Universidade Federal do Rio de Janeiro
Instituto de Matemática - IM
Departamento de Ciência da Computação - DCC

TirinhaZ – Um Sistema de Recomendação de Tirinhas

Autores

Monografia de Conclusão do Curso de Ciência da Computação
Fabio Perrotta de Andrade e Pedro Henrique Conilh de Beyssac Ramos

Orientador: Geraldo Zimbrão da Silva – D.Sc. COPPE/UFRJ

Coorientador: Filipe Braida do Carmo – M.Sc. COPPE/UFRJ

Rio de Janeiro, 2013

Fabio Perrotta de Andrade
Pedro Henrique Conilh de Beyssac Ramos

TirinhaZ – Um Sistema de Recomendação de Tirinhas

Projeto Final de Curso submetido ao Departamento de Ciência da Computação do Instituto de Matemática da Universidade Federal do Rio de Janeiro como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciências da Computação.

Professores orientadores:

Geraldo Zimbrão da Silva – D.Sc. COPPE/UFRJ

Filipe Braida do Carmo – M.Sc COPPE/UFRJ

Rio de Janeiro, 2013

TirinhaZ: Sistema de Recomendação de Tirinhas

Fabio Perrotta de Andrade e Pedro Henrique Conilh de Beyssac Ramos

Projeto Final de Curso submetido ao Departamento de Ciência da Computação do Instituto de Matemática da Universidade Federal do Rio de Janeiro como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciências da Computação.

APROVADA em ____ de _____ de 2013.

Apresentado por:

_____
Fabio Perrotta de Andrade

_____
Pedro Henrique Conilh de Beyssac Ramos

Aprovado por:

_____
Geraldo Zimbrão da Silva – D.Sc. COPPE/UFRJ

_____
Filipe Braida do Carmo – M.Sc. COPPE/UFRJ

_____
Jano Moreira de Souza – D.Sc. COPPE/UFRJ

_____
Carlos Eduardo Melo – M.Sc. COPPE/UFRJ

RIO DE JANEIRO, RJ – BRASIL
FEVEREIRO DE 2013

Agradecimentos

Fabio Perrotta de Andrade

Primeiramente agradeço aos meus pais Walter Barboza e Angelina Perrotta por me ensinarem a ser uma pessoa que valoriza a educação e a ética, não deixando de lado a humildade. Agradeço às minhas irmãs, Deborah e Patrícia Perrotta, aos meus falecidos avós Walter e Léa Barboza, e aos meu avós ainda com vida, Antônio e Maria Perrotta, por me apoiarem sempre em tudo que faço, me dando forças para superar os obstáculos que a vida nos impõe.

Agradeço também aos amigos que conquistei ao longo da faculdade, que não se negavam em momento algum a ajudar quem estivesse precisando entender melhor a matéria. Sem eles eu não chegaria a conclusão do curso. Um agradecimento especial ao Filipe Braida, pois além de ser um amigo, me orientou e ajudou diretamente nesta monografia.

E, por fim, porém não menos importante, agradeço pelo grande apoio e incentivo dos meus amigos anteriores a faculdade, que me incentivaram muito para a conclusão de mais essa etapa.

Agradecimentos

Pedro Henrique Conilh de Beyssac Ramos

Agradeço primeiramente ao meu pai, José Airton, por todo esforço em me proporcionar apoio em todos os momentos de minha vida, transmitindo seus valores de persistência, honestidade e sabedoria. Agradeço também a minha namorada, Paloma Freire, por estar ao meu lado me dando suporte emocional e estímulo para persistir sempre. Agradeço aos amigos, Filipe Braida, Fernando Magalhães , César Barbosa e Danielle Caled, que contribuíram para que esse projeto fosse realizado, ajudando a buscar soluções e darem conselhos importantes.

“A tarefa não é tanto ver aquilo que ninguém viu, mas pensar o que ninguém ainda pensou sobre aquilo que todo mundo vê.”

Arthur Schopenhauer

RESUMO

TirinhaZ: SISTEMA DE RECOMENDAÇÃO DE TIRINHAS

Fabio Perrotta de Andrade e Pedro Henrique C. Beyssac Ramos

Orientadores: Geraldo Zimbrão da Silva e Filipe Braida do Carmo

Devido ao enorme volume de informação encontrado na Internet, procurar algo em específico torna-se uma tarefa que demanda cada vez mais tempo do usuário. Os Sistemas de Recomendação surgiram com a intenção de realizar uma filtragem dessas informações, reduzindo assim o tempo despendido pelo usuário na sua busca. Eles combinam várias técnicas computacionais para realizar recomendações personalizadas com base nos interesses dos usuários e conforme o contexto no qual estão inseridos.

Uma grande dificuldade na área dos Sistemas de Recomendação é a captação de dados, os quais possibilitam a realização de testes dos algoritmos de recomendação. Atualmente, esse tipo de sistema é muito usado em lojas de comércio online, por exemplo, que vendem produtos considerados de lento consumo, como eletrodomésticos, eletrônicos, etc.

Essa monografia aborda um sistema de recomendação de tirinhas, que utiliza a Filtragem Colaborativa como técnica de recomendação. Esse produto foi escolhido devido a sua característica de consumo rápido, possibilitando assim que um grande volume de informação seja recolhido em um curto espaço de tempo. O TirinhaZ foi desenvolvido com o intuito de acolher diferentes técnicas de recomendação sem a necessidade de grandes modificações em sua estrutura e facilitar o teste de diferentes algoritmos de recomendação.

Palavras-chave: Sistemas de Recomendação, MDA, MDArte, filtragem colaborativa, rede social.

ABSTRACT

Título Monografia

Fabio Perrotta de Andrade e Pedro Henrique C. Beyssac Ramos

Supervisors: Geraldo Zimbrão da Silva and Filipe Braida do Carmo

Searching for a specific thing on the Internet become a hard task due to the great volume of information, which takes more and more time from the user. The Recommendation Systems appeared in order to filter that amount of information, reducing the time users spend on their search. Such Systems match a number of computer techniques to make personalized recommendations based on users' interests and on the context they are put in.

One of the great difficulties of the Recommendation Systems is the capture of data which enable the recommendation algorithm tests. Currently this type of system is very used in online stores, for example, which are considered slow consumption products, like domestic appliance, electronics and so on.

This monography broaches a recommendation system of comic strips which uses the Colaborative Filtering as a recommendation technique. This product has been chosen due to your characteristic of rapid consumption, enabling that a great volume of information be collected in a faster way. The TirinhaZ has been developed in order to gather different techniques of recommendation without needing to make great modifications in your structure and, therefore, to make it easy the testing of different recommendation algorithms.

Keywords: Recommendation Systems, MDA, MDArte, collaborative filtering, social network.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

1.1 Motivação

Quanto mais o tempo passa, mais opções e oportunidades o mundo nos oferece, seja na aquisição de um eletrônico, de um carro, de um serviço, ou na escolha de um livro, de um filme, de uma roupa. Devido a esse cenário, é muito comum o surgimento de dúvidas no momento da escolha de um produto, e, consequentemente, pessoas recorrem a experiências alheias a fim de tentar obter informações sobre o objeto de desejo podendo, assim, tomar uma decisão mais racional.

Transbordando a situação acima para o meio digital, a quantidade de informação e serviços existentes nesse meio e seu crescimento é um fator que vem ocorrendo de forma exponencial nas últimas décadas. Grande parte, não somente devido à facilidade de se criá-la com as novas tecnologias de informação e comunicação, mas também à possibilidade de uma maior capacidade de armazenamento de dados devido ao barateamento dos dispositivos responsáveis por isto.

Unindo tal fato com a centralização dessas informações proporcionada pela internet e suas ferramentas de busca, como o *Google*[1] por exemplo, torna-se cada vez mais difícil achar exatamente o que se está procurando. Para isso, tornou-se necessário o investimento em pesquisas de métodos inteligentes de apresentação e filtragem dos dados, a fim de facilitar não só quem carece de um algo, mas aquele que fornece a informação.

Nessa conjuntura surge o conceito de recomendação, que é a filtragem de dados utilizando algum nível de inteligência para otimizar uma certa busca. Para tal fim, existem inúmeras

1 http://google.com

técnicas que podem ser aplicadas a mecanismos de busca por um produto. Pode-se citar entre elas: métodos que utilizam notas previamente dadas por outros avaliadores ou até mesmo algoritmos que levam em consideração o conteúdo dos itens a serem recomendados de acordo com o perfil de cada usuário.

Paralelamente a isso, ao longo dos últimos anos a Internet foi inundada de informações de rápido consumo para diversão. Sites como *9gag*[2], por exemplo, possuem grande volume de dados que se propõe a entreter o usuário com imagens ou tirinhas em quadrinhos. Mas a maioria não aplica algum tipo de técnica que permita que o usuário tenha uma experiência personalizada de acordo com seu gosto, apenas sorteiam de forma aleatória e apresentam ao usuário. Com esse trabalho pretende-se demonstrar a aplicabilidade de técnicas de recomendação e utilizá-las em um cenário real. Para isso foi escolhido um produto de rápido consumo (tirinhas), a fim de recolher uma maior quantidade de informação em menos tempo. O produto final desse trabalho trata-se de um sistema de recomendação de tirinhas cômicas.

1.2 Objetivo do Trabalho

Este projeto tem como propósito o desenvolvimento e a implementação de um sistema multiusuário de recomendação de tirinhas através de uma aplicação *web* colaborativa. Através dele os usuários terão a possibilidade de ler, incluir e compartilhar tiras de humor.

Com este sistema, pretende-se proporcionar um ambiente em que o usuário compartilhe sua opinião sobre o conteúdo, atribuindo uma nota a ele. Com isso, o sistema recomendará ao usuário itens relevantes e agradáveis, essa recomendação será feita de forma personalizada, de acordo com a identificação das preferências do usuário.

E por fim, esse sistema será modelado com o objetivo de facilitar a utilização de diferentes tipos de recomendação, para que seja utilizado como base para análise de diferentes algoritmos.

---

2 Http://www.9gag.com

1.3 Organização do trabalho

Os capítulos 2 e 3 deste trabalho discutem conceitos referentes à fundamentação teórica, abordando, respectivamente, os conceitos de sistemas de recomendação e desenvolvimento dirigido por modelos. O capítulo 4 apresenta a proposta do trabalho, descrevendo seus principais requisitos. No capítulo 5 serão discutidos os aspectos sobre implementação da solução proposta, as dificuldades encontradas e para finalizar, o Capítulo 6 apresenta a conclusão deste trabalho.

2 SISTEMA DE RECOMENDAÇÃO

2.1 Cenário Geral

Pode-se observar que pessoas usualmente antes de experimentarem algo novo, ou uma situação diferente, recorrem a experiência de outras em busca de mais detalhes. Isso ocorre devido ao sentimento que o desconhecido traz; a incerteza sobre decisões é algo que não é, em sua maioria das vezes, bem-vinda. Como não se pode prever a consequência de uma escolha, tenta-se então aproximar o resultado ao que se deseja obter. Apesar de muito genérico esse conceito, isso é totalmente válido para qualquer tipo de escolha, incluindo também para bens de consumo. Tudo que é possível usar ou fazer é passível de dúvidas na hora de uma decisão e elas são muitas a serem tomadas no dia a dia. Optar por: livros, filmes, eletrônicos, produtos de limpeza, revistas, carros, destinos de viajem, roupas, restaurantes entre outras. Junto a isso, existe a internet, que nesse contexto age como um concentrador de opções em um único lugar.

Um contingente gigantesco de dados encontra-se misturado na rede e devido a essa grande quantidade de conteúdo, surgiu a necessidade de criar ferramentas que auxiliem na busca por informações relevantes. Foi assim que surgiram, por exemplo, ferramentas como a de buscas do *Google*[3]. Essas ferramentas de busca foram e são muito úteis na hora de se procurar por uma informação. Porém, com o enorme volume de dados existente na internet, essas ferramentas retornam, às vezes, milhares de resultados, fazendo com que o usuário tenha que "filtrar" manualmente as informações relevantes para ele.

---

3 http://google.com

Entretanto, ao longo das últimas décadas, a internet além de um grande concentrador de informação, passou também a concentrar bens e serviços. Entre os serviços encontra-se a venda de produtos on-line por exemplo. Cada vez mais torna-se comum ver empresas trocarem suas lojas físicas por lojas virtuais. Muitos são os motivos dessas mudanças, menores custos, rapidez no atendimento e o mais importante, personalização do atendimento ao cliente.

Visando essa personalização e otimização da entrega de informações ao cliente, surgiram então os Sistemas de Recomendação (Primo & Loh, 2006). Segundo Burke, um Sistema de Recomendação é qualquer sistema que entrega ao usuário uma recomendação individual ou que mostre o caminho ao usuário, que o guie à informação de seu interesse.

O objetivo dos Sistemas de Recomendação é fornecer recomendações personalizadas para cada usuário, partindo do princípio que cada um deles seja único. Esses sistemas vêm sendo implementados mediante o uso de diversas técnicas, que variam de acordo com a dinâmica e a forma de recomendação.

Dentro da literatura, os Sistemas de Recomendação podem ser divididos em diversas abordagens devido a sua forma de recomendar, de acordo com a variação dos aspectos citados no parágrafo acima. Contudo, neste projeto serão abordadas três dessas técnicas:

1. Recomendação baseada em conteúdo;
2. Filtragem colaborativa;
3. Abordagem Híbrida.

2.2 Abordagens de Recomendação

2.2.1 Baseado em Conteúdo

O método de filtragem baseada em conteúdo utiliza-se das informações e características dos produtos para poder recomendá-los. Em outras palavras, esses algoritmos recomendam produtos com aspectos similares àqueles que já foram bem aceitos pelo usuário. Por exemplo, em um sistema de recomendação de filmes, as melhores avaliações dadas por um usuário são

nos filmes de ação. Então esse padrão de comportamento é identificado e o sistema ficará responsável por recomendar filmes com essa característica. Além do gênero do filme poderiam ser utilizados outros atributos como elenco, diretores, etc.

Uma das dificuldades desse tipo de abordagem é encontrar uma similaridade entre os itens oferecidos, principamente quando o sistema é alimentado por produtos de categorias diferentes e, consequentemente, atributos distintos. Por exemplo, uma loja de departamentos virtual vende, entre outros itens, roupas e eletrônicos. Ou seja, normalmente características típicas de uma roupa(tamanho, tecido) não são as mesmas de um eletrônico(peso, fonte de energia). A fim de encontrar alguma similaridade, seria preciso levar em conta aspectos mais abstratos do produto, como: sexo do público-alvo, faixa etária do consumidor, média salarial do comprador, informações mais gerais relacionadas a produtos cujas especificações são totalmente distintas.

Os sistemas de filtragem baseada em conteúdo possuem algumas vantagens: (Machado & Cazella, 2006)

- **Independência do número de usuários**: a análise da similaridade é feita entre avaliações de itens de um mesmo usuário. Portanto, uma quantidade significativa de usuário não influencia na qualidade da recomendação;

- **Gostos únicos**: pelo mesmo motivo acima, essa técnica consegue recomendações bem específicas por se basear apenas no histórico do próprio usuário;

- **Itens novos e não populares**: mesmo um produto que possua poucas avaliações ou nenhuma, tem a mesma chance de ser recomendado que um produto mais antigo.

Por outro lado, essa técnica baseada em conteúdo possui algumas limitações (Adomavicius & Tuzhilin, 2005; Burke, 2002 apud Oliveira, 2007):

- **Análise de conteúdo limitada**: as características dos produtos são estruturadas de forma que seja possível tratá-las automaticamente pelo computador. A extração de características de dados multimídia como som e vídeo, por exemplo, é de difícil aplicação, e extraí-las manualmente é inviável;

- **Superespecialização**: por essa técnica se basear nas avaliações passadas do próprio usuário para recomendar um novo produto, o usuário acaba tendo apenas recomendações de produtos muito similares aos que ele já experimentou. Por exemplo, um usuário que nunca experimentou um livro de suspense, jamais receberá uma recomendação do mesmo.

- **Elasticidade versus plasticidade**: esse problema ocorre quando um usuário, que já tenha um perfil consolidado com várias avaliações, muda de preferência ou de gosto. Exemplo: um usuário que costuma comprar artigos esportivos, após um acidente passa a não consumir mais tais produtos. Porém continuará a receber recomendações de produtos da categoria esportes até haver avaliações suficientes para mudar o seu perfil no sistema.

2.2.2 Filtragem Colaborativa

O termo Filtragem Colaborativa foi primeiramente escrito num artigo chamado: *Using collaborative filtering to weave an information tapestry (Goldberg, Nichols, Oki, & Terry, 1992),* no qual falava acerca de um sistema experimental desenvolvido no centro de pesquisa da empresa Xerox que possibilitava a colocação de anotações nos documentos pelos usuários, marcando-os como interessante ou desinteressante. O sistema incluía suporte para filtragem colaborativa e baseada em conteúdo, sendo aperfeiçoado a partir das ações tomadas pelos usuários ao lerem as mensagens. Com isso, filtrava os documentos para seus usuários (SEGARAN, 2008). O intuito de tal trabalho era mostrar que a filtragem de informação pode ser mais efetiva se houverem pessoas envolvidas no processo.

Contrastando com a filtragem baseada em conteúdo, a filtragem colaborativa dispensa informações sobre os itens a serem recomendados, pois baseia-se na similaridade de usuários para gerar a recomendação. Por exemplo, o usuário Paulo explicita sua preferência sobre alguns produtos. Em seguida, é identificado um subconjunto de usuários que possuem um perfil similar ao de Paulo. Logo, é possível predizer a preferência dos usuários desse subconjunto, que será a mesma de Paulo.

A Filtragem Colaborativa possui algumas vantagens quando comparada a baseada em conteúdo. Utilizando essa técnica o usuário poderá receber recomendações de itens que não possuem relação alguma com itens previamente avaliados pelo mesmo, mas que se encaixam no perfil similar de outros usuários. Outra vantagem é que ela não necessita de informações sobre o item e, consequentemente, isso não afeta a qualidade da recomendação.

Entretanto,a Filtragem Colaborativa também possui desvantagens, como:

- **Cold-start:** existem dois casos de cold-start: novo usuário e novo item. O primeiro se refere a um usuário recém cadastrado no sistema que possui nenhum ou poucas recomendações de produtos. Ou seja, o perfil desse usuário é muito restrito, implicando numa recomendação pouco confiável. Já o segundo caso (novo item) se refere a um item que possui nenhuma ou poucas recomendações e, consequentemente, esse item raramente será recomendado;

- **Dispersão da base de dados:** quando o sistema possui poucos usuários, torna-se mais difícil encontrar similaridades entre perfis;

- **Similaridade:** alguns usuários podem apresentar um perfil bem diferente de outros, o que dificulta o sistema de encontrar recomendações para este usuário.

Apesar desses obstáculos descritos acima, a filtragem colaborativa alcançou um lugar de grande importância se comparada com outras abordagens de recomendação. A apresentação de bons resultados, a facilidade de implementação, a não utilização de conhecimento adicional além da prórpia dinâmica da recomendação são algum dos motivos desse sucesso.

Essa esparsidade e a escalabilidade ainda são objetos de estudo de cientistas da área. A fim de direcionar esses estudos, essa abordagem é dividida em duas classes de algoritmos: baseada em memória e baseada em modelo (Adomavicius & Tuzhilin, 2005).

2.2.2.1 Modelo

Os algoritmos baseados em modelo usam os dados existentes para criar um modelo que simule o comportamento dos usuários. É através desse modelo que é possível reconhecer determinados padrões que não seriam revelados por outros métodos. Para criar esses modelos é

preciso o auxílio técnico de áreas distintas da literatura como Decomposição de Matrizes (DM), probabilidade e Aprendizado de Máquina (AM).

A filtragem colaborativa baseada em modelo pode ser dividida em: baseada no usuário ou baseada no item. A primeira constrói um modelo com base no usuário, podendo usar uma técnica chamada de clusterização, na qual usuários similares são acoplados em *clusters*. O objetivo desse modelo é calcular a probabilidade de um usuário pertencer a um determinado *cluster*, possibilitando prever as avaliações desse usuário.

Já a baseada no item constrói o modelo com base no item, utilizando medidas de similaridade como a Probabilidade Condicional (Deshpande & Karypis, 2004) que pode ser expressa de acordo com a Equação 1.

$$P(j|i) = \frac{(Fr(i,j))}{(Fr(i))}$$

Equação 1: Probabilidade Condicional

Onde $P(j|i)$ é a probabilidade de se comprar o item j sendo o produto i já comprado e $Fr(x)$ corresponde ao número de usuários que compraram os itens no conjunto x. Essa probabilidade é a divisão entre o número de consumidores que compraram os itens i e j e o número total de usuários que compraram i. O resultado da construção desse modelo pode ser expresso através de uma matriz de similaridade S, que contém os resultados da probabilidade condicional, conforme mostrado na Figura 1.

	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8
i_1	1	0,1	0	0,3	0,2	0,4	0	0,1
i_2	0,1	1	0,8	0,9	0	0,2	0,1	0
i_3	0	0,8	1	0	0,4	0,1	0,3	0,5
i_4	0,3	0,9	0	1	0	0,3	0	0,1
i_5	0,2	0	0,4	0	1	0,1	0	0
i_6	0,4	0,2	0,1	0,3	0,1	1	0	0,1
i_7	0	0,1	0,3	0	0	0	1	0
i_8	0,1	0	0,5	0,1	0	0,1	0	1

Figura 1: Matriz de Similaridade entre itens S

Baseado nessa matriz, a recomendação de uma lista contendo N itens a um usuário, pode ser feita utilizando o algoritmo de recomendação Top-N baseado em item, proposto por Karypis (2001), que será explicado a seguir.

Considere $U=\{i1, i5, i8\}$ sendo o conjunto de itens comprados pelo usuário (u). O primeiro passo desse algoritmo é identificar um conjunto C de itens candidatos a serem recomendados. Isso é feito unindo os K itens mais similares para cada item $i \in U$. Dessa união, remove-se os itens que já foram comprados pelo usuário. Considerando $k=3$ e utilizando os dados da matriz S da Figura 1, o conjunto C será formado pelos itens $C=\{i3, i4, i5\}$. Agora a similaridade entre cada item $c \in C$ e o conjunto U é calculada através da soma das similaridades entre todos os itens $i \in U$ e c, usando somente os K itens mais similares de j. O resultado desse cálculo da similaridade é mostrado na Figura 2, onde os valores em negrito correspondem aos 3 itens mais similares aos itens $i \in U$, estes que por sua vez correspondem às linhas destacadas.

	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8
i_1	1	0,1	0	**0,3**	**0,2**	**0,4**	0	0,1
i_2	0,1	1	0,8	0,9	0	0,2	0,1	0
i_3	0	0,8	1	0	0,4	0,1	0,3	0,5
i_4	0,3	0,9	0	1	0	0,3	0	0,1
i_5	**0,2**	0	**0,4**	0	1	**0,1**	0	0
i_6	0,4	0,2	0,1	0,3	0,1	1	0	0,1
i_7	0	0,1	0,3	0	0	0	1	0
i_8	**0,1**	0	**0,5**	**0,1**	0	0,1	0	1
			0,9	0,4		0,5		

Figura 2: Soma das similaridades de todos os itens $i \in U$ e c.

$$k=3\,;\, ua=\{i1, i5, i8\}$$

Finalmente, os itens em C são classificados em ordem decrescente de similaridade, e os primeiros N itens são selecionados como o conjunto de itens top-N a serem recomendados. Neste exemplo, os $N=2$ itens recomendados seriam $i6$ e $i3$.

A Filtragem colaborativa baseada em modelo supera o problema da escalabilidade da baseada em memória (sessão 2.2.2.2), pois quando uma recomendação é requisitada, as similari-

dades são buscadas no modelo que está previamente construído, e não calculada em tempo real.

2.2.2.2 Memória

Também conhecidos como algoritmos baseados em heurísticas (Adomavicius & Tuzhilin, 2005), os algoritmos baseados em memória utilizam-se de heurísticas para tentar chegar a melhores resultados. Estas são denotadas por $\mathrm{sim}(xy)$, que mede a similaridade dos usuários x e y. As recomendações para um determinado usuário são calculadas através de uma função que tem como parâmetros as avaliações dadas pelos N usuários similares em relação aos itens ainda desconhecidos por ele.

Dado um usuário u e um item i, ainda não avaliado por u, é necessário prever qual será a avaliação de u em relação à i, denotada por $r_{u,i}$. Assume-se que o item i já foi avaliado por um conjunto U de usuários. Assim, a previsão será dada por uma função que agregará as avaliações de i dadas por cada membro u' de U. Abaixo se encontram algumas das funções de agregação que podem ser utilizadas, onde $r(x)$ é a média das notas dadas pelo usuário x e β é um fator de normalização explicitado na Equação 5.

$$r_{u,i} = \frac{1}{|U|} \sum_{(u' \in U)} r_{u',i}$$

Equação 2: Função de agregação baseada na média.

$$r_{u,i} = \sum_{(u' \in U)} \mathrm{sim}(u,u').r_{u',i}$$

Equação 3: Função de agregação baseada na média ponderada.

$$r_{u,i} = r(u) + \beta . \sum_{(u' \in U)} \mathrm{sim}(u,u').(r_{u',i} - r(u'))$$

Equação 4: Função de agregação baseada na média com fator de normalização.

$$\beta = 1 / \sum_{(u' \in U)} |\mathrm{sim}(u,u')|$$

Equação 5: Fator de normalização.

O primeiro caso (Equação 2) é o mais simples, pois a função de agregação é simplesmente a média das notas dadas pelos usuários similares. A função representada na Equação 3 é uma das mais populares na literatura, ela introduz o peso da similaridade no calculo da média. Contudo, na Equação 4 é levado em consideração a variação das notas em relação a média de cada usuário. Analisando ainda a mesma equação, ela possui um fator de normalização das similaridades. A importância dessa normalização se fundamenta no fato de que a medida de similaridade é uma heurística, que pode variar de acordo com a aplicação. Sendo assim, é importante que a escala de similaridade seja a mesma para todos os tipos de heurísticas.

Cosseno e a correlação de *Pearson* são as medidas de similaridade mais populares na literatura. Dados dois usuários, i e j, retira-se dois vetores da matriz usuário item, contendo as avaliações dos itens em comum que foram avaliados por esses usuários. O conjunto dado pela interseção dos itens avaliados de i e j é denotado por I. A medida do cosseno expressa na Equação 6 é dada pelo cosseno do ângulo formado por esses dois vetores:

$$\cos(i,j)=\frac{(\sum_{(k\in I)} r_{i,k}.r_{j,k})}{(\sqrt{(\sum_{(k\in I)} r_{i,k}^2)}.\sqrt{(\sum_{(k\in I)} r_{j,k}^2)})}$$

Equação 6: Cálculo da similaridade através do cosseno.

A medida da correlação de *Pearson* na Equação 7 é bastante parecida com a do *cosseno*, a diferença é que são subtraídas as médias de cada usuário objetivando a normalização das avaliações:

$$\cos(i,j)=\frac{(\sum_{(k\in I)} (r_{i,k}-r(i)).(r_{j,k}-r(j)))}{(\sqrt{(\sum_{(k\in I)} (r_{i,k}-r(i))^2)}.\sqrt{(\sum_{(k\in I)} (r_{j,k}-r(j))^2)})}$$

Equação 7: Cálculo da similaridade através do coeficiente de correlação de Pearson.

Existem outras medidas de similaridade como a Frequência Inversa de usuário (FIU), que propõe a multiplicação das notas por um fator que penaliza avaliações de itens muito populares, pois esses não agregam muito valor no cálculo da similaridade. Nessa mesma linha de raciocínio, a Amplificação de Caso (AC) propõe a multiplicação dos pesos por um fator exponencial, enfatizando os pesos maiores, pois são eles que mais contribuem para o cálculo . O

Voto por Padrão (VP) consiste em colocar notas com valor padrão quando a interseção dos itens avaliados é muito pequena.

Voltando ao tema principal, a Filtragem Colaborativa baseada em memória possui algumas limitações. Uma delas é que todo cálculo é feito sobre notas de itens dadas por usuários, sendo que quando se tem um item ou um usuário novo, essa informação não existe (problema do Cold-Start, abordado em 2.2.2). Outra limitação consiste na quantidade de usuários e itens, tornando o cálculo da similaridade muito custoso, pois é preciso iterar em todos os elementos. Uma forma de contornar essa problema é fazendo previamente os cálculos de similaridades e atualizá-los em um intervalo de tempo razoável, já que não são tão voláteis.

Analisando por uma outra perspectiva, existe uma abordagem conhecida como Filtragem Colaborativa baseada em itens. Ela consiste em calcular a similaridade entre itens, não mais entre usuários, e mostra resultados tão bons quanto as Filtragens Colaborativas tradicionais. Uma das justificativas de ser ver o problema dessa forma, é que a similaridade dos itens é considerada estática, o que não se pode dizer o mesmo sobre os usuários. E, consequentemente, os recálculos de similaridade para os itens são bem menos frequentes.

2.2.3 Abordagem Híbrida

Os algoritmos híbridos consistem na mistura de técnicas distintas de filtragem. Eles surgiram com o objetivo de reforçar os pontos fortes de um determinada técnica e minimizar as limitações apresentadas com o uso de apenas uma técnica (Huang, Chung, Ong, & Chen, 2002).

De modo geral algumas técnicas utilizam características da filtragem baseada em conteúdo na colaborativa. Usando essa combinação, pode-se alcançar os benefícios da Filtragem Baseada no Conteúdo pois, independente do número de usuários e avaliações dos itens, inclui a predição para todos os usuários e itens. Somado a isso, oferece exatidão das predições com Filtragem Colaborativa conforme o número de usuários e recomendações cresce (Claypool, Gokhale, & Miranda, 1999).

Existem diferentes maneiras de combinar métodos de filtragem colaborativa e filtragem baseada em conteúdo como, por exemplo, incorporar características da abordagem baseada em conteúdo em uma abordagem colaborativa. Outra forma seria implementar os métodos baseados em conteúdo e colaborativos separadamente e depois combinar as predições (Adomavicius & Tuzhilin, 2005).

As principais características herdadas pela abordagem híbrida daquelas descritas acima são (Lopes, 2007):

- Precisão independente do número de usuários;
- Bons resultados para usuários incomuns;
- Descoberta de novos relacionamentos entre usuários;
- Recomendação de itens diretamente relacionados ao histórico.

As duas primeiras características são herdadas da Filtragem Baseada em Conteúdo, pois não levam em consideração qualquer relacionamento entre perfis de usuários e não recomendam itens não relacionados ao perfil do usuário. Enquanto que as duas últimas são herdadas da Filtragem Colaborativa, pois trabalha com “perfis similares”.

3 MODEL DRIVEN ARCHITECTURE

3.1 Cenário Geral

O progresso no desenvolvimento de software, nas últimas décadas, não demonstrou um grande avanço se comparado ao progresso na área de hardware. Constantes avanços na tecnologia acabam obrigando grandes partes do código a serem refeitas, desenvolvimentos anteriores acabam não podendo ser reutilizados, pois foram desenvolvidos sobre certa tecnologia.

Um outro problema que assola essa questão é a constante mudança de requisitos. Em um projeto de desenvolvimento de um sistema, as fases iniciais praticamente só produzem papel (especificações e diagramas). Ao ser codificado, esses papéis, na maioria das vezes, perdem valor pois, ao realizar uma manutenção, o desenvolvedor atualiza diretamente o código, sem alterar os modelos.

A Arquitetura Dirigida a Modelos (*Model Driven Architecture* - MDA) proposta pelo *Object Management Group*[4] (OMG) reconhece a importância dos modelos no processo, tornando-os o foco principal no desenvolvimento.

3.2 Desenvolvimento Dirigido por Modelos

O desenvolvimento de sistemas de computação é, até hoje, uma tarefa muito complexa, que ainda apresenta problemas como qualidade, prazos de entrega, orçamentos. Vários fatores contribuem para essa complexidade, como o crescimento do tamanho dos projetos, as diversas plataforma e ambientes computacionais disponíveis no mercado, a mudança frequente de re-

4 http://www.omg.org/

quisitos, entre outros. Além disso, os sistemas de informação precisam evoluir com o tempo, precisam sempre ser atualizados, independente do domínio da aplicação, do seu tamanho ou de sua complexidade (Pressman, 2006).

O uso de modelos no processo de desenvolvimento de um sistema é um dos caminhos possíveis para lidar com o tamanho e a complexidade existente na construção desses sistemas. Assim, os modelos passam a ser os elementos chaves no desenvolvimento, servindo não só como documentação do sistema, mas principalmente como uma ferramenta para a implementação. Através do uso de modelos no processo de desenvolvimento de sistemas de informação, acredita-se ser possível aumentar a produtividade e o reuso dos artefatos gerados (MACIEL, R.; SILVA, B.; ROSA, N , 2008).

Os modelos representam nada mais que a realidade simplificada, contendo apenas informações essenciais para o entendimento do objeto de estudo. Assim, é possível lidar com a complexidade relativa ao que está sendo modelado, a fim de permitir a melhor visualização, manipulação e raciocínio sobre a realidade estudada.

O Desenvolvimento Dirigido por Modelos (DDM) ou *Model Driven Development* (MDD) é um conjunto de abordagens de desenvolvimento de sistemas de informação baseados nos conceitos de modelagem, modelos e transformação de modelos. No DDM, os aspectos essenciais de um sistema são expressos na forma de modelos e as transformações desses modelos são o centro do desenvolvimento dos sistemas de informação. Além disso, os modelos são utilizados para entender o domínio do problema e o domínio da solução proposta; dessa forma, a relação entre os modelos torna possível a compreensão das implicações oriundas das mudanças que neles ocorrem (S. Beydeda, M. Book, 2005).

Sendo assim, ao possibilitar o desenvolvimento de um software num nível alto de abstração, construindo modelos próximos da realidade do problema, esse tipo de desenvolvimento apresenta vários benefícios como: aumento da produtividade, redução de tempo de construção do sistema, escalabilidade, etc.

3.3 Arquitetura Dirigida por Modelos

A Arquitetura Dirigida por Modelos (MDA) é um arcabouço conceitual para o desenvolvimento de softwares baseado nos princípios do Desenvolvimento Dirigido por Modelos. Lançada como especificação pelo OMG em 2001, a MDA define os modelos desse arcabouço, como eles devem ser usados e as relações entre esses modelos. Além disso, o MDA indica padrões e tecnologias a serem utilizados no processo de desenvolvimento de forma a tornar possível a realização do DDM.

A ideia principal do MDA é tentar separar ao máximo a especificação das funcionalidades do sistema e a especificação da implementação dessas funcionalidades numa determinada plataforma tecnológica (Miller & Mukerji, 2003).

Abordagens baseadas no MDA permitem o desenvolvimento de sistemas através da criação de modelos, os quais sofrem sucessivas transformações que são realizadas durante o processo de desenvolvimento, de acordo com a Figura 3. Assim, o processo de desenvolvimento de sistemas passa a ser direcionado pela atividade de modelagem e os modelos passam a ser o objetivo do processo (KLEPPE, A., WARMER, J., BAST, 2003).

Os principais artefatos produzidos durante o desenvolvimento de um sistema de informação usando MDA são os modelos e o código fonte. Esse processo de desenvolvimento decorre de uma sequência de transformações de modelo para modelo e modelo para código. A ideia é criar modelos abstratos que serão transformados em modelos baseados em alguma plataforma específica, e então serem transformados em código. Vale ressaltar que a partir de um mesmo modelo de negócio pode-se migrar para diferentes plataformas, sem a necessidade de remodelar o sistema. Todas essas etapas de transformações podem ser automatizadas, aumentando a velocidade do processo e também a consistência do código, além de facilitar sua manutenção.

O MDA divide o desenvolvimento em vários modelos. Nas seções que se seguem serão abordados o modelo PIM (do inglês *Plataform Independent Model*) e o modelo PSM (do inglês *Plataform Specific Model*), nessa mesma ordem.

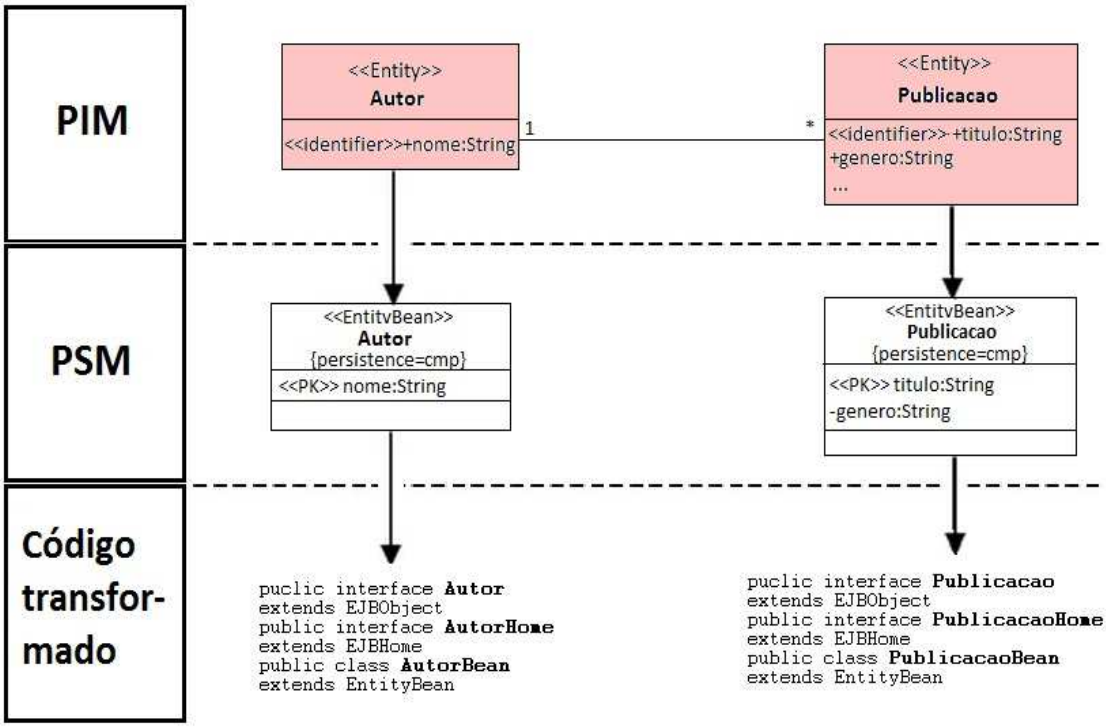

Figura 3: Representação do Fluxo de MDA

3.4 Modelo PIM

O Modelo Independente de Plataforma (PIM), como o próprio nome diz, possui uma visão computacional independente da plataforma de implementação do sistema. Dessa forma, o PIM mostra as funções e a estrutura do sistema, mas esconde informações específicas de uma tecnologia.

É a partir de um modelo PIM que se faz possível a geração de modelos para diferentes plataformas (PSM). Transformar os modelos UML em novos modelos com outras formas de representação, com especificações da tecnologia, é um diferencial do MDA em relação a outras abordagens de construção de *software*, pois o processo é feito utilizando padrões de projeto que necessariamente são códigos previamente testados e adotados pelo mercado aplicando as melhores práticas de programação.

3.5 Modelo PSM

Gerado o modelo PIM, o próximo passo é escolher uma plataforma específica sobre a qual será desenvolvido o sistema. Em seguida, combina-se o modelo anterior com as especificações dessa plataforma, gerando um modelo PSM (do inglês *Plataform Specific Model*), que nada mais é que uma instanciação de UML para uma determinada tecnologia. Essa instanciação se dá através de extensões UML, como marcas, metamodelos ou estereótipos. Essas marcações informam, na hora da transformação do PIM para o PSM, as intenções do desenvolvedor para o modelo, e podem ser de baixo ou alto nível, gerando modelos PSM mais ou menos específicos, respectivamente.

As marcações são específicas para cada plataforma, o que significa que a partir de um único PIM há possibilidade de gerar diversos PSMs. Isso se torna uma vantagem significativa pois a maioria dos sistemas utiliza mais de uma tecnologia. Isso significa também que, como o PSM é proveniente de um modelo orientado a objeto que descreve o modelo PIM, a qualquer momento ele pode ser convertido para diferentes plataformas, precisando apenas mudar os detalhes que especificam, como a aplicação deve fazer uso de um determinado tipo de plataforma para as marcações da tecnologia desejada.

Com o PSM finalizado, a geração de código passa a ser um passo relativamente simples, devido à similaridade do PSM com a tecnologia particular em uso. Desse modo, o código é o produto final da MDA e após sua geração, provavelmente será necessário alguns ajustes e complementações, mas, se comparado com um software produzido sem essa arquitetura, é um trabalho mais rápido devido a todo processo de desenvolvimento por modelos.

3.6 Benefícios do Uso de MDA

A característica chave do MDA que o destaca se deve ao fato de que a lógica de negócio fica separada da infraestrutura que a implementa. Com isso, há a possibilidade de desenvolvimento e entrega rápida de novas especificações que usam tecnologias novas, mas que são baseados em modelos de negócios já produzidos e testados.

Abaixo serão explicitados os principais benefícios do MDA:

- **Produtividade:** o tempo de desenvolvimento será melhor aproveitado, pois será gasto na produção de modelos de mais alto nível. Tarefas repetitivas podem ser implementadas nas transformações, poupando tempo e esforço que podem ser aproveitados em tarefas mais importantes. Como um modelo PIM pode gerar diversos modelos PSM, dependendo da plataforma de desenvolvimento escolhida, a transformação precisa ser definida uma única vez e pode ser aplicada no desenvolvimento de diversos sistemas.

- **Portabilidade:** a portabilidade é alcançada através da ênfase dada no desenvolvimento do PIM, que é independente de plataforma. Como um mesmo PIM pode ser transformado automaticamente em vários PSMs de diferentes plataformas, isso permite o aumento do reuso da aplicação e reduz custo e a complexidade do desenvolvimento de manutenção.

- **Interoperabilidade:** diferentes PSMs gerados a partir de um mesmo PIM podem conter ligações entre eles, que são chamadas de pontes, possibilitando a comunicação entre esses PSMs. Para isso, são utilizados rigorosos métodos para garantir que padrões baseados em tecnologias de múltipla implementação contenham sempre regras de negócio idênticas.

- **Reutilização**: Diversos autores, tais como (Krueger, 1992), (Griss, 1994), (Frakes, W. B. and Isoda, 1994), (Griss, ML., Jacobson, I., Jonsson, 1997), afirmam que a reutilização de artefatos de alto nível é muito mais vantajoso do que a reutilização de código-fonte. No desenvolvimento convencional, reutilizar um modelo inclui a transformação manual para reutilizar também o código a ele associado. No MDA, isto é mais facilmente resolvido, pois o código pode ser automaticamente regerado para o novo contexto.

3.7 AndroMDA

3.7.1 Introdução

AndroMDA[5] (AndroMDA, 2010) é um *framework* de código aberto que implementa transformações da MDA. Ele é dividido em núcleo e *plugins*, chamados de cartuchos, que são adicionados ao núcleo para realizar as transformações. Cartuchos são componentes que contém um mapeamento para uma determinada plataforma.

Essa ferramenta utiliza a UML como linguagem de modelagem, e utiliza apenas um modelo, usado como entrada para as transformações e geração direta de código. Esse modelo é chamado de PIM do sistema, e o AndroMDA exige que ele não pode conter dados de nenhuma plataforma específica nos seus elementos, apenas tipos de dados genéricos. O modelo utiliza diagramas de casos de uso, diagramas de classes e diagramas de atividade da UML.

Para garantir que os tipos de dados corretos sejam utilizados, o AndroMDA desenvolveu um perfil UML que deve ser incluído como parte do modelo. Esse perfil UML contém todos os tipos de dados, estereótipos e valores genéricos anotados que devem ser utilizados com a ferramenta e todos os seus cartuchos. O modelo deve ser construído utilizando os tipos de dados genéricos definidos no perfil UML e os elementos do modelo podem ser anotados com os estereótipos para que eles sejam transformados de maneira diferente

Para gerar o código a partir do PIM do sistema, o modelo deve ser usado como entrada para a ferramenta e os cartuchos necessários devem ser escolhidos. Os cartuchos contêm o mapeamento para as plataformas, então a escolha dos cartuchos deve ser baseada na plataforma destino, consequentemente, a escolha dos cartuchos define a plataforma do sistema. Mapeamentos no AndroMDA são feitos utilizando regras de mapeamento, as quais declaram os elementos do PIM do sistema esperados como entrada na regra e os elementos ou as *templates* alvos.

As regras de mapeamento do AndroMDA definem entradas baseadas em três possibilidades: tipos dos elementos, valores de propriedades ou estereótipos e saídas baseadas em *tem-*

---

5 http://www.andromda.org

plates. Os tipos e os estereótipos estão presentes no perfil UML. Os elementos do modelo de entrada são analisados pelas regras e, ao encontrar elementos cujos tipos, valores de propriedades ou estereótipos sejam os requeridos por uma determinada regra, ela é aplicada. A *template* definida na regra e os valores do elemento de entrada são passados para um gerador de *templates* (ou máquina de *templates*), que irá gerar o código seguindo o padrão descrito e utilizando os valores recebidos para preencher os valores necessários da *template*. O gerador de *templates* Velocity acompanha a ferramenta, mas pode ser substituído por outro gerador, de acordo com a necessidade. Os mapeamentos são descritos em XML e interpretados pela ferramenta utilizando a linguagem Java.

3.7.2 MDArte

O MDArte (MDARTE, 2010) surgiu com a necessidade de se incorporar mais funcionalidades a transformação de modelos seguindo a abordagem MDA. Assim, ele pode ser definido como uma evolução do *framework* AndroMDA, voltado para o desenvolvimento de softwares para o Governo Brasileiro, sendo disponibilizado como *software* de domínio público.

O *framework* MDArte compreende um conjunto de cartuchos AndroMDA com diversas soluções de projeto, entre eles: Hibernate, EJB, Bpm4Struts, jBPM e Java. Possibilitando ainda a adição de novos cartuchos de acordo com a necessidade dos projetos. Com o intuito de continuar a melhorar o MDArte, com adição e evolução de cartuchos, sua comunidade foi disponibilizada no Portal do Software Público Brasileiro, permitindo a comunicação das pessoas que já o utilizam e daquelas que se interessam em começar a utilizá-lo.

Ao contrário do Andromda, o MDArte possui um arcabouço de segurança orientado a modelos. O arcabouço contribui para especificação e implementação dos requisitos de segurança durante o processo de desenvolvimento de sistemas, simplificando o desenvolvimento. O desacoplamento entre a implementação do negócio propriamente dito e a solução de segurança facilita a evolução do sistema gerado. Além disso, ele permite que a definição de permissões seja feita de forma dinâmica, aumentando a flexibilidade do sistema, já que informações tipicamente relacionadas a implantação como, por exemplo, perfis de usuários, não precisam ser modeladas em tempo de projeto (Cunha, 2007).

4 TIRINHAZ

4.1 Proposta

Devido à grande quantidade de informações e a crescente facilidade do acesso as mesmas através da internet, as pessoas se sentem inseguras nas escolhas que fazem, seja comprando um produto, alugando um filme ou até simplesmente escolhendo um vinho. Quando ocorre essa insegurança, elas geralmente recorrem a uma recomendação de alguém que já tenha experimentado aquele produto.

Como a cada dia que passa, tornam-se mais populares as lojas *online* e as redes sociais, os consumidores estão deixando de sair de casa para consumir um certo produto através da internet. Somado a grande quantidade de opções e a preferência do uso das lojas *online*, existe a necessidade de se ter uma forma de recomendar produtos aos consumidores, de maneira que eles não precisem gastar muito tempo procurando nas inúmeras opções disponíveis o que realmente querem.

Nesse cenário surgiram os Sistemas de Recomendação. Sistemas nos quais os usuários fornecem suas avaliações sobre determinados produtos e o sistema as agrega direcionando-as para indivíduos que são considerados similares a quem as recomendou. Um dos grandes desafios desses sistemas é justamente achar a correta similaridade entre indivíduos que fornecem a recomendação e aqueles que a recebem. Uma dificuldade desse tipo de sistema é a capacidade de captura de informação para que sejam testados e validados os algoritmos.

O TirinhaZ foi desenvolvido sob esse contexto. É um sistema multiusuário de recomendação de tirinhas, que visa criar um ambiente interativo, onde o usuário lerá uma tirinha, atribuirá uma nota e receberá uma recomendação de outra tirinha, esta baseada no seu perfil (ca-

racterísticas pessoais e notas atribuídas). Este sistema possui integração com a rede social *Facebook*[6], onde o usuário poderá compartilhar e comentar uma tirinha no perfil da rede social. Há também a possibilidade do usuário adicionar tirinhas no sistema, compartilhando-as com outros usuários.

4.2 Arquitetura e Modelagem

Devido a necessidade de rápida propagação de informação que o sistema TirinhaZ precisaria ter para obter um desempenho aceitável de acordo com o objetivo do trabalho, optou-se por utilizar uma arquitetura Web, dando assim capacidade do sistema de se auto promover através de indicações em redes sociais por meio de *plugins*, e acesso facilitado sem a necessidade de instalação de nenhum software. Assim, o único requisito para se utilizar o sistema é possuir um navegador corretamente conectado a internet. Como resultado, pode-se não só alcançar uma quantidade elevada de usuários como também de informações geradas pelos mesmos, resultando em um funcionamento do sistema mais próximo ao desejado.

A ferramenta foi implementada em Java utilizando o banco de dados MySql (Oracle Corporation 2010b) e o servidor de aplicação JBoss (Red Hat 2010b). O sistema foi desenvolvido utilizando o *framework* MDArte, o qual foi escolhido devido a diversos fatores, dentre eles, a obrigação de realizar a separação entre a lógica da aplicação e a interface com o usuário, através da utilização do padrão MVC e do padrão baseado em camadas, comum nos sistemas Web, facilitando o desenvolvimento, a manutenção do código e, até mesmo, reutilização de trechos. Outro benefício é a adoção de padrões no projeto que utiliza tal *framework*, obrigando, assim, o desenvolvimento do sistema - por parte dos programadores - segundo os padrões mais utilizados. Por fim, outro fator muito importante é a criação de modelos que reflitam o sistema como um todo, através de Diagramas de Classe e de Casos de Uso, ambos em UML, o MDArte gera boa parte do escopo do projeto, resultando em uma melhor documentação e maior facilidade de uso do mesmo.

---

6 http://www.facebook.com

A arquitetura do sistema é separada em 4 camadas, de acordo com a Figura 4:

1. **Apresentação:** contém todas as classes e os artefatos (páginas JSP) relacionados à interação visual entre o usuário e a aplicação;

2. **Controle:** faz a comunicação da apresentação com os serviços, recebendo dados desta, tratando-os e chamando os serviços que convém;

3. **Serviço:** são classes de negócios que representam a modelagem do domínio por meio de regras e validações;

4. **Domínio:** são classes responsáveis por: cálculos baseados em dados digitados e informações armazenadas, validação de informações vindas da camada de apresentação e qual fonte de dados será acionada.

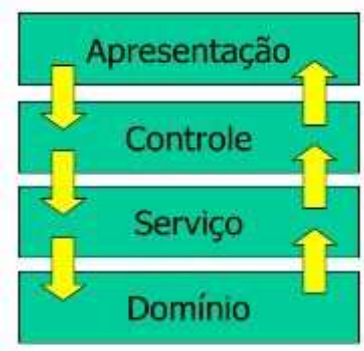

Figura 4: Arquitetura do MDArte (MDArte, 2012)

Além disso, cada camada foi também dividida em módulos, visando uma melhor separação dos elementos por funcionalidade, possibilitando assim um melhor reaproveitamento de determinadas partes do sistema. A camada de Domínio do sistema, por exemplo, foi dividida nos módulos “Core” e “Tirinha”. Abaixo são apresentados os diagramas de cada módulo e as respectivas descrições. Na Figura 5 é exibido o diagrama de classes do módulo “Core”, e na Figura 6, o diagrama do módulo “Tirinha”.

O módulo “Core” do Modelo possui 5 classes: “Usuário”, “Imagem”, ”Estado” , “Nota” e “Produto”. A classe “Usuário” possui os atributos dos usuários do sistema, como login, *e-mail*, nome, data de nascimento, *status*, citação e descrição pessoal. E ela se associa com as classes “Imagem”, que armazena a foto do usuário, “Estado”, que armazena o estado que resi-

de o usuário, e a classe “Nota”, que armazena o valor e a data de uma nota dada pelo usuário de um determinado produto. E finalmente a classe “Produto”, que é uma classe que armazena dados de um produto no sistema. Esta classe em especial é genérica, foi feita com o intuito de não especializar o modelo, possibilitando que outros tipos de produto, e não apenas tirinhas, possam utilizar a mesma estrutura descrita acima, tornando o modelo escalável.

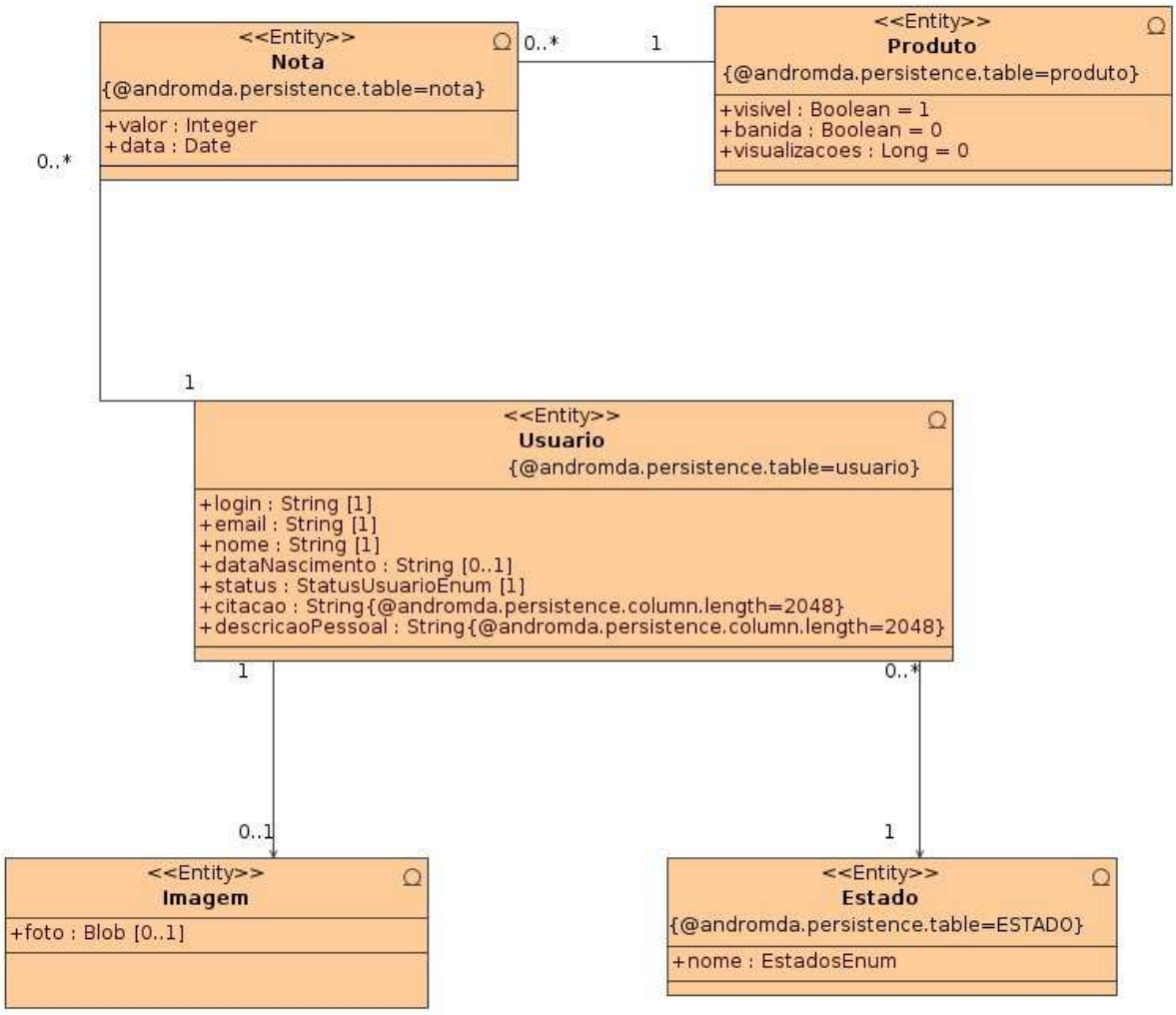

Figura 5: Modelo de Classes de Módulo “Core”

O módulo “Tirinha” do Domínio é composto pelas classes “Tirinha”, “Imagem” e “AutorTirinha” e pela classe externa “Produto” pertencente ao módulo “Core”. A classe “Tirinha” é uma especialização de “Produto” e armazena a descrição das tirinhas. Ela se associa à classe “Imagem”, responsável por armazenar a tirinha propriamente dita e a classe “AutorTirinha”, que guarda as informações do nome e do site do autor da tirinha.

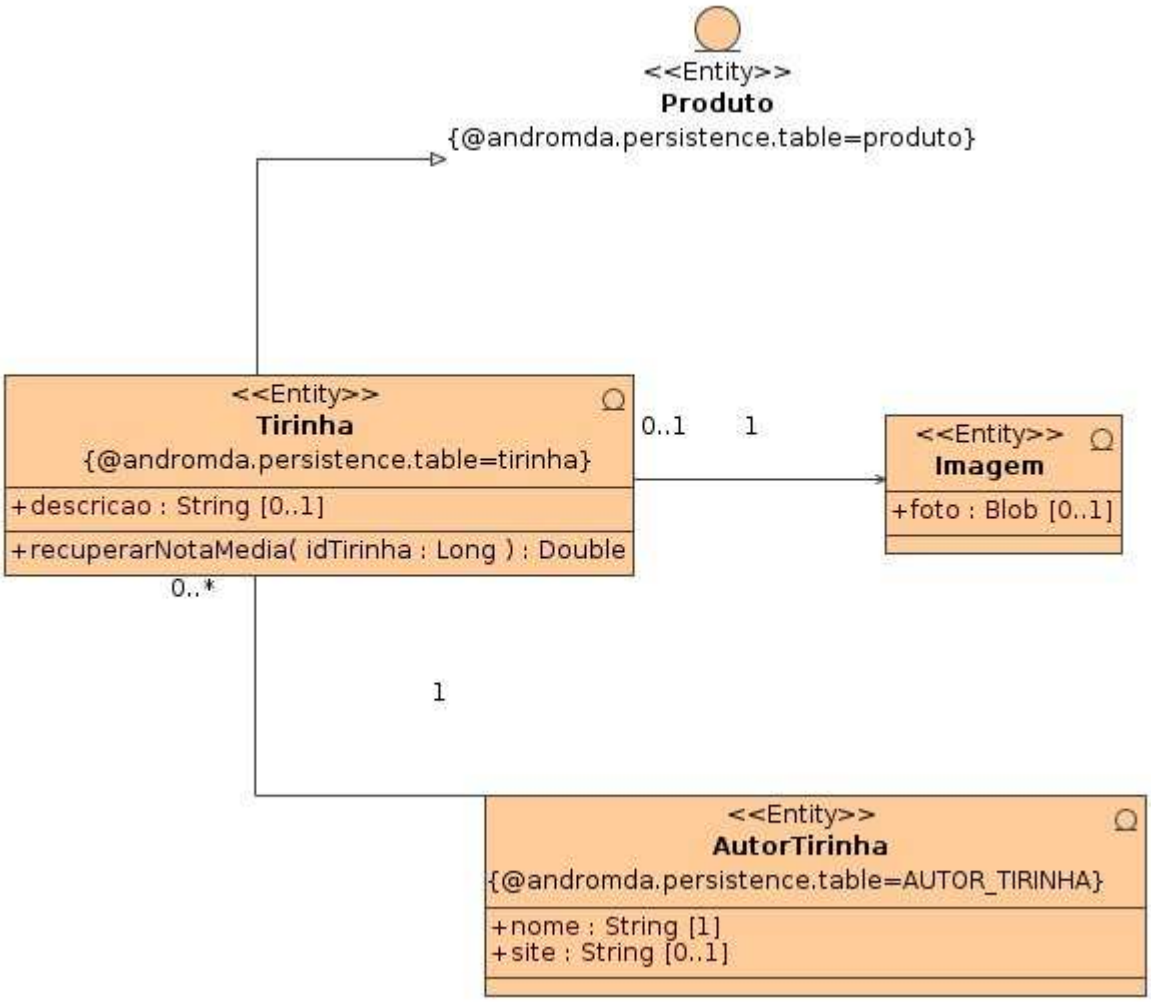

Figura 6: Modelo de Classes do Módulo Tirinha

A camada de Serviço foi dividida em 3 módulos: “GeralHandler” (serviços relacionados ao usuário), “ProdutoHandler” (serviços relacionados ao produto) e “TirinhaHandler” (serviços relacionados a tirinha). As regras de negócio da aplicação são especificadas nesses serviços.

A camada de “Apresentação” também foi dividida em módulos: “GeralControle” e o “TirinhasControle”. Esses módulos serão especificados no item 4.3.1, onde serão descritos os casos de uso.

4.3 Descrição do Sistema

Nesta seção o TirinhaZ será detalhado, serão descritos os requisitos do sistema através de casos de uso, qual algoritmo de recomendação foi implementado e seu funcionamento.

4.3.1 Casos de Usos

No TirinhaZ, existirão apenas dois atores: o visitante e o usuário logado. Qualquer pessoa poderá acessar o sistema. No primeiro acesso, esta pessoa estará com o perfil de visitante. Após cadastrar-se e, consequentemente, logar-se no sistema, ela terá o perfil de Usuário Logado, no qual poderá avaliar as tirinhas, adicionar tirinhas e receber recomendações significativas.

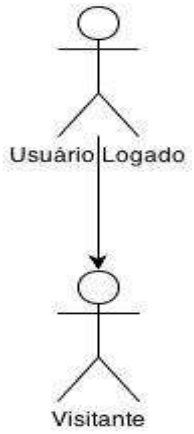

Figura 7:
Hierarquia Autores

O visitante, por tanto, poderá efetuar o login, cadastrar-se e visualizar uma tirinha pública, como representado na Figura 8. Em contrapartida, o Usuário Logado terá acesso a diversas ações, que estão representadas nos diagramas de Caso de Uso dos módulos “UsuárioControle” (Figura 9) e “TirinhasControle” (figura 9):

- **Efetuar Login:** caso de uso responsável por validar o registro do usuário no sistema;
- **Cadastrar Login:** caso de uso onde o visitante se cadastra no sistema, inserindo informações como nome, e-mail, estado, etc;

- **Visualizar Tirinha Pública:** caso de uso no qual o visitante pode visualizar uma tirinha, mas não podendo avaliá-la;

- **Consulta Usuário:** neste caso de uso o usuário pode consultar outros usuários cadastrados no sistema. Essa busca pode ser através do nome, e-mail ou estado de origem. Após preenchido pelo menos um desses campos, é retornada uma lista de usuários que correspondem ao filtro escolhido. Para acessar o perfil do usuário consultado, basta clicar sobre a foto deste, sendo redirecionado para o caso de uso Detalhar Usuário;

- **Detalhar Usuário:** exibe as informações do perfil de um usuário cadastrado, como nome, e-mail, foto, data_nascimento, estado de origem, etc. Caso esteja sendo exibido o perfil do usuário que está logado, ele poderá ir para o caso de uso Mantem Usuário, através do botão “Editar Perfil”;

- **Mantem Usuário:** esse caso de uso tem mais de uma funcionalidade, como cadastrar um novo usuário ou editar informações de um usuário previamente cadastrado. A diferença é que no momento da inclusão de um novo usuário, o campo login aparece para ser preenchido e, obviamente, os campos restantes não estão carregados com qualquer valor;

- **Consulta Tirinha:** nesse caso de uso o usuário poderá buscar tirinhas através do nome do grupo ou do site do grupo. Após clicar em “Consultar”, o usuário receberá uma lista de tirinhas correspondentes aos filtros da busca. Caso queira detalhar a tirinha, basta clicar sobre o nome, sendo assim redirecionado para o caso de uso Detalhar Tirinha;

- **Detalhar Tirinha:** exibe as informações de uma tirinha, como nome do autor, site, descrição, etc. Caso o usuário logado seja o dono da tirinha, ele poderá editá-la clicando no botão “Editar Tirinha”, e será encaminhado ao caso de uso Mantem Tirinha;

- **Mantem Tirinha:** caso de uso responsável tanto por editar informações de uma tirinha (nome, autor, etc) quanto para salvar uma nova tirinha no sistema;

- **Recomendar Tirinha:** neste caso de uso o usuário logado atribuirá uma nota de 1 a 5 (representada por estrelas) à tirinha que está sendo visualizada;

- **Curtir Tirinha *Facebook*:** caso de uso em que o usuário logado "curte" uma tirinha, o que aparecerá no seu perfil do *Facebook*;

- **Comentar Tirinha *Facebook*:** caso de uso no qual o usuário logado escreve um comentário sobre a tirinha diretamente no perfil da rede social *Facebook*;

- **Compartilhar Tirinha *Facebook*:** caso de uso responsável por compartilhar uma tirinha no *Facebook*. Ou seja, a tirinha fica no seu perfil para ser vista por amigos da rede social em questão.

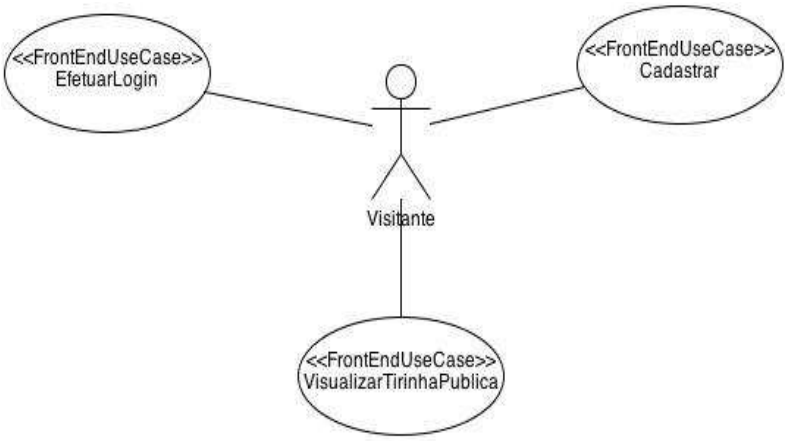

Figura 8: Diagrama de Caso de Uso do Visitante

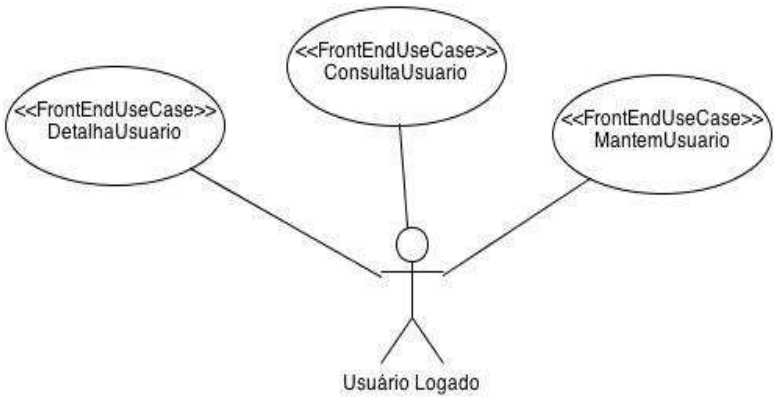

Figura 9: Diagrama de Casos de Uso do módulo "UsuárioControle" do Usuário Logado.

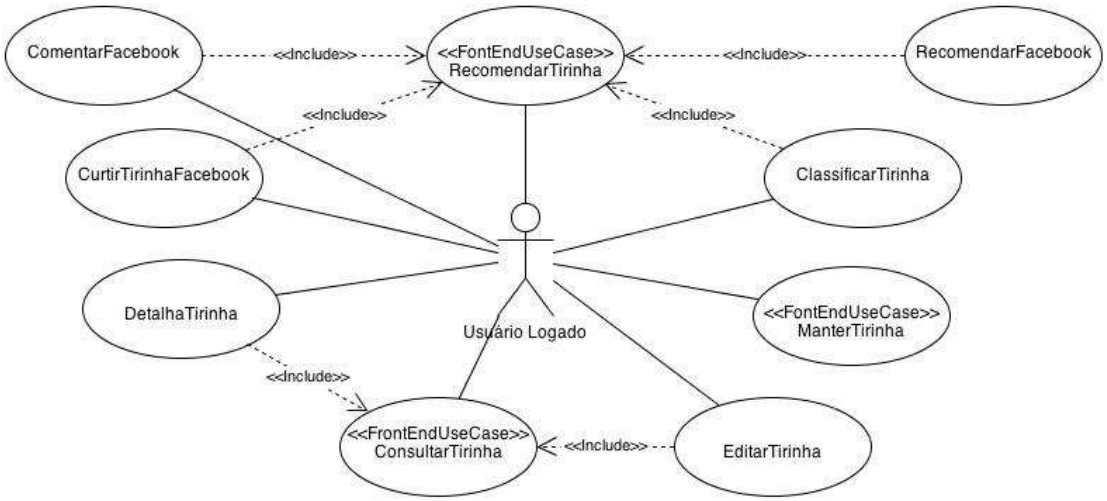

Figura 10: Diagrama de Casos de Uso módulo “TirinhasControle” do Usuário Logado.

4.3.2 Recomendação

O TirinhaZ foi desenvolvido objetivando a recomendação de tirinhas. Por essas se tratarem de um produto que tem característica volátil (podem ser mudadas a qualquer momento por um usuário), o tipo de recomendação escolhida foi a filtragem colaborativa. Além disso, evita-se o problema da superespecialização e da elasticidade *versus* plasticidade, descritos na sessão 2.2.1.

O método escolhido de filtragem colaborativa foi o baseado em memória (sessão 2.2.2.2), onde dois usuários são tratados como vetor e a similaridade é medida através do cálculo do cosseno do ângulo entre esses vetores.

A modelagem da recomendação foi pensada e desenvolvida de forma genérica, a fim de que possam ser implementados diferentes algoritmos de recomendação sem ter a necessidade de modificações no modelo. Para tal foi criada uma classe abstrata Recomendador, que define um *template* para os métodos implementados pelos algoritmos de recomendação utilizados.

A partir dessa classe abstrata e de uma estrutura de herança, os métodos abstratos do “Recomendador” são implementados por subclasses concretas. O modelo dessa estrutura é representada na Figura 11, no qual pode ser observado o relacionamento da superclasse “Recomendador” e as subclasses “CollaborativeFiltering” e “Aleatorio”.

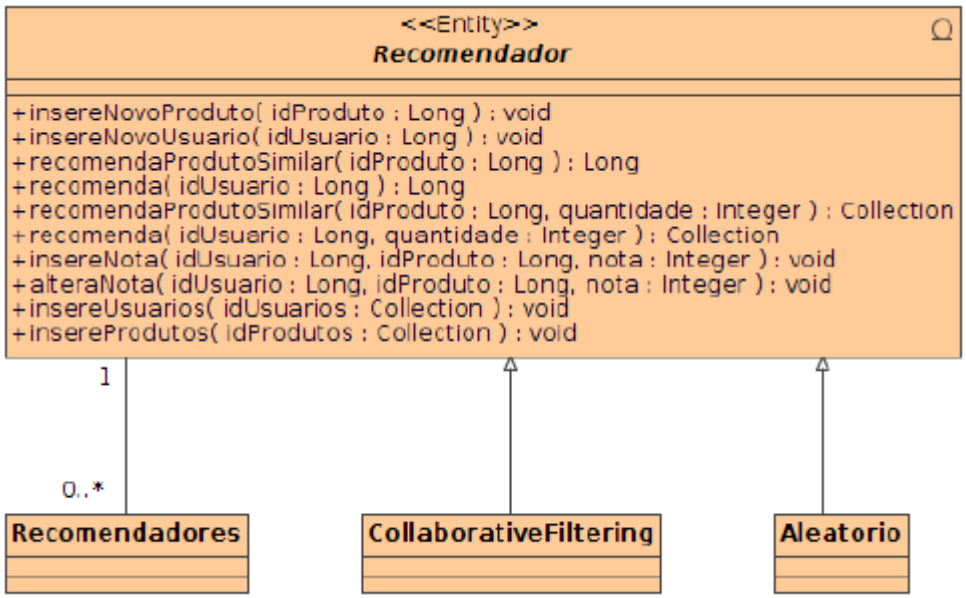

Figura 11: Modelagem da Recomendação

As subclasses acima herdam as assinaturas dos métodos da superclasse e os implementam independentemente, utilizando as definições específicas do seu comportamento. Essa forma que o processo de recomendação foi modelado permite a inclusão de novos algoritmos, para isso é preciso apenas que estes novos algoritimos sigam o template proposto pela classe abstrata “Recomendador”.

A classe “CollaborativeFiltering” implementa o algoritmo de recomendação filtragem colaborativa, enquanto que a classe “Aleatorio” implementa o algoritmo aleatório. Esse último foi desenvolvido para mostrar o funcionamento genérico do sistema, não se baseando em qualquer técnica de recomendação.

A classe “Recomendadores” é um *singleton* que foi criado com o objetivo de garantir a existência de apenas uma instância da classe “Recomendador”, ou seja, definir qual algoritmo de recomendação será utilizado. Essa classe define os parâmetros iniciais de cada subclasse de “Recomendador”, além de realizar a sua instanciação. A grande vantagem da utilização do padrão de projeto *singleton* é o encapsulamento dos dados e do comportamento, além da garantia de consistência do sistema, para casos em que é necessário restringir a instanciação de determinada classe.

5 IMPLEMENTAÇÃO

Neste capítulo serão encontrados detalhes da implementação do TirinhaZ, desde códigos a telas desenvolvidas. Na última seção serão abordados problemas e soluções que surgiram ao longo da implementação.

5.1 Controle de Acesso

O MDArte, discutido no capítulo 3, possui um *framework* de segurança orientado a objetos. O uso desse framework facilita o desenvolvimento do sistema na medida que desacopla a implementação do negócio propriamente dita da definição dinâmica das permissões (CUNHA, 2007). Essa vantagem proporciona um significativo aumento da flexibilidade do sistema, como por exemplo, a não necessidade de informações do tipo perfis do usuário serem modeladas em tempo de projeto.

No desenvolvimento do Controle de Acesso do TirinhaZ, foi utilizado o *framework* ArchiMDAs (CUNHA, 2007) em uma versão mais simplificada, que está disponível no Portal do Software Público Brasileiro. Essa versão permite o controle das ações do operador do sistema, de acordo com o perfil que lhe foi atribuído, além de implementar o gerenciamento de serviços e dos perfis que podem acessá-los.

5.2 TirinhaZ

O TirinhaZ foi desenvolvido utilizando o framework MDArte e a arquitetura em camadas descrita na seção 4.2. A fim de preservar uma boa organização, a estrutura dos diretórios do

TirinhaZ foi montada de acordo com a Figura 12. A pasta “recsys” representa a pasta principal do módulo “Tirinha”. Essa contém as subpastas “cd”, “cs”, “eo” e “vo”.

A Camada de Domínio da aplicação fica no diretório “cd”, onde contém as entidades descritas na seção 4.2 como as classes modeladas. Essas entidades estão dispostas em duas pastas: “nucleo” e “tirinha”. Na primeira ficam as classes que podem ser usadas para um outro produto, sem ser a tirinha. E na segunda ficam as classes exclusivas do módulo “Tirinha” (especialização de “Produto”, seção 4.2), o que reforça o caráter genérico da aplicação.

O diretório “cs” contém os elementos da Camada de Serviço e as classes modeladas como serviço (“GeralHandler”, “ProdutoHandler”, “TirinhaHandler” e “PublicoHandler”). Além disso, os elementos da Camada de Serviço possuem dependência em relação aos elementos da Camada de Domínio. A pasta “vo” reúne os *Value Objects* (VO), cuja função principal é transferir dados entre as camadas. E a pasta “eo” contém os Enumerados, que armazenam constantes pré-definidas.

Figura 12: Estrutura do Modelo do TirinhaZ

As camadas de Apresentação e Controle não estão representadas nessa estrutura, entretanto abrigam os Casos de Uso e os respectivos Diagramas de Atividades, definido em um módulo do sistema, conforme a Figura 13. Esses diagramas descrevem o fluxo de dados de um determinado caso de uso.

Figura 13: Estrutura web do TirinhaZ.

Partindo para a análise das telas, ao acessar o sistema, primeiramente o usuário visualiza a tela de login representada na Figura 14. Nessa tela há duas opções: efetuar o *login* ou cadastrar-se. Ao clicar no botão “Cadastrar” o usuário é encaminhado para a tela indicada na Figura 15. Ainda na mesma tela, o usuário irá preencher seus dados como login, e-mail, nome, etc. Todos esse dados passam por uma validação instantânea, como a verificação da já existência de um mesmo login, por exemplo, a fim de auxiliar esse usuário a se cadastrar.

Em contrapartida, ao preencher os campos da tela de login e clicar no botão “Login”, ao ser autenticado, o usuário será redirecionado para a tela inicial do Tirinhaz, representada na Figura 16. Nela ele encontrará uma lista das tirinhas mais visualizadas, outra com as melhores avaliadas e, por fim, as recomendadas.

Figura 14: Tela de login

Figura 15: Tela de cadastro de usuário

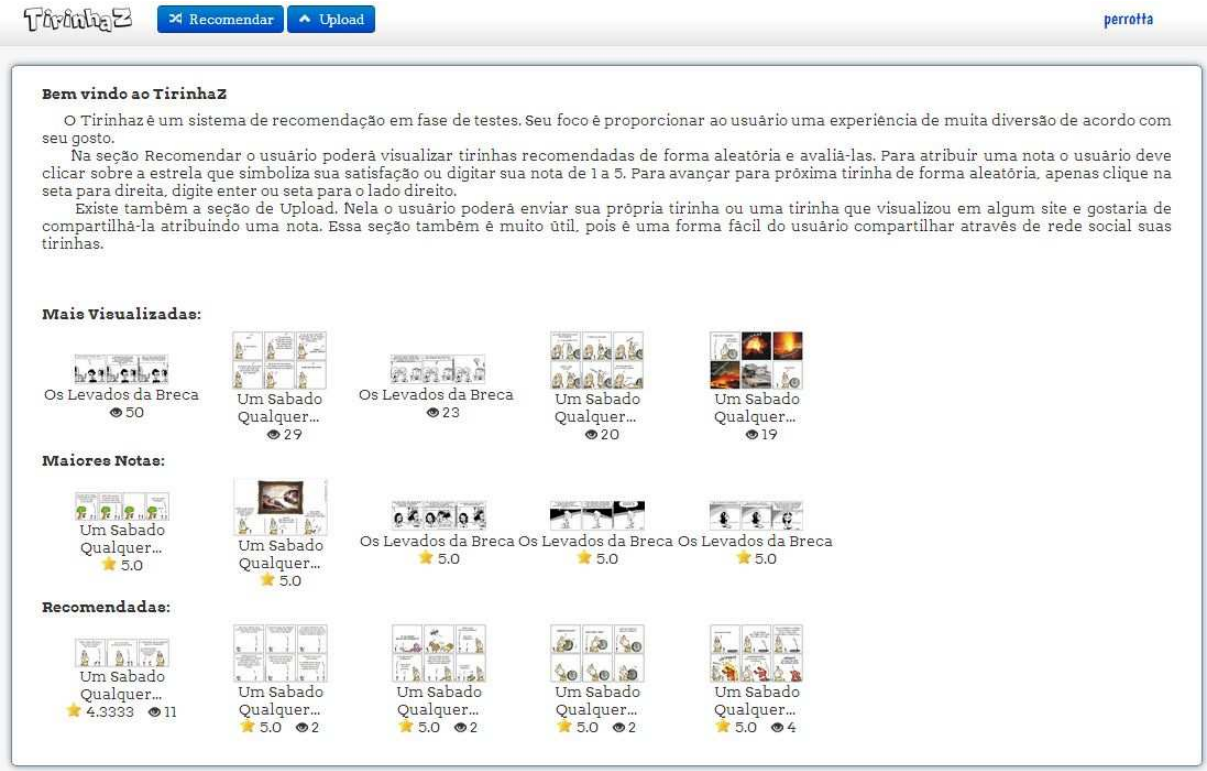

Figura 16: Tela inicial

Na parte superior da tela inicial, encontram-se os botões “Recomendar” e “Upload”. Ao clicar no primeiro botão, entra-se no principal caso de uso do sistema, chamado “Recomendar Tirinha”, que é representado pelo Diagrama de Atividades mostrado na Figura 17, e então o usuário é levado a uma tela semelhante àquela exibida na Figura 18, porém com a ausência do histograma “Avaliação Geral” e do preenchimento das estrelas, ambos vistos mais a frente.

Nessa tela do sistema pode-se observar a direita da tela: uma lista de tirinhas em miniatura recomendadas pelo TirinhaZ, abaixo um *link* dando a opção do usuário acessar a página do sistema no *Facebook* e em seguida o *plugin facepile* que mostra quantas pessoas já curtiram a página na respectiva rede social, assim como fotos de alguns desses usuários.

Ao centro há uma tirinha principal acompanhada de cinco estrelas que representam a avaliação que o usuário poderá atribuir à mesma e um pequeno quadro a esquerda com os *plugins* “curtir” e “enviar” da rede social *Facebook.*

Abaixo do quadro central, existe a área de comentários. Para construí-la também foi utilizado um *plugin* para integração com a rede social que permite o usuário postar comentários

acerca da tirinha principal, podendo escolher a opção de também publicá-los em seu perfil. Ao decidir por marcar a opção “Publicar no *Facebook”,* aparecerá para o usuário algo semelhante a Figura 19 em sua *timeline*.

O comportamento do quadro de comentários difere de acordo com a quantidade enviada. A partir de 5 comentários, a área de escrita é escondida e surge a o botão “comentar” que ao clicá-lo, faz com que a área ressurja na tela. Após alcançar 10 comentários, os mais antigos vão sendo escondidos e o sistema passa a mostrar os 10 mais recentes seguindo uma ordenação “social” (que levará em conta gostos e comentários de outros usuários) ou outra escolhida pelo usuário (cronológica ou cronológica inversa) que pode ser selecionada clicando na seta ao lado da quantidade total. Ao alcançar tal quantidade o sistema dá a possibilidade do usuário clicar em “ver mais”, no canto inferior, que poderá mostrar os omitidos.

Outro aspecto que merece destaque, é o fato de toda interação envolvendo marcação de usuário (digitar um nome de um outro usuário do *facebook* que seja “amigo” do usuário autenticado) funciona normalmente. Isso é muito interessante, pois possibilita referenciar pessoas e as mesmas serem notificadas sobre tal comentário em seus perfis, aumentando assim a divulgação do sistema.

Portanto, nessa tela o usuário tem a liberdade de ler a tirinha, decidir avaliá-la, passar para uma próxima aleatória ou clicar em uma recomendada entre as indicadas pelo sistema. Assim como a possibilidade de "enviar" a outra pessoa, "curtir" ou apenas deixar comentários para tirinha principal.

Outra opção é clicar em uma das tirinhas recomendadas ou próxima aleatória(indicada pela seta apontando para direita) e ser transferido para uma tela semelhante, cuja tirinha principal será a nova tirinha selecionada ou uma escolhida de forma randômica, dependendo da escolha.

Então supondo que o usuário atribua uma classificação para essa tirinha, uma tela igual a da figura Figura 18 será exibida. Nela um gráfico que expressa as avaliações anteriores da tirinha é exibido e a classificação dada pelo usuário atual é mostrada pelas estrelas preenchidas com a cor amarela.

Esse gráfico citado trata-se de uma Avaliação Geral, é um gráfico de barras horizontais, onde o eixo horizontal representa o número de notas e o eixo vertical representa a nota dada: “Ruim” para nota 1 (uma estrela), “Fraca” para nota 2 (duas estrelas) e assim sucessivamente. Ele é exibido somente após o usuário logado avaliar a tirinha exibida, a fim de evitar que ele seja influenciado por avaliações de terceiros.

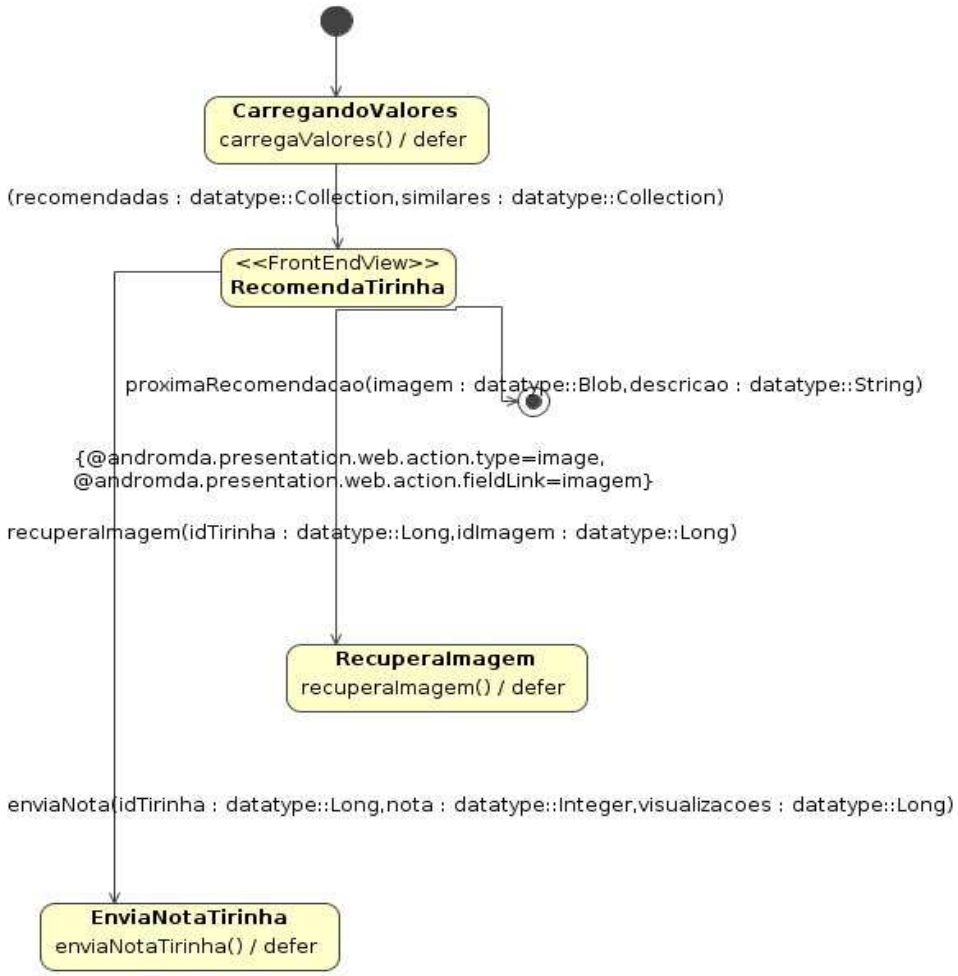

Figura 17: Diagrama de Atividades do Caso de Uso "Recomendar Tirinha".

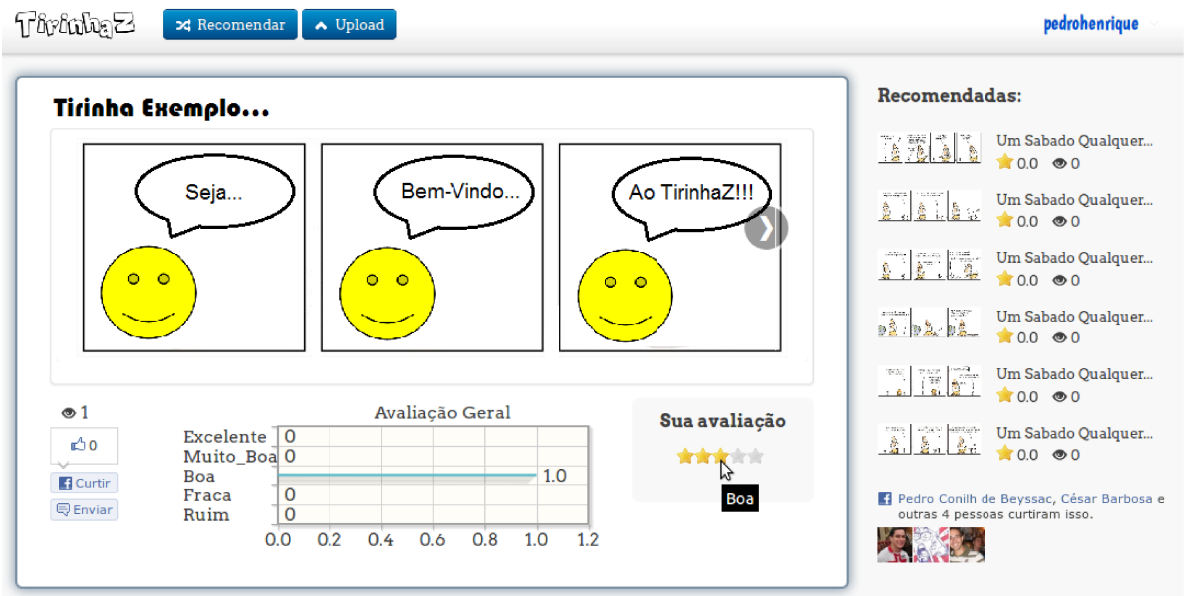

Figura 18: Tela Recomendar

Figura 19: Comentário publicado na *timeline* do usuário do *Facebook*

Voltando a tela inicial, outra opção que o sistema possui é a possibilidade da publicação de uma tirinha pelo usuário. Para isso, basta que ele clique no botão “Upload” e assim seja encaminhado a tela de *upload* representada pela Figura 20.

Essa funcionalidade do sistema é de grande importância pois permite que o usuário compartilhe de forma simples tirinhas vistas em outros sítios ou tirinhas de sua própria autoria. Para isso, basta preencher os campos da tela em questão, como nome e sítio do grupo, descrição, e clicar em “Salvar Tirinha”. Com isso o usuário será redirecionado para a tela “recomen-

dar tirinha" (Figura 18), onde estará sendo exibida a tirinha submetida como principal para ser avaliada.

Figura 20: Tela de *Upload*

De forma alternativa existe um fluxo mostrado no diagrama da Figura 21 no qual o usuário acessa o sistema pelo caso de uso "ExibirTirinhaPublica". Isso ocorre quando alguém recebe uma recomendação de uma tirinha via *Facebook*, acessando o TirinhaZ através desse link. Nesse ocasião, há um ponto de decisão no sistema que automaticamente redireciona o usuário para tela correta de acordo com sua situação, se autenticado ou não.

Caso encontra-se autenticado, o usuário acessará a tela de recomendação como descrita anteriormente. Por outro lado, se não houver realizado a autenticação, o usuário é redirecionado para a tela "Exibir Tirinha Pública" (Figura 22), na qual é exibida apenas a tirinha em questão , quantas visualizações obteve e as opções de "curtir" e "enviar" do *Facebook.*

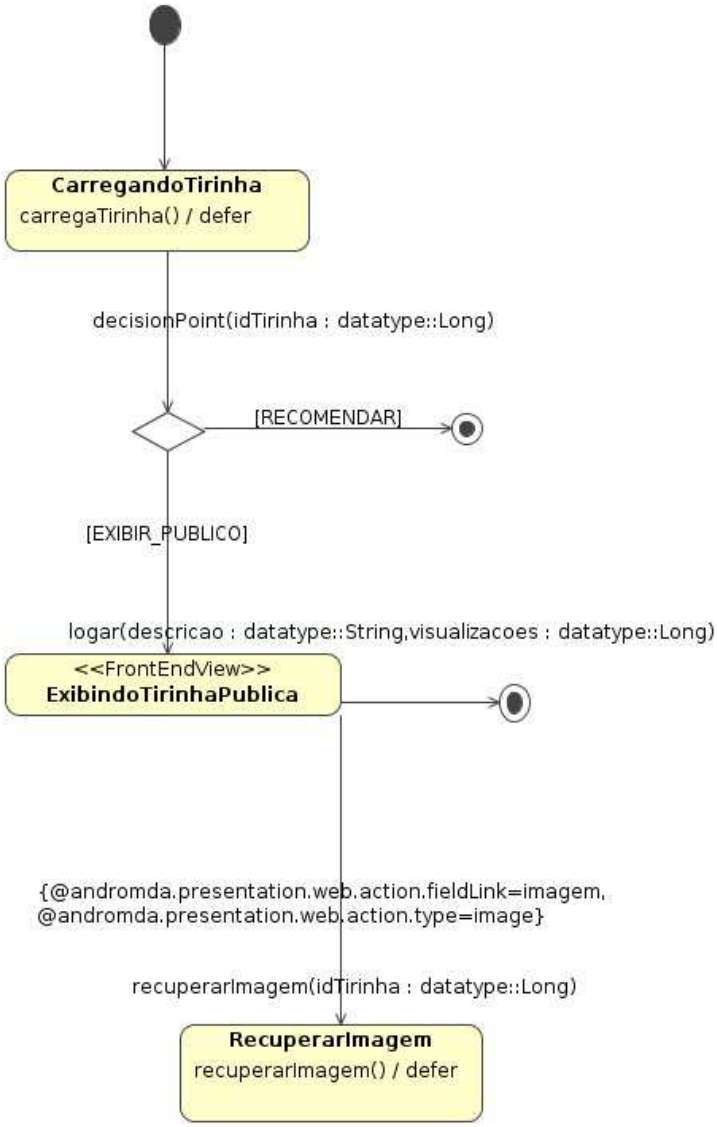

Figura 21: Diagrama de Atividades do Caso de Uso “ExibirTirinhaPublica”

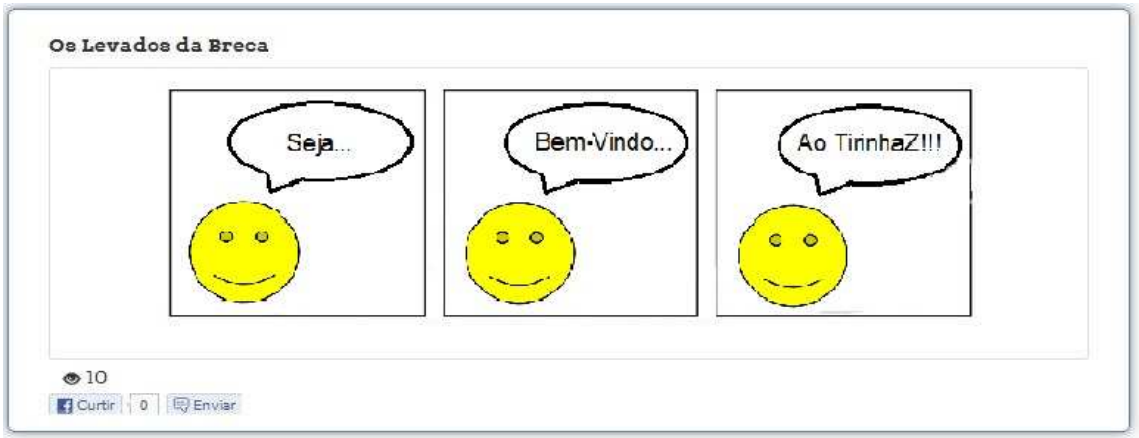

Figura 22: Tela Exibir Tirinha Pública

Quanto a recomendação, foi criada uma classe abstrata “Recomendador”, explicitada na Figura 23. Essa classe define um *template* para os métodos implementados pelo algoritmos de recomendação. Para atender a proposta deste trabalho, os algoritmos escolhidos foram “CollaborativeFiltering” e “Aleatório”.

Os algoritmos de recomendação utilizados devem implementar esses métodos propostos. O algoritmo “Aleatório” foi desenvolvido para gerar recomendações de forma não determinística, utilizando a classe “Random” da linguagem de programação Java, como mostrado na Figura 24.

O algoritmo “CollaborativeFiltering”, por sua vez, necessita da matriz de avaliações de itens por usuários previamente instanciada, pois será utilizada no cálculo das predições dos itens. Esse algoritmo é exibido na Figura 25.

```
public abstract class Recomendador {
     public abstract void insereNovoProduto(Long idProduto);
     public abstract void insereNovoUsuario(Long idUsuario);

     public abstract Long recomendaProdutoSimilar(Long idProduto);
     public abstract Long recomenda(Long idUsuario);
     public abstract Collection<Long> recomendaProdutoSimilar(Long idProduto,
     Integer quantidade);
     public abstract Collection<Long> recomenda(Long idUsuario, Integer quantidade);

     public abstract void insereNota(Long idUsuario, Long idProduto, Integer nota);
     public abstract void alteraNota(Long idUsuario, Long idProduto, Integer nota);
     public abstract void insereUsuarios(Collection<Long> idUsuarios);
     public abstract void insereProdutos(Collection<Long> idProdutos);
}
```

Figura 23: Classe abstrata Recomendador

```
public Collection<Long> recomenda(Long idUsuario, Integer quantidade) {
     Collection<Long> produtosRecomendados = new ArrayList<Long>();
     Long[] produtosArray = (Long[]) produtos.toArray(new
     Long[produtos.size()]);

     for (int i = 0; i < quantidade; i++){
          int j = (new Random()).nextInt(produtos.size());
          if(!produtosRecomendados.contains((produtosArray[j]))){
               produtosRecomendados.add(produtosArray[j]);
          }
          else{
               i--;
          }
     }
     return produtosRecomendados;
}
```

Figura 24: Classe "Recomenda" - implementa o algoritmo "Aleatório"

```
public CollaborativeFilteringUser(Collection idsUsuarios, Collection
idsProdutos) {
     this.qtdUsuarios = idsUsuarios.size();
     this.qtdProdutos = idsProdutos.size();
     this.produtosMap = convertToMap(idsProdutos);
     this.usuariosMap = convertToMap(idsUsuarios);
     this.mUsersItems = new double[this.qtdUsuarios] [this.qtdProdutos];

     /* Preeche matriz com notas de cada produto dada por usuario.*/
     for(int i = 0; i < idsUsuarios.size(); i++) {

          for(int j = 0; j < idsProdutos.size(); j++){
               NotaVO notaVO = new NotaVO();
               notaVO.setIdUsuario(getKeyByValue(this.usuariosMap, i));
               notaVO.setIdProduto(getKeyByValue(this.produtosMap, j));
               NotaDAO dao = new NotaDAOImpl();
               Collection notasProdutos = null;

               try {
                    notasProdutos = dao.filter(notaVO, null);
               }
               catch (Exception exception) {
                    exception.printStackTrace();
               }

               if (notasProdutos != null && !notasProdutos.isEmpty()) {
                     Nota nota = (Nota) notasProdutos.iterator().next();
                     this.mUsersItems[i][j] = nota.getValor();
               }
               else {
                     this.mUsersItems[i][j] = 0;
               }
          }
     }
     this.userBasedCF = new UserBasedCF(mUsersItems);
}
```

Figura 25: Classe "CollaborativeFiltering" - gera a matriz de avaliações de itens por usuários

Após gerada a matriz de avaliações de itens por usuários, é preciso fazer a predição das avaliações que ainda não foram feitas, para que o sistema identifique as prováveis preferências de cada usuário. Para isso, os usuários com perfil similar, informalmente chamados de “vizinhos”, são identificados a partir do método de similaridade do Cosseno. Uma vez que estejam identificados os vizinhos de um usuário, utiliza-se o método *k-nearest neighbor* (KN-Neigh-bors) para calcular a preferência de cada vizinho de forma a estimar a nota que o usuário daria ao item.

Após a execução do algoritmo de predição “predictRating”, conforme Figura 26, teremos a matriz completa. Ou seja, as notas dos itens ainda não avaliados estão preenchidas com as

predições calculadas por esse algoritmo. A partir de agora, é possível fazer a recomendação dos itens. Para isso, basta pegar os itens cujas avaliações foram estimadas como mais positivas e apresentá-los ao usuário. Porém, neste passo da recomendação, foi encontrada uma limitação do algoritmo de filtragem colaborativa. Esse e outros problemas encontrados são abordados na seção 5.3.

```
public double predictRating(int userI, int itemJ){
     /* Get target rows and target columns.*/
     double[] itensTargetUser = this.ratingGraph.getRow(userI);
     double[] consumers = this.ratingGraph.getCol(itemJ);

     /* Set up variables involved in the estimative*/
     KNNeighbors nn = new KNNeighbors(kNN);

     /* For each consumer who rated item does... */
     for (int i = 0; i < consumers.length; i++) {
          if (consumers[i] > 0) {
               /* products rated by user id */
               double[] itensRow = ratingGraph.getRow(i);
               /* calculate similarity between vector of products rated by
               user id and _user */
               double similarity = 0;
               if (similarityMatrix[userI][i] == 0){
                    similarity = similari-
                    tyMeasure.calculateSimilarity(itensRow, itensTargetUser,
                    this.similarityWidth);
                    if (similarity == 0){
                         similarity = -1;
                    }
                    similarityMatrix[userI][i] = similarity;
                    similarityMatrix[i][userI] = similarityMa-
                    trix[userI][i];
               }
               similarity = similarityMatrix[userI][i];
               /* If there is similarity between users, computes it on the
               average */
               if (similarity > 0){
                    nn.add(similarity, itensRow[itemJ], avgUserRating[i]);
               }
          }
     }
     double result = nn.predict(avgUserRating[userI]);
     return result;
}
```

Figura 26: Método "predictRating" - responsável por predizer a nota de um usuário para um item.

5.3 Problemas Encontrados

Ao longo do desenvolvimento do TirinhaZ foram encontrados alguns obstáculos que precisaram ser contornados de certa forma. O primeiro foi uma limitação do *framework* MDArte, que foi identificada ao tentar carregar a matriz de similaridade calculada ao iniciar o servidor. O problema era que o método que gerava a matriz de similaridade pertencia ao pacote do domínio da aplicação e chamava uma instância do pacote de serviços, que por sua vez, dependia do domínio. Portanto, não era possível chamar um serviço enquanto o módulo do domínio estava sendo carregado. Para solucionar esse problema, a estrutura estabelecida foi desrespeitada, e os arquivos que implementavam as classes foram inseridos no pacote de serviços.

Somado a isso, também houve a necessidade de serem contornados problemas com relação a um caso de uso público solicitando acesso a serviços privados. O sistema possui, em sua grande maioria, casos de uso privados, porém fez-se necessário a criação de 2 casos de uso públicos, um para criação de contas de novos usuários e outro para exibição de imagens sem a necessidade do usuário estar autenticado no sistema. Entretanto, esses dois casos de uso não possuem permissão para acessar serviços internos, para isso, foi necessário utilizar-se de uma autenticação temporária a fim de conseguir acessar esses serviços. Esta autenticação funciona da seguinte forma: ao acessar um caso de uso público que necessite acessar um serviço privado, o sistema realiza uma rápida autenticação como “visitante”, utiliza o serviço e ao fim se desconecta desse acesso.

Outro empecilho encontrado está na integração com a rede social *Facebook.* Seus *plugins* são mantidos de forma a prevenirem distúrbios causados por sites maliciosos Toda via, isso acabou sendo em entrave para o funcionamento adequado no sistema TirinhaZ, pois além de não implementar o protocolo “https”, o TirinhaZ não possui links explícitos e estáticos em suas *meta tags*, devido a necessidade de formação do link a cada vez que uma página é carregada, de acordo com um id referente a uma tirinha. Então, em algumas situações, isso não é muito bem resolvido pelo *crawler* que executa as varreduras em busca das *meta tags* extremamente grandes e não amigáveis utilizadas para mapear *plugins* como o “curtir” e o “recomendar”.

Por fim, ainda em relação a integração com a rede social escolhida, o grande tempo para atualização das *meta tags* no *cache* do *Facebook* foi algo observado. O sistema demora em média até 24 horas para poder atualizá-lo por completo, dificultando assim o funcionamento adequado de certos *plugins* que utilizam *url* fomada de maneira implícita e dinâmica, pois não é toda vez que são atualizadas imediatamente após o usuário acessar.

CONSIDERAÇÕES FINAIS

O trabalho buscou mostrar como a ferramenta TirinhaZ de recomendação de tirinhas foi concebida, explorando cada etapa do seu desenvolvimento. Além disso, demonstrou como a mesma pode ter finalidade de proporcionar um ambiente agradável e inteligente, onde o usuário recebe sugestões personalizadas de acordo com as suas próprias preferências. Além de oferecer uma integração com a rede social de grande importância no momento, o *Facebook*.

Uma das principais contribuições deste trabalho é a estrutura genérica do modelo conceitual do sistema de recomendação desenvolvido, pois será de grande utilidade para o meio acadêmico devido a independência de algoritmos. Com isso, possibilita a adoção de diferentes algoritmos de recomendação bastando apenas adaptar a interface proposta neste trabalho.

As dificuldades encontradas no processo de criação do sistema como as limitações do *framework* MDArte e do algoritmo de recomendação, são pontos que merecem atenção. Apesar de serem soluções consolidadas e amplamente aceitas, possuem uma significativa fragilidade explicitada neste projeto. Futuramente, esses pontos fracos podem ser vistos como pontos de melhoria destas abordagens.

Dessa forma, identifica-se como trabalhos futuros a adição de novos parâmetros a serem considerados pelo algoritmo de recomendação. Acredita-se que padrões de humor seriam uma boa escolha para incrementar a taxa de acerto das recomendações. Ao acessar o sistema, o usuário seria questionado sobre como está se sentindo naquele dia, se feliz, muito feliz, triste ou categorias emocionais que melhor contemplem o cenário de avaliação.

Sendo o volume de captura de informações e consequente teste de algoritmos um dos objetivos desse trabalho, futuramente o sistema será traduzido para outras línguas, principalmente a língua inglesa, para que seja atingido um público cada vez maior e diversificado.

Paralelamente, dados como idade, sexo e até mesmo região onde vive, poderão ser utilizados para otimizar a eficácia do algoritmo ou para criação de algoritmos extras a serem utilizados em conjunto. Igualmente importante, haverá a possibilidade de explorar dados extraídos do usuário em interações de redes sociais, a principal delas, o *Facebook* (o qual possui uma API (*open graph*) que permite o acesso a dados como atividades recentes, horários, gostos e costumes que o usuário compartilhou através da rede).

Entrando na esfera de avaliação da acurácia do sistema, será possível analisar os dados obtidos e concluir o quanto eficaz foi o algoritmo implementado para cada aspecto do sistema. A capacidade de identificar como se originou cada nota, se ela foi dada através de uma recomendação, se foi atribuída por um usuário que acessou a tirinha por ter visto que a mesma possuía uma nota média muito alta ou até mesmo a avaliou devido ao seu número alto de visualizações, é uma característica do sistema que permite realizar muitas analises sobre os dados recolhidos e consequentemente a melhora dos algoritmos envolvidos.

O sistema também foi construído para, de forma fácil, testar cenários nos quais a introdução de um novo viés possa fazer total diferença na análise de dados. Um exemplo disso é a apresentação prévia do gráfico de notas ao usuário ao visualizar uma tirinha a ser recomendada. Podendo-se, assim, analisar dados que reflitam o comportamento do quanto informações prévias podem afetar na atribuição de notas.

Muitas são as possibilidades de expansão deste trabalho. Indo do simples melhoramento da forma de como os dados são apresentados à introdução de novos algoritmos, sejam eles baseados em conteúdo ou utilizando formas conjuntas de métodos diferentes. O fato é que independente da escolha de qual algoritmo será utilizado, a capacidade do sistema em rastrear as notas e posteriormente as mesmas poderem ser analisadas, é mantida, preservada, possibilitando assim inúmeras análises estatísticas em diferentes cenários.

REFERÊNCIAS BIBLIOGRÁFICAS

Adomavicius, G., & Tuzhilin, a. (2005). Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, *17*(6), 734–749. doi:10.1109/TKDE.2005.99

AndroMDA. (2010). Model Driven Architecture Framework. Retrieved from http://www.andromda.org

Balabanovic, M., & Shoham, Y. (1997). Fab: content-based, collaborative recommendation. *Communications of the ACM*. Retrieved from http://dl.acm.org/citation.cfm?id=245124

Burke, R. (2002). Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, 1–29. Retrieved from http://www.springerlink.com/index/N881136032U8K111.pdf

Cazella, S., Nunes, M., & Reategui, E. (2010). A Ciência da Opinião: Estado da arte em Sistemas de Recomendação. *André Ponce de Leon F. de …*. Retrieved from http://200.17.141.213/~gutanunes/hp/publications/JAI4.pdf

Claypool, M., Gokhale, A., & Miranda, T. (1999). Combining content-based and collaborative filters in an online newspaper. *Proceedings of ACM* .

Cunha, A. (2007). CE-235 Real-time Embedded Systems Lecture Notes. *Brazilian Aeronautics Institute of Technology–ITA.*

Deshpande, M., & Karypis, G. (2004). Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems (…*, *22*, 143–177.

Frakes, W. B. and Isoda, S. (1994). Success factors of systematic software reuse, 14–19.

Goldberg, D., Nichols, D., Oki, B. M., & Terry, D. (1992). Using collaborative filtering to weave an information Tapestry ., *61*(10), 1–10.

Griss, M. (1994). Software reuse experience at Hewlett-Packard. *Software Engineering, 1994. Proceedings. ICSE-16.*

Griss, ML., Jacobson, I., Jonsson, P. (1997). Software reuse architecture, process, and organization for business success. *Computer Systems and Software Engineering, 1997.*

Huang, Z., Chung, W., Ong, T., & Chen, H. (2002). A graph-based recommender system for digital library. … *-CS joint conference on Digital* …. Retrieved from http://dl.acm.org/citation.cfm?id=544231

Karypis, G. (2001). Evaluation of item-based top-n recommendation algorithms. *Proceedings of the 14th ACM international* …, 446–452. Retrieved from http://dl.acm.org/citation.cfm?id=1099683

KLEPPE, A., WARMER, J., BAST, W. (2003). MDA Explained: The Model Driven Architecture.

Krueger, C. (1992). Software reuse. *ACM Computing Surveys (CSUR).* Retrieved from http://dl.acm.org/citation.cfm?id=130856

Lopes, G. R. (2007). Sistemas de Recomendação para Bibliotecas Digitais sob a Perspectiva da Web Semântica. *69f. Dissertação (Mestrado em Ciência da Computação) - Instituto de Informática, UFRGS, Porto Alegre.).* Retrieved from http://hdl.handle.net/10183/10747

Machado, M. D. O., & Cazella, S. (2006). RecImóveis: Um Sistema de Recomendação para auxiliar consumidores na decisão de compra de imóveis.

MDARTE. (2010). MDArte. Retrieved from http://www.softwarepublico.gov.br

Miller, J., & Mukerji, J. (2003). MDA guide. *Object Management Group*.

Pressman. (2006). *Engenharia de software*. Rio de janeiro. Retrieved from http://www.vqv.com.br/es/ES_JE01d_Pressman.pdf

Primo, T., & Loh, S. (2006). Técnicas de Recomendação para usuários de Bibliotecas Digitais. *SBSI-Simpósio Brasileiro de Sistemas de* …, 1–19. Retrieved from http://paginas.ucpel.tche.br/~loh/pdfs/sbsi-2006-primo.pdf

S. Beydeda, M. Book, V. G. (Eds. . (2005). *Model-driven software development*. Retrieved from www.springerlink.com/index/d4mr0231v6j73244.pdf

SEGARAN, T. (2008). Programando a inteligência coletiva. *Alta Books*, 282.