Universidade Federal do Piauí
Campus Senador Helvídio Nunes de Barros
Curso Bacharelado em Sistemas de Informação

Francisca Pâmela Carvalho dos Santos

Sistema de Recomendação de *Softwares* para Suporte de Computadores com o uso de Filtragem Colaborativa

PICOS
2013

Francisca Pâmela Carvalho dos Santos

Sistema de Recomendação de *Softwares* para Suporte de Computadores com o uso de Filtragem Colaborativa

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Sistemas de Informação do Campus Senador Helvídio Nunes de Barros da Universidade Federal do Piauí como parte dos requisitos para obtenção do Grau de Bacharel em Sistemas de Informação, sob orientação da Professora Msc. Juliana Oliveira de Carvalho.

PICOS

2013

Eu, **Francisca Pâmela Carvalho dos Santos**, abaixo identificado(a) como autor(a), autorizo a biblioteca da Universidade Federal do Piauí a divulgar, gratuitamente, sem ressarcimento de direitos autorais, o texto integral da publicação abaixo discriminada, de minha autoria, em seu site, em formato PDF, para fins de leitura e/ou impressão, a partir da data de hoje.

Picos-PI 20 de setembro de 2013.

Francisca Pâmela Carvalho dos Santos.
Assinatura

FICHA CATALOGRÁFICA
Serviço de Processamento Técnico da Universidade Federal do Piauí
Biblioteca José Albano de Macêdo

S237s Santos, Francisca Pâmela Carvalho dos.
Sistema de recomendação de softwares para suporte de computadores com o uso de filtragem colaborattiva / Francisca Pâmela Carvalho dos Santos. – 2013.
CD-ROM : il. ; 4 ¾ pol. (56 p.)

Monografia(Bacharelado em Sistemas de Informação) – Universidade Federal do Piauí. Picos-PI, 2013.
Orientador(A): Prof. Msc. Juliana Oliveira de Carvalho

1. Sistema de Recomendação. 2. Dispositivos Móveis. 3. Suporte de Computadores. I. Título.

CDD 005.1

Francisca Pâmela Carvalho dos Santos

Sistema de Recomendação de *Softwares* para Suporte de Computadores com o uso de Filtragem Colaborativa

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Sistemas de Informação do Campus Senador Helvídio Nunes de Barros da Universidade Federal do Piauí como parte dos requisitos para obtenção do Grau de Bacharel em Sistemas de Informação, sob orientação da Professora Msc. Juliana Oliveira de Carvalho.

Data de Aprovação:

13/09/2013

Juliana Oliveira de Carvalho _____ UFPI-CSHNB

Rayner Gomes Sousa _____ UFPI-CSHNB

Ismael de Holanda Leal _____ UFPI-CSHNB

PICOS

2013

Dedico este trabalho a Deus, pelo dom da vida e por guiar sempre os meus passos, sem Ele eu nada sou. Dedico também aos meus pais, a quem tenho como exemplo de perseverança e humildade e a minha irmã (eterna cúmplice), amo muito vocês! Enfim dedico a finalização desse trabalho a todos aqueles que fizeram e fazem parte da minha caminhada, família e verdadeiros amigos.

Hoje posso dizer que essa longa caminhada resultou na realização de um sonho, que exigiu muita determinação e ousadia para ser concretizado. Agradeço a todos aqueles que de alguma forma contribuiram para finalização de mais essa etapa na minha vida. Em especial agradeço...

À Deus que me proporcionou força para realização desse trabalho e de todas as minhas demais conquistas. *"Digno és, Senhor, de receber glória, e honra, e poder; porque Tu criaste todas as coisas, e por Tua vontade são e foram criadas." (Apocalipse 4:11).*

À minha família que sempre me apoiou em todos os momentos. Em especial, aos meus pais Carmo Leal dos Santos e Francisca Almeida de Carvalho dos Santos, meus maiores exemplos, e a minha irmã Carmem Jéssica Carvalho dos Santos, por me incentivarem e me orientarem nos momentos mais difíceis, apresentando os meus sonhos e objetivos em suas orações.

À minha orientadora, Juliana Oliveira de Carvalho (mãezona), com quem aprendi muito durante todo esse tempo e a quem tenho como um exemplo a seguir. Obrigada pela disposição em dividir seu tempo e conhecimento, pelos conselhos ofertados, pelas vezes que me acalmou pelo telefone e principalmente pela sua amizade, fortalecida ao longo desses anos de curso.

À professora Ana Maria Koch, que me ensinou como ir em busca do que quero, mostrando a importância de dedicar tempo e esforço para conseguir chegar lá. Obrigada pela confiança e pelos ensinamentos, com certeza serão muito importantes ao longo dessa jornada.

À todos os demais professores, que agregaram conhecimento a minha vida, desde o jardim de infância até a finalização desse curso superior. Em especial aos professores: Ismael Holanda, Ivenilton Alexandre, Patrícia Medyna, Frank César, Dennis Sávio, Rayner Gomes, Fredson Muniz, Patrícia Vieira, Arlino Henrique, Algeir Sampaio, Júlio César, Leonardo Sousa, Isabel Orquiz, Laurindo Neto, Ricardo Viana e Ryan Azevedo.

Ao meu namorado, Celles Nunes, por todo o carinho, cuidado, paciência e compreensão que me dedicou, assim como pela demonstração de confiança que depositou em mim. Obrigada por estar sempre comigo. Te adoro muito fofinho :* !

À todos os meus amigos, em particular Danila Feitosa (irmã), Ataniel Soares (irmão), Kaio Moura (maninho), Cliciano Sabino, Ohana Morais, Guilherme Feitosa, Klisanderson de Sousa, Ilzilene Reis, Carlos Henrique, Erismar Araújo, Pamela Graziella, Rafael Morais, Jonnison Lima e a toda a galerinha do RELIH, pela grande contribuição e incentivo no desenvolvimento desse TCC e pela cumplicidade e amizade oferecida em todos os momentos.

Obrigada a todos que, mesmo não tendo seus nomes citados aqui, contribuiram direta e indiretamente para conclusão dessa etapa.

“O mais importante de tudo é nunca deixar de se perguntar. A curiosidade tem sua própria razão de existir.”

Albert Einstein

“Não deixe o barulho da opinião dos outros abafar sua voz interior. E mais importante, tenha a coragem de seguir seu coração e sua intuição. Eles de alguma forma já sabem o que você realmente quer se tornar. Tudo o mais é secundário.”

Steve Jobs

Resumo

Com a evolução da tecnologia houve uma grande ampliação na quantidade de informações disponibilizadas em meios eletrônicos, o que pode acarretar em uma sobrecarga de conteúdo irrelevante, assim como uma maior procura por *softwares* para esses dispositivos. Com o uso de um sistema de recomendação, pode-se abstrair o que é relevante a ser usado, através de recomendações que satisfaçam ao interesse do usuário. A proposta deste trabalho é desenvolver um sistema de recomendação de *softwares* para suporte de computadores, para plataforma Android, com objetivo de entender como se desenvolve sistemas de recomendação, de implementar aplicações para dispositivos móveis, para os quais existem um crescente número de usuários e uma maior acessibilidade para mesmos, além de desenvolver uma aplicação que possa contribuir com recomendações de *softwares* para auxiliar na solução de problemas do sistema e de *softwares* para o usuário.

Palavras-chave: Sistemas de Recomendação, Dispositivos Móveis, Suporte de Computadores.

Abstract

With the evolution of technology there has been a huge increasing in the amount of information available on electronic media, which may result in an overload of irrelevant content, as well as a higher demand for software for these devices. With the use of a recommendation systems, you can abstract what is relevant to be used, through recommendations that satisfy the interest of the user. The purpose of this work is to develop a software recommendation system of softwares for computer support for the Android platform, aiming to understand how to develop recommendation systems, to implement mobile applications, for which there are a growing number of users and greater accessibility to them, and to develop an application that can help with software recommendation to assist in system troubleshooting and softwares for the user.

Keywords: Recommendation Systems, Mobiles, Computer Support.

Lista de Figuras

Lista de Tabelas

Lista de abreviaturas e siglas

ADT	*Android Development Tools*
API	*Application Programming Interface*
GPS	Sistema Global de Posicionamento
HTTP	*Hyper Text Transfer Protocol*
IA	Inteligência Artificial
IDE	*Integrated Development Environment*
PDA	*Personal Data Assistent*
RAM	*Random Access Memory*
REST	*Representational State Transfer*
SDK	*Software Development Kit*
SGBD	Sistema Gerenciador de Banco de Dados
SO	Sistema Operacional
SOA	*Service Oriented Architecture*
SR	Sistema de Recomendação
SR Soft	Sistema de Recomendação de Softwares para Suporte Técnico de Computadores
URL	*Uniform Resource Locator*
WS	*Web Service*
WSO	*Weighted Slope One*

Sumário

1 Introdução

Com o avanço da tecnologia, os meios de realização de pesquisas se tornaram amplos, assim como as informações disponíveis na *Web*, dificultando a escolha de conteúdos relevantes, visto que a grande variedade de opções retornadas em uma pesquisa faz com que o leitor perca muito tempo selecionando os dados que considera importante.

Atualmente há um grande desenvolvimento de tecnologias para comunicação móvel: *Smartphones, Tablets, PDAs*, etc.. Esses dispositivos evoluem a cada dia e são muito populares, por oferecerem cada vez mais recursos, tornando-se uma realidade no nosso cotidiano. Aparelhos mais modernos e completos requerem sistemas operacionais complexos e com grande capacidade de gerenciamento, dentre os quais se podem citar: *Windows Mobile*, *Symbian OS*, e *Google Android*.

Sistemas colaborativos são sistemas de interação que proporcionam maior facilidade de se chegar a uma solução. Um grupo consegue avaliar e obter uma solução mais rápido do que um indivíduo isolado. Os sistemas de recomendação possibilitam ao usuário o conhecimento de ítens e/ou informações variadas através da análise do seu perfil e possivelmente da avaliação positiva de outros usuários.

Com o objetivo de auxiliar o pesquisador a encontrar elementos que sejam do seu interesse, os sistemas de recomendação filtram essa diversidade de opções que são retornadas, com base no seu perfil, identificando a pretensão do mesmo.

O *Google Android*, baseado no SO (Sistema Operacional) *Linux*, se mostrou o mais indicado para ser usado na atividade de desenvolvimento, por apresentar um sistema de desenvolvimento flexível ao programador e por ter o código-fonte aberto e de livre acesso aos desenvolvedores, além de oferecer suporte a diversos serviços e *hardwares*, o qual utiliza uma máquina virtual *Dalvik* que pode converter o *bytecode* compilado a partir da máquina virtual *Java* em *bytecode* da *Dalvik* . A plataforma *Android*, cuja arquitetura é baseada em componentes, possui integração com IDE's populares como *Eclipse* e *Netbeans* e as aplicações desenvolvidas são de fácil instalação nos dispositivos.

Esse grande número de dispositivos tecnológicos faz com que seja necessária a utilização de *softwares* que atentem para o trabalho e entretenimento do usuário.

Existem muitos programas desenvolvidos para o suporte técnico de computadores, porém poucos possuem a especificidade que o usuário procura ao realizar uma pesquisa. O obje-

tivo deste trabalho é propor um Sistema de Recomendação para o Suporte Técnico de Computadores que, a partir do perfil do usuário, faça recomendações de *softwares* aprovados por outros usuários com perfil semelhante, procurando assim sugerir programas que sejam relevantes para sua necessidade.

Após a introdução, que relatou sobre o conceito e objetivos do sistema serão apresentados os próximos capítulos que estão organizados da seguinte forma:

- Capítulo 2 - São apresentados os conceitos de Sistemas de Recomendação, sua importância para sociedade e sua classificação, apresentando as técnicas disponíveis para geração das recomendações.
- Capítulo 3 - É ostentado o desenvolvimento para dispositivos móveis, explicando sobre a plataforma *android*, a linguagem *JAVA* utilizada para o desenvolvimento do sistema nessa plataforma, e sobre a arquitetura do *Android*. Também será tratado nesse capítulo sobre o conceito de *WebService* e da técnica *REST*.
- Capítulo 4 - Será mostrada a documentação e a implementação do sistema, abordando a metodologia utilizada no desenvolvimento do aplicativo e os resultados obtidos, demonstrando os detalhes técnicos necessários para constituição dele. As ferramentas utilizadas são melhor especificadas e explicadas nesse capítulo.
- Capítulo 5 - Por fim, o capítulo de conclusão mostra os objetivos alcançados e os aspectos positivos do sistema, com sugestões para trabalhos futuros.

2 Sistemas de Recomendação

Este capítulo aborda o conceito e a importância dos Sistemas de Recomendação, retrata sobre sistemas colaborativos e mostra as técnicas mais utilizadas para gerar as recomendações aos usuários.

Com o avanço da tecnologia a *Internet* passou a ser uma das maiores ferramentas de busca por informação, porém com a grande quantidade de arquivos e informações disponíveis hoje em dia na *Web* acaba por acontecer um fenômeno conhecido como "sobrecarga de informação" (LOPES, 2007), dificultando assim ao usuário obter informações realmente importantes ao contexto de suas necessidades. Com o uso de sistemas de recomendação é possível a seleção de itens (livros, *softwares*, músicas e etc.) com base no interesse dos usuários. Eles têm por objetivo auxiliar na obtenção de informações que sejam relevantes ao usuário, seja pela avaliação do seu perfil ou pelo perfil do grupo, relata Cazzela (CAZELLA, 1998), para então diminuir a sobrecarga de informação.

Os sistemas de recomendação necessitam do estudo do comportamento dos usuários de acordo com suas preferências e com seu perfil, para que assim o mesmo possa enviar e receber recomendações. Esses sistemas fazem parte da área da Inteligência Artificial (IA) (CARDONA, 2010) e utilizam suas técnicas para relacionar o interesse das pessoas em algum assunto específico e a indicação de outras pessoas que possuam importâncias em comum, para separar o que seria mais importante para as mesmas.

Os Sistemas de Recomendação são utilizados com o objetivo de identificar usuários, para a partir do armazenamento de suas preferências recomendar os itens (BARCELLOS, 2007). Eles almejam sugerir itens de um determinado domínio que melhor se encaixam no perfil de interesse do usuário e assim realizar a recomendação.

2.1 Sistemas Colaborativos

Os sistemas colaborativos são ambientes que proporcionam o compartilhamento e a interação entre pessoas, visando assim facilidade em propagar informações e aprendizagens, assim como conteúdos físicos (vídeos, músicas, programas e etc.). Através de ambientes colaborativos é possível obter melhores resultados pelo trabalho em grupo, podendo haver discussões, reflexões e troca de experiências das pessoas envolvidas, dando assim suporte a realização de

um objetivo comum (FILIPPETTO, 2011).

A medida que usuários trocam informações e interagem uns com os outros, vai sendo criada uma "inteligência coletiva". O termo inteligência coletiva é empregado para distinguir "o conhecimento que emerge da interação e da colaboração" (GEROSA, 2009) entre pessoas em vários meios digitais como, por exemplo, em redes sociais, no *e-commerce* e em sistemas de recomendação.

Os sistemas de recomendação são uma especificidade dos sistemas colaborativos, sendo uma das áreas de pesquisa em técnicas para colaboração (ISOTANI, 2012). Para quem recebe a recomendação, a colaboração funciona como um filtro ou uma interpretação específica de uma variedade de possibilidades normalmente difíceis de se encontrar.

2.2 Definição de Sistemas de Recomendação

Segundo Barbosa (BARBOSA, 2009) os Sistemas de Recomendação (SR) são sistemas que utilizam técnicas computacionais para fazer a seleção de itens que satisfaçam determinado usuário, objetivando auxiliá-lo na obtenção de informações relevantes ao seu perfil.

De acordo com Mendez (MENDEZ, 2010) os Sistemas de Recomendação são sistemas usados para identificar usuários do sistema, salvar em histórico suas preferências e recomendar itens que podem ser serviços, produtos e /ou conteúdos, levando-se em consideração as necessidades e interesses do usuário e utilizando algoritmos de recomendação para gerar a saída, como mostra a Figura 1.

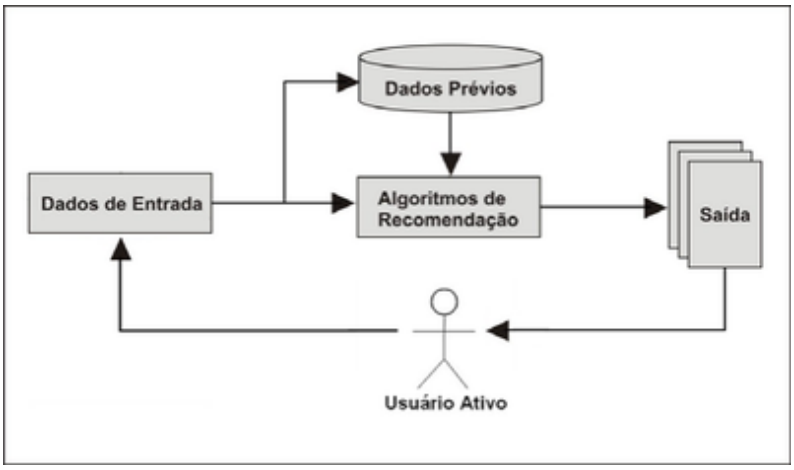

***Figura 1** – Sistema de Recomendação*

Disponível em: <http://bd.iff.edu.br/acervo/um-sistema-de-recomendacao-para-locadoras-de-video-teresina-2010/nsiimagegrain.2012-06-21.9940679673>

A colaboração em um SR ocorre no momento em que um usuário pode compartilhar informações e outros usuários possam deixar seu ponto de vista em relação a essa informação

partilhada, para que então possam ser identificados conteúdos adequados para cada indivíduo com base em suas particularidades (CAZELLA, 1998).

Pelas definições citadas, percebe-se que o foco dos sistemas de recomendação é satisfazer os usuários através de recomendações baseadas nas suas precisões e interesses e realizadas através de técnicas computacionais. Esses elementos compartilhados através das recomendações proporcionam a colaboração do sistema.

2.3 Classificação dos Sistemas de Recomendação

Existem algumas técnicas para a geração da recomendação, que procuram auxiliar na solução de problemas que causam a sobrecarga de informações. Essas técnicas são tecnologias que utilizam abordagens colaborativas para gerar uma recomendação, para melhor compreensão dessa abordagem, podemos observar a afirmativa:

> A abordagem colaborativa de recomendação mantém suas características principais e consiste fundamentalmente em recomendar um artefato (e.g. um livro, um filme, uma página da *web*) que foi preferido por usuários similares ao usuário que recebe a recomendação (ALBUQUERQUE, 2008).

Os sistemas de recomendação podem ser classificados em três divisões principais: Filtragem baseada no conteúdo, Filtragem colaborativa e Filtragem Híbrida. Essas classificações, segundo Mack (MACK, 2010), se diferenciam pelo modo de utilização dos padrões de comportamento e relacionamento com os usuários.

2.3.1 Filtragem Baseada em Conteúdo

A abordagem de filtragem baseada em conteúdo *(Content-based Filtering)* é aplicada através da análise de itens semelhantes ao que está sendo recomendado, que foram procurados anteriormente pelo usuário, podendo então também ser alvo de interesse do mesmo. Essa técnica visa recomendar itens relacionados ao perfil do usuário.

O perfil do item consiste de algumas informações que descrevam seu conteúdo, enquanto que o perfil do usuário é criado de acordo com atributos que descrevam os interesses do usuário com relação ao perfil dos itens. Para realizar a recomendação é necessário utilizar funções de similaridade, ou seja, comparar a descrição de cada item com a descrição da necessidade do usuário, para fazer a análise dos dois perfis traçados visando à satisfação do usuário (LOPES, 2007).

Essa técnica focaliza os algoritmos capazes de aprender as preferências do usuário e separar os itens que estejam mais próximos dessas prioridades (NODARI, 2008).

Uma desvantagem do uso dessa técnica para geração da recomendação ao usuário, de acordo com Bezerra (BEZERRA, 2004), é a limitação da representação do conteúdo dos itens,

pois em alguns casos apenas variáveis de escalas quantitativas e qualitativas não são suficientes para uma recomendação satisfatória. Lucas (LUCAS, 2010) declara que outra desvantagem do uso dessa técnica são os itens sinônimos, ou seja, itens iguais ou semelhantes mas com atributos distintos, fazendo com que eles sejam tratados como diferentes.

2.3.2 Filtragem Colaborativa

Na filtragem colaborativa *(Collaborative Filtering)* analisa-se o perfil de um grupo de usuários para selecionar os usuários que possuem perfil semelhante ao que receberá a recomendação. A partir dessa seleção, a avaliação feita por esses usuários será de total relevância para a realização da recomendação. A "filtragem colaborativa gera as recomendações baseando-se em avaliações de outras pessoas com gostos semelhantes" (NODARI, 2008).

Quando as pessoas precisam escolher algo sem ter conhecimento das opções é natural que elas procurem saber das experiências e opiniões dos outros, buscando assim a recomendação de pessoas com quem tenha interesses em comum. Essa técnica se baseia na análise da similaridade do interesse dessas pessoas pelos itens, para então gerar a recomendação, ou seja a filtragem é baseada na avaliação feita pelos usuários daquele item, ao invés do conteúdo do mesmo (BARBOSA, 2009).

Para encontrar a similaridade entre os usuários é necessário realizar o cálculo do coeficiente de similaridade e o cálculo da predição. Entende-se por cálculo do coeficiente de similaridade a "etapa preliminar de um processo de Filtragem Colaborativa, e da seleção de subconjuntos de usuários com maiores similaridades, denominados vizinhos, que serão considerados na etapa de predição" (BARBOSA, 2009). Esses "vizinhos" são assim classificados por possuírem interesses semelhantes.

O Coeficiente de Correlação Linear de Pearson é vastamente utilizado para realizar o cálculo da similaridade entre duas variáveis, medindo assim o grau de relacionamento entre elas, variando de -1 (sem correlação) a 1 (grande correlação) (BARBOSA, 2008).

Já o cálculo das predições é feito para "indicar o quão apropriado é um item para determinado usuário" (BARBOSA, 2009) e ela é calculada sem dependência com o coeficiente utilizado para definir a similaridade entre os usuários.

Segundo Lucas (LUCAS, 2010) o uso dessa técnica pode ter algumas desvantagens, tais como:

- **Arranque a Frio** - Acontece quando um usuário ou item novo entra no sistema e há poucas informações sobre ele. Apresenta-se assim, dificuldade de encontrar itens semelhantes a um determinado item ou de encontrar usuários semelhantes a um novo usuário.
- **Classificações Falsas** - Está relacionado a possibilidade de existir usuário com classifi-

cações que não correspondem as suas preferências reais.

A Filtragem Colaborativa é subdividida em duas categorias: a filtragem baseada no item que analisa as avaliações atribuidas a um determinado item e a filtragem baseada no usuário que observa o perfil dos usuários, atentando para seus interesses. O algoritmo *SlopeOne*, do tipo baseada no item, destaca-se nesse meio por ser uma abordagem simples e eficiente (SOUZA, 2012).

Coeficiente de Correlação de Pearson

O "Coeficiente de Correlação" é uma das criações do estatístico britânico Karl Pearson (1857-1936), conhecido como o criador da Estatística Aplicada (BENITEZ, 2007). Devido ao nome do coeficiente de correlação, muitos acreditam que essa medida estatísca foi desenvolvida unicamente por Karl Pearson, mas como relata Stanton (STANTON, 2001) a origem desse coeficiente é um trabalho conjunto de Pearson com Sir Francis Galton, cientista do século 19, que colaborou no desenvolvimento com noções modernas de correlação e regressão. Esse coeficiente é geralmente representado pela letra *r*, sua fórmula é demonstrada na Figura 2:

$$r = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\left(\sum (x_i - \bar{x})^2\right)\left(\sum (y_i - \bar{y})^2\right)}}$$

Figura 2 *– Fórmula Coeficiente de Correlação de Pearson*

Disponível em: <http://medstatweb.med.up.pt/cursop/regressao/imagens/formula_correlacao.gif>

De acordo com Filho (FILHO, 2009) uma definição do conceito do Coeficiente de Correlação de Pearson em uma frase seria: "é uma medida de associação linear entre variáveis". O mesmo aponta que as palavras "associação" e "linearidade" são chaves para entender o coeficiente, pois duas variáveis irão se associar apartir do compartilhamento de variância e essa variação será distribuida linearmente.

Esse método criado para medir a correlação entre duas variáveis, pode ser usado na Análise de Componentes Principais, Análise Fatorial, Análise de Confiabilidade, entre outras (LIRA, 2004).

De acordo com Casarotto (CASAROTTO, 2012) o sinal obtido como resultado do Coeficiente de Correlação de Pearson indica o sentido da correlação (positiva ou negativa), ou seja, se elas são diretamente ou inversamente proporcionais e o valor numérico indica a intensidade (oscila entre -1 e 1), sendo -1 uma correlação negativa perfeita, 1 uma correlação positiva perfeita e 0 a ausência de relação linear. Quanto mais próximo o valor chegar de 1, maior será a

correlação positiva entre as variáveis e quanto mais próximo de -1, maior a correlação negativa, como pode ser visto na Figura 3.

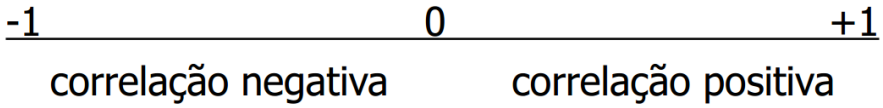

Figura 3 *– Sentido e intensidade do Coeficiente de Correlação de Pearson*

Disponível em: <http://ensino.univates.br/čhaet/Materiais/Bioestatistica_3.pdf>

Podemos observar na Figura 4 um exemplo da variação conjunta entre duas variáveis representadas em um plano cartesiano (X, Y) ou gráfico de dispersão, para demonstrar o grau de relacionamento entre elas. No quadro **A** as variáveis *x* e *y* possuem uma relação linear negativa, ou seja, elas crescem em sentido contrário, se *x* cresce, *y* diminui e vice-versa. No quadro **B**, podemos observar uma relação linear positiva, as variáveis *x* e *y* crescem no mesmo sentido. No quadro **C**, as variáveis em questão, não apresentam nenhuma relação e no **D** não há relação linear entre elas.

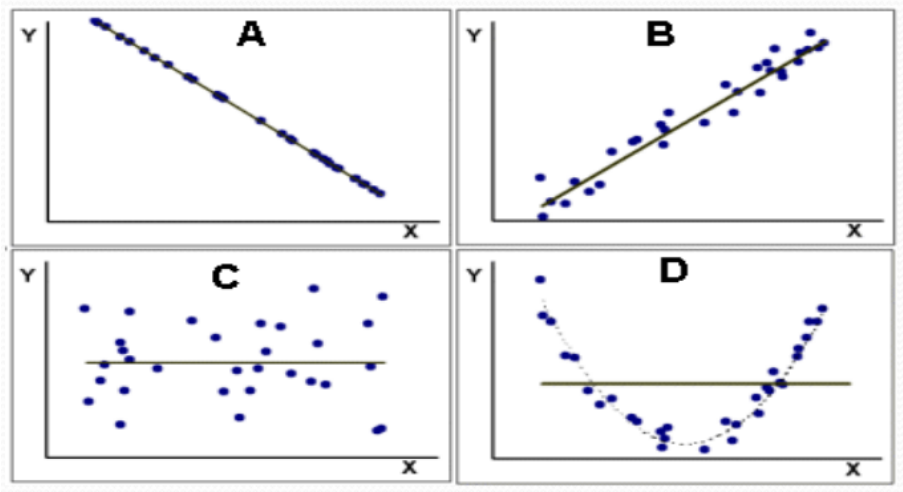

Figura 4 *– Relação Linear entre duas variáveis*

Disponível em: <http://www.de.ufpb.br/ luiz/AED/Aula9.pdf>

Método SlopeOne

Atualmente tem-se utilizado a abordagem colaborativa baseada em ítens para o desenvolvimento de vários SR. O *Weighted Slope One (WSO)* segue essa abordagem e, apesar de ser

um algoritmo simples, possui o desempenho comparado ao de algoritmos de referência (SANTOS, 2012).

O algoritmo denominado *Slope One* foi uma proposta feita por Daniel Lemire e Anna Maclachlan, para sistemas colaborativos baseados em itens (MACLACHLAN, 2005) e calcula a predição, ou seja, a suposta avaliação que um usuário daria para determinado item.

Em concordância com Souza (SOUZA, 2012), o algoritmo *SlopeOne* supõe que o usuário atribuirá notas não binárias aos itens expostos. O algoritmo compara avaliações de outros usuários a esses determinados itens, para então predizer qual seria a nota dada pelo usuário que ainda não avaliou esse item, como por exemplo:

- Usuário A avaliou os itens *x* com nota 3 e *y* com nota 4.5.
- Usuário B avaliou o item *x* com nota 3.5.
- A provável nota que o usuário B daria para o item *y* seria igual a 5.

O cálculo da predição se dá da seguinte forma:

(Nota de A a item x - Nota de A a item y) = (Nota de B a item x - Predição de B em y)

Então:

$$(3 - 4.5) = (3.5 - P)$$
$$P = 3.5 + 1.5$$
$$P = 5$$

Apesar de ser uma algoritmo simples, o Slope One possui pontos fortes a serem destacados (SANTOS, 2012):

- As avaliações realizadas pelos usuários induzem diretamente nas recomendações feitas pelo sistema. Diferente dos sistemas baseados no usuário, cujas similaridades são calculadas de vez em quando e não quando há novas avaliações.
- Os novos usuários são alvos, com poucas avaliações feitas, de recomendações apropriadas.
- Algumas experiências apontam que o algoritmo *Slope One* mostra-se preciso como um algoritmo de referência, tanto baseados no usuário, como baseados em ítens (MACLACHLAN, 2005).

2.3.3 Filtragem Híbrida

A filtragem híbrida *(Hybrid Filtering)* utiliza as duas técnicas citadas: filtragem baseada em conteúdo e filtragem colaborativa (figura 5). A vantagem de se utilizar as duas técnicas, dá-

se pelo fato de que:

> As duas abordagens são complementares. Nesse sentido vários estudos sugerem formas de combinar as técnicas de filtragem baseada em conteúdo com as técnicas de filtragem colaborativa, com o propósito de resolver suas limitações intrínsecas (PEREIRA, 2010)

.

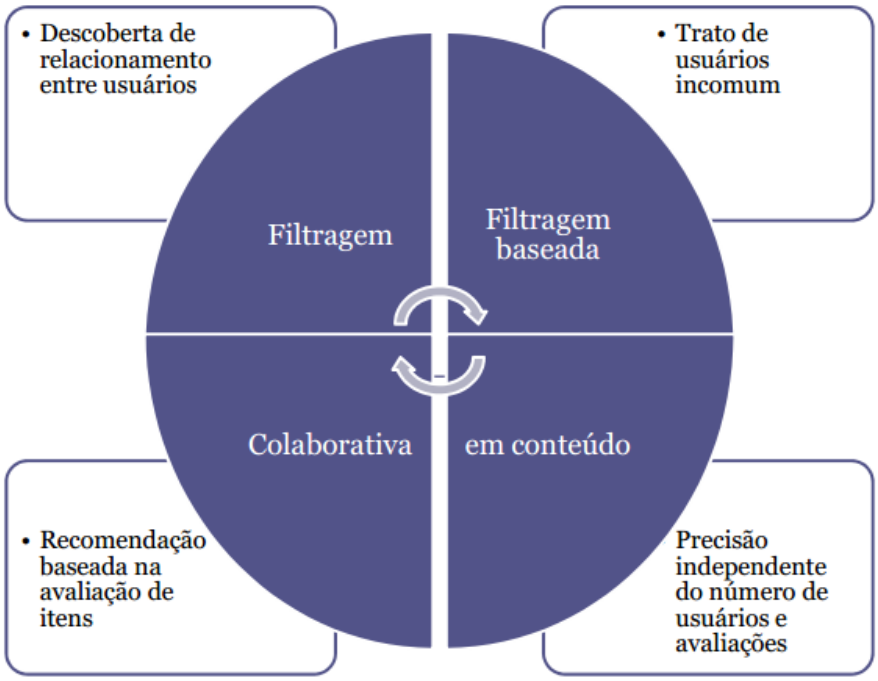

Figura 5 *– Filtragem Híbrida*

Disponível em: <http://kessia.blogs.unipar.br/files/2008/07/sistemas-de-recomendacao.pdf>

Uma das grandes motivações para a criação de sistemas híbridos (PEREIRA, 2010) é o aumento da qualidade de recomendações, ao mesmo tempo em que torna o sistema resultante menos suscetível às desvantagens de uma das técnicas componentes.

3 Desenvolvimento Para Dispositivos Móveis

O número de usuários de dispositivos móveis (*tablets, smartphones*, etc.) cresce a cada dia, incentivado pela maneira como os fabricantes vem oferecendo aparelhos cada vez mais completos (CRUZ, 2011). São diversos os aplicativos criados para os mesmos e que agradam toda uma sociedade, além dos serviços oferecidos, tais como: utilização da tecnologia *Wireless* para transmissão de informações sem o uso de fio, uso do Sistema Global de Posicionamento (GPS), de diversos sensores (acelerômetro, de presença, etc), entre outros recursos.

A escolha por aplicações baseadas em *web* cresce em relação a aplicativos pelas vantagens de disponibilidade ao usuário, menor custo com taxas de manutenção e compatibilidade com múltiplas plataformas (LOPES, 2013). Porém, também existem desvantagens de se utilizar aplicações *web*, como o caso de uma conexão lenta com a *Internet* resultar em demora na execução da aplicação e na maior preocupação que se deve ter com a segurança dos dados.

3.1 Plataforma *Android*

O *Android* é uma plataforma de código aberto próprio para dispositivos móveis, "baseado no sistema operacional *Linux* e com um ambiente de desenvolvimento flexível e poderoso" (SILVA e BRACHT, 2010). Além do *Android*, existem outros sistemas operacionais para dispositivos móveis, tais como: *Symbian OS*, *Iphone OS* e *Windows Mobile*. No entanto podemos destacar a vantagem de se utilizar o *Android*, Cruz (CRUZ, 2011) refere que o sistema operacional *Google Android*, se mostrou o mais recomendado para ser trabalhado, por ter sido arquitetado visando fornecer maior flexibilidade ao programador, além de seu código-fonte ser de livre acesso para comunidade de desenvolvedores. Cruz frisa ainda que há a possibilidade de instalar tal sistema em dispositivos que não o portam nativamente, aumentando assim o número de aparelhos suscetíveis a receber o aplicativo.

O *Android* oferece diversas alternativas de linguagens de programação, como por exemplo: *Java* (adotada pela *Google* para o desenvolvimento de aplicações para a plataforma móvel), *C/C++, .NET Framework, Scala, Lua e Python.*

3.1.1 Linguagem JAVA

"*Java* é a linguagem de programação orientada a objetos, desenvolvida pela *Sun Microsystems*, capaz de criar tanto aplicativos para *desktop*, aplicações comerciais, *softwares* robustos, completos e independentes, aplicativos para a *Web*" (SOBRAL, 2008). Uma das vantagens dessa linguagem são os mecanismos oferecidos para garantir a segurança dos aplicativos.

Ao escrever programas, na linguagem *JAVA*, podem ser gerados *Applets*, Aplicativos ou ainda *Servlets*. Para a execução de aplicativos é necessário ter um interpretador instalado na máquina. O JDK (*Java Developement Kit*) é um kit de desenvolvimento para a linguagem *JAVA* que abrange compilador, interpretador e utilitários (SOBRAL, 2008).

O JAVA também pode ser utilizado para programação de aplicativos para dispositivos móveis, como é o caso do desenvolvimento de aplicativos para *Android*, que utiliza um kit próprio denominado SDK.

3.1.2 Arquitetura

A arquitetura do sistema *Android* introduz a concepção de integração e flexibilidade, ou seja, os aplicativos que são fornecidos juntamente com o sistema podem interagir ou até mesmo serem substituidos por aplicações diversas que não são nativas do sistema (CRUZ, 2011).

Segundo Silva (SILVA, 2010), a estrutura da arquitetura *Android* é dividida em cinco camadas, responsáveis pelo gerenciamento de processos, são elas: Aplicativos, *Framework*, Ambiente de Execução, Bibliotecas e *Kernel Linux*. A Figura 6 mostra essas cinco camadas que o *Android* estabelece e como é o relacionamento entre elas.

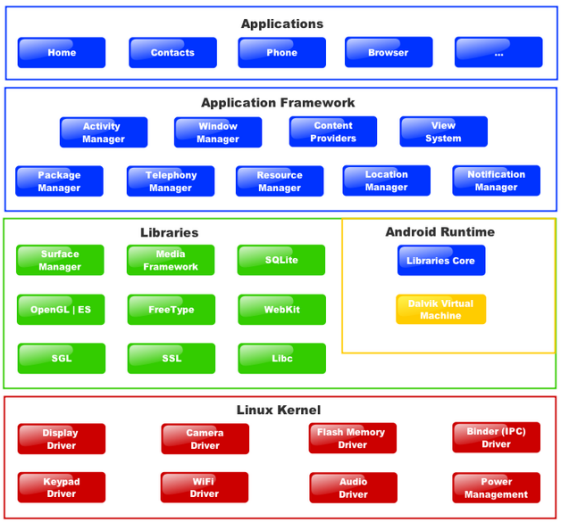

Figura 6 *– Arquitetura da plataforma Android*

Disponível em: <http://blog.clubedaprogramacao.com/2011/12/26/android-parte-2-historia/>

Aplicativos: onde se encontram os aplicativos desenvolvidos em *Java* que são executados pelo sistema. Alguns aplicativos já são disponibilizados pela plataforma *Android*, como por exemplo: calendários, navegador, mapas, etc.

Frameworks: observa-se que o desenvolvimento do mesmo está voltado para o uso de APIs (*Application Programming Interface*) das aplicações chaves do *Android*. São elas:

- *Activity* - é basicamente uma tela do *Android*, uma atividade que funciona como *interface* visual da aplicação com a qual o usuário irá interagir. Cada *activity* é independentes uma da outra.
- *Service* - é a classe responsável pelas tarefas que são executadas em segundo plano, que permite criar um serviço. Não possui uma *interface* visual para o usuário.
- *BroadCastReceiver* - é o componente que tem a função de receber e reagir a uma mensagem do SO. É um mecanismo de alerta.
- *ContentProvider* - é o provedor de conteúdo que serve para compartilhar um conjunto específico de dados para outras aplicações do *Android*, assim como recuperar ou armazenar dados em um repositório.
- *Intent* - é o objeto que contém as mensagens que são utilizadas para facilitar a ligação entre os componentes da aplicação, descrevendo a ação e o que se deseja executar.
- *View* - é todo componente da *interface* gráfica do *Android*, usado para definir os objetos visuais exibidos na tela.

Bibliotecas: é uma coleção de bibliotecas C/C++ utilizadas por vários componentes do sistema. Adaptadas para *Linux*, essas bibliotecas são responsáveis por dispor diversos formatos de vídeo e áudio, funções de gráficos, imagens e acesso a banco de dados e navegador. Tais funcionalidades estão disponíveis aos desenvolvedores através do *framework* de aplicação. Abaixo são apresentadas algumas dessas bibliotecas encontradas na plataforma *Android*:

- *System C Library (libc)* - Implementação aprimorada da biblioteca C para dispositivos baseados em *Linux*.
- *Media Library* - esta biblioteca suporta os principais tipos de imagem, áudio e vídeo, permitindo a gravação e a reprodução desses formatos.
- *Surface Manager* - responsável por gerenciar o acesso ao subsistema de exibição do dispositivo.
- *LibWebCore* - engine moderna para *web browsers*. Moderno navegador *Android*.
- *SGL* - engine para gráficos 2D subjacentes.

- *3D Libraries* - biblioteca utilizada tanto para aceleração de *hardware* como de *software*.
- *FreeType* - biblioteca utilizada para renderização de fontes vetoriais e/ou *bitmap*.
- *SQLite* - banco de dados relacional leve e poderoso utilizado para o gerenciamento de dados por qualquer aplicação.

Ambiente de Execução: nessa camada localiza-se a máquina virtual do *Android*, a *Dalvik*, otimizada especialmente para dispositivos móveis. A *Dalvik* foi desenvolvida para ter suporte a múltiplas instâncias rodando ao mesmo tempo, assim cada aplicação pode rodar seu processo em uma instância da máquina.

Kernel Linux: composta pelo *Kernel* do *Linux*, responsável pelos principais serviços do sistema, como gerenciamento de memória, segurança e gerenciamento de processos. Utilizado como base do sistema operacional *Android*, foi desenvolvido de acordo com as necessidades e peculiaridades dos dispositivos móveis.

3.1.3 Desenvolvimento Com *Android*

Segundo Lecheta (LECHETA, 2010) o *Android*, plataforma de desenvolvimento para aplicativos móveis, utiliza a linguagem *Java* para desenvolvimento das aplicações, aproveitando todos os recursos disponíveis.

O *Android* conta com uma loja virtual, a *Android Market*, que permite ao desenvolvedor disponibilizar uma aplicação gratuitamente para o usuário final, ou ganhar com a sua venda, expõe (MARTINS, 2009) (ARIMA, 2009a).

Uma das grandes apostas dessa plataforma é a disponibilidade do *Android SDK* (*Software Development Kit*), que é composta por um conjunto de *softwares* para dispositivos móveis, um *middleware*, aplicativos e um sistema operacional (PEREIRA, 2010).

3.2 Web Service

Web Service é uma aplicação distribuída para *web*, cuja comunicação é feita principalmente por HTTP (Hyper Text Transfer Protocol), que disponibiliza seus componentes para serem utilizados em dispositivos diversos (KALIN, 2010). Kalin, afirma ainda que, uma das principais vantagens de um *web service* é a transparência de linguagem, ou seja, o cliente e os seus serviços não necessariamente precisam ser escritos em uma mesma linguagem para interagirem.

Segundo Paulino (PAULINO, 2008) um *WebService* é uma maneira de apresentar funcionalidade para usuários *web* através de protocolos da *Internet* padrão. Os WS *(Web Service)*

aplicam a arquitetura orientada a serviços (SOA - *Service Oriented Architecture*), porém, podem ser gerados componentes SOA com tecnologias que não sejam WS, pois os *(Web Service)* são 'instancias' dessa coleção de serviços.

Os WSs podem exercer a função de integrar sistemas independentes, via *web*, tornando acessíveis recursos de uma ou mais aplicações de forma definida (PISA, 2012).

De acordo com Lima (LIMA, 2012) há três papéis importantes incorporados a arquitetura de um WS, são eles: Provedor de Serviços, Consumidor de Serviços, e o Registro de Serviços. Abaixo são citadas a função de cada um desses papéis:

- Provedor de Serviços: Responsável pela execução e liberação dos *Web Services* na *Internet*, ou seja, pela publicação da definição de um determinado serviço. O serviço deve ser descrito em um formato padrão, de fácil compreensão e deve ter suas características publicadas em um registro central disponível.
- Consumidor de Serviços: É a pessoa que utiliza a descrição disponível na *Internet*, que foi disponibilizada por um provedor de serviços, para encontrar o serviço implementado.
- Registro de Serviços: Refere-se ao local aonde o serviço se encontra. Contém informações técnica dos serviços e os detalhes da empresa.

3.2.1 Arquitetura REST

O *REST* (*Representational State Transfer*), criado por um dos autores do protocolo *HTTP* (Dr. Roy Fielding, em 2000), é um modelo estruturado que possui as capacidades do protocolo como plataforma. Atualmente, essa técnica de engenharia pode ser usada para descrever qualquer *interface web*. Se destaca como competências usadas pelo *REST*, os distintos métodos de comunicação (*GET, POST, PUT, DELETE, HEAD, OPTIONS*) e a declaração de arquivos como recursos (cada um deles com seu próprio endereço) (GONCALVES, 2012).

Fielding buscou as práticas consideradas ótimas nos estilos de arquiteturas atuais para constituir um estilo novo, que reunisse todas elas, o qual ficou conhecido como *REST*. Este estilo de arquitetura é constituido, basicamente, por dois papéis: Cliente e Servidor. O cliente faz um requerimento para o servidor e o servidor disponibiliza serviços para que o cliente possa fazer uso (LIMA, 2012).

Conforme relata Laube (LAUBE, 2009), para construção de um *WebService* com base na técnica *REST* são necessários apenas: um cliente, um serviço, a informação, uma forma de "encapsular" essa informação (XML, JSON, etc.) e um meio para acessá-la (HTTP).

De acordo com (SAUDATE, 2012) a principal explicação para o êxito dessa técnica se dá pelos padrões e capacidades do protocolo *HTTP*, onde cada um dos métodos possui uma

função diferente. Saudate expõe ainda que esse modelo tão relacionado ao *HTTP* é organizado com base em alguns princípios que proporcinaram sucesso a própria *web*. Estes princípios são:

- *URLs* bem precisas para recursos, ou seja, cada recurso precisa ter uma *URL* bem definida. Outros parâmetros, que não estejam incluídos na definição dos recursos, podem ser passados como forma de "anexo" à *URL*. Essas seguem uma estrutura hierárquica, ou seja, o elemento que vem a seguir está relacionado com o elemento anterior.
- Uso dos métodos *HTTP* em conformidade com seus propósitos. Os *WebServices REST* podem ter métodos *HTTP* bem definidos para realização das operações. Através de diferentes métodos HTTP, pode-se tanto utilizar recursos existentes como criar novos.
- Uso de *headers HTTP* de forma efetiva. Os *headers HTTP* possuem a descrição ou as características a respeito daquilo que se está comunicando/recebendo do servidor.
- Uso de códigos de *status HTTP* para realizar a comunicação com os serviços e facilitar o reconhecimento do estado da requisição (se teve sucesso ou não).
- Uso de Hipermídia como motor de estado da aplicação. Ao se solicitar uma página *web* obtemos, em conjunto com o texto, diversos recursos adicionais: imagens, *scripts*, etc. Estes recursos geralmente não estão presentes no *HTML*, mas apenas as referências a eles. Roy Fielding supôs que seria interessante se cada revinda de recurso trouxesse *URLs* para novas operações, ao invés de conservar as *URLs* para recursos na própria aplicação.

4 SR Soft - Sistema de Recomendação de *Softwares* para Suporte de Computadores

Este capítulo descreve o modelo proposto neste trabalho. Será detalhada a tecnologia utilizada, assim como o funcionamento e a implementação do Sistema de Recomendação de *Softwares* para Suporte de Computadores, mostrando assim os resultados obtidos.

As atividades do Projeto de Trabalho de Conclusão de Curso Sistema de Recomendação de *Softwares* para Suporte de Computadores foram: o estudo sobre os sistemas de recomendação, onde foram analisadas as técnicas possíveis para realização das recomendações para a partir dessa análise escolher a que melhor se aplicaria ao projeto em desenvolvimento. Seguido do estudo sobre programação para *Android*, nessa fase houve a instalação dos *softwares* necessários, tais como: o SDK, que contém o emulador e todas as ferramentas necessárias para o desenvolvimento, a IDE (*Integrated Development Environment*) de Desenvolvimento *Eclipse*, as plataformas necessárias para o SDK, o plugin ADT (*Android Development Tools*), fornecido pelo *Google*, que permitiu criar projetos do tipo *Android* e testá-los no emulador. Após as instalações, foi criada uma configuração virtual (AVD), para que o emulador pudesse simular exatamente uma configuração de um celular real;

Os demais passos para a implementação do sistema foram: o desenvolvimento de aplicativos simples para o aprendizado na linguagem *JAVA* para *Android*, o desenvolvimento de diagramas com a simulação do funcionamento do sistema, a criação de um protótipo do sistema, através de *wireframes*, a criação de um Banco de Dados com *PostgreSQL* usando o *pgAdmin III*, a instação do *Apache2 Tomcat* para criação de um servidor local, estudo e implementação de um *WebService* e o desenvolvimento do Sistema de Recomendação de *Softwares* para Suporte de Computadores.

4.1 Instalação e Configuração

O SO usado para fornecer a gerência e a interação das tarefas no decorrer do desenvolvimento da aplicação foi o *Linux Ubuntu* 12.04 LTS 32-bit, *software* livre e que não precisa de programas anti-vírus. Foram necessárias a instalação de vários programas, bibliotecas e plugins para a implementação do sistema, assim como a configuração dos mesmos, citados abaixo:

- **Eclipse:** O IDE *Eclipse* versão *Juno* foi utilizado para implementação do aplicativo, usando como linguagem de desenvolvimento *JAVA* em ambiente *Linux*, seguindo o modelo *open source* de desenvolvimento de *software*.
- **SDK:** O *kit* de desenvolvimento para *Android* SKD versão 1.7.0, foi instalado e configurado para implementação de aplicativos.
- **ADT:** O plugin ADT (*Android Development Tools*) versão 22.0.5 foi instalado no *Eclipse*, para o desenvolvimento *Android*.
- **PostgreSQL 9.1.9:** A escolha desse SGBD (Sistema Gerenciador de Banco de Dados) se deu por ele ser um poderoso banco de dados, de código fonte aberto.
- **PgAdmin III:** Escolhido por ser uma ferramenta para administração do banco de dados *PostgreSQL* totalmente gratuita.
- **Apache 2.2:** Um dos mais usados e bem sucedidos servidores *web*, compatível com o protocolo HTTP, e um *software* gratuito.

Após a instalação da IDE *Eclipse* e do *kit* de desenvolvimento *SDK*, faz-se necessário dar permissão de acesso as pastas recentemente instaladas, observando sua localização, para que não ocorra o risco da classe R, classe gerada automaticamente na criação de um projeto *Android*, responsável pela comunicação do código JAVA com o XML, não ser gerada, impossibilitando assim, o correto funcionamento do *Android*. Essa permissão pode ser dada através do comando mostrado na figura 7.

```
chmod 777 CaminhoDaPasta -R
```

***Figura 7** – Comando para dar permissão a pasta.*

O AVD, responsável por carregar a imagem do sistema e simular o *hardware* e *software* de um sistema *Android*, foi configurado da seguinte forma: versão 2.2 (API *level* 8) do Sistema Operacional *Android*, *Screen* 4.0" (correspondente ao tamanho da tela do dispositivo), resolução(px) 350 X 600 e memória RAM (Random Access Memory) de 512 MB.

4.2 Modelagem do Sistema

Com o objetivo de se ter uma visão externa do sistema foi criado um Diagrama de Casos de Uso do *SR Soft* (Figura 8) para representar graficamente os atores, os casos de uso e os relacionamentos entre eles, ou seja, para descrever as principais funcionalidades do sistema do ponto de vista do usuário.

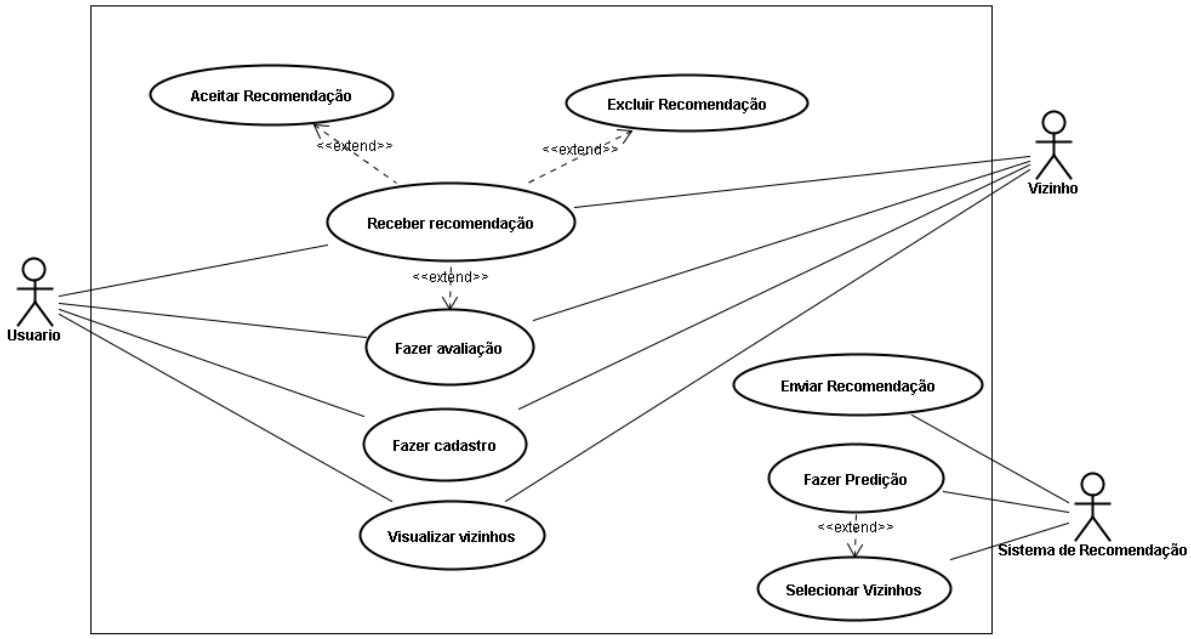

Figura 8 *– Diagrama de Casos de Uso do SR Soft.*

Descrição dos casos de uso do *SR Soft*:

- **Receber recomendação:**
  - **Ator:** Usuario, Vizinho.
  - **Comunicação 1:** O usuário tem a opção de aceitar ou excluir a recomendação.
  - **Comunicação 2:** O usuário tem a opção de o item que lhe foi recomendado.
  - **Descrição:** Responsável por receber as recomendações feitas pelos usuários.
- **Fazer avaliação:**
  - **Ator:** Usuario, Vizinho.
  - **Descrição:** Responsável por fazer a avaliação dos softwares cadastrados.
- **Fazer cadastro:**
  - **Ator:** Usuario, Vizinho.
  - **Descrição:** Responsável por fazer o cadastro dos dados do perfil do usuário.
- **Visualizar vizinhos:**
  - **Ator:** Usuario, Vizinho.
  - **Descrição:** Responsável por listar as informações dos usuários com perfil semelhante ao usuário ativo.

- **Enviar Recomendação:**
  - **Ator:** Sistema de Recomendação.
  - **Descrição:** Responsável por enviar as recomendações de acordo com as predições.
- **Fazer Predição:**
  - **Ator:** Sistemas de Recomendação.
  - **Comunicação:** Após o cálculo da predição, vizinhos poderão ser selecionados.
  - **Descrição:** Responsável por predizer a nota que o usuário dará para o item.
- **Selecionar Vizinhos:**
  - **Ator:** Sistemas de Recomendação.
  - **Descrição:** Responsável por avaliar a similaridade entre os usuários cadastrados.

4.3 Funcionamento do Sistema

O Sistema de Recomendação de *Softwares* para Suporte de Computadores foi desenvolvido para plataforma *Android* e atenderá usuários que desejem conhecer *softwares* que possam ser relevantes as suas necessidades e interesses, a partir de avaliações de outros usuários com características comuns.

O usuário ao ser cadastrado no sistema poderá fazer ou receber recomendações de *softwares* (edição de vídeos, manutenção técnica, redes de computadores, etc), assim como conhecer o perfil de outras pessoas (com quem tenha interesses em comum).

A Figura 9 mostra os *wireframes* do sistema, criados no início do projeto, como um protótipo visual do possível *layout* e funcionalidades da aplicação.

O sistema consiste em um aplicativo para fins de colaboração, onde os usuários cadastrados poderão interagir uns com os outros através da avaliação de *softwares*.

Figura 9 *– Wireframes para o Desenvolvimento do Aplicativo.*

Nesse sistema o usuário irá fazer seu cadastro especificando características importantes, para determinar seu perfil, que direcionarão a quais *softwares* ele poderá ter interesse (*wireframes* B, C e D) e de quais pessoas ele poderá receber recomendações. Após o cadastro, o usuário poderá se "logar" no sistema (*wireframe* A). Esse *login* é importante para que o mesmo possua uma identificação, tornando possível assim, avaliar a existência de perfis similares, para gerar uma ligação, onde outros usuários estarão vinculados a uma rede que pode ser denominada "vizinhança". Após "logado", o usuário terá um menu de opções (*Wireframe* F), onde poderá decidir por: verificar se há *softwares* recomendados (*wireframe* H), onde os "vizinhos" são responsáveis por fazer as avaliações dos *softwares* cadastrados e os que forem avaliados positivamente serão recomendados aos demais usuários vinculados; avaliar *softwares* cadastrados (*wireframe* G) atribuindo uma nota de 1 a 5 àquele item, de acordo com sua avaliação e essa servirá de recomendação para outros usuários "vizinhos"; conhecer o perfil de pessoas com características semelhantes as suas (*wireframe* I), de quem poderá receber ou fazer recomendações e; cadastrar *softwares* (*wireframe* E), abordando especificidades relevantes a análise do usuário, tais como: nome, área a que se aplica, *link* para *download* e descrição técnica. Essa última funcionalidade será implementada como trabalho futuro, inicialmente os *softwares* já estão cadastrados, aguardando as avaliações dos usuários.

Após instalar o aplicativo no dispositivo móvel, o usuário precisará se logar no sistema

para ter acesso as funcionalidades do aplicativo, se o mesmo ainda não tiver um *login*, precisará primeiramente fazer seu cadastro, para então ser admitido. A tela referente ao *login* (Figura 10) é a primeira tela do sistema, dando as opções para o usuário se logar ou realizar o cadastro. O código referente a implementação da classe *Login*, pode ser visto no Apêndice A.

Figura 10 *– Tela de Login.*

O usuário ao escolher a opção para fazer seu cadastro, na tela de *login*, será direcionado para tela seguinte (figura 11), onde deverá especificar o usuário e a senha que serão usados para a autenticação, assim como outros dados responsáveis pela formação do seu perfil. O código para implementação dessa funcionalidade de inserção dos dados cadastrados pode ser observado no Apêndice B.

Após realizar o cadastro dos dados pessoais, o usuário deverá especificar as áreas de interesse e domínio, para que outros usuários possam avaliar o perfil de quem recebeu recomendação. Ao total, o sistema possui três telas de cadastro, na primeira é exigida informações simples e relevantes para proporcionar a interação entre as pessoas cadastradas, tais como: nome, e-mail, telefone, e cidade onde reside, na segunda tela (Figura 15) o usuário poderá escolher áreas em que tenha o interesse de receber *softwares* e na terceira (Figura 16) poderá especificar áreas ao qual tenha experiência ou domínio de programas referentes a área citada. O intuito dessas três telas de cadastro é obter informações relevantes sobre as pessoas cadastradas que possam contribuir para a interação entre as mesmas, possibilitando para o usuário, não apenas a recomendação dos *softwares*, mas o conhecimento dos interesses e domínios de cada

um. Os dados para contato torna possível a colaboração entre eles.

Figura 11 *– Tela de Cadastro - Dados Pessoais.*

As informações inseridas pelo usuário são enviadas para o servidor para serem armazenadas em um banco de dados. Essa interação se dá através do uso de *Web Service*, onde o cliente faz uma requisição HTTP simples e o servidor *Web Service* faz um retorno com os dados processados. Na classe *Cadastro1REST* (Figura 12) pode-se observar a conexão do cliente com o servidor através da requisição via *URL*, onde coloca-se o nome do servidor criado, e o nome da classe que irá conectar o cliente ao *Web Service* (Figura 13).

```
public class Cadastro1REST {

    private static final String URL_WS = "http://10.0.2.2:8080/ProjetoSR-WS/cadastro1/";

       public Cadastro1 getCadastro1(String id) throws Exception {

        String[] resposta = new WebServiceUsuario().get(URL_WS + id);

        if (resposta[0].equals("200")) {
            Gson gson = new Gson();
            Cadastro1 cadastro1 = gson.fromJson(resposta[1], Cadastro1.class);
            return cadastro1;
        } else {
            throw new Exception(resposta[1]);
        }
       }
```

Figura 12 *– Classe Cadastro1REST - getCadastro1.*

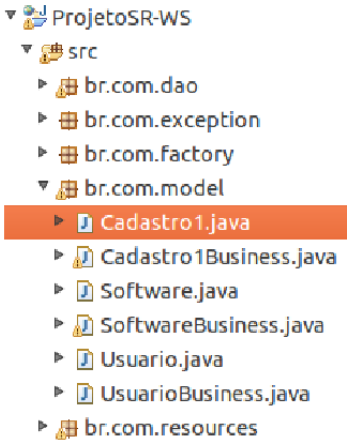

Figura 13 *– Web Service - ProjetoSR-WS.*

A Figura 14 mostra a interação da aplicação com o *Web Service* no quesito armazenar os dados do perfil do usuário.

```
//********************** INSERIR ****************************************
public String inserirCadastro1(Cadastro1 cadastro1) throws Exception {

 Gson gson = new Gson();
 String cadastro1JSON = gson.toJson(cadastro1);
 String[] resposta = new WebServiceUsuario().post(URL_WS + "inserir", cadastro1JSON);
 if (resposta[0].equals("200")) {
     return resposta[1];
 } else {
     throw new Exception(resposta[1]);
 }
}
```

Figura 14 *– Classe Cadastro1REST - inserirCadastro1.*

Após a realização do cadastro, o usuário poderá acessar a aplicação, logo após a verificação do usuário e da senha. Depois da tela de *login*, aparecerá para o utente um menu de opções (Figura 17) com as funcionalidades oferecidas pelo sistema e esse poderá escolher a próxima ação a ser executada. As alternativas disponíveis são: Conhecer Vizinhos, Softwares Recomendados, Avaliar Item e Sair. O código do menu pode ser visto no Apêndice C.

Figura 15 *– Cadastro - Interesses do usuário.*

Figura 16 *– Cadastro - Domínio do usuário.*

No quadro “Conhecer Vizinhos” será listado o perfil das pessoas cadastradas no sistema que possuem interesses similares aos do usuário em questão, como pode-se observar na Figura 18. Através dessa listagem de perfis é possível conhecer as pessoas que possuem características em comum, pela área de interesse e domínio cadastrada, e também facilitar a comunicação entre os “vizinhos”, pelas informações de contato presentes no perfil listado.

Figura 17 *– Tela de Opções.*

Para calcular a similaridade entre os usuários, foi utilizado o Coeficiente de Correlação Linear de Pearson, para determinar se a correlação entre os mesmos é positiva, negativa ou se não há correlação entre eles. O algorítmo do cálculo desse coeficiente de correlação está presente no Apêndice D. É possível distinguir os usuários com gostos semelhantes, através de resultados que variam entre -1 (forte relação negativa) e 1 (forte relação positiva), sendo que o resultado 0 equivale a ausência de relação entre os dados comparados.

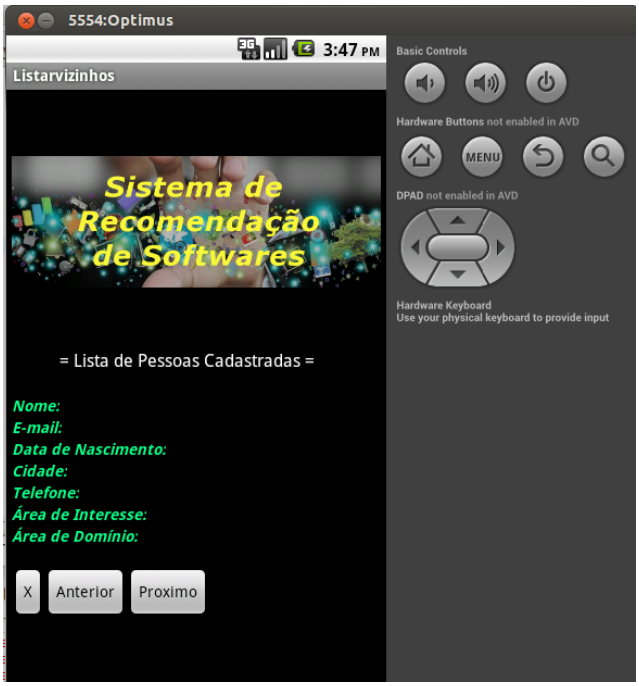

Figura 18 *– Tela de Pessoas Cadastradas.*

A Tabela 1 mostra os valores atribuidos ao resultado do coeficiente de correlação. A partir do resultado 0.5 o usuário será considerado "vizinho", tendo ele correlação média ou alta. Raramente encontra-se resultados que constem forte relação positiva (0.9 ou 1), devido a variedade de itens e tendências de cada pessoa.

Resultado do Coeficiente	**Intensidade da Correlação**
-1 a -0.9	Forte Correlação Negativa
0.8 a 0.5	Média Correlação Negativa
0.4 a 0.1	Baixa Correlação Negativa
0	Não há Correlação
0.1 a 0.4	Baixa Correlação Positiva
0.5 a 0.8	Média Correlação Positiva (vizinho)
0.9 a 1	Alta Correlação Positiva (vizinho)

Tabela 1 *– Grau de Correlação entre Variáveis*

A Tabela 3 (Apêndice E) apresenta o resultado da similaridade do teste de correlação feito com vinte e seis (26) alunos do 3º ao 8º período, do Curso Bacharelado em Sistemas de

Informação da Universidade Federal do Piauí, Campus Senador Helvídio Nunes de Barros. O teste consistiu em analisar a similaridade entre um usuário, nomidado "A" e os demais usuários apresentados, com base nas notas que os mesmos deram à quinze (15) *softwares* relacionados a área de programação, manutenção e multimídia. O resultado da similaridade entre os usuários foi obtida através do cálculo do Coeficiente de Correlação Linear de Pearson. Como pode-se observar na tabela citada, não houve resultados de coeficiente com valor acima de 0.6 e dos vinte e cinco (25) usuários comparados com o usuário "A", apenas quatro (4) foram considerados "vizinhos" ("B", "D", "J" e "Y").

Para o cálculo desse coeficiente são necessários dois vetores dispostos com notas de determinados usuários a para programas específicos. Através da comparação entre a quantidade de avaliações com valores próximos ou análogos, pode-se perceber a semelhança entre o gosto de certos usuários, determinando assim a similaridade entre os usuários cadastrados.

O algorítmo que calcula o Coeficiente de Correlação Linear de Pearson está presente no Apêncide C. Na *class* Formula(), podemos observar o método *getPearson()* que retorna o resultado final do coeficiente, assim como os métodos necessários para construção da fórmula: *N()*, retorna o total de números presentes no vetor; *Sx()*, calcula o somatório dos números presentes no primeiro vetor; *Sy()*, calcula o somatório dos números do segundo vetor; *SPxx()*, calcula o somatório dos números do primeiro vetor ao quadrado; *SPyy()*, calcula o somatório dos números do segundo vetor ao quadrado; *SPxy()*, calcula o somatório do produto dos números do vetor 1 pelos do vetor 2; *setArray()*, corresponde a declaração dos vetores que serão usados na fórmula.

Na opção "Softwares Recomendados" será listado os *softwares* que foram avaliados positivamente pelos vizinhos do usuário, ou seja, que tiveram nota 4 ou 5. A quantidade de estrelas marcadas corresponderá a nota que o usuário deu ao item, representando assim o valor atribuído a classificação de acordo com a Tabela 2:

Número de Estrelas	**Valor da Marcação**
1 estrela	Péssimo
2 estrelas	Ruim
3 estrelas	Regular
4 estrelas	Bom
5 estrelas	Excelente

***Tabela 2** – Tabela de Valores da Classificação dos Usuários*

O cálculo da predição prevê que um determinado *software* poderá ser avaliado positivamente pelo usuário. O método de recomendação usado para calcular a predição foi o *SlopeOne*, que segundo Souza (SOUZA, 2012) apesar de ser de fácil implementação, é uma abordagem eficiente, escalável e que apresenta bons resultados práticos.

No quadro "Avaliar Item" (figura 19), são mostrados todos os *softwares* cadastrados

no sistema que aguardam a avaliação do usuário, o mesmo visualizará as informações de um *software* por vez, podendo passar para o item seguinte ou retornar ao anterior quando achar necessário. Para realizar a avaliação, o usuário deverá marcar o número de estrelas correspondente a sua análise e então clicar no botão *Avaliar*. Essa nota que cada usuário atribui aos *softwares* cadastrados, será utilizada tanto no cálculo do Coeficiente de Correlação Linear de Pearson, para avaliar a similaridade dos usuários, como no cálculo *SlopeOne*, para predizer qual nota o usuário daria para determinado software com base nas notas dadas pelos "vizinhos", e então gerar as recomendações.

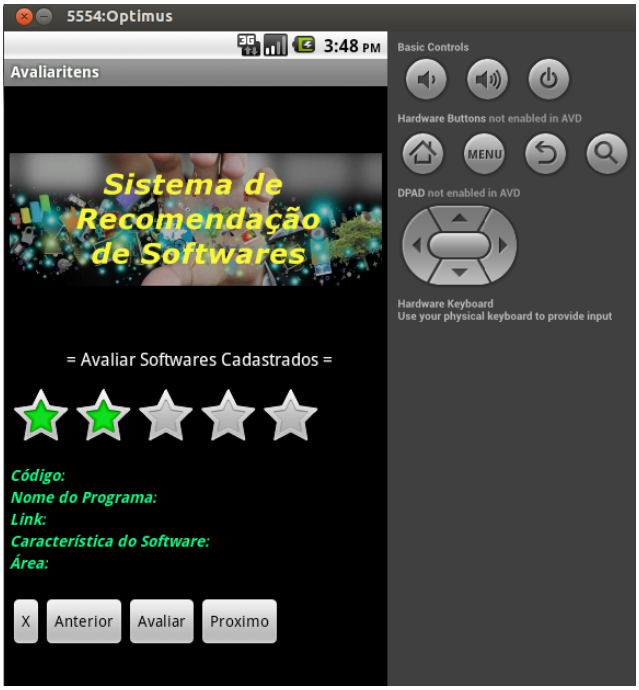

Figura 19 *– Tela de Avaliação dos Softwares.*

5 Conclusões e Trabalhos Futuros

5.1 Conclusões

O trabalho proposto foi desenvolvido com o objetivo de apresentar o Sistema de Recomendação de *Softwares* para Suporte de Computadores, nomeado de *SR Soft*. O sistema foi desenvolvido para dispositivos móveis com plataforma *Android*, voltado para discentes e docentes do curso Sistemas de Informação e áreas afins, podendo ser usado por pessoas de diferentes áreas com interesse em *softwares* de programação, manutenção, entre outros.

Com o aplicativo *Sr Soft* é possível oferecer aos usuários da área de Tecnologia da Informação a recomendação de *softwares* que possivelmente sejam do seu interesse e a possibilidade de avaliar itens que poderão ser recomendados a pessoas com perfil semelhante, proporcionando a colaboração entre os usuários.

Com o desenvolvimento do Sistema de Recomendação de *Softwares* para Suporte de Computadores foi possível a utilização de *WebService* como uma solução para estabelecer a comunicação da aplicação com o banco de dados. O uso dessa tecnologia se faz interessante quando uma aplicação necessitar requisitar tarefas de outra aplicação que não use a mesma linguagem de programação ou que esteja em sistemas diferentes.

O WebService foi construído inspirado nos princípios da arquitetura REST, onde, em geral, usa-se as *URLs* (*Uniform Resource Locator*) para o acesso de recursos. Utilizando o REST pode-se ter um serviço implementado de maneira simples para a criação de um *WebService*.

O *Android* possui um SGBD nativo da plataforma, o *SQLite*, que foi usado inicialmente na implementação da aplicação e depois substituído pelo *PostgreSQL*. Apesar de ser uma base de dados leve e poderosa, o SQLite, armazena dados localmente, possuindo um limite de 2TB para esse armazenamento, além de não ser recomendado em situações onde o banco de dados poderá ser acessado ao mesmo tempo em diversos dispositivos por meio de um sistema de arquivos de redes. O *PostgreSQL* por sua vez, é um SGBD com armazenamento de dados ilimitado (AGUILAR, 2010) que atendeu bem aos requisitos esperados pela aplicação.

Não foi conseguido sanar o problema de trazer os dados para os cálculos de correlação e predição, sendo esses inseridos para realização dos testes.

5.2 Trabalhos Futuros

Atualmente as recomendações realizadas pelo *SR Soft* são feitas utilizando apenas Filtragem Colaborativa, por atender bem aos resultados esperados com base na proposta.

Para trabalhos futuros, são os novos objetivos a serem realizados: (a) inserir a opção do usuário cadastrar os *softwares*, dando aos mesmos não apenas a opção de avaliar e receber recomendações mas também de inserir novos itens para serem avaliados e recomendados; (b) inserir novos critérios para avaliação da similaridade dos usuários, analisando além do perfil de notas semelhantes atribuidas pelos mesmos, o seu perfil pessoal de interesses; e (c) prover todas as informações diretamente do servidor para geração das recomendações.

Referências

AGUILAR, Emílio Hernandez. *Banco de Dados PostgreSQL*. In: . Centro Paula Souza - Competência em Educação Pública Profissional, 2010. Disponível em: <http://gilbertexbom.com/bd2/2InfoN_110/postgresql.pdf>.

ALBUQUERQUE, Fernando M. Figueira Filho Paulo Lício de Geus; João Porto de. *Sistemas de Recomendação e Interação na Web Social*. I Workshop de Aspectos da Interação Humano-Computador na Web Social, 2008. ISBN 978-85-7669-213-3. Disponível em: <http://www.ic.unicamp.br/~fmarques/papers/websocial_ihc08.pdf>. Acesso em: 04 mai 2013.

BARBOSA, Sílvio César Cazella; Eliseo Berni Reategui; Munique Machado; Jorge Luis V. *Recomendação de Objetos de Aprendizagem Empregando Filtragem Colaborativa*. XX Simpósio Brasileiro de Informática na Educação, Universidade do Vale do Rio dos Sinos - UNISINOS, 2009. Disponível em: <http://ceie-sbc.educacao.ws/pub/index.php/sbie/article/download/1158/1061>. Acesso em: 04 mai 2012.

BARBOSA, Sílvio César Cazella; Irismar Corrêa das Chagas; Jorge Luis V. *Um Modelo para Recomendação de Artigos Acadêmicos Baseado em Filtragem Colaborativa Aplicado a Ambientes Móveis*. CINTED-UFRGS, Dezembro 2008.

BARCELLOS, Daniela Leal Musa; André Luiz Brandão; Mariusa Warpechowski; Carla Duarte. *Sistema de Recomendação Acadêmico para Apoio a Aprendizagem*. CINTED-UFRGS, 2007. Disponível em: <http://seer.ufrgs.br/renote/article/view/14236>. Acesso em: 12 abr 12.

BENITEZ, Jorge Alberto Castro. *Vinheta Histórica Karl Pearson Sesquicentenário de seu Nascimento*. *VITALLE*, Rio Grande/RS-Brasil, v. 19, n. 2, p. 7–9, 2007.

BEZERRA, Byron Leite Dantas. *Uma Solução em Filtragem de Informação para Sistemas de Recomendação Baseada em Análise de Dados Simbólicos*. Dissertação (Mestrado), 2004. Disponível em: <http://www.liber.ufpe.br/teses/arquivo/20040930134807.pdf>.

CARDONA, Sidnei Renato Silveira: Márcio A. *SRISA: Desenvolvimento de um Sistema de Recomendação para Instalação de Som Automotivo*. V WET Workshop de Engenharia e Tecnologia - IV CCTEC Congresso de Ciência e Tecnologia do Vale do Taquari, 2010. Disponível em: <http://ensino.univates.br/~cetec/wet/anais2010/C07-wet2010.pdf>. Acesso em: 18 jul 2012.

CASAROTTO, Alberto Cargnelutti Filho; Sidinei José Lopes; Betânia Brum; Marcos Toebe; Tatiani Reis da Silveira; Gabriele. *Sample size to estimate the Pearson correlation coefficient among characters of castor bean*. Semina: Ciências Agrárias, Londrina, junho 2012. Disponível em: <http://www.uel.br/revistas/uel/index.php/semagrarias/article/download/6638/10821>. Acesso em: 10 ago 2013.

CAZELLA, Silvio César; Jonas Vinicius Drumm; Jorge Luis V. Barbosa. *Um Serviço de para Recomendação de Artigos Científicos Baseado em Filtragem de Conteúdo Aplicado a Dispositivos Móveis*. V.8 nº 3. Universidade Federal do Rio Grande do Sul: Novas Tecnologias na Educação, 1998.

CRUZ, Bruno Henrique Andrade; Josué Fernandes Dall Agnese; Bruno José Fagundes; Marcelo Teixeira Bastos; Rolf Fred Molz; Jacques Nelson Corleta Schreiber. *Desenvolvimento de uma Aplicação Embarcada em Celular Visando Controle de Robô Via Wi-Fi*. Revista brasileira de computação aplicada (issn 2176-6649). v. 3, n. 1, p. 43-52. DOI: 10.5335/rbca.2011.005: [s.n.], 2011.

FILHO, José Alexandre da Silva Júnior; Dalson Britto Figueiredo. *Desvendando os Mistérios do Coeficiente de Correlação de Pearson (r)*. Revista Política Hoje, 2009. Disponível em: <http://www.revista.ufpe.br/politicahoje/index.php/politica/article/viewFile/6/6>. Acesso em: 13 ago 2013.

FILIPPETTO, Alexsandro ; Giovane Barcelos; Marcelo Batista; Clovis da Silveira. *Grupos de Estudos Baseados em Ferramentas Colaborativas*. Revista iTEC, 2011. Disponível em: <http://www.facos.edu.br/old/index.php?option=com_content&view=article&id=1327&Itemid=377>. Acesso em: 27 mar 12.

GEROSA, Marco Aurélio. *CWTools (Collaborative Web Tools)*: Componentes de software para interação social e inteligência coletiva. RPN, 2009. Disponível em: <http://www.rnp.br/pd/gts2009-2010/gt_cwtools.html>.

GONCALVES, Ricardo Frenedoso Da Silva; Pablo Rodrigo. *Web Services*: Uma análise comparativa. Janeiro/Dezembro 2012.

ISOTANI, Seiji. *Sistemas Colaborativos, Fundamentos e Aplicações*. Laboratório de Engenharia de Software, Departamento de Sistemas de Informação, Universidade de São Paulo, 2012. Disponível em: <http://disciplinas.stoa.usp.br/pluginfile.php/24164/mod_resource/content/1/SISTEMAS%20COLABORATIVOS.pdf>.

KALIN, Martin. *Java Web Services*: Implementando. Rio de Janeiro: [s.n.], 2010. Disponível em: <http://www.altabooks.com.br, publisher=Alta Books>.

LAUBE, Klaus Peter. *Webservices: Conhecendo o REST*. PTI, 2009. Disponível em: <http://www.profissionaisti.com.br/2009/03/webservices-conhecendo-o-rest/>. Acesso em: 16 ago 2013.

LIMA, Jean Carlos Rosário. *Web Services (SOAP X REST)*. Faculdade de Tecnologia de São Paulo, 2012. Disponível em: <http://www.fatecsp.br/dti/tcc/tcc00056.pdf>.

LIRA, Sachiko Araki. *Análise de Correlação:*: Abordagem teórica e de construção dos coeficientes com aplicações. 196 f. Dissertação (Dissertação apresentada ao Curso de Pós-Graduação em Métodos Numéricos em Engenharia dos Setores de Ciências Exatas e de Tecnologia, como requisito parcial à obtenção do Grau de "Mestre em Ciências") — Universidade Federal do Paraná, Curitiba, 2004.

LOPES, Giseli Rabelo. *Sistema de Recomendação para Bibliotecas Digitais Sob a Perspectiva da Web Semântica*. 69 f. Dissertação (Programa de Pós-Graduação em Computação) — Universidade Federal do Rio Grande do Sul, Porto Alegre, 2007.

LOPES, Sérgio. *A Web Mobile*: Programe para um mundo de muitos dispositivos. [S.l.]: Casa do Código, 2013.

LUCAS, André de Trigueiros Pinção. *Recomendação de Programas de Televisão*. 90 p. Dissertação (Programa de Pós-Graduação em Engenharia Informática e de Computadores) — Universidade Técnica de Lisboa, 2010.

MACK, Roger Schneider. *Sistema de recomendação baseado na localização e perfil utilizando a plataforma android*. In: . Universidade Federal do Rio Grande do Sul, 2010. Disponível em: <http://www.lume.ufrgs.br/bitstream/handle/10183/28328/000767836.pdf?sequence=1>.

MACLACHLAN, Daniel Lemire; Anna. Slope one predictors for online rating-based collaborative filtering. In: *Proceedings of SIAM Data Mining (SDM'05)*. [s.n.], 2005. Disponível em: <http://www.daniel-lemire.com/fr/documents/publications/lemiremaclachlan_sdm05.pdf>.

MENDEZ, Sidnei Renato Silveira; Daniel Loureiro. *SRFit -Sistema Inteligente para apoio à Recomendação de Treinos Físicos*. Faculdade de Informática – Centro Universitário Ritter dos Reis (UniRitter), 2010. Disponível em: <http://ensino.univates.br/~cetec/wet/anais2010/C06-wet2010.pdf>. Acesso em: 04 mai 2012.

NODARI, Antônio Régis. *Os Sistemas de Recomendação como Instrumento para Atingir Mercados de Nicho*. In: . Universidade de Caxias do Sul, 2008. Disponível em: <http://www.ucs.br/ucs/tplPOSAdministracao/posgraduacao/strictosensu/administracao/dissertacoes/dissertacao?identificador=212>.

PAULINO, André Luiz da Silva. *Teste Baseado em Defeitos para Web Services*. Dissertação (Programa de Pós-Graduação em Informática) — Universidade Federal do Paraná, 2008.

PEREIRA, Pedro Romão. *Sistema de Recomendação para Condutores de Veículos Elétricos*. 104 p. Dissertação (Dissertação de natureza científica realizada para obtenção do grau de Mestre em Engenharia Informática e de Computadores) — Instituto Superior de Engenharia de Lisboa, Área Departamental de Engenharia de Eletrônica e Telecomunicações e de Computadores, 2010.

PISA, Rodrigo Dantas da Silva; Felipe Renó Oliveira. *WebGraphs - Plataforma para armazenamento e execução de algoritmos sobre grafos através da Web* . In: . Universidade Federal do Paraná, 2012. Disponível em: <http://www.inf.ufpr.br/andre/files/PisaSilva2012.pdf>.

SANTOS, Romenigue Mendes Barbosa Vieira dos. *Recomendação de Notícias*. 85 f. Dissertação (Dissertação para obtenção do Grau de Mestre em Engenharia Informática e de Computadores) — Universidade Técnica de Lisboa, Lisboa, 2012.

SAUDATE, Alexandre. *SOA aplicado*: Integrando com web service e além. São Paulo: Casa do Código, 2012. ISBN 9788575221846.

SILVA, Andréia Michelle da Cunha de Noronha; Érica Rossana Pinto Correia; Thiago Pereira Nunes e. *Jogo para Aplicativos Móveis Utilizando o Android*. In: . Brasília - DF: Universidade Católica de Brasília, 2010. Disponível em: <http://www.trabalhosfeitos.com/ensaios/Android/237389.html>.

SOBRAL, Daniela Barreiro Claro; João Bosco Mangueira. *Programação em Java*. Florianópolis, SC: Copyleft Pearson Education, 2008.

SOUZA, Renata Ghisloti Duarte de. *Sistemas de Recomendação*: Aplicando sistemas de recomendação em situações práticas. IBM, 2012. Disponível em: <http://www.ibm.com/developerworks/br/local/data/sistemas_recomendacao/>.

STANTON, Jeffrey M. *Galton, Pearson, and the peas:* : A brief history of linear regression for statistics instructors. Journal of Statistical Education, 2001. Disponível em: <http://www.amstat.org/publications/JSE/v9n3/stanton.html>. Acesso em: 14 ago 2013.

APÊNDICE A – Classe Login

Essa classe tem a função de verificar se o usuário e a senha digitados conferem com os referidos dados cadastrados no Banco.

Figura 20 *– Código - class Login.*

```
13 public class Login extends Activity {
14
15    protected void onCreate(Bundle savedInstanceState) {
16        super.onCreate(savedInstanceState);
17        setContentView(R.layout.activity_logininnovo);
18
19        final EditText editLogin = (EditText) findViewById(R.id.edUsuario);
20        final EditText editSenha = (EditText) findViewById(R.id.edSenha);
21        final Button btnLogar = (Button) findViewById(R.id.btLogar);
22
23        btnLogar.setOnClickListener(new View.OnClickListener() {
24            public void onClick(View v) {
25                String login = editLogin.getText().toString();
26                String senha = editSenha.getText().toString();
27
28                if(login.equals("") && senha.equals("")){
29                    gerarToast("Este campo não pode ser vazio!");
30                } else{
31                UsuarioREST userREST = new UsuarioREST();
32
33                try{
34                    Usuario usuario = userREST.getUsuario(login);
35                    if(login.equals(usuario.getLogin()) && senha.equals(usuario.getSenha())){
36                        Intent i = new Intent (Login.this, Opcoes.class);
37                        startActivityForResult(i, 1);
38                    } else {
39                        gerarToast("Senha Incorreta!");}
40                    } catch (Exception e) {
41                        e.printStackTrace();
42                        gerarToast(e.getMessage());
43                    }}
44            }});
```

APÊNDICE B – Classe Cadastro

A classe Cadastroperfil1 é responsável por obter os dados digitados nos *EditText* pertinentes ao perfil simplificado do usuário (Nome, E-mail, Cidade, etc.) e mandar para o Banco de Dados, com o uso de *Webservice*.

Figura 21 *– Código - class Cadastroperfil1.*

```
public class Cadastroperfil1 extends Activity {

        protected void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.activity_cadastroperfil1);

                final Button btcadastrar = (Button) findViewById(R.id.btCadastro);
                final EditText edNome = (EditText) findViewById(R.id.editNome);
                final EditText edLogin = (EditText) findViewById(R.id.editLogin);
                final EditText edSenha = (EditText) findViewById(R.id.editSenha);
                final EditText edEmail = (EditText) findViewById(R.id.editEmail);
                final EditText edDataNascimento = (EditText) findViewById(R.id.editDatanascimento);
                final EditText edCidade = (EditText) findViewById(R.id.editCidade);
                final EditText edTelefone = (EditText) findViewById(R.id.editTelefone);

        //Botão voltar para cadastrar -- insere informações no BD e passa para a próxima tela
                btcadastrar.setOnClickListener(new View.OnClickListener() {
                        public void onClick(View v) {
                        ArrayList<Cadastro1> listaCadastro1 = new ArrayList<Cadastro1>();

                                String nome = edNome.getText().toString();
                                String login = edLogin.getText().toString();
                                String senha = edSenha.getText().toString();
                                String email = edEmail.getText().toString();
                                String dataNasc = edDataNascimento.getText().toString();
                                String cidade = edCidade.getText().toString();
                                String fone = edTelefone.getText().toString();
                                Cadastro1 cad1 = new Cadastro1();
                                cad1.setNome(nome);
                                cad1.setLogin(login);
                                cad1.setSenha(senha);
                                cad1.setEmail(email);
                                cad1.setData_nascimento(dataNasc);
                                cad1.setCidade(cidade);
                                cad1.setTelefone(fone);
                                listaCadastro1.add(cad1);
                                Cadastro1REST cad1REST = new Cadastro1REST();

                                try{
                                        String resposta = cad1REST.inserirListaCadastro1(listaCadastro1);
                                        gerarToast(resposta);
                                        Intent i = new Intent (Cadastroperfil1.this, Cadastroperfil2.class);
                                        startActivityForResult(i, 1);

                                } catch (Exception e){
                                        gerarToast("Cadastro não efetuado!");
                                }}});
```

APÊNDICE C – Algoritmo Menu de Opções do Aplicativo SR Soft

Figura 22 *– Código - Menu de Opções (parte 1).*

```
1  package br.com.android;
2
3  import android.os.Bundle;
9
10 public class Opcoes extends Activity {
11
12     protected void onCreate(Bundle savedInstanceState) {
13         super.onCreate(savedInstanceState);
14         setContentView(R.layout.activity_opcoes);
15
16 // Button List Peoples ************************************************
17         Button btListarpessoas = (Button) findViewById(R.id.btlistpess);
18
19         btListarpessoas.setOnClickListener(new View.OnClickListener() {
20
21             public void onClick(View v) {
22                 Intent i = new Intent (Opcoes.this, Listarvizinhos.class);
23                 startActivityForResult(i, 1);
24             }
25         });
26
27 // Button Assessing Softwares *****************************************
28         Button btavaliaritens = (Button) findViewById(R.id.btavalitem);
29
30         btavaliaritens.setOnClickListener(new View.OnClickListener() {
31
32             public void onClick(View v) {
33                 Intent i = new Intent (Opcoes.this, Avaliaritens.class);
34                 startActivityForResult(i, 1);
35             }
36         });
37
```

Figura 23 – *Código - Menu de Opções (parte 2).*

```
39
40  // Button Recommended Softwares ***********************************************
41          Button btsoftrec = (Button) findViewById(R.id.btsoftrec);
42
43          btsoftrec.setOnClickListener(new View.OnClickListener() {
44
45              public void onClick(View v) {
46                  Intent i = new Intent (Opcoes.this, SoftRecomendado.class);
47                  startActivityForResult(i, 1);
48              }
49          });
50
51  // Button Exit ****************************************************************
52          Button btsair = (Button) findViewById(R.id.sair);
53          btsair.setOnClickListener(new View.OnClickListener() {
54
55              @Override
56              public void onClick(View v) {
57                  Intent intent = new Intent(Intent.ACTION_MAIN);
58                  intent.addCategory(Intent.CATEGORY_HOME);
59                  intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
60                  startActivity(intent);
61              }
62          });
63  }
64
65      public boolean onCreateOptionsMenu(Menu menu) {
66          getMenuInflater().inflate(R.menu.opcoes, menu);
67          return true;
68      }
69  }
```

APÊNDICE D – Algoritmo Coeficiente de Correlação Linear de Pearson

A classe Formula, recebe as notas atribuidas aos vetores referentes as avaliações de dois usuários para calcular o Coeficiente de Correlação Linear de Pearson.

Figura 24 *– Código - Algorítmo Coeficiente de Correlação de Pearson (parte 1).*

```
5  public class Formula {
6
7      private double array1 [];
8      private double array2 [];
9
10 //***** PEARSON CORRELATION COEFFICIENT *******************************************************
11     public double getPearson(){
12
13         double formula;
14          formula = (N()*SPxy()-Sx()*Sy()) / (Math.sqrt( (N()*SPxx()-Sx()*Sx()) * (N()*SPyy()-Sy()*Sy())));
15
16          DecimalFormatSymbols dfs = new DecimalFormatSymbols();//Number of decimal places
17          dfs.setDecimalSeparator('.');
18          DecimalFormat dformt = new DecimalFormat("0.0", dfs);
19
20          return formula;
21     }
22
23 //***** QUANTITY NUMBER OF VECTOR ***************************************************************
24     public double N(){
25         double QntNumeros = (array1.length);
26         return QntNumeros;
27     }
28
29 //***** CALCULATE SUM array1 (Sx) ***************************************************************
30     public double Sx(){
31         double total = 0;
32         for(int i=0; i < array1.length; i++){
33             total += array1[i];
34         }
35         return total;
36     }
```

Figura 25 *– Código - Algorítmo Coeficiente de Correlação de Pearson (parte 2).*

```
41  //***** CALCULATE SUM array2 (Sy) ************************************************************
42      public double Sy(){
43          double total = 0;
44          for(int i=0; i < array2.length; i++){
45              total += array2[i];
46          }
47          return total;
48      }
49
50  //***** CALCULATE SUM PRODUCT array1 THE SQUARE (SPxx) ****************************************
51      public double SPxx(){
52          double total = 0;
53          for (int counter=0; counter < array1.length; counter ++){
54              total += Math.pow(array1[counter], 2);
55          }
56          return total;
57      }
58
59  //***** CALCULATE SUM PRODUCT array2 THE SQUARE (SPyy) ****************************************
60      public double SPyy(){
61          double total = 0;
62          for (int counter=0; counter < array2.length; counter ++){
63              total += Math.pow(array2[counter], 2);
64          }
65          return total;
66      }
```

Figura 26 *– Código - Algorítmo Coeficiente de Correlação de Pearson (parte 3).*

```
65  //***** CALCULATING THE SUM OF PRODUCT array1 Array2 (SPxy) ************************
66      public double SPxy(){
67          double total = 0;
68          for (int i=0, j=0; (i < array1.length) && (j < array2.length) ; i++, j++){
69              total += array1[i]* array2[j];
70          }
71          return total;
72      }
73
74  //*********************************************************************************
75  // Declaration of vectors
76      public void setArray(double[] array1, double[] array2) {
77
78          this.array1 = array1;
79          this.array2 = array2;
80      }
81  }
```

APÊNDICE E – Teste do Algorítmo Coeficiente de Correlação Linear de Pearson

Usuários Comparados	Resultado do Coeficiente	Situação da Similaridade
A - B	0.59	Média Correlação Positiva (vizinho)
A - C	0.27	Baixa Correlação Positiva
A - D	0.52	Média Correlação Positiva (vizinho)
A - E	0.49	Baixa Correlação Positiva
A - F	0.37	Baixa Correlação Positiva
A - G	0.06	Baixa Correlação Positiva
A - H	0.39	Baixa Correlação Positiva
A - I	0.43	Baixa Correlação Positiva
A - J	0.51	Média Correlação Positiva (vizinho)
A - K	0.04	Baixa Correlação Positiva
A - L	0.49	Baixa Correlação Positiva
A - M	0.23	Baixa Correlação Positiva
A - N	- 0.01	Correlação Negativa
A - O	0.11	Baixa Correlação Positiva
A - P	0.25	Baixa Correlação Positiva
A - Q	0.30	Baixa Correlação Positiva
A - R	- 0.26	Correlação Negativa
A - S	0.22	Baixa Correlação Positiva
A - T	0.29	Baixa Correlação Positiva
A - U	0.22	Baixa Correlação Positiva
A - V	0.27	Baixa Correlação Positiva
A - W	0.33	Baixa Correlação Positiva
A - X	0.17	Baixa Correlação Positiva
A - Y	0.53	Média Correlação Positiva (vizinho)
A - Z	0.44	Baixa Correlação Positiva

Tabela 3 *– Teste de Similaridade entre Usuários*