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

FREDERICO SOUZA GERMANO BETTING

Gerenciando Tarefas: Aplicativo *Kanban* para Gerenciamento de Projetos

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

Orientador: Prof. Dr. Marcelo Soares Pimenta

Porto Alegre
2014

AGRADECIMENTOS

Agradeço aos meus pais e ao meu irmão por sempre terem me apoiado durante todos os anos da minha vida.

À minha esposa e à minha filha por ficarem ao meu lado dias e noites, incentivando e ajudando a vencer mais uma etapa.

Aos meus professores e à UFRGS pelo conhecimento e todas as oportunidades dadas.

Ao Diogo Cravo e à Maitê Dupont pela enorme contribuição no início desse projeto.

Ao Jean Giacomin por sua contribuição na criação da identidade da ferramenta.

RESUMO

O gerenciamento de tarefas é um processo importante no contexto de um projeto. Encontrar e utilizar uma ferramenta que se adapte a sua realidade e que efetivamente ajude na sua organização é fundamental.

Este trabalho expõe a análise e o desenvolvimento de um aplicativo de gerenciamento de tarefas que atendam a todas as expectativas de um grupo específico, além de propor o *Kanban* como um mecanismo visual para exposição das tarefas cadastradas. Uma comparação do *software* desenvolvido e os já existentes no mercado também é realizada, enfatizando as principais características de cada um.

Palavras-chave: Gerenciamento de projetos. Gerenciamento de tarefas. *Kanban*. Desenvolvimento de *software*.

Managing tasks: Kanban software to project management

ABSTRACT

Task management is an important process in the context of a project. Finding and using a tool that suits project reality and in fact helps in the project organization is fundamental.
This paper presents the analysis and development of a management task application that meets all the expectations of a specific group, in addition to proposing Kanban as a visual mechanism to expose submitted tasks. A comparison of the developed and some existing software is also performed, emphasizing the main features of each one.

Keywords: Project management. Task management. Kanban. Software development.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CSV	Comma-separated Values
HTML	Hyper Text Markup Language
IC	Iniciação científica
MVC	Model-View-Controller
PHP	Hypertext Preprocessor
SQL	Structured Query Language ou Linguagem de Consulta Estruturada
STM	Smart Task Manager
SVN	Subversion
UFRGS	Univerdade Federal do Rio Grande do Sul

SUMÁRIO

1 INTRODUÇÃO

Gerenciar projetos pode ser uma tarefa difícil quando não há uma estrutura adequada, em que os processos estão mal definidos e o conjunto de tarefas está desorganizado. Para garantir o fluxo eficaz de desenvolvimento de um projeto, muitas vezes é necessário utilizar artifícios como instrumento para facilitar o cotidiano dos envolvidos.

Compreendendo tais dificuldades, um grupo de pesquisadores do Instituto de Informática da UFRGS identificou que, para administrar e organizar o andamento da pesquisa, os orientandos vinculados ao projeto e as tarefas associadas a eles, seria necessário utilizar alguma ferramenta que apresentasse campos específicos a fim de suprir as suas necessidades.

No mercado atual existe uma grande disponibilidade de ferramentas livres para gerenciamento de projetos. Em geral, essas ferramentas disponibilizam cadastro de usuários e seus contatos, cadastro de tarefas e a listagem das tarefas associadas ao projeto. Em alguns casos, também estão presentes mecanismos gráficos para mostrar o andamento do projeto.

Entretanto, nem sempre as informações disponíveis nessas ferramentas atendem totalmente às necessidades daqueles que precisam de um *software* de gerenciamento, sejam por campos obrigatórios existentes, porém, desnecessários ou pela ausência de algum dado específico. Muitas vezes a customização do aplicativo é imprescindível para que este possa realmente atender aos requisitos dos usuários.

Em outros casos são desenvolvidas ferramentas específicas com a finalidade de atender a todos os requisitos descritos pelo usuário. Nesse contexto, este trabalho tem como objetivo mostrar o processo de análise e desenvolvimento de um *software* de gerenciamento de projetos e tarefas que correspondam a todas expectativas específicas apresentadas pelos membros do grupo de pesquisa. Apresenta o *Kanban*, ferramenta da Qualidade utilizada como um mecanismo gráfico para expor as tarefas disponíveis, facilitando a visualização de escopo e prioridade de trabalho aos usuários do aplicativo, que aqui nomearemos “Smart Task Manager – STM”.

Dessa forma, esse trabalho está organizado conforme descrito abaixo:

No capítulo 2, serão apresentadas as necessidades demandadas, a contextualização da problemática e a motivação para desenvolvimento do trabalho.

No capítulo 3, são apresentadas as ferramentas de referência e também a comparação com o aplicativo desenvolvido.

No capitulo 4, é apresentado o aplicativo “STM” e as suas principais funcionalidades, além de descrever o processo de desenvolvimento.

No capítulo 5, é exposta a conclusão do trabalho.

2 GERENCIAMENTO DE TAREFAS DE BOLSISTAS

A partir da problemática apresentada por um grupo específico de pesquisadores do Instituto de Informática da UFRGS, iniciou-se o processo de investigação para entender quais informações seriam imprescindíveis para melhor organizar os dados de um projeto. Além disso, buscou-se identificar quais ferramentas livres, existentes no mercado, atenderiam a todos os requisitos requeridos por esse grupo.

O trabalho iniciou o seu desenvolvimento como um projeto de sala de aula, definido e idealizado por um professor e coordenador de pesquisa, em que o foco era a construção de uma ferramenta para o gerenciamento de bolsistas e suas tarefas.

2.1 Dificuldades no Gerenciamento de Tarefas de Bolsistas

Foi realizado um primeiro contato com pessoas envolvidas no projeto para entender a real necessidade do grupo. A proposta de aplicativo sugerida era desenvolver um ambiente de trabalho *web* para bolsistas de iniciação científica (IC) e que disponibilizasse as seguintes funcionalidades:

- Lista de tarefas: cadastro das tarefas a serem executadas pelos bolsistas;
- Lista de bolsistas: cadastro de todos os bolsistas como usuário do sistema;
- Contatos dos bolsistas: possibilidade de adição de múltiplos contatos;
- Planilha de horários: campo para adição de intervalos de horários de projeto dos bolsistas;

Tais informações eram necessárias pois havia dificuldade em gerenciar quais eram as tarefas atribuídas aos bolsistas de IC vinculados. E, uma vez atribuídas, o acompanhamento do seu desenvolvimento não era concentrado em um único local, dificultando a geração de relatórios e o entendimento do *status* atual.

Outra dificuldade identificada era a comunicação com os bolsistas. Por vezes, esse contato era difícil e sem respostas, uma vez que o *e-mail* do bolsista disponível ao orientador era o institucional, porém, este não era o seu endereço eletrônico principal, gerando assim falhas de comunicação. O contato por telefone, também era difícil, já que os telefones cadastrados, quando existiam, estavam desatualizados.

2.2 Ferramentas Disponíveis no Mercado

Após o primeiro contato com a descrição das funcionalidades exigidas, foram analisadas as possibilidades de *softwares* livres disponíveis. Baseado em uma prévia avaliação do gerente de projetos, os principais aplicativos que apresentavam as características próximas às necessárias, ainda que de paradigmas diferentes, eram: "dotProject", "Mantis", "Redmine" e "Trac".

Apesar de nenhuma ferramenta possuir cadastro de horários, todas disponibilizam cadastro de tarefas, de usuários e seus respectivos contatos. Entretanto, as duas primeiras foram consideradas mais rígidas, possuindo vários campos que não seriam utilizados no formulário de tarefas. Ainda, havia campos obrigatórios, determinados pelo sistema, que não seriam de uso do projeto.

As duas últimas ferramentas - "Redmine" e "Trac" - foram consideradas mais simples, em que havia os cadastros necessários, porém, de forma restrita. O cadastro de contatos era limitado e não apresentava diversificação de tipos de contatos a serem cadastrados.

Além disso, havia uma limitação que deveria ser considerada para implantação e utilização da ferramenta. Foi imposto pelo departamento de administração de rede do Instituto de Informática da Universidade, que o *software* a ser instalado deveria ser desenvolvido na linguagem *PHP* - sem uso de *frameworks*[1] - e que utilizasse o banco de dados *MySQL*.

Considerando essa limitação, as ferramentas "Redmine" e "Trac", embora as mais adequadas para uso, seriam descartadas uma vez que, apesar de utilizar o banco de dados *MySQL*, foram desenvolvidos nas linguagens *Ruby on Rails* e *Python*, respectivamente.

2.3 Identificação dos Envolvidos no Projeto e da Real Demanda

Uma vez identificadas as necessidades do sistema e analisados os aplicativos existentes com as suas limitações, chegou-se à conclusão de que seria interessante desenvolver um aplicativo simples e robusto, que atendesse as necessidades do projeto, assim como as restrições do ambiente de instalação.

[1] No desenvolvimento de aplicativos orientado a objetos, *framework* ou *framework* de aplicação é "um projeto de subsistema composto por um conjunto de classes abstratas e concretas e as interfaces entre elas". (WIRFS-BROCK; JOHNSON, 1990 *apud* SOMMERVILLE, 2007, p. 282)

Como primeiro passo desse processo, foram definidos quem seriam os *stakeholders* e, a partir deles, seria mapeada a real demanda de desenvolvimento.

2.3.1 *Stakeholders* ou Grupo de Interesse: identificação e suas responsabilidades

Em um projeto customizado, complexo e de longo período, como a construção de um sistema de gerenciamento, geralmente surge a necessidade de decisões por parte do seu desenvolvedor, que embora discricionárias, levam em consideração o interesse dos grupos, destacando o alto grau de envolvimento do "cliente" e o seu poder de influência dentro desse tipo de projeto.

> As partes interessadas são pessoas e organizações, tais como clientes, patrocinadores, a organização executora e o público, que estão ativamente envolvidos no projeto ou cujos interesses podem ser positiva ou negativamente afetados pela execução ou pelo término do projeto. Também podem exercer influência sobre o projeto e suas entregas (PROJECT MANAGEMENT INSTITUTE, 2008, p.206).

Sendo assim, *stakeholders* ou Grupo de Interesse são pessoas ou grupos que possuem algum tipo de interesse no projeto e que, dessa forma, devem ser tratados e comunicados de acordo com sua visão e importância dentro do processo de construção.

O relacionamento e a comunicação com as partes devem estar voltadas a viabilizar a aceitação do projeto, minimizar preocupações relacionadas aos riscos, esclarecer questões e, ainda, determinar mudanças de planos e estratégias (PMI, 2008).

A consulta aos *stakeholders* deve ser agressiva, coletando informações completas e consistentes que ajudem no desenvolvimento do projeto. Já o processo de comunicação deve ser adequado para cada nível de interesse, influência e envolvimento, evitando informações desnecessárias que possam trazer impactos negativos (SLACK, CHAMBERS, JOHNSTON, 2009).

Durante a construção do "Smart Task Manager" foram identificados diferentes tipos de *stakeholders*. O tipo Idealizador, grupo que vislumbrou os primeiros passos e definiu as necessidades iniciais para a construção do *software*. O tipo Incentivador, que deu suporte evolutivo e crítico, auxiliando em decisões relevantes. O tipo Usuário, a quem foi preciso considerar as características para que todas as informações cadastrais de horários, contato e níveis de acesso fossem adicionadas. E, por fim, o tipo Administrador, que definiu funcionalidades específicas do projeto, a quem foi necessário considerar suas opiniões sobre itens indispensáveis para melhor gestão e o seu nível de capacitação para manipulação do aplicativo.

Considerando as opiniões e o grau de participação dos envolvidos no processo, é possível indicar como cada um destes tipos de *stakeholders* pode contribuir nas definições de desenvolvimento do ”STM”.

No grupo dos s*takeholders* do tipo Idealizador está o professor coordenador do projeto. Ele definiu a problemática básica do sistema, indicou quais seriam os primeiros módulos a serem construídos e interagiu na criação dos primeiros protótipos de telas do sistema.

Já o tipo Incentivador é representado pelo professor orientador desse trabalho, que propôs e opinou nas diferentes formas de apresentar os dados do *software* aos usuários e sugeriu informações que poderiam ser agregadas ao formulário de tarefa – tal como o campo de comentários nas tarefas - apresentando a sua relevância nesse contexto.

O *stakeholder* do tipo Administrador pode ser representado pelo setor de Administração de Rede do Instituto de Informática da UFRGS e pelo gerente de projetos. Ambos foram responsáveis pela definição da estrutura do “STM”. A partir das informações e limitações que eram fornecidas pelo setor de Administração de Rede foi possível definir qual seria a linguagem de programação a ser utilizada e quais recursos seriam necessários para o desenvolvimento e execução do aplicativo nos servidores da Universidade. Já o gerente de projetos participou nas decisões mais críticas de todo o processo, indicando quais seriam as funcionalidades e informações relevantes que deveriam ter alta prioridade de desenvolvimento.

Finalmente, o *stakeholder* do tipo Usuário é representado por alunos bolsistas e também pelo gerente de projetos. Ambos contribuíram ao definir algumas funcionalidades que iriam ajudar no controle e no gerenciamento dos usuários e das tarefas no sistema.

2.3.2 Demanda de desenvolvimento

Uma vez identificados todos os *stakeholders*, iniciou-se a discussão e a coleta dos detalhes das funcionalidades e comportamento almejados para o gerenciador de tarefas de bolsistas.

No processo de análise e elucidação, foi possível identificar outras necessidades e desejos dos *stakeholders* a fim de facilitar o seu dia-a-dia e garantir melhor desempenho do grupo. Os dispositivos que diferenciavam das ferramentas já existentes são:

- Apresentação das tarefas utilizando *Kanban* (Seção 2.3.2.1);
- Criação de tarefas em lote para contextos predefinidos;

- Possibilidade de salvar um texto padrão no formulário de criação de tarefas, *template*;
- Possibilidade de adicionar múltiplos contatos;
- Capacidade de adicionar novos tipos de contatos;
- Obrigatoriedade de ao menos um *e-mail* no cadastro de um usuário;
- Envio de *e-mail* pela ferramenta ao manipular uma tarefa.

2.3.2.1 Kanban (processo puxado)

O *Kanban* surgiu em 1953, criado por Taiichi Ohno, devido à necessidade de um sistema que "puxasse" a produção através de estímulos visuais simples (AGUIAR, PEINADO, 2007).

Nesse sistema visual, criado para controle da produção, foram utilizadas cores para marcar os níveis de autoridade para as necessidades. A sinalização era feita geralmente pelas cores:

- Vermelho: representa urgência ou início imediato da atividade;
- Amarelo: indica média urgência ou atenção às necessidades; e
- Verde, processo normal.

Assim como no Sistema de Controle Puxado, o aplicativo para gerenciamento de tarefas possui regras de priorização para as atividades que seguem os princípios do *Kanban*, em que o recebimento de um aviso visual dispara o movimento, a produção ou o fornecimento de uma nova unidade. Dessa forma, a paleta de cores funciona como um sinal para que o usuário possa administrar as atividades destinadas a ele, e seguir a melhor decisão para produção de seu trabalho. Assim como um bolsista tomará suas decisões baseadas no que é prioridade ou não, o orientador também utilizará da ferramenta para indicar e direcionar os esforços para as atividades de maior urgência.

3 FERRAMENTAS RELACIONADAS

Considerando as necessidades dos *stakeholders* para uma ferramenta de gerenciamento de tarefas, foram identificadas quatro possíveis aplicações *web* que continham, em partes, características e campos necessários a serem adicionados no *software* desenvolvido. Porém, para a utilização de qualquer uma delas, seriam necessárias várias alterações e customizações.

As ferramentas analisadas foram: “dotProject”, “Mantis”, “Redmine” e “Trac”.

3.1 Ferramentas Utilizadas como Base de Desenvolvimento

O “STM” foi desenvolvido baseado nas ferramentas apresentadas a seguir.

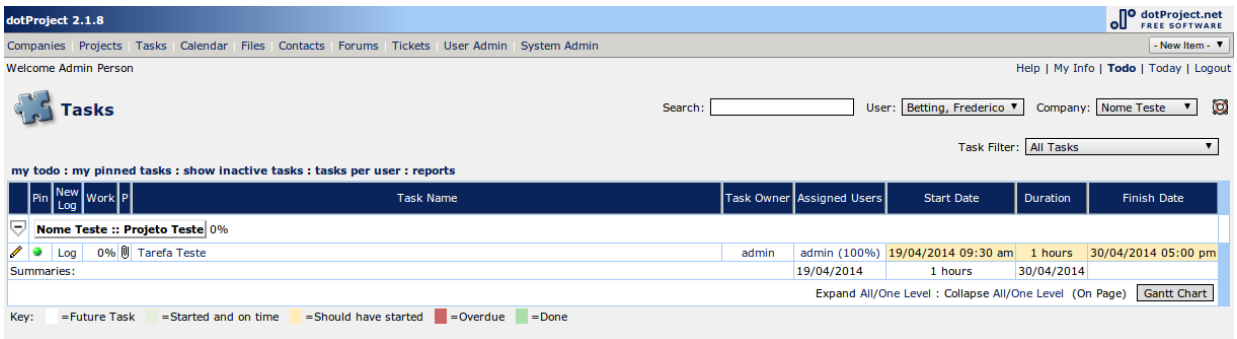
Foram utilizados como base de desenvolvimento do aplicativo parâmetros como *layout* e a forma de apresentação de dados. Além disso, alguns dos campos relacionados a tarefas - disponíveis nos *softwares* de referência - também foram considerados.

A seguir, uma breve descrição de cada ferramenta analisada, apresentando suas referências e principais características.

3.1.1 dotProject

A aplicação “dotProject”, exemplificado através da tela principal de tarefas (Figura 3.1), é uma ferramenta livre e *open source*, possui interface *web* e tem como foco principal o gerenciamento de projetos.

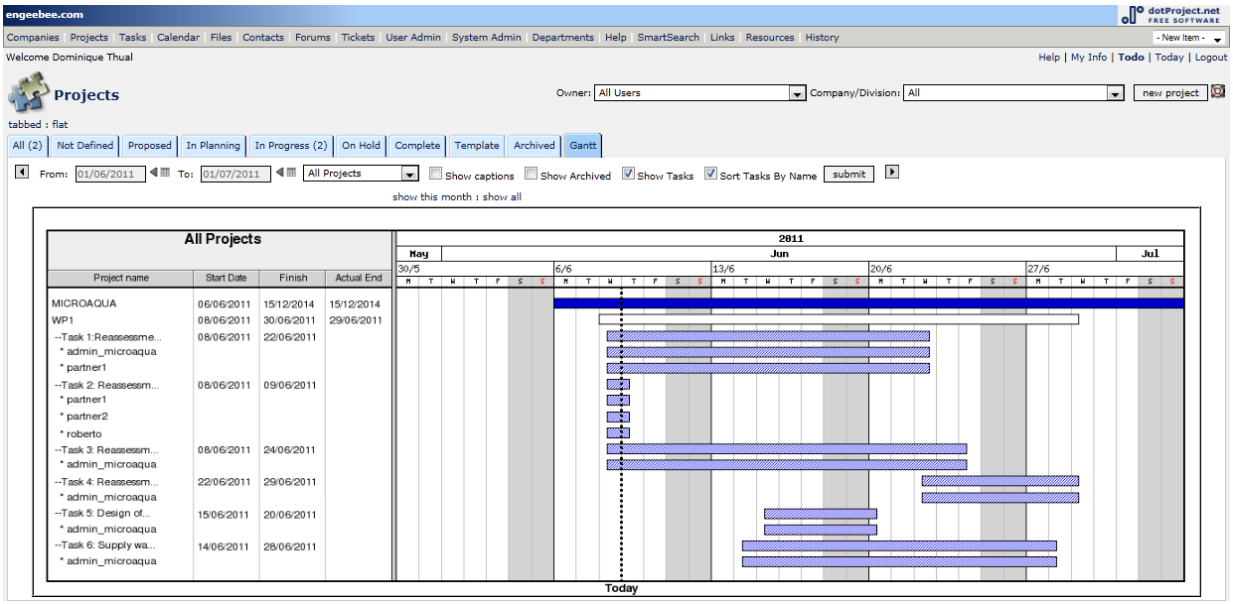
Figura 3.1 – Tela principal de tarefas do “dotProject”



Fonte: DOTPROJECT, 2014.

Dentre as várias funcionalidades disponíveis, desde as mais básicas como cadastro de usuários, de projetos e de tarefas, também estão presentes bons mecanismos para visualização geral do planejamento de um projeto na forma de listagem de tarefas (Figura 3.1) e Diagrama de *Gantt* (Figura 3.2).

Figura 3.2 – Diagrama de *Gantt* do "dotProject"



Fonte: DOTPROJECT, 2014.

Algumas das suas principais características do "dotProject" são:

- Processo de instalação simplificado. Uma vez que o servidor *web* e o banco de dados estão instalados, é necessário apenas executar um *wizard* para completar o processo de instalação.
- Permite gerenciar múltiplos projetos com uma única instalação. É possível determinar quais projetos estarão disponíveis para um usuário, informando o seu nível de acesso.
- Possui espaço para anexar arquivos. Porém, o seu uso não é simplificado, sendo o usuário o responsável por referenciar os arquivos nas tarefas, indicando diretório a ser salvo, categoria de arquivo, nome do projeto e tarefa.
- Há espaço para comentários – *logs* - em cada tarefa existente. Estão disponíveis vários campos de entrada de dados que têm como principal objetivo informar, em porcentagem, o *status* atual da produção. A

comunicação geral e o histórico de alterações não são o foco, ficando em segundo plano.

- Possui fórum de discussões *on-line*. Todas as discussões criadas podem ser vinculadas a projetos e tarefas existentes.
- Permite o envio de tarefas por *e-mail* ao criá-las ou alterar o seu *status*.
- As mensagens de erro não são amigáveis, sendo apresentadas sem tratamento para o usuário.

Apesar de parte dos campos exigidos pelos *stakeholders* estar presente no sistema, ainda existem outros, considerados obrigatórios pelo sistema, que não seriam utilizados.

Além disso, a alteração de uma tarefa não é intuitiva, com um fluxo confuso. Há também outros setores da ferramenta cujo fluxo não é linear, exigindo assim prévio treinamento para sua utilização.

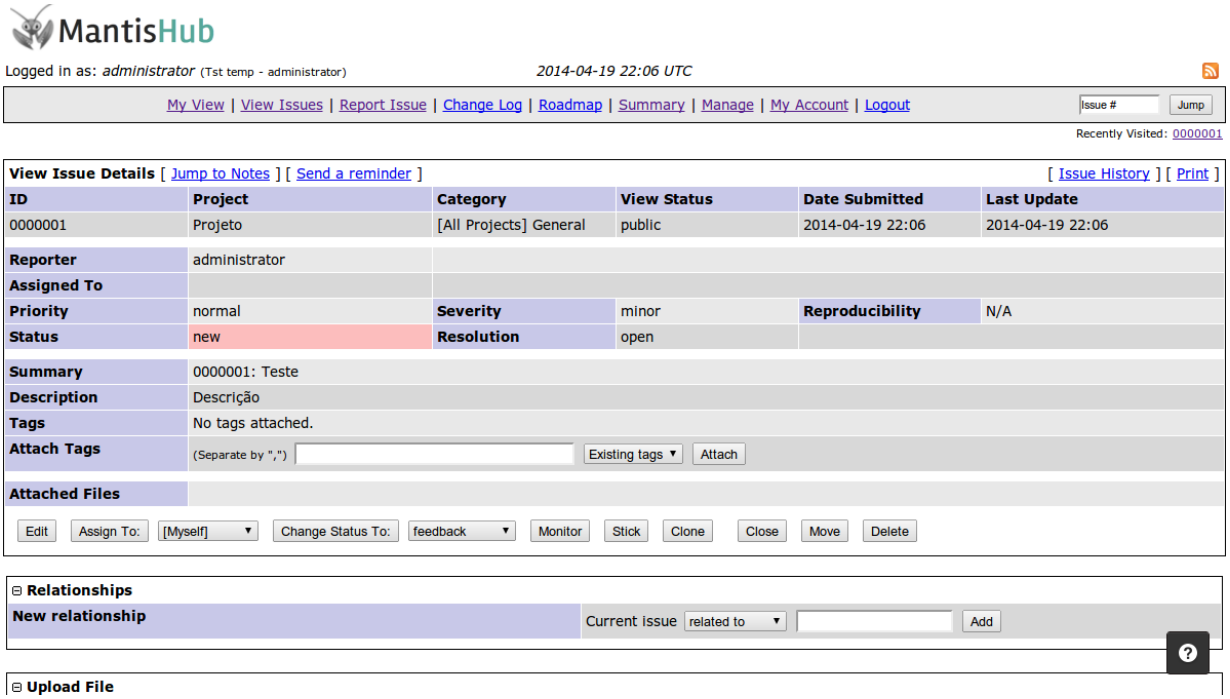
Considerando as necessidades dos *stakeholders*, é possível identificar a ausência de algumas informações:

- A lista de contatos disponível é restrita, não podendo ser adicionados novos tipos ou mesmo alterar os já existentes de forma dinâmica. E, nenhum tipo de contato é obrigatório.
- Não existe local para entrada de grade de horários dos usuários.
- A lista de tarefas é simplificada e não é possível aplicar filtros mais complexos ou salvar filtros favoritos.

3.1.2 Mantis

O *software* “Mantis” é uma ferramenta *web*, livre e *open source* com o foco principal no gerenciamento de defeitos (Figura 3.3). Para fins gerais, é possível utilizá-la como um gerenciador de tarefas.

Figura 3.3 – Janela com descrição de um defeito ou tarefa do “Mantis”



Fonte: MANTIS, 2014.

Apesar de ser uma aplicação voltada para o acompanhamento de defeitos de *software*, “Mantis” é totalmente adaptável para o gerenciamento de tarefas, já que boa parte das informações necessárias para a manipulação de ambos é a mesma.

Porém, por ser uma adaptação de uso, pode tornar a sua utilização onerosa em alguns casos. Isso se deve ao fato da necessidade de adicionar valores a campos específicos relacionados a defeitos, tais como “Reprodutibilidade”, “Severidade” ou “Passos para Reproduzir”, que são dispensáveis quando se fala em gerenciamento de tarefas em uma forma genérica. Entretanto, com pequenos ajustes, é possível ignorar esses campos adicionando valores como “N/A” - Não Aplicável - ou, em outros casos, fazer pequenas alterações no banco de dados e código fonte para mostrar valores padrão de “não aplicável”.

Algumas das principais características da ferramenta “Mantis” são:

- Cada instalação permite gerenciar múltiplos projetos. O acesso a cada projeto por usuário pode ser configurado através de níveis de acesso.
- Interface é simplificada. O *layout* da ferramenta é simples, com itens de menus intuitivos.
- Campo de comentário voltado para comunicação. Está disponível um único campo, de texto livre, para adição de comentários e/ou questionamentos da tarefa em questão.

- Adição de filtros complexos. O *software* disponibiliza interface para criação ampla de filtros.
- Visualização de tarefas a partir de uma lista. Na tela principal, todas as tarefas submetidas ao projeto selecionado são mostradas. Esse é o único mecanismo de visualização geral de tarefas do sistema.
- Permite anexar arquivos a tarefas. A opção está disponível ao manipular um defeito/tarefa. Sua utilização é simples e intuitiva.
- Permite vínculo entre tarefas. O usuário pode referenciar tarefas naquela que está sendo criada através do "número de tarefa".

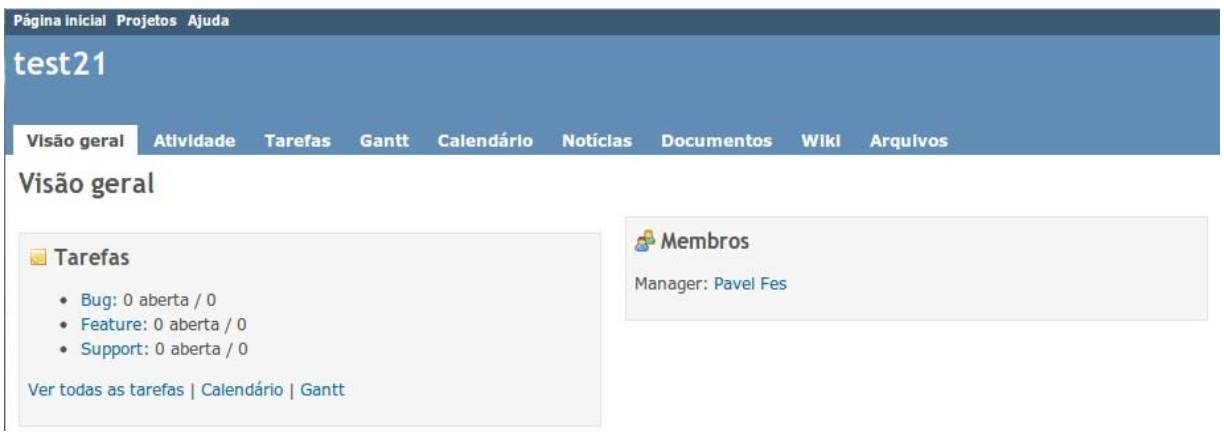
Esta é uma ferramenta simples de se utilizar, entretanto, nem todas as características exigidas pelos *stakeholders* estão disponíveis. Como exemplo, o cadastro de contatos é muito simplificado, estando disponível apenas um *e-mail* por usuário. Não é possível cadastrar a grade de horários, como também não há um campo para indicar um prazo para término de tarefa.

3.1.3 Redmine

A ferramenta *web* "Redmine" é livre e *open source*. É direcionada ao gerenciamento de projetos e de defeitos de *software*. Disponibiliza várias seções na sua interface inicial (Figura 3.4), visando à comunicação entre os seus usuários.

Dentre as funcionalidades presentes no *software* estão uma página *wiki*, onde é possível adicionar textos gerais sobre o projeto, e um repositório de arquivos, local para disponibilização de documentos, ferramentas e/ou outros arquivos relacionados ao projeto.

Figura 3.4 – Tela inicial do "Redmine"

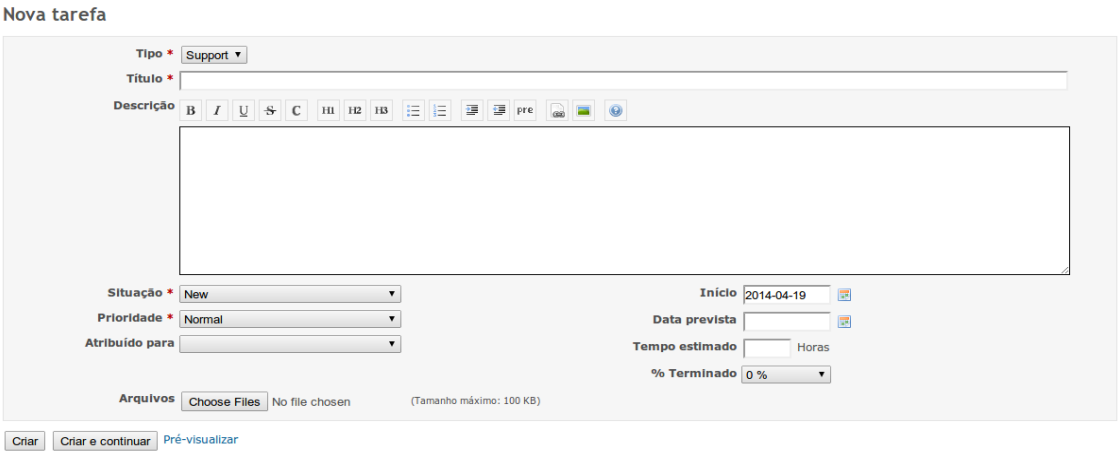


Fonte: REDMINE, 2014.

Algumas das principais características do “Redmine” são:

- Possui um fórum *on-line*. Permite discussão sobre funcionalidades ou problemas encontrados no projeto.
- Filtros complexos. Possui interface para criação de filtros avançados.
- Diagrama de *Gantt*. Contém diagrama gráfico para melhor visualização do projeto.
- Integração com versionadores de código. Possibilita integração com *softwares* de versionamento tais como *SVN*, *Git* e outros.
- Formulário de tarefas simples (Figura 3.5). Contém todas as informações básicas para criação de uma tarefa.
- Cada instalação faz referência a um único projeto. São necessárias múltiplas instalações para gerenciar múltiplos projetos.

Figura 3.5 – Formulário para criação de uma nova tarefa no “Redmine”



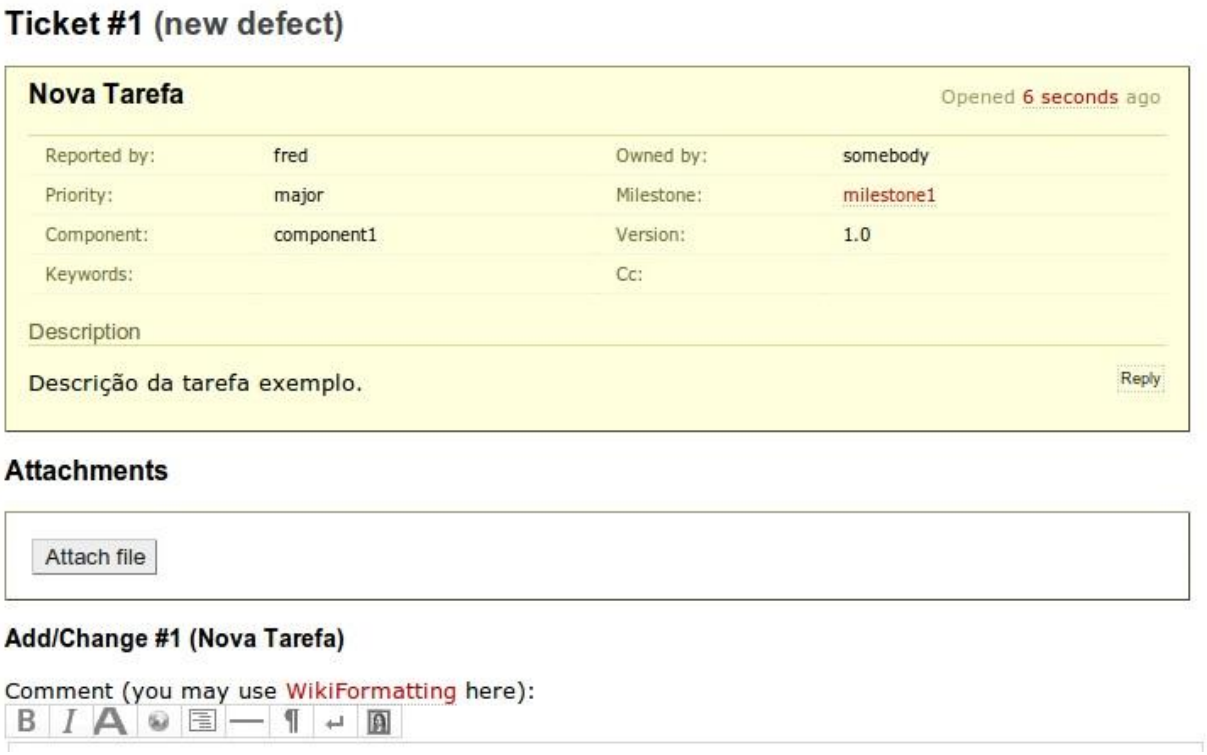
Fonte: REDMINE, 2014.

“Redmine”, assim como as ferramentas já citadas, não possui campos e funcionalidades exigidas pelos *stakeholders* como, por exemplo, grade de horários, múltiplos contatos e criação de tarefas baseados em *templates*.

3.1.4 Trac

“Trac” é uma ferramenta *web* simples e *open source*. Cada instalação permite gerenciar um único projeto, em que disponibiliza interface para cadastro de tarefas (Figura 3.6) e controle de mudanças do processo de desenvolvimento, uma vez que possibilita a integração com servidores de versionamento.

Figura 3.6 – Tela de descrição de uma tarefa/defeito do “Trac”



Fonte: TRAC, 2014.

As principais características do “Trac” são:

- Inexistência de cadastro formal de usuários. Não é possível cadastrar dados pessoais e contatos de um usuário na ferramenta.
- Possibilita criação de filtros favoritos. Através de interface específica, é possível gerar um filtro complexo e salvá-lo como favorito.
- Interface simples e intuitiva. Todas as opções essenciais estão dispostas na tela principal.
- Disponibiliza página *wiki* para os usuários. Possui módulo *wiki* para adição de documentação e páginas de ajuda ao usuário.
- Formulário de tarefa simples. Contém os campos essenciais para criação de uma tarefa.

Apesar de ser uma ferramenta completa quanto ao cadastro de tarefas, algumas das informações básicas não estão disponíveis. Cadastro de usuários, notificações por *e-mail* e mecanismos gráficos para visualização do projeto são exigências dos *stakeholders*.

3.2 Comparando o “Smart Task Manager” com as Aplicações de Referências

Considerando as funcionalidades disponíveis nas ferramentas de referência e as do aplicativo desenvolvido (Capítulo 4), é possível sumarizar e comparar algumas das principais características de cada ferramenta.

Nesse contexto, pode-se dividir a análise em duas partes: características gerais das ferramentas e características específicas de tarefa.

3.2.1 Características Gerais das Ferramentas

A seguir, serão citadas algumas das principais características gerais identificadas nas ferramentas. Tais características estão, também, resumidas em tabela a seguir (Tabela 3.1).

3.2.1.1 Mecanismos de visualização de tarefas

Das ferramentas analisadas, todas apresentam uma interface com tabela para exibir as tarefas existentes no projeto. Porém, apenas duas, “Mantis” e “Trac”, não possuem um mecanismo gráfico para apresentar as tarefas cadastradas de um projeto.

“Redmine” e “dotProject”, além de utilizarem uma tabela para apresentar as tarefas cadastradas, também as expõem através do Diagrama de *Gantt*, um diagrama que utiliza barras sobre um eixo horizontal para representar intervalos de tempo. Cada barra representa uma tarefa de um usuário e cujo seu comprimento indica o tempo dispendido para sua realização.

Já o “STM” expõe as tarefas associadas ao usuário utilizando *Kanban*. Através dessa interface é possível identificar a demanda de trabalho baseado na prioridade e no *status* das tarefas apresentadas.

3.2.1.2 Filtros de tarefas

Quanto a filtros de tarefas, é possível comparar a disponibilidade de filtros avançados e de lista de filtros favoritos.

Com exceção do “dotProject”, todas as outras ferramentas possuem interface para criação de filtros avançados, permitindo melhor precisão nos resultados de uma consulta.

Além da criação de filtros, o “Smart Task Manager” e o “Trac” também possibilitam salvar filtros favoritos. Embora no “Trac” seja possível salvar apenas um filtro como favorito, no “STM” há disponibilidade de salvar uma lista deles.

3.2.1.3 Criação de tarefas por templates

A disponibilidade de criar em lote uma ou um conjunto de tarefas previamente definidas está disponível apenas no “STM”, sendo um grande diferencial em relação às outras ferramentas quando há necessidade de se criar tarefas com as mesmas características repetidas vezes.

3.2.1.4 Mecanismos de comunicação

São avaliados alguns mecanismos de comunicação disponíveis nas ferramentas. Um deles é a existência de página *wiki*. Presente no “dotProject”, “Redmine” e “Trac”, é possível adicionar informações gerais e documentos do projeto para acesso de todos os usuários.

A lista de contatos está presente no “STM”, “dotProject” e “Mantis. Entretanto, no “Mantis” e no “dotProject” a lista é muito restrita, permitindo adicionar apenas uma quantidade limitada e pré-definida de contatos. Ainda na última ferramenta, nenhum contato é obrigatório.

3.2.1.5 Grade de horários

Apenas no “Smart Task Manager” está disponível uma tabela de horários, em que o usuário poderá informar qual é o período que ele está disponível para o projeto.

3.2.1.6 Múltiplos projetos

A possibilidade de gerenciamento de projetos em uma única instalação de *software* está disponível apenas para os aplicativos “STM”, “dotProject” e “Mantis”. Essa funcionalidade contribui para melhor gerenciamento dos projetos vinculados, uma vez que todas as informações estão concentradas em uma única ferramenta.

Tabela 3.1 – Características gerais das ferramentas

	Views de Tarefas	*Filtros avançados / Favoritos*	*Template*	*Wiki*	*Contatos*	*Multi Projetos*	*Grade de horários*
STM	Kanban, Listagem	Sim / Sim	Sim	Não	Sim	Sim	Sim
dotProject	Gantt, Listagem	Não / Não	Não	Sim	Sim	Sim	Não
Mantis	Listagem	Sim / Não	Não	Não	Sim	Sim	Não
Redmine	Gantt, Listagem	Sim / Não	Não	Sim	Não	Não	Não
Trac	Listagem	Sim / Sim	Não	Sim	Não	Não	Não

Fonte: Autoria própria, 2014.

3.2.2 Características Específicas de Tarefas

Abaixo, serão discutidas algumas das principais características específicas de tarefas, resumidamente apresentadas na Tabela 3.2.

3.2.2.1 Facilidade de uso

Considerando o fluxo de criação e edição de uma tarefa nas ferramentas analisadas, foi possível identificar dificuldades para manipular tarefas no “dotProject”. Nele, é necessário sempre selecionar um projeto para então manipular cada tarefa desejada.

Já a ferramenta “Mantis” possui grande quantidade de informações para preenchimento de uma tarefa, o que torna seu uso custoso.

As demais ferramentas apresentam um formulário simples e conciso, sendo consideradas de fácil uso e preenchimento, ou seja, com poucas etapas para efetivar o cadastro e a manipulação de uma tarefa.

No “STM”, o usuário encontra todas as funcionalidades explícitas na tela principal, sendo necessário apenas um passo para a criação de uma nova tarefa.

As demais ferramentas apresentam um formulário simples e conciso, sendo consideradas de fácil uso e preenchimento.

3.2.2.2 Informações disponíveis no cadastro de tarefas

Em geral, analisando os formulários de tarefas disponíveis nas ferramentas, todos apresentam os campos mínimos necessários para gerenciar um projeto de acordo com as necessidades dos *stakeholders*. Porém, os formulários do "dotProject" e "Mantis" disponibilizam diversos campos não úteis para seu contexto. Sendo que em alguns casos, campos desnecessários são de preenchimento obrigatório.

3.2.2.3 Notificações por e-mail

Somente nos aplicativos "STM" e "dotProject" está disponível o envio de notificações por *e-mail* ao manipular tarefas.

3.2.2.4 Espaço para comentários

Todas as ferramentas analisadas disponibilizam espaço para comunicação geral através de comentários. Entretanto, na ferramenta "dotProject", o principal foco deste espaço é para disponibilização do *status* atual do desenvolvimento da tarefa, através de campos específicos. Para as outras ferramentas, é disponibilizado apenas um campo de texto livre em que se pode adicionar qualquer conteúdo ou questionamento aos usuários.

3.2.2.5 Arquivos anexos

Todas as ferramentas avaliadas possuem, com graus diferentes de dificuldade de utilização, um mecanismo que possibilita anexar arquivos a tarefas.

Tabela 3.2 – Características de uma tarefa

	Facilidade de Uso	*Informações básicas (tarefas)*	*Notificações por e-mail*	*Espaço para comentários*	*Arquivos anexos*
STM	Alta	Sim	Sim	Sim	Sim
dotProject	Baixa	Não	Sim	Sim	Sim
Mantis	Média	Não	Não	Sim	Sim
Redmine	Alta	Sim	Não	Sim	Sim
Trac	Alta	Sim	Não	Sim	Sim

Fonte: Autoria própria, 2014.

4 GERENCIADOR DE TAREFAS “SMART TASK MANAGER”

Neste capítulo será apresentada a ferramenta de gerenciamento de tarefas “STM”, mostrando sua arquitetura, funcionalidades e alguns aspectos do processo de desenvolvimento.

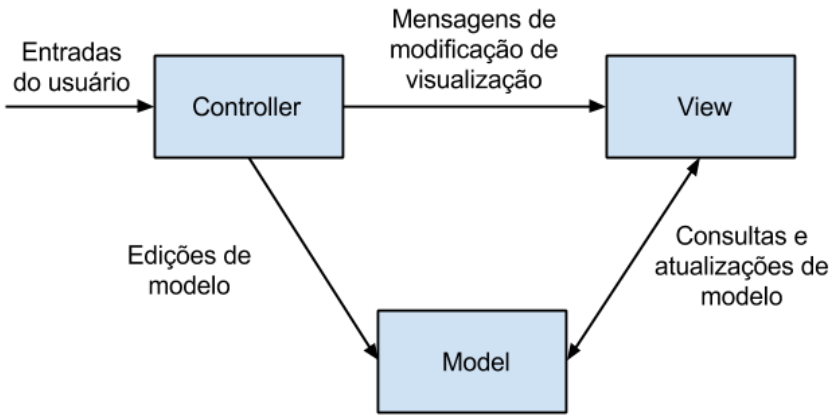
4.1 Arquitetura da Ferramenta

O aplicativo “Smart Task Manager” foi criado conforme o padrão de arquitetura em três camadas *MVC*, ou *Model-View-Controller*. O principal objetivo desse padrão é separar a lógica de negócio da interface com o usuário. Essa abordagem contribui para o processo de desenvolvimento e manutenção do aplicativo já que cada camada pode ser manipulada de forma independente.

A sua estrutura básica consiste em criar três componentes distintos, muito utilizados em desenvolvimento de aplicações *web* complexas. São eles: o *model*, que representa a lógica de negócio e gerencia o comportamento da aplicação de domínio; a camada *view*, responsável por gerenciar as informações apresentadas aos usuários do sistema; e, por fim, a camada *controller*, que manipula as informações entradas pelo usuário e comunica as outras camadas (MICROSOFT DEVELOPER NETWORK, 2014).

A Figura 4.1 ilustra as três camadas da arquitetura e como elas se relacionam. É importante destacar a dependência existente entre a *view* e o *model* do *controller*, que é utilizado como uma interface de comunicação entre os componentes.

Figura 4.1 – Modelo MVC de interação com o usuário



Fonte: Adaptado de SOMMERVILLE (2007, p. 246).

4.2 Características da Ferramenta

O aplicativo desenvolvido "STM" é *open source*, possui uma interface *web* e seu principal objetivo é o gerenciamento de projetos e tarefas.

A partir de uma instalação, é possível criar múltiplos projetos e gerenciar todas as tarefas vinculadas a cada projeto em ambientes separadas. Ainda estão presentes:

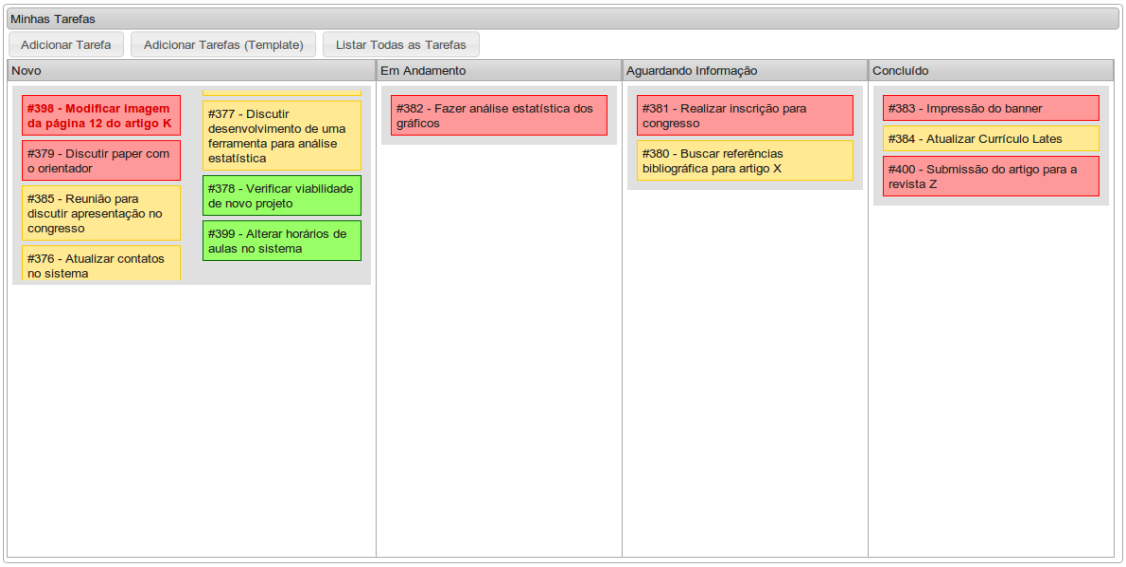
- Cadastro de horário de usuários;
- Criação de filtros avançados;
- Armazenamento de diferentes filtros em lista de favoritos;
- *Upload* de arquivos anexos a tarefas;
- Adição de múltiplos contatos do usuário;
- Envio de *e-mails* pelo aplicativo após manipulação de tarefa.

A seguir, serão apresentados detalhes de duas das principais características desenvolvidas.

4.2.1 Interface *Kanban*

No "STM", a interface *Kanban* é a tela principal do aplicativo, após o usuário se autenticar no sistema (Figura 4.2). Essa interface - cuja apresentação é feita por caixas coloridas dispostas em colunas representando *status* - proporciona ao usuário melhor visualização das tarefas e suas prioridades dentro do projeto, uma vez que é possível identificar, através das cores, quais são as tarefas de maior prioridade vinculadas a ele.

Figura 4.2 – Interface *Kanban* do "STM"



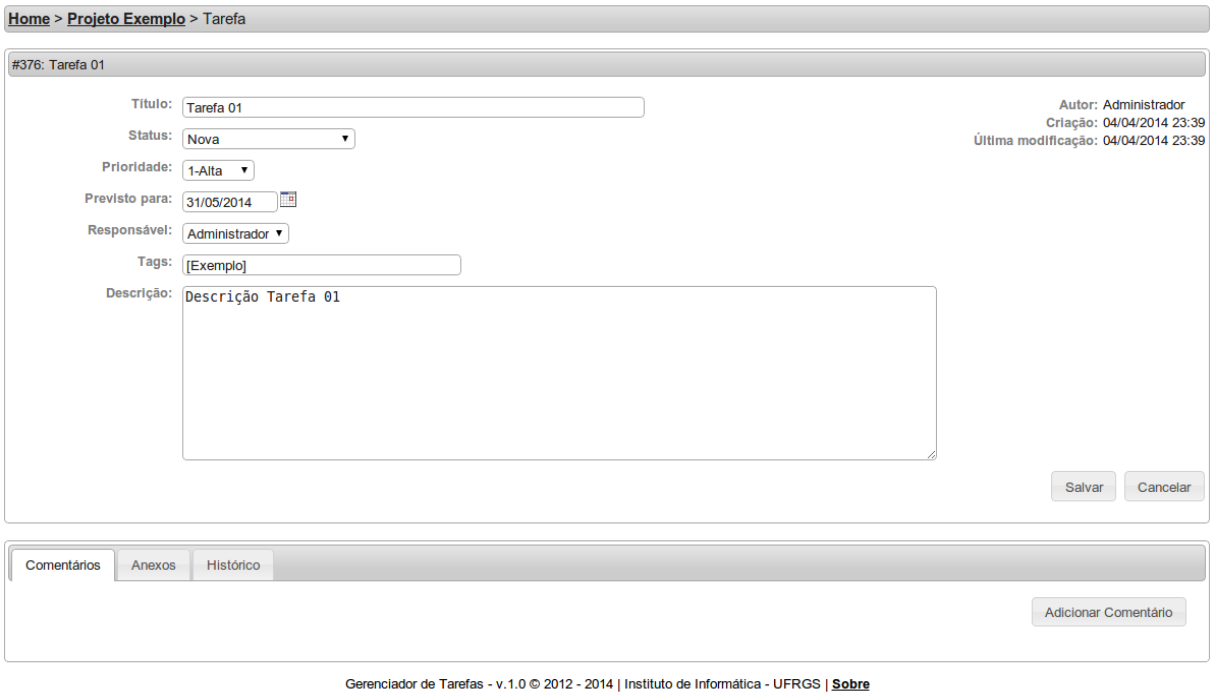
Fonte: Autoria própria, 2014.

Dessa forma, se houver tarefas cuja representação seja uma caixa na cor vermelha, indica que o usuário possui tarefas de alta prioridade a serem resolvidas. Àquelas tarefas representadas pela cor amarela, indica ser de média prioridade. E, por fim, todas as tarefas que forem representadas pela cor verde são de baixa prioridade.

Além das cores das caixas, ainda é possível diferenciar as cores das letras. Normalmente, após uma tarefa ser criada, o seu título será mostrado com letras na cor preta. Porém, uma vez que a data limite de entrega tenha sido atingida e/ou ultrapassada, a tarefa será colocada no topo da lista de tarefas com a mesma classificação de prioridade e as letras do título serão alteradas para a cor vermelha, chamando a atenção do usuário.

A adição de uma tarefa é realizada através de um formulário específico (Figura 4.3), ao selecionar o botão “Adicionar Tarefa” na tela principal ou na tela de lista geral de tarefas do projeto. Ao especificar o *status* inicial e a prioridade da nova tarefa no processo de criação, automaticamente, ela será adicionada com a cor e na coluna da tabela *Kanban* correspondentes aos dados inseridos pelo usuário.

Figura 4.3 – Formulário de adição de tarefas do “STM”



Fonte: Autoria própria, 2014.

Uma vez que o usuário identifique a tarefa a ser manipulada, é possível alterar o seu *status* apenas arrastando a caixa da tarefa entre as colunas disponíveis (“Novo”, “Em Andamento”, “Aguardando Informações” e “Concluído”) e assim que a caixa é liberada na

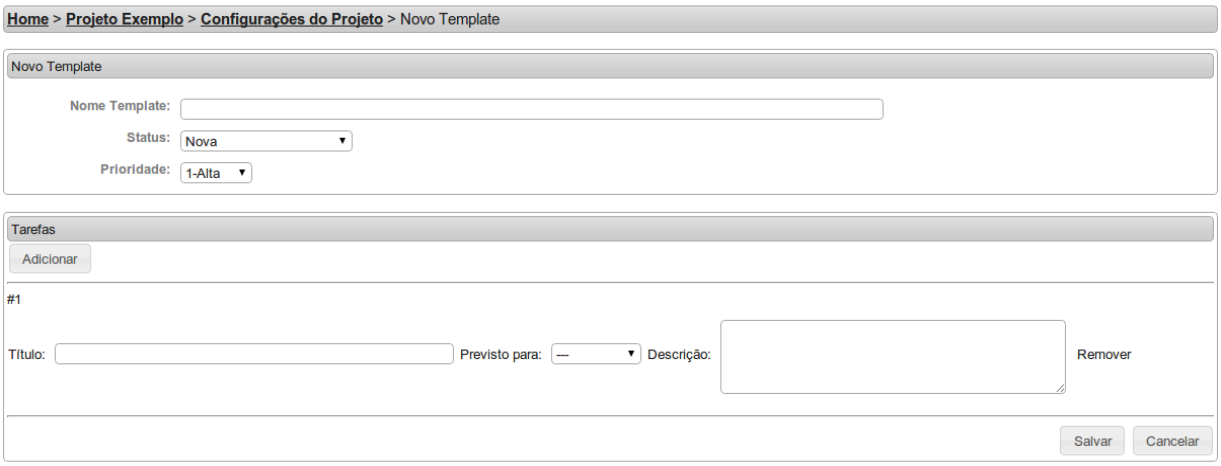
coluna desejada, o seu *status* será atualizado. Se o usuário desejar, ainda é possível ler e alterar a descrição e os detalhes de cada tarefa clicando no título da tarefa.

4.2.2 Criação de tarefas baseado em *templates*

Também está disponível no "STM" a possibilidade que criar um lote de tarefas previamente preenchidas. Essa funcionalidade pode ser encontrada na opção "Adicionar Tarefa (Template)" na página inicial ou na lista geral de tarefas do projeto.

Para utilizá-la, é necessário que um usuário habilitado do sistema cadastre um *template* de tarefa nas "Configurações do Projeto". Na interface de criação (Figura 4.4), poderá ser definido o conjunto de tarefas a ser criado ao utilizar o *template*.

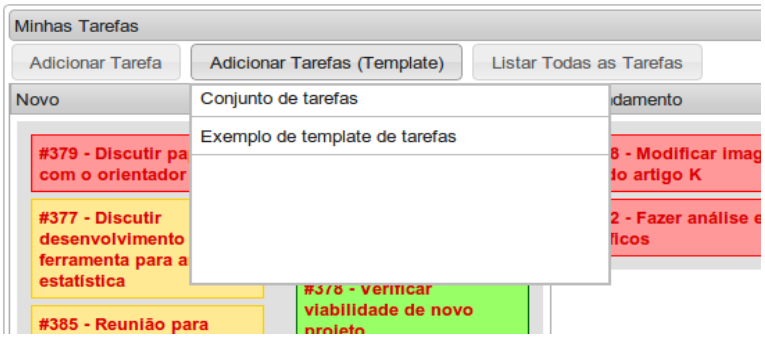
Figura 4.4 – Interface de criação de *templates* do "STM"



Fonte: Autoria própria, 2014.

Uma vez que o *template* esteja criado, ele estará disponível na lista de *templates* do projeto (Figura 4.5). Para utilizá-lo, basta o usuário selecionar um *template* disponível na lista.

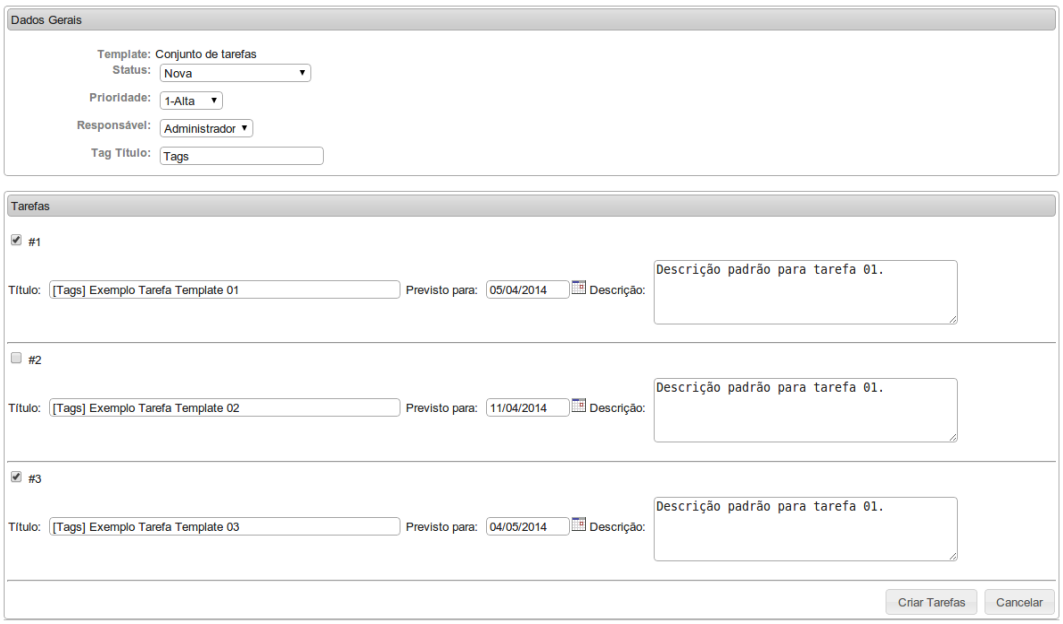
Figura 4.5 – Lista de *templates* disponíveis para o projeto



Fonte: Autoria própria, 2014.

Após a seleção, será mostrado ao usuário o conjunto de tarefas e as características previamente definidas daquele *template* (Figura 4.6). Se o usuário desejar, ele poderá alterar o conteúdo pré-definido de qualquer tarefa disponível para um texto mais específico, já que todos os campos do formulário são editáveis. Porém, todas as modificações realizadas serão armazenadas apenas nas tarefas criadas, não alterando o conteúdo do *template* original, que é utilizado apenas como valores de referência. Também será possível adicionar uma *tag* no início do título de todas as tarefas a serem criadas, adicionando-a no campo "Tag Título".

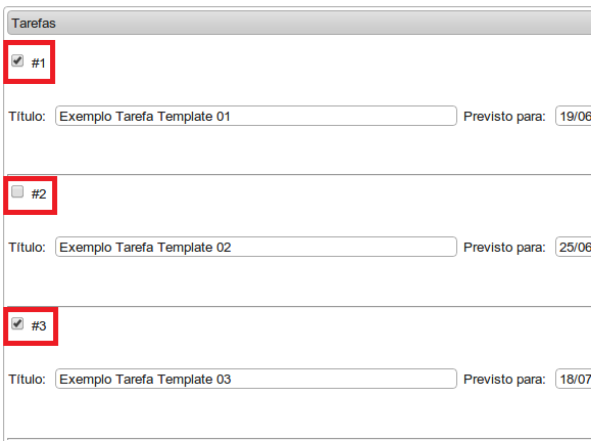
Figura 4.6 – Interface de criação de tarefas baseado em *templates*



Fonte: Autoria própria, 2014.

Caso não seja necessário adicionar alguma das tarefas do *template*, basta o usuário desmarcar a caixa de seleção da tarefa não desejada (Figura 4.7).

Figura 4.7 – Caixa de seleção de tarefas em criação de tarefas baseado em *templates*



Fonte: Autoria própria, 2014.

4.3 Características do Processo de Desenvolvimento

Para criação de um *software*, é necessário planejar todo o processo de desenvolvimento a fim de produzir um produto de qualidade. No desenvolvimento do "STM" foi utilizado um conjunto de atividades fundamentais com a finalidade de tornar o produto final uniforme e de fácil manutenção, contribuindo com o seu processo de qualificação.

4.3.1 Especificação de *software*

O processo de especificação de *software* consiste em entender e mapear as necessidades do cliente. São criados, em alto nível, documentos com descrição de funcionalidades que serão desenvolvidas e alguns protótipos das principais telas da aplicação. Nesse momento também serão identificados quais são as principais restrições de projeto, sejam de infraestrutura ou de desenvolvimento (SOMMERVILLE, 2007). Essa documentação com a visão geral do *software* irá facilitar o entendimento do cliente com relação ao que será desenvolvido.

Baseado na premissa de que era necessário um sistema de gerenciamento de bolsistas - em que seria possível cadastrar um bolsista, os seus horários disponíveis no laboratório e as tarefas associadas a ele - foram identificadas as principais funcionalidades e módulos do aplicativo (Tabela 4.1).

Tabela 4.1 – Módulos a serem desenvolvidos no "STM"

Módulo	*Descrição*
Gerenciamento de usuários	Consiste na manutenção (adição e alteração) do cadastro de usuários do sistema e, também, criação de vínculo entre usuário e projeto(s).
Gerenciamento de tarefas	Módulo para criação, edição e acompanhamento das tarefas cadastradas pelo usuário.
Gerenciamento de projetos	Utilizado para cadastro e consulta de dados dos projetos.
Gerenciamento de reuniões	Módulo de agendamento de reuniões e armazenamento de atas.
Gerenciamento de atividades e mensagens	Área destinada à comunicação com o usuário. Nela seria possível enviar mensagens entre os usuários cadastrados no sistema, além de expor as últimas atividades concluídas e/ou modificadas.

Fonte: Autoria própria, 2014.

Uma vez apontado pelos *stakeholders* quais seriam as principais características do aplicativo, iniciou-se o processo de elucidação e priorização dos requisitos iniciais. A partir desse processo foi criado um *backlog*[2] (RUBIN, 2012) contendo tarefas e as principais características dos componentes a serem desenvolvidas.

Na especificação inicial do aplicativo foram gerados outros documentos para ilustrar ao usuário o que seria criado. Com a determinação inicial dos campos para alguns módulos, foi produzido um documento inicial da modelagem do banco de dados (APÊNDICE A) apresentando as primeiras tabelas da base de dados e seus relacionamentos como também foi executada a prototipação de algumas telas iniciais (APÊNDICE B) do sistema.

4.3.2 Projeto e implementação de *software*

Esta é a fase em que é produzido um aplicativo funcional a partir de uma especificação de sistema. Também é o momento em que algumas especificações são discutidas e, em alguns casos, refinadas para corrigir quaisquer detalhes ou erros encontrados devido a, por exemplo, alguma limitação técnica identificada após o período de especificação.

Este também é o período em que são estabelecidas as bases do sistema. Um documento contendo definições de padrões de desenvolvimento (APÊNDICE C) foi criado visando uma codificação uniforme, facilitando futuras manutenções dos artefatos desenvolvidos. Foi estabelecido nesse momento qual seria a estrutura de diretórios (APÊNDICE D) utilizada, seja para alocação dos arquivos desenvolvidos assim como a alocação de diretórios específicos para adição de bibliotecas livres adquiridas de terceiros.

Uma vez melhor definidas as descrições de requisitos e estabelecido os padrões, iniciou-se o processo de desenvolvimento do "STM". O período inicial de codificação focou no desenvolvimento dos módulos base da aplicação como, por exemplo, as classes básicas de conexão e manipulação com banco de dados. Após, iniciou-se a implementação dos primeiros componentes definidos pelos *stakeholders* do sistema.

4.3.3 Validação de *software*

O processo de validação de *software* consiste em garantir que os componentes desenvolvidos estejam de acordo com a descrição do requisito, ou seja, a funcionalidade foi projetada e implementada conforme as necessidades dos *stakeholders*. Nesta etapa também

[2] No processo ágil *backlog* ou *product backlog* é "uma lista de priorização das funcionalidades desejadas de um produto. Provê um entendimento centralizado e compartilhado do que se deve construir e a ordem em que se deve ser construído". (RUBIN, 2012, p. 99) - (tradução livre)

são criados testes manuais e/ou automatizados, que irão exercitar os fluxos previamente descritos de forma a garantir que não há falhas de codificação.

Para o “STM”, inicialmente foi lançado aos usuários o acesso a alguns módulos da ferramenta para validação e teste dos componentes já disponíveis. Dessa forma, era possível identificar eventuais falhas e a conformidade do conteúdo construído em relação à descrição do requisito. À medida que novas funcionalidades eram desenvolvidas e concluídas, elas eram adicionadas ao repositório de teste para acesso dos usuários.

Além da validação com usuários, foram criados testes unitários para os principais componentes e testes de sistema para validação das principais interfaces disponíveis.

4.4 Evolução do *Software*

A evolução ou manutenção de *software* se faz necessário quando é identificada uma falha no sistema ou quando surge a necessidade de disponibilizar novos campos ou novas funcionalidades.

Durante o desenvolvimento do “STM” houve a necessidade de modificações e nova priorização das tarefas a serem implementadas. A seguir, serão ilustradas algumas das mudanças realizadas.

4.4.1 Interface

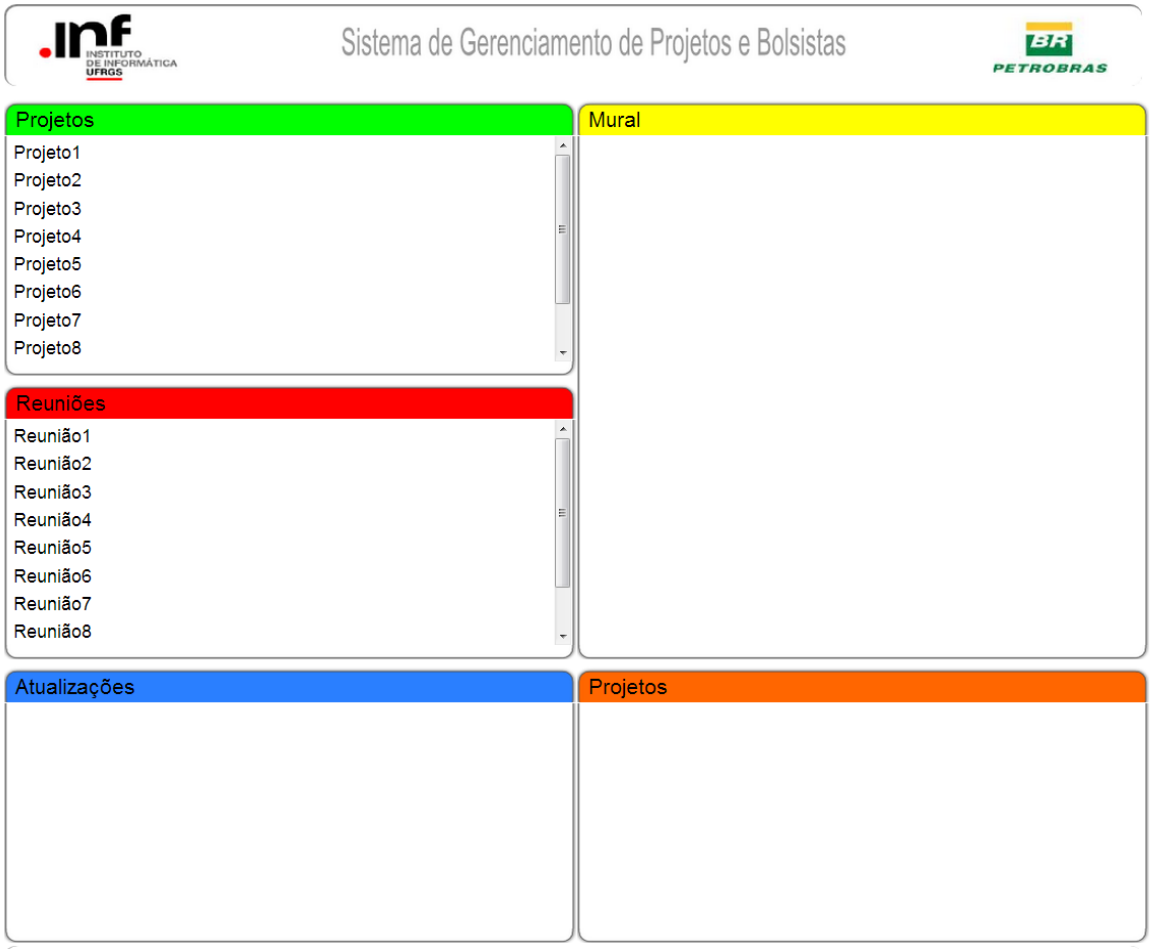
A prototipação das telas foi útil no processo de desenvolvimento do “STM”. A sua utilização facilitou na validação dos dados adicionados às tarefas e na disposição dos módulos exibidos aos usuários, uma vez que os *stakeholders* compreendiam melhor o seu comportamento. Para tal, foram criados protótipos em *HTML* estático e em esboço, em papel (APÊNDICE E), para apresentação das funcionalidades que estavam sendo criadas.

Também baseado nos protótipos de interface, foi possível priorizar o desenvolvimento de algumas funcionalidades, já que a visualização dos componentes facilita a tomada de decisão, o que possibilita eventuais alterações de escopo de desenvolvimento. Ou seja, mesmo apresentando esse tipo de documentação, ainda houve diversas modificações nas definições do aplicativo durante o desenvolvimento.

A remodelagem das telas foi necessária em alguns momentos para que novas funcionalidades pudessem ser inseridas ao contexto da aplicação. Um exemplo de

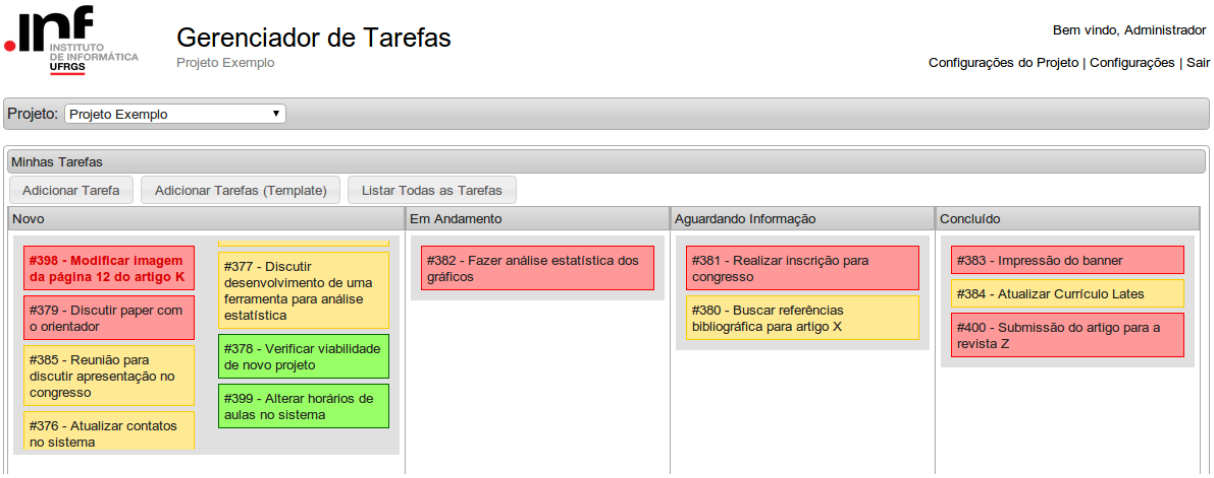
modificação foi a alteração da página principal (Figura 4.8 e Figura 4.9), necessária para incorporar a apresentação de tarefas no formato *Kanban*.

Figura 4.8 – Primeira versão da tela principal do "STM"



Fonte: Autoria própria, 2014.

Figura 4.9 – Última versão da tela principal do "STM" - após alteração de *layout* para incorporar *Kanban*



Fonte: Autoria própria, 2014.

4.4.2. Funcionalidades

Inicialmente foram definidos e priorizados os módulos descritos anteriormente na Tabela 4.1. Entretanto, durante o processo de desenvolvimento, identificou-se que algumas das funcionalidades apresentadas eram de baixo valor agregado.

Os módulos de “gerenciamento de reuniões” e “gerenciamento de atividades e mensagens” são exemplos de funcionalidades consideradas de baixa prioridade e foram removidas do escopo de desenvolvimento do sistema. Após especificação detalhada e de sua apresentação aos *stakeholders* foi possível identificar que ambas não seriam necessárias, já que tais informações não precisavam ser armazenadas no “STM”, uma vez que é possível gerenciar compromissos, reuniões e comunicação geral entre usuários através de *e-mail*.

Além disso, também foram identificadas funcionalidades a serem adicionadas, o que permitiria melhor experiência aos usuários do aplicativo. São elas:

- Mecanismo gráfico de visualização de tarefa - *Kanban*;
- Criação de tarefas baseada em *templates*;
- Envio de *e-mails* na manipulação de tarefas;
- Possibilidade de anexar arquivos em tarefas.

4.5 Refatoração e Melhorias de Classes e Módulos

As consequências de, inicialmente, não se utilizar de forma adequada os processos de engenharia de *software*, pode fazer com que a manutenção e implementação de outros módulos durante o desenvolvimento seja bem difícil, algumas vezes demandando esforço extra para novas definições de desenvolvimento e necessitando o refatoramento de módulos até então dados como concluídos (ENGHOLM, 2010). Além disso, seguir todos os padrões definidos não implica em ter um código eficiente.

Durante o desenvolvimento do “STM” foi possível identificar alguns problemas que exigiram a refatoração de código para permitir a criação de novos módulos e melhorar o desempenho.

Um dos problemas mais impactantes foi o de performance. Todas as classes de domínio haviam sido construídas com consultas completas à base de dados. Dessa forma, todo o conteúdo dos objetos criados era coletado para qualquer consulta.

Baseado no resultado da execução de um teste, em que um usuário possuía 280 tarefas não concluídas ou canceladas vinculadas a ele, foi identificado que eram realizadas mais de

13000 consultas ao banco de dados. Ou seja, nesse contexto, demorava aproximadamente 22 segundos para renderizar a página principal. Para a solução do problema, foi reduzido o escopo das consultas, além de construir consultas mais complexas em *SQL*. Isso fez com que o tempo de renderização diminuísse para apenas aproximadamente 3 segundos.

4.6 Cenários de Uso do "STM"

Algumas das principais funcionalidades da ferramenta "STM" são a adição de horários e adição de contatos de um usuário, em que os cenários de uso serão mostrados a seguir.

4.6.1 Horários do Usuário

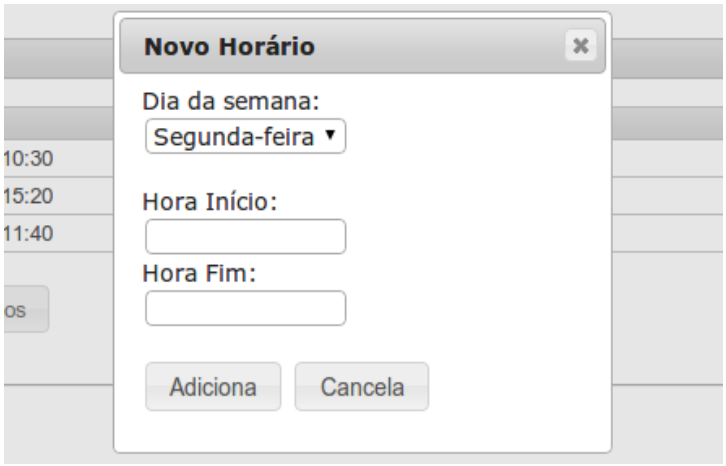
Uma vez cadastrado no "STM", estará disponível ao usuário um formulário contendo, além de suas informações pessoais e projetos vinculados, uma tabela para inserção dos horários alocados para o projeto (Figura 4.10).

Figura 4.10 – Tabela de horários do usuário.

Fonte: Autoria própria, 2014.

Para adicionar uma nova entrada na tabela, o usuário deve pressionar o botão "Adicionar". Uma nova janela será exibida (Figura 4.11), em que o usuário deverá determinar o dia da semana disponível, além do horário de início e de fim da sua disponibilidade para o projeto.

Figura 4.11 – Tela de cadastro de horário



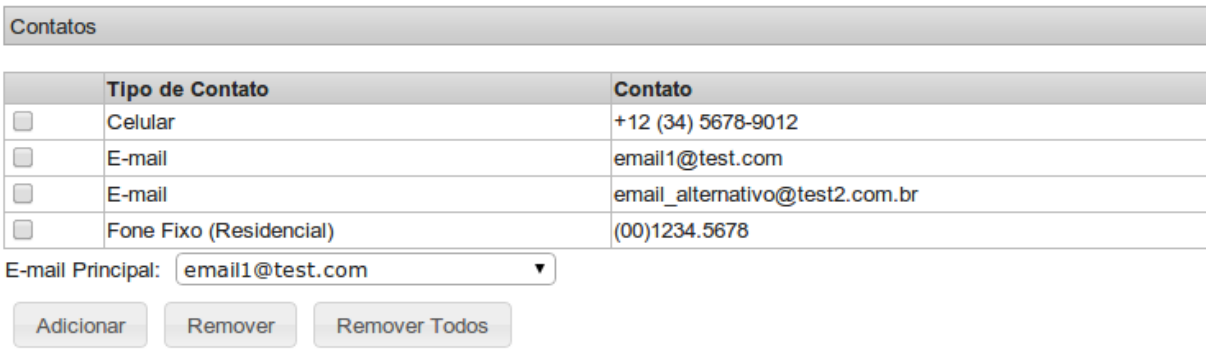
Fonte: Autoria própria, 2014.

Ao pressionar o botão “Adiciona” da tela, a nova entrada será inserida na tabela de horários. Por fim, para salvar a nova entrada, deverá ser pressionado o botão “Salvar” da tela principal.

4.6.2 Contatos do Usuário

Todos os contatos cadastrados podem ser encontrados na seção “Contatos” do formulário de detalhes do usuário (Figura 4.12). Estão presentes na tabela o tipo de contato e o seu valor atribuído. Neste formulário também é possível determinar, através do seletor “*E-mail* principal”, qual deverá ser o endereço de *e-mail* primário de comunicação com o usuário.

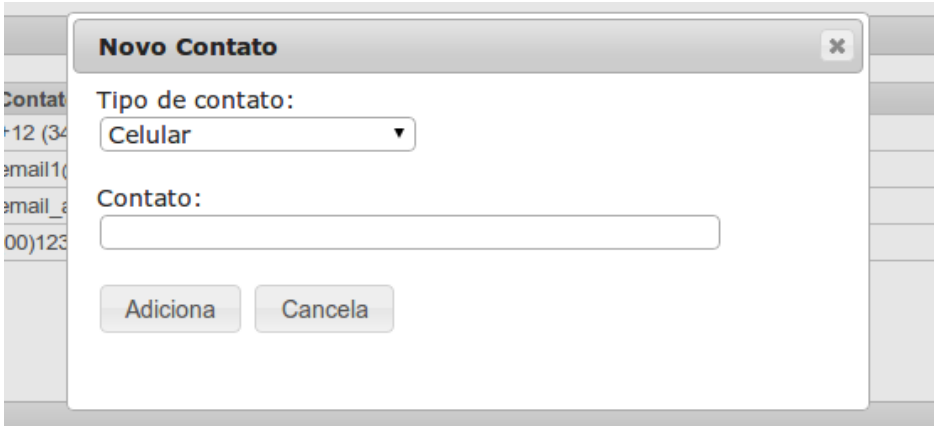
Figura 4.12 – Seção de contatos do formulário do usuário



Fonte: Autoria própria, 2014.

Uma vez que o usuário deseje adicionar um novo contato, o botão “Adicionar” deverá ser pressionado, resultando na abertura de uma nova janela (Figura 4.13). Neste formulário, será especificado o tipo de contato, previamente cadastrado, e o contato do usuário.

Figura 4.13 – Formulário de cadastro de contatos



Fonte: Autoria própria, 2014.

Assim que finalizado o seu preenchimento, o botão “Adiciona” deverá ser pressionado. O novo valor de contato será acrescentado na tabela de contatos. No caso de ser inserido um novo *e-mail* como contato, este também será adicionado no campo “*E-mail* principal”, possibilitando sua seleção. Para armazenar a novo contato e a opção de contato principal, o botão “Salvar” da janela deverá ser pressionado.

5 CONCLUSÃO

Durante o desenvolvimento do “Smart Task Manager” foi possível perceber a importância do contato frequente entre o desenvolvedor e os *stakeholders*. Nesse processo, houve alterações e correções importantes nas definições dos requisitos, assim como, no comportamento de alguns módulos, além da adição de novas funcionalidades que inicialmente não haviam sido previstas. Tais modificações só foram realizadas apropriadamente devido a esse contato.

Considera-se também que para obter resultados satisfatórios no final de cada ciclo de desenvolvimento, foi fundamental apoiar-se nos conceitos da engenharia de *software*, na produção de documentos de especificação e planejamento, como também, na execução do projeto.

Além disso, conhecer ferramentas de Gerenciamento de Projetos e Gestão da Qualidade para expor informações, como *Kanban*, contribuiu no processo de identificação de qual seria uma maneira adequada de apresentar ao usuário as tarefas vinculadas a ele em um contexto mais amplo, incluindo detalhes de *status* e, principalmente, prioridades.

Por fim, ao comparar o “STM” com as ferramentas existentes foi possível perceber que não há grandes diferenças entre elas. Entretanto, o *software* desenvolvido é o que melhor está adaptado às necessidades dos *stakeholders* no momento, pois os principais requisitos acordados foram desenvolvidos, incluindo funcionalidades específicas. Para que ele continue atendendo tais necessidades, torna-se importante incluir o processo de manutenção contínua, desenvolvendo funcionalidades e adaptando a ferramenta às novas demandas e realidades do projeto.

5.1 Trabalhos Futuros

Surgiram algumas ideias durante a execução desse trabalho que, uma vez desenvolvidas, iriam contribuir para se obter melhores resultados e experiências tanto na utilização do aplicativo como no gerenciamento de projetos e tarefas. São eles:

- Desenvolvimento de *web services* como interface para acesso de clientes terceiros aos dados da ferramenta;
- Criação de interface para dispositivos móveis;

- Adição de novos relatórios conforme a necessidade;
- Implementação de mecanismos gráficos para exposição de dados - Diagrama de *Gantt*;
- Desenvolvimento de mecanismos de exportação de dados de projetos e tarefas para arquivos no formato *CSV* e formato texto;
- Desenvolvimento de mecanismos de *backup* dos dados da ferramenta;
- Inclusão de módulo para customização do *layout*;
- Adição de *checklist* ao módulo de tarefas.
- Avaliação da quantidade de recursos necessários para instalação e uso da ferramenta em um ambiente de produção.

REFERÊNCIAS

AGUIAR, Giancarlo de França; PEINADO, Jurandir. Compreendendo o *Kanban*: um ensino interativo ilustrado. **Da Vinci**, Curitiba, v. 4, n. 1, p. 133-146, 2007. Disponível em: <http://www.up.com.br/davinci/4/08%20Compreendendo%20o%20Kanban%20um%20ensino%20interativo%20ilustrado.pdf>. Acesso em: 05 abr. 2014.

DOTPROJECT. Disponível em: <http://www.dotproject.net/>. Acesso em: 20 mai. 2014.

ENGHOLM, Hélio Jr. **Engenharia de software na prática.** 1.ed. São Paulo: Novatec, 2010.

MANTIS. Disponível em: <https://www.mantisbt.org/>. Acesso em: 20 mai. 2014.

MICROSOFT DEVELOPER NETWORK. Disponível em <http://msdn.microsoft.com/en-us/library/ff649643.aspx>. Acesso em: 23 jun. 2014.

PROJECT MANAGEMENT INSTITUTE. **Um Guia do Conhecimento em Gerenciamento de Projetos**: Guia PMBOK. 4.ed. Pennsylvania: [s.n.], 2008.

REDMINE. Disponível em: <http://www.redmine.org/>. Acesso em: 20 mai. 2014.

RUBIN, Kenneth S. **Essential Scrum**: a practical guide to the most popular agile process. Upper Saddle River: Addison-Wesley, 2012.

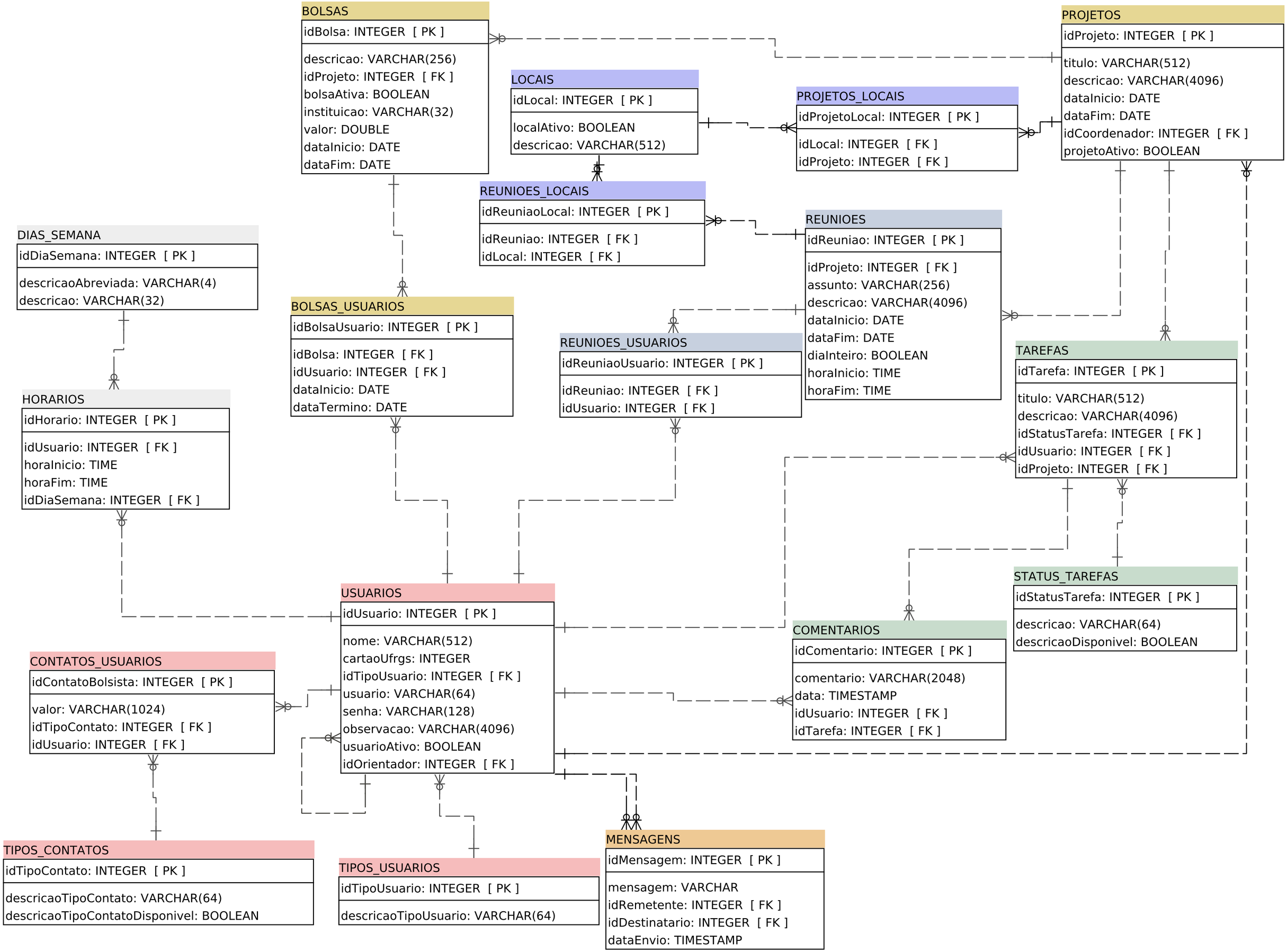
SLACK, Nigel; CHAMBERS, Stuart; JOHNSTON, Robert. **Administração da Produção.** 3.ed. São Paulo: Atlas, 2009.

SOMMERVILLE, Ian. **Engenharia de Software.** 8.ed. São Paulo: Pearson Addison-Weley, 2007.

TRAC. Disponível em: <http://trac.edgewall.org/>. Acesso em: 20 mai. 2014.

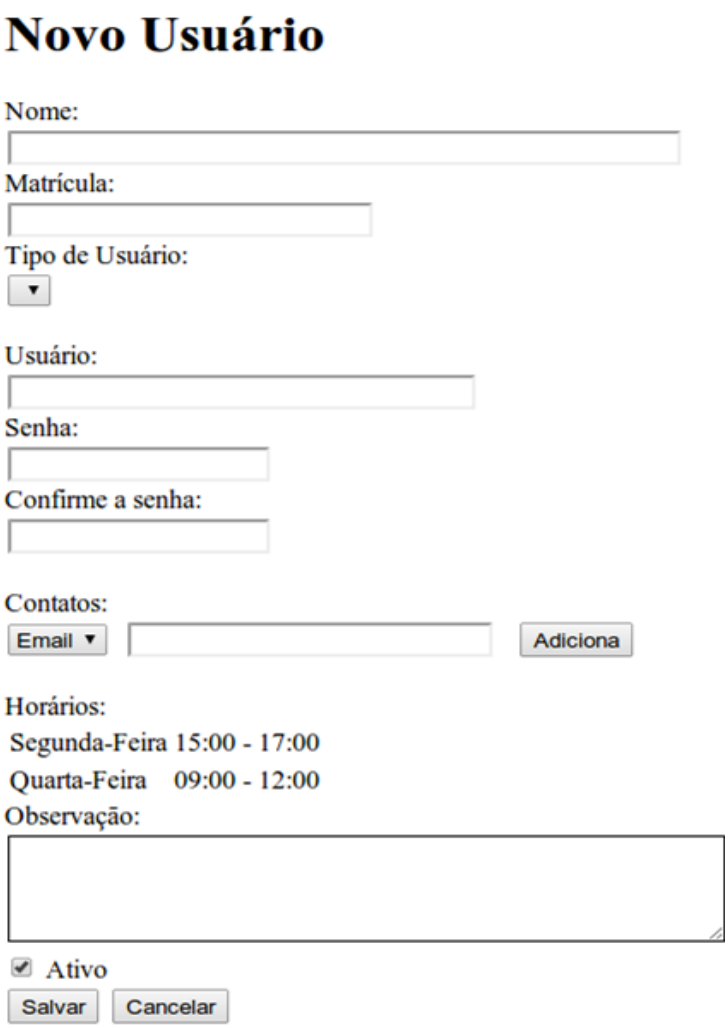
WIRFS-BROCK, Rebecca J.; JOHNSON, Ralph E. Surveying current research in object-oriented design. **Communication of ACM.** v. 33, n. 9, p. 104-124, set. 1990.

APÊNDICE A – MODELAGEM DE BANCO DE DADOS


BOLSAS
idBolsa: INTEGER [PK]
descricao: VARCHAR(256)
idProjeto: INTEGER [FK]
bolsaAtiva: BOOLEAN
instituicao: VARCHAR(32)
valor: DOUBLE
dataInicio: DATE
dataFim: DATE
PROJETOS
idProjeto: INTEGER [PK]
titulo: VARCHAR(512)
descricao: VARCHAR(4096)
dataInicio: DATE
dataFim: DATE
idCoordenador: INTEGER [FK]
projetoAtivo: BOOLEAN
LOCAIS
idLocal: INTEGER [PK]
localAtivo: BOOLEAN
descricao: VARCHAR(512)
PROJETOS_LOCAIS
idProjetoLocal: INTEGER [PK]
idLocal: INTEGER [FK]
idProjeto: INTEGER [FK]
REUNIOES_LOCAIS
idReuniaoLocal: INTEGER [PK]
idReuniao: INTEGER [FK]
idLocal: INTEGER [FK]
REUNIOES
idReuniao: INTEGER [PK]
idProjeto: INTEGER [FK]
assunto: VARCHAR(256)
descricao: VARCHAR(4096)
dataInicio: DATE
dataFim: DATE
diaInteiro: BOOLEAN
horaInicio: TIME
horaFim: TIME
DIAS_SEMANA
idDiaSemana: INTEGER [PK]
descricaoAbreviada: VARCHAR(4)
descricao: VARCHAR(32)
BOLSAS_USUARIOS
idBolsaUsuario: INTEGER [PK]
idBolsa: INTEGER [FK]
idUsuario: INTEGER [FK]
dataInicio: DATE
dataTermino: DATE
REUNIOES_USUARIOS
idReuniaoUsuario: INTEGER [PK]
idReuniao: INTEGER [FK]
idUsuario: INTEGER [FK]
TAREFAS
idTarefa: INTEGER [PK]
titulo: VARCHAR(512)
descricao: VARCHAR(4096)
idStatusTarefa: INTEGER [FK]
idUsuario: INTEGER [FK]
idProjeto: INTEGER [FK]
HORARIOS
idHorario: INTEGER [PK]
idUsuario: INTEGER [FK]
horaInicio: TIME
horaFim: TIME
idDiaSemana: INTEGER [FK]
STATUS_TAREFAS
idStatusTarefa: INTEGER [PK]
descricao: VARCHAR(64)
descricaoDisponivel: BOOLEAN
USUARIOS
idUsuario: INTEGER [PK]
nome: VARCHAR(512)
cartaoUfrgs: INTEGER
idTipoUsuario: INTEGER [FK]
usuario: VARCHAR(64)
senha: VARCHAR(128)
observacao: VARCHAR(4096)
usuarioAtivo: BOOLEAN
idOrientador: INTEGER [FK]
COMENTARIOS
idComentario: INTEGER [PK]
comentario: VARCHAR(2048)
data: TIMESTAMP
idUsuario: INTEGER [FK]
idTarefa: INTEGER [FK]
CONTATOS_USUARIOS
idContatoBolsista: INTEGER [PK]
valor: VARCHAR(1024)
idTipoContato: INTEGER [FK]
idUsuario: INTEGER [FK]
TIPOS_CONTATOS
idTipoContato: INTEGER [PK]
descricaoTipoContato: VARCHAR(64)
descricaoTipoContatoDisponivel: BOOLEAN
TIPOS_USUARIOS
idTipoUsuario: INTEGER [PK]
descricaoTipoUsuario: VARCHAR(64)
MENSAGENS
idMensagem: INTEGER [PK]
mensagem: VARCHAR
idRemetente: INTEGER [FK]
idDestinatario: INTEGER [FK]
dataEnvio: TIMESTAMP

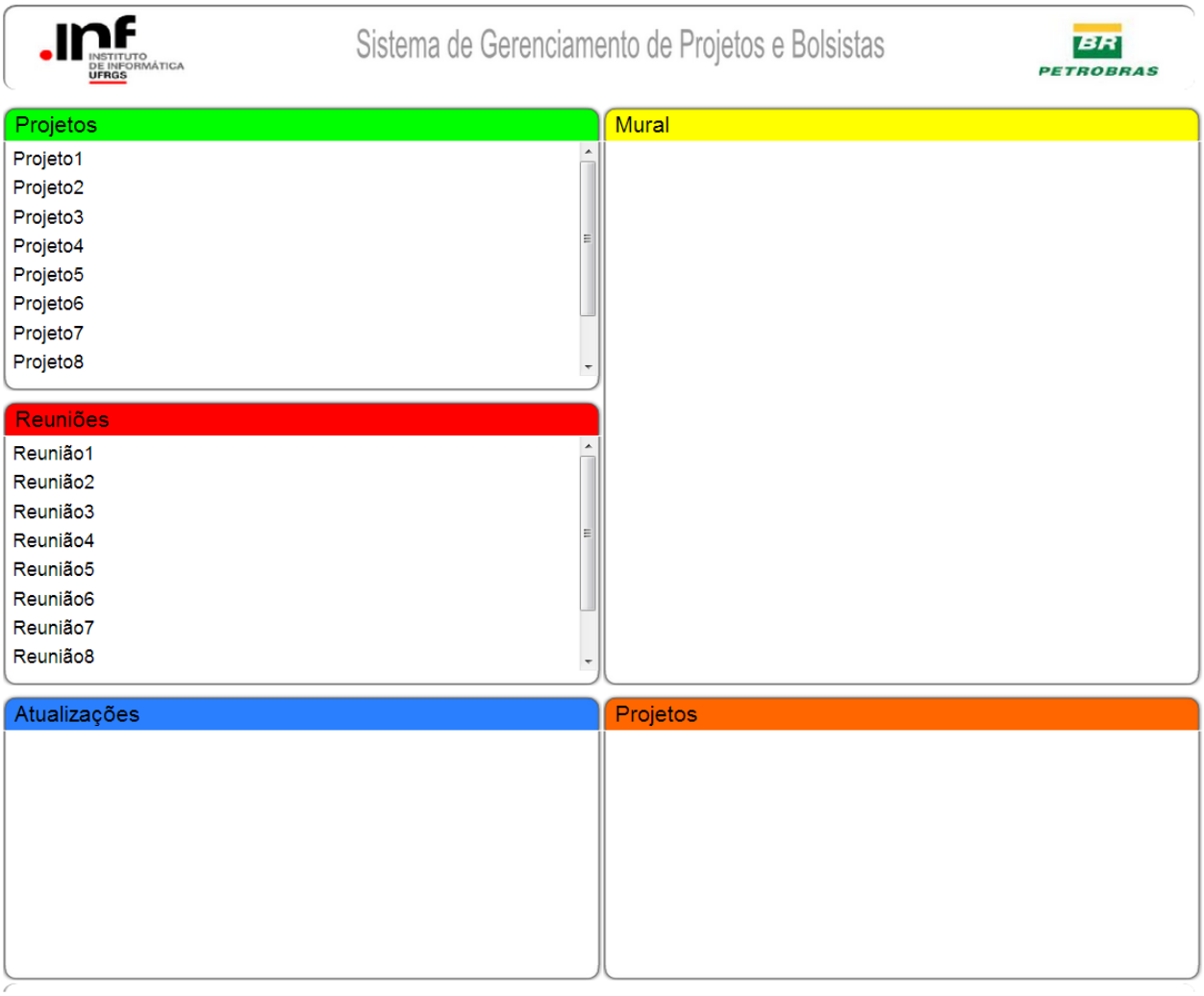
APÊNDICE B – PROTÓTIPO INICIAL DE TELAS

Formulário para criação de novo usuário



Formulário para criação de nova tarefa

Tela inicial do gerenciador



APÊNDICE C – PADRÃO DE CODIFICAÇÃO DO SISTEMA

Padrão de Desenvolvimento - Gerenciador de Tarefas

1. Padrões de Nomes

1.1. Arquivos

1.1.1. Camada de apresentação/layout (View): Será utilizado o padrão: <nomeDoArquivo>.php.
Exemplo: “criaUsuario.php”

1.1.2. Camada de Controle (Controller) - JavaScript: <nomeDoArquivo>.js.
Exemplo: “validaDados.js”.

1.1.3. Camada de Controle (Controller) - PHP: <nomeDoArquivo>.php.
Exemplo: “validaDados.php”.

1.1.4. Camada de Modelo (Model): <NomeDoArquivo>.php.
Exemplo:“ConexaoBanco.php”.

1.1.5. Interface de classes: I<NomeDoArquivo>.php.
Exemplo:“IConexaoBanco.php”.

1.2. Classes

O padrão a ser seguido para criação de classes será com todas as iniciais maiúsculas e concatenadas, ou seja, Class <NomeDaClasse>. O nome da classe deverá ser o mesmo do nome do arquivo.
Exemplo: Class ConexaoBancoDados

1.3. Variáveis
Os nomes de variáveis terá a primeira letra inicial minúscula e as demais palavras concatenadas com iniciais maiúsculas $<nomeDaVariavel>.
Exemplo: protected $nomeCompleto

1.4. Métodos
As regras de escritas para métodos serão idênticas a aplicada a variáveis.
Exemplo: private function calculaTotal()

1.5. Constantes
Para a declaração de constantes, utiliza-se todas as letras em maiúsculas, separadas por “_” (underscore), quando composta de mais de uma palavra, ou seja, <NOME_CONSTANTE>.
Exemplo: const SEGUNDA_FEIRA

2. Comentários

2.1. Cabeçalho do arquivo

```
/*
*  Arquivo:    <NomeDoArquivo>
*  Data:       <DataDaCriação>
*  Descricao:  <DescriçãoDaClasseOuLayout>
*/
```

2.2. Métodos

```
// Descrição do método desenvolvido
//
// @param      <TipoVariavel1> $<nomeVariavel1>                Descrição variavel 1
// @param      <TipoVariavel2> $<nomeVariavel2>                Descrição variavel 2
// @return     <TipoRetorno>                                   Descrição retorno
```

3. Espaçamentos e disposições

3.1. Posicionamento de chaves
Ao abrir e fechar chaves, as mesmas devem estar alinhadas conforme a identação do código.

Exemplo:

```
private function teste()
{
      ...
       if (Verdadeiro)
       {
             Código;
       }
       else
       {
             Código;
       }
       …
       try
       {
            ...
       }
       catch (Exception $e)
       {
            …
       }

}
```

APÊNDICE D – ESTRUTURA DE DIRETÓRIOS

Padrão de Desenvolvimento - Gerenciador de Tarefas

1. Estrutura de diretórios

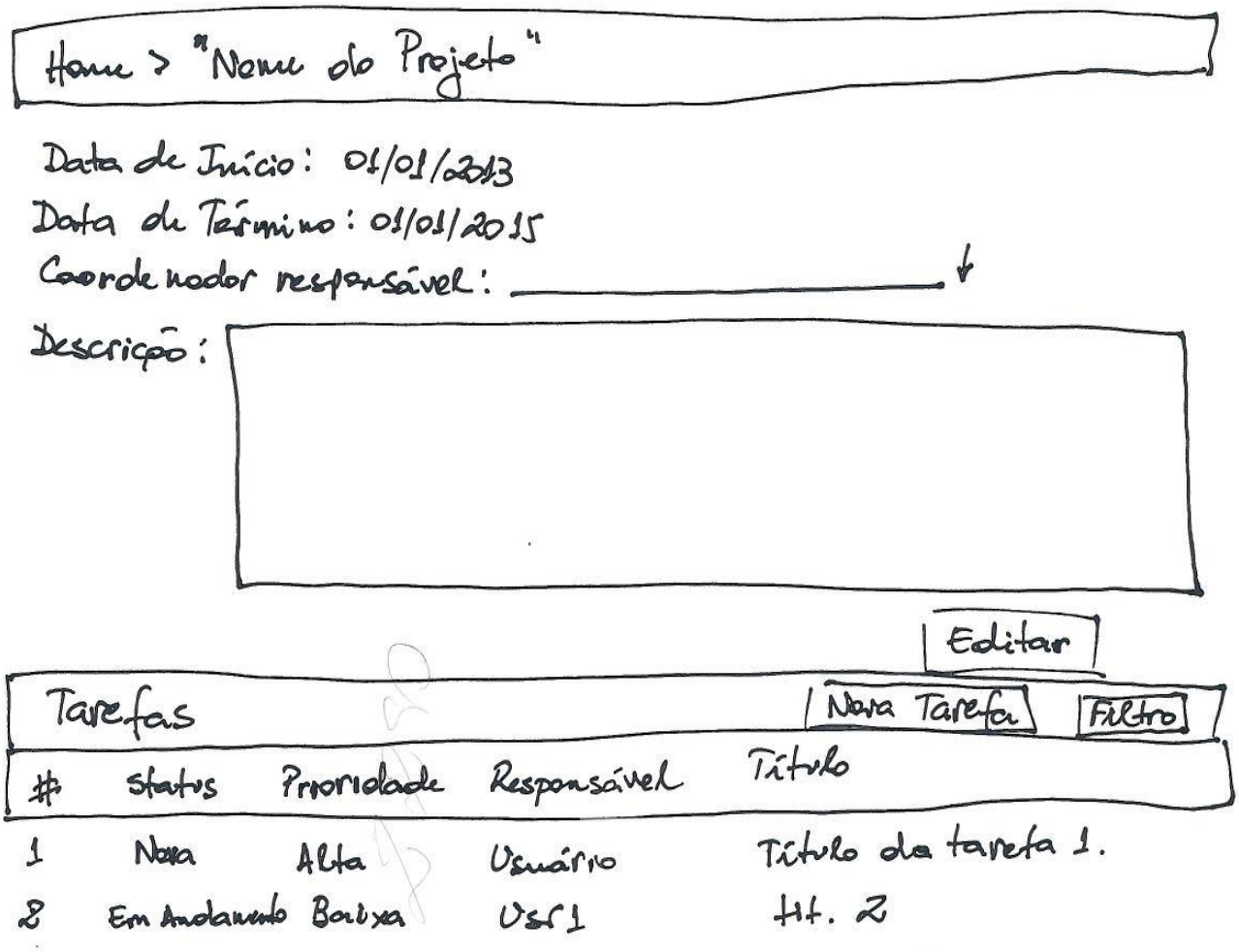
```
|-- config
|-- docs
|-- scripts
|-- src
|     |-- class
|     |-- controllers
|     |-- css
|     |-- img
|     |-- inc
|     |-- js
|     |-- lib
|     |-- tests
|     `-- views
`-- tools
```

Descrição:

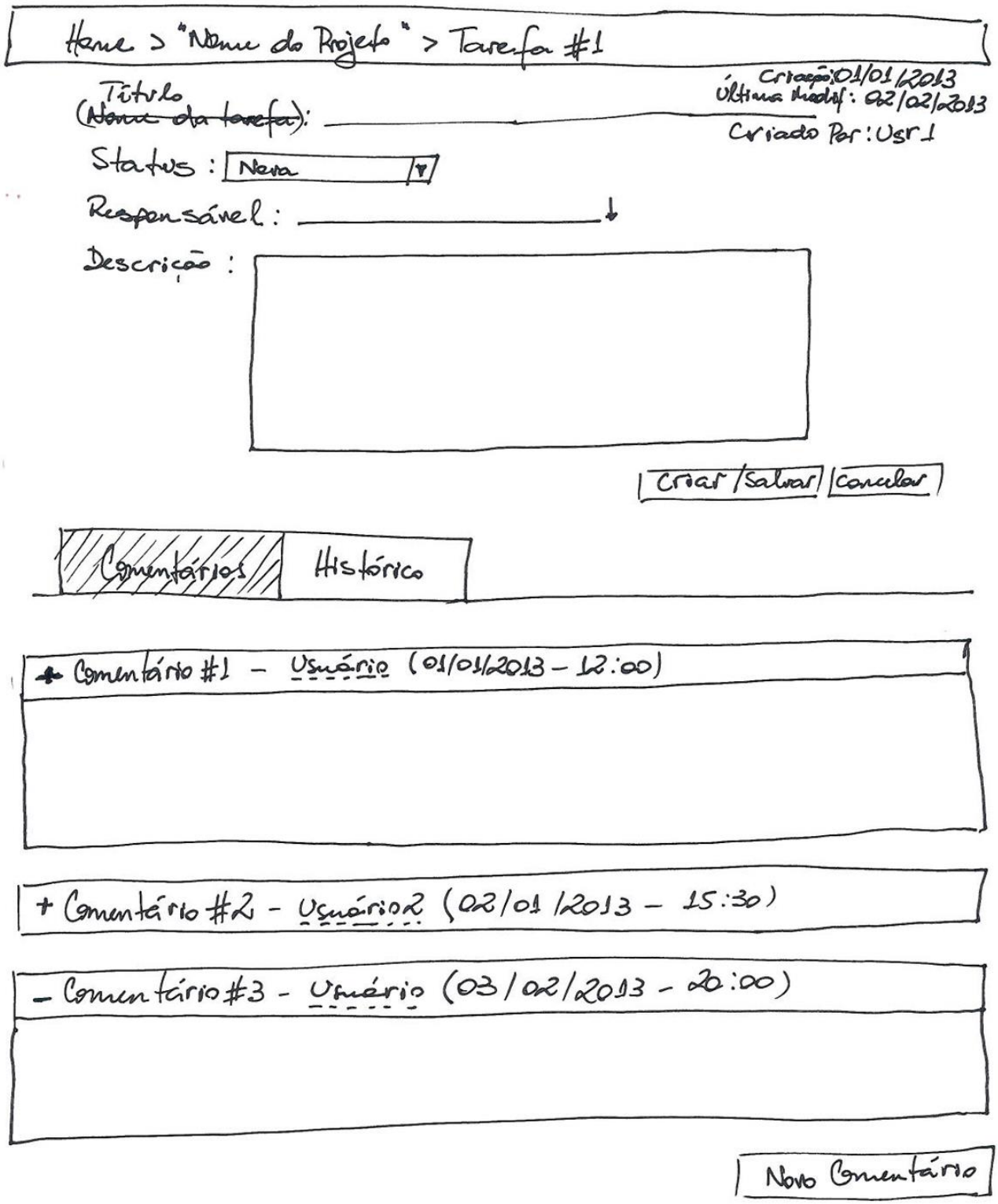
I	config	Arquivos de configuração do software desenvolvido.
II	docs	Arquivos de ajuda e documentação do software
III	scripts	Estrutura do banco de dados e informações iniciais
IV	src	Código fonte do aplicativo
V	src->class	Classes base do sistema (model)
VI	src->controllers	Controllers do sistema
VII	src->css	Arquivos Stylesheets
VIII	src->img	Arquivos de imagens geral
IX	src->inc	Arquivos includes (header/footer/etc)
X	src->js	Scripts JavaScript customizados
XI	src->lib	Bibliotecas terceiras (PHP e Javascript)
XII	src->tests	Testes unitários e do sistema
XIII	src->views	Arquivos de view do sistema
XV	tools	Ferramentas utilizadas para o desenvolvimento

APÊNDICE E – PROTÓTIPO DE TELAS: ESBOÇOS

VISUALIZAÇÃO DE PROJETO E TAREFAS



VISUALIZAÇÃO DE TAREFA



NOVO USUÁRIO