UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

GABRIEL BRAGA VISCONTI

Sistema de Gerenciamento de Equipes e Tarefas

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação

Orientador: Prof. Dr. Leandro Krug Wives

Porto Alegre, julho de 2015.

RESUMO

Gerenciamento de pessoas num ambiente de desenvolvimento de software, de maneira a otimizar o tempo de realização das tarefas e a alocação dos recursos, tanto tecnológicos como humanos, é uma responsabilidade complexa que exige, além de outros, conhecimento acerca das preferências e habilidades de cada integrante. No caso de grupos grandes, essa tarefa pode se tornar difícil para os gerentes e frustrante para os desenvolvedores. Assim, este trabalho tem como objetivo propor uma ferramenta para tornar esse ambiente mais agradável e incentivador para os envolvidos, criando maneiras de recomendar as pessoas mais indicadas para cada tarefa, além de incorporar elementos de *gamification* ao processo, de modo a motivar os integrantes da comunidade.

Palavras-chave: Sistema de gerenciamento, software, equipes, recomendação, gamification.

SisGET - Tasks and Teams Management System

RESUMO

People management is important in a software development environment in order to optimize the time of the accomplishment of tasks and allocation of resources, both technological and human. However, it is a complex responsibility that requires, among others, knowledge about the preferences and skills of each member. For large groups, this task can be difficult for managers and frustrating for developers. This work propose an application to make such environment more pleasant and supportive for those involved, creating ways to recommend the best people for each task, as well as incorporating elements of *gamification* to the process in order to motivate community members.

Palavras-chave: Management system, software, team, recomendation, gamification.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

SisGET	Sistema de Gerenciamento de Equipes e Tarefas
API	Application Programming Interface
REST	Representational State Transfer
Yii	Yes, It Is
Gii	Yii Code Generator
MVC	Model-View-Controller
DRY	Don't Repeat Yourself
CRUD	Create Read Update Delete
PHP	Hypertext Preprocessor
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
DOM	Document Object Model
AJAX	Asynchronous JavaScript and XML
DDL	Data Definition Language

SUMÁRIO

1 INTRODUÇÃO

Gerenciar muitas pessoas com conhecimentos variados em um ambiente de desenvolvimento de software, alocando tarefas, otimizando o tempo e a produtividade é uma grande responsabilidade. Geralmente esse compromisso está restrito a poucas pessoas, que devem criar, distribuir e validar projetos e tarefas. Ferramentas para gerenciamento de desenvolvimento de software, como o Redmine[1] ou o Trac[2], auxiliam nessas reponsabilidades, pois oferecem controle para tarefas, projetos, prazos, horas usadas no desenvolvimento, versionamento de código, registro das demandas e dificuldades dos usuários.

De forma ideal, para o gerenciamento e a alocação de serviços, é necessário que os gerentes estejam a par da carga de trabalho de seus subordinados, suas preferências e limitações, pois, ao longo do tempo, o conhecimento de um desenvolvedor é direcionado para determinadas competências que se tornam preferências pessoais. No entanto, nem sempre acontece dessa forma e alguns problemas podem surgir, tais como a atribuição de uma tarefa desproporcional (muito fácil ou muito difícil) ao conhecimento; atribuição de tarefas para alguém que não goste de fazê-la; existência de ócio, pois o gerente pode não perceber que alguém já acabou o que estava fazendo e não aloque novos trabalhos; perda de motivação, ao passar trabalhos desinteressantes de modo geral, etc.

Outro ponto importante, considerando que o administrador tenha conhecimento sobre tais preferências pessoais, a substituição ou desligamento dele pode ser prejudical ao grupo pelos mesmos motivos, pois um novo gerente terá de se ambientar para administrar com harmonia. Portanto, permitir que um novo gerente não perca tempo conhecendo previamente todas aquelas preferências individuais da comunidade e tenha uma ferramenta de auxílio é vantajoso.

Como proposta de solução desses problemas, é interessante propor um sistema que permita o acompanhamento frequente de um grupo de desenvolvedores e que recomende as pessoas mais indicadas para uma atividade, baseando-se em preferências e no histórico delas, listando as mais qualificadas para determinadas tarefas ou projetos. As métricas para essa listagem podem ser tanto abstratas (gostar e/ou saber) como concretas (quantidade de tarefas no momento e/ou nível de experiência em algum conhecimento). Dessa forma, espera-se que o tempo gasto em cada tarefa seja otimizado e que o desenvolvedor sinta-se motivado. Com o intuito de aumentar mais ainda os níveis de envolvimento e de motivação dos participantes, elementos de *gamification* podem ser utilizados como parte da ferramenta, tais como evolução de níveis e troca de pontos acumulados por prêmios. Esses elementos consistem na aplicação de elementos característicos dos jogos em

[1] http://www.redmine.org/
[2] http://trac.edgewall.org/

contextos não lúdicos.

Assim, este trabalho tem como objetivo propor um **sis**tema de **g**erenciamento de **e**quipes e **t**arefas (SisGET) capaz de recomendar, por meio da análise do histórico de desenvolvimento de cada indivíduo e de suas preferências, os mais indicados para determinadas tarefas ou projetos em um centro de processamento de dados, lançando mão de técnicas de *gamification*.

O texto está estruturado da seguinte forma: no próximo capítulo encontram-se a fundamentação teórica e as tecnologias utilizadas para desenvolvimento do sistema. No terceiro capítulo é apresentada a estrutura do sistema proposto, suas entidades do banco de dados e funcionalidades requeridas por meio de user stories. No quarto capítulo são apresentados os resultados obtidos, demonstrando a utilização da aplicação. No quinto capítulo é feita uma breve avaliação e comparação com sistemas com propostas similares, além dos resultados de um questionário sobre a usabilidade. Por fim, no capítulo final são feitas as conclusões e sugestões de trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA E TECNOLOGIAS RELACIONADAS

Este capítulo apresenta conceitos, técnicas, ferramentas, tecnologias e demais fundamentos relacionados com o sistema desenvolvido. Neste capítulo também será descrito o fluxo de trabalho utilizado no Redmine no que diz respeito à atribuição de tarefas e ao motivo de acessar as informações nele contidas.

2.1 *Gamification* e seus elementos

De acordo com (Simões 2012), o termo "*gamification*"(em português, uma tradução aproximada seria "ludificação") começou a ter uso por volta de outubro de 2010. É definido pelo uso de elementos característicos de jogos em contextos usualmente não recreativos, para fortalecer o engajamento dos participantes e incentivar comportamentos desejados, por exemplo, na área de marketing, são aplicadas campanhas de fidelização de clientes por meio da acumulação de pontos que podem ser trocados por algum produto.

Gamification oferece uma alternativa para o acompanhamento de desempenho. Quando o participante é desafiado a superar obstáculos para realizar uma tarefa, se o fizer com sucesso, o fato de obter recompensas, como pontos, distintivos, níveis, gera grande satisfação. Além disso, o sistema baseado em *gamification* precisa indicar, a qualquer momento, o quão bem os membros estão cumprindo tarefas, não só para o próprio interessado, mas para toda comunidade. Essa informação ajuda os líderes a selecionar os membros da equipe com os pontos fortes desejados e determinar se eles podem auxiliar outro membro durante uma tarefa. Outro ponto positivo é diminuir a discriminação dos gestores em relação aos subordinados, seja de gênero, raça ou idade (entre as mais comuns). Sistemas e processos baseados nela geram informações claras e objetivas, que tornarão decisões tendenciosas mais difíceis de serem justificadas.

Em termos gerais, *gamification* é usada para motivar pessoas, criar competições saudáveis entre equipes ou pessoas, encorajar a lealdade de consumidores, entre outros benefícios. Deve ser empregada em contextos onde é difícil influenciar o comportamento das pessoas e aumentar seu engajamento.

Identificam-se três conceitos recorrentes em ambientes onde se implementa *gamification* (Zicherman 2012):

– **Feedback**: formas de se comunicar com os jogadores, apresentando seus resultados e conquistas obtidos por suas realizações. Isso contribui para manter um bom nível de interação entre os membros.

– **Friends**: esse conceito está ligado à ajuda (altruísta ou não), divisão de tarefas e colaboração, ou seja, fatores que produzem uma comunidade fortalecida.

– **Fun**: o conceito que representa a satisfação e diversão gerada por conquistas.

2.2 Elementos de Jogos

Elementos de jogos são um conjunto de componentes e funcionalidades que podem ser usados nos contextos normalmente não recreativos. Podem ser para informar os jogadores sobre seu desempenho, para recompensá-los, ou, de modo mais dinâmico, mostrar sua evolução ao longo do tempo, com métricas para cálculo de níveis, por exemplo. Na tabela abaixo, estão descritas formas de simular os componentes básicos num ambiente com *gamification*.

Os elementos mais comuns utilizados são desafios, níveis, recompensas e avatares. Desafios podem ser tarefas com objetivos claros que devem ser alcançados, recompensas são itens ganhos, podem ser tanto virtuais como reais e o nível indica a evolução conforme desafios são cumpridos, implicando participar de desafios cada vez maiores.

Componentes Básicos	**Componentes de Jogos**
Comunicação e Recompensa	*pontos, barras de progresso, distintivos*
Dinâmica Social	*ajudar, solicitar prêmios, convidar amigos*
Tempo de Jogo	*níveis, regras, economia virtual, escalonamento de recompensa*

Tabela 2.1: Componentes Básicos vs. Componentes de Jogos

Fonte: Zicherman, 2012 (traduzido pelo autor)

2.3 Framework Yii

Yii[1] é um framework de código aberto para desenvolvimento de aplicações web em PHP5. Implementa o padrão MVC e possibilita a criação de códigos DRY (don't repeat yourself). Em Engenharia de Software, esses códigos visam reduzir a repetição de informação de qualquer tipo, útil na arquiterura MVC ou qualquer outra que faça separação entre dados e interface (Winesett 2012).

Além disso, o Yii conta com uma ferramenta chamada Gii, que permite geração automatizada da estrutura MVC e código CRUD para todas entidades do banco de dados. Por exemplo, caso exista uma tabela chamada *Pessoa* no banco de dados, seriam criados os

[1] http://www.yiiframework.com/

arquivos de código fonte para a controller (*PessoaController*), para a model (*Pessoa*) e o diretório pessoa dentro do diretório de views para as interfaces que aquela controller deve utilizar.

2.4 Xampp

XAMPP[2] é um servidor independente de plataforma, que consiste na base de dados MySQL, no servidor web Apache e nos interpretadores para as linguagens PHP e Perl. O nome provém da abreviação de **X** (opera em diferentes sistemas operacionais), **A**pache, **M**ySQL, **P**HP, **P**erl. É um software livre que atua como um servidor web.

2.5 PHP

PHP (Hypertext Preprocessor) é uma linguagem interpretada para desenvolvimento web ou propósitos gerais. Sua interpretação acontece no servidor, que, após isso, envia a saída, em código HTML, imagem ou qualquer outro dado para o cliente.

2.6 MySQL

Para o desenvolvimento do sistema, foi usado o MySQL. O XAMPP oferece uma interface gráfica para manipular as tabelas e as relações com facilidade, além de criar e exportar a DDL.

2.7 CSS

CSS é utilizado para colocar estilos em páginas web. Suas regras podem ser genéricas para elementos do HTML ou específicas para certas classes ou identificadores, tornando fácil sua manutenção, pois não é necessário colocar as mesmas regras em cada elemento. Além disso, a separação do HTML e do CSS em locais diferentes, permite melhor clareza de código, facilitando a manutenção

2.8 JQuery

JQuery é uma biblioteca JavaScript de código aberto desenvolvida para simplificar a interação com o DOM. Sua sintaxe torna mais simples a navegação do documento HTML, a seleção de elementos DOM, a criação de animações, a manipulação de eventos e o uso de AJAX. A biblioteca também oferece a possibilidade de criação de plugins.

2.9 MVC

Com a preocupação de proporcionar a reusabilidade de código e a sua manutenção, visto que ainda se tem muito a melhorar e/ou implementar, este sistema foi construído no padrão model-view-controller (MVC), que é uma arquitetura de software criada em 1979 por Trygve Reenskaug (MVC XEROX PARC 1978-79).

[2] www.apachefriends.org/

Neste padrão, a Model é responsável por manter os dados da aplicação. Normalmente os dados são permanentes, sendo armazenados num banco de dados. Além disso, a model deve aplicar as regras de negócio da aplicação, centralizando o gerenciamento dos dados para evitar a repetição de código em outras áreas, uma prática que pode gerar inconsitência dos dados.

A view deve gerar a interface para o usuário, utilizando os dados fornecidos pela model.

Por fim, a controller deve coordenar a aplicação. Ela recebe os eventos gerados pelo usuário, interage com a model correta e então mostra uma view apropriada para aquele evento ao usuário.

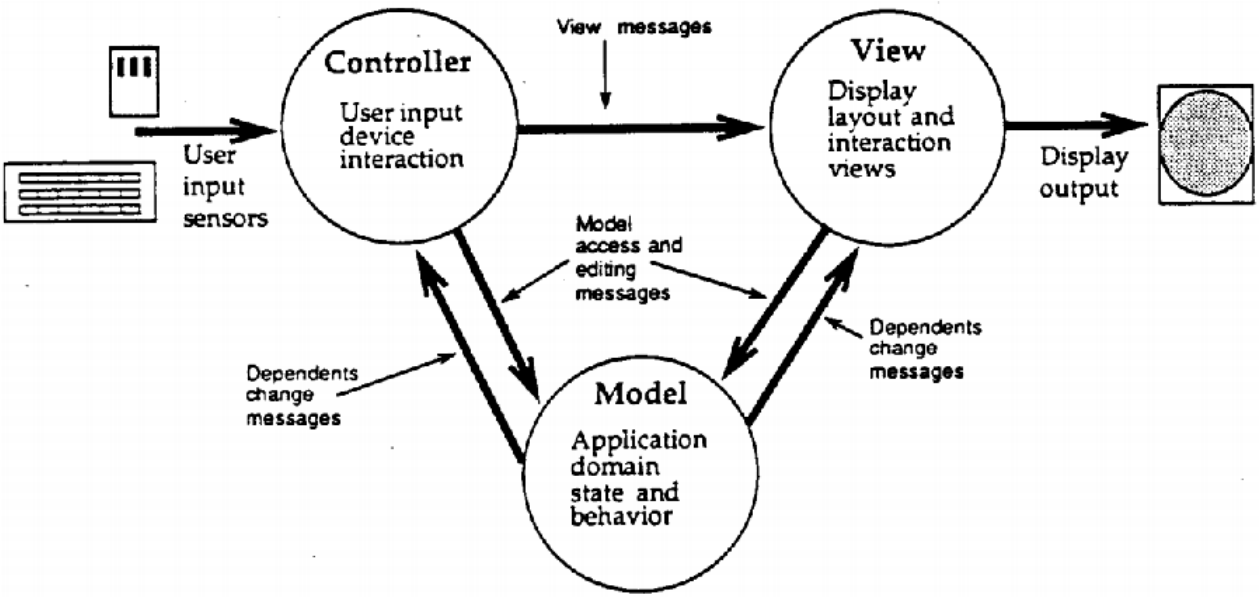

Figura 2.1: Diagrama de interação do MVC

Fonte: (Krasner e Pope 1988)

Outro motivo se dá em razão da utilização de orientação a objetos, que torna a complexidade do código maior, portanto separar dados e apresentação é benéfico, pois alterações no visual das páginas não afetam a manipulação de dados e vice-versa.

2.10 Redmine

Redmine[3] é um sistema projetado para web de código aberto para gerenciamento e acompanhamento de tarefas e projetos, lançado em 2006. Desenvolvido em Ruby on Rails, é *cross-plataform* e *cross-database* (Lesyuk 2013).

Esse gerenciador é usado como principal ferramenta de distribuição e validação de tarefas no centro de processamento de dados onde o SisGET será implementado inicialmente. Além disso, permite melhor acompanhamento dos subordinados pelos gestores, pois faz controle de horas gastas na realização de atividades em comparação às horas previstas inicialmente, mostra relatórios de desempenho, envia avisos por email, permite que

[3]http://www.redmine.org/

o desenvolvedor faça questionamentos para sanar dúvidas ou mal entendidos, mantendo o histórico dessa interação.

Juntamente com o TortoiseSVN[4] (controlador de versões), mantém log de versões enviadas e suas diferenças incrementais (útil no caso de perdas indevidas de código ou arquivos).

Importante salientar que o SisGET tem plenas condições de funcionar por conta própria, controlando *gamification* e recomendação, porém muitas funcionalidades existentes no Redmine não foram implementadas por não serem o foco. Como o centro de processamento de dados já está muito acostumado com o Redmine e não poderia abrir mão daquelas funcionalidades, neste primeiro momento, o SisGET vai funcionar como um sistema complementar, atendendo algumas necessidades que não são providas pelo Redmine. As informações sobre tarefas e suas situações serão acessadas utilizando a API REST do Redmine, dessa forma, pelo menos, a alimentação dos dados não precisa ser redundante.

2.11 Fases das Tarefas no Redmine

De maneira objetiva, evitando descrever detalhadamente o funcionamento interno do gerenciador, o fluxo de tarefas, utilizando as situações predefinidas pelo Redmine, é estabelecido da seguinte forma:

1. Novas tarefas são criadas e marcadas como *New* ou *In Progress* (ou qualquer outra criada de modo personalizado). Essas situações devem ser do tipo aberta.

2. O administrador atribui essa tarefa a alguém.

3. O desenvolvedor, ao finalizar, marca a tarefa como *Resolved* (ou outra, também personalizando). Essa situação deve ser do tipo neutra e a tarefa deve ser atribuida de volta a um administrador para que ele a valide.

4. O administrador valida e marca a tarefa como *Closed*, que é uma situação do tipo fechada (igualmente outras denominações podem existir), e deixa sem atribuição a alguém (internamente, o campo *assignee* no banco de dados do Redmine fica vazio).

Seguindo esse fluxo, 13 casos de possíveis eventos foram elaborados para compor as ações que o SisGET deve realizar na sincronização, utilizando a API REST do Redmine (Redmine Project). Os mesmos são apresentados na tabela seguinte.

[4]http://tortoisesvn.net/

Caso nº	Situação no Redmine	Situação no SisGET	Ação	Automática
#1	Aberta	Não existe	Mostrar para importar.	Não
#2	Aberta	Aberta	-	-
#3	Fechada	Aberta s/ atribuição	Apagar	Sim
#4	Fechada	Aberta c/ atribução à pessoa X	Fechar; Conceder pontos à X.	Sim
#5	Fechada	Fechada com atribuição	-	-
#6	Fechada	Não existe	-	-
#7	Aberta s/ atribuição	Fechada	Retirar pontos; Reabrir; Remover histórico.	Sim
#8	Aberta s/ atribuição	Aberta c/ atribução à pessoa X	Retirar de X	Sim
#9	Aberta c/ atribução à pessoa X	Fechada c/ atribução à pessoa X	Retirar pontos; Reabrir.	Sim
#10	Aberta c/ atribução à pessoa X	Fechada c/ atribução à pessoa Y	Reabir; Remover histórico de Y; Retirar pontos de Y; Atribuir à X.	Sim
#11	Aberta c/ atribução à pessoa X	Aberta s/ atribuição	Atribuir à X.	Sim
#12	Aberta c/ atribução à pessoa X	Aberta c/ atribução à pessoa X	-	-
#13	Aberta c/ atribução à pessoa X	Aberta c/ atribução à pessoa Y	Retirar de Y; Atribuir à X.	Sim

Figura 2.2: Tabela de Eventos

3 O SISTEMA DE GERENCIAMENTO DE EQUIPES E TAREFAS (SISGET)

Neste capítulo serão descritas as bases do sistema proposto, primeiramente mostrando as *user stories* coletadas ao longo do desenvolvimento e a modelagem final do banco de dados com o diagrama Entidade-Relacionamento.

3.1 User Story

Uma *user story* (história de usuário) descreve uma funcionalidade do sistema que é requisitada pelo usuário final.

User stories são compostas por três aspectos:

- **Título e descrição** breves, nos quais focam numa funcionalidade e/ou objetivo.
- **Conversa** sobre a aceitação e possíveis mudanças dessa funcionalidade ao longo do desenvolvimento.
- **Testes** que cubram os detalhes e determinem a conclusão da história.

De maneira simplificada, os desenvolvedores de software mantém as descrições curtas, apenas como um lembrete, comumente no formato:

Como um <tipo de usuário> eu quero <uma funcionalidade que> de modo a <benefícios gerados>.

Segundo (Cohn 2008) os benefícios podem ser suprimidos, produzindo o seguinte formato:

Como um <tipo de usuário> eu quero <uma funcionalidade que>.

As histórias, tabeladas a seguir, foram criadas e validadas em encontros com os usuários finais, em geral a cada 15 ou 30 dias.

Como...	Eu quero...
Administrador	CRUD Prêmio, Habilidade, Dificuldade, Capacitação
	CRUD Pessoa, Horários
	CRUD Usuário - além de reiniciar senhas
	Visualizar preferências e habilidades por pessoa
	Visualizar histórico de tarefas por pessoa
	Acompanhar em tempo real tarefas abertas e ocupação dos desenvolvedores
	Alterar níveis das habilidades das pessoas
	Receber recomendação de desenvolvedores para tarefas
	Distribuir prêmios
	Aproveitar o máximo de informação do Redmine
	Ser notificado sobre novas tarefas
Desenvolvedor	Indicar em quais conhecimentos prefiro desenvolver
	Indicar quais conhecimentos quero aprender
	Visualizar o progresso de minhas habilidades
	Visualizar o progresso da comunidade
	Visualizar meus dados pessoais
	Solicitar prêmios

Figura 3.1: Histórias de Usuários

3.2 Modelagem do Banco de Dados

O modelo de dados foi refinado ao longo do desenvolvimento, recebendo inclusões de entidades e tendo outras removidas, de forma a simplificar o resultado e facilitar a legibilidade do diagrama, representado na figura 3.2. A descrição textual das entidades está nas subseções seguintes.

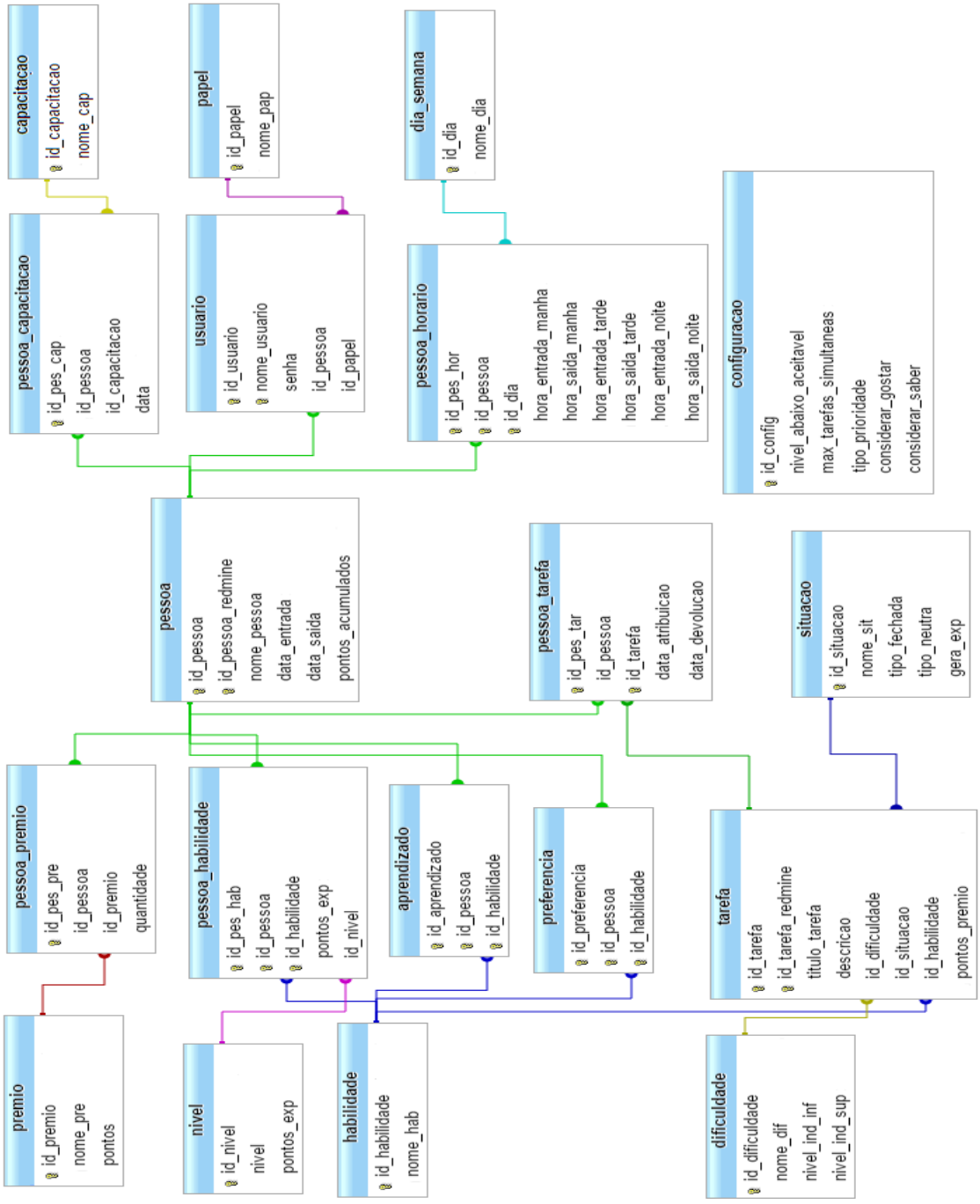

Figura 3.2: Diagrama Entidade-Relacionamento

– Entidade aprendizado: cada usuário do sistema pode indicar quais habilidades (dentre as existentes) ele deseja aprender. É útil para indicar aos administradores quais são os conhecimentos mais desejados para que façam capacitações.

– Entidade capacitação: armazena as capacitações que os usuários poderão fazer.

– Entidade configuracao: permite aos administradores configurar algumas opções para o algoritmo de recomendação. Tais opções serão melhor explicadas na seção 4.2.2

– Entidade dia_semana: registra os dias da semana.

– Entidade dificuldade: mantém os tipos de dificuldades criadas pelos gerentes, cada dificuldade tem um intervalo preferencial de níveis que devem ser considerados no momento da recomendação.

– Entidade habilidade: armazena as habilidades (conhecimentos) do ambiente de desenvolvimento.

– Entidade nível: registra os níveis que os usuários podem ter. Cresce indefinidamente, de acordo com o progresso deles.

– Entidade papel: guarda os tipos de usuários que podem existir, por exemplo, administrador, usuário convencional, convidado.

– Entidade pessoa: registra dados básicos de uma pessoa. O campo id_pessoa_redmine é o identificador dela no Redmine, obrigatório para possibilitar todo controle e sincronização dos casos da seção 2.11.

  O campo pontos_acumulados é aumentado quando uma tarefa é completada com sucesso (ou diminuído se for reaberta). Esse pontos servem para trocar por prêmios.

– Entidade pessoa_capacitação: registra todas capacitações feitas por uma pessoa, para manter um histórico.

– Entidade pessoa_habilidade: mantém todas as habilidades que uma pessoa tenha e qual o seu nível nela.

– Entidade pessoa_horario: salva os horários da pessoa em cada dia da semana.

– Entidade pessoa_premio: quando uma pessoa solicita um prêmio, isso é salvo nessa tabela. Ao ser concedida a premiação, esse dado é apagado e os pontos necessários são consumidos do campo pontos_acumulados da entidade pessoa.

– Entidade pessoa_tarefa: mantém o registo de todas as tarefas que uma pessoa realizou, inclusive aquelas não a tenha concluído ainda. Ao concluir a tarefa, o desenvolvedor ganha os pontos referentes à ela, acumulando pontos.

– Entidade preferencia: de maneira similar à entidade aprendizado, cada usuário do sistema pode indicar quais habilidades (dentre as existentes) ele prefere desenvolver. É utilizado como um dos critérios da recomendação.

– Entidade prêmio: com o objetivo de incentivar *gamification*, essa tabela contém prêmios que podem ser dados quando o desenvolvedor tem tal quantidade de pontos (que serão debitados quando o prêmio é concedido). Podem ser coisas simples como brindes ou mais sérias como o direito de faltar um dia de trabalho.

– Entidade situação: registra as possíveis situações de uma tarefa. O sistema, automaticamente, importa as mesmas situações existentes no Redmine, devendo o administrador indicar se é uma situação (dentre as fechadas) que gera experiência ao desenvolvedor que a realizou. Outra indicação indispensável são as situações neutras. Ver seção 2.11

– Entidade tarefa: registra dados da tarefa. Pelas mesmas razões da entidade pessoa, o campo id_tarefa_redmine é obrigatório para a sincronização funcionar corretamente.

– Entidade usuário: armazena informações sobre os usuários do sistema, para controle de login.

3.3 Estrutura de Classes

Nas seções seguintes, o resultado final da estrutura de classes é apresentado. Conforme explicado no capítulo anterior, a estrutura segue o padrão MVC, utilizando o framework Yii como base para o desenvolvimento.

– Classes Model: as models são responsáveis por interagir com o banco de dados, cada entidade do banco de dados é representada por uma. Todas as consultas que dizem respeito àquela entidade são feitas apenas pela model correspondente, assim facilitando a manutenção ou evolução do sitema. Por exemplo, todas consultas e CRUD em cima da entidade Pessoa, devem ser feitas pela Model Pessoa, mesmo partindo de outra model, pois assim se garante que num eventual erro do sistema, somente um trecho de código deve ser corrigido. Além disso, evita a repetição de código fonte em diversos locais.

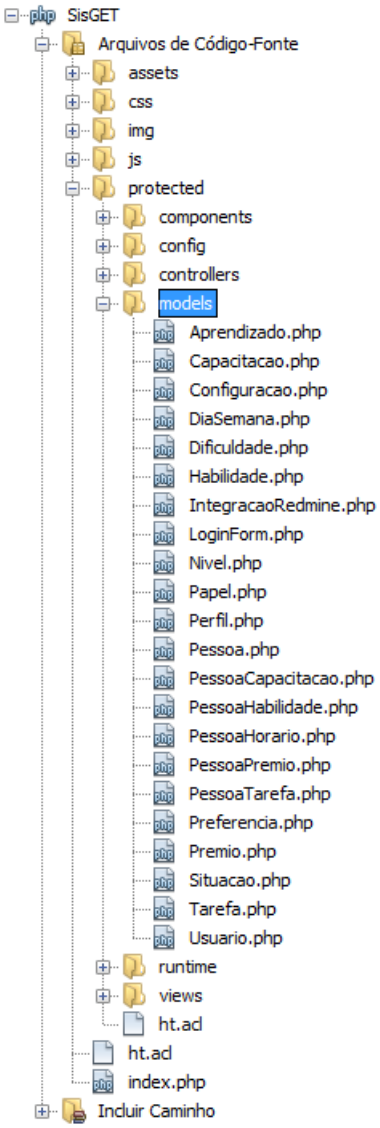

Figura 3.3: Estrutura das Classes Model

– Visualização: os arquivos de código fonte das views são organizados em diretórios, correspondentes às entidades do banco de dados. Cada um dos arquivos php é acessado por uma controller (da mesma entidade), que envia os dados necessários para o evento ativado pelo usuário.

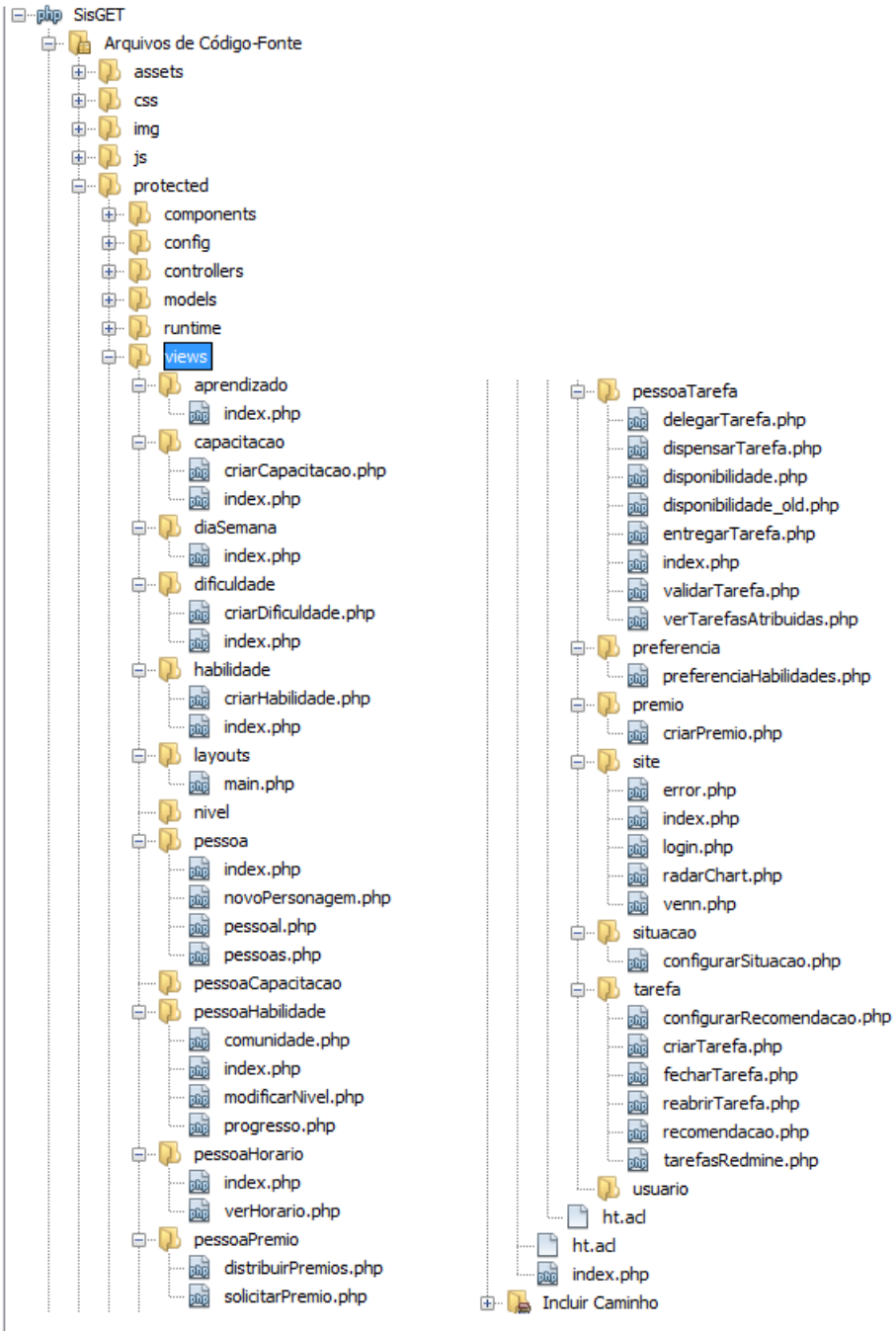

Figura 3.4: Estrutura das Views

– Classes Controller: as controllers devem receber os eventos ativados pelos usuários (pressionamento de botões, cliques em links, submissão de formulários, etc) e utilizar a(s) model(s) necessária(s) para juntar os dados e enviar para a view que os manipulará e mostrará formatado ao usuário.

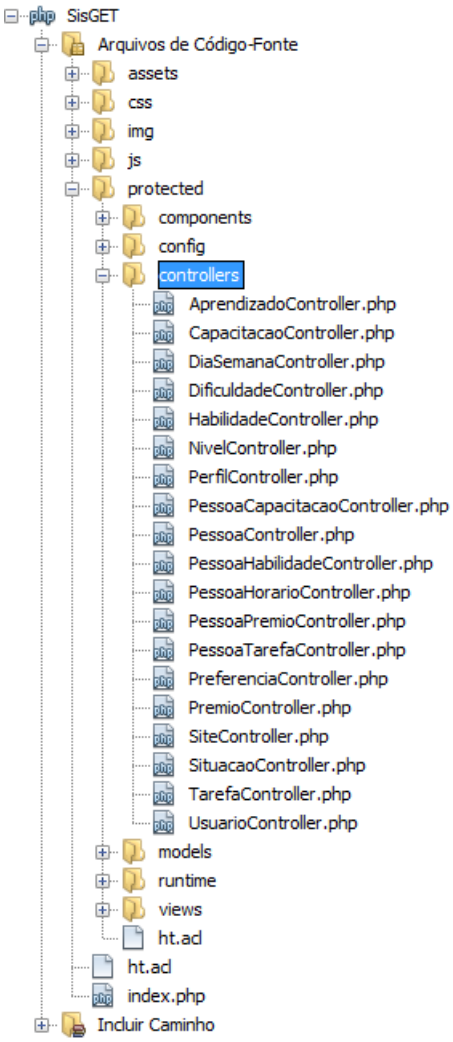

Figura 3.5: Estrutura das Classes Controller

3.4 Gamification no SisGET

No SisGET, os elementos de *gamification* usados foram pontos, níveis, barras de progresso e prêmios. Conforme uma pessoa conclui, com sucesso, tarefas, cujos pontos de recompensa são definidos pelos administradores, ela evolui no conhecimento exigido daquela tarefa, gerando pontos de experiência - que aumentam infinitamente.

Para cada conhecimento existente, o desenvolvedor visualiza uma barra de progresso, nela também é possível verificar quanto falta, em porcentagem, para avançar ao próximo nível. A função que rege a quantidade de pontos necessários para um nível "x"é dada pela seguinte fórmula:

$$f(x) = \frac{50}{3}(x^3 - 6x^2 + 17x - 12)$$

4 GUIA DE USO DO SISTEMA

Nesta seção são descritas as telas e formulários do sistema, para os diferentes tipos de usuário.

4.1 Usuário Convencional

O usuário convencional apenas acessa os itens do menu geral, descrito a seguir.

4.1.1 Menu Geral

Este menu permite ao Usuário Convencional interagir com o sistema, utilizando as seções de menu descritas a seguir.

– Seção Pessoal: nesta seção, a pessoa logada visualiza seus dados, tais como nome, data de entrada e seu papel no sistema, entre outros que venham a ser criados. A alteração da senha de acesso também é realizada aqui.

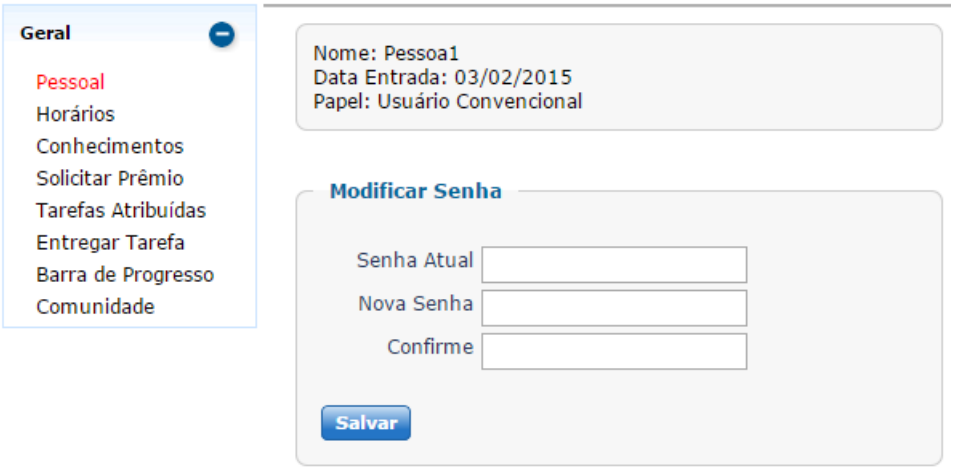

Figura 4.1: Seção Pessoal

– Seção Horários: aqui o usuário acessa sua tabela de horários, somente um administrador pode criá-la ou editá-la.

Figura 4.2: Seção Horários

– Seção Conhecimentos: nesta área, o usuário pode escolher os conhecimentos pelos quais tem preferência em trabalhar.

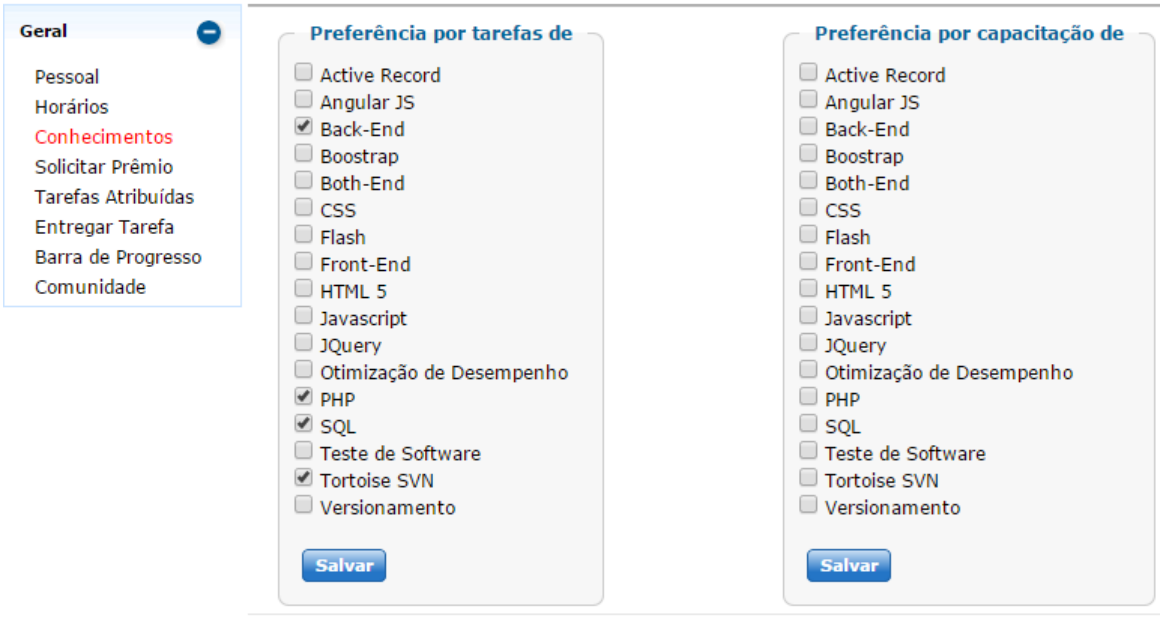

Figura 4.3: Seção Conhecimentos

– Seção Solicitar Prêmio: nesta tela, a lista de prêmios e a quantidade de pontos necessários para obtê-los (unitariamente) é mostrada.

  Na imagem de exemplo seguinte, percebe-se que é possível solicitar qualquer combinação de prêmios e quantidade, desde que não somem mais de 1200 pontos.

Figura 4.4: Seção Solicitar Prêmio

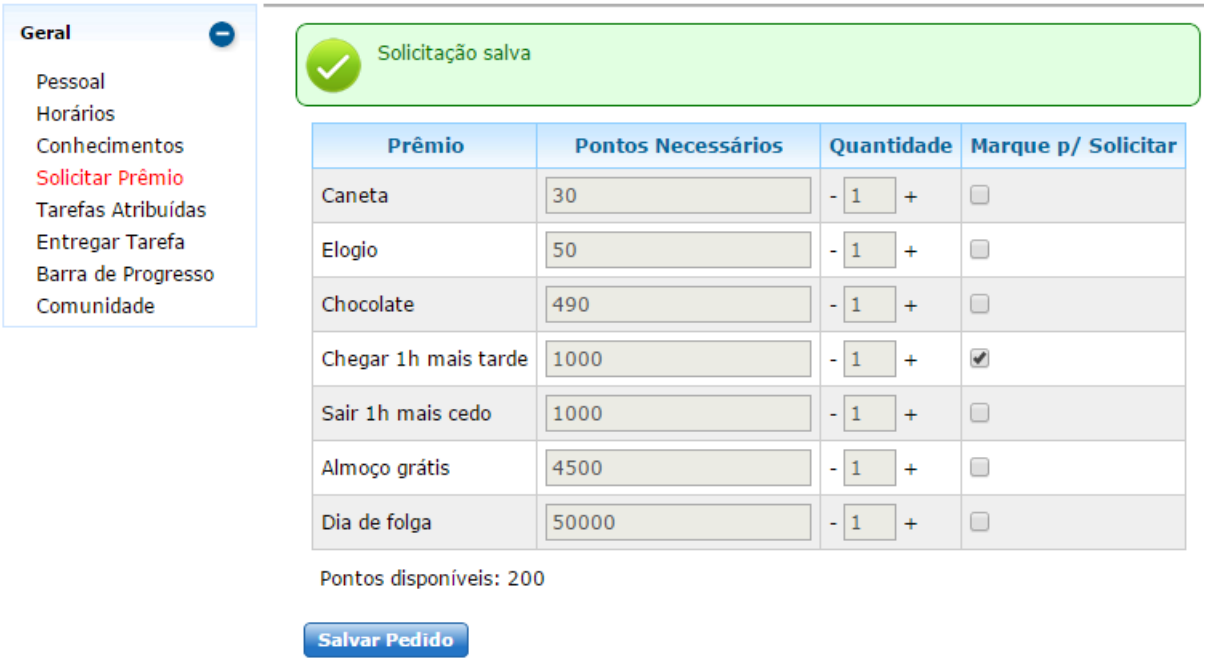

Figura 4.5: Seção Solicitar Prêmio - Confirmação

– Seção Tarefas Atribuídas: nesta seção, a pessoa logada vê listada as tarefas atribuídas à ela que se referem à situações do tipo aberta, nesse exemplo, composto pelas situações *New* e *In Progress*. Essas nomenclaturas de situações são importadas automaticamente do Redmine para manter as formas de uso iguais nas duas ferramentas.

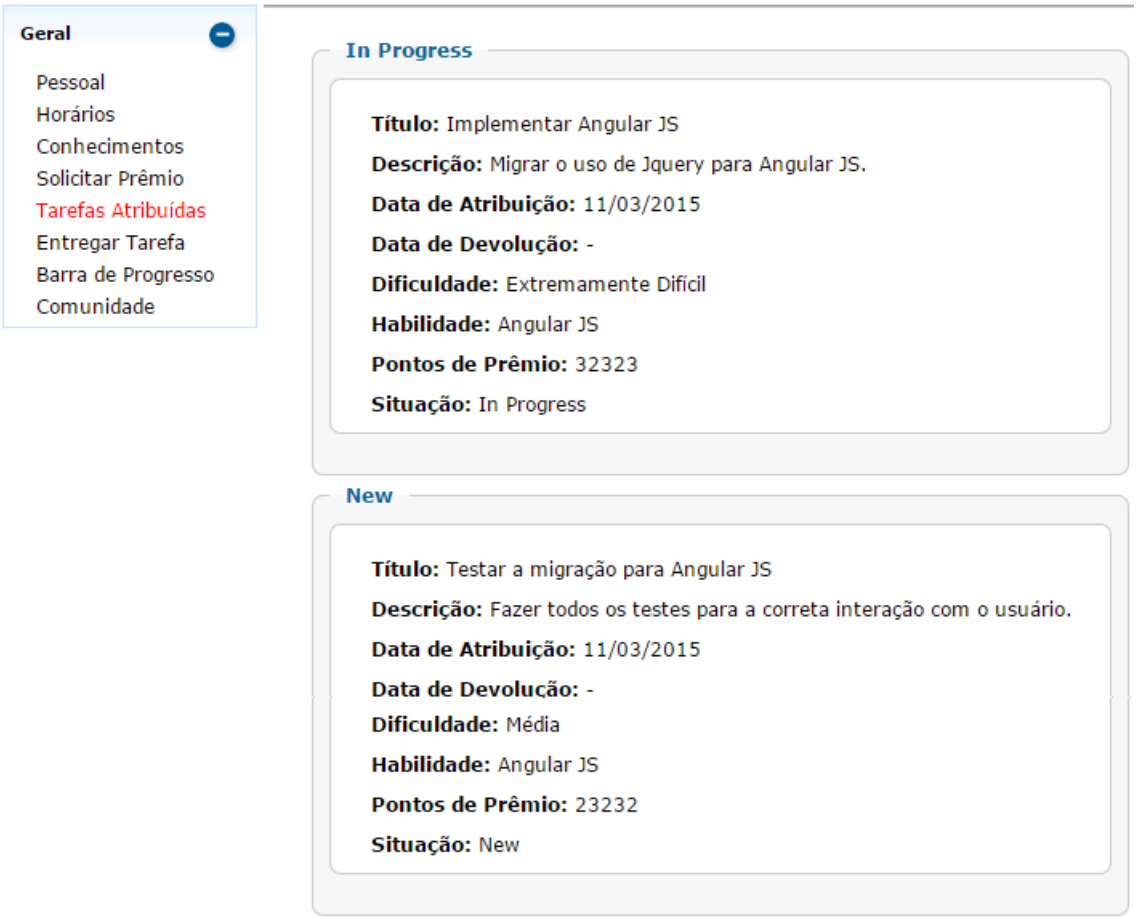

Figura 4.6: Seção Tarefas Atribuídas

– Seção Entregar Tarefa: neste item, o desenvolvedor pode entregar as tarefas que estão atribuídas a ele. Apesar do funcionamento estar implementado e correto, neste momento essa ação deve ser feita apenas no Redmine para depois ser importada. Tornar o SisGET independente será uma melhoria futura, incorporando funcionalidades do Redmine para torná-lo uma ferramenta única.

– Seção Barra de Progresso: aqui são mostradas as barras de progresso em cada habilidade que o desenvolvedor possui, ou seja, se desenvolveu alguma atividade naquela competência. Nelas estão seu nível atual, pontos de experiência e quanto falta até o próximo nível pela barra horizontal.

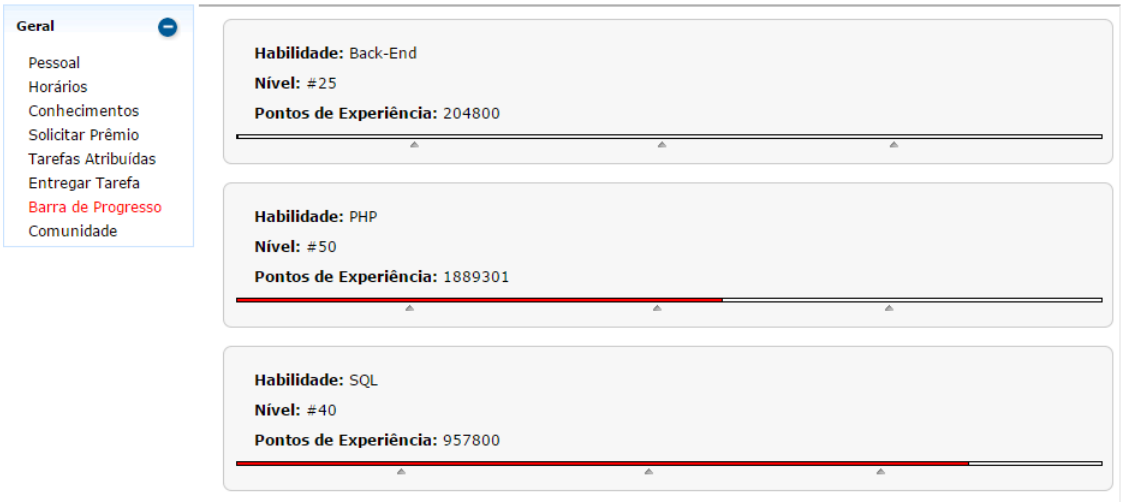

Figura 4.7: Seção Barra de Progresso

– Seção Comunidade: finalmente, nesta seção, um integrante pode olhar as barras de progressos de qualquer outra pessoa.

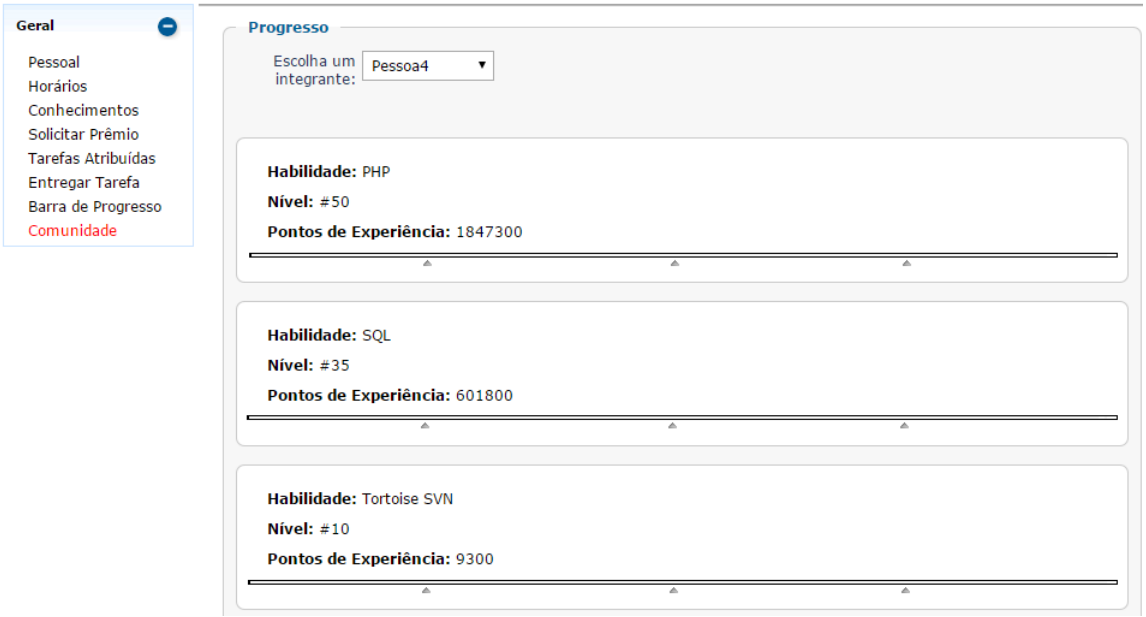

Figura 4.8: Seção Comunidade

4.2 Administrador

Um usuário com o papel de administrador, acessa não só o menu descrito acima, sem distinção das regras e ações expostas, mas também os menus descritos abaixo.

4.2.1 Menu Editor

No menu Editor está o CRUD de Prêmio, Habilidade, Dificuldade e Capacitação. Como as implementações do fluxo são semelhantes nos quatro, por simplificação, a descrição do funcionamento será feita com a tela do CRUD Habilidade, inclusive por ele ter uma funcionalidade extra em relação aos outros.

– Seção Criar Habilidade: as quatro ações do CRUD são feitas numa única tela. Ao carregá-la, é criado uma lista contendo as habilidades existentes além de um item específico, chamado -*Criar Nova*-. Selecionando esse item, todos os campos ficam em branco e o único botão disponível é o de salvar.

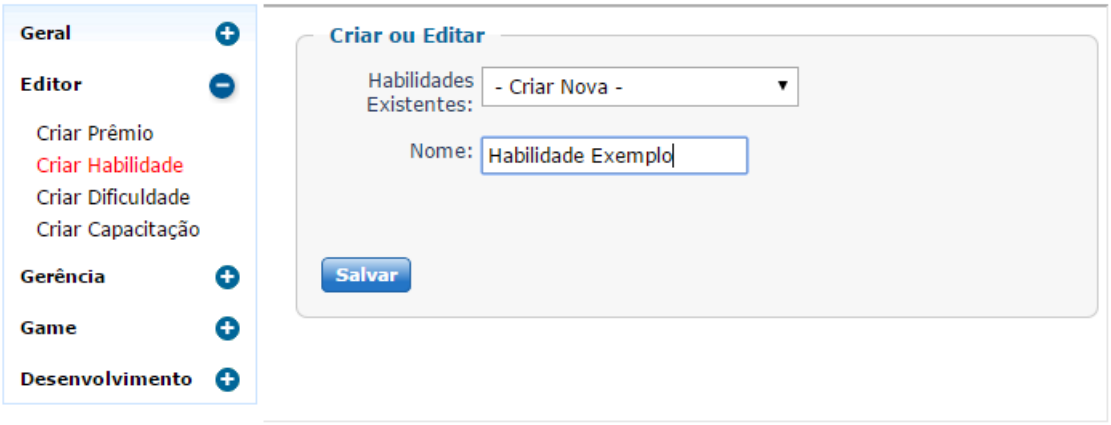

Figura 4.9: Criar Nova Habilidade

Ao selecionar um dos itens existentes, são carregadas suas informações. Nesse momento, o administrador pode editar os dados ou excluir esse item.

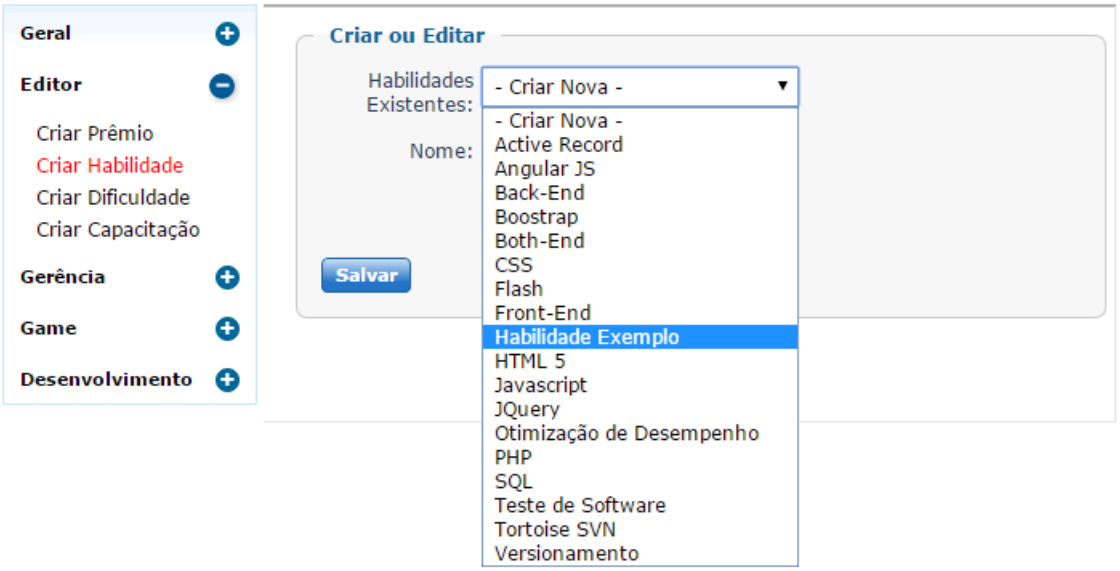

Figura 4.10: Editar Habilidade

Na figura seguinte, selecionou-se o item *Habilidade Exemplo* recém criado, o qual terá sua nomenclatura alterada para *Habilidade Exemplo II* ao clicar no botão *Atualizar*.

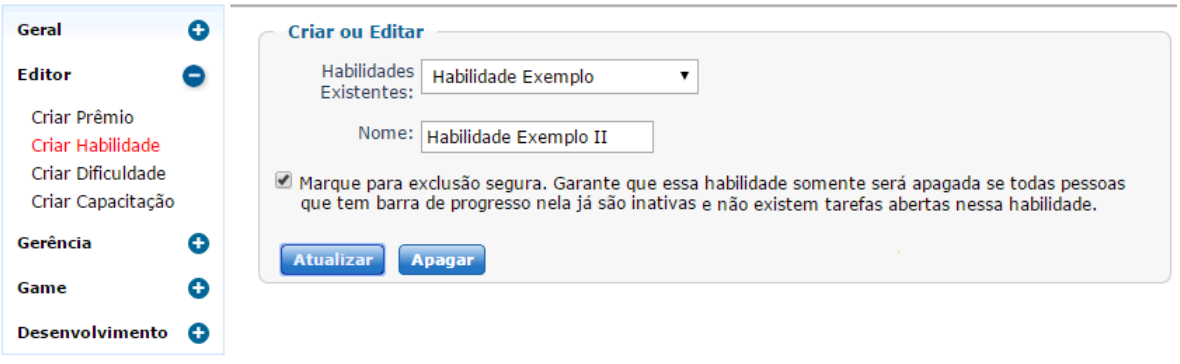

Figura 4.11: Editar Habilidade

Por fim, exclui-se o item ao clicar no botão *Apagar*. Nessa ação, existe a seguinte opção:

Marque para exclusão segura. Garante que essa habilidade somente será apagada se todas pessoas que tem barra de progresso nela já são inativas e não existem tarefas abertas nessa habilidade.

Isso é essencial, pois como o identificador de habilidade é uma chave estrangeira em outras entidades e a exclusão teria de ser em cascata, primeiro se faz o controle proposto visto que o administrador pode não perceber que algum dado importante seria apagado. Mesmo assim, excluir pode ser uma opção arriscada, visto que, ao remover uma tarefa, mesmo em situação de concluída, o histórico de desenvolvedores que sabem aquela habilidade se perde. Isso pode afetar o funcionamento preciso do algoritmo de recomendação, caso essa mesma habilidade venha a ser criada novamente.

4.2.2 Menu Gerência

Neste menu estão, entre outras, as seções importantes para configuração da recomendação e da maneira que o sistema deve reagir dependendo das situações em que as tarefas estão no Redmine e no SisGET.

- Seção Pessoas: nesta área, os gerentes visualizam dados relativos às pessoas, tais como nome, data de entrada, data de saída, pontos acumulados e tabela horária. Todos esses dados podem ser editados. Ademais são mostrados as tarefas correntes do desenvolvedor, suas barras de progresso e suas preferências.

- Seção Disponibilidade: nesta área estão relacionadas as tarefas que não estão sendo resolvidas e a carga de trabalho atual de cada indivíduo. Conforme exposto na Figura 2.2, o único caso que requer intervenção de um gerente é aquele no qual a tarefa ainda não foi importada, todos outros são atualizados automática e periodicamente. Como a tela de disponibilidade está configurada para atualizar a cada quinze minutos, ela mostra um retrato praticamente em tempo real do andamento. É a tela mais interessante para ser mantida em foco por um administrador quando não estiver usando outros módulos.

  Os elementos não tem nem tamanho, nem posição fixas, eles variam de acordo com a quantidade de texto dentro, exceto que a região global se divide com 25% da área para as tarefas à esquerda e 75% para as informações pessoais à direita, com isso espera-se simular um conjunto de lembretes colados num quadro, onde é mostrado nome, número de atividades correntes e quais conhecimentos necessários a elas.

  As legendas de cores são formadas a partir do máximo de tarefas que o administrador configura como limite recomendável de trabalho para seus colaboradores, nesse exemplo, cinco tarefas é o limite ideal, que colore em vermelho, três tarefas (limite máximo menos dois) colore em amarelo, pelo menos uma tarefa colore em azul e nenhuma tarefa colore em verde, sendo esses os que contém pessoas mais aptas a receber trabalho.

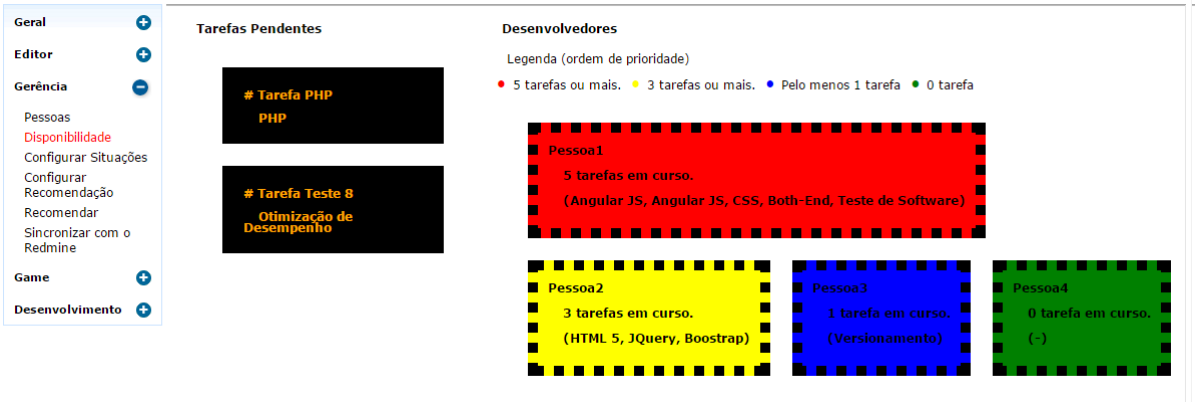

Figura 4.12: Tela Disponibilidade

– Seção Configurar Situações: no que diz respeito aos casos da figura 2.2, para o sistema funcionar corretamente, é necessário configurar quais situações do tipo fechada geram pontos para o desenvolvedor.

  Por padrão do Redmine, as situações desse tipo são *Closed* e *Rejected* (pode-se criar outras que serão automaticamente importadas para o SisGET). Sendo assim, somente quando uma tarefa, ao passar de uma situação aberta (por padrão *New* ou *In Progress*) para a situação fechada *Closed* irá gerar pontos, ao passo que a passagem para *Rejected* não.

  Outra configuração que deve ser explicitada diz respeito às situações neutras, descritas no fluxo de trabalho da seção 2.11, pois são casos em que o desenvolvedor devolve a tarefa ao administrador. Por padrão é usada a situação *Resolved*. Sem essa informação, a sincronização do SisGET acharia que deve modificar o responsável da tarefa, realizando as ações do caso #13 da figura 2.2.

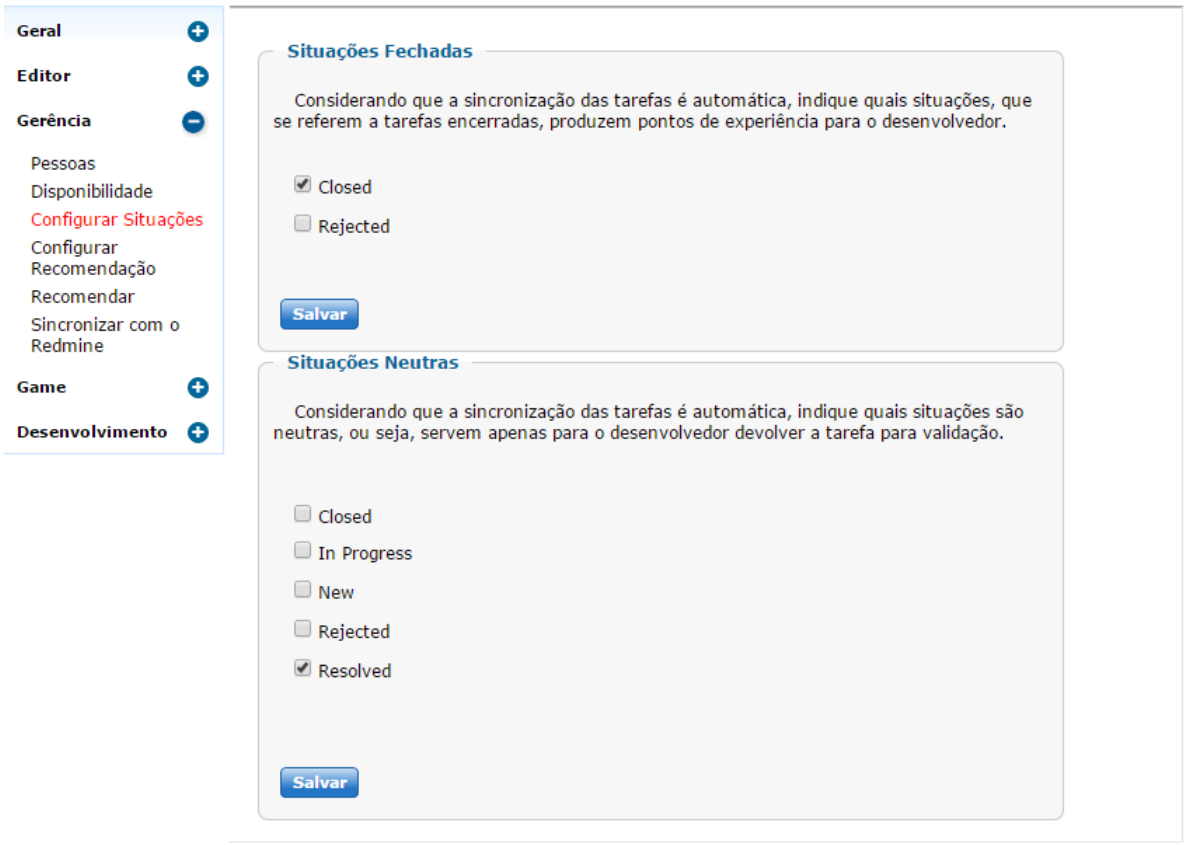

Figura 4.13: Tela Configurar Situações

– Seção Configurar Recomendação: para a recomendação das pessoas mais aptas à determinadas tarefas, é possível configurar os itens listados abaixo.

  – **Tolerância de nível**: quando não existe uma pessoa, cujo nível esteja entre os níveis inferior e superior recomendados da dificuldade da tarefa, a classificação **fortemente recomendados** ficará vazia. Para tentar evitar isso, o gerente pode configurar um limiar de aceitação para repescagem. Por exemplo, se a tarefa exige uma dificuldade em que os níveis recomendados são do 20 ao 30, configurando tal limiar de tolerância como 5, uma pessoa do nível 15 ou mais subiria para a classificação forte. Isso pode motivar desafios, melhorando a experiência e evolução mais rapidamente.
  – **Limite de tarefas simultâneas**: indica a quantidade máxima de serviço que uma pessoa pode ter alocada a si. Desenvolvedores com essa quantidade ou mais serão desconsiderados. Em tempo, esse limite serve apenas para desconsiderar pessoas na recomendação, o sistema não proibi o gerente de alocar tarefas acima de tal limite caso queira.
  – **Que gostam**: buscar desenvolvedores que gostam do conhecimento da tarefa (e não necessariamente sabem). Baseia-se nas preferências escolhidas da seção 4.1.1.
  – **Que sabem**: buscar desenvolvedores que sabem o conhecimento da tarefa (e não necessariamente gostam). Baseia-se estritamente no histórico de desenvolvimento.

Além dessas configurações, existem dois tipo de prioridades de ordenamento de pessoas.

Ordem de Prioridade

- **Prioridade tipo 1** : ordenar por maior experiência e, entre aqueles com mesma experiência, trazer os menos atarefados à frente.
- **Prioridade tipo 2** : ordenar por menor número de tarefas e, entre aqueles com mesmo número de tarefas, trazer os de maior experiência à frente.

– Seção Recomendação: nesta tela, o administrador seleciona alguma tarefa entre as importadas e recebe sugestões de quais pessoas são mais qualificadas para realizá-la, considerando a configuração escolhida.

Existem três tipos de classificação sobre as opções **gostar** e **saber**, que levam em conta as preferências e habilidades individuais, são elas:

- **Gosta e sabe (1)**
- **Gosta, porém não sabe (2)**
- **Não gosta, porém sabe (3)**

A classificação **não gosta e não sabe** não é utilizada, visto que não há interesse em sugerir alguém nessa situação. Na tabela abaixo, são mostradas as categorias que serão populadas caso existam pessoas para elas, dependendo da configuração. Vale notar que o *Sim* tem prioridade sobre o *Não*, ou seja, se uma das opções estiver marcada como *Sim* e a outra como *Não*, isso não impede a busca pela opção que é *Sim*.

Que gostam	Que sabem	Classificações Populadas
Sim	Sim	(1), (2), (3)
Sim	Não	(1), (2)
Não	Sim	(1), (3)
Não	Não	-

Tabela 4.1: Tabela de criação das categorias

Fonte: Elaborado pelo autor

Dentro de cada classificação estão três pertinências: *fortemente recomendados*, *fracamente recomendados* e *para emergências*. Na primeira estarão pessoas, cujo nível está de dentro do intervalo recomendado para a dificuldade; Na segunda, pessoas cujo nível está abaixo do limite inferior do intervalo; Na última, pessoas cujo nível está acima do limite superior do intervalo. Apesar desse último estar num nível mais avançado, deve-se evitar usá-lo quando há indicação na lista *fortemente recomendados*, pois assim o grupo evolui em conjunto. No trecho de código fonte abaixo, é mostrado o método da classe TarefaController que utiliza outros métodos para compor a lista dos mais indicados a uma atividade.

```
1  public function actionRecomendacaoTarefa() {
2
3    $objTar=new Tarefa;
4    $objHab=new Habilidade;
5    $objDif=new Dificuldade;
6
7    if(!isset($_POST["tipoFiltro"]) || $_POST["tipoFiltro"]=="todas"
       )
8      $listaTarefas=$objTar->listarTodasTarefas();
9    else
10     $listaTarefas= $objTar->buscarTarefasSemDesenvolvedor();
11
12   if(isset($_POST["idTarefa"]) && $_POST["idTarefa"] != 0) {
13
14     $dadosTarefa=
15       $objTar->buscarDadosTarefa($_POST["idTarefa"]);
16     $dadosHab=
17       $objHab->buscarDadosHabilidade($dadosTarefa["id_habilidade"
       ]);
18     $dadosDif=
19       $objDif->buscarDadosDificuldade($dadosTarefa["id_dificuldade
       "]);
20
21     $objPessHab=new PessoaHabilidade;
22     $objPessTar=new PessoaTarefa;
23
24     /*Trazer pessoas disponiveis de acordo com seus gostos e
       conhecimentos.
25     Monta um array onde o indice eh o id da tarefa que aponta pra
       um array de id pessoas */
26     $pessDisp=$objPessHab->buscarPessoasDisponiveis($dadosTarefa);
27
28
29     /*Recomendar considerando a dificuldade vs nivel (tentar
       aproximar iniciante com facil | expert com dificil) */
30     $pessDisp=
31       $objPessHab->aproximarDificuldadeNivel($pessDisp,
       $dadosTarefa);
32
33
34     /*Recomendar considerando a quantidade de tarefa (tentar
       aproximar recomendacaoo forte com pouco atarefado |
       recomendacaoo fraca com muito atarefado) */
35     $pessDisp=
36       $objPessTar->removerAtarefadoRecomendacao($pessDisp);
37
38     /*Antes de mostrar: ordenar (dentro de cada pertinencia) cada
       um dos arrays das estruturas de dados (categoria e pertinencia
       da recomendacao). Pela experiencia, quantidade de tarefas
       simultaneas, entre outros que venham a ser criados. */
39     $pessDisp=$objPessTar->ordenarPessoasCriterios($pessDisp);
40   }
41
42   $this->render("recomendacao", array(
43
44     "tarefasDisponiveis"=>$listaTarefas
45     , "filtro"=>(isset($_POST["tipoFiltro"])?$_POST["tipoFiltro"]:
```

```
    '')
46      , "idTarefa"=>(isset($dadosTarefa["id_tarefa"])?$dadosTarefa["
    id_tarefa"]:null)
47      , "pessoasDisponiveis"=>(isset($pessDisp)?$pessDisp:null)
48      , "dadosTarefa"=>(isset($dadosTarefa)?$dadosTarefa:null)
49      , "dadosHabilidade"=>(isset($dadosHab)?$dadosHab:null)
50      , "dadosDificuldade"=>(isset($dadosDif)?$dadosDif:null)));
51  }
```

O algoritmo da recomendação é feito em 3 passos. Alguns trechos de códigos-fontes são mostrados a seguir.

1º Passo: buscar todas as pessoas disponíveis. Isso é feito levando-se em consideração as preferências marcadas (gostar) ou ter no histórico a realização com sucesso de tarefas no conhecimento necessário (saber). Tudo isso, claro, depende da configuração estabelecida pelo administrador.

```
1  public function buscarPessoasDisponiveis($dadosTarefa) {
2
3    /*
4     GOSTA   SABE
5     V       V
6     V       F
7     F       V
8     F       F
9     */
10
11   $valoresLogicos=
12     array('gosta_sabe'=>array(true, true)
13       , 'gosta_nao-sabe'=>array(true, false)
14       , 'nao-gosta_sabe'=>array(false, true)
15       , 'nao-gosta_nao-sabe'=>array(false, false));
16
17   $arrPes=array();
18   $sql="select considerar_gostar, considerar_saber from
       configuracao";
19   $configuracao=Yii::app()->db->createCommand($sql)->queryRow(true
       );
20
21   $considerarGostar=$configuracao["considerar_gostar"];
22   $considerarSaber=$configuracao["considerar_saber"];
23
24   foreach($valoresLogicos as $cat=>$valores) {
25
26     $gosta=$valores[0];
27     $sabe=$valores[1];
28     $sql=null;
29
30     if(($considerarGostar || $considerarSaber) && $gosta && $sabe)
       {
31       /*traz quem gosta e sabe */
32
33       $sql="select PR.id_pessoa
34         from preferencia PR
35         join pessoa_habilidade PH on (PR.id_pessoa=PH.id_pessoa
36                         and PR.id_habilidade=PH.id_habilidade)
37         left join pessoa PE on (PE.id_pessoa=PR.id_pessoa)
38         where PE.data_saida is null and PR.id_habilidade=:idHab";
39
40     }elseif($considerarGostar && $gosta && !$sabe) {
41       /*traz quem gosta, porem nao sabe */
42
43       $sql="select distinct PR.id_pessoa
44         from preferencia PR
45         left join pessoa PE on (PE.id_pessoa=PR.id_pessoa)
46         where
47           not exists (select 1 from pessoa_habilidade
48                 where id_habilidade=:idHab and id_pessoa=PR.
     id_pessoa)
49           and PE.data_saida is null and PR.id_habilidade=:idHab";
50
51     }elseif($considerarSaber && !$gosta && $sabe) {
52       /*traz quem nao gosta, porem sabe*/
53
```

```
54          $sql="select distinct PH.id_pessoa
55            from pessoa_habilidade PH
56            left join pessoa PE on (PE.id_pessoa=PH.id_pessoa)
57            where not exists
58              (select 1 from preferencia
59                where id_habilidade= :idHab and id_pessoa=PH.id_pessoa
    )
60              and PE.data_saida is null and PH.id_habilidade=:idHab";
61
62       }elseif(!$gosta && !$sabe) { /* apenas pula*/
63         continue;
64       }
65
66       if($sql !== null)
67         $arrPes[$dadosTarefa["id_tarefa"]][$cat]=
68           Yii::app()->db
69           ->createCommand($sql)
70           ->queryColumn(array(':idHab'=>$dadosTarefa["id_habilidade"
    ]));
71     }
72
73     return $arrPes;
74   }
```

2º Passo: as pessoas disponíveis são classificadas de acordo com seu nível e a dificuldade estabelecida, ou seja, iniciantes são ordenados para cima numa tarefa fácil, ao passo que serão ordenados para baixo nas difíceis. Pela mesma lógica, numa tarefa difícil, os mais experientes serão mostrados com preferência em relação aos menos experientes.

```
1  public function aproximarDificuldadeNivel($pessDisp, $dadosTarefa)
       {
2
3    $objTar=new Tarefa;
4    $objDif=new Dificuldade;
5    $arrRecom=array();
6
7    $idDif=$dadosTarefa["id_dificuldade"];
8    $idHab=$dadosTarefa["id_habilidade"];
9    $dadosDif=
10     $objDif->buscarDadosDificuldade($idDif);
11
12   /*1a Etapa: criar uma estrutura de dados onde, para cada tarefa,
       para cada categoria (gosta_sabe, gosta_nao-sabe, nao-
      gosta_sabe), testa-se o nivel da pessoa na habilidade da
      tarefa levando em conta o intervalo de niveis inferior e
      superior recomendado das dificuldades. Com isso, classifica-se
       da forma abaixo.
13   Se o nivel da pessoa esta DENTRO da faixa de niveis recomendada
      -> indica com nivel FORTE de recomendacao
14   Se o nivel da pessoa esta ABAIXO da faixa de niveis recomendada
      -> indica com nivel FRACO de recomendacao
15   Se o nivel da pessoa esta ACIMA da faixa de niveis recomendada
      -> indica com nivel CURINGA de recomendacao (sera usado na
      falta do melhor caso e em detrimento do pior caso)
16
```

```
17  2a Etapa: rodar um segundo turno, para refazer (ou nao) os
      niveis de recomendacao, pois esses niveis tambem devem levar
      em conta o fato de nao haver pessoas na classificacao mais
      indicada.
18  */
19
20  /*-----1a Etapa -------*/
21  foreach($pessDisp as $idTar=>$arrCateg) {
22
23    foreach($arrCateg as $cat=>$arrPes) {
24
25      foreach($arrPes as $idPessoa){
26
27        $dadosNvl=
28          $this->buscarNivelHabilidade($idPessoa, $idHab);
29
30        /* pessoa que nao tem registro de habilidade (vindo da
      categoria gosta_nao-sabe) */
31        if($dadosNvl===null) { continue; }
32
33        $nvlPesHab =
34        (is_null($dadosNvl["nivel"])?0:$dadosNvl["nivel"]);
35        $exp =
36        (is_null($dadosNvl["pontos_exp"])?0:$dadosNvl["pontos_exp"
      ]);
37
38        $arrayAttr=
39          array('idPessoa'=>$idPessoa
40          , 'nivel'=>$nvlPesHab
41          , "exp"=>$exp);
42
43        if($nvlPesHab>=$dadosDif["nivel_ind_inf"]
44          && $nvlPesHab<=$dadosDif["nivel_ind_sup"]) {
45          $arrRecom[$idTar][$cat]["recomendacao_forte"][]=
46            $arrayAttr;
47        }elseif($nvlPesHab < $dadosDif["nivel_ind_inf"]){
48          $arrRecom[$idTar][$cat]["recomendacao_fraca"][]=
49            $arrayAttr;
50        }elseif($nvlPesHab>$dadosDif["nivel_ind_sup"]){
51          $arrRecom[$idTar][$cat]["recomendacao_curinga"][]=
52            $arrayAttr;
53        }
54      }
55    }
56  }
57
58  /*------2a Etapa -------*/
59
60  $sql="select nivel_abaixo_aceitavel from configuracao";
61  $nvlAbAceit=
62    intval(Yii::app()->db->createCommand($sql)->queryScalar());
63
64  foreach($pessDisp as $idTar=>$arrCateg) {
65
66    $dadosTarefa=$objTar->buscarDadosTarefa($idTar);
67    $idDif=$dadosTarefa["id_dificuldade"];
68    $idHab=$dadosTarefa["id_habilidade"];
69    $dadosDif=$objDif->buscarDadosDificuldade($idDif);
```

```
70
71      if(empty($arrRecom[$idTar]))
72        continue;
73
74      foreach($arrCateg as $cat=>$tiposRecom) {
75
76        /*Resgata as pessoas que estao na recomendacao fraca, mas
      que estao dentro do limiar aceito pela tolerancia. Caso queria
       que somente se faca isso quando nao ha pessoas na
      recomendacao forte, inclua no if abaixo o seguinte:
77        //empty($arrRecom[$idTar][$cat]["recomendacao_forte"]) */
78
79        if(!empty($arrRecom[$idTar][$cat]["recomendacao_fraca"])
80          && empty($arrRecom[$idTar][$cat]["recomendacao_forte"])) {
81
82          foreach($arrRecom[$idTar][$cat]["recomendacao_fraca"] as
      $i=>$dadosPes) {
83
84            $nPes=intval($dadosPes["nivel"]);
85            $nDifer=intval($dadosDif["nivel_ind_inf"]) - $nvlAbAceit
      ;
86
87            if($nPes>=$nDifer) {
88              /*significa que a pessoa 'fraca' pode ser recomendada.
      */
89
90              $arrRecom[$idTar][$cat]["recomendacao_forte"][]=
      $dadosPes;
91              unset($arrRecom[$idTar][$cat]["recomendacao_fraca"][$i
      ]);
92            }
93          }
94        }
95      }
96    }
97    return $arrRecom;
98  }
```

3º Passo: é feita a ordenação dentro de cada pertinência, considerando-se o que foi configurado como prioridade (se experiência ou quantidade de tarefas).

```
1  public function removerAtarefadoRecomendacao($pessDisp) {
2
3    $maxTarSimul=$this->maxTarefasSimultaneas();
4
5    foreach($pessDisp as $idTar=>$arrCateg) {
6
7      foreach($arrCateg as $cat=>$tiposRecom) {
8
9        foreach($tiposRecom as $pertin=>$arrPes) {
10
11
12         foreach($arrPes as $i=>$pess){
13
14           $dadosOcup=
15             $this->verificarOcupacaoDesenvolvedores($pess["
```

```
    idPessoa"]);
16
17              $nroTarAtuais = intval($dadosOcup["tarefas_ativas"]);
18
19              if($nroTarAtuais>intval($maxTarSimul)) {
20                unset($pessDisp[$idTar][$cat][$pertin][$i]);
21                if(count($pessDisp[$idTar][$cat][$pertin])==0)
22                  unset($pessDisp[$idTar][$cat][$pertin]);
23              } else {
24                $pessDisp[$idTar][$cat][$pertin][$i]["nome_pessoa"] =
25                $dadosOcup["nome_pessoa"];
26                $pessDisp[$idTar][$cat][$pertin][$i]["tarefas_ativas"]
      =
27                $dadosOcup["tarefas_ativas"];
28              }
29            }
30          }
31        }
32      }
33
34      return $pessDisp;
35  }
36
37  public function ordenarPessoasCriterios($pessDisp) {
38
39      foreach($pessDisp as $idTar=>$arrCateg) {
40
41        foreach($arrCateg as $cat=>$tiposRecom) {
42
43          foreach($tiposRecom as $pertin=>$arrPes) {
44
45            @usort($pessDisp[$idTar][$cat][$pertin]
46                , array('PessoaTarefa','metodoOrdenador'));
47          }
48        }
49      }
50      return $pessDisp;
51  }
52
53  public static function metodoOrdenador($a, $b) {
54
55      $sql = "select tipo_prioridade from configuracao";
56      $tipoPrioridade=
57        Yii::app()->db->createCommand($sql)->queryScalar();
58
59      /* Se quer o criterio...
60       *
61       * do maior pro menor -> b - a
62       * do menor pro maior -> a - b
63       *
64       */
65
66      if($tipoPrioridade=="1") {
67
68        if($a["exp"]==$b["exp"]){
69          return $a["tarefas_ativas"] - $b["tarefas_ativas"];
70        }
71        return $b["exp"] - $a["exp"];
```

```
72      }
73
74      elseif($tipoPrioridade=="2") {
75
76        if($a["tarefas_ativas"]==$b["tarefas_ativas"]){
77          return $b["exp"] - $a["exp"];
78        }
79        return $a["tarefas_ativas"] - $b["tarefas_ativas"];
80      }
81  }
```

Como exemplo de uso, considere uma tarefa sobre Bootstrap3, cuja dificuldade é *muito fácil* - dificuldade ideal para pessoas do nível 8 ao 25. Um cenário possível é exposto na figura abaixo.

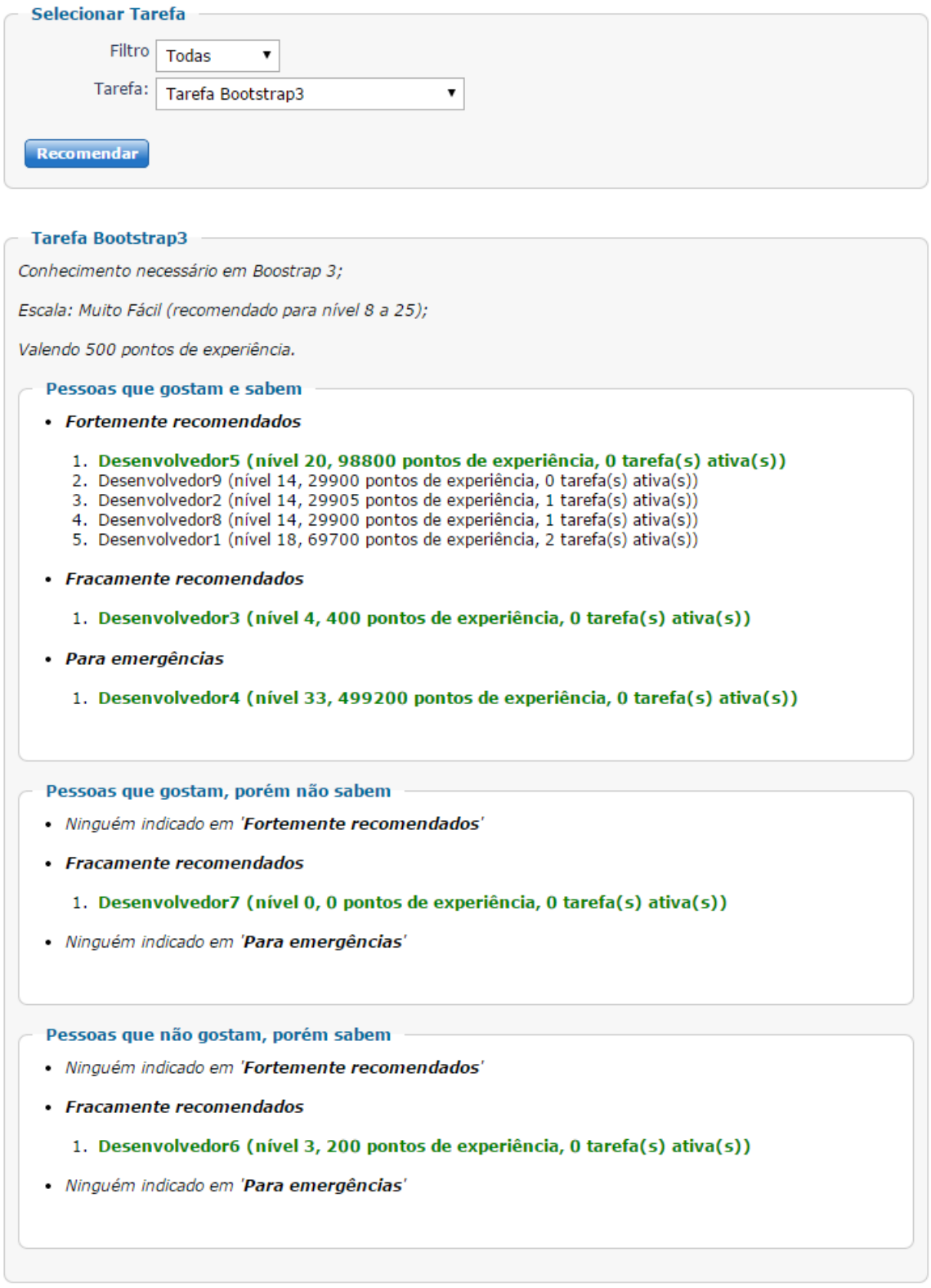

Figura 4.14: Tela Recomendação

Os itens **gostar e saber** são autoexplicativos, dependendo da configuração escolhida se considera preferência e/ou histórico. Da mesma forma, o item **limite de tarefas simultâneas** é trivial.

Provavelmente os itens mais complexos de entender num primeiro momento sejam os de **tolerância de nível** e a prioridade de ordenamento, que depende do **número de tarefas** e dos **pontos de experiência**.

Para esclarecer o primeiro, tome como exemplo o seguinte fragmento da imagem 4.14 com tolerância de nível configurada como 2.

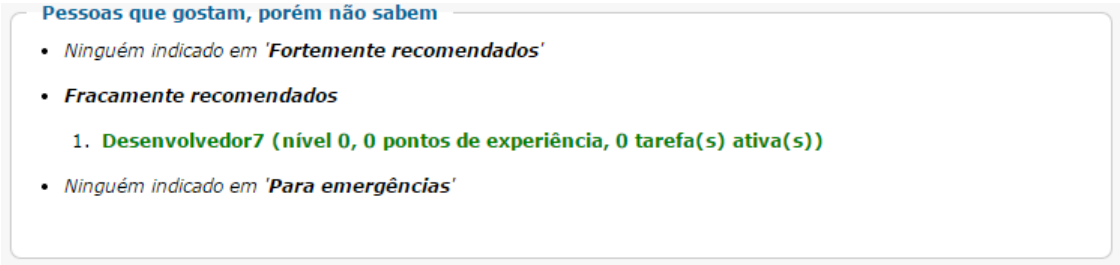

Figura 4.15: Tolerância de Nível

Considerando que o Desenvolvedor7 não possui experiência, logo é do nível 0, ele será listado na classificação fraca por estar num nível abaixo do limite inferior da dificuldade. Porém, modificando a tolerância para 8, o 'Desenvolvedor7' será realocado para a classificação forte, pois o nível dele será maior ou igual ao limite inferior da dificuldade da tarefa menos a tolerância, ou seja, de forma matemática, 0>=(8-8).

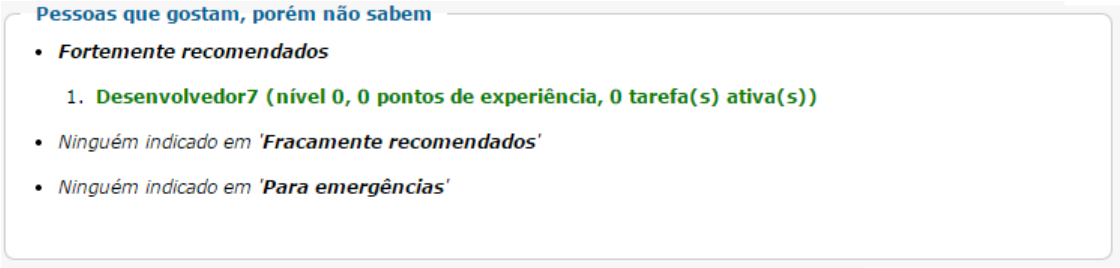

Figura 4.16: Tolerância de Nível

Para esclarecer a ordenação e suas prioridades, foi criado o seguinte cenário.

- Desenvolvedor1 (nível 18, 69700 pontos de experiência, 2 tarefa(s) ativa(s))
- Desenvolvedor2 (nível 14, 29905 pontos de experiência, 1 tarefa(s) ativa(s))
- Desenvolvedor3 (nível 4, 400 pontos de experiência, 0 tarefa(s) ativa(s))
- Desenvolvedor4 (nível 33, 499200 pontos de experiência, 0 tarefa(s) ativa(s))
- Desenvolvedor5 (nível 20, 98800 pontos de experiência, 0 tarefa(s) ativa(s))
- Desenvolvedor8 (nível 14, 29900 pontos de experiência, 1 tarefa(s) ativa(s))
- Desenvolvedor9 (nível 14, 29900 pontos de experiência, 0 tarefa(s) ativa(s))

Caso se configure como prioridade 1 - *ordenar por maior experiência e, entre aqueles com mesma experiência, trazer os menos atarefados à frente* - a lista de recomendação terá a seguinte ordenação.

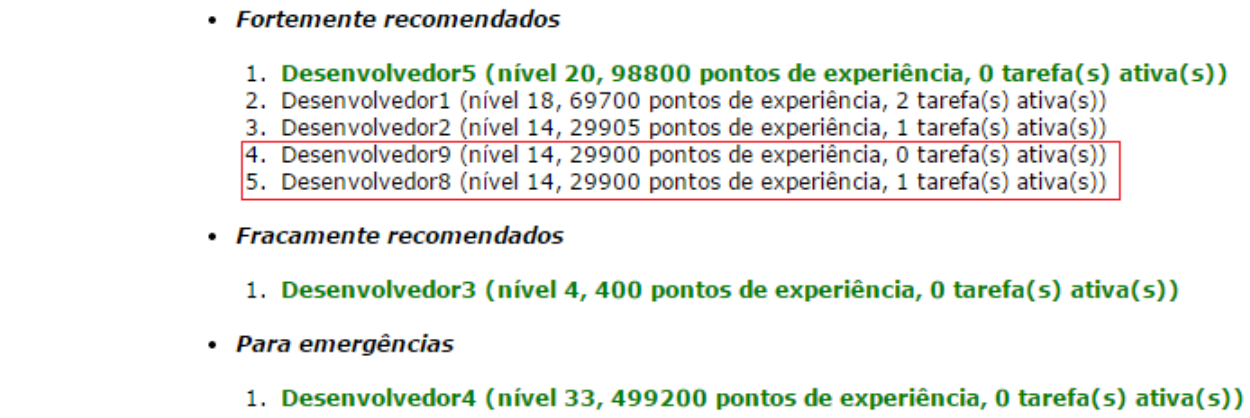

Figura 4.17: Ordenação com Prioridade Tipo 1

Conforme destacado pelo retângulo, os dois desenvolvedores com a mesma experiência foram ordenados de acordo com a quantidade de serviço que têm alocado no momento.

Finalmente, configurando a prioridade do tipo 2 - *ordenar por menor número de tarefas e, entre aqueles com mesmo número de tarefas, trazer os de maior experiência à frente* - o resultado apresentado será o que segue.

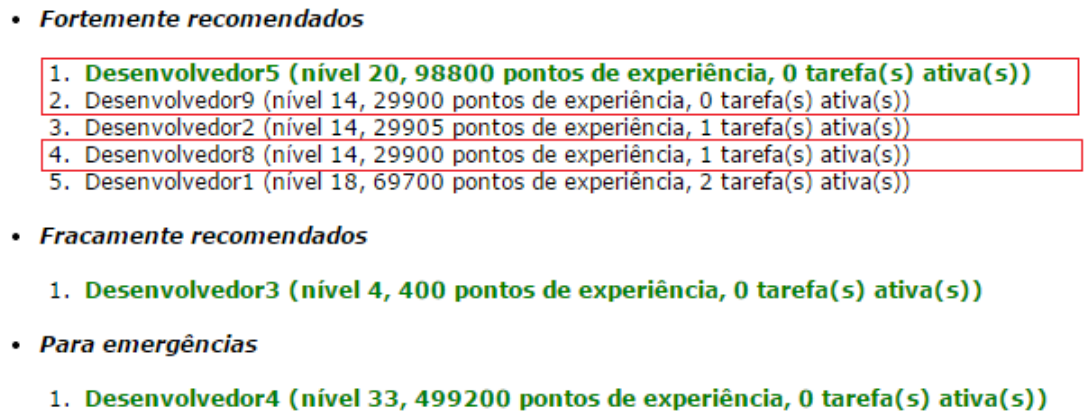

Figura 4.18: Ordenação com Prioridade Tipo 2

– Seção Sincronizar com o Redmine:conforme visto na figura 2.2, somente uma ação precisa da intervenção de um administrador, pois é preciso informar qual a dificuldade estimada, qual a habilidade necessária e quantos pontos de bonificação serão concedidos. Existe um controlador no SisGET que verifica, de tempo em tempo, se existe tarefas a serem importadas, em caso positivo, um aviso deverá ser mostrado, com o intuito de manter os dois sistemas alinhados continuamente, além disso, esse mesmo controlador executa todas as outras ações listadas como automática.

  Nesta tela, estarão listadas todas as tarefas que foram criadas no Redmine e precisam ser importadas. O administrador deve informar qual o conhecimento que julga ser útil para a resolução e quantos pontos ela proporcionará de bonificação ao ser finalizada corretamente.

  Caso a atribuição já tenha sido feita no redmine, no momento de salvar a tarefa, essa atribuição será automática se tal pessoa existe no SisGET também (verificado pelo campo id_pessoa_redmine da tabela pessoa), caso contrário, a tarefa não será mostrada para importar.

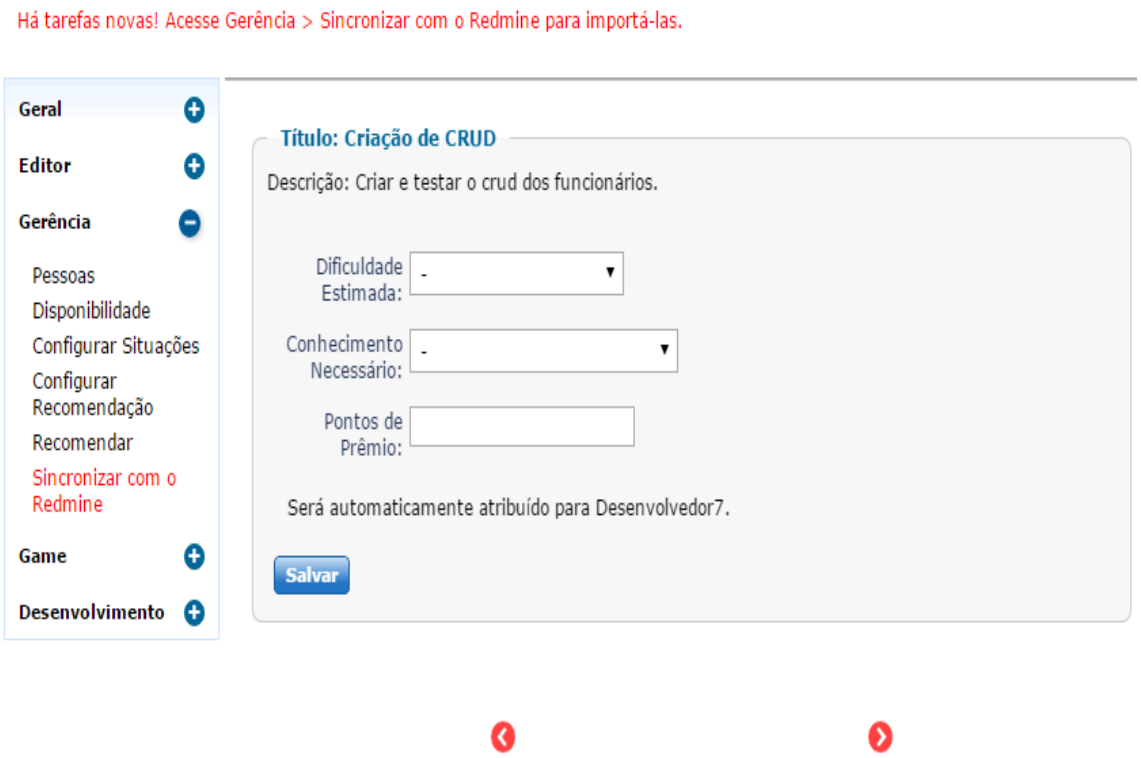

Figura 4.19: Tela de Sincronização

4.2.3 Menu Jogo

No menu Jogo, um gerente pode acessar estatísticas sobre a comunidade, criar novos personagens, conceder prêmios e fazer alterações nos níveis dos desenvolvedores.

– Seção Novo Participante: o cadastro de uma pessoa requer nome, data de entrada e o papel dela no sistema (administrador ou usuário convencional). O nome de usuário também será criado e é feito automaticamente conforme o nome é digitado, por exemplo, o nome *João da Silva* teria como sugestão *jsilva*, que terá sua unicidade verificada após o campo nome perder o foco. Também pode ser feito o preenchimento da tabela com os horários, porém não é obrigatória. Para facilitar, podem ser utilizadas as expressões =seg, =ter, =qua, =qui, =sex, =sab ou =dom para igualar a coluna àquela da expressão. Na figura 4.20, quarta-feira será preenchida igual à terça-feira. É necessário informar também o identificador dessa pessoa no redmine, caso se queira que todas as sincronizações sejam feitas automaticamente.

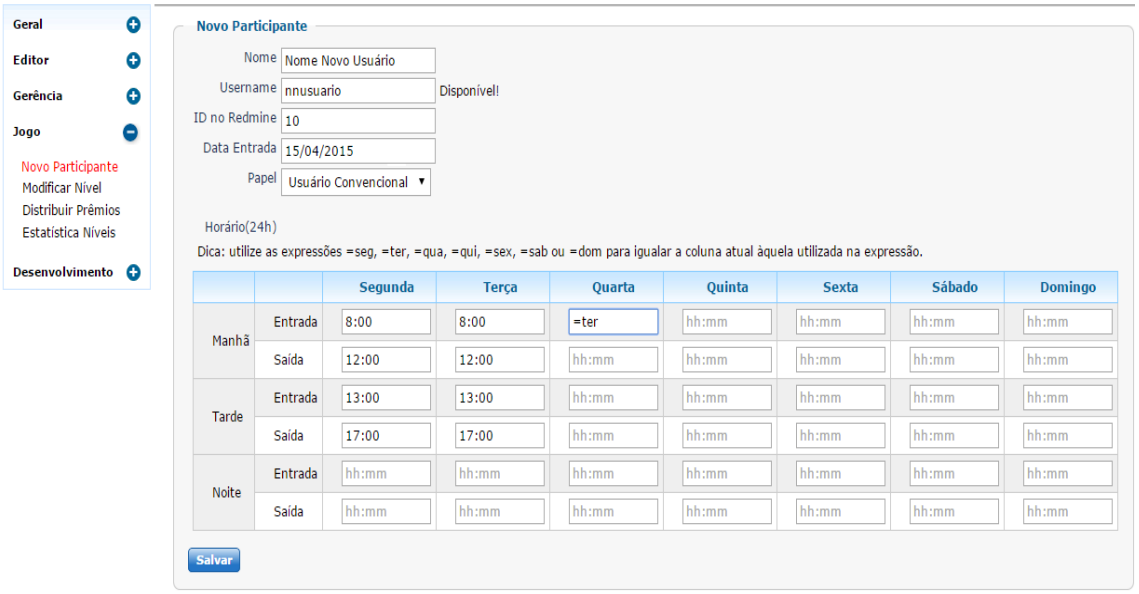

Figura 4.20: Cadastro de Novo Participante

– Seção Modificar Nível: existe a possibilidade do administrador achar conveniente modificar o nível de alguma pessoa, por exemplo, caso ele perceba que um desenvolvedor tem mais capacidade de realizar tarefas mais complexas e não precise progredir com pontos das tarefas. Ao selecionar uma pessoa, os dados são carregados, mostrando o nível atual e com um botão para salvar.

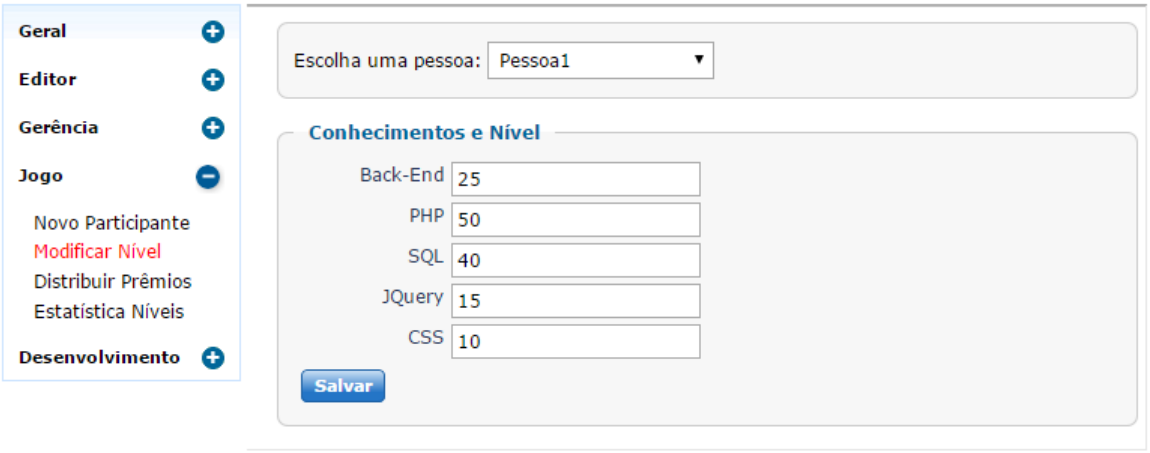

Figura 4.21: Modificação de Nível

– Seção Distribuir Prêmios: os pedidos de prêmios estão listados nessa área. Ao entrar o administrador vê cada um, agrupados por pessoa e suas quantidades. Basta clicar na varinha mágica para que os pontos correspondentes sejam descontados, nessa interação, espera-se que o administrador não faça mal uso da concessão, pois não há como validar se o prêmio realmente foi entregue.

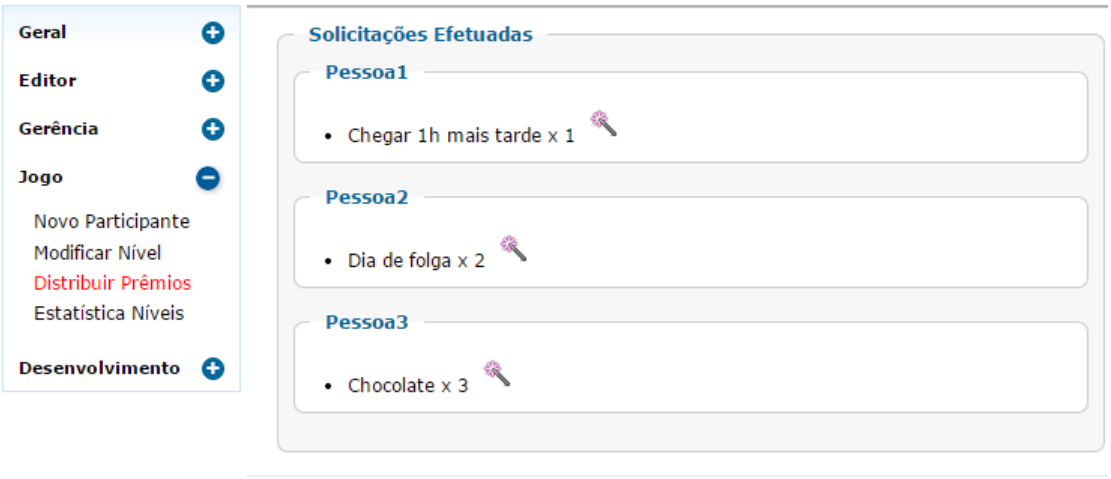

Figura 4.22: Distribuir Prêmios

– Seção Estatística Níveis: nessa seção, estão todas as habilidades cadastradas no sistema, onde é possível marcá-las para traçar um gráfico radar, onde estarão as médias dos níveis dos desenvolvedores em cada conhecimento.

  Um gráfico de radar (ou gráfico aranha) é um método para exibição de dados multivariados na forma de um gráfico bidimensional de variáveis quantitativas representada em eixos a partir do mesmo ponto, preferencialmente com, no mínimo, três eixos, para que forme uma figura geométrica.

  Um dos objetivos é incentivar a evolução conjunta, assim o conhecimento deve ser distribuído, compartilhado. Como o gráfico é montado com a média, a entrada de um membro com pouca sabedoria joga a média do grupo para baixo. Os gerentes devem perceber isso e incentivar a evolução desse novo membro, além de fazer um levantamento naquelas habilidades com média baixa e considerar capacitar o grupo.

  Para mostrar um exemplo, foram escolhidos PHP, SQL e JQuery, nos quais os níveis estão na figura 4.23, gerando o gráfico da figura 4.24 com as médias corretas.

	PHP	SQL	JQuery
Pessoa1	*50*	*40*	*15*
Pessoa2	-	-	*30*
Pessoa3	*25*	-	-
Pessoa4	*50*	*35*	-
Média	41.67	37.50	22.50

Figura 4.23: Desenvolvedores x Níveis

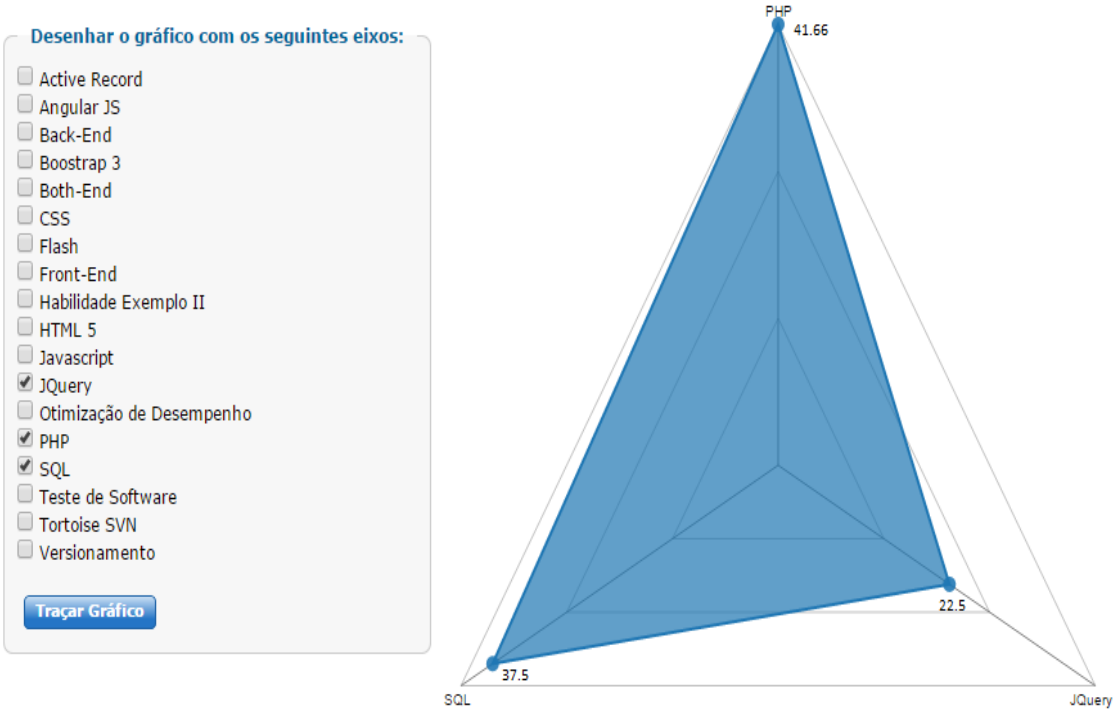

Figura 4.24: Gráfico Radar - PHP x SQL x JQuery

Ainda a título de demonstração, o gerador do gráfico consegue criar vários tipos de polígonos, tais como os da figura 4.25.

Figura 4.25: Polígonos do Gráfico Radar

4.2.4 Menu Desenvolvimento

Este menu é uma base com o protótipo de todas interações existentes no Redmine, pois a intenção é tornar o SisGET independente, que é uma das melhorias melhor discutida em 6.2. As ações que já estão funcionais são as seguintes:

- **Delegar Tarefa**: atribui a um desenvolvedor.
- **Dispensar Tarefa**: remove de um desenvolvedor.
- **Validar Tarefa**: permite aprovar a tarefa, fazendo o autor do trabalho ganhar os pontos de prêmio.
- **Fechar Tarefa**: simplesmente fecha, sem que gere pontuação.
- **Reabrir Tarefa**: pode ser uma tarefa que foi temporariamente suspensa ou que se constatou um problema após sua implementação. Se for esse caso, os pontos são removidos, pois o merecimento não foi válido na entrega anterior.

5 AVALIAÇÃO DO SISTEMA

Este Capitulo descreve o resultado da aplicação de um questionário de avaliação que foi utilizado para avaliar o sistema desenvolvido.

As *user storeis* especificadas na figura 3.1 são todas oferecidas pela ferramenta, cumprindo, portanto, os requisitos desejados. Um questionário sobre a usabilidade e com pedidos de sugestões e melhorias foi aplicado e respondido por quatorze pessoas, entre elas, dois administradores, que acessaram mais questões. Para o questionário sobre a visão dos administradores, serial ideal mais respostas, logo, esse questionário será aplicado em outros lugares onde o autor deseja implantar o uso desse sistema.

As questões objetivaram avaliar os seguintes tópicos.

– Facilidade de navegação: na questão 1, na qual se verifica a facilidade de encontrar rapidamente um conteúdo que se deseje acessar, olhando os nomes dados para as seções. Já na questão 4, o desenvolvedor deve indicar se sabe qual o objetivo de marcar suas preferências.

– Utilização dos formulários: na questão 2, verificou-se se o formulário de pedido de prêmios é de fácil entendimento. Foi sugerido que se modifique a utilização das checkbox por outra maneira de marcar a solicitação. Na questão 5, exclusiva para gerente, foi pedido que eles indiquem o gerenciamento de dados está simples, de modo geral para todo sistema. Nessa parte, foi sugerido que os gráficos sejam melhorados ou novos sejam incluídos. Nessa categoria, também estão as questões 6, 7, 8, 9 e 10, que contemplam os formulários para configuração do uso dos dados recebidos do Redmine, o formulário da configuração da recomendação e o de inclusão de um novo participante.

– Clareza das informações: a questão 3 auferiu se todos dados dispostos na tela da barra de progresso são inteligíveis e se o desenvolvedor sabe como os níveis são aumentados. Esse item também é avaliado pela questão 12, onde tenta-se verifcar se as informações dos gráficos são úteis.

A seguir, os resultados são mostrados graficamente.

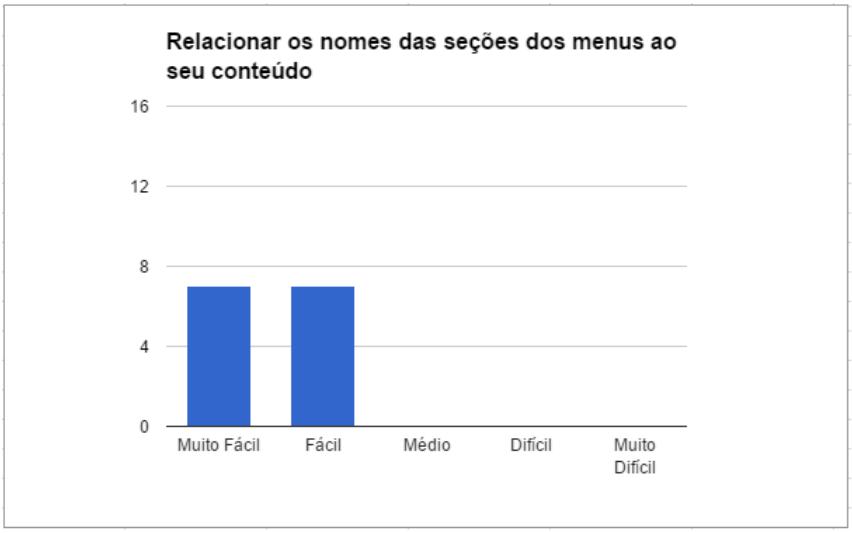

Figura 5.1: Questionário sobre usabilidade: questão 1

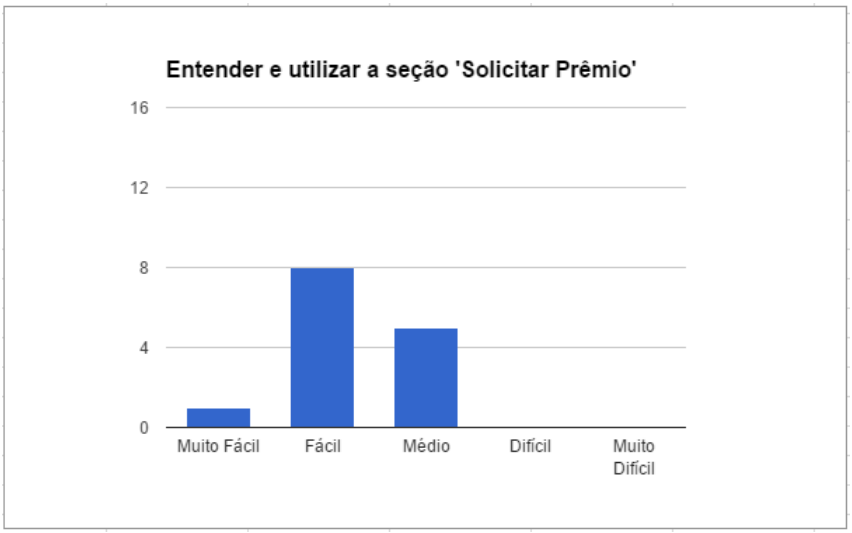

Figura 5.2: Questionário sobre usabilidade: questão 2

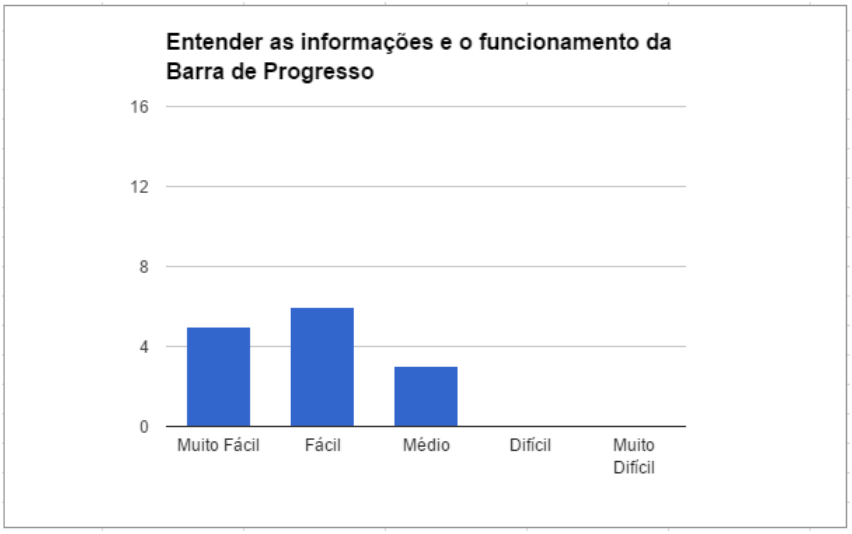

Figura 5.3: Questionário sobre usabilidade: questão 3

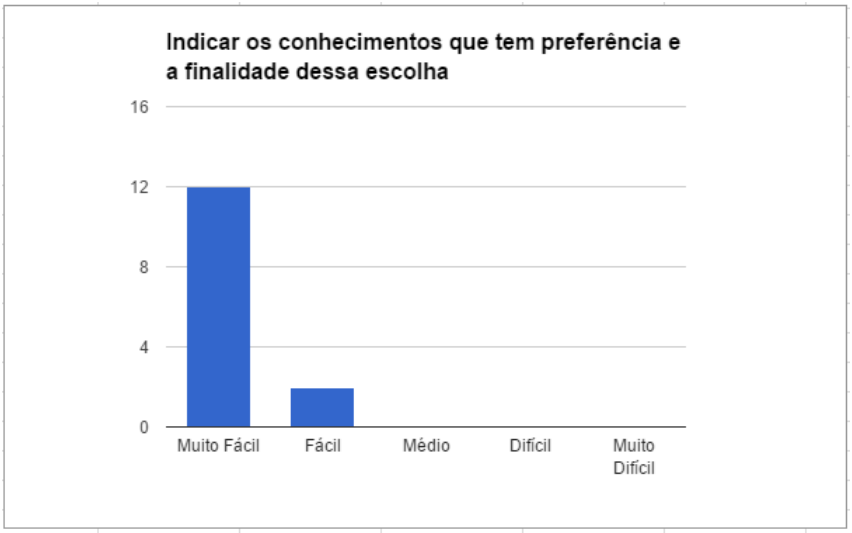

Figura 5.4: Questionário sobre usabilidade: questão 4

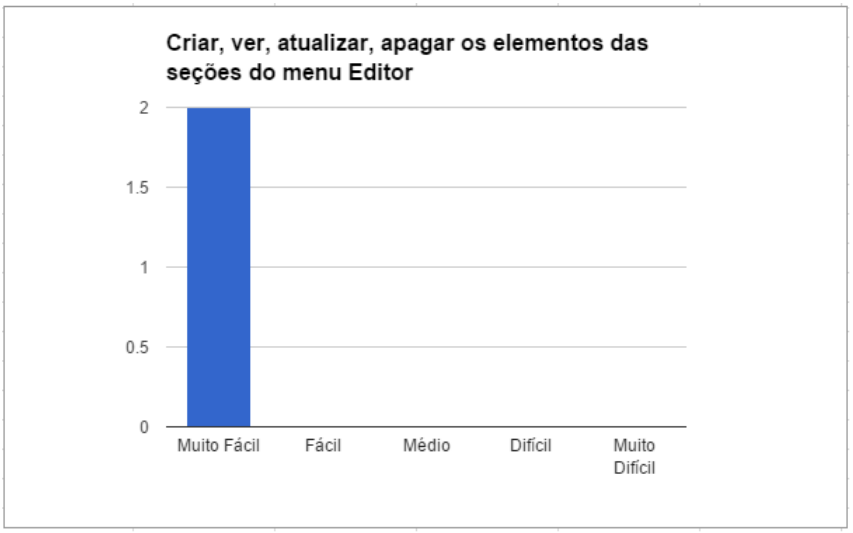

Figura 5.5: Questionário sobre usabilidade: questão 5

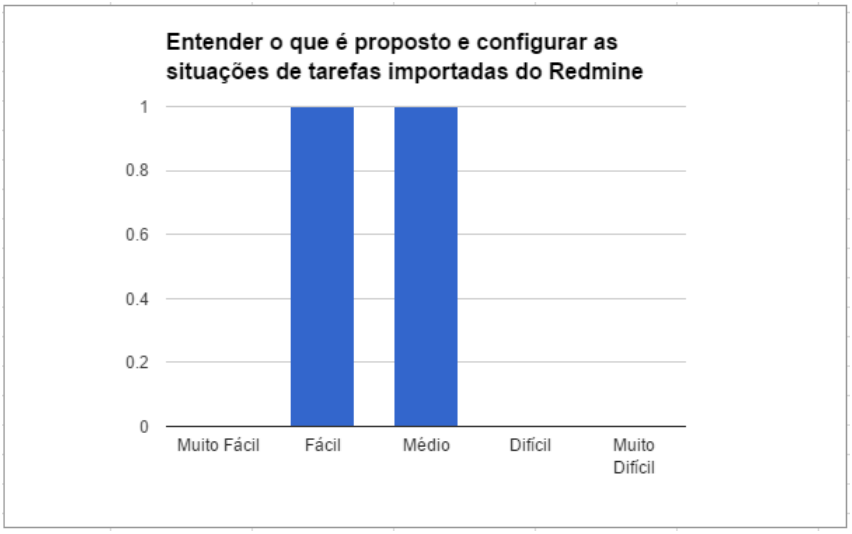

Figura 5.6: Questionário sobre usabilidade: questão 6

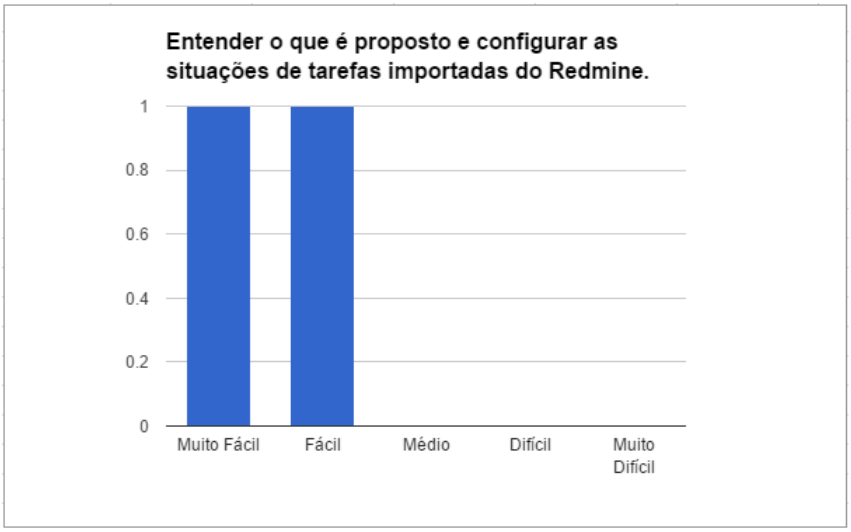

Figura 5.7: Questionário sobre usabilidade: questão 7

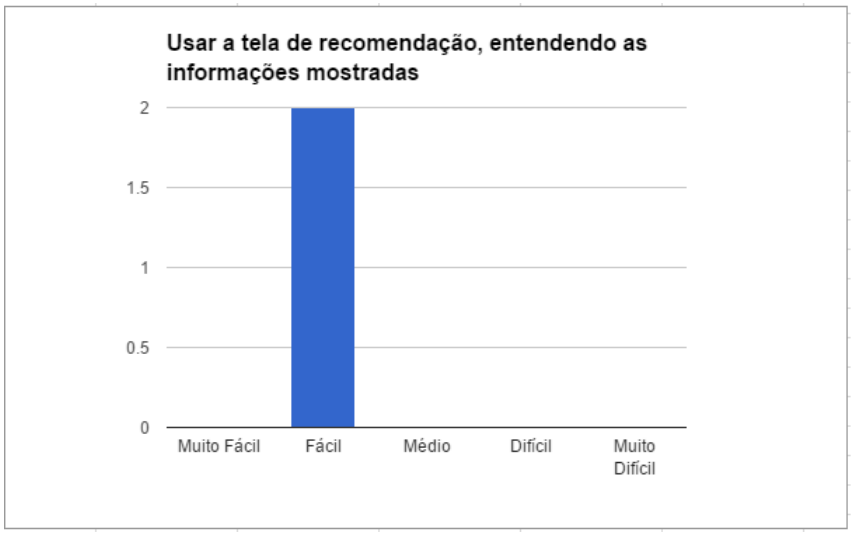

Figura 5.8: Questionário sobre usabilidade: questão 8

Figura 5.9: Questionário sobre usabilidade: questão 9

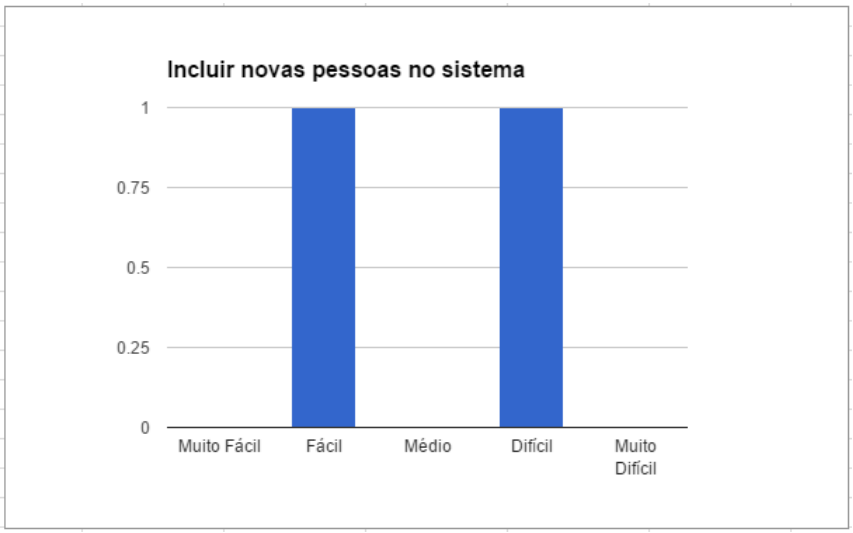

Figura 5.10: Questionário sobre usabilidade: questão 10

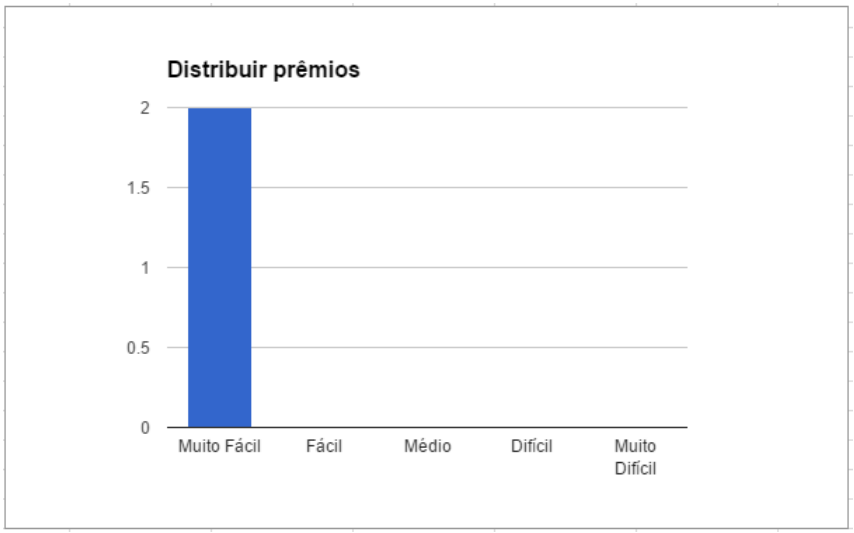

Figura 5.11: Questionário sobre usabilidade: questão 11

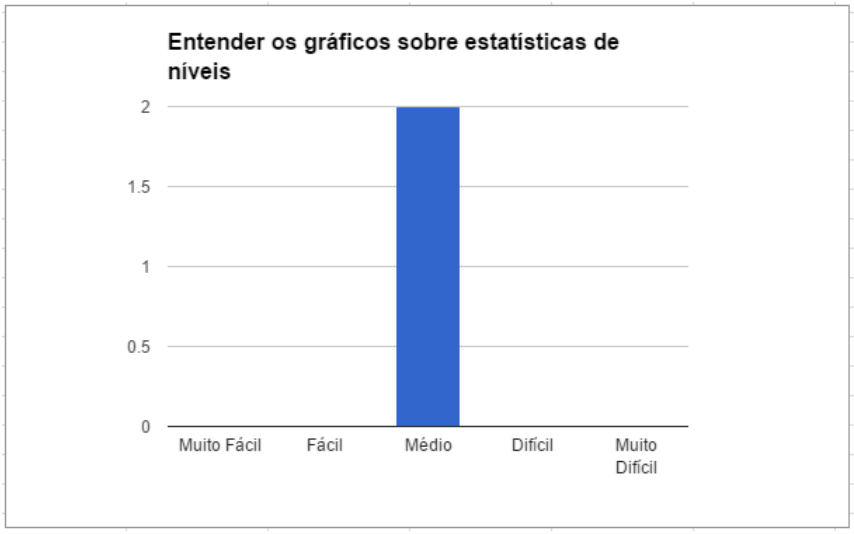

Figura 5.12: Questionário sobre usabilidade: questão 12

O resultado do questionário foi dentro do esperado, indicando melhorias que devem ser feitas, sugestões, e, de modo geral, a boa usabilidade do sistema, visto que houve marcação nas alternativas *difícil* ou *muito difícil* apenas na questão 10. Conforme as modificações sejam implementadas, novas questões podem ser abordadas, inclusive para outros grupos onde se utilize o sistema.

5.1 Comparação com Sistemas Similares

Alguns sites utilizam conceitos de *gamification* para motivar seus membros, por exemplo, o site *StackOverflow*[1] agrupa um enorme número de usuários, onde são feitas perguntas sobre os mais diversos assuntos (matemática, físíca, softwares, programação nas mais diversas linguagens, etc). Nele, perguntas bem formuladas e respostas construtivas são marcadas com +1 por qualquer outro membro (ou -1 para o contrário). Assim, membros que agregam valor ao conhecimento têm sua reputação aumentada. Quanto maior a reputação, mais distintivos a pessoa alcança. É um trabalho totalmente altruísta, o máximo que se pode ganhar é tornar-se um moderador para ajudar a melhorar a ferramenta, porém, mesmo assim, pode ser bastante viciante.

Sobre recomendação, geralmente essa é mais abstrata, tratando de cargos e experiências profissionais, como, por exemplo, o *LinkedIn*[2]. Restringindo para desenvolvimento de software, foram encontrados plataformas web educacionais, cujo foco é aprender e melhorar as habilidades em programação. Por exemplo, no Code Hunt
(Microsoft Research - Code Hunt 2014), a diversão é um componente vital para acelerar o aprendizado e manter o interesse no que poderia ser uma viagem longa e às vezes tediosa para a obtenção de uma determinada habilidade (Bishop 2014). Nele, todo desafio deve ser resolvido com um trecho de código, que será verificado pelo sistema, caso esteja errado, uma mensagem de erro é gerada, caso contrário, avança ao próximo passo, que será um pouco mais difícil.

Saindo da questão de aprendizado apenas, em 2012, Microsoft Visual Studio introduzu um plug-in que proporciona a premiação com distintivos, durante a fase de desenvolvimento. Além disso, é possível desafiar os colegas, para determinar quem receberá mais destaque e mostrar onde cada pessoa é melhor. Contudo a ideia é ter espírito esportivo, serve apenas por diversão (Studio 2012).

No site technologyadvice[3] existem diversos sistemas que usam *gamification*, nas mais diversas áreas, nos quais os elementos mais comuns são níveis, prêmios, distintivos, reputação, porém não se encontrou algum sistema que citou alguma capacidade de utilizar esses dados para recomendar os mais aptos para determinados desafios, o que por enquanto é um diferencial para o SisGET. Alguns sistemas mais ousados podem oferecer como premiação, aumento na participação dos lucros, cargos de chefia, dinheiro (seja virtual ou real), entre outros.

[1] https://stackoverflow.com/

[2] https://www.linkedin.com/nhome/

[3] http://technologyadvice.com/gamification/smart-advisor/ (Acessado em 15/04/2015)

6 CONCLUSÕES

O presente trabalho teve como objetivo o desenvolvimento e a implementação de um sistema web capaz de gerenciar pessoas e tarefas, além de recomendar integrantes, baseando-se nas suas características individuais de conhecimento e preferências, para auxiliar os administradores a otimizar tanto a distribuição de tarefas como o tempo necessário para cada uma ser realizada. Como complemento, elementos de *gamification* foram utilizados para tornar o serviço de desenvolvedores de software mais motivante.

Apesar de vários sistemas disponíveis no mercado, nas mais diversas áreas, utilizarem *gamification*, nem todos incorporam a recomendação em nível mais específico para tarefas e projetos de desenvolvimento de software, algo mais comum nesse sentido é apenas relacionar experiências profissionais com atribuições de um cargo, mas de forma mais genérica. Sendo assim, esse diferencial é interessante, considerando ainda que novos critérios podem ser formulados para compor a lista de pessoas mais aptas a determinado serviço.

O resultado do trabalho foi satisfatório, pois atendeu os requisitos, descritos nas *user stories*, e as expectativas quanto à recomendação. Além disso, a integração com a ferramenta *Redmine* dispensou a necessidade de alimentar dois sistemas, o que seria inconveniente.

6.1 Limitações

Apesar do resultado satisfatório, o sistema proposto ainda apresenta limitações. A principal nessa versão inicial é toda parte que depende do Redmine para funcionar, na tela de sincronização e importação, pois não conseguindo executar comandos REST para o REdmine, algumas falhas podem acontecer. Outro ponto é o SGBD, que roda em MySQL, sendo necessário reescrever as models para executarem corretamente as queries em outro SGBD, além da necessidade de recriar todas as tabelas, pois a DDL para instalação do sistema está escrita com os padrões do MySQL. Além disso, também é necessária a instalação do framework Yii para utilizar o sistema.

6.2 Trabalhos Futuros

Mesmo contribuindo com a área de desenvolvimento de software, após a implantação do sistema e durante a escrita deste trabalho, algumas melhorias e novas funcionalidades foram identificadas, tanto pelo autor como pelos usuários, por meio do questionário aplicado. Todas elas necessitam de mais definições sobre o impacto de sua implementação e

regras de negócio claras. Por enquanto, são as seguintes:

- Mapear uma tarefa para várias habilidades. Deste modo, o sistema pode recomendar mais de uma pessoa, compondo, assim, uma equipe de trabalho. Nesse caso a tarefa seria, na verdade, um projeto.

- Tornar independente do Redmine, realizando por conta própria a maior parte das funcionalidades existentes nele, as mais úteis, como geração de relatórios e melhores interfaces de comunicação remota entre desenvolvedor e gerente para manter documentado o processo.

- Utilizar mais informações para gerar uma recomendação, por exemplo, o horário de casa pessoa, pois é mais otimizado indicar alguém que está começando seu turno em vez de alguém que esteja terminando, dessa adiantaria pelo menos um dia de serviço. Outro critério poderia ser uma estimativa de tempo de realização da tarefa, feita pelo administrador. Com isso, seria possível ordenar à frente aqueles que tem um tempo médio de desenvolvimento similar ao tempo estimado.

- Criar títulos para pessoas, tais como, expert, gênio, mestre, guru. Esses títulos poderiam ser comprados com pontos. Fica a definir qual a utilidade de obter tais títulos, por exemplo, um tipo de título poderia aumentar alguma porcentagem nos pontos de prêmio de uma tarefa.

- Possibilitar ao desenvolvedor alguma configuração para quais recomendações ele quer aparecer (ou não aparecer).

- Implementar a ajuda entre membros da comunidade ou programação em pares, também gerando pontos (ideia sugerida no questionário).

- Conceder um acréscimo de pontos quando se acaba uma tarefa antes do previsto (ideia sugerida no questionário).

- Poderia existir desafios pré-prontos e, caso se consiga realizá-los corretamente, conceder pontos também (ideia sugerida no questionário).

- Poderia haver uma classificação melhor sobre gostar. Por exemplo, gosta muito, gosta, gosta pouco e tentar usar isso no algoritmo de recomendação (ideia sugerida no questionário).

- Os conhecimentos poderiam ter um mapeamento que indicasse quais outros conhecimentos são afins. Por exemplo, alguém que goste de tarefas de front-end também é indicado como alguém que gosta de tarefas de css ou javascript, pois essas duas estão dentro dá área de desenvolvimento front-end (ideia sugerida no questionário).

- Outra ideia sugerida é fazer uma reformulação da possibilidade dos usuários verem publicamente as barras de progresso de qualquer outra pessoa. É possível que uma pessoa sinta-se depreciada ou superior em relação aos outros, algo que deve ser evitado.

REFERÊNCIAS

[Bishop 2014]BISHOP, J. *Code Hunt*. 2014. Available from Internet: <http://blogs.msdn.com/b/msr_er/archive/2014/05/15/what-if-coding-were-a-game.aspx>.

[Cohn 2008]COHN, M. *User Story*. 2008. Available from Internet: <http://www.mountaingoatsoftware.com/blog/advantages-of-the-as-a-user-i-want-user-story-template>.

[Krasner e Pope 1988]KRASNER, G. E.; POPE, S. T. *A cookbook for using the model-view controller user interface paradigm in Smalltalk-80. J. Object Oriented Program.* [S.l.: s.n.], 1988. `http://www.ics.uci.edu/~redmiles/ics227-SQ04/papers/KrasnerPope88.pdf`,.

[Lesyuk 2013]LESYUK, A. *Mastering Redmine*. [S.l.]: Packt Publishing., 2013. ISBN 978-1-849519-14-4.

[Microsoft Research - Code Hunt 2014]MICROSOFT Research - Code Hunt. 2014. Available from Internet: <https://www.codehunt.com/>.

[MVC XEROX PARC 1978-79]MVC XEROX PARC 1978-79. Available from Internet: <http://heim.ifi.uio.no/ trygver/themes/mvc/mvc-index.html>.

[Redmine Project]REDMINE Project. Available from Internet: <http://www.redmine.org>.

[Simões 2012]SIMõES, J. *A Social Gamification Framework for a K-6 Learning Platform, Computers in Human Behavior*. 2012. 1–1 p.

[Studio 2012]STUDIO, M. V. *Visual Studio*. 2012. Available from Internet: <http://blogs.microsoft.com/blog/2012/01/18/visual-studio-achievements-program-brings-gamification-to-development/>.

[Winesett 2012]WINESETT, J. *Web Application Development with Yii and PHP*. [S.l.]: Packt Publishing., 2012. ISBN 978-1-84951-872-7.

[Zicherman 2012]ZICHERMAN, G. *Getting 3 Fs in Gamification*. Jan 2012. Available from Internet: <http://www.gamification. co/2012/01/19/getting-three-fs-in-gamification/>.

Anexo

O questionário, até a conclusão dessa monografia, foi relizado por quatorze pessoas, sendo apenas duas administradores do sistema. Dessa forma, existe uma carência de mais respostas de usuários desse papel para as questões 5 a 12, marcadas com um asterisco vermelho, pois são exclusivas dos administradores.

No questionário, as pessoas foram convidadas a avaliar a usabilidade do sistema, escolhendo uma alternativa para cada questão de acordo com a seguinte escala:

1 - Muito Fácil
2 - Fácil
3 - Médio
4 - Difícil
5 - Muito Difícil

Questão 1: Relacionar os nomes das seções dos menus ao seu conteúdo.

Questão 2: Entender e utilizar a seção 'Solicitar Prêmio'.

Questão 3: Entender as informações e o funcionamento da Barra de Progresso.

Questão 4: Indicar os conhecimentos que tem preferência e entender a finalidade dessa escolha.

* Questão 5: Criar, ver, atualizar, apagar os elementos das seções do menu Editor.

* Questão 6: Entender o que é proposto e configurar as situações de tarefas importadas do Redmine.

* Questão 7: Entender e configurar os itens do algoritmo de recomendação de pessoas.

* Questão 8: Usar a tela de recomendação, entendendo as informações mostradas.

* Questão 9: Importar tarefas no menu 'Sincronizar com o Redmine'.

* Questão 10: Incluir novas pessoas no sistema.

* Questão 11: Distribuir prêmios.

* Questão 12: Entender os gráficos sobre estatísticas de níveis.

Questão livre: Atualmente, cada tarefa concluída com sucesso gera pontos - que podem ser trocados por prêmios - e níveis - mostrados em barras de progressos. Além de níveis, pontos e prêmios, o que mais você acha interessante existir para motivar os desenvolvedores?

* Questão livre: Além dos critérios quantidade de tarefas e nível do desenvolvedor para listar os mais recomendados, que outros critérios você gostaria que fossem levados em conta?

Questão livre: Por favor, faça sugestões de melhorias para o sistema.