SYSCOMA, a System for Competence Management

João Marco Maciel da Silva[1], Valdemar W. Setzer (Orientador)[1]

[1]IME-USP - Instituto de Matemática e Estatística da Universidade de São Paulo,
Departamento de Ciência da Computação

Introdução

SYSCOMA é um projeto *open-source* baseado na Internet para o gerenciamento de competências.

Motivação

Em grandes empresas, com quadro de centenas de funcionários, é comum o surgimento da necessidade de se encontrar um funcionário conhecedor de determinado assunto ou especialista no mesmo. O *modus operandi* de tais empresas que não possuem um *software* específico que poderia auxiliá-las nessa tarefa é, em geral, a análise de currículos impressos e conhecimento pessoal. O sistema aqui proposto não visa substituir totalmente o julgamento humano, mas auxiliá-lo restringindo o número de pessoas que devem ser examinadas com mais detalhe.

Dado, Informação, Conhecimento e Competência

Este projeto é baseado em [1], onde é definido o que são e como funcionam matrizes de competência e de onde são tiradas a maior parte das definições e caracterizações a seguir.

- **Dado** é "uma seqüência de símbolos quantificados ou quantificáveis".
- **Informação** "é uma abstração informal (isto é, não pode ser formalizada através de uma teoria lógica ou matemática), que está na mente de alguém, representando algo significativo para essa pessoa."
- **Conhecimento** é uma "abstração interior, pessoal, de alguma coisa que foi experimentada por alguém."
- **Área de conhecimento** indica uma área da atuação humana, para a qual se aplicam certas habilidades em sua execução.
- **Competência** é uma "capacidade de executar uma tarefa no 'mundo real'"e é caracterizada "como a confluência de uma habilidade em uma área de conhecimento, com a respectiva representação matricial".
- **Matriz de Competência** é o agrupamento em forma de matriz das áreas de conhecimento (representadas nas linhas) que possuem as mesmas habilidades (colunas). Cada pessoa da empresa tem uma ou mais matrizes de competência. Em cada célula de uma matriz é inserido um grau de competência, que pode ir desde uma simples informação até um valor indicando uma grande competência.

Mais sobre matrizes de competência

A tabela a seguir é um exemplo simples e bem característico de uma matriz de competência de línguas estrangeiras, onde inglês e francês são áreas de conhecimento, leitura, escrita e conversação são habilidades, e as competências são fluente, intermediário, iniciante e nulo. Esses dados são comumente vistos em currículos e cadastros de empresas.

	Leitura	*Escrita*	*Conversação*
Inglês	Fluente	Fluente	Fluente
Francês	Iniciante	Nulo	Intermediário
Italiano	Intermediário	Intermediário	Avançado

Tabela: Exemplo de Matriz de Competência

Arquitetura e design

Modelo conceitual de dados

Uma matriz de competência possui áreas de conhecimento como colunas e habilidades como linhas; para cada habilidade há uma lista de valores possíveis; cada combinação entre área de conhecimento e habilidade de uma mesma matriz é uma célula da matriz de competências; cada pessoa pode ter atribuído um valor, dentre os possíveis, em cada célula de cada matriz; uma pessoa participa de um grupo e um grupo possui um gerente, também um grupo pode ter subgrupos. Assim chegamos no seguinte modelo conceitual.

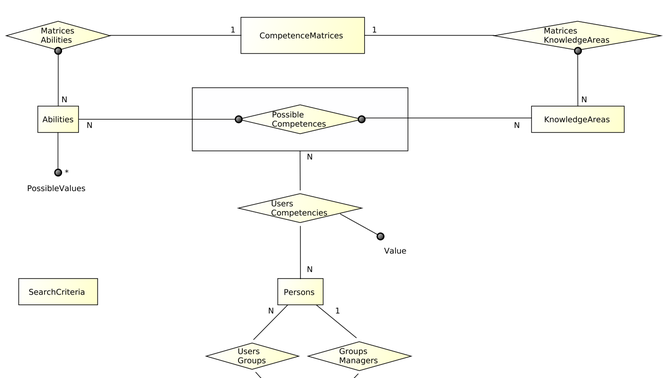

Figura: Modelo Conceitual de Dados seguindo o padrão de [4]

Estilo arquitetural

A decisão pelo uso do arcabouço Rails implica em um conjunto de decisões arquiteturais segundo a definição de estilo arquitetural de [6]. Essas decisões incluem:

- Padrão de projeto *Active Record*, que funciona como um *wrapper* de banco de dados orientado a objetos para banco de dados relacional, tornando o sistema quase independente de qual SGBD está sendo usado;
- Padrão de projeto *Model2* (*a.k.a. MVC2* ou *Rails MVC*) para a organização das camadas do sistema;
- Padrão arquitetural RESTful para a comunicação cliente-servidor;
- Geração de código para os passos iniciais do sistema e modelagem dos dados;
- Princípio "DRY"(não repita você mesmo), acompanhado de várias funcionalidades para repetir o mínimo de código possível.
- Princípio "Convenções sobre configurações", para facilitar o processo de decisões arquiteturais, de desenvolvimento e para que qualquer outro desenvolvedor Rails possa entender facilmente o código desenvolvido.
- Conjunto de bibliotecas disponibilizando ferramentas para testes, internacionalização, envio de mensagens, tratamento de erros, entre outras funcionalidades.

Metodologia

O desenvolvimento do projeto foi feito em três etapas. A primeira foi a procura por referências para conhecer mais sobre gerenciamento de competências e conhecer outros sistemas similares. A segunda foi o planejamento arquitetural, onde foi definido o modelo de dados e foi decidido o desenvolvimento com *Ruby on Rails*. A terceira foi o desenvolvimento em si.
No desenvolvimento foram usadas técnicas derivadas de programação extrema, como TDD (desenvolvimento dirigido a testes) e *baby steps* (dividir o desenvolvimento em pequenas evoluções, registradas por um controle de versões, no caso GIT). Foi usada também uma ferramenta para verificar o quanto do código está coberto por testes, que, no momento deste poster, era de mais de 97% do código.

Resultado

O resultado obtido é um sistema de gerenciamento de competências com cadastro e *login* de usuários, gerenciamento de equipes, criação e gerenciamento das definições das matrizes de competência, preenchimento das matrizes dos usuários e busca de usuários a partir de suas competências. A busca pode ser feita com critérios de graus de competência envolvendo várias matrizes. O resultado é exemplificado abaixo com as três principais funcionalidades.

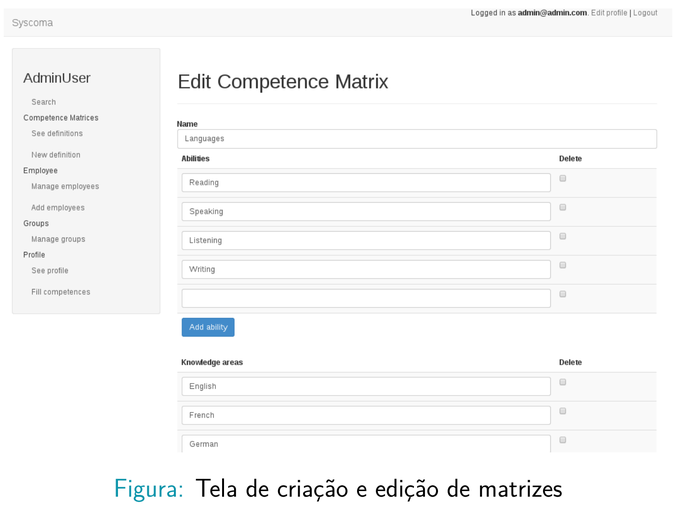

Figura: Tela de criação e edição de matrizes

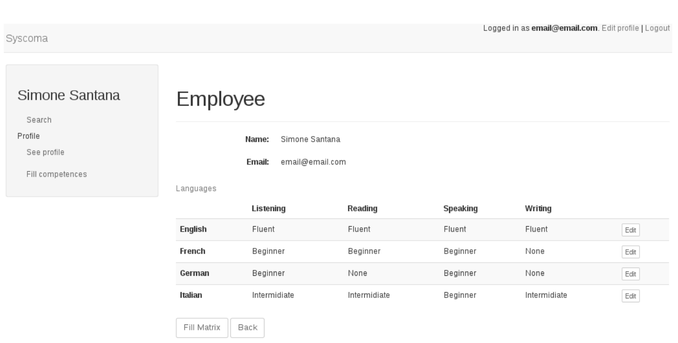

Figura: Tela de exibição de uma matriz preenchida

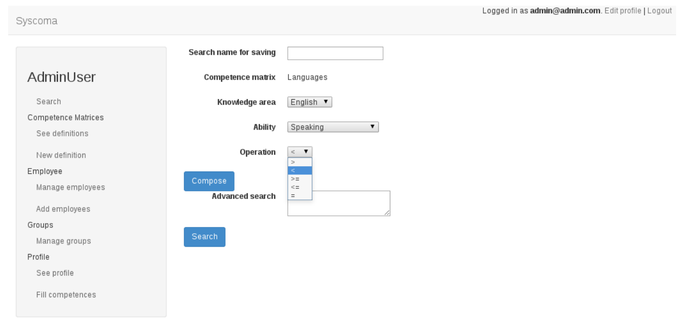

Figura: Tela de busca avançada de pessoas satisfazendo critérios de competência

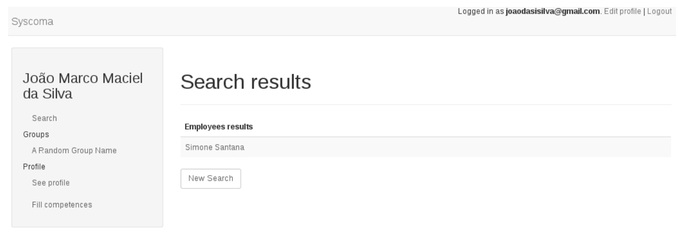

Figura: Tela de exibição dos resultados de busca

O sistema é fácil de ser instalado, com poucos comandos. É fácil de ser entendido, pois segue os padrões de projeto usados pelo arcabouço Rails e a sintaxe fácil de ser lida do Ruby. O uso é por meio de um navegador, podendo ser acessado pela Internet ou por uma *intranet*.

Futuro

Algumas tarefas não puderam ser concluídas a tempo e precisam ser feitas para ter o sistema em uso.
A tarefa mais importante para a próxima fase do projeto é fazer um mapeamento de competências em alguma área ou empresa. Uma sugestão seria mapear as necessidades do CCSL.

Conclusão

Com esse trabalho foi possível treinar diversas técnicas de desenvolvimento, aprender a colocar em produção um sistema baseado na Internet e colocar em prática diversos conceitos sobre computação aprendidos dentro e fora do curso.
O sistema resolve o problema ao qual se propõe e é fácil de ser instalado e usado por empresas e por qualquer pessoa. O código é fácil de ser entendido, modificado e estendido.
Com o uso desse sistema é possível, de modo prático, rápido e fácil, acelerar o processo de seleção interna de funcionários, de busca por habilidades incomuns, mas necessárias em uma empresa, de avaliação de funcionários, de seleção dos funcionários capazes de dar treinamento e dos que o necessitam.

Referências

[1] Valdemar W Setzer (1999). *Dado, informação, conhecimento e competência*. http://www.ime.usp.br/~vwsetzer/dado-info.html

[2] Hugo Pena Brandão e T de A Guimarães (2001). *Gestão de competências e gestão de desempenho: tecnologias distintas ou instrumentos de um mesmo construto*. RAE.

[3] Pedro Paulo Carbone, Hugo Pena Brandão e João Batista Diniz Leite (2005). *Gestão por competências e gestão do conhecimento*. FGV.

[4] Valdemar W Setzer e Flávio Soares C da Silva (2005). *Bancos de dados: aprenda o que são, melhore seu conhecimento, construa os seus*. Editora Edgard Blücher.

[5] Martin Fowler (2002). *Patterns of enterprise application architecture*. Addison-Wesley

[6] Ian Sommerville (2004). *Software engineering*. International computer science series.

[7] Taylor, Richard N / Medvidovic, Nenad / Dashofy, Eric M (2009): *Software architecture: foundations, theory, and practice*. Wiley Publishing.

[8] Robert C Martin (2008). *Clean code: a handbook of agile software craftsmanship*. Pearson Education.

[9] *Rails, web development that doesn't hurt*. http://rubyonrails.org/. Accessado: 15/10/2014.

[10] *Ruby on Rails guides*. http://guides.rubyonrails.org/. Accessado: 15/10/2014.

[11] *A guide to testing rails applications*. http://guides.rubyonrails.org/testing.html. Acessado: 15/10/2014.

[12] Aslak Hellesøy, Joseph Wilk, Matt Wynne, Gregory Hnatiuk e Mike Sassa (2014). *Cucumber: Behaviour driven development with elegance and joy*.

[13] *Ruby, o melhor amigo do programador*. https://www.ruby-lang.org/pt/. Accessado: 15/10/2014.

[14] *Simplecov*. https://github.com/colszowka/simplecov. Accessado: 15/10/2014.