interacção do utilizador.

Referências

[Azevedo, 2013] AZEEDO, Cátia - Introdução ao tema das Redes Bayesianas, a 25/01/2013, FCUP, Departamento de matemática, Engenharia Matemática

[Bayes e Price, 1763] BAYES, Thomas; PRICE, Richard - An Essay Towards Solving a Problem in the Doctrine of Chances. : By the late Rev. Mr. Bayes, communicated by Mr. Price, in a letter to John Canton, MA. and F.R.S., 1763, Philosophical Transactions of the Royal Society of London, 53, 370-418.

[Bellotti *et al*., 2010] BELLOTI, Francesco; BERTA, Riccardo; GLORIA, Alessandro; POGGI, Andrea - Java APIs for optimized 2D graphics, 2010

[Castillo e Melin, 2008] CASTILHO, Oscar; MELIN, Patricia - Type-2 Fuzzy Logic: Theory and Applications, 2008 ISBN: 978-3-540-76283-6 (Print) 978-3-540-76284-3 (Online)

[Camarinha, 2009] CAMARINHA, Maria Margarida Oliveira - Tese de Mestrado em Análise de Dados e Sistemas de Apoio à Decisão, Dezembro de 2006, na FEP – Faculdade de Economia do Porto, orientada por Professor Doutor João Manuel Portela da Gama

[Campos, 2011] CAMPOS, Thais Maia - Importância da Interface Gráfica, 2011, último acesso em 07/10/2013 ao link http://novidadesemti.wordpress.com/2011/11/29/importancia-da-interface-grafica/

[Chagas, 2007] CHAGAS - Ricardo Pedreti - Aplicação de redes Bayesianas na previsão do crescimento de fluxos de caixa, 2007, Fundação Getúlio Vargas, Escola de economia de São Paulo

[Cortez e Neves, 2000] CORTEZ, Paulo; NEVES, José - Redes Neuronais Artificiais, 2000, Universidade de Ensino, Departamento de Informática, Escola de Engenharia Universidade do Minho, Braga, Portugal.

[Coudert *et al*., 2010] COURDER, Thierry; GRABOT, Bernard; ARCHMÈDE, Bernard - Production/maintenance cooperative scheduling using multi-agents and fuzzy logic, 2010

[Cross *et al*., 2012] CROSS, Jason; RALPH, Nate; ALBRO, Edward - PC World. May2012, Maio de 2012, Vol. 30 Issue 5, p61-67. 7p. 7 Color Photographs, 3 Charts

[Eric, 2012] ERIC, Bruno -The JVM's Popularity, 2012, ISSN: 87506874

[Faria, 2012a] Apontamentos teóricos de sistemas baseados em conhecimento, Luiz Faria em 2012, Departamento de Engenharia Informática do Instituto Superior de Engenharia do Porto

[Faria, 2012b] FARIA, Luís - Apontamentos teóricos de sistemas baseados em conhecimento, Aplicação da Lógica Difusa aos Sistemas Periciais, 2012, Departamento de Engenharia Informática do Instituto Superior de Engenharia do Porto

[Faria, 2012c] FARIA, Luís - Apontamentos teóricos de sistemas baseados em conhecimento, Apresentação da Lógica difusa, 2012, Departamento de Engenharia Informática do Instituto Superior

de Engenharia do Porto

[Figueiredo, 2011] FIGUEIREDO, Almeida Baptista; NEVES, Maria - Apontamentos de Sistemas de apoio à decisão, 2011, Departamento de Engenharia Informática do Instituto Superior de Engenharia do Porto

[Cazella *et al*., 2005] GAZELLA, Sílvio César; NUNES, Maria Augusta; REATEGUI, Eliseo Berni - A Ciência da Opinião: Estado da arte em Sistemas de Recomendação, 2005

[Gulyás, 2006] GULYÁS, Csaba; THESIS, Master's - Creation of a Bayesian network-based meta spam filter, using the analysis of different spam filters, 2006, Budapest

[Harmon e King, 1995] HARMON, Paul; KING, David - Artificial Intelligence in Business, 1995

[Lia e Sheng, 2010] LIA, Jing; CHINA, Nanjing; SHENG Zhaohan - Papper: A multi-agent model for the reasoning of uncertainty information in supply chains, 2010

[Malheiro *et al*., 2004] Sistemas de classificação musical com redes neuronais, autores Ricardo Malheiro, Rui Pedro Paiva, António José Mendes, Teresa Mendes e Amílcar Cardoso em 2004, Universidade Católica Portuguesa, ISSN: 0872-556X.

[Matsuura, 2003] MATSUURA, Jackson Paul - Discretização para aprendizagem Bayesiana: Aplicação no auxilio à validação de dados em protecção ao voo, 2003, Campo Montenegro, São José dos Campos, SP- Brasil

[McCarthy, 1984] MCCARTHY, John - Some Expert System Need Common Sense, 1984, Computer Science Department, Stanford University, CA 94305

[Microsoft Research, 2010] Microsoft Research, http://research.microsoft.com/en-us/um/redmond/groups/adapt/msbnx/ , Microsoft Bayesian Network Editor, actualizado em 2010, última visita a 18/07/2013.

[MSDN, 2013] Develop Windows Store apps using Visual Studio em 2013 http://msdn.microsoft.com/en-us/library/windows/apps/br211384.aspx Microsoft MSDN, último acesso 21/06/2013

[Nikola e Ramazan, 2013] NIKOLA, Gradojevic; RAMAZAN, Gençay, - Fuzzy logic, trading uncertainty and technical trading: Journal of Banking and Finance, 2013, Vol.37(2), pp.578-586

[Nunes, 2012] NUNES, Maria Augusta Silveira Netto; OLIVEIRA, Otávio Cordeiro Siqueira; GAZELLA, Silvio César - Personal_Movie – Um modelo de Sistema de Recomendação de filmes geolocalizados em eventos, 2012 publicado na Revista de Sistemas de Informação da FSMA n. 10 (2012) pp. 44-52

[Oliveira, 2012] OLIVEIRA, Beatriz - Microsoft DEVCAMPS, Windows 8 Metro APP, 28/05/2012, Porto

[Pinto, 2005] PINTO, Carlos Manuel Silva - Algoritmos Incrementais para Aprendizagem Bayesiana, 2005, Departamento de Ciência de Computadores, Faculdade de Economia da Universidade do Porto

[Ramos e Silva, 2011] RAMOS, Carlos; SILVA, António - Apontamentos teóricos de sistemas baseados em agentes, 2011

[Robbins, 2012] ROBBINS, Drew - Microsoft DEVCAMPS, Windows 8 Metro APP, 28/05/2012, Porto

[Sabo, 2006] SABO, Letícia Andrade - Análise da incerteza na representação de classes temáticas, 2006, Universidade Estadual Paulista,

Faculdade de Ciências e Tecnologia, Campus de Presidente Prudente

[Shortliffe, 1976] SHORTLIFFE, Edward Hance - Computer-Based Medical Consultations: MYCIN; Elsevier - North Holland, 1976

[Souza, 2012] SOUZA, Renata Ghisloti Duarte - Aplicando Sistemas de Recomendação em Situações Práticas, 2012 , Software Engineer, Linux Technology Center – IBM.

[Souza, 2013] SOUZA, Leandro - BAGUETE: http://www.baguete.com.br/noticias/28/06/2013/netflix-lanca-sistema-amigo-para-recomendar-filmes, última visita a 22/08/2013.

[Wiki, 2012] Metro (Design Language) http://en.wikipedia.org/wiki/Metro_(design_language), ultimo acesso 12/05/2013.

[Wooldridge, 1995] WOOLDRIDGE, Michael; JENNINGS, Nicholas - Intelligent Agents-Theories, Architectures, and Languages, 1995, Volume 890

[Zhaa *et al*.,, 2001] ZHAA, X.F; DUA, H.J; QIU J.H - Knowledge-based approach and system for assembly-oriented design, Part II: the system implementation, 2001, Institute of Fluid Science, Tohoku University Sendai 980-8577, Japan

Anexo A Códigos

A1 – Segmentação no login

O extracto de código que se segue pertencente ao método getUserSegmentation ilustra a forma como a componente servidora pesquisa, calcula e retorna a segmentação de cada utilizador relativamente a cada tipo de artefacto tratado.

```
private double[] getUserSegmentation(string user_id)
{
     DBConnect bd = new DBConnect();
     double[] resultado = new double[3];

     //carregar rede de bayes dos segmentos de utilizadores
     MSBN3Lib.Node n;

     n = modelAuto.ModelNodes["Percentagem"];
     ArrayList tags = new ArrayList();
     tags.Add("total");
     int total_g = 0, total_f = 0, total_s = 0;
     ArrayList result = bd.Select("select count(*) total from util_jogo where
                                   id_util=" + user_id, tags);
     if (result.Count > 0)
          total_g = Convert.ToInt32(((ArrayList)result[0])[0]);

 result = bd.Select("select count(*) total from util_filme where id_util=" +
                     user_id, tags);
 if (result.Count > 0)
      total_f = Convert.ToInt32(((ArrayList)result[0])[0]);
 result = bd.Select("select count(*) total from util_temporada where id_util=" +
                     user_id, tags);
 if (result.Count > 0)
      total_s = Convert.ToInt32(((ArrayList)result[0])[0]);

n.get_Dist()[0, "RacioJogos"] = Convert.ToDouble(total_g) / (total_g + total_f +
total_s);
n.get_Dist()[0, "RacioFilmes"] = Convert.ToDouble(total_f) / (total_g + total_f
+ total_s);
n.get_Dist()[0, "RacioTemporadas"] = Convert.ToDouble(total_s) / (total_g +
total_f + total_s);

tags = new ArrayList();
tags.Add("valor");
int s = 0, i = 0;

n = modelAuto.ModelNodes["TarefaPartilhaG"];
result = bd.Select("select valor from propriedade where propriedade=51 and
                    user_id=" + user_id, tags);
if (result.Count > 0)
     s = Convert.ToInt32(((ArrayList)result[0])[0]);
result = bd.Select("select valor from propriedade where propriedade=52 and
                    user_id=" + user_id, tags);
if (result.Count > 0)
```

```
     i = Convert.ToInt32(((ArrayList)result[0])[0]);
n.get_Dist()[0, "Sucesso"] = Convert.ToDouble(s) / (s + i);
n.get_Dist()[0, "Insucesso"] = Convert.ToDouble(i) / (s + i);

n = modelAuto.ModelNodes["TarefaPartilhaF"];
s = 0; i = 0;
result = bd.Select("select valor from propriedade where propriedade=53 and
                     user_id=" + user_id, tags);
if (result.Count > 0)
     s = Convert.ToInt32(((ArrayList)result[0])[0]);
result = bd.Select("select valor from propriedade where propriedade=54 and
                    user_id=" + user_id, tags);
if (result.Count > 0)
     i = Convert.ToInt32(((ArrayList)result[0])[0]);
n.get_Dist()[0, "Sucesso"] = Convert.ToDouble(s) / (s + i);
n.get_Dist()[0, "Insucesso"] = Convert.ToDouble(i) / (s + i);

n = modelAuto.ModelNodes["TarefaPartilhaS"];
s = 0; i = 0;
result = bd.Select("select valor from propriedade where propriedade=55 and
                    user_id=" + user_id, tags);
if (result.Count > 0)
     s = Convert.ToInt32(((ArrayList)result[0])[0]);
result = bd.Select("select valor from propriedade where propriedade=56 and
                    user_id=" + user_id, tags);
if (result.Count > 0)
     i = Convert.ToInt32(((ArrayList)result[0])[0]);
n.get_Dist()[0, "Sucesso"] = Convert.ToDouble(s) / (s + i);
n.get_Dist()[0, "Insucesso"] = Convert.ToDouble(i) / (s + i);

n = modelAuto.ModelNodes["TarefaPesquisaG"];
s = 0; i = 0;
result = bd.Select("select valor from propriedade where propriedade=57 and
                    user_id=" + user_id, tags);
if (result.Count > 0)
     s = Convert.ToInt32(((ArrayList)result[0])[0]);
result = bd.Select("select valor from propriedade where propriedade=58 and
                    user_id=" + user_id, tags);
if (result.Count > 0)
     i = Convert.ToInt32(((ArrayList)result[0])[0]);
n.get_Dist()[0, "Sucesso"] = Convert.ToDouble(s) / (s + i);
n.get_Dist()[0, "Insucesso"] = Convert.ToDouble(i) / (s + i);

n = modelAuto.ModelNodes["TarefaPesquisaF"];
s = 0; i = 0;
result = bd.Select("select valor from propriedade where propriedade=59 and
                    user_id=" + user_id, tags);
if (result.Count > 0)
     s = Convert.ToInt32(((ArrayList)result[0])[0]);
result = bd.Select("select valor from propriedade where propriedade=60 and
                    user_id=" + user_id, tags);
if (result.Count > 0)
     i = Convert.ToInt32(((ArrayList)result[0])[0]);
n.get_Dist()[0, "Sucesso"] = Convert.ToDouble(s) / (s + i);
n.get_Dist()[0, "Insucesso"] = Convert.ToDouble(i) / (s + i);

n = modelAuto.ModelNodes["TarefaPesquisaS"];
s = 0; i = 0;
result = bd.Select("select valor from propriedade where propriedade=61 and
                    user_id=" + user_id, tags);
```

```
if (result.Count > 0)
     s = Convert.ToInt32(((ArrayList)result[0])[0]);
result = bd.Select("select valor from propriedade where propriedade=62 and
                    user_id=" + user_id, tags);
if (result.Count > 0)
     i = Convert.ToInt32(((ArrayList)result[0])[0]);
n.get_Dist()[0, "Sucesso"] = Convert.ToDouble(s) / (s + i);
n.get_Dist()[0, "Insucesso"] = Convert.ToDouble(i) / (s + i);


//ler crenças
resultado[0] = eng.Belief("Resultado", "JOGOS");
resultado[1] = eng.Belief("Resultado", "FILMES");
resultado[2] = eng.Belief("Resultado", "TEMPORADAS");

return resultado;

}
```

A2 – Ler e retornar a segmentação do utilizador

O extracto parcial de código que se segue, pertencente ao método publico getUserSegmentation, demonstra como a autenticação do utilizador é obrigatória e que o retorno é sempre feito no formato JSon.

```
[WebMethod]
public string getUserSegmentation(string user_id, string key)
{
    DBConnect bd = new DBConnect();

    //verificar autenticidade e caso seja um pedido autenticado do

    double[] resultado = getUserSegmentation(user_id);

    return "{\"belief_g\":\"" + resultado[0] + "\",\"belief_f\":\"" +
    resultado[1] + "\",\"belief_s\":\"" + resultado[2] + "\"}";
}
```

A3 – Percentagem relevante na segmentação

O extracto de código do método geraSugestões representa a forma como a segmentação do utilizador é relevante para a apresentação das sugestões.

Nota: O método *"public string updatePropriedade(string user_id, string key, string propriedade, string valor, bool incremento)"* que é utilizador no método *geraSugestoes* desta secção incrementa o *"valor"* ou regista-o conforme o parâmetro *"incremento"* seja verdadeiro ou falso respectivamente.

```
private void geraSugestoes(string user_id, string key)
{
       double[] segmentacao = getUserSegmentation(user_id);
```

```
        double[] show_sugestoes = getIncerteza(user_id);

        DBConnect bd = new DBConnect();
        ArrayList tags = new ArrayList();
        tags.Add("valor");

        int contador=0;
        ArrayList result;
        result = bd.Select("select valor from propriedade where propriedade=114
                            and user_id=" + user_id, tags);
        if (result.Count > 0)
               contador = Convert.ToInt32(((ArrayList)result[0])[0]);

        if (segmentacao[0] > 0.1 || (segmentacao[0] > 0.0 && contador>10))
        {
               //procurar sugestões de jogos
        }
        else
               updatePropriedade(user_id, key, "114", "1", true);

        contador = 0;
        result = bd.Select("select valor from propriedade where propriedade=115
                            and user_id=" + user_id, tags);
        if (result.Count > 0)
             contador = Convert.ToInt32(((ArrayList)result[0])[0]);

        if (segmentacao[1] > 0.1 || (segmentacao[1] > 0.0 && contador > 10)) {
               // procurar sugestões de filmes
        }
        else
               updatePropriedade(user_id, key, "115", "1", true);

        contador = 0;
        result = bd.Select("select valor from propriedade where propriedade=116
                            and user_id=" + user_id, tags);
        if (result.Count > 0)
               contador = Convert.ToInt32(((ArrayList)result[0])[0]);

        if (segmentacao[2] > 0.1 || (segmentacao[2] > 0.0 && contador > 10))
        {
               //procurar sugestões de series
        }
        else
               updatePropriedade(user_id, key, "116", "1", true);


}
```

A4 – Actualizar propriedade

A estrutura “se-então” que se segue mostra como após uma acção relevante do utilizador a aplicação cliente actualiza a propriedade referente a essa acção.

```
if (temporadas.Count == 0)
       await ws.updatePropriedadeAsync(MainPage.user_id, MainPage.user_key, "62",
```

```
                                        "1", true);
else
        await ws.updatePropriedadeAsync(MainPage.user_id, MainPage.user_key, "61",
                                        "1", true);
```

A5 – Actualizar propriedade

No código que se segue é visível o tratamento do retorno do método que retorna a segmentação do utilizador. Este retorno é então armazenado em variáveis locais na aplicação cliente.

```
if (Convert.ToInt32(MainPage.user_id) > 0)
{
        GFS_WS.getUserSegmentationResponse belief_result = await
        ws.getUserSegmentationAsync(MainPage.user_id, MainPage.user_key);
        string seg = belief_result.Body.getUserSegmentationResult;
        JObject obj = JObject.Parse(seg);
        MainPage.belief_g=Convert.ToDouble(obj["belief_g"].ToString());
        MainPage.belief_f=Convert.ToDouble(obj["belief_f"].ToString());
        MainPage.belief_s = Convert.ToDouble(obj["belief_s"].ToString());
}
```

A6 – Análise de pertinência

No extracto de código que se segue é visível como a componente servidora utiliza para os cálculos acerca da apresentação de sugestões, não só a segmentação do utilizador como também a incerteza sobre a pertinência de visualização de um dado grupo de sugestões.

```
private void geraSugestoes(string user_id, string key)
{
        double[] segmentacao = getUserSegmentation(user_id);
        double[] show_sugestoes = getIncerteza(user_id);

        if (segmentacao[0] > 0.1 || (segmentacao[0] > 0.0 && contador>10))
        { //procurar sugestões de jogos
                //identical game
                if (show_sugestoes[0]>.5)
                        geraSugestoesTipo(user_id, key, 0);
                //new version game
                if (show_sugestoes[1]>.5)
                        geraSugestoesTipo(user_id, key, 1);

        }

        if (segmentacao[1] > 0.1 || (segmentacao[1] > 0.0 && contador > 10))
        { //procurar sugestões de filmes
                //identical film
                if (show_sugestoes[2]>.5)
                        geraSugestoesTipo(user_id, key, 2);
                //new version film
                if (show_sugestoes[3]>.5)
                        geraSugestoesTipo(user_id, key, 3);
```

```
        }
        if (segmentacao[2] > 0.1 || (segmentacao[2] > 0.0 && contador > 10))
        { //procurar sugestões de series
               //identical serie
               if (show_sugestoes[4]>.5)
                      geraSugestoesTipo(user_id, key, 4);
               //new season
               if (show_sugestoes[5]>.5)
                      geraSugestoesTipo(user_id, key, 5);
               //new episode
               if (show_sugestoes[6]>.5)
                     geraSugestoesTipo(user_id, key, 6);
         }
```

A7 – Implementação de pesquisa

A classe que se segue demonstra como como uma aplicação metro deve tratar o evento de pesquisa quando este ocorre.

```
public sealed partial class Procura : Store_Share_GFS.Common.LayoutAwarePage
{

       protected async override void LoadState(Object navigationParameter,
Dictionary<String, Object> pageState)
        {
              String query = (string)navigationParameter;
              // REALIZAR A PESQUISA E APRESENTAR O RESULTADO;
         }

       public static void Activate(SearchActivatedEventArgs args)
       {
            string queryText = args.QueryText.ToLower();

            frame.Navigate(typeof(Procura), queryText);
            Window.Current.Activate();
       }

}
```

A8 - receber dados partilhados

A classe que se segue demonstra como uma aplicação metro deve tratar o evento de receber uma partilha, ou seja, receber um pacote de dados de outra aplicação que usa o mesmo protocolo.

```
public sealed partial class Partilha : Store_Share_GFS.Common.LayoutAwarePage
{

       private string dados;
       private Windows.ApplicationModel.DataTransfer.ShareTarget.ShareOperation
       _shareOperation;
```

```
        DataTransferManager dataTransferManager =
        DataTransferManager.GetForCurrentView();

         public Partilha()
         {
             this.InitializeComponent();
               dataTransferManager.DataRequested += new
               TypedEventHandler<DataTransferManager,
             DataRequestedEventArgs>(this.ShareTextHandler);
         }

        public async void Activate(ShareTargetActivatedEventArgs args)
         {
             this._shareOperation = args.ShareOperation;
             dados = this._shareOperation.Data.GetTextAsync().GetResults()

             var shareProperties = this._shareOperation.Data.Properties;
             var thumbnailImage = new BitmapImage();
             this.DefaultViewModel["Title"] = shareProperties.Title;
             this.DefaultViewModel["Description"] = shareProperties.Description;
             this.DefaultViewModel["Image"] = thumbnailImage;
             this.DefaultViewModel["Comment"] = String.Empty;
             this.DefaultViewModel["SupportsComment"] = true;
             Window.Current.Content = this;
             Window.Current.Activate();
             if (shareProperties.Thumbnail != null)
             {
                 var stream = await shareProperties.Thumbnail.OpenReadAsync();
                 thumbnailImage.SetSource(stream);
                 this.DefaultViewModel["ShowImage"] = true;
             }
         }

        private async void ShareButton_Click(object sender, RoutedEventArgs e)
        {

             this.DefaultViewModel["Sharing"] = true;
             this._shareOperation.ReportStarted();
             if (dados != null)
              {

                      //TRATAR OS DADOS RECEBIDOS E GUARDA-LOS

               }

        }

}
```

A9 - partilhar dados

A classe que se segue demonstra como uma aplicação metro deve tratar o evento de partilhar, ou seja, enviar um pacote de dados para outra aplicação que reconheça o mesmo protocolo.

```
public sealed partial class addFilm : Store_Share_GFS.Common.SharePage
{
```

```
    private DataTransferManager dataTransferManager;

    protected override bool GetShareContent(DataRequest request)
        {
            bool succeeded = false;

            String titulo = txtTitulo.Text;
            if (cmbTitles.SelectedIndex != 0)
                titulo = cmbTitles.SelectedItem+"";

            if (!String.IsNullOrEmpty(titulo))
            {
               String dataPackageText = " Film: " + titulo + "\n Gender: " +
              cmbGenero.SelectedItem + "\n Year: " + cmbAno.SelectedItem + "\n
              Duration: " + txtDuracao.Text + "\n Avaliation: " +
              cmbAval.SelectedItem + "\n Actores: " + txtActores.Text;

                DataPackage requestData = request.Data;
                requestData.Properties.Title = MainPage.user + " shared the film
                " + titulo + " with you.";
                requestData.Properties.Description = "Share of film (" + titulo
                + ") by GFS"; // The description is optional.
                requestData.SetText(dataPackageText);
                requestData.SetHtmlFormat(dataPackageText.Replace("\n","<br/>"));
                succeeded = true;
            }
            else
            {
                    request.FailWithDisplayText("Enter the text you would like
                    to share and try again.");
            }
            return succeeded;
        }

}
```