UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CURSO DE CIÊNCIAS DA COMPUTAÇÃO

Guilherme Furtado Pacheco

DESENVOLVIMENTO DE UM *SOFTWARE TOUCH SCREEN* PARA AUXILIAR NA GERÊNCIA DO *TASKBOARD*

Florianópolis - SC
2010/2

Guilherme Furtado Pacheco

DESENVOLVIMENTO DE UM *SOFTWARE TOUCH SCREEN* PARA AUXILIAR NA GERÊNCIA DO *TASKBOARD*

Trabalho de Conclusão de Curso apresentado como exigência parcial para obtenção do Diploma de Graduação em Ciências da Computação da Universidade Federal de Santa Catarina.

Orientadora: Prof. Dr. rer. nat. Christiane Gresse von Wangenheim, PMP

Florianópolis - SC
2010/2

Resumo

Contexto

Hoje no Brasil a grande maioria das empresas de *software* é micro ou pequena empresa, e o índice de mortalidade entre elas é muito alto. As empresas não conseguem manter-se no mercado e uma das causas é a falta de gerência sistemática de projetos. Uma das abordagens, sendo amplamente utilizada para gerência de projetos de *software*, é o SCRUM, que usa como um dos principais elementos de gerência o *taskboard*, tipicamente em papel.

Objetivo

O objetivo deste trabalho é desenvolver um *software multi touch* para auxiliar na gerência de *taskboards* a ser utilizado durante as reuniões do SCRUM. Esse aplicativo será desenvolvido para uso em telas *multi touch*, ficando mais fácil o manuseio durante as reuniões.

Metodologia

Para a realização do trabalho, primeiramente, será analisada a fundamentação teórica envolvendo as áreas de gerência de projetos, SCRUM e tecnologias *multi touch* além da análise do estado da arte. Em seguida será desenvolvida a aplicação, incluindo o desenvolvimento dos requisitos, projeto, implementação e testes. Ao final, o *software* será aplicado e avaliado numa situação na prática.

Resultados

Espera-se deste projeto um *software* para auxiliar na gerência do *taskboard* que seja viável a utilização do mesmo nas empresas, diminuindo os seus problemas de adoção de gerência de projetos e assim aumentando suas chances de sobrevivência e de sucesso no mercado.

Lista de Figuras

Sumário

1 Introdução

1.1 Contextualização

O mercado mundial de *software* e serviços atingiu em 2008 o valor de US$ 873 bilhões, e o Brasil se manteve no 12º lugar do *ranking* mundial, com um mercado interno de US$ 14,67 bilhões, segundo dados da pesquisa realizada pela Associação Brasileira de Empresas de *Software* (ABES, 2009).

Ainda segundo a ABES, somavam em 2008 aproximadamente 2.000 empresas de desenvolvimento de *software* no Brasil, dentre as quais 94% são micro ou pequenas empresas (MPEs). Empresas prestadoras de serviço com até 9 funcionários são consideradas micro empresas, as que possuem de 10 a 49 são consideradas pequenas empresas (MINISTÉRIO DA FAZENDA, 2000).

O índice de mortalidade das empresas de tecnologia é de cerca de 65% no período de cinco anos de existência (Universia, 2004). Diversos fatores contribuem para que essa taxa se mantenha alta como, por exemplo, as barreiras burocráticas impostas pelo governo ou pelo setor em que a companhia atua, ausência de capacidade gerencial e falta de planejamento adequado (Conferência IADIS Ibero-Americana, 2003).

Uma pesquisa realizada no mundo inteiro pelo The Standish Group Internationals mostrou que mais de 20% dos projetos de *software* são cancelados acarretando somente prejuízos (2009). Segundo o PMP Rodrigo Giraldelli, um fator contribuinte para essas estatísticas é a falta de tempo dedicada à fase de planejamento do projeto.

Gerência de projetos é definida como a aplicação de conhecimento, habilidades, ferramentas e técnicas a uma ampla gama de atividades para atender aos requisitos de um determinado projeto (PMBOK, 2008).

Porém, um grande problema das MPEs é a adequação às práticas de gerência de projetos, pois há certo custo para implantação, tempo de mudança e poucos recursos. (IADIS, 2003).

Uma solução para esse problema pode ser a adoção de métodos ágeis para gerência de projetos. O manifesto ágil é um conjunto de metodologias de gerência e

desenvolvimento de *software*, que visa criar uma estrutura de regras para ser aplicada nos projetos (Manifesto Ágil, 2001). A metodologia ágil engloba principalmente métodos para auxiliar no desenvolvimento de *software*, entre eles estão o *Feature Driven Development* (FDD) e o *Open Unified Process* (OpenUP). Dentre este conjunto de métodos ágeis também se destaca um voltado para gerência de projetos, o SCRUM (PMI, 2009).

O SCRUM foi criado inicialmente como um método para gerenciamento de projetos em uma fábrica automobilística, porém atualmente está sendo largamente aplicado a projetos de *software*. É um método alternativo para gerenciamento de projetos, pois coloca desenvolvedores e executivos lado a lado para alcançar um objetivo em comum. Além de ser fácil de aplicar, espera-se tornar a equipe muito mais produtiva (WIKIPEDIA, 2010).

O método SCRUM é caracterizado por iterações chamadas *sprints* (Figura 1). Estas *sprints* são intervalos curtos que podem variar de duas a quatro semanas. No início de um projeto é feito o planejamento do projeto, onde são feitas as especificações do projeto e deve ser criado o *backlog* do produto. A partir do *backlog* do produto são feitas reuniões de planejamento no início de cada *sprint*, onde são definidas as tarefas que cada membro da equipe deverá realizar dentro desta *sprint*. Para acompanhar o andamento da *sprint* são realizadas reuniões diárias onde os membros da equipe monitoram o andamento das tarefas. No fim de cada *sprint* deverá ser gerado um produto incrementado a ser incorporado no produto final. Ao final da *sprint* é realizada uma reunião de retrospectiva, onde a equipe reflete sobre a *sprint* passada. Até o produto final estar concluído serão realizados novos ciclos de *sprints*.

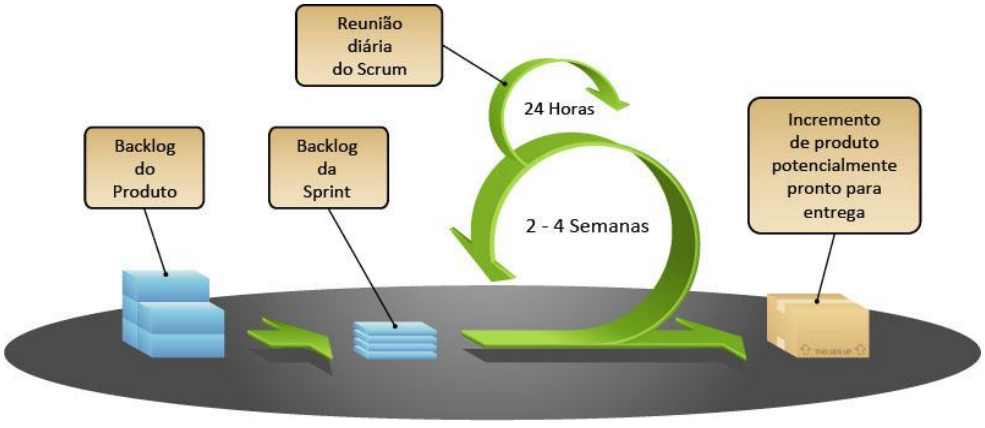

Figura 1. Processo do SCRUM (traduzido do site http://www.digibit.co.uk)

Entre os artefatos usados na gerência de projetos do SCRUM, um deles é o *taskboard* (Figura 2). *Taskboard* é um quadro de tarefas, geralmente divididas em quatro grupos: as tarefas a serem feitas, em andamento, as impedidas (que estão encontrando problemas na execução) e as finalizadas. O *taskboard* é usado tanto no planejamento da *sprint* quanto no monitoramento e controle do andamento da *sprint* atual. Desta forma o taskboard representa o principal artefato usado para a gerência de projetos no método SCRUM.

Figura 2. Taskboard, principal artefato do SCRUM

1.2 Problema

Tipicamente o *taskboard* é um quadro físico na sala da equipe do projeto, e as tarefas estão escritas em papéis colados no quadro. Desde a criação do SCRUM essa tem sido a forma mais prática de se gerenciar as *sprints*. A solução em papel mostrou-se bastante boa, principalmente pelo fato que ela suporta adequadamente a dinâmica do SCRUM. Porém, ela também tem algumas desvantagens, como por exemplo, o fato de que para manter um histórico do *taskboard* é preciso tirar uma fotografia do quadro todo dia.

Existem muitas ferramentas de *software* voltadas à assistência do método tradicional de gerência de projetos, por exemplo, DotProject (www.dotproject.net) e Microsoft Office Project (office.microsoft.com/en-us/project). Porém, do ponto de vista da gerência de projeto usando SCRUM, estas ferramentas não fornecem um suporte

customizado a este método. Dentro deste contexto existem também ferramentas computacionais de gerência do *taskboard*, por exemplo, FireSCRUM (www.firescrum.com) e o Pronto! (pronto.bluesoft.com.br), mas nenhuma delas apresenta a praticidade do quadro físico com papéis.

Desta forma, procurando uma melhoria na representação do *taskboard*, mas mantendo a dinâmica do SCRUM, uma solução pode ser o desenvolvimento de uma tela sensível ao toque para representar o *taskboard*.

1.3 Objetivo

O objetivo desse trabalho é desenvolver um *software multi touch* para auxiliar na gerência de *taskboards*, como elemento principal da gerência de projetos baseada no método SCRUM.

O *software* será um aplicativo para gerenciar o *taskboard*, a ser utilizado durante as reuniões do SCRUM. Esse aplicativo será desenvolvido para uso em telas *multi touch*, ficando mais fácil o manuseio durante as reuniões.

Objetivos específicos:

1. Analisar o contexto do trabalho, cobrindo a área do SCRUM e principalmente a parte de gerência do *taskboard*.
2. Analisar o contexto em relação às interfaces de computador *multi touch*.
3. Analisar o estado da arte com relação às ferramentas existentes para gerência de *taskboards*.
4. Desenvolver um sistema *multi touch* para gerência do *taskboard* em uma mesa interativa.
5. Aplicar e avaliar o sistema desenvolvido.

Limitações:

Foco principalmente na gerência do *taskboard* do SCRUM e em telas interativas, dando menos importância ao suporte a outras características do SCRUM, como o *burn down chart*.

A visão do uso do *software* é voltada às telas interativas verticais, porém por restrições de recursos, neste trabalho será desenvolvido um aplicativo para uma mesa

interativa, possibilitando uma transferência fácil no futuro também para as telas verticais.

1.4 Justificativa

Dispositivos *multi touch* estão em pleno crescimento, é uma tecnologia nova que promete mudar a maneira como o ser humano se relaciona com as máquinas. Com esse trabalho pretende-se adentrar nessa nova tecnologia, desenvolvendo uma aplicação feita para uso em telas sensíveis ao toque com a finalidade de gerenciar o *taskboard*.

A aplicação do aplicativo a ser desenvolvido também permitirá obter novo conhecimento sobre a aplicabilidade deste tipo de ferramenta no método SCRUM.

Esse trabalho contribuiria tanto para as empresas que já utilizam a metodologia SCRUM para gerência de projetos quanto para empresas que não tem um método de gerência bem definido, tornando mais fácil a aplicação de um método que é conhecido por toda a comunidade de profissionais de *software*.

Aumentando as chances de empresas adotarem o SCRUM para gerenciar seus projetos, aumentam também as chances de conseguirem entregar os projetos no prazo, dentro do orçamento e com o escopo definido, aumentando o sucesso e a competitividade das empresas no mercado.

1.5 Metodologia

O presente trabalho é realizado em quatro etapas:

Etapa 1. Fundamentação teórica

É realizada a análise da literatura referente à gerência de projetos, mais profundamente sobre o método SCRUM, buscando documentar os princípios básicos, características e artefatos necessários para uma boa aplicação do método. Buscar também fatores que influenciam no sucesso ou não na gerência de projetos, principalmente na questão do uso do *taskboard*.

Posteriormente, é feita uma pesquisa com relação às interfaces de computador *multi touch*. Essa pesquisa tem como finalidade fundamentar os princípios da

tecnologia, buscar dados sobre a aplicação desta e sobre o crescimento de dispositivos que utilizam esta tecnologia.

Etapa 2. Análise do estado de arte

É feita uma análise sistemática de softwares existentes para SCRUM, a fim de analisar e comparar as funcionalidades dos *softwares* existentes em comparação com os requisitos de alto-nível identificados dentro do objetivo do presente trabalho.

Etapa 3. Desenvolvimento do aplicativo

Em seguida é desenvolvido um *software* para auxílio na gerência do *taskboard* com base no método SCRUM. Dentro desta etapa serão identificados os requisitos do sistema, realizado o projeto do *software* e a implementação.

Etapa 4. Aplicação e avaliação

O aplicativo desenvolvido é aplicado e avaliado por meio de um estudo de caso na prática acompanhando a gerência do *taskboard* em uma ou mais reuniões diárias.

É feita a definição da avaliação, onde deverão ser definidos os dados a serem coletados e como esses dados serão analisados. Tendo definida a avaliação, é utilizado o *software* desenvolvido e coletado os dados. Por fim é realizada a análise dos dados, seguindo a definição da avaliação feita anteriormente.

2 Fundamentação Teórica

Neste capítulo serão abordados os temas que tem relação com o trabalho sendo desenvolvido.

2.1 Gerência de Projetos

O Project Mangement Institute (PMI), uma das maiores entidades mundiais voltada à gerência de projetos, define gerência de projetos como sendo o processo através do qual se aplicam conhecimentos, capacidades, instrumentos e técnicas às atividades do projeto de forma a satisfazer as necessidades e expectativas de todos os interessados (PMI, 2009).

O PMI publicou *A Guide to the Project Management Body of Knowledge* (PMBOK Guide) pela primeira vez em 1987, numa tentativa de padronizar práticas e informações da gerência de projetos. Hoje é um padrão reconhecido internacionalmente, que define os fundamentos da gerência de projetos, atualmente definidos entre grupos de processos e áreas de conhecimento (PMI, 2009).

Segundo o PMBOK Guide, a gerência de projetos é composta de cinco grupos de processos, que são um conjunto de ações que levam o projeto adiante. Os grupos são ligados pelos resultados que produzem, geralmente a saída de um grupo de processos é a entrada do outro. De acordo com o PMI, são eles:

- **Iniciação** - Define e autoriza um projeto ou fase de um projeto.
- **Planejamento** - Define e refina os objetivos e planeja a ação necessária para alcançar os objetivos e o escopo para os quais o projeto foi realizado.
- **Execução** - Integra pessoas e outros recursos para realizar o plano de gerenciamento do projeto para o projeto.
- **Monitoramento e Controle** - Mede e monitora regularmente o progresso para identificar variações em relação ao plano de gerenciamento do projeto, de forma que possam ser tomadas ações corretivas quando necessário para atender aos objetivos do projeto.
- **Encerramento** - Formaliza a aceitação do produto, serviço ou resultado e conduz o projeto ou uma fase do projeto a um final ordenado.

As Áreas de Conhecimento da Gerência de Projetos descrevem os conhecimentos e práticas em gerência de projetos em termos dos processos que as compõem. Estes processos foram organizados em nove áreas de conhecimentos como descrito abaixo (PMI, 2009):

- **Gerência da Integração do Projeto** - Descreve os processos necessários para que os diversos elementos do projeto sejam adequadamente coordenados.
- **Gerência do Escopo do Projeto** - Define os processos necessários para que o projeto contemple todo o trabalho requerido, e nada mais que isso, para completar o projeto com sucesso.
- **Gerência do Tempo do Projeto** - Trata dos processos necessários para que o projeto termine dentro do prazo previsto.
- **Gerência do Custo do Projeto** - Relata os processos necessários para que o projeto seja completado dentro do orçamento previsto.
- **Gerência da Qualidade do Projeto** - Descreve os processos necessários para que as necessidades que originaram o desenvolvimento do projeto sejam satisfeitas.
- **Gerência dos Recursos Humanos do Projeto** - Considera os processos necessários para proporcionar a melhor utilização das pessoas envolvidas no projeto.
- **Gerência das Comunicações do Projeto** - Refere-se aos processos necessários para que a geração, a captura, a distribuição, o armazenamento e a pronta apresentação das informações do projeto sejam feitas de forma adequada e no tempo certo.
- **Gerência dos Riscos do Projeto** - Trata da identificação, análise e resposta a riscos do projeto.
- **Gerência das Aquisições do Projeto** - Aborda os processos necessários para a aquisição de mercadorias e serviços fora da organização que desenvolve o projeto.

Atualmente vem se falando muito sobre dois tipos de metodologias de gerência de projetos, as metodologias “tradicionais”, e as metodologias ágeis.

As metodologias tradicionais dão ênfase ao processo e ao controle do andamento do projeto. Utilizando essas metodologias o gerente de projeto tem uma boa aproximação de quanto o *software* vai custar e o tempo que vai levar. O preço dessa precisão é a pouca tolerância a mudanças no projeto.

As metodologias ágeis dão ênfase à adaptabilidade, o projeto deve adaptar-se de acordo com as necessidades do cliente. Têm como prioridade entregar periodicamente o *software* funcional à medida que os requisitos vão sendo cumpridos. Mudanças são aceitas no projeto, mesmo em estágio avançado de desenvolvimento, visando vantagens competitivas para o cliente.

2.1.1 SCRUM

SCRUM é um método ágil de gerência de projetos iterativo e incremental desenvolvido por Ken Schwaber e Jeff Sutherland na década de 90 baseando-se em suas próprias experiências no desenvolvimento de sistemas e processos (SCHWABER, 2001).

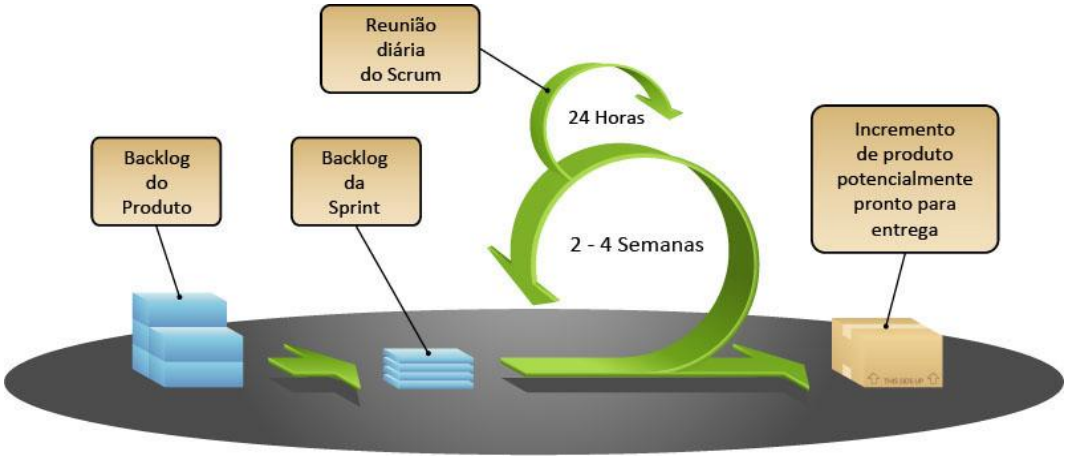

Figura 3. Processo do SCRUM (traduzido do site http://www.digibit.co.uk)

No início de um projeto a primeira coisa a ser feita é uma reunião de planejamento do projeto. Nessa reunião deve ser criado o *backlog* do produto, que contém as funcionalidades que o cliente espera que tenha no produto a ser desenvolvido.

A partir do *backlog* do produto podemos iniciar as *sprints*, que são os ciclos principais do SCRUM. A *sprint* tem duração de três a quatro semanas geralmente, e durante uma *sprint* temos várias reuniões do SCRUM.

A reunião de planejamento da *sprint* acontece no início da cada *sprint*. A primeira parte da reunião acontece com o cliente, com ele são definidas as prioridades do *backlog* do produto. A segunda parte da reunião acontece somente entre a equipe, com base no *backlog* do produto são definidos que tarefas serão realizadas no ciclo atual e definido o *backlog* da *sprint*, e o tempo que deverá durar a *sprint*.

Durante uma *sprint* acontecem diariamente reuniões de monitoramento do projeto. São reuniões curtas, cerca de 15 minutos, para manter a dinâmica do SCRUM e não atrapalhar a produtividade da equipe. Essas reuniões têm como objetivo monitorar o andamento da *sprint*, cada membro deverá informar no que ele trabalhou desde a última reunião e no que ele irá trabalhar até a próxima, além dos problemas que está enfrentando, caso haja algum. Com base nestas informações é atualizado o *taskboard* e o e o *burndown chart*.

Ao fim da *sprint* são feitas mais duas reuniões. A reunião de revisão da sprint é para revisar quais tarefas foram e quais tarefas não foram completadas, e demonstrar o que foi feito para o cliente. Na reunião de retrospectiva os membros da equipe refletem sobre a *sprint* finalizada, sobre o que ocorreu bem na *sprint*, e sobre o que pode ser melhorado para a próxima visando o levantamento de lições aprendidas.

O *taskboard* é um quadro dividido geralmente em três colunas. A primeira coluna representa o *backlog* da *sprint*, durante a *sprint* ficarão nessa coluna as tarefas que ainda não foram iniciadas. Na segunda coluna ficam as tarefas nas quais a equipe está trabalhando atualmente. Na terceira coluna ficam as tarefas já concluídas pela equipe naquela *sprint*. Podem ser adicionadas outras colunas, uma coluna do *backlog* do produto, ou para tarefas impedidas, mas deve-se evitar deixar o *taskboard* mais complexo. O *burndown chart* é um gráfico, que mostra quanto trabalho ainda resta ser feito em função do tempo restante do projeto.

Uma das principais características do SCRUM é a dinâmica. Manter a equipe participativa e motivada é essencial, esse é um dos papéis do *taskboard*. A figura 4 demonstra um *taskboard* no seu estado após a reunião de planejamento da *sprint* e antes da primeira reunião diária, quando o *backlog* da *sprint* já foi definido, mas ainda não foram designadas quais tarefas serão feitas na *sprint*.

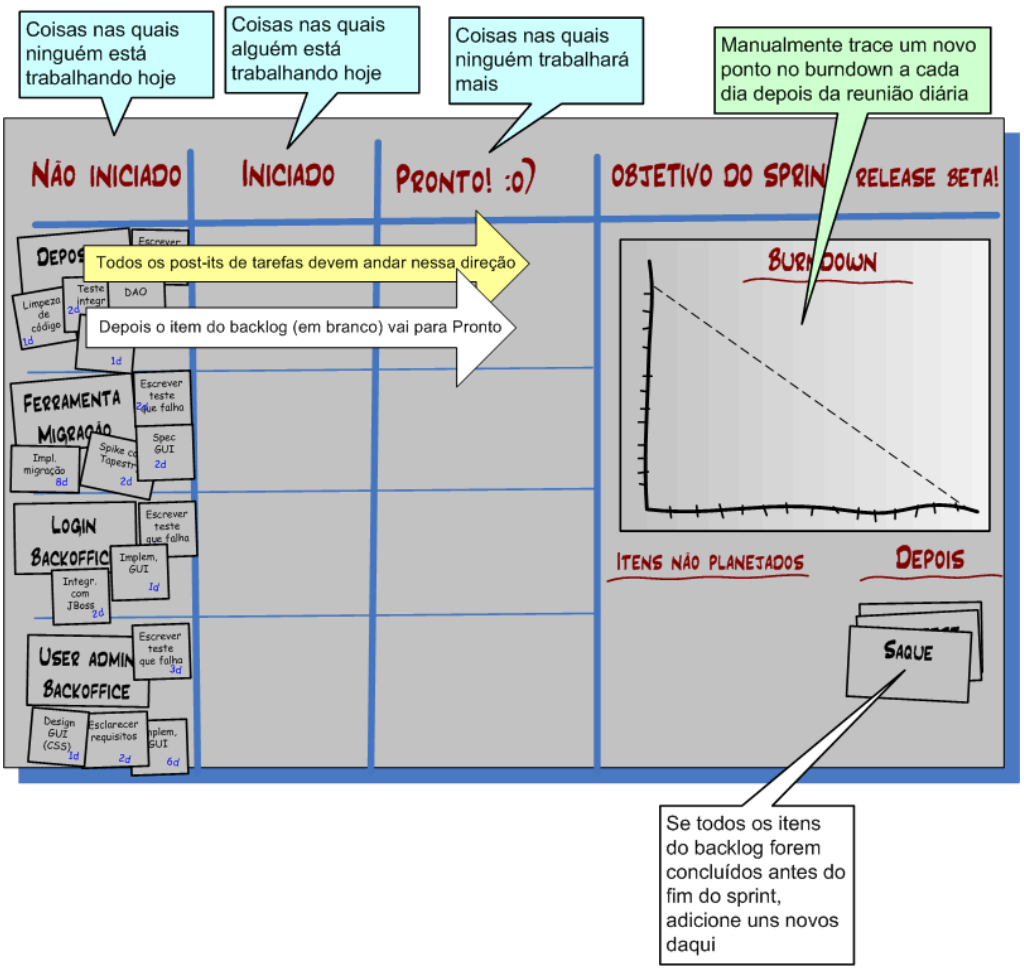

Figura 4. Explicação do *Taskboard* (retirada do livro Scrum e XP direto das Trincheiras)

A figura 5 mostra como poderia ficar o *taskboard* após a primeira reunião diária. No exemplo, assumimos que foram designadas três tarefas para a equipe, serão nessas tarefas que a equipe trabalhará. Estes tarefas foram transferidas para a coluna de tarefas iniciadas, pelo menos até a próxima reunião, onde deverá ser feita uma análise para verificar os estados das tarefas em andamento e designar, ou não, novas tarefas para a equipe.

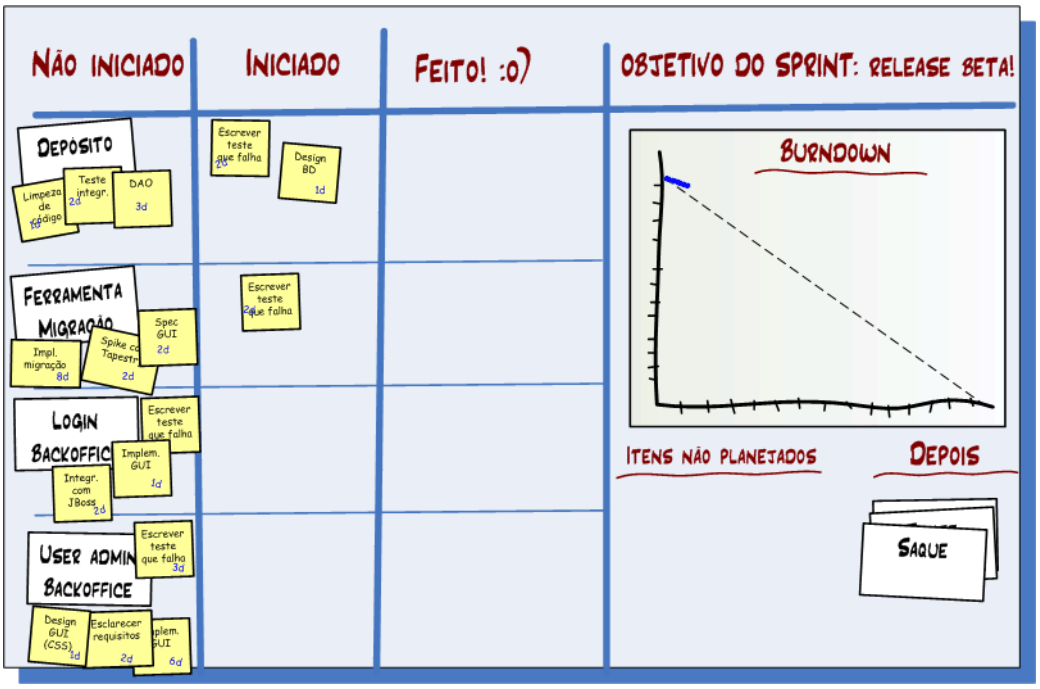

Figura 5. *Taskboard* após a primeira reunião diária (retirada do livro Scrum e XP direto das Trincheiras)

A figura 6 mostra um possível estado para o *taskboard* após algumas reuniões diárias. Podemos ver todas as tarefas da estória Depósito concluídas, algumas tarefas iniciadas, ou seja, tarefas em que a equipe está trabalhando no momento, e algumas tarefas ainda na coluna das tarefas não iniciadas, que deverão ser desenvolvidas até o final da *sprint*.

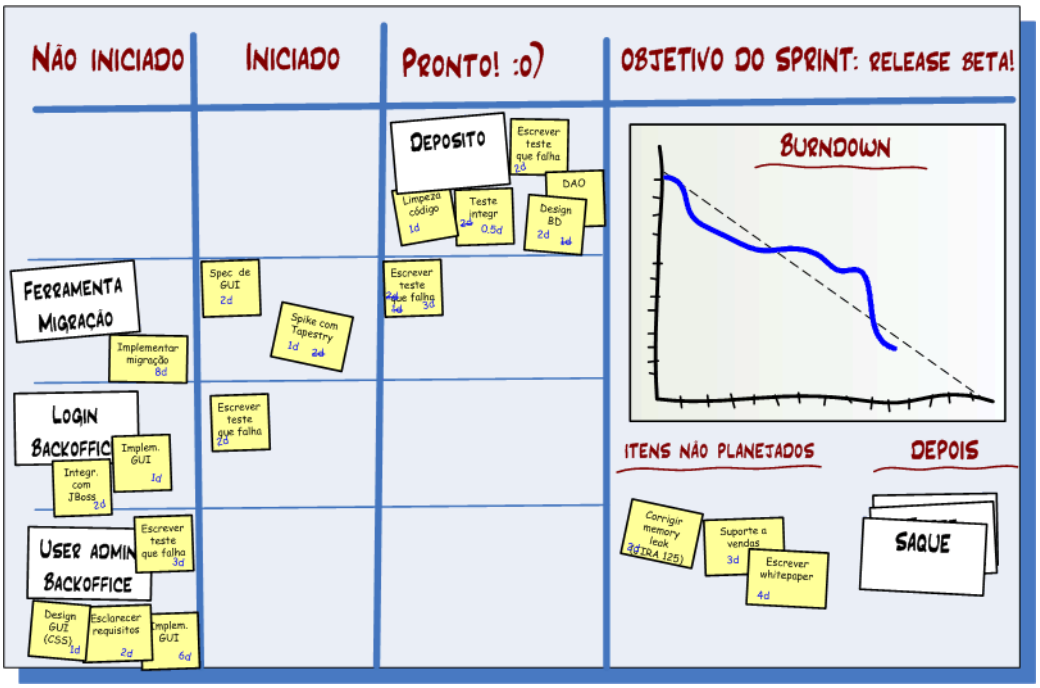

Figura 6. *Taskboard* após algumas reuniões (retirada do livro Scrum e XP direto das Trincheiras)

Dessa forma foi mostrado como o *taskboard* é usado como um artefato central para gerenciar o projeto usando SCRUM.

2.2Touch Screen

Como o foco deste projeto é o desenvolvimento de um *software* voltado para o uso em dispositivos *multi touch*, nesta seção abordaremos as idéias básicas dessa tecnologia.

O mercado de telas sensíveis ao toque, ou *touch screen*, é bastante novo, porém já é grande. Pesquisas apontam que as vendas de telas interativas cresceram 19,5% em 2008 (iSuppli's report, 2009). Educação e salas de reuniões são os setores em que essas telas são mais vendidas, somando cerca de 85% das vendas (Ooh-tv, 2010).

Uma tela com a tecnologia *touch screen* é capaz de detectar a presença e o local de um toque na sua superfície. Uma tela *touch screen* pode detectar desde toques com o dedo, com a mão, uma *stylus*, espécie de caneta especial para esses dispositivos ou até objetos maiores como celulares e câmeras fotográficas (www.microsoft.com/surface).

Uma das grandes vantagens da tecnologia *touch screen* é a possibilidade de interagir diretamente com o que está sendo mostrado na tela. A interação ocorre sem o intermédio de um *mouse* ou qualquer periférico. Um computador, por exemplo, pode ser utilizado sem a necessidade de *mouse* ou teclado.

Já as telas com tecnologia *multi touch* tem a capacidade de detectar múltiplos toques na sua superfície, dando maior liberdade ao usuário de interagir com o dispositivo.

Telas sensíveis ao toque são utilizadas em diversos dispositivos, desde grandes paredes interativas (Figura 7), mesas sensíveis ao toque (Figura 8), até pequenos dispositivos móveis como celulares e receptores GPS (Figura 9).

Figura 7. *Touch Wall*, tela interativa vertical

Figura 8. *Touch Table*, tela interativa horizontal

Figura 9. *Smartphone*, celular com tela sensível ao toque

2.2.1 Multi Touch

A diferença entre telas *touch screen* e telas *multi touch* é que as telas com tecnologia *multi touch* reconhecem múltiplos toques na tela ao mesmo tempo. Na prática, toda tela *multi touch* é uma tela *touch screen*, mas nem toda tela *touch screen* é *multi touch* (Como funcionam as telas sensíveis ao toque, 2009).

A maioria dos sistemas voltados para uso em dispositivos *touch screen* detecta somente um toque de cada vez, já que grande parte das telas *touch screen* atualmente reconhecem apenas um toque. Essas telas estão, em sua maioria, presentes em dispositivos portáteis, como celulares, tocadores de músicas e receptores GPS (Como funcionam as telas sensíveis ao toque, 2009).

3 Estado da Arte

O suporte para o *taskboard* existe de diferentes formas, como *software* específico, como planilha eletrônica ou, na sua forma mais dinâmica, como um quadro na parede. O quadro físico na parede tem sido uma das formas mais utilizadas do *taskboard*, e é também uma das mais indicadas (Scrum e XP direto das Trincheiras, 2007). Isso porque o quadro físico mantém a dinâmica do SCRUM, um dos princípios essenciais do SCRUM.

Nesse trabalho estamos buscando um suporte por meio de um sistema de *software* que substitua o *taskboard* físico, porém mantendo a praticidade do quadro e agregando algumas vantagens, como o acesso fácil ao histórico do projeto.

Features importantes de ferramentas de *software* para suportar a gerência do *taskboard* são:

- Oferecer um modo de visualização do *taskboard*;
- Suportar a inclusão, modificação, remoção de itens (estórias e tarefas);
- Suportar o movimento de um item de uma coluna a outra.

Além dessas características, procuramos um *software* que ofereça suporte à edição dos itens por meio de um teclado virtual para eliminar a necessidade de um teclado físico no momento da reunião diária.

Conforme o foco deste trabalho, o estado da arte revisa quais ferramentas de *software* já existem para suportar a gerência de *taskboards* seguindo SCRUM.

Dentro deste contexto realizamos uma busca no Google no dia 28 de junho com as seguintes palavras-chave “SCRUM *touch screen taskboard*” procurando identificar ferramentas relacionadas. No total foram retornados 3.570 resultados da busca. Revisando os primeiros 100 resultados da busca e analisando o resumo dos resultados identificamos 4 ferramentas relacionadas.

Incluímos como ferramentas relevantes somente ferramentas explicitamente voltadas à gerência de *taskboard* seguindo o método SCRUM, incluindo sistemas *web*, *multi touch* e *desktop*. Não consideramos neste momento ferramentas para suportar SCRUM em geral, sem foco especificamente no *taskboard*.

Como resultado, identificamos então quatro ferramentas relacionadas:

- Scrum Dashboard – Access It
- ScrumWall – Dot Net Solutions
- Scrum Dashboard – EPiSERVER
- Virtual SCRUM Board – MicroGuru Corporation

Scrum Dashboard – Access It

A ferramenta Scrum Dashboard desenvolvida pelo grupo Access It é uma aplicação *multi touch* que utiliza um *software* servidor já existente de onde obtém as tarefas e monta o *taskboard* digital. Essa ferramenta é apenas um complemento para um sistema existente, o Team Foundation Server (http://msdn.microsoft.com/en-us/library/ms181232.aspx), dando a ele a funcionalidade de mostrar as tarefas na forma de um *taskboard*. Esta ferramenta ainda está em fase de testes.

Figura 10. Screenshot da ferramenta Scrum Dashboard – Access It

Scrum Wall – Dot Net Solutions

O sistema Scrum Wall da empresa Dot Net Solutions foi desenvolvida para ser uma ferramenta de baixo custo de implantação para auxiliar as equipes que tem um ou mais membros longe do restante da equipe. Sendo uma aplicação *web*, todos os integrantes da equipe podem acompanhar o *taskboard* pela internet.

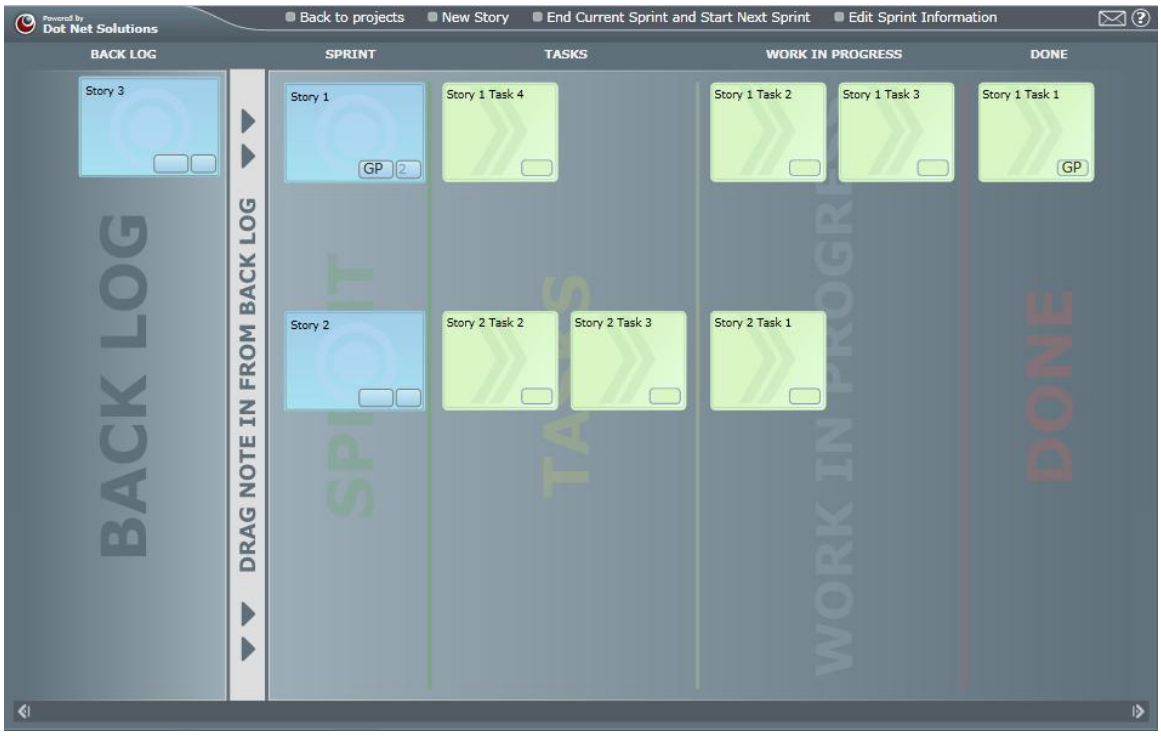

Figura 11. Screenshot da ferramenta Scrum Wall – Dot Net Solutions

Scrum Dashboard – EPiSERVER

Da mesma forma que a ferramenta do grupo Access It, a ferramenta desenvolvida pela empresa EPiSERVER é também apenas um complemento para o sistema Team Foundation Server. Com a diferença de ser uma aplicação *web* e mais completa, porém não conta com a tecnologia *multi touch*.

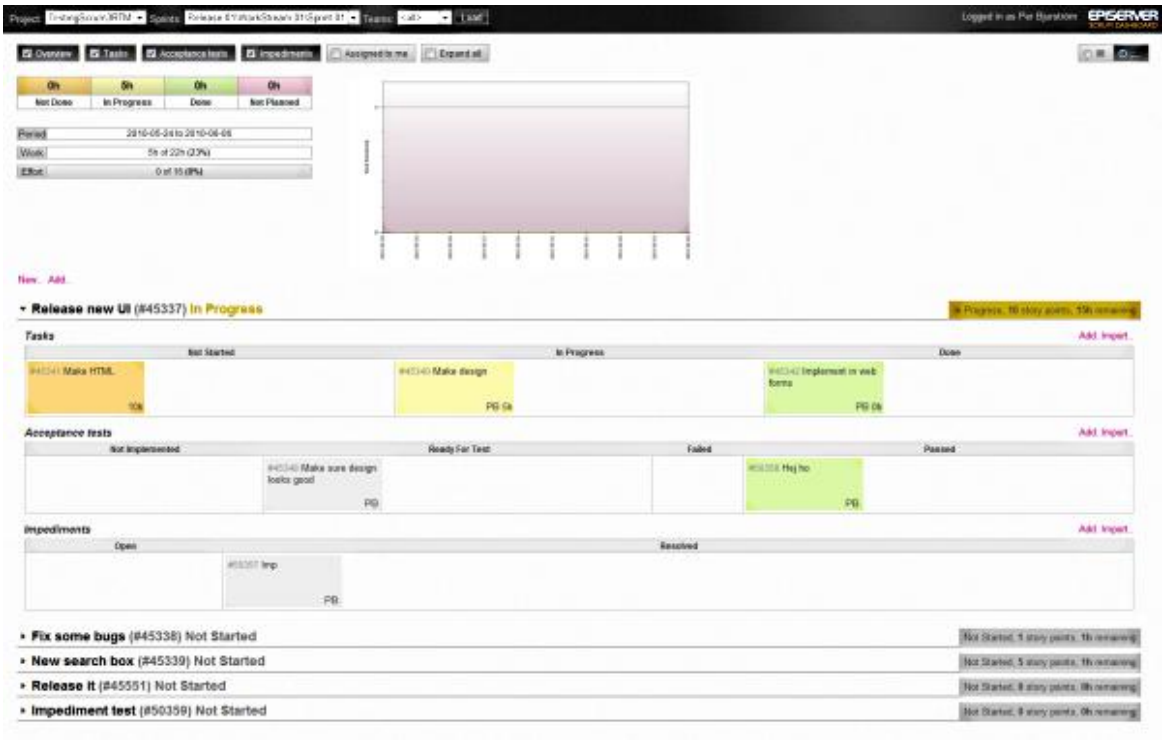

Figura 12. Screenshot da ferramenta Scrum Dashboard – EPiSERVER

Virtual SCRUM Board – MicroGuru Corporation

A ferramenta Virtual SCRUM Board é um *software* completo para ajudar equipes de desenvolvimento a gerenciar o *taskboard*. Os dados da ferramenta podem ser compartilhados com todos os membros da equipe por meio da rede de computadores.

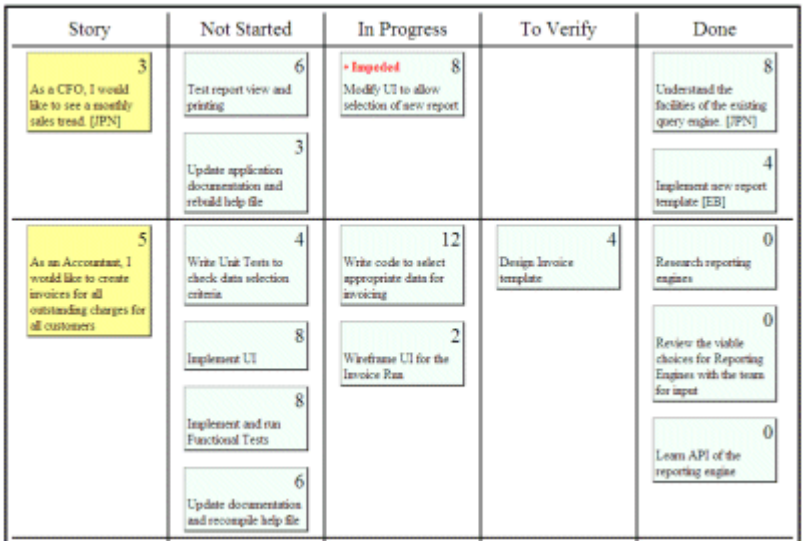

Figura 13. Screenshot da ferramenta Virtual SCRUM Board – MicroGuru Corporation

3.1 Discussão

Em geral, podemos observar que mesmo existindo diversas ferramentas para suportar a aplicação do SCRUM na prática (como por exemplo, FireScrum (www.firescrum.com), Pronto! (pronto.bluesoft.com.br) e ScrumWorks (www.danube.com/scrumworks)), poucos tem um foco especificamente na gerência do *taskboard*. Dentre estes só identificamos neste momento uma única ferramenta com interface *multi touch* para gerenciar o *taskboard*, o Scrum Dashboard da Access IT.

Comparando as ferramentas que foram identificadas voltadas para a gerência do *taskboard* (Tabela 1), podemos observar que nenhuma atende a todos os requisitos. Algumas ferramentas atendem à maioria deles, mas deixam a desejar no requisito da interface *multi touch*. Quanto aos requisitos básicos, como a visualização do *taskboard* e movimentação dos itens entre as colunas, todas as ferramentas atendem bem.

Observando que a única ferramenta identificada com interface *multi touch* ainda está em fase de desenvolvimento enfatiza o caráter inovador desta intenção de pesquisa.

Ferramenta	**Scrum Dashboard**	**Scrum Wall**	**Scrum Dashboard**	**Virual SCRUM Board**
Empresa	Access It	Dot Net Solutions	EPiSERVER	MicroGuru Corporation
Web site	labs.access-it.net/en-US/Projects/TFS/ScrumDashboard	www.dotnetsolutions.co.uk/about-us/scrumwall	scrumdashboard.codeplex.com	www.virtualscrumboard.com
Licença	Informação não disponível	*Freeware*	*Opensource*	*Shareware*
Interface	*Multi touch*	*Web*	*Web*	*Desktop*
Língua	Inglês	Inglês	Inglês	Inglês
Estágio de desenvolvimento	Em fase de testes internos	Sendo distribuído	Sendo distribuído	Sendo distribuído
Ferramenta completa/módulo adicional	Módulo adicional	Ferramenta Completa	Módulo adicional	Ferramenta Completa
Features				
Visualizar o *taskboard*	Sim	Sim	Sim	Sim
Suportar a edição de itens	Não	Sim	Sim	Sim
Movimentar itens	Sim	Sim	Sim	Sim
Multi touch	Sim	Não	Não	Não
Teclado virtual	Não	Não	Não	Não

Tabela 1. Comparação das ferramentas identificadas

Com base nestes resultados, o presente trabalho visa melhorar este estado da arte com o desenvolvimento de uma ferramenta *multi touch* que suporta todas as *features* identificadas.

4 Solução

Neste capítulo é apresentada uma possível solução para a questão apresentada anteriormente.

4.1 Concepção

Uma proposta de solução para o problema é desenvolver um *software multi touch* para auxiliar na gerência de *taskboards* como elemento principal da gerência de projetos baseada no método SCRUM.

O *software* é um aplicativo para gerenciar o *taskboard*, a ser utilizado durante as reuniões do SCRUM. Esse aplicativo é desenvolvido para uso em telas *multi touch*, ficando mais fácil o manuseio durante as reuniões.

Este sistema poderá ser utilizado em *smartboards* e desta forma substituindo a forma em papel do *taskboard*, atualmente mais utilizada. Pode ser também utilizado em outros dispositivos *multi touch* como, por exemplo, mesas *multi touch* ou computadores *multi touch*. Atualmente o sistema está sendo desenvolvido para computadores *multi touch* devido a restrições de recursos. Assim pretende-se prototipar o sistema para aprovar o conceito, e futuramente adaptá-lo para o uso em *smartboards*.

O uso de um *software* para gerenciar o *taskboard* substituindo a sua versão em papel traz várias vantagens como, por exemplo, a possibilidade de utilização de uma base de dados já existente e o registro digital de cada reunião diária.

Para manter a conformidade com o formato atual de um *taskboard*, que já se mostrou adequado na prática do SCRUM, a idéia da concepção do *software* é simular um taskboard no formato atualmente adotado.

O sistema tem características como visualização do *taskboard* na forma de uma tabela, onde as colunas dividem as tarefas do *backlog*, em andamento e tarefas prontas naquela *sprint*, e as linhas dividem as *stories* (Figura 14).

Figura 14. Desenho da interface do sistema

As tarefas apresentadas no *taskboard* então podem ser movimentadas através do toque na tela do dispositivo. Além disso, será incluído no sistema o suporte à edição de tarefas (nome, duração, membro da equipe a quem a tarefa foi designada e anotações) via teclado virtual.

Dessa forma, buscamos desenvolver uma ferramenta inovadora e que poderá ser usada para substituir o taskboard físico na gerência de projetos utilizando SCRUM.

4.2 Requisitos

Analisando as necessidade e características em relação ao uso de um *taskboard* seguindo a metodologia SCRUM (vide seção 2.1.1), identificamos os seguintes requisitos funcionais e não-funcionais.

1. Requisitos funcionais:

   O sistema de software permitirá:

   1.1. Visualizar o *taskboard* na forma de uma tabela com 3 colunas: *backlog*, *in progress*, *done*.

   1.2. Incluir, modificar e remover elementos do *taskboard* (estórias e tarefas). As estórias são descritas por um nome e as tarefas por um nome, descrição, duração e membro a qual foi designada.

1.3. Movimentar um item de uma coluna a outra no modo de visualização do *taskboard*.

2. Requisitos não funcionais:
   2.1. A interface do sistema de software precisa ser adequada para dispositivos *multi touch*.
   2.2. O sistema de software precisa ser independente de plataforma.
   2.3. O teclado virtual deve seguir o padrão de teclado brasileiro ABNT-2.
   2.4. Fácil de adaptar para um *smartboard*.
   2.5. Usabilidade:
      2.5.1. Os usuários devem conseguir usar o sistema sem treinamento algum.
      2.5.2. Qualquer tarefa deve ser concluída com sucesso em 95% dos casos.
      2.5.3. As tarefas de adição e remoção de itens não devem levar mais de 5 segundos.
      2.5.4. A edição de itens não deve levar mais de trinta segundos para serem realizadas.
      2.5.5. A tarefa de movimentação dos itens não deve levar mais de 5 segundos.
   2.6. Desempenho:
      2.6.1. O sistema deve responder em menos de 1 segundo aos toques do usuário.
      2.6.2. A movimentação das tarefas no *taskboard* deve ser suave e rápida, acompanhando o gesto do usuário.

Precondições ao uso do sistema:

1.1. Os usuários devem ter conhecimento sobre o método SCRUM.

4.3 Casos de Uso

Analisando os requisitos identificamos os casos de uso, mostrados no diagrama da Figura 15 e detalhados em seguida.

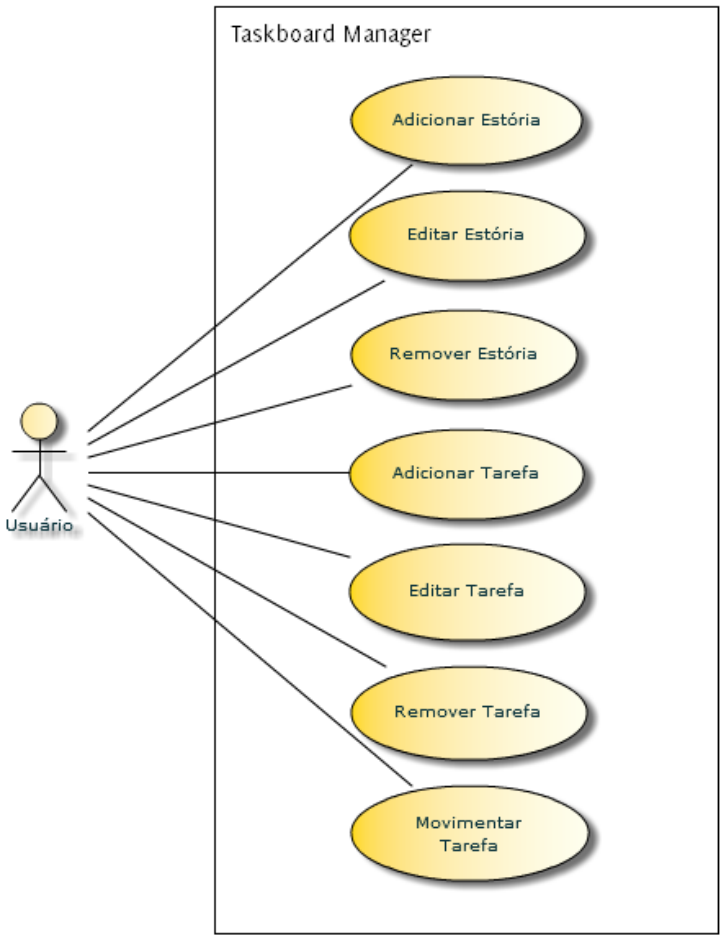

Figura 15. Diagrama de casos de uso

Caso de uso: Adicionar uma estória

Ator primário: Usuário

Fluxo normal:

1. O usuário toca o botão de adicionar uma estória;
2. O sistema adiciona a nova estória no *taskboard*;
3. O sistema exibe o painel de edição da estória recém criada.

Caso de uso: Editar uma estória

Ator primário: Usuário

Fluxo normal:

1. O usuário toca a estória que deseja editar;
2. O sistema exibe o botão editar estória;

3. O usuário toca o botão editar estória;
4. O sistema exibe o painel de edição de estória;
5. O usuário pode alterar o nome da estória, ou as tarefas da estória (inserir, editar ou excluir);
6. O usuário toca no botão Ok;
7. O sistema executa as alterações no *taskboard*;
8. O sistema fecha o painel de edição e volta para o modo de visualização do *taskboard*.

Caso de uso: Remover uma estória

Ator primário: Usuário

Fluxo normal:

1. O usuário toca a estória que deseja remover;
2. O sistema exibe o botão remover estória;
3. O usuário toca o botão remover estória;
4. O sistema pede a confirmação do usuário para realizar a remoção;
5. O usuário confirma a remoção da estória;
6. O sistema remove a estória do *taskboard*.

Fluxo alternativo:

5a. O usuário não confirma a remoção da estória;
    1. O caso de uso é cancelado.

Caso de uso: Adicionar uma tarefa

Ator primário: Usuário

Fluxo normal:

1. O usuário toca o botão adicionar tarefa na estória desejada;
2. O sistema adiciona uma tarefa na base de dados da estória selecionada;
3. O sistema exibe o painel de edição da tarefa recém criada.

Caso de uso: Editar uma tarefa

Ator primário: Usuário

Fluxo normal:

1. O usuário seleciona a tarefa que deseja editar;

2. O sistema mostra o botão editar tarefa;
3. O usuário toca o botão editar tarefa
4. O sistema exibe o painel de edição de tarefa;
5. O usuário altera os dados que deseja (nome, descrição, duração e membro responsável);
6. O usuário toca o botão Ok;
7. O sistema executa as alterações na tarefa;
8. O sistema fecha o painel de edição e volta para o modo de visualização do *taskboard*.

Fluxo alternativo:

4a. O usuário cancela as alterações
   1. O caso de uso é cancelado.

Caso de uso: Remover uma tarefa

Ator primário: Usuário

Fluxo normal:

1. O usuário toca a tarefa que deseja remover;
2. O sistema mostra o botão remover tarefa;
3. O usuário toca o botão remover tarefa;
4. O sistema pede a confirmação do usuário para remover a tarefa;
5. O usuário confirma a remoção;
6. O sistema remove a tarefa do *taskboard*.

Fluxo alternativo:

5a. O usuário não confirma a remoção da estória
   1. O caso de uso é cancelado.

Caso de uso: Movimentar uma tarefa

Ator primário: Usuário

Fluxo normal:

1. O usuário seleciona a tarefa que deseja movimentar;
2. O usuário movimenta a tarefa para a coluna que desejar;
3. O sistema executa a mudança no *taskboard*.

4.4 Arquitetura do Sistema

O sistema é desenvolvido com base no padrão *Model View Controller* (MVC), que visa separar o domínio lógico do sistema da interface de usuário. Permitindo o desenvolvimento independente das duas partes e possibilitando mudanças na interface sem interferir na lógica da aplicação.

A figura 16 mostra o diagrama de classes do domínio lógico da aplicação.

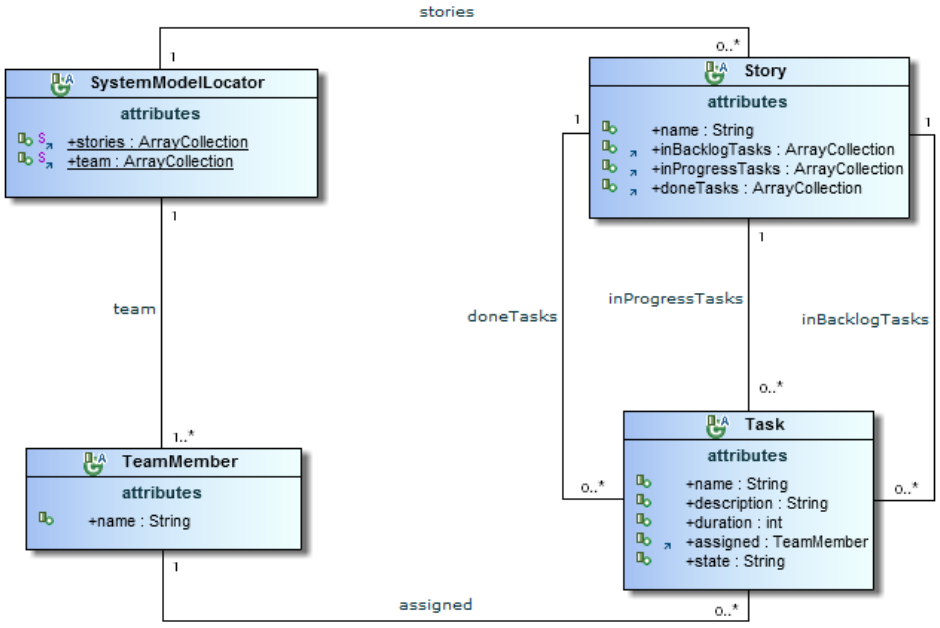

Figura 16. Diagrama de classes conceitual

4.5 Design da interface do Sistema

A concepção do design da interface do sistema é baseada no *taskboard* de papel, para torná-lo mais intuitivo para os usuários que já estão familiarizados com a versão em papel do *taskboard*. Porém precisamos adaptar o modelo em papel para utilizar melhor o espaço da tela (Figura 17), desenvolvendo este protótipo para uma tela de 22''.

A seguir são mostradas algumas *screenshots* do sistema em funcionamento e os casos de uso que elas representam.

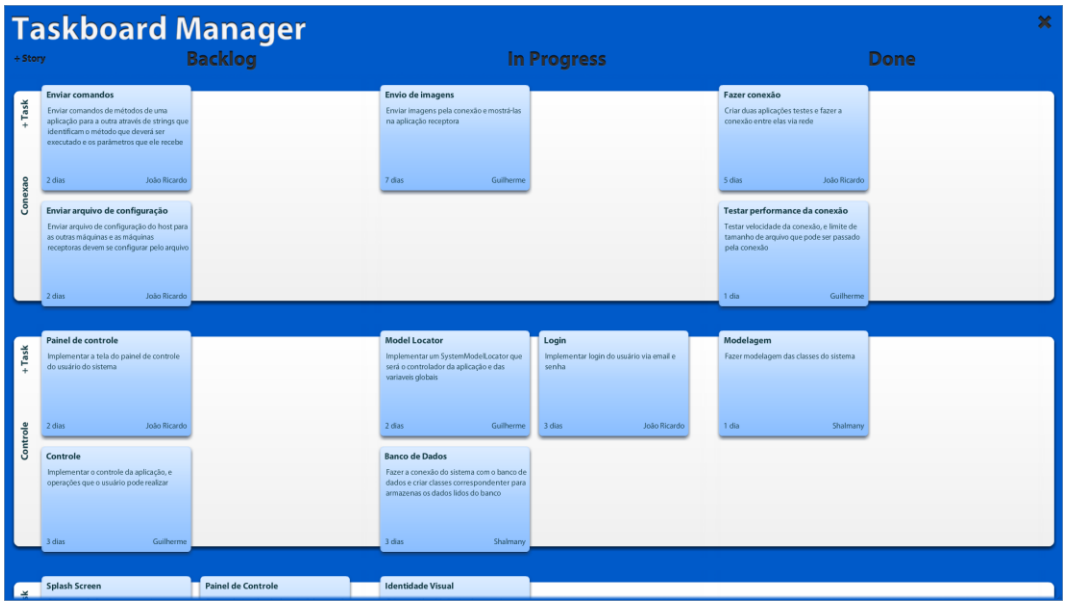

Figura 17. Modo de visualização do *taskboard*

Caso de uso: Adicionar uma estória

Ao tocar sobre o botão adicionar estória (Figura 18) o sistema adiciona uma estória no *taskboard* e automaticamente mostra o painel de edição de estórias (Figura 20).

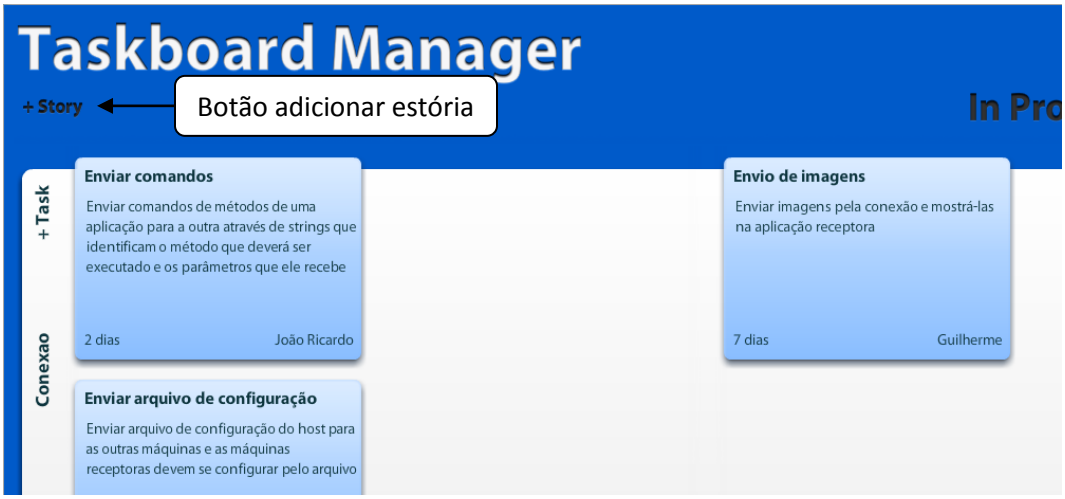

Figura 18. Botão adicionar estória

Caso de uso: Editar uma estória

Para editar uma estória basta tocar sobre o nome da estória que o sistema mostra o botão de edição de estória (Figura 19). Esse botão fica normalmente oculto para deixar a interface do sistema mais limpa. Ao tocar o botão editar estória o sistema

mostra o painel de edição de estória (Figura 20). Após o usuário alterar os dados da estória (nome e tarefas), basta tocar o botão confirmar as alterações.

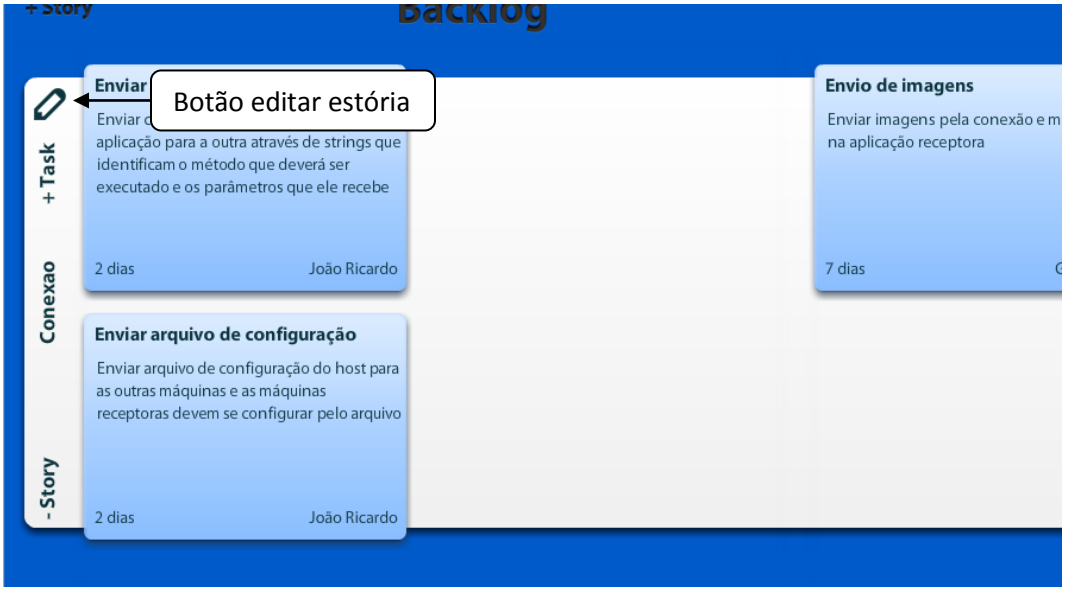

Figure 19. Botão editar estória

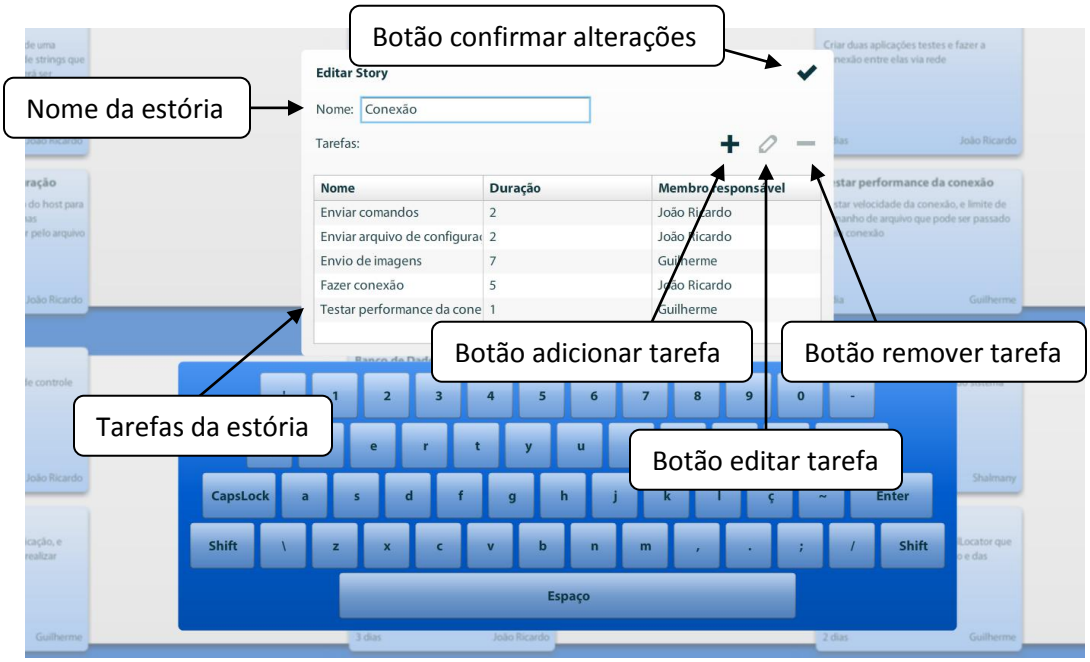

Figura 20. Painel de edição de estórias

Caso de uso: Remover uma estória

Para remover uma estória basta tocar sobre o nome da estória que o sistema mostra o botão remover estória (Figura 21). Assim como o botão editar estória, o botão

remover estória também fica normalmente oculto. Ao tocar sobre o botão de remoção de estória o sistema pede uma confirmação do usuário antes de executar a ação.

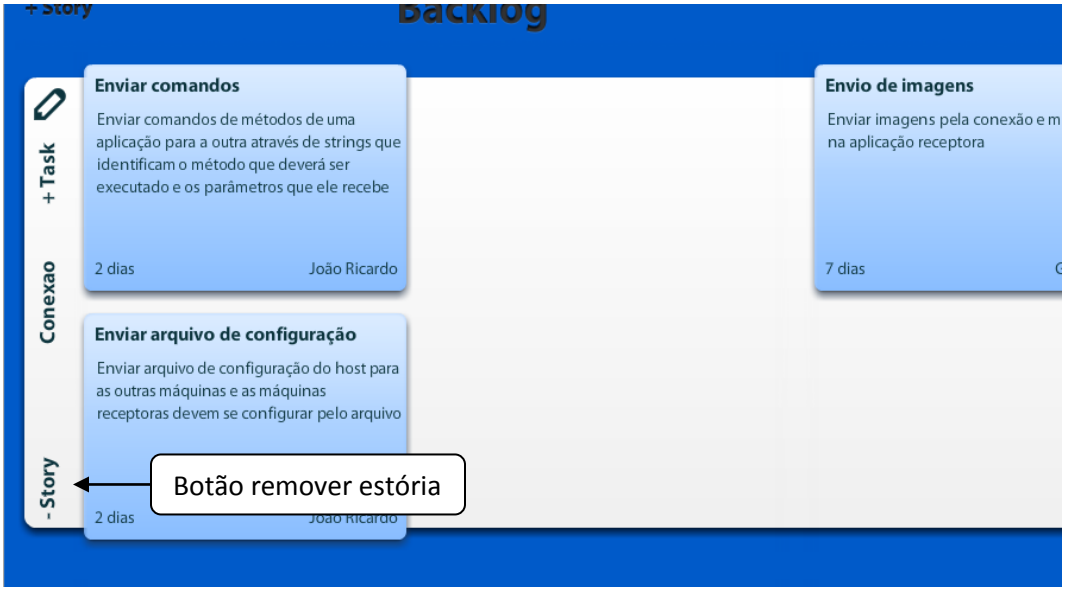

Figure 21. Botão remover estória

Caso de uso: Adicionar uma tarefa

Adicionar tarefas é uma função bastante utilizada, por isso o botão de adição de tarefas fica sempre visível (Figura 22). Ao tocar sobre ele o sistema mostra o painel de edição de tarefas (Figura 24).

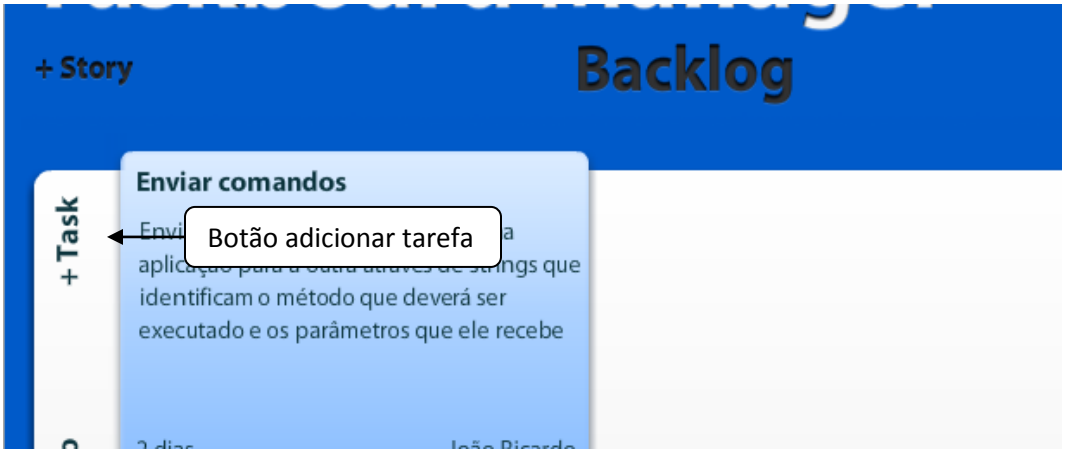

Figura 22. Botão adicionar tarefa

Caso de uso: Editar uma tarefa

Para editar uma tarefa já existente no *taskboard* basta tocar sobre a tarefa escolhida que o sistema mostra o botão de edição de tarefa (Figura 23). Assim como os

botões de edição e remoção de estória, os botões de edição e remoção de tarefa também ficam ocultos normalmente. Ao tocar no botão editar tarefa o sistema mostra o painel de edição de tarefa (Figura 24). Após adicionar os dados da tarefa (nome, descrição, duração e membro designado) o usuário pode ou não salvar as alterações.

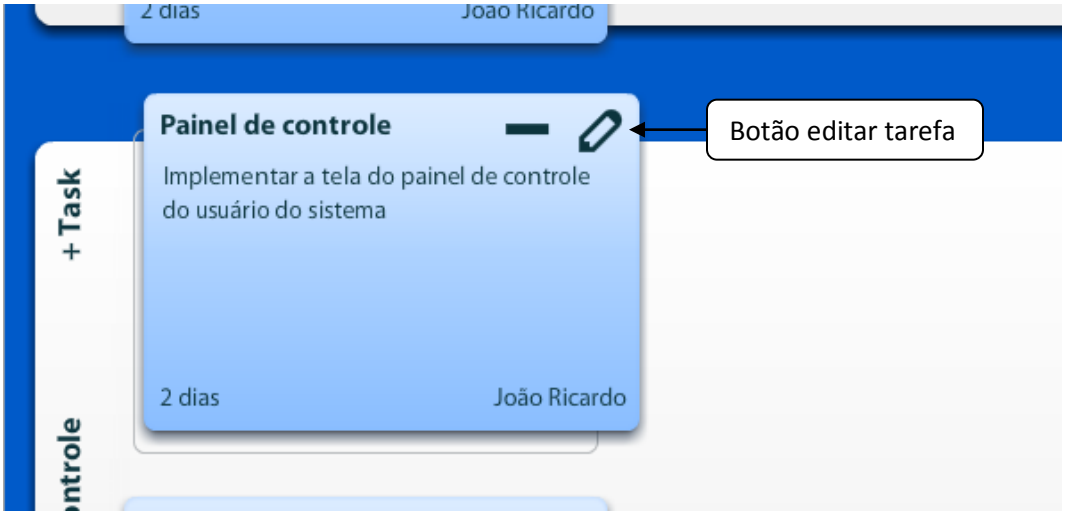

Figura 23. Botão editar tarefa

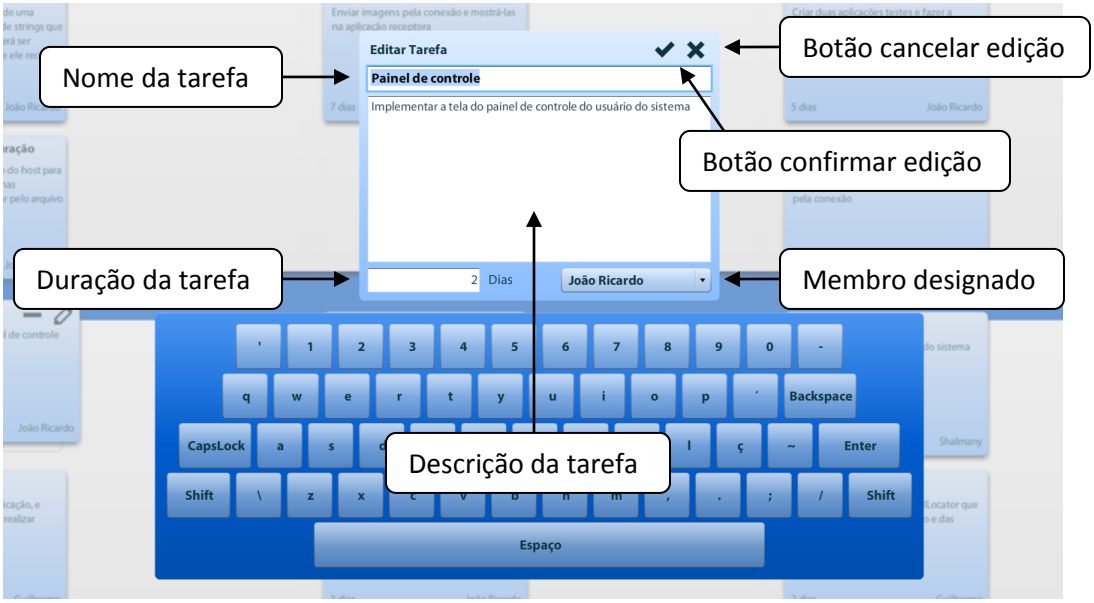

Figura 24. Painel de edição de tarefa

Caso de uso: Remover uma tarefa

Para remover uma tarefa já existente no *taskboard* basta clicar sobre a tarefa escolhida que o sistema mostra o botão de remoção de tarefa (Figura 25). Ao clicar

sobre o botão de remoção de tarefa o sistema pede uma confirmação do usuário antes de executar a ação.

Figura 25. Botão editar tarefa

Caso de uso: Movimentar uma tarefa

Para movimentar uma tarefa de uma coluna para outra basta clicar sobre a tarefa que o sistema “destaca” a tarefa do *taskboard*, assim o usuário pode movê-la para a coluna desejada (Figura 26).

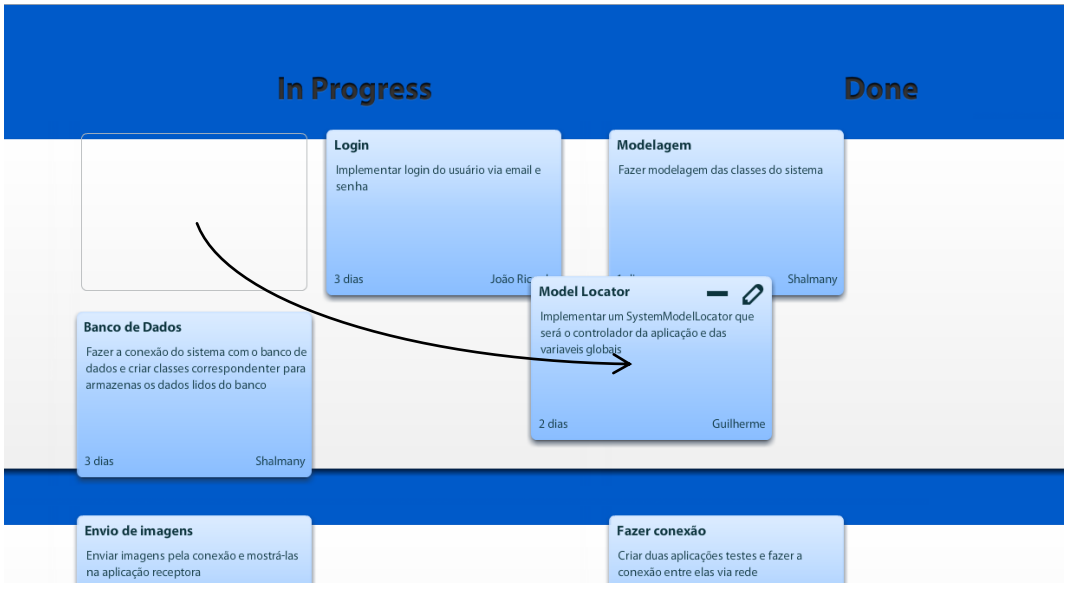

Figura 26. Movendo a tarefa de uma coluna para outra

4.6 Implementação do Sistema

Para implementar o sistema proposto foi utilizada a plataforma Adobe AIR (http://www.adobe.com/products/air). Essa plataforma dá suporte à tecnologia *multi touch* nativamente e de forma simples, além de ser multiplataforma, atendendo a um dos requisitos não-funcionais do sistema. Para o desenvolvimento foi utilizada a ferramenta Flash Builder 4 [1] (http://www.adobe.com/products/flashbuilder), uma ferramenta, também da Adobe, para desenvolvimento baseada na IDE Eclipse.

Para persistência dos dados no computador não foi utilizado nenhum banco de dados específico. Como o sistema é pequeno e não demanda muitos recursos foi utilizada a persistência em arquivo. Os dados do projeto são armazenados em um único arquivo na máquina e lidos quando o sistema inicia. Sempre que o *software* é encerrado ele salva automaticamente as alterações no arquivo de dados.

O código fonte do sistema pode ser encontrado no Apêndice D ou no CD anexado ao trabalho.

4.7 Testes do Sistema

A partir dos casos de uso identificados são definidos os testes do *software*. O planejamento dos testes e o resultado deles são mostrados na Tabela 2.

No.	Caso de uso	Dados de teste	Pré requisitos	Passos	Resultado esperado	Status
1	Adicionar Estória	Adicionar uma estória com nome “Minha Nova Estória”	Nenhum	Clicar no botão de adição de estória; Editar nome da estória; Confirmar a adição	Estória “Minha Nova Estória” adicionada no *taskboard*	Ok
2	Editar Estória Fluxo normal	Editar a estória do teste anterior e trocar o nome para “Minha Velha Estória” e confirmar as alterações	Estória “Minha Nova Estória” deve existir no *taskboard*	Clicar sobre o nome da estória; Clicar no botão de edição de estória; Mudar o nome da estória para “Minha Velha Estória”; Clicar no botão confirmar alterações	Nome da estória “Minha Nova Estória” mudado para “Minha Velha Estória”	Ok

[1] Foi obtida uma licença educacional para o uso da ferramenta para a realização do projeto

3	Remover Estória	Cancelar a remoção da estória “Minha Velha Estória”	Estória “Minha Velha Estória” deve existir no *taskboard*	Clicar sobre o nome da estória; Clicar no botão de remoção de estória; Cancelar remoção	Estória “Minha Velha Estória” continuar existindo no *taskboard*	Ok
4	Remover Estória	Remover estória “Minha Velha Estória”	Estória “Minha Velha Estória” deve existir no *taskboard*	Clicar sobre o nome da estória; Clicar no botão de remoção de estória; Confirmar remoção	Estória “Minha Velha Estória” removida do *taskboard*	Ok
5	Adicionar Tarefa	Adicionar tarefa com nome “Minha Nova Tarefa”, descrição “Descrição da minha nova tarefa”, duração 2 dias e atribuída ao “Membro 1” na estória “Minha Outra Estória”	Estória “Minha Outra Estória” deve existir no *taskboard*	Clicar no botão de adição de tarefa da estória “Minha Outra Estória”; Editar o nome, descrição, duração e membro da tarefa; Confirmar alterações	Tarefa com nome “Minha Nova Tarefa”, descrição “Descrição da minha nova tarefa”, duração 2 dias e atribuída ao “Membro 1” adicionada no *taskboard*	Ok
6	Editar Tarefa	Editar tarefa do teste anterior e trocar o nome para “Minha Velha Tarefa”, descrição para “Descrição da minha velha tarefa”, duração para 1 dia e membro para “Membro 2” e cancelar alterações	Estória “Minha Outra Estória” deve existir no *taskboard* e a tarefa “Minha Nova Tarefa” deve existir na estória “Minha Outra Estória”	Clicar sobre a tarefa; Clicar no botão de edição de tarefa; Editar os dados da tarefa; Clicar no botão cancelar edição	Tarefa “Minha Nova Tarefa” não alterada	Ok
6	Editar Tarefa	Editar tarefa do teste anterior e trocar o nome para “Minha Velha Tarefa”, descrição para “Descrição da minha velha tarefa”, duração para 1 dia e membro para “Membro 2”	Estória “Minha Outra Estória” deve existir no *taskboard* e a tarefa “Minha Nova Tarefa” deve existir na estória “Minha Outra Estória”	Clicar sobre a tarefa; Clicar no botão de edição de tarefa; Editar os dados da tarefa; Confirmar a alteração	Nome da “Minha Nova Tarefa” alterado para “Minha Velha Tarefa”, descrição para “Descrição da minha velha tarefa”, duração para 1 dia e membro para “Membro 2”	Ok
7	Remover Tarefa Fluxo alternativo	Remover a tarefa “Minha Velha Tarefa” da estória “Minha Outra Estória”	Estória “Minha Outra Estória” deve existir no *taskboard* e a tarefa “Minha Velha Tarefa” deve existir na estória “Minha Outra Estória”	Clicar sobre a tarefa; Clicar no botão de remoção de tarefa; Cancelar a remoção	Tarefa “Minha Velha Tarefa” continuar existindo no *taskboard*	Ok

8	Remover Tarefa Fluxo normal	Remover a tarefa “Minha Velha Tarefa” da estória “Minha Outra Estória”	Estória “Minha Outra Estória” deve existir no *taskboard* e a tarefa “Minha Velha Tarefa” deve existir na estória “Minha Outra Estória”	Clicar sobre a tarefa; Clicar no botão de remoção de tarefa; Confirmar a remoção	Tarefa “Minha Velha Tarefa” removida do *taskboard*	Ok
9	Movimentar tarefa	Movimentar a tarefa “Minha Outra Tarefa” da estória “Minha Outra Estória” para a coluna *done*	Estória “Minha Outra Estória” deve existir no *taskboard* e a tarefa “Minha Outra Tarefa” deve existir na estória “Minha Outra Estória”	Clicar sobre a tarefa; Arrastá-la até a coluna *done*	Tarefa “Minha Outra Tarefa” movida para a coluna *done*	Ok

Tabela 2. Resultados dos testes dos casos de uso

O planejamento dos testes e o resultado referente aos requisitos não-funcionais são mostrados na Tabela 3.

No.	**Requisito não-funcional**	**Observações do testador**
1	O teclado virtual deve seguir o padrão de teclado brasileiro ABNT-2	Ok
2	Os usuários devem conseguir usar o sistema sem treinamento algum	Confirmado na avaliação do sistema
3	Qualquer tarefa deve ser concluída com sucesso em 95% dos casos	Confirmado na avaliação do sistema
4	As tarefas de adição e remoção de itens não devem levar mais de 5 segundos	Durante a avaliação levou em média 3 segundos
5	A movimentação das tarefas no taskboard deve ser suave e rápida, acompanhando o gesto do usuário	Ok
6	A edição de itens não deve levar mais de trinta segundos para serem realizadas	Durante a avaliação levou em média 10 segundos
7	A tarefa de movimentação dos itens não deve levar mais de 5 segundos	Durante a avaliação levou em média 3 segundos
8	O sistema deve responder em menos de 1 segundo aos toques do usuário	Ok

Tabela 3. Resultados dos testes dos requisitos não-funcionais

5 Avaliação

Neste capítulo é apresentada uma aplicação do *software* desenvolvido e uma avaliação inicial para obter um *feedback* com relação à viabilidade do conceito proposto neste trabalho.

5.1 Definição

Com o objetivo de avaliar se o sistema desenvolvido é útil e pode auxiliar a gerência do *taskboard*, é realizada uma avaliação com base em uma aplicação simulando uma reunião diária no contexto de uma empresa de *software*.

A avaliação é realizada por um grupo de profissionais da área de desenvolvimento de *software*. O grupo é formado por um gerente de projeto e por membros da equipe de desenvolvimento do projeto de uma pequena empresa de *software* em Florianópolis/SC voltado para o desenvolvimento de *softwares multi touch*. A empresa atualmente adota o método SCRUM para gerenciar os seus projetos utilizando um *taskboard* de papel durante as reuniões diárias para monitorar o andamento dos projetos.

A avaliação é feita com base em uma reunião diária simulada utilizando dados de um projeto real da própria empresa. Para a execução da aplicação será utilizado um computador Positivo Touch 2500 rodando o sistema operacional Windows 7 tendo o sistema desenvolvido previamente instalado.

Após a realização da reunião diária simulada, os profissionais responderão um questionário (Apêndice B) sobre o sistema. O questionário consiste em perguntas de múltipla escolha sobre a usabilidade e utilidade do sistema, além de perguntas discursivas onde podem apontar vantagens, desvantagens, erros encontrados e sugerir melhorias.

5.1.1 Identificação das questões e métricas

O objetivo desta avaliação é analisar a viabilidade do conceito proposto neste trabalho principalmente em termos de utilidade e usabilidade. Para identificar as

medidas a serem levantadas para analisar esta questão está sendo adotado o método GQM - *Goal/Question/Metric* (Basili, 1994). GQM é um método de medição de *software*, primeiro identificando os objetivos da pesquisa.

Seguindo o método GQM são identificados os seguintes objetivos de avaliação:

Objetivo 1: Analisar se o sistema desenvolvido é útil para auxiliar na gerência do *taskboard* do SCRUM.

Objetivo 2: Analisar se o sistema é melhor que a versão em papel do *taskboard*.

A partir dos objetivos da avaliação, identificamos as perguntas que precisam ser respondidas para que os objetivos sejam alcançados, para que sejam respondidas as perguntas:

G1. O sistema desenvolvido é útil para auxiliar na gerência do *taskboard* do SCRUM?

Q1.1. Supre as necessidades da equipe com relação à gerência do *taskboard*?

M1.1.1. Oferece a possibilidade de edição dos itens?

M1.1.2. Suporta a movimentação dos itens?

M1.1.3. É possível fazer anotações nos itens?

Q1.2. A usabilidade do sistema é boa?

M1.2.1. Conseguiu realizar todas as tarefas típicas de uma reunião diária?

M1.2.2. As funções são intuitivas?

M1.2.3. O *software* pode substituir a versão em papel do *taskboard*?

G2. O sistema é melhor que a versão em papel do *taskboard*?

Q2.1. O sistema é mais vantajoso que a versão física?

M2.1.1. Quais são as vantagens do *software* com relação ao *taskboard* em papel?

M2.1.2. Quais as desvantagens do sistema quando comparado à versão em papel do *taskboard*?

Além das medidas identificadas, mostra-se importante perguntar aos participantes se foram encontrados erros no sistema e que sugestões eles teriam para melhorar o sistema, como *feedback* geral ao sistema proposto. Com base no refinamento dessas medidas é formulado um questionário (Apêndice B) que é respondido pelos participantes da avaliação.

5.2 Execução

A avaliação do sistema foi realizada no dia 19 de outubro de 2010 com a participação de um gerente de projetos e quatro membros da equipe de desenvolvimento. Foi simulada uma reunião diária com cerca de 10 minutos, sobre um projeto existente na empresa.

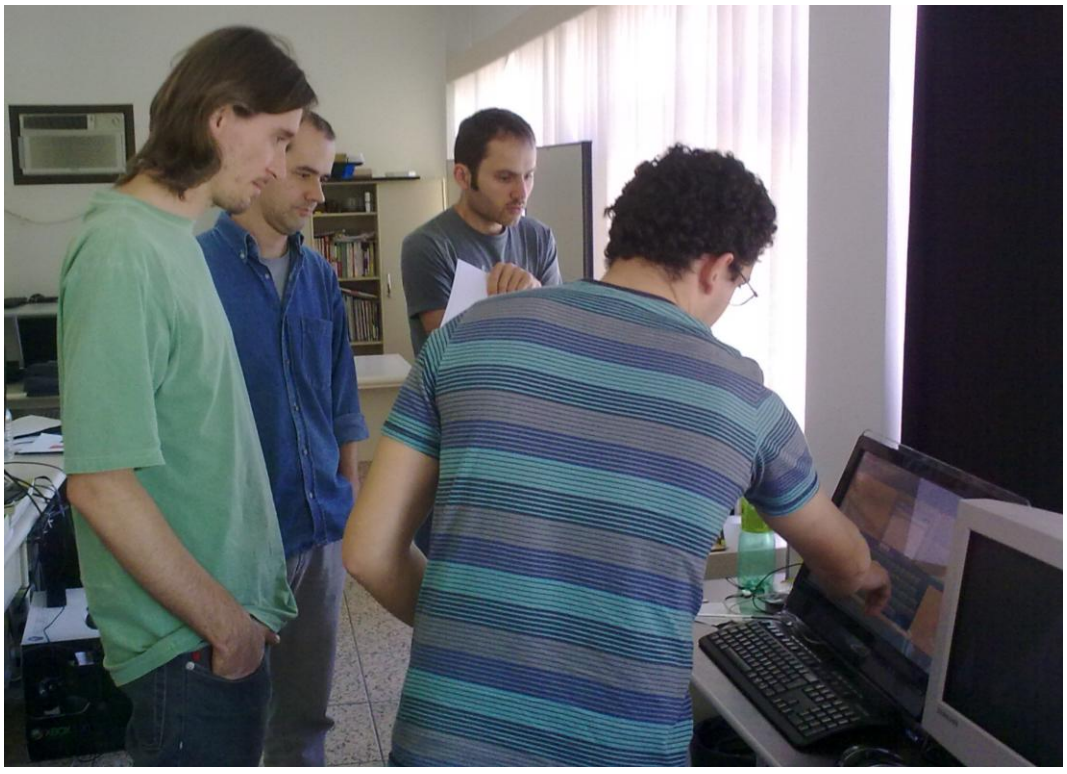

Figura 27. Reunião diária simulada

Ao fim da reunião foram coletados dados sobre a aplicação do sistema utilizando o questionário definido anteriormente (Apêndice B) e também foram recolhidos comentários sobre o sistema verbalmente após a simulação.

5.3 Análise

Os dados coletados são analisados seguindo a estrutura de objetivos/perguntas/métricas do GQM. O GQM determina que a ordem de interpretação das medidas deva ser *bottom-up*, facilitando a suposição dos objetivos através das respostas das perguntas. A compilação com os dados coletados pode ser encontrada no Apêndice C.

G1. O sistema desenvolvido é útil para auxiliar na gerência do *taskboard* do SCRUM?

Q1.1. Supre as necessidades da equipe com relação à gerência do *taskboard*?

M1.1.1. Oferece a possibilidade de edição dos itens?

M1.1.2. Suporta a movimentação dos itens?

M1.1.3. É possível fazer anotações nos itens?

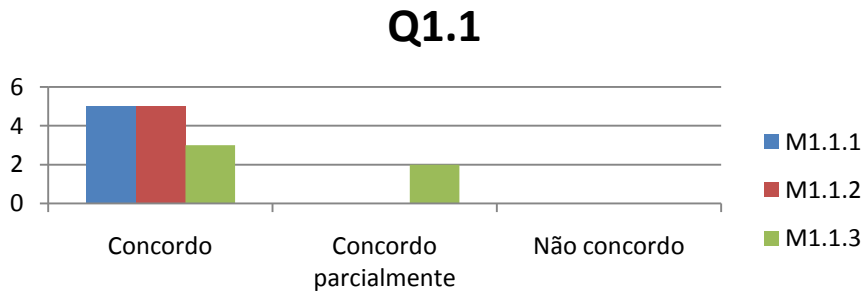

Figura 28. Gráfico do resultado das medidas M1.1.1, M1.1.2 e M1.1.3

Analisando o gráfico na figura 28 percebe-se que o *software* supre as necessidades da equipe em relação à gerência do *taskboard*, oferecendo suporte à edição e movimentação dos itens. Um ponto fraco é a falta da possibilidade de fazer anotações à mão livre nas tarefas.

Q1.2. A usabilidade do sistema é boa?

M1.2.1. Conseguiu realizar todas as tarefas típicas de uma reunião diária?

M1.2.2. As funções são intuitivas?

M1.2.3. O *software* pode substituir a versão em papel do *taskboard*?

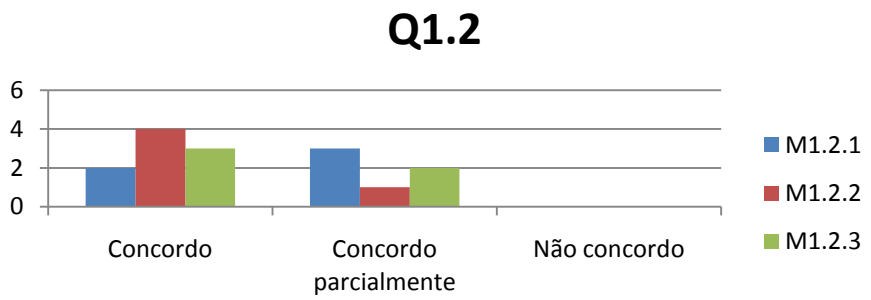

Figura 29. Gráfico do resultado das medidas M1.2.1, M1.2.2 e M1.2.3

A avaliação da usabilidade do sistema foi razoável. Analisando o gráfico na figura 29, podemos identificar que as funções do sistema não foram consideradas tão intuitivas pelos participantes da avaliação. Porém, observando a simulação notou-se que após 5 minutos de reunião as tarefas realizadas pelos integrantes da avaliação eram concluídas em tempo consideravelmente menor.

G2. O sistema é melhor que a versão em papel do *taskboard*?

Q2.1. O sistema é mais vantajoso que a versão física?

M2.1.1. Quais são as vantagens do *software* com relação ao *taskboard* em papel?

M2.1.2. Quais as desvantagens do sistema quando comparado à versão em papel do *taskboard*?

Analisando os questionários respondidos podemos perceber que as vantagens mais identificadas pelos participantes da avaliação foram a possibilidade de registro digital das reuniões (armazenamento seguro) e a economia de despesas da empresa (papel e fita adesiva).

Em contrapartida temos o investimento inicial da empresa com relação ao equipamento *multi touch* para ser utilizado nas reuniões. Outro ponto citado é que o *software* não suporta a inserção de imagens e desenhos no *taskboard*, o que na versão papel facilita para a equipe entender melhor a tarefa a ser feita.

P1. Encontrou algum erro no sistema? Se sim, cite-os.

Foi identificado em alguns poucos momentos a perda de sensibilidade ao toque, problema que pode ser causado por falha de *hardware* ou *software*, para identificar a causa do problema precisam ser realizados mais testes.

Foi identificado também que ao se destacar uma tarefa para realizar a movimentação, pode-se remover a estória que continha aquela tarefa, ficando a tarefa sem estória ainda no *taskboard*.

P2. Caso queira, sugira melhorias para sistema.

Foram sugeridas algumas melhorias na interface, para melhorar a usabilidade e tornar o software mais intuitivo. Uma delas é mostrar uma barra de rolagem lateral para dar *feedback* de onde o usuário está na lista de estórias. Outra sugestão recebida é diferenciar a coluna em que a tarefa está através de cores diferentes para cada coluna. O suporte a imagens e desenhos também foi citado.

Já o gerente de *software* que participou da reunião simulada, pensando mais longe, sugeriu funções novas ao sistema, como possibilidade de impressão e envio das tarefas por e-mail para a equipe do projeto.

5.4 Resultado

Em geral a aceitação do sistema foi boa, já que ele cumpre as necessidades da equipe com relação à gerência do *taskboard*, apesar de pecar um pouco na usabilidade. Isso não significa que o *software* pode substituir a versão física do *taskboard* em qualquer situação, mas é um primeiro indício de que esse é um bom caminho para a digitalização do *taskboard*. Entretanto, é importante considerar as ameaças à validade dos resultados da avaliação.

A avaliação foi uma reunião simulada apenas uma vez, em apenas uma empresa, com apenas uma equipe, em apenas um projeto. Dessa forma, a generalização dos resultados está extremamente limitada e outras avaliações em outros contextos precisam ser repetidas para poder generalizar os resultados.

Outra questão é a validade das medidas utilizadas na avaliação do sistema para analisar o objetivo da avaliação. Tentou-se amenizar essa ameaça pela adoção do

método GQM, que permite derivar de forma sistemática as medidas a serem analisadas.

Dentro deste contexto, a intenção da avaliação não é ser uma avaliação empírica da nova ferramenta, mas sim uma prova da viabilidade do conceito, a fim de saber se existe a possibilidade de um sistema desses um dia vir a substituir o *taskboard* físico.

6 Discussão

Comparando as ferramentas identificadas no capítulo 3 com a ferramenta proposta nesse trabalho, “Taskboard Manager”, podemos perceber que além da ferramenta proposta nenhuma outra atende a todos os requisitos identificados na seção 4.2, conforme mostrado na tabela 4.

Ferramenta	**Scrum Dashboard**	**Scrum Wall**	**Scrum Dashboard**	**Virual SCRUM Board**	**Taskboard Manager**
Empresa	Access It	Dot Net Solutions	EPiSERVER	MicroGuru Corporation	-
Features					
Visualizar o *taskboard*	Sim	Sim	Sim	Sim	Sim
Suportar a edição de itens	Não	Sim	Sim	Sim	Sim
Movimentar itens	Sim	Sim	Sim	Sim	Sim
Multi touch	Sim	Não	Não	Não	Sim
Teclado virtual	Não	Não	Não	Não	Sim

Tabela 4. Comparação das ferramentas identificadas com a ferramenta proposta

Como a especialidade de todas as ferramentas é a gerência do taskboard, já era esperado que todas possuíssem a possibilidade de visualizar o *taskboard*. Da mesma maneira, todas as ferramentas oferecem também a possibilidade de movimentar os itens de uma coluna para outra.

As ferramentas que já estão sendo distribuídas comercialmente também possuem a funcionalidade de editar os itens do taskboard, com exceção da ferramenta Scrum Dashboard da empresa Access It, ainda em fase de testes, que não suporta essa *feature*.

Não foi identificada nenhuma ferramenta comercial com suporte à tecnologia *multi touch*, a única ferramenta identificada com essa característica é a Scrum Dashboard da empresa Access It. Porém esta ferramenta ainda está em fase de testes. O teclado virtual não é oferecido por nenhuma das ferramentas identificadas além da ferramenta proposta nesse trabalho.

Para o fim proposto no trabalho, a ferramenta desenvolvida é a única que atende todos os requisitos identificados. Isso não significa que a ferramenta é melhor do que as existentes, mas considerando estes requisitos pode ser uma boa alternativa.

7 Conclusão

Neste trabalho foi analisado o contexto do tema proposto, abordando os princípios da gerência de projetos utilizando o método SCRUM dando ênfase à gerência do *taskboard*. Foi abordado também o tema de interfaces *multi touch*.

Foi analisado o estado da arte com relação às ferramentas existentes para gerência de *taskboards*. A partir dessa análise foi possível identificar e comparar as principais ferramentas existentes.

Com base na fundamentação teórica do trabalho e da análise do estado da arte foi desenvolvido um *software* para dispositivos com tecnologia *multi touch* para auxiliar na gerência do *taskboard* do SCRUM.

O sistema desenvolvido foi avaliado em uma empresa durante uma reunião diária simulada para obter uma primeira indicação da sua utilidade.

Com esse trabalho espera-se contribuir para a adoção e melhoria da aplicação de métodos ágeis de gerência de projetos. Mudar para um *taskboard* em formato digital ajudará principalmente a documentação e registro do *taskboard* e na comunicação das informações.

Para trabalhos futuros pretende-se evoluir a ferramenta desenvolvida com base nas sugestões de melhorias apontadas pelos participantes da avaliação do sistema, melhorando o sistema existente e adicionando novas funções. A classificação das tarefas por cores, impressão de tarefas e envio por e-mail são algumas dessas melhorias. Depois de realizadas as melhorias devem ser realizadas também novas avaliações do sistema, desta vez aplicando o *software* durante um projeto inteiro, coletando dados sobre a aplicação durante a execução do projeto.

8 Referências Bibliográficas

ACCESS IT. **Scrum Dashboard**. Disponível em: <http://labs.access-it.net/en-US/Projects/TFS/ScrumDashboard>. Acesso em: 28 junho 2010.

ABES. **Mercado Brasileiro de Software**: panorama e tendências, 2009. Disponível em: <http://www.abes.org.br>. Acesso em: 10 maio 2009.

BAIXAKI. **Como funcionam as telas sensíveis ao toque**. Disponível em: <http://www.baixaki.com.br/info/2449-como-funcionam-as-telas-sensiveis-ao-toque-touch-screen-.htm>. Acesso em: 25 abril 2010.

BASILI, V. R., G. Caldiera, H. D. Rombach. **Goal/Question/Metric Approach**. In: J. MARCINIAK (ed.), Encyclopedia of Software Engineering, volume 1. John Wiley & Sons, 1994.

DOT NET SOLUTIONS. **Scrum Wall**. Disponível em: <http://www.dotnetsolutions.co.uk/about-us/scrumwall>. Acesso em: 28 junho 2010.

EPISERVER. **Scrum Dashboard**. Disponível em: <http://scrumdashboard.codeplex.com>. Acesso em: 28 junho 2010.

IADIS. **Conferência Ibero-Americana 2003**. Disponível em: <http://www.iadis.org>. Acesso em: 14 maio 2010.

ISUPPLI. **Touch Screens Have the Magic Touch for Growth**. Disponível em: <http://www.isuppli.com/display-materials-and-systems/news/pages/touch-screens-score-touchdown-in-signage-and-professional-displays.aspx >. Acesso em: 21 abril 2010.

KNIBERG, H. Scrum e XP direto das Trincheiras. InfoQ, 2007. Disponível em: < http://www.infoq.com>. Acesso em: 10 maio 2010.

MICROGURU CORPORATION. **Scrum Dashboard**. Disponível em: <http://www.virtualscrumboard.com/>. Acesso em: 28 junho 2010.

OOH-TV. **Global Touch-Screen Market to Grow More Than Five-fold**. Disponível em: <http://en.ooh-tv.com/2010/01/14/global-touch-screen-market-to-grow-more-than-fivefold-by-2013/ >. Acesso em: 22 abril 2010.

PROJECT MANAGEMENT INSTITUTE. **Um Guia do Conjunto de Conhecimentos em Gerenciamento de Projetos: Guia PMBOK**. Project Management Institute, 2004. 388 p.

SCHWABER, K.; SUTHERLAND, J. **Scrum Guide**. Disponível em: <http://www.scrum.org>. Acesso em: 28 maio 2010.

MINISTÉRIO DA FAZENDA. **Tratamento Tributário da Micro e Pequena**

Empresa no Brasil. Disponível em: <http://www.receita.fazenda.gov.br/Publico/estudotributarios/estatisticas/09TratamentoTributarioMicroPequenaEmpresa.pdf>. Acesso em: 10 maio 2010.

THE STANDISH GROUP. **CHAOS Summary 2009**. Disponível em: <http://www.standishgroup.com>. Acesso em: 10 maio 2010.

UNIVERSIA. **Cietec: a maior incubadora da América Latina**. Disponível em: < http://www.universia.com.br/carreira/materia.jsp?materia=2783>. Acesso em: 12 maio 2010.

WIKIPEDIA. **Scrum (development)**. Disponível em: <http://en.wikipedia.org/wiki/Scrum_(development)>. Acesso em: 30 maio 2010.

APÊNDICE A.

DECLARAÇÃO DE CONCORDÂNCIA COM AS CONDIÇÕES PARA O DESENVOLVIMENTO DO TCC NA INSTITUIÇÃO

Declaro estar ciente das premissas para a realização de Trabalhos de Conclusão de Curso (TCC) de Ciência da Computação da UFSC, particularmente da necessidade de que se o TCC envolver o desenvolvimento de um software o código fonte e documentação completa correspondente ao *software multi-touch* para suportar a gerencia de *taskboard* deverá ser entregue integralmente, como parte integrante do relatório final do TCC.

Ciente dessa condição básica declaro estar de acordo com a realização do TCC identificado pelos dados apresentados a seguir.

Instituição	Sandbox Mídias Interativas
Nome do responsável	Luiz Daniel Lima
Cargo/Função	Sócio
Fone de contato	(48) 2107-2753
Acadêmico	Guilherme Furtado Pacheco
Título do trabalho	Aplicação de Monitores Touch Screen na Gerência do Taskboard
Curso	Ciência da Computação - INE - UFSC

Florianópolis, 15 de abril de 2010

Assinatura do responsável: _____

APÊNDICE B.

QUESTIONÁRIO DE AVALIAÇÃO DO SISTEMA

No.	Afirmação	Concordo	Concordo Parcialmente	Não Concordo
M1	Oferece a possibilidade de edição dos itens	◯	◯	◯
M2	Suporta a movimentação dos itens	◯	◯	◯
M3	É possível fazer anotações nos itens	◯	◯	◯
M4	As funções são intuitivas	◯	◯	◯
M5	Conseguiu realizar todas as tarefas típicas de uma reunião diária	◯	◯	◯
M6	O *software* pode substituir a versão em papel do *taskboard*	◯	◯	◯

M7. Quais são as vantagens do *software* com relação ao *taskboard* em papel?

_____

_____

_____

M8. Quais as desvantagens do sistema quando comparado à versão em papel do *taskboard*?

_____

_____

_____

P1. Encontrou algum erro no sistema? Se sim, cite-os.

_____

_____

P2. Caso queira, sugira melhorias para sistema.

_____

_____

APÊNDICE C.

DADOS COLETADOS DOS QUESTIONÁRIOS DE AVALIAÇÃO DO SISTEMA

Número de pessoas que responderam cada possível resposta:

No.	Afirmação	Concordo	Concordo Parcialmente	Não Concordo
M1	Oferece a possibilidade de edição dos itens	5	0	0
M2	Suporta a movimentação dos itens	5	0	0
M3	É possível fazer anotações nos itens	3	2	0
M4	As funções são intuitivas	2	3	0
M5	Conseguiu realizar todas as tarefas típicas de uma reunião diária	4	1	0
M6	O *software* pode substituir a versão em papel do *taskboard*	3	2	0

Respostas mais citadas:

M7. Quais são as vantagens do *software* com relação ao *taskboard* em papel?

- Registro digital das reuniões
- Economia nas despesas

M8. Quais as desvantagens do sistema quando comparado à versão em papel do *taskboard*?

- Investimento inicial por parte da empresa
- Falta suporte a imagens e desenhos

P1. Encontrou algum erro no sistema? Se sim, cite-os.

- Remoção de estória enquanto uma tarefa está destacada
- Perda de sensibilidade em alguns instantes

P2. Caso queira, sugira melhorias para sistema.

- Classificar tarefas por cores para cada coluna
- Impressão e envio das tarefas para os membros via e-mail

APÊNDICE D.

CÓDIGO FONTE DO SISTEMA DESENVOLVIDO

TaskboardManager.mxml

```
<?xml version="1.0" encoding="utf-8"?>
<mx:WindowedApplication xmlns:components="components.*"
                xmlns:smoothscroller="util.smoothscroller.*"
                xmlns:renderers="renderers.*"
                showStatusBar="false"
                showTitleBar="false"
                layout="absolute"
                applicationComplete="init()"
                horizontalScrollPolicy="off"
                verticalScrollPolicy="off"
                xmlns:mx="http://www.adobe.com/2006/mxml">

   <mx:HTTPService id="databaseLoader"
             url="database.xml"
             resultFormat="e4x"
             result="onDatabaseLoaded(event)"
             fault="readDatabase(SystemModelLocator.defaultDatabase)"/>

   <mx:Style source="assets/styles/style.css"/>

   <mx:Script>
      <![CDATA[
         import control.Story;
         import control.Task;
         import control.TeamMember;

         import events.StoryEvent;
         import events.TaskEvent;

         import model.SystemModelLocator;

         import mx.managers.PopUpManager;
         import mx.managers.ToolTipManager;
         import mx.rpc.events.ResultEvent;

         import renderers.EditStoryRenderer;
         import renderers.EditTaskRenderer;
         import renderers.TaskRenderer;

         private var databaseUrl:String;

         private var storyProvider:XMLList;
         [Bindable] private var backlogProvider:XMLList;
         [Bindable] private var inProgressProvider:XMLList;
         [Bindable] private var doneProvider:XMLList;
```

```
private var targetTask:TaskRenderer;
private var auxPoint:Point;
private var auxPoint2:Point;

[Bindable] private var dropList:String;

private function init():void
{
   stage.displayState = StageDisplayState.FULL_SCREEN;
   Multitouch.inputMode = MultitouchInputMode.TOUCH_POINT;
   ToolTipManager.enabled = false;

   databaseUrl = File.applicationStorageDirectory.nativePath;
   while (databaseUrl.indexOf("\\") >= 0)
   {
      databaseUrl = databaseUrl.replace("\\", "/");
   }
   databaseUrl += "/database.xml";
   databaseLoader.url = databaseUrl;
   databaseLoader.send();

   stage.addEventListener(StoryEvent.ADD, addStory);
   stage.addEventListener(StoryEvent.EDIT, editStory);
   stage.addEventListener(StoryEvent.REMOVE, removeStory);
   stage.addEventListener(TaskEvent.POPUP, popupTask);
   stage.addEventListener(TaskEvent.CLOSE, closeTask);
   stage.addEventListener(TaskEvent.ADD, addTask);
   stage.addEventListener(TaskEvent.EDIT, editTask);
   stage.addEventListener(TaskEvent.REMOVE, removeTask);
}

protected function onDatabaseLoaded(event:ResultEvent):void
{
   var xml:XML;
   try {
      xml = XML(event.result);
   }
   catch(error:Error)
   {
      readDatabase(SystemModelLocator.defaultDatabase);
   }
   readDatabase(xml);
}

private function readDatabase(xml:XML):void
{
   SystemModelLocator.project = xml.@name;

   for each (var memberXML:XML in xml.team.member)
   {
      var teamMember:TeamMember = new TeamMember();
      teamMember.name = memberXML.@name;
      SystemModelLocator.team.addItem(teamMember);
   }

   for each (var storyXML:XML in xml.story)
   {
```

```
            var story:Story = new Story(storyXML.@name);
            for each (var taskXML:XML in storyXML.task)
            {
              var task:Task = new Task();
              task.story = story;
              task.name = taskXML.@name;
              task.description = taskXML.@description;
              task.duration = taskXML.@duration;
              task.assigned =
SystemModelLocator.getTeamMember(taskXML.@assigned);
              task.state = taskXML.@state.toString();
              switch (task.state)
              {
                case "backlog":
                  story.inBacklogTasks.addItem(task);
                  break;
                case "inProgress":
                  story.inProgressTasks.addItem(task);
                  break;
                case "done":
                  story.doneTasks.addItem(task);
                  break;
              }
            }
            SystemModelLocator.stories.addItem(story);
          }
        }

        private function popupTask(event:TaskEvent):void
        {
          event.taskRenderer.state = "border";
          var task:TaskRenderer = new TaskRenderer();
          task.sameTask = event.taskRenderer;
          task.x = event.taskPoint.x;
          task.y = event.taskPoint.y;
          task.isPopup = true;
          task.addEventListener(TouchEvent.TOUCH_BEGIN, startMove);
          addChild(task);
        }

        private function closeTask(event:TaskEvent):void
        {
          event.taskRenderer.sameTask.state = "default";
          removeChild(event.taskRenderer);
          event.taskRenderer.removeEventListener(TouchEvent.TOUCH_BEGIN,
startMove);
          dropList = null;
        }

        private function addStory(event:StoryEvent):void
        {
          var story:Story = new Story("Nova Story");
          SystemModelLocator.stories.addItem(story);

          var storyEvent:StoryEvent = new StoryEvent(StoryEvent.EDIT);
          storyEvent.story = story;
          stage.dispatchEvent(storyEvent);
```

```
        }

        private function editStory(event:StoryEvent):void
        {
          var editStoryRenderer:EditStoryRenderer = new EditStoryRenderer();
          editStoryRenderer.data = event.story;
          PopUpManager.addPopUp(editStoryRenderer, this, true);
        }

        private function removeStory(event:StoryEvent):void
        {

  SystemModelLocator.stories.removeItemAt(SystemModelLocator.stories.getItemI
ndex(event.story));
        }

        private function addTask(event:TaskEvent):void
        {
          var task:Task = new Task();
          task.story = event.story;
          task.name = "Nova Tarefa";
          task.state = "backlog";
          task.description = "Descrição da nova tarefa";
          task.duration = 1;
          task.assigned = SystemModelLocator.team.getItemAt(0) as TeamMember;
          event.story.inBacklogTasks.addItem(task);

          var taskEvent:TaskEvent = new TaskEvent(TaskEvent.EDIT);
          taskEvent.task = task;
          stage.dispatchEvent(taskEvent);
        }

        private function editTask(event:TaskEvent):void
        {
          var editTaskRenderer:EditTaskRenderer = new EditTaskRenderer();
          editTaskRenderer.data = event.task;
          PopUpManager.addPopUp(editTaskRenderer, this, true);
        }

        private function removeTask(event:TaskEvent):void
        {
          if (event.story.inBacklogTasks.contains(event.task))
          {

  event.story.inBacklogTasks.removeItemAt(event.story.inBacklogTasks.getItemI
ndex(event.task));
          }
          else if (event.story.inProgressTasks.contains(event.task))
          {

  event.story.inProgressTasks.removeItemAt(event.story.inProgressTasks.getIte
mIndex(event.task));
          }
          else if (event.story.doneTasks.contains(event.task))
          {
```

```
  event.story.doneTasks.removeItemAt(event.story.doneTasks.getItemIndex(event
.task));
            }
        }

        private function startMove(event:TouchEvent):void
        {
            targetTask = event.currentTarget as TaskRenderer;
            auxPoint = new Point(targetTask.x, targetTask.y);
            auxPoint2 = new Point(event.stageX - auxPoint.x, event.stageY -
auxPoint.y);
            stage.addEventListener(TouchEvent.TOUCH_MOVE, doMove);
            stage.addEventListener(TouchEvent.TOUCH_END, endMove);
        }

        private function doMove(event:TouchEvent):void
        {
            var p:Point = globalToLocal(new Point(event.stageX - auxPoint2.x,
event.stageY - auxPoint2.y));
            targetTask.x = p.x;
            targetTask.y = p.y;

            if (p.y < 100)
            {
                dropList = null;
            }
            else if (p.x + 150 < width / 3)
            {
                dropList = "backlog";
            }
            else if (p.x + 150 < width * 2 / 3)
            {
                dropList = "inProgress";
            }
            else
            {
                dropList = "done";
            }
        }

        private function endMove(event:TouchEvent):void
        {
            if (dropList)
            {
                SystemModelLocator.changeTaskState(targetTask.data as Task,
dropList);
            }
            targetTask.sameTask.state = "default";
            removeChild(targetTask);
            dropList = null;
            targetTask.removeEventListener(TouchEvent.TOUCH_BEGIN, startMove);
            stage.removeEventListener(TouchEvent.TOUCH_MOVE, doMove);
            stage.removeEventListener(TouchEvent.TOUCH_END, endMove);
        }

        private function saveAndClose():void
```

```
        {
          var databaseString:String = "<project
name='"+SystemModelLocator.project+"'>\n";
          databaseString += "    <team>\n";
          for each(var teamMember:TeamMember in SystemModelLocator.team)
          {
            databaseString += "        <member
name='"+teamMember.name+"'/>\n";
          }
          databaseString += "    </team>\n";
          for each(var story:Story in SystemModelLocator.stories)
          {
            databaseString += "    <story name='"+story.name+"'>\n";
            for each(var task:Task in story.getAllTasks())
            {
              databaseString += "        <task name='"+task.name+"'\n";
              databaseString += "              state='"+task.state+"'\n";
              databaseString += "
duration='"+task.duration+"'\n";
              databaseString += "
assigned='"+task.assigned+"'\n";
              databaseString += "
description='"+task.description+"'/>\n";
            }
            databaseString += "    </story>\n";
          }
          databaseString += "</project>"

          var configStream:FileStream = new FileStream();
          configStream.open(new File(databaseUrl), FileMode.WRITE);
          configStream.writeUTFBytes(databaseString);
          configStream.close();

          close();
        }
    ]]>
  </mx:Script>

  <mx:VBox right="10"
        left="10"
        top="0"
        bottom="0"
        verticalGap="-10">
    <mx:HBox width="100%" verticalAlign="bottom">
      <mx:Label width="100%"
            fontSize="60"
            text="Taskboard Manager"
            color="#f8f8f8"
            fontWeight="bold">
        <mx:filters>
          <mx:DropShadowFilter angle="90"
                      blurX="3"
                      blurY="3"
                      color="#000000"
                      distance="1"
                      inner="true"/>
          <mx:BevelFilter angle="-90"
```

```
                blurX="1"
                blurY="1"
                distance="1"
                highlightColor="#388bf0"
                highlightAlpha=".75"
                shadowColor="#000000"
                shadowAlpha=".75"
                type="{BitmapFilterType.OUTER}"/>
    </mx:filters>
  </mx:Label>
</mx:HBox>
<mx:HBox width="100%" paddingLeft="5" verticalAlign="middle">
  <mx:Label text="+ Story"
        fontSize="18"
        fontWeight="bold"
        touchTap="stage.dispatchEvent(new StoryEvent(StoryEvent.ADD))"
        color="#333333"
        width="70">
    <mx:filters>
      <mx:DropShadowFilter angle="90"
                  blurX="2"
                  blurY="2"
                  color="#000000"
                  distance="1"
                  inner="true"/>
      <mx:BevelFilter angle="-90"
                blurX="1"
                blurY="1"
                distance="1"
                highlightColor="#2b74cd"
                highlightAlpha=".75"
                shadowColor="#000000"
                shadowAlpha=".75"
                type="{BitmapFilterType.OUTER}"/>
    </mx:filters>
  </mx:Label>
  <mx:Label width="100%"
        text="Backlog"
        textAlign="center"
        fontSize="36"
        fontWeight="bold"
        color="#333333">
    <mx:filters>
      <mx:DropShadowFilter angle="90"
                  blurX="2"
                  blurY="2"
                  color="#000000"
                  distance="1"
                  inner="true"/>
      <mx:BevelFilter angle="-90"
                blurX="1"
                blurY="1"
                distance="1"
                highlightColor="#2b74cd"
                highlightAlpha=".75"
                shadowColor="#000000"
                shadowAlpha=".75"
```

```
                    type="{BitmapFilterType.OUTER}"/>
          </mx:filters>
        </mx:Label>
        <mx:Label width="100%"
               text="In Progress"
               textAlign="center"
               fontSize="36"
               fontWeight="bold"
               color="#333333">
          <mx:filters>
            <mx:DropShadowFilter angle="90"
                        blurX="2"
                        blurY="2"
                        color="#000000"
                        distance="1"
                        inner="true"/>
            <mx:BevelFilter angle="-90"
                      blurX="1"
                      blurY="1"
                      distance="1"
                      highlightColor="#2b74cd"
                      highlightAlpha=".75"
                      shadowColor="#000000"
                      shadowAlpha=".75"
                      type="{BitmapFilterType.OUTER}"/>
          </mx:filters>
        </mx:Label>
        <mx:Label width="100%"
               text="Done"
               textAlign="center"
               fontSize="36"
               fontWeight="bold"
               color="#333333">
          <mx:filters>
            <mx:DropShadowFilter angle="90"
                        blurX="2"
                        blurY="2"
                        color="#000000"
                        distance="1"
                        inner="true"/>
            <mx:BevelFilter angle="-90"
                      blurX="1"
                      blurY="1"
                      distance="1"
                      highlightColor="#2b74cd"
                      highlightAlpha=".75"
                      shadowColor="#000000"
                      shadowAlpha=".75"
                      type="{BitmapFilterType.OUTER}"/>
          </mx:filters>
        </mx:Label>
      </mx:HBox>
      <mx:Spacer height="20"/>
      <mx:Canvas width="100%" height="100%" verticalScrollPolicy="off"
horizontalScrollPolicy="off">
        <smoothscroller:SmoothScroller id="scroller1" width="100%"
height="100%">
```

```
            <smoothscroller:child>
              <mx:List id="storiesList"
                    width="100%"
                    itemRenderer="renderers.StoryRenderer"
                    dataProvider="{SystemModelLocator.stories}"
                    selectable="false"
                    backgroundAlpha="0.0"
                    rowCount="{SystemModelLocator.stories.length}"
                    borderStyle="none"/>
            </smoothscroller:child>
          </smoothscroller:SmoothScroller>
          <mx:Image top="0"
source="@Embed(source='/assets/images/storyGradientTop.png')" width="100%"
maintainAspectRatio="false"/>
          <mx:Image bottom="0"
source="@Embed(source='/assets/images/storyGradientBottom.png')" width="100%"
maintainAspectRatio="false"/>
        </mx:Canvas>
      </mx:VBox>

      <mx:Label right="-20"
            top="5"
            text="x"
            fontSize="40"
            touchTap="saveAndClose()"
            styleName="iconButton"
            color="#333333">
        <mx:filters>
          <mx:DropShadowFilter angle="90"
                       blurX="2"
                       blurY="2"
                       color="#000000"
                       distance="1"
                       inner="true"/>
          <mx:BevelFilter angle="-90"
                    blurX="1"
                    blurY="1"
                    distance="1"
                    highlightColor="#2b74cd"
                    highlightAlpha=".75"
                    shadowColor="#000000"
                    shadowAlpha=".75"
                    type="{BitmapFilterType.OUTER}"/>
        </mx:filters>
      </mx:Label>

</mx:WindowedApplication>
```

TaskboardManager-app.xml

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<application xmlns="http://ns.adobe.com/air/application/2.0">

  <!-- The application identifier string, unique to this application.
Required. -->
  <id>TaskboardManager</id>

  <!-- Used as the filename for the application. Required. -->
  <filename>TaskboardManager</filename>

  <!-- The name that is displayed in the AIR application installer.
       May have multiple values for each language. See samples or xsd schema
file. Optional. -->
  <name>TaskboardManager</name>

  <!-- An application version designator (such as "v1", "2.5", or "Alpha 1").
Required. -->
  <version>v1</version>


  <!-- Settings for the application's initial window. Required. -->
  <initialWindow>
    <!-- The main SWF or HTML file of the application. Required. -->
    <!-- Note: In Flash Builder, the SWF reference is set automatically. -->
    <content>[This value will be overwritten by Flash Builder in the output
app.xml]</content>

    <!-- The type of system chrome to use (either "standard" or "none").
Optional. Default standard. -->
    <systemChrome>none</systemChrome>

    <!-- The window's initial width. Optional. -->
    <width>1280</width>

    <!-- The window's initial height. Optional. -->
    <height>720</height>

    <!-- The window's initial x position. Optional. -->
    <x>0</x>

    <!-- The window's initial y position. Optional. -->
    <y>0</y>

  </initialWindow>

</application>
```

control.Story.as

```
package control
{
   import flash.events.Event;
   import flash.events.EventDispatcher;

   import mx.collections.ArrayCollection;
   import mx.events.CollectionEvent;

   [Bindable]
   public class Story extends EventDispatcher
   {

      public var name:String;
      public var inBacklogTasks:ArrayCollection = new ArrayCollection();
      public var inProgressTasks:ArrayCollection = new ArrayCollection();
      public var doneTasks:ArrayCollection = new ArrayCollection();

      public function Story(name:String)
      {
         this.name = name;
         inBacklogTasks.addEventListener(CollectionEvent.COLLECTION_CHANGE,
updateAllTasksList);
         inProgressTasks.addEventListener(CollectionEvent.COLLECTION_CHANGE,
updateAllTasksList);
         doneTasks.addEventListener(CollectionEvent.COLLECTION_CHANGE,
updateAllTasksList);
      }

      private function updateAllTasksList(event:Event):void
      {
         dispatchEvent(new Event("tasksChanged"));
      }

      [Bindable("tasksChanged")]
      public function getAllTasks():ArrayCollection
      {
         var allTasks:ArrayCollection = new ArrayCollection();
         allTasks.addAll(inBacklogTasks);
         allTasks.addAll(inProgressTasks);
         allTasks.addAll(doneTasks);
         return allTasks;
      }

   }
}
```

control.Task.as

```
package control
{

   [Bindable]
   public class Task
   {
      public var story:Story;
      public var name:String;
      public var description:String;
      public var duration:int;
      public var assigned:TeamMember;
      public var state:String;

      public function Task()
      {
      }
   }
}
```

control.TeamMember.as

```
package control
{

   [Bindable]
   public class TeamMember
   {
      public var name:String;

      public function TeamMember()
      {
      }

      public function toString():String
      {
         return name;
      }
   }
}
```

events.StoryEvent.as

```
package events
{
   import control.Story;

   import flash.events.Event;
   import flash.geom.Point;

   import renderers.StoryRenderer;
   import renderers.TaskRenderer;

   public class StoryEvent extends Event
   {
      public static const ADD:String = "addStory";
      public static const EDIT:String = "editStory";
      public static const REMOVE:String = "removeStory";

      public var story:Story;

      public function StoryEvent(type:String, bubbles:Boolean=false,
cancelable:Boolean=false)
      {
         super(type, bubbles, cancelable);
      }

   }
}
```

control.TaskEvent.as

```
package events
{
   import control.Story;
   import control.Task;

   import flash.events.Event;
   import flash.geom.Point;

   import renderers.TaskRenderer;

   public class TaskEvent extends Event
   {
      public static const POPUP:String = "popupTask";
      public static const CLOSE:String = "closeTask";
      public static const ADD:String = "addTask";
      public static const EDIT:String = "editTask";
      public static const REMOVE:String = "removeTask";

      public var story:Story;
      public var task:Task;
      public var taskRenderer:TaskRenderer;
      public var taskPoint:Point;
```

```
     public function TaskEvent(type:String, bubbles:Boolean=false,
cancelable:Boolean=false)
     {
        super(type, bubbles, cancelable);
     }

   }
}
```

control.StringEvent.as

```
package events
{
   import flash.events.Event;

   public class StringEvent extends Event
   {
      public var str:String;

      public function StringEvent(type:String, str:String)
      {
         super(type);
         this.str = str;
      }

   }
}
```

model.SystemModelLocator.as

```
package model
{
   import control.Story;
   import control.Task;
   import control.TeamMember;

   import mx.collections.ArrayCollection;

   [Bindable]
   public class SystemModelLocator
   {
      public static var project:String = "";
      public static var stories:ArrayCollection = new ArrayCollection();
      public static var team:ArrayCollection = new ArrayCollection();

      public static var isScrolling:Boolean = false;

      public static function getTeamMember(memberName:String):TeamMember
      {
         for each (var member:TeamMember in team)
         {
            if(member.name == memberName)
            {
               return member;
            }
         }
         return null;
      }

      public static function changeTaskState(taskToChange:Task,
state:String):void
      {
         var taskStory:Story;
         var found:Boolean = false;
         for each (var story:Story in stories)
         {
            for each (var task:Task in story.inBacklogTasks)
            {
               if(taskToChange == task)
               {

   story.inBacklogTasks.removeItemAt(story.inBacklogTasks.getItemIndex(task));
                  taskStory = story;
                  found = true;
                  break;
               }
            }
            if(found)
            {
               break;
            }
            for each (task in story.inProgressTasks)
            {
```

```
            if(taskToChange == task)
            {

  story.inProgressTasks.removeItemAt(story.inProgressTasks.getItemIndex(task)
);
              taskStory = story;
              found = true;
              break;
            }
          }
          if(found)
          {
            break;
          }
          for each (task in story.doneTasks)
          {
            if(taskToChange == task)
            {

  story.doneTasks.removeItemAt(story.doneTasks.getItemIndex(task));
              taskStory = story;
              found = true;
              break;
            }
          }
          if(found)
          {
            break;
          }
        }
        switch(state)
        {
          case "backlog":
            taskStory.inBacklogTasks.addItem(taskToChange);
            break;
          case "inProgress":
            taskStory.inProgressTasks.addItem(taskToChange);
            break;
          case "done":
            taskStory.doneTasks.addItem(taskToChange);
            break;
        }
        taskToChange.state = state;
      }

      public static var defaultDatabase:XML =
        <project name="Projeto 1">
          <team>
            <member name="Membro 1"/>
            <member name="Membro 2"/>
            <member name="Membro 3"/>
          </team>
        </project>;

  }
}
```

renderers.StoryRenderer.mxml

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Canvas xmlns:mx="http://www.adobe.com/2006/mxml"
        width="100%"
        xmlns:smoothscroller="util.smoothscroller.*"
        horizontalScrollPolicy="off"
        verticalScrollPolicy="off"
        height="441"
        clipContent="false">

  <mx:Script>
    <![CDATA[
      import control.Story;

      import events.StoryEvent;
      import events.TaskEvent;

      import mx.collections.XMLListCollection;
      import mx.controls.Alert;
      import mx.core.Application;
      import mx.effects.easing.Sine;
      import mx.events.CloseEvent;

      [Bindable] private var taskProvider:XMLListCollection;
      [Bindable] private var isPopup:Boolean = false;

      private function addTask():void
      {
        var taskEvent:TaskEvent = new TaskEvent(TaskEvent.ADD);
        taskEvent.story = data as Story;
        stage.dispatchEvent(taskEvent);
      }

      private function editStory():void
      {
        var storyEvent:StoryEvent = new StoryEvent(StoryEvent.EDIT);
        storyEvent.story = data as Story;
        stage.dispatchEvent(storyEvent);
      }

      private function confirmRemove():void
      {
        Alert.show("Você deseja realmente remover essa estória?", "Atenção",
Alert.YES|Alert.NO, TaskboardManager(Application.application), removeStory,
null, Alert.YES);
      }

      private function removeStory(event:CloseEvent):void
      {
        if(event.detail == Alert.YES)
        {
          var storyEvent:StoryEvent = new StoryEvent(StoryEvent.REMOVE);
          storyEvent.story = data as Story;
          stage.dispatchEvent(storyEvent);
```

```
        }
      }
   ]]>
</mx:Script>

<mx:Canvas styleName="storyCanvas"
        left="5"
        top="30"
        bottom="30"
        right="5"
        cornerRadius="10"
        clipContent="false">
   <mx:filters>
      <mx:DropShadowFilter angle="90"
                  distance="4"
                  blurX="10"
                  blurY="10"
                  alpha="0.8"/>
   </mx:filters>
   <mx:Label x="5"
          y="376"
          width="371"
          text="{data.name}"
          fontSize="18"
          fontWeight="bold"
          textAlign="center"
          rotation="-90"
          touchTap="isPopup = !isPopup;"/>
   <mx:Label x="5"
          y="{isPopup? 110 : 70}"
          text="+ Task"
          fontSize="18"
          fontWeight="bold"
          rotation="-90"
          moveEffect="Move"
          touchTap="addTask()"/>
   <mx:Label text="P"
          fontSize="40"
          left="6"
          top="-2"
          width="35"
          touchTap="editStory()"
          touchBegin="event.stopPropagation()"
          styleName="iconButton"
          visible="{isPopup}"
          showEffect="Fade"
          hideEffect="Fade"/>
   <mx:Label x="5"
          y="376"
          text="- Story"
          fontSize="18"
          fontWeight="bold"
          rotation="-90"
          moveEffect="Move"
          touchTap="confirmRemove()"
          visible="{isPopup}"
          showEffect="Fade"
```

```
            hideEffect="Fade"/>
    </mx:Canvas>
    <mx:HBox left="45"
          right="20"
          top="10"
          bottom="10"
          horizontalGap="10">
      <smoothscroller:SmoothScroller width="100%" height="100%">
        <smoothscroller:child>
          <mx:TileList id="backlogList"
                  backgroundAlpha="0.0"
                  borderStyle="none"
                  height="100%"
                  rowCount="2"
                  width="{Math.ceil(backlogList.dataProvider.length / 2) *
TaskRenderer.taskWidth + 5}"
                  dataProvider="{data.inBacklogTasks}"
                  direction="vertical"
                  itemRenderer="renderers.TaskRenderer"
                  horizontalScrollPolicy="off"
                  selectable="false"
                  paddingLeft="5"/>
        </smoothscroller:child>
      </smoothscroller:SmoothScroller>
      <smoothscroller:SmoothScroller width="100%" height="100%">
        <smoothscroller:child>
          <mx:TileList id="inProgressList"
                  backgroundAlpha="0.0"
                  borderStyle="none"
                  height="100%"
                  rowCount="2"
                  width="{Math.ceil(inProgressList.dataProvider.length / 2) *
TaskRenderer.taskWidth + 5}"
                  dataProvider="{data.inProgressTasks}"
                  direction="vertical"
                  itemRenderer="renderers.TaskRenderer"
                  horizontalScrollPolicy="off"
                  selectable="false"
                  paddingLeft="5"/>
        </smoothscroller:child>
      </smoothscroller:SmoothScroller>
      <smoothscroller:SmoothScroller width="100%" height="100%">
        <smoothscroller:child>
          <mx:TileList id="doneList"
                  backgroundAlpha="0.0"
                  borderStyle="none"
                  height="100%"
                  rowCount="2"
                  width="{Math.ceil(doneList.dataProvider.length / 2) *
TaskRenderer.taskWidth + 5}"
                  dataProvider="{data.doneTasks}"
                  direction="vertical"
                  itemRenderer="renderers.TaskRenderer"
                  horizontalScrollPolicy="off"
                  selectable="false"
                  paddingLeft="5"/>
        </smoothscroller:child>
```

```
      </smoothscroller:SmoothScroller>
   </mx:HBox>
   <mx:HBox left="45"
         right="20"
         top="10"
         bottom="10"
         horizontalGap="0">
      <mx:Image source="@Embed(source='/assets/images/taskGradientLeft.png')"/>
      <mx:Spacer width="100%"/>
      <mx:Image
source="@Embed(source='/assets/images/taskGradientRight.png')"/>
      <mx:Spacer width="10"/>
      <mx:Image source="@Embed(source='/assets/images/taskGradientLeft.png')"/>
      <mx:Spacer width="100%"/>
      <mx:Image
source="@Embed(source='/assets/images/taskGradientRight.png')"/>
      <mx:Spacer width="10"/>
      <mx:Image source="@Embed(source='/assets/images/taskGradientLeft.png')"/>
      <mx:Spacer width="100%"/>
      <mx:Image
source="@Embed(source='/assets/images/taskGradientRight.png')"/>
   </mx:HBox>
</mx:Canvas>
```

renderers.TaskRenderer.mxml

```
<?xml version="1.0" encoding="utf-8"?>
<mx:ViewStack width="{taskWidth}"
          height="200"
          xmlns:mx="http://www.adobe.com/2006/mxml"
          resizeToContent="true"
          touchEnd="popupTask()"
          keyDown="Alert.show(''+event.keyCode)"
          xmlns:renderers="renderers.*"
          clipContent="false"
          creationComplete="init()"
          showEffect="{creationEffect}">

   <mx:Fade id="creationEffect" target="{this}" duration="500"/>

   <mx:Move id="popupEffect"
         target="{this}"
         xBy="15"
         yBy="-15"
         easingFunction="{Elastic.easeOut}"
         duration="500"/>

   <mx:Script>
      <![CDATA[
         import control.Task;
         import control.TeamMember;

         import events.TaskEvent;
```

```
import model.SystemModelLocator;

import mx.controls.Alert;
import mx.core.Application;
import mx.effects.easing.Elastic;
import mx.events.CloseEvent;

public static const taskWidth:int = 288;

private var _sameTask:TaskRenderer;

[Bindable] public var isPopup:Boolean = false;

private function init():void
{
   if (isPopup)
   {
      popupEffect.play();
   }
   else
   {
      creationEffect.play();
   }
}

public function set state(value:String):void
{
   switch (value)
   {
      case "default":
         selectedIndex = 0;
         break;
      case "border":
         selectedIndex = 1;
         break;
   }
}

public function set sameTask(value:TaskRenderer):void
{
   _sameTask = value;
   data = value.data;
}

public function get sameTask():TaskRenderer
{
   return _sameTask;
}

private function popupTask():void
{
   if (!isPopup && !SystemModelLocator.isScrolling)
   {
      var taskEvent:TaskEvent = new TaskEvent(TaskEvent.POPUP);
      taskEvent.taskRenderer = this;
      var pt:Point = localToGlobal(new Point(0, 0));
```

```
            taskEvent.taskPoint = pt;
            stage.dispatchEvent(taskEvent);
          }
        }

        private function editTask():void
        {
          var taskEvent:TaskEvent = new TaskEvent(TaskEvent.EDIT);
          taskEvent.task = data as Task;
          taskEvent.taskRenderer = this;
          stage.dispatchEvent(taskEvent);
        }

        private function confirmRemove():void
        {
          Alert.show("Você deseja realmente remover essa tarefa?", "Atenção",
Alert.YES|Alert.NO, TaskboardManager(Application.application), removeTask,
null, Alert.YES);
        }

        private function removeTask(event:CloseEvent):void
        {
          if(event.detail == Alert.YES)
          {
            var taskEvent:TaskEvent = new TaskEvent(TaskEvent.REMOVE);
            taskEvent.story = (data as Task).story;
            taskEvent.task = data as Task;
            stage.dispatchEvent(taskEvent);
            parent.removeChild(this);
          }
        }
      ]]>
    </mx:Script>

    <mx:Canvas id="defaultState"
            width="100%"
            height="100%"
            clipContent="false"
            verticalScrollPolicy="off"
            horizontalScrollPolicy="off">
      <mx:filters>
        <mx:DropShadowFilter angle="90"
                    distance="4"
                    blurX="10"
                    blurY="10"
                    alpha="0.8"/>
        <mx:GlowFilter alpha=".3"
                  blurX="4"
                  blurY="4"
                  color="#74b3ff"
                  inner="true"/>
      </mx:filters>
      <mx:Canvas right="5"
              left="5"
              top="5"
              bottom="5"
```

```
            styleName="{(data.state == 'backlog')? 'backlogTaskCanvas' :
((data.state == 'inProgress')? 'inProgressTaskCanvas' : 'doneTaskCanvas')}"
            verticalScrollPolicy="off"
            horizontalScrollPolicy="off">
        <mx:Label left="5"
                  top="5"
                  text="{data.name}"
                  fontWeight="bold"
                  fontSize="16"
                  textAlign="left"
                  width="{isPopup? 195 : 280}"/>
        <mx:Text top="36"
                 text="{data.description}"
                 selectable="false"
                 fontSize="14"
                 textAlign="left"
                 width="261"
                 left="7"
                 height="122"/>
        <mx:Label left="5"
                  bottom="5"
                  text="{data.duration + ((data.duration > 1)? ' dias' : ' dia')}"
                  fontSize="14"
                  width="130"
                  textAlign="left"/>
        <mx:Label right="5"
                  bottom="5"
                  text="{data.assigned.name}"
                  fontSize="14"
                  width="130"
                  textAlign="right"/>
        <mx:Label text="P"
                  fontSize="36"
                  right="0"
                  top="-2"
                  width="35"
                  touchTap="editTask()"
                  touchBegin="event.stopPropagation()"
                  styleName="iconButton"
                  visible="{isPopup}"/>
        <mx:Label text="-"
                  fontSize="36"
                  top="-2"
                  width="35"
                  touchTap="confirmRemove()"
                  touchBegin="event.stopPropagation()"
                  styleName="iconButton"
                  visible="{isPopup}"
                  right="41"/>
    </mx:Canvas>
  </mx:Canvas>

  <mx:Canvas id="borderState" width="100%" height="100%">
    <mx:Canvas right="10"
               left="10"
               top="10"
               bottom="10"
```

```
            borderStyle="solid"
            cornerRadius="7"/>
  </mx:Canvas>
</mx:ViewStack>
```

renderers.EditStoryRenderer.mxml

```
<?xml version="1.0" encoding="utf-8"?>
<mx:VBox xmlns:mx="http://www.adobe.com/2006/mxml"
      creationComplete="init()"
      fontSize="18"
      xmlns:renderers="renderers.*"
      horizontalAlign="center">

  <mx:Script>
    <![CDATA[
      import control.Story;
      import control.Task;
      import control.TeamMember;

      import events.StringEvent;
      import events.TaskEvent;

      import model.SystemModelLocator;

      import mx.collections.ArrayCollection;
      import mx.managers.PopUpManager;

      private function init():void
      {
        x = (stage.width - width) / 2;
        y = (stage.height - height) / 2;
        nameInput.setFocus();
      }

      private function onKeyDown(event:StringEvent):void
      {
        var beforeText:String = nameInput.text.substring(0,
nameInput.selectionBeginIndex);
        var afterText:String =
nameInput.text.substring(nameInput.selectionEndIndex, nameInput.text.length);
        nameInput.text = beforeText + event.str + afterText;
        nameInput.setSelection(nameInput.selectionBeginIndex +
event.str.length, nameInput.selectionBeginIndex + event.str.length);
      }

      private function onBackspaceDown(event:Event):void
      {
        var beforeText:String = nameInput.text.substr(0,
nameInput.selectionBeginIndex);
        var afterText:String =
nameInput.text.substring(nameInput.selectionEndIndex, nameInput.text.length);
        if (nameInput.selectionBeginIndex != nameInput.selectionEndIndex)
```

```
            {
              nameInput.text = beforeText + afterText;
              nameInput.setSelection(nameInput.selectionBeginIndex,
nameInput.selectionBeginIndex);
            }
            else
            {
              nameInput.text = beforeText.substr(0, beforeText.length - 1) +
afterText;
              nameInput.setSelection(nameInput.selectionBeginIndex - 1,
nameInput.selectionBeginIndex - 1);
            }
          }

          private function addTask():void
          {
            var taskEvent:TaskEvent = new TaskEvent(TaskEvent.ADD);
            taskEvent.story = data as Story;
            stage.dispatchEvent(taskEvent);
          }

          private function editTask():void
          {
            if (tasksDataGrid.selectedIndex >= 0)
            {
              var taskEvent:TaskEvent = new TaskEvent(TaskEvent.EDIT);
              var taskRenderer:TaskRenderer = new TaskRenderer();
              taskRenderer.data = tasksDataGrid.selectedItem;
              taskEvent.taskRenderer = taskRenderer;
              stage.dispatchEvent(taskEvent);
            }
          }

          private function removeTask():void
          {
            if (tasksDataGrid.selectedIndex >= 0)
            {
              var taskEvent:TaskEvent = new TaskEvent(TaskEvent.REMOVE);
              taskEvent.story = data as Story;
              taskEvent.task = tasksDataGrid.selectedItem as Task;
              stage.dispatchEvent(taskEvent);
            }
          }

          private function save():void
          {
            data.name = nameInput.text;
            PopUpManager.removePopUp(this);
          }
        ]]>
      </mx:Script>

      <mx:VBox width="700"
               paddingBottom="15"
               paddingLeft="15"
               paddingRight="15"
               paddingTop="5"
```

```
        styleName="storyCanvas">
      <mx:HBox verticalAlign="middle" width="100%">
        <mx:Label width="100%" text="Editar Story" fontWeight="bold"/>
        <mx:Label text="y"
               fontSize="40"
               width="35"
               touchTap="save()"
               styleName="iconButton"/>
      </mx:HBox>
      <mx:HBox verticalAlign="middle">
        <mx:Label text="Nome:"/>
        <mx:TextInput id="nameInput"
                 width="300"
                 text="{data.name}"
                 change="data.name = nameInput.text;"/>
      </mx:HBox>
      <mx:HBox width="100%" verticalAlign="middle" horizontalGap="15">
        <mx:Label width="100%" text="Tarefas:"/>
        <mx:Label width="35"
               text="+"
               fontSize="36"
               touchTap="addTask()"
               styleName="iconButton"/>
        <mx:Label width="35"
               text="P"
               fontSize="36"
               touchTap="editTask()"
               styleName="iconButton"
               enabled="{tasksDataGrid.selectedIndex >= 0}"/>
        <mx:Label width="35"
               text="-"
               fontSize="36"
               touchTap="removeTask()"
               styleName="iconButton"
               enabled="{tasksDataGrid.selectedIndex >= 0}"/>
      </mx:HBox>
      <mx:Spacer height="5"/>
      <mx:DataGrid id="tasksDataGrid" width="100%"
dataProvider="{Story(data).getAllTasks()}">
        <mx:columns>
          <mx:DataGridColumn headerText="Nome" dataField="name"/>
          <mx:DataGridColumn headerText="Duração" dataField="duration"/>
          <mx:DataGridColumn headerText="Membro responsável"
dataField="assigned"/>
        </mx:columns>
      </mx:DataGrid>
    </mx:VBox>
    <renderers:TouchKeyboard horizontalCenter="0"
                  bottom="5"
                  touchKeyDown="onKeyDown(event)"
                  touchBackspaceDown="onBackspaceDown(event)"
                  focusEnabled="false"/>

</mx:VBox>
```

renderers.EditTaskRenderer.mxml

```
<?xml version="1.0" encoding="utf-8"?>
<mx:VBox xmlns:mx="http://www.adobe.com/2006/mxml"
      xmlns:renderers="renderers.*"
      horizontalAlign="center"
      touchBegin="event.stopPropagation()"
      creationComplete="init()"
      verticalGap="15">

   <mx:Script>
      <![CDATA[
         import events.StringEvent;
         import events.TaskEvent;

         import model.SystemModelLocator;

         import mx.controls.Alert;
         import mx.managers.PopUpManager;

         private var textInput:*;

         private function init():void
         {
            x = (stage.width - width) / 2;
            y = (stage.height - height) / 2;
            nameInput.setFocus();
            textInput = nameInput;
         }

         private function save():void
         {
            try
            {
               var dur:int = parseInt(durationInput.text);
               data.name = nameInput.text;
               data.description = descriptionInput.text;
               data.duration = parseInt(durationInput.text);
               data.assigned = memberInput.selectedItem;
               PopUpManager.removePopUp(this);
            }
            catch (e:Error)
            {
               Alert.show("A duração da tarefa\nsó pode conter números", "Erro");
            }
         }

         private function onKeyDown(event:StringEvent):void
         {
            var beforeText:String = textInput.text.substring(0,
textInput.selectionBeginIndex);
            var afterText:String =
textInput.text.substring(textInput.selectionEndIndex, textInput.text.length);
            textInput.text = beforeText + event.str + afterText;
```

```
          textInput.setSelection(textInput.selectionBeginIndex +
event.str.length, textInput.selectionBeginIndex + event.str.length);
        }

        private function onEnterDown(event:Event):void
        {
          if (textInput is TextArea)
          {
            var beforeText:String = textInput.text.substr(0,
textInput.selectionBeginIndex);
            var afterText:String =
textInput.text.substring(textInput.selectionEndIndex, textInput.text.length);
            textInput.text = beforeText + "\n" + afterText;
            textInput.setSelection(textInput.selectionBeginIndex + 1,
textInput.selectionBeginIndex + 1);
          }
        }

        private function onBackspaceDown(event:Event):void
        {
          var beforeText:String = textInput.text.substr(0,
textInput.selectionBeginIndex);
          var afterText:String =
textInput.text.substring(textInput.selectionEndIndex, textInput.text.length);
          if (textInput.selectionBeginIndex != textInput.selectionEndIndex)
          {
            textInput.text = beforeText + afterText;
            textInput.setSelection(textInput.selectionBeginIndex,
textInput.selectionBeginIndex);
          }
          else
          {
            textInput.text = beforeText.substr(0, beforeText.length - 1) +
afterText;
            textInput.setSelection(textInput.selectionBeginIndex - 1,
textInput.selectionBeginIndex - 1);
          }
        }
      ]]>
    </mx:Script>

    <mx:Canvas width="480"
           height="360"
           cornerRadius="7"
           fontSize="18"
           styleName="{(data.state == 'backlog')? 'backlogTaskCanvas' :
((data.state == 'inProgress')? 'inProgressTaskCanvas' : 'doneTaskCanvas')}">
      <mx:HBox verticalAlign="middle" left="10" right="10">
        <mx:Label width="100%" text="Editar Tarefa" fontWeight="bold"/>
        <mx:Label text="y"
              fontSize="40"
              width="35"
              touchTap="save()"
              styleName="iconButton"/>
        <mx:Label text="x"
              fontSize="40"
              width="35"
```

```
                touchTap="PopUpManager.removePopUp(this)"
                styleName="iconButton"/>
        </mx:HBox>
        <mx:TextInput id="nameInput"
                right="10"
                left="10"
                fontSize="18"
                fontWeight="bold"
                text="{data.name}"
                touchTap="textInput = nameInput" top="47"/>
        <mx:TextArea id="descriptionInput"
                right="10"
                left="10"
                top="87"
                bottom="50"
                borderStyle="inset"
                fontSize="16"
                text="{data.description}"
                touchTap="textInput = descriptionInput"/>
        <mx:TextInput id="durationInput"
                x="4"
                left="10"
                y="90"
                bottom="10"
                width="150"
                fontSize="18"
                text="{data.duration}"
                touchTap="textInput = durationInput;
durationInput.setSelection(0, durationInput.text.length)"
                textAlign="right"/>
        <mx:Label x="4"
                left="168"
                y="162"
                bottom="12"
                text="{(durationInput.text == '1')? 'Dia' : 'Dias'}"
                fontSize="18"/>
        <mx:ComboBox id="memberInput"
                x="4"
                right="10"
                y="126"
                bottom="10"
                width="200"
                editable="false"
                fontSize="18"
                dataProvider="{SystemModelLocator.team}"
                selectedItem="{data.assigned}"
                labelField="name"
                focusEnabled="false"/>
    </mx:Canvas>
    <renderers:TouchKeyboard horizontalCenter="0"
                    bottom="5"
                    touchKeyDown="onKeyDown(event)"
                    touchEnterDown="onEnterDown(event)"
                    touchBackspaceDown="onBackspaceDown(event)"
                    focusEnabled="false"/>
</mx:VBox>
```

renderers.TouchKeyboard.mxml

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Canvas xmlns:mx="http://www.adobe.com/2006/mxml"
        width="1024"
        height="356"
        styleName="keyboardCanvas">

   <mx:Metadata>
      [Event(name="touchKeyDown", type="events.StringEvent")]
      [Event(name="touchEnterDown", type="flash.events.Event")]
      [Event(name="touchBackspaceDown", type="flash.events.Event")]
   </mx:Metadata>

   <mx:Script>
      <![CDATA[
         import events.StringEvent;

         [Bindable] public var buttonSize:int = 60;

         private var accent:String;
         private var capslock:Boolean = false;

         private function keyDown(key:String):void
         {
            if(keyboardMode.selectedChild == upperCase && !capslock)
            {
               keyboardMode.selectedChild = lowerCase;
            }
            switch(key)
            {
               case "ENTER":
                  dispatchEvent(new Event("touchEnterDown"));
                  break;
               case "BACKSPACE":
                  dispatchEvent(new Event("touchBackspaceDown"));
                  break;
               case "CAPSLOCK":
                  if(keyboardMode.selectedChild == upperCase)
                  {
                     capslock = false;
                     keyboardMode.selectedChild = lowerCase;
                  }
                  else
                  {
                     capslock = true;
                     keyboardMode.selectedChild = upperCase;
                  }
                  break;
               case "SHIFT":
                  if(keyboardMode.selectedChild == lowerCase)
                  {
                     keyboardMode.selectedChild = upperCase;
                  }
                  else
```

```
                {
                    keyboardMode.selectedChild = lowerCase;
                }
                break;
            case "SPACE":
                if(!accent) {
                    key = " ";
                }
                else
                {
                    key = accent;
                    accent = null;
                }
                dispatchEvent(new StringEvent("touchKeyDown",key));
                break;
            case "´":
            case "~":
            case "^":
            case "`":
                if(!accent) {
                    accent = key;
                }
                else
                {
                    dispatchEvent(new
StringEvent("touchKeyDown",addAccent(key)));
                }
                break;
            default:
                dispatchEvent(new StringEvent("touchKeyDown",addAccent(key)));
                break;
        }
    }

    private function addAccent(str:String):String {
        if(accent)
        {
            switch(accent) {
                case "´":
                    if(str == "a") { str = "á"; }
                    else if(str == "A") { str = "Á"; }
                    else if(str == "e") { str = "é"; }
                    else if(str == "E") { str = "É"; }
                    else if(str == "i") { str = "í"; }
                    else if(str == "I") { str = "Í"; }
                    else if(str == "o") { str = "ó"; }
                    else if(str == "O") { str = "Ó"; }
                    else if(str == "u") { str = "ú"; }
                    else if(str == "U") { str = "Ú"; }
                    else if(str == " ") { str = "´"; }
                    else { str = "´"+str; }
                    break;
                case "~":
                    if(str == "a") { str = "ã"; }
                    else if(str == "A") { str = "Ã"; }
                    else if(str == "o") { str = "õ"; }
                    else if(str == "O") { str = "Õ"; }
```

```
                else if(str == "n") { str = "ñ"; }
                else if(str == "N") { str = "Ñ"; }
                else if(str == " ") { str = "~"; }
                else { str = "~"+str; }
                break;
              case "^":
                if(str == "a") { str = "â"; }
                else if(str == "A") { str = "Â"; }
                else if(str == "e") { str = "ê"; }
                else if(str == "E") { str = "Ê"; }
                else if(str == "i") { str = "î"; }
                else if(str == "I") { str = "Î"; }
                else if(str == "o") { str = "ô"; }
                else if(str == "O") { str = "Ô"; }
                else if(str == "u") { str = "û"; }
                else if(str == "U") { str = "Û"; }
                else if(str == " ") { str = "^"; }
                else { str = "^"+str; }
                break;
              case "`":
                if(str == "a") { str = "à"; }
                else if(str == "A") { str = "À"; }
                else if(str == "e") { str = "è"; }
                else if(str == "E") { str = "È"; }
                else if(str == "i") { str = "ì"; }
                else if(str == "I") { str = "Ì"; }
                else if(str == "o") { str = "ò"; }
                else if(str == "O") { str = "Ò"; }
                else if(str == "u") { str = "ù"; }
                else if(str == "U") { str = "Ù"; }
                else if(str == " ") { str = "`"; }
                else { str = "`"+str; }
                break;
            }
          }
          accent = null;
          return str;
        }

    ]]>
  </mx:Script>

  <mx:ViewStack id="keyboardMode" right="15" left="15" top="15" bottom="15">

    <mx:VBox id="lowerCase" horizontalAlign="center">
      <mx:HBox>
        <mx:Button label="'" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('\'')" fontSize="18"/>
        <mx:Button label="1" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('1')" fontSize="18"/>
        <mx:Button label="2" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('2')" fontSize="18"/>
        <mx:Button label="3" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('3')" fontSize="18"/>
        <mx:Button label="4" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('4')" fontSize="18"/>
```

```
            <mx:Button label="5" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('5')" fontSize="18"/>
            <mx:Button label="6" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('6')" fontSize="18"/>
            <mx:Button label="7" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('7')" fontSize="18"/>
            <mx:Button label="8" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('8')" fontSize="18"/>
            <mx:Button label="9" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('9')" fontSize="18"/>
            <mx:Button label="0" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('0')" fontSize="18"/>
            <mx:Button label="-" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('-')" fontSize="18"/>
        </mx:HBox>
        <mx:HBox>
            <mx:Button label="q" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('q')" fontSize="18"/>
            <mx:Button label="w" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('w')" fontSize="18"/>
            <mx:Button label="e" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('e')" fontSize="18"/>
            <mx:Button label="r" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('r')" fontSize="18"/>
            <mx:Button label="t" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('t')" fontSize="18"/>
            <mx:Button label="y" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('y')" fontSize="18"/>
            <mx:Button label="u" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('u')" fontSize="18"/>
            <mx:Button label="i" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('i')" fontSize="18"/>
            <mx:Button label="o" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('o')" fontSize="18"/>
            <mx:Button label="p" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('p')" fontSize="18"/>
            <mx:Button label="´" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('´')" fontSize="18"/>
            <mx:Button label="Backspace" width="{1.6 * buttonSize}"
height="{buttonSize}" touchTap="keyDown('BACKSPACE')" fontSize="18"/>
        </mx:HBox>
        <mx:HBox>
            <mx:Button label="CapsLock" width="{1.6 * buttonSize}"
height="{buttonSize}" touchTap="keyDown('CAPSLOCK')" fontSize="18"/>
            <mx:Button label="a" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('a')" fontSize="18"/>
            <mx:Button label="s" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('s')" fontSize="18"/>
            <mx:Button label="d" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('d')" fontSize="18"/>
            <mx:Button label="f" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('f')" fontSize="18"/>
            <mx:Button label="g" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('g')" fontSize="18"/>
            <mx:Button label="h" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('h')" fontSize="18"/>
```

```
            <mx:Button label="j" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('j')" fontSize="18"/>
            <mx:Button label="k" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('k')" fontSize="18"/>
            <mx:Button label="l" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('l')" fontSize="18"/>
            <mx:Button label="ç" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('ç')" fontSize="18"/>
            <mx:Button label="~" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('~')" fontSize="18"/>
            <mx:Button label="Enter" width="{1.8 * buttonSize}"
height="{buttonSize}" touchTap="keyDown('ENTER')" fontSize="18"/>
        </mx:HBox>
        <mx:HBox>
            <mx:Button label="Shift" width="{1.4 * buttonSize}"
height="{buttonSize}" touchTap="keyDown('SHIFT')" fontSize="18"/>
            <mx:Button label="\\" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('\\')" fontSize="18"/>
            <mx:Button label="z" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('z')" fontSize="18"/>
            <mx:Button label="x" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('x')" fontSize="18"/>
            <mx:Button label="c" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('c')" fontSize="18"/>
            <mx:Button label="v" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('v')" fontSize="18"/>
            <mx:Button label="b" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('b')" fontSize="18"/>
            <mx:Button label="n" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('n')" fontSize="18"/>
            <mx:Button label="m" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('m')" fontSize="18"/>
            <mx:Button label="," width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown(',')" fontSize="18"/>
            <mx:Button label="." width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('.')" fontSize="18"/>
            <mx:Button label=";" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown(';')" fontSize="18"/>
            <mx:Button label="/" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('/')" fontSize="18"/>
            <mx:Button label="Shift" width="{1.4 * buttonSize}"
height="{buttonSize}" touchTap="keyDown('SHIFT')" fontSize="18"/>
        </mx:HBox>
        <mx:HBox>
            <mx:Button label="Espaço" width="{10 * buttonSize}"
height="{buttonSize}" touchTap="keyDown('SPACE')" fontSize="18"/>
        </mx:HBox>
    </mx:VBox>

    <mx:VBox id="upperCase" horizontalAlign="center">
        <mx:HBox>
            <mx:Button label="&quot;" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('&quot;')" fontSize="18"/>
            <mx:Button label="!" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('!')" fontSize="18"/>
            <mx:Button label="@" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('@')" fontSize="18"/>
```

```
            <mx:Button label="#" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('#')" fontSize="18"/>
            <mx:Button label="$" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('$')" fontSize="18"/>
            <mx:Button label="%" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('%')" fontSize="18"/>
            <mx:Button label="¬" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('¬')" fontSize="18"/>
            <mx:Button label="&amp;" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('&amp;')" fontSize="18"/>
            <mx:Button label="*" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('*')" fontSize="18"/>
            <mx:Button label="(" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('(')" fontSize="18"/>
            <mx:Button label=")" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown(')')" fontSize="18"/>
            <mx:Button label="+" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('+')" fontSize="18"/>
        </mx:HBox>
        <mx:HBox>
            <mx:Button label="Q" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('Q')" fontSize="18"/>
            <mx:Button label="W" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('W')" fontSize="18"/>
            <mx:Button label="E" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('E')" fontSize="18"/>
            <mx:Button label="R" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('R')" fontSize="18"/>
            <mx:Button label="T" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('T')" fontSize="18"/>
            <mx:Button label="Y" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('Y')" fontSize="18"/>
            <mx:Button label="U" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('U')" fontSize="18"/>
            <mx:Button label="I" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('I')" fontSize="18"/>
            <mx:Button label="O" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('O')" fontSize="18"/>
            <mx:Button label="P" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('P')" fontSize="18"/>
            <mx:Button label="`" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('`')" fontSize="18"/>
            <mx:Button label="Backspace" width="{1.6 * buttonSize}"
height="{buttonSize}" touchTap="keyDown('BACKSPACE')" fontSize="18"/>
        </mx:HBox>
        <mx:HBox>
            <mx:Button label="CapsLock" width="{1.6 * buttonSize}"
height="{buttonSize}" touchTap="keyDown('CAPSLOCK')" fontSize="18"/>
            <mx:Button label="A" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('A')" fontSize="18"/>
            <mx:Button label="S" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('S')" fontSize="18"/>
            <mx:Button label="D" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('D')" fontSize="18"/>
            <mx:Button label="F" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('F')" fontSize="18"/>
```

```
            <mx:Button label="G" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('G')" fontSize="18"/>
            <mx:Button label="H" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('H')" fontSize="18"/>
            <mx:Button label="J" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('J')" fontSize="18"/>
            <mx:Button label="K" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('K')" fontSize="18"/>
            <mx:Button label="L" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('L')" fontSize="18"/>
            <mx:Button label="Ç" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('Ç')" fontSize="18"/>
            <mx:Button label="^" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('^')" fontSize="18"/>
            <mx:Button label="Enter" width="{1.8 * buttonSize}"
height="{buttonSize}" touchTap="keyDown('ENTER')" fontSize="18"/>
        </mx:HBox>
        <mx:HBox>
            <mx:Button label="Shift" width="{1.4 * buttonSize}"
height="{buttonSize}" touchTap="keyDown('SHIFT')" fontSize="18"/>
            <mx:Button label="|" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('|')" fontSize="18"/>
            <mx:Button label="Z" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('Z')" fontSize="18"/>
            <mx:Button label="X" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('X')" fontSize="18"/>
            <mx:Button label="C" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('C')" fontSize="18"/>
            <mx:Button label="V" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('V')" fontSize="18"/>
            <mx:Button label="B" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('B')" fontSize="18"/>
            <mx:Button label="N" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('N')" fontSize="18"/>
            <mx:Button label="M" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('M')" fontSize="18"/>
            <mx:Button label="&lt;" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('&lt;')" fontSize="18"/>
            <mx:Button label="&gt;" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('&gt;')" fontSize="18"/>
            <mx:Button label=":" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown(':')" fontSize="18"/>
            <mx:Button label="?" width="{buttonSize}" height="{buttonSize}"
touchTap="keyDown('?')" fontSize="18"/>
            <mx:Button label="Shift" width="{1.4 * buttonSize}"
height="{buttonSize}" touchTap="keyDown('SHIFT')" fontSize="18"/>
        </mx:HBox>
        <mx:HBox>
            <mx:Button label="Espaço" width="{10 * buttonSize}"
height="{buttonSize}" touchTap="keyDown('SPACE')" fontSize="18"/>
        </mx:HBox>
    </mx:VBox>

  </mx:ViewStack>

</mx:Canvas>
```

util.smoothscroller.ScrollerController.as

```
package util.smoothscroller
{
   import caurina.transitions.Tweener;

   import flash.utils.getTimer;
   import flash.utils.setTimeout;

   import mx.core.Container;
   import mx.effects.AnimateProperty;

   public class ScrollerController
   {
      public static const PIXELS_PER_MILISECOND : Number = 0.5;

      public var effect : AnimateProperty;

      protected var _interactionDelay:Number;
      [Bindable]
      public function set interactionDelay(value:Number):void
      {
         if (!isNaN(value) && value != _interactionDelay)
         {
            _interactionDelay = value;
         }
      }
      public function get interactionDelay():Number
      {
         return _interactionDelay;
      }

      protected var _scrollPosition:Number;
      public function set scrollPosition(value:Number):void
      {
         if (!isNaN(value) && value != _scrollPosition)
         {
            _scrollPosition = value;
            setTimeout(animateScrolling, _interactionDelay, value);
         }
      }

      public var viewContainer:Container;

      public var easing:String = "easeinoutsine";

      public var animatedProperty:String;

      public var maxDuration:Number = 2;

      protected function animateScrolling(position:Number):void
      {
         if (animatedProperty == null || viewContainer == null) return;

         var fromValue:Number = viewContainer[animatedProperty];
```

```
        var toValue:Number = position;
        var duration : Number = Math.min(
          0.001 * Math.abs(toValue - fromValue) / PIXELS_PER_MILISECOND,
          maxDuration);

        var tweenObject:Object = new Object();
        tweenObject[animatedProperty] = toValue;
        tweenObject["time"] = duration;
        tweenObject["transition"] = easing;

        if (Tweener.isTweening(viewContainer))
          Tweener.removeTweensByTime(viewContainer, tweenObject, getTimer(),
getTimer() + duration * 1000);
        Tweener.addTween(viewContainer, tweenObject);
      }

    }
}
```

util.smoothscroller.SmoothScroller.mxml

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Canvas xmlns:mx="http://www.adobe.com/2006/mxml"
        xmlns:smoothscroller="util.smoothscroller.*"
        verticalScrollPolicy="off"
        horizontalScrollPolicy="off"
        width="400"
        height="300"
        creationComplete="onCreationComplete()"
        touchBegin="startScroll(event)">

  <smoothscroller:ScrollerController interactionDelay="5"
                        easing="easeOutSine"
                        maxDuration="1"
                        animatedProperty="verticalScrollPosition"
                        scrollPosition="{_verticalScrollPosition}"
                        viewContainer="{this}"/>
  <smoothscroller:ScrollerController interactionDelay="5"
                        easing="easeOutSine"
                        maxDuration="1"
                        animatedProperty="horizontalScrollPosition"
                        scrollPosition="{_horizontalScrollPosition}"
                        viewContainer="{this}"/>

  <mx:Script>
    <![CDATA[
      import model.SystemModelLocator;

      import mx.core.UIComponent;

      [Bindable] public var _verticalScrollPosition:Number = 0;
      [Bindable] public var _horizontalScrollPosition:Number = 0;

      private var _child:UIComponent;
```

```
        private var _created:Boolean = false;
        private var _aux:Point = new Point();
        private var _startedHorizontalScroll:Boolean = false;
        private var _startedVerticalScroll:Boolean = false;

        private function onCreationComplete():void
        {
           _created = true;
           if (_child)
           {
              addChild(_child);
           }
        }

        private function drawInMask():void
        {
           var maskingShape:Shape = new Shape();
           maskingShape.cacheAsBitmap = true;
           cacheAsBitmap = true;

           var mat:Matrix = new Matrix();
           var colors:Array = [0xFFFFFF, 0xFFFFFF];
           var alphas:Array = [1, 0];
           var ratios:Array = [200, 255];
           mat.createGradientBox(width, height);
           maskingShape.graphics.lineStyle();
           maskingShape.graphics.beginGradientFill(GradientType.RADIAL, colors,
alphas, ratios, mat);
           maskingShape.graphics.drawRect(0, 0, width, height);
           maskingShape.graphics.endFill();

           addChild(maskingShape);
           this.mask = maskingShape;
        }

        public function set child(value:UIComponent):void
        {
           if (_created && _child != value)
           {
              removeAllChildren();
              addChild(value);
           }
           _child = value;
        }

        public function get child():UIComponent
        {
           return _child;
        }

        private function startScroll(event:TouchEvent):void
        {
           if (!_child)
              return;

           _aux.x = event.stageX;
           _aux.y = event.stageY;
```

```
            stage.addEventListener(TouchEvent.TOUCH_MOVE, doScroll);
            stage.addEventListener(TouchEvent.TOUCH_END, endScroll);
        }

        private function doScroll(event:TouchEvent):void
        {
            var dx:Number = event.stageX - _aux.x;
            var dy:Number = event.stageY - _aux.y;
            trace("(" + dx + "," + dy + ")");

            if (!_startedVerticalScroll && (Math.abs(dx) > 10 ||
_startedHorizontalScroll))
            {
                _startedHorizontalScroll = true;
                var newX:Number = Math.min(_horizontalScrollPosition - dx,
_child.width - width);
                newX = Math.max(newX, 0);
                _horizontalScrollPosition = newX;
            }

            if (!_startedHorizontalScroll && (Math.abs(dy) > 10 ||
_startedVerticalScroll))
            {
                _startedVerticalScroll = true;
                var newY:Number = Math.min(_verticalScrollPosition - dy,
_child.height - height);
                newY = Math.max(newY, 0);
                _verticalScrollPosition = newY;
            }

            if (_startedHorizontalScroll)
            {
                SystemModelLocator.isScrolling = true;
                _aux.x = event.stageX;
            }
            else if(_startedVerticalScroll)
            {
                SystemModelLocator.isScrolling = true;
                _aux.y = event.stageY;
            }
        }

        private function endScroll(event:TouchEvent):void
        {
            SystemModelLocator.isScrolling = false;
            _startedHorizontalScroll = false;
            _startedVerticalScroll = false;
            stage.removeEventListener(TouchEvent.TOUCH_MOVE, doScroll);
            stage.removeEventListener(TouchEvent.TOUCH_END, endScroll);
        }
    ]]>
  </mx:Script>

</mx:Canvas>
```