Paulo João Rodrigues Neto

IMPLEMENTAÇÃO DE UM MÓDULO PARA INTEGRAÇÃO DA ABORDAGEM USE-CASE 2.0 COM UM SISTEMA DE GERENCIAMENTO DE PROJETOS

Trabalho de Conclusão de Curso em Ciências da Computação do Centro Tecnológico da Universidade Federal de Santa Catarina como requisito para a obtenção do Título de Bacharel em Ciências da Computação
Orientador: Prof. Dr. Jean Carlo Rossa Hauck
Coorientador: Msc. João Marcus Alves

Florianópolis
2017

Ficha de identificação da obra elaborada pelo autor através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Rodrigues Neto, Paulo João
Implementação de um módulo para integração da abordagem Use-Case 2.0 com um sistema de gerenciamento de projetos / Paulo João Rodrigues Neto ; orientador, Jean Carlo Rossa Hauck, coorientador, João Marcus Alves, 2018.
142 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Ciências da Computação, Florianópolis, 2018.

Inclui referências.

1. Ciências da Computação. 2. Use-Case 2.0. 3. Caso de uso. 4. Gerência. I. Hauck, Jean Carlo Rossa . II. Alves, João Marcus. III. Universidade Federal de Santa Catarina. Graduação em Ciências da Computação. IV. Título.

Paulo João Rodrigues Neto

FERRAMENTA PARA INTEGRAÇÃO DA ABORDAGEM USE-CASE 2.0 A UM SISTEMA DE GERENCIAMENTO DE PROJETOS

Este Trabalho Conclusão de Curso foi julgado adequado para obtenção do Título de "Bacharel em Ciências da Computação" e aprovado em sua forma final pelo curso de Ciências da Computação

Florianópolis, 19 de junho de 2018.

_____
Prof. Rafael Luiz Cancian, Dr.
Coordenador do Curso

Banca Examinadora:

_____
Prof. Jean Carlo Rossa Hauck, Dr.
Orientador
Universidade Federal de Santa Catarina

_____
João Marcus Alves, Msc.
Coorientador
Universidade Federal de Santa Catarina

_____
Prof. Alexandre Savaris, Dr.
Universidade Federal de Santa Catarina

Aos meus pais, Maria de Fátima e Gelson, por acreditarem no meu potencial e por me apoiarem incondicionalmente.

AGRADECIMENTOS

Ao meu orientador, Jean Carlo Hauck, por aceitar meu pedido de orientação e também por estar sempre disponível para dúvidas e correções.

Ao meu chefe, Alexandre Savaris, por aceitar fazer parte da banca avaliadora e por todos os ensinamentos ao longo dos anos em que fiz estágio no Laboratório de Telemedicina.

Aos colegas do Laboratório de Telemedicina por avaliarem a ferramenta por mim desenvolvida.

À Camila dos Reis, amiga e parceira de estágio, por todo o incentivo nos últimos anos.

A João Marcus Alves, amigo que, além coorientar este trabalho, esteve presente em todos os momentos, inclusive quando eu nem sabia por onde começar, pelas excelentes dicas, desde o projeto inicial até os instantes que precederam a defesa deste trabalho de conclusão de curso.

A Bruno Schwantes, pela grande contribuição na formatação e revisão desta “belezinha”, mas principalmente pelos bons conselhos e pela amizade de mais de vinte anos.

À minha namorada Francini de Rezende Madeira, que esteve ao meu lado durante toda a produção deste trabalho, por todo o amor, companheirismo e suporte emocional.

"*Yes, I get by with a little help from my friends*" (The Beatles, 1967).

RESUMO

Há um problema recorrente na indústria de desenvolvimento de *software*. De forma geral, as empresas desenvolvedoras de *software* têm dificuldade para entregar ou concluir o produto idealizado pelo cliente. Esse problema ocorre em várias fases da gerência de projeto e de desenvolvimento de *software*. Um dos principais motivos é relacionado à engenharia de requisitos, que desencadeia alterações, afetando custos e prazos. Muitas abordagens clássicas de engenharia de requisitos lidam com essa complexidade, como a abordagem de casos de uso. As práticas ágeis e iterativas possibilitam às empresas desenvolvedoras adaptar-se às mudanças requisitadas pelo cliente ou identificar requisitos não compreendidos com rapidez. Uma abordagem utilizada atualmente para a coleta de requisitos é a *Use-Case* 2.0, que estende os casos de uso clássicos para um contexto ágil e iterativo. O Kanboard é uma ferramenta que implementa o Kanban, uma abordagem para a gerência de atividades que segue valores ágeis complementares ao *Use-Case* 2.0, sendo ágil e iterativo. Analisando este cenário, é proposto neste trabalho o desenvolvimento de um módulo que modele o *Use-Case* 2.0 e o integre com a plataforma de gerência de projeto Kanboard. A ferramenta é implementada e avaliada em duas organizações de desenvolvimento de *software*. Os resultados das avaliações levantam os primeiros indícios de que a ferramenta tem boa usabilidade e está alinhada à abordagem *Use-Case* 2.0, não necessitando de esforço relevante para sua adoção.

Palavras-chave: Software. Requisitos. Gerência. Projeto. Caso de uso. *Use-Case* 2.0. Kanban. Kanboard. Ágil. Iterativo. Evolutivo.

ABSTRACT

There is a recurring problem in the software development industry. In general, software development companies have difficulty in delivering or completing the scope of the product idealized by the customer. This problem occurs at various stages of project management and software development. One of the main reasons is related to requirements engineering, affecting costs and deadlines. Many classical requirements engineering approaches deal with this complexity, such as the use-case approach. Agile and iterative practices enable developers to adapt to the changes required by the customer or identify requirements that are not understood quickly. One approach currently used for requirements collection is *Use-Case* 2.0, which extends classic use cases to an agile and iterative context. Kanboard is a tool that implements Kanban, an approach to activity management that follows agile values complementary to *Use-Case* 2.0, being agile and iterative. Analyzing this scenario, it is proposed in this work the development of a module that models Use-Case 2.0 and integrates it with the Kanboard project management platform. The tool is implemented and evaluated in two software development organizations. The results of the evaluations identify the first indications that the tool has good usability and is aligned with the *Use-Case* 2.0 approach, requiring no relevant effort for its adoption.

Keywords: Software. Requirements. Management. Project. Use case. *Use-Case* 2.0. Kanban. Kanboard. Agile. Iterative. Evolutionary.

LISTA DE FIGURAS

Figura 1: Caso de uso clássico 33
Figura 2: Quadro Kanban físico 38
Figura 3: Quadro Kanban virtual..... 40
Figura 4: Modelagem de caso de uso com GenMyModel. 34
Figura 5: Modelagem de caso de uso com Lucidchart 34
Figura 6: Modelagem de caso de uso com Visual Paradigm..... 35
Figura 7: Modelagem de caso de uso com StarUML. 36
Figura 8: Modelagem de caso de uso com DrawIO 36
Figura 9: Ferramentas para integração com Lucidchart. 38
Figura 10: Casos de uso do Agrupamento 1..... 43
Figura 11: Casos de uso do Agrupamento 2..... 44
Figura 12: Protótipo da tela de criação de casos de uso. 50
Figura 13: Protótipo da tela de criação de um *slice*..... 51
Figura 14: Protótipo do diagrama de casos de uso. 52
Figura 15: Interação entre protótipos. 53
Figura 16: Diagrama de componentes..... 54
Figura 17: *Plug-ins* desenvolvidos para a ferramenta Kanboard 57
Figura 18: Criação do ator..... 61
Figura 19: Tela de criação de um caso de uso..... 62
Figura 20: Opção para criação do *slice* dentro do caso de uso..... 62
Figura 21: Tela de criação de um *slice*. 63
Figura 22: Diagrama de casos de uso implementado no Kanboard 64
Figura 23: Quadro Kanban com instâncias do *Use-Case* 2.0 64
Figura 24: Escala SUS..... 68

LISTA DE QUADROS

Quadro 1: Características para a implementação do *Use-Case* 2.0. 28
Quadro 2: Características em ferramentas que implementam *Use-Case* 2.0. 28
Quadro 3: Termos de busca. 29
Quadro 4: *Strings* de busca para *Use-Case* 2.0. 30
Quadro 5: Resultados da busca para *Use-Case* 2.0 31
Quadro 6: *Strings* de busca para casos de uso. 31
Quadro 7: Resultados da busca para os casos de uso. 32
Quadro 8: Referências relevantes encontradas na pesquisa. 33
Quadro 9: Características das ferramentas analisadas. 39
Quadro 10: Requisitos funcionais. 41
Quadro 11: Requisitos não funcionais 42
Quadro 12: *Unadjusted Use Case Point* para ator 46
Quadro 13: *Unadjusted Use Case Point* para casos de uso. 46
Quadro 14: Cálculo UUCP 47
Quadro 15: Fatores de complexidade técnica. 47
Quadro 16: Fatores de ambiente 48
Quadro 17: Tecnologias utilizadas no desenvolvimento. 58
Quadro 18: Comparação de funcionalidades 65
Quadro 19: Itens SUS 68
Quadro 20: Questões específicas. 69
Quadro 21: Escala valorada 73
Quadro 22: Cálculo SUS. 74
Quadro 23: Tempos observados. 75

LISTA DE ABREVIATURAS E SIGLAS

ACM – Association for Computing Machinery
CAPES – Comissão de Aperfeiçoamento de Pessoal do nível Superior
DDL – Data Definition Language
GUI – Graphical User Interface
IBM – International Business Machines
IEEE – Institute of Electrical and Electronics Engineers
INCoD – Instituto Nacional para Convergência Digital
JSON – JavaScript Object Notation
MSL – Mapeamento Sistemático da Literatura
PHP – PHP Hypertext Preprocessor
SQL – Structured Query Language
TFD – Tratamento Fora de Domicílio
UFSC – Universidade Federal de Santa Catarina
UML – Unified Modeling Language

SUMÁRIO

1 INTRODUÇÃO 27
1.1 OBJETIVOS 28
1.1.1 Objetivo geral 28
1.1.2 Objetivos específicos 28
1.2 MÉTODO DE PESQUISA 29
1.3 ESTRUTURA DO TRABALHO 30
2 FUNDAMENTAÇÃO TEÓRICA 31
2.1 CASOS DE USO 31
2.2 *USE-CASE* 2.0 33
2.2.1 Casos de uso 35
2.2.2 *Slices* 35
2.2.3 *Stories* 36
2.3 KANBAN 37
2.4 UMA FERRAMENTA KANBAN WEB 39
3 ESTADO DA ARTE 27
3.1 DEFINIÇÃO DAS FONTES DE PESQUISA 27
3.2 QUESTÕES DA PESQUISA 28
3.3 TERMOS E *STRINGS* DE BUSCA 29
3.4 EXECUÇÃO DA PESQUISA E RESULTADOS 30
3.5 DISCUSSÃO 39
4 MODELAGEM DA SOLUÇÃO 41
4.1 ENGENHARIA DE REQUISITOS 41
4.1.1 Requisitos funcionais 41
4.1.2 Requisitos não funcionais 42
4.1.3 Modelagem do Sistema em *Use-Case* 2.0 42
4.2 ESTIMATIVA DOS PONTOS DE CASOS DE USO 45
4.3 PROTOTIPAÇÃO 49

4.3.1 AG01-UC04: criar caso de uso 50
4.3.2 AG01-UC07: criar um *slice* 51
4.3.3 AG02-UC01: visualizar o diagrama de casos de uso de um projeto 52
4.4 MODELAGEM DOS COMPONENTES 54
5 DESENVOLVIMENTO 57
5.1 DIFICULDADES NA IMPLEMENTAÇÃO 57
5.2 TECNOLOGIAS 58
5.3 IMPLEMENTAÇÃO 60
5.3.1 Tela de manipulação do ator 60
5.3.2 Tela de criação de casos de uso 61
5.3.3 Tela de criação do *slice* 62
5.3.4 Tela de visualização do diagrama de casos de uso 63
5.4 COMPARAÇÃO DE FUNCIONALIDADES DAS FERRAMENTAS 65
6 AVALIAÇÃO 67
6.1 PLANEJAMENTO DA AVALIAÇÃO 67
6.1.1 Avaliação da usabilidade com o método SUS (*System Usability Scale*) 67
6.1.2 Avaliação da aplicação de *Use-Case* 2.0 69
6.2 EXECUÇÃO DA AVALIAÇÃO 70
6.3 ANÁLISE DOS DADOS 71
6.3.1 Análise da avaliação de usabilidade 71
6.3.2 Análise da avaliação da aplicação de *Use-Case* 2.0 74
6.4 DISCUSSÃO 77
6.5 AMEAÇAS A VALIDADE 77
7 CONCLUSÃO 79
REFERÊNCIAS 81
APÊNDICE I – Detalhamento dos casos de uso 85

APÊNDICE II – Tutorial para utilizar o Kanboard com *Use-Case* 2.0 93

APÊNDICE III – Respostas das questões auxiliares 97

APÊNDICE IV – Gráficos com os resultados da avaliação por item SUS 99

1 INTRODUÇÃO

Aproximadamente 70% dos produtos de *software* falham, e um dos principais motivos é a baixa qualidade na engenharia de requisitos (KAUR; SENGUPTA, 2011). Requisito, segundo a *Institute of Electrical and Electronics Engineers*, é definido como:

> *"Um requisito é uma declaração que traduz ou expressa uma necessidade e suas restrições e condições associadas"* (IEEE-29148, 2011, p. 5)

Nesse contexto, uma engenharia de requisitos de qualidade é necessária para minimizar problemas que implicam em consequências indesejadas no desenvolvimento de *software*. Assim, por meio de atividades como: elicitação, desenvolvimento, análise, verificação, validação, comunicação, documentação e gerência de requisitos, uma hierarquia de requisitos é gerada (IEEE-29148, 2011). Esta hierarquia de requisitos possibilita a validação e a compreensão dos requisitos entre os *stakeholders* — indivíduos ou organizações com interesse em um sistema de acordo com suas necessidades e expectativas. A validação dos requisitos em relação às necessidades reais dos *stakeholders* reforça a compreensão dos requisitos e fornece uma base que facilita o projeto do sistema (IEEE-29148, 2011).

Existem diversas abordagens e técnicas que auxiliam a engenharia de requisitos. Dentre elas, a abordagem de casos de uso é utilizada para a coleta, modelagem e gerenciamento dos requisitos dos usuários (PRESSMAN, 2007). Nesse contexto, um caso de uso representa todos os modos de utilização do sistema para atingir os objetivos específicos de um determinado ator, portanto, o conjunto dos casos de uso abrange todos os caminhos possíveis para a utilização do sistema e o valor que esse sistema fornecerá (JACOBSON; SPENCE; BITTNER, 2011).

Uma modernização da abordagem dos casos de uso, chamada de *Use-Case 2.0*, surgiu grande parte em resposta à necessidade de envolver práticas ágeis na engenharia de requisitos. Com o advento das abordagens ágeis na engenharia de *software*, a forma de coletar, modelar, analisar e gerenciar requisitos tem mudado, focando em *software* eficiente no lugar de uma documentação extensa, colaboração com o cliente e resposta a mudanças (FOWLER; HIGHSMITH, 2001). O *Use-Case 2.0* é uma prática escalável, que usa casos de uso para capturar um conjunto de requisitos, guia o desenvolvimento incremental do sistema, não necessita de uma documentação preditiva extensa e enquadra-se perfeitamente na mentalidade ágil (JACOBSON; SPENCE; BITTNER, 2011).

Essa mentalidade ágil e incremental vem sendo aplicada na gerência de projetos do laboratório de Telemedicina[1] do Instituto Nacional para Convergência Digital (INCoD[2]) vinculado à Universidade Federal de Santa Catarina (UFSC). O Laboratório de Telemedicina, através do Sistema Integrado Catarinense de Telemedicina e Telessaúde[3], encurta a distância entre pacientes e provedores de saúde. Desde 2005, esse sistema já emitiu mais de 5 milhões de laudos de inúmeras modalidades de exames.

O sistema de gerenciamento de projeto Kanban (BOEG, 2012) vem sendo adotado no contexto do laboratório de Telemedicina e tem proporcionado melhoria significativa nos prazos de entregas das tarefas do laboratório e espera-se que a adição do *Use-Case 2.0*, em conjunto com essas práticas, melhore a coleta, a análise e a manutenção dos requisitos.

Atualmente existem muitas ferramentas para plataformas *desktop* e *web* que permitem a criação de casos de uso na sua forma clássica. Contudo, a despeito da importância de uma eficaz engenharia de requisitos e dos benefícios das práticas ágeis, identifica-se uma lacuna em relação a ferramentas que integrem o uso do *Use-Case 2.0* com um sistema de gerência de projeto.

1.1 OBJETIVOS

1.1.1 Objetivo geral

O objetivo geral deste trabalho é desenvolver e avaliar uma ferramenta que permita a modelagem de casos de uso seguindo a abordagem *Use-Case 2.0*. A ferramenta será integrada a um sistema de gerenciamento de projetos para auxiliar a utilização de práticas ágeis, escaláveis e incrementais.

1.1.2 Objetivos específicos

Os objetivos específicos do trabalho são:

O1: Fundamentação teórica sobre casos de uso e a abordagem *Use-Case 2.0*.

[1] http://site.telemedicina.ufsc.br/
[2] http://www.incod.ufsc.br/
[3] https://telemedicina.saude.sc.gov.br/rctm/

O2: Analisar o estado da arte em relação a ferramentas que implementem a abordagem *Use-Case 2.0*.
O3: Modelar uma ferramenta *web* — fundamentado no estado da arte — que empregue a abordagem *Use-case 2.0* para auxiliar a engenharia de requisitos de acordo com as práticas ágeis.
O4: Implementar e testar a ferramenta modelada em O3.
O5: Avaliar a ferramenta desenvolvida em O4 em um contexto real de aplicação.

1.2 MÉTODO DE PESQUISA

Este trabalho é classificado como uma pesquisa aplicada, pois gera conhecimento para a solução de problemas práticos específicos (GERHARDT, SILVEIRA, 2009). As etapas para os objetivos do presente trabalho são:

Etapa 1: Fundamentação teórica: Nesta etapa é realizada a pesquisa e análise da fundamentação teórica.

Atividade 1.1: Pesquisa bibliográfica.
Atividade 1.2: Análise da literatura encontrada na pesquisa bibliográfica.
Atividade 1.3: Descrição do conteúdo analisado.

Etapa 2: Mapeamento sistemático da literatura: Direcionado pelo procedimento proposto por Kitchenham (2004), será realizado um estudo para identificar e analisar trabalhos similares que contenham evidências relevantes disponíveis.

Atividade 2.1: Definição das questões de busca.
Atividade 2.2: Definição das ferramentas e termos de busca.
Atividade 2.3: Execução da busca.
Atividade 2.4: Extração e análise das informações.

Etapa 3: Modelagem e desenvolvimento da ferramenta: Os estudos realizados nas etapas anteriores direcionarão o desenvolvimento da ferramenta para sistemas *web* com a finalidade de integrar a abordagem *Use-Case 2.0* a um sistema de gerenciamento de projetos.

Atividade 3.1: Análise de requisitos.
Atividade 3.2: Modelagem da ferramenta proposta.

Atividade 3.3: Desenvolvimento da ferramenta proposta.
Atividade 3.4: Testes sobre a ferramenta proposta.

Etapa 4: Integração e validação da ferramenta proposta.

Atividade 4.1: Integração da ferramenta proposta com o sistema de gerenciamento de projetos.
Atividade 4.2: Testes e validação sobre a integração da ferramenta proposta.

Etapa 5: Aplicação e avaliação da ferramenta desenvolvida. Nesta etapa, avalia-se a utilidade, completude, consistência e usabilidade da ferramenta desenvolvida.

Atividade 5.1: Definir objetivo da avaliação
Atividade 5.2: Planejar a avaliação.
Atividade 5.3: Executar a avaliação.
Atividade 5.4: Análise dos resultados.

1.3 ESTRUTURA DO TRABALHO

O trabalho está organizado como segue: o Capítulo 2 descreve a fundamentação teórica e define conceitos necessários para a melhor compreensão deste trabalho. O Capítulo 3 apresenta um mapeamento sistemático do estado da arte sobre ferramentas de modelagem usando a abordagem *Use-Case* 2.0. A modelagem da solução proposta é descrita no Capítulo 4. No Capítulo 5 é descrito como foi realizado o desenvolvimento da solução proposta. No Capítulo 6 a implementação do módulo é avaliada e no Capítulo 7 a conclusão sobre o trabalho proposto é explicitado.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os fundamentos essenciais para a compreensão deste trabalho de conclusão de curso.

2.1 CASOS DE USO

A história dos casos de uso inicia-se nos anos 60, criada por Ivar Jacobson enquanto trabalhava no sistema de telefonia da Ericsson. Nos anos 80 os casos de uso foram apresentados à comunidade de programação orientada a objetos, onde foi reconhecida como uma técnica importante no levantamento de requisitos. Em 1994, Cockburn formulou os modelos conceituais de atores e objetivos enquanto escrevia guias de casos de uso para a IBM, enriquecendo a ferramenta casos de uso (COCKBURN, 2001). E em 1996 os casos de uso são incorporados a UML (*Unified Modeling Language*) — linguagem padronizada para a elaboração da estrutura de projetos de *software* (BOOCH, 1996).

Avaliando mais precisamente, POHL (2016) define um requisito de *software* de duas formas: uma condição ou capacidade de um usuário resolver um problema ou alcançar um objetivo; ou uma condição que precisa ser atingida, ou possuída, por um sistema (ou módulo do sistema), para satisfazer um contrato, padrão, especificação ou outro documento formal. Consequentemente, POHL (2016) apresenta o processo de engenharia de requisitos composto de 4 atividades fundamentais:

- Elicitação: diferentes técnicas usadas para obter os requisitos de *stakeholders* e de outras fontes;
- Documentação: descrição adequada dos requisitos obtidos na atividade de elicitação. Diferentes técnicas podem ser usadas para documentar os requisitos, usando linguagem natural ou modelos conceituais.
- Validação e negociação: para garantir coerência, os requisitos devem ser negociados e validados pelos *stakeholders*.
- Gerenciamento de requisitos: atividades necessárias para estruturar requisitos, prepará-los para que possam ser compreendidos por diferentes pessoas com diferentes responsabilidades, e principalmente manter consistência depois de mudanças para garantir a implementação adequada.

Requisitos de *software* licitados podem ser documentados de diversas formas como, histórias de usuário ou casos de uso por exemplo (WIEGERS, BEATTY, 2013). Nesse cenário, uma técnica importante

para a análise de requisitos que tem sido largamente aplicada no desenvolvimento de *software*, são os casos de uso.

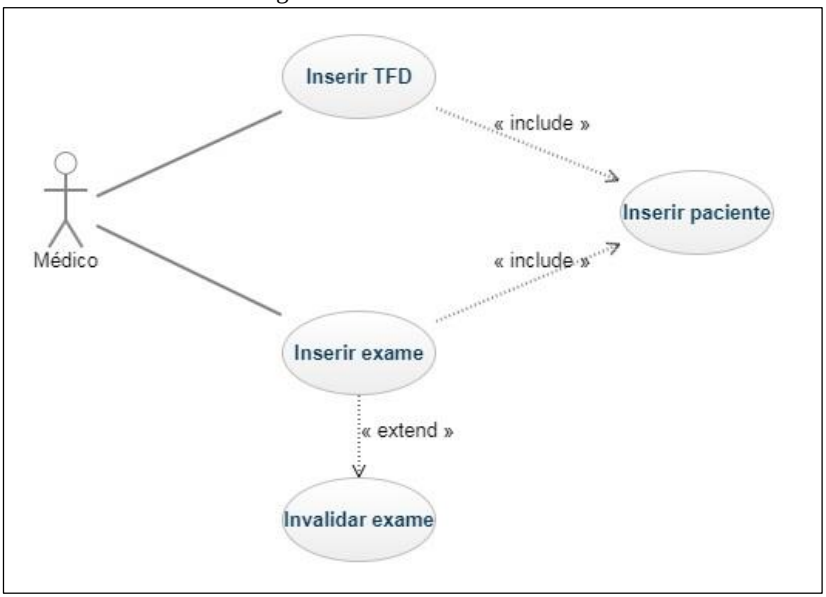
Dentro da elicitação, uma das 4 atividades fundamentais da engenharia de requisitos, os requisitos podem ser obtidos através de *stories* — narrativa que apresenta formas de atingir um objetivo e lidar com os problemas que podem ocorrer no caminho — e inseridas nas narrativas de caso de uso para completar a descrição dos casos de uso, tornando os requisitos acessíveis e testáveis (JACOBSON; SPENCE; BITTNER, 2011).

O conjunto de casos de uso forma o modelo de casos de uso, que compreende o contexto para que os requisitos do sistema sejam descobertos, compartilhados e compreendidos. No modelo de casos de uso, os *stakeholders* que usam o sistema e contribuem para completar os objetivos são modelados como atores, e os caminhos para alcançar esses objetivos são modelados como casos de uso (JACOBSON; SPENCE; BITTNER, 2011).

Atualmente não há um consenso da comunidade acadêmica e na indústria para o formato correto do uso dos casos de uso. Consequentemente, diferentes propósitos acarretam em diferentes modelos.

Classicamente um caso de uso deve descrever um fluxo principal de atividades de um ator, baseado em um conjunto de premissas, possíveis fluxos alternativos e pós-condições esperadas após a conclusão do caso de uso (COCKBURN, 2001). Uma ferramenta gráfica para ilustrar casos de uso é o diagrama UML de casos de uso. Neste diagrama, um ator é relacionado a um conjunto de casos de uso que pode incluir ainda mais informações com o uso de notações UML. Por exemplo, a Figura 1 mostra a notação *include*, que implica que o caso de uso “Inserir TFD” (Tratamento Fora de Domicílio) e “Inserir exame” tem um fluxo de atividades em comum, que é representado no caso de uso “Inserir paciente”. Da mesma maneira, há um fluxo de atividades opcionais em “Inserir exame”, que é representado no caso de uso “Invalidar exame”.

Figura 1: Caso de uso clássico



Fonte: autor

2.2 *USE-CASE* 2.0

Em 2011, Jacobson, Spence e Bittner apresentam uma extensão da técnica de casos de uso, chamada *Use-Case* 2.0, e a define como uma prática leve, escalável, ágil, versátil e fácil de usar. Essa abordagem utiliza-se dos casos de uso para capturar um conjunto de requisitos e direcionar o desenvolvimento incremental de um sistema sendo bastante abrangente e podendo ser utilizada não somente em pequenas equipes, mas também em grandes equipes com desenvolvimento de sistemas complexos (JACOBSON; SPENCE; BITTNER, 2011).

Como o *Use-Case* 2.0 pode ser utilizado no desenvolvimento de um sistema, as fases do desenvolvimento como análise, *design*, planejamento, estimativa, rastreamento e teste são suportadas pela abordagem. O *Use-Case* 2.0 não prescreve como planejar, gerenciar e desenvolver um sistema, mas provê uma estrutura para auxiliar nas práticas de gerenciamento e de desenvolvimento selecionadas (JACOBSON; SPENCE; BITTNER, 2011).

Para a aplicação do *Use-Case* 2.0 com sucesso, 6 princípios são utilizados:

1. Simplificar através de *stories*: as *stories* são utilizadas para descrever os requisitos, englobando como alcançar o objetivo com sucesso e como lidar com problemas que podem ocorrer. O resultado das *stories* está presente como parte da narrativa dos casos de uso que acompanha cada caso de uso;
2. Conhecer o todo: sem entender completamente o sistema não é possível tomar as decisões corretas sobre o que incluir no sistema, o que retirar, seu custo e seus benefícios. O modelo de casos de uso dá a visualização do sistema como um todo e o diagrama de casos de uso pode auxiliar na parte visual para o entendimento do modelo de casos de uso;
3. Foco no valor: priorizar um objetivo específico de um ator específico é importante para obter o fluxo principal que conterá o principal caminho para se alcançar o objetivo. Através das *stories* vários fluxos são identificados, dentre eles o principal;
4. Construir o sistema em *slices*: construir um sistema por inteiro aumenta as chances de erro, para isso o sistema deve ser dividido em *slices* — pedaços menores de um caso de uso;
5. Entregar o sistema em incrementos: auxilia na evolução do sistema, na qual cada incremento adiciona novas funcionalidades ou melhorias;
6. Adaptar para se enquadrar nas necessidades da equipe: diferentes times e diferentes situações requerem diferentes níveis de detalhe.

Para aprofundar o conhecimento na abordagem *Use-Case* 2.0 é importante compreender quais itens são trabalhados, como a abordagem trabalha estes itens e o papel de todas as partes que compõem esta abordagem. Os itens a serem trabalhados pelo *Use-Case* 2.0 são os requisitos, o sistema a ser desenvolvido de acordo com os requisitos e os testes para demonstrar que o sistema está de acordo com os requisitos. Esses itens citados acima são capturados pelos casos de uso, as *stories* e os *slices*, que são o coração do *Use-Case* 2.0 (JACOBSON; SPENCE; BITTNER, 2011).

Os requisitos são essenciais pois definem as funcionalidades que um sistema deve ter. Os casos de uso capturam os requisitos através das *stories*, que são contadas pelos *stakeholders*, e auxiliam a identificar os *slices* e os testes. Os testes por sua vez, verificam a qualidade e completude da implementação dos *slices* e consequentemente do sistema, cujo o escopo e objetivos são modelados pelos casos de uso.

O *Use-Case* 2.0 prevê e suporta, através de novas *stories*, a adição ou mudanças nas funcionalidades do sistema, que poderão gerar novos casos de uso e/ou *slices* (JACOBSON; SPENCE; BITTNER, 2011). As

partes que compõem a estrutura interna do *Use-Case* 2.0 são importantes para a compreensão do escopo deste trabalho.

2.2.1 Casos de uso

No contexto do *Use-Case* 2.0, os casos de uso são divididos em estados desde a identificação inicial até o cumprimento total dos requisitos pelo sistema.

- Objetivo estabelecido: quando o objetivo do caso de uso foi estabelecido.
- Compreensão da estrutura das *stories*: as estruturas das narrativas de caso de uso são compreendidas e os primeiros *slices* são identificados e implementados.
- *Stories* mais simples cumprida: quando o sistema cumpre as narrativas mais simples que permitem que um usuário alcance o objetivo
- *Stories* suficientes cumpridas: o sistema cumpriu *stories* suficiente para prover uma solução utilizável.
- Todas as *stories* cumpridas: todas as *stories* compreendidas pela narrativa de caso de uso foram cumpridas, ou seja, fluxos básicos e alternativos completados.

Conforme o sistema for implementado através dos *slices*, os estados do caso de uso vão sendo percorridos, estimando o progresso do sistema (JACOBSON; SPENCE; BITTNER, 2011).

2.2.2 *Slices*

Os *slices* possibilitam a divisão dos casos de uso, permitindo a seleção dos pedaços a serem trabalhados e provendo unidades menores aproximadamente do mesmo tamanho. Como os *slices* são geralmente pequenos e acabam fluindo rápido, o desenvolvimento e o teste tornam-se ágeis. Com isso os *slices* tem uma maior vazão a cada iteração do sistema possibilitando um panorama mais preciso do estado de desenvolvimento do sistema como um todo.

Os *slices* possuem estados que organizam as etapas de trabalho.

- Escopo: quando as *stories* são esclarecidas o escopo consegue ser delimitado.

- Pronto: com o escopo delimitado, há a possibilidade de melhoria na narrativa e nos casos de testes, orientando como implementar com sucesso o *slice*.
- Análise: verifica o impacto sobre os componentes do sistema e qual decisão tomar sobre as partes afetadas para disponibilizá-las para codificação.
- Implementação: o *slice* está pronto para ser implementado e posteriormente testado.
- Verificação: o *slice* foi verificado como finalizado e está pronto para a inclusão no versionamento.

Os estados fornecem uma maneira simples de avaliar o progresso do ciclo de vida de um *slice* de um caso de uso (JACOBSON; SPENCE; BITTNER, 2011).

2.2.3 *Stories*

As *stories* são a forma de explorar os casos de uso com o auxílio dos *stakeholders*, auxiliando na identificação dos *slices* e dos casos de teste. Um caso de uso pode ser decomposto em várias *stories*. Cada uma com sua importância para atores e outros *stakeholders*. A *story* por sua vez é descrita como parte da narrativa de caso de uso. Um ou mais fluxos formam uma rede que pode ser entendida como o resumo de todas as *stories* necessárias para descrever um caso de uso.

Há duas formas de identificar as *stories* e criar as narrativas de caso de uso.

A abordagem *top-down* inicia identificando os casos de uso, esboçando os passos do fluxo básico e verificando, através de sugestões ou *brainstorms*, os fluxos alternativos.

A abordagem *bottom-up* inicia com sugestões ou *brainstorms* para identificar *stories*, separando-as por tema para identificar os casos de uso. Em seguida as *stories* são analisadas para tentar identificar o fluxo básico e alguns fluxos alternativos. Conforme a estrutura de casos de uso for sendo completada, mais *stories* e fluxos alternativos podem ser identificados.

Com a descrição das partes mais importantes do *Use-Case* 2.0, principalmente dos *slices*, verifica-se a proximidade com técnicas de gerenciamento iterativas e incrementais (JACOBSON; SPENCE; BITTNER, 2011).

2.3 KANBAN

No ano de 1991, o termo desenvolvimento ágil de *software* foi popularizado pelo Manifesto Ágil. Este manifesto definiu formalmente princípios e valores que alteraram drasticamente a abordagem clássica e preditiva de desenvolvimento de *software* (FOWLER; HIGHSMITH, 2001). Devido à natureza iterativa, incremental e evolucionária, o *Use-Case* 2.0 enquadra-se perfeitamente no contexto de desenvolvimento ágil de *software*.

A metodologia de visualização de fluxo de trabalho do Kanban foi adaptada da manufatura utilizada pela companhia japonesa Toyota (SUGIMORI, et al., 1977). Um indicador de sucesso nos anos 70 no Japão, no contexto da produção baseado em demanda, é a habilidade de prever a demanda. Nesse contexto o "toyotismo" inovou com o Kanban ao utilizar a demanda real observada (OHNO, 1988).

O Kanban é uma abordagem que usa um quadro físico para visualizar tarefas. Dessa forma melhora-se a compreensão e o fluxo de trabalho. A metodologia Kanban também sugere a limitação no progresso de trabalho, assim reduz desperdício de tempo e esforço devido a multitarefas e mudanças de contexto, estimulando a colaboração para melhorar o sistema (BOEG, 2012). O método não sugere um número de passos ou procedimentos, contudo estimula mudanças contínuas, incrementais e evolucionárias no sistema. Assim um dos objetivos do Kanban é minimizar resistência à mudança e facilitá-la (BOEG, 2012).

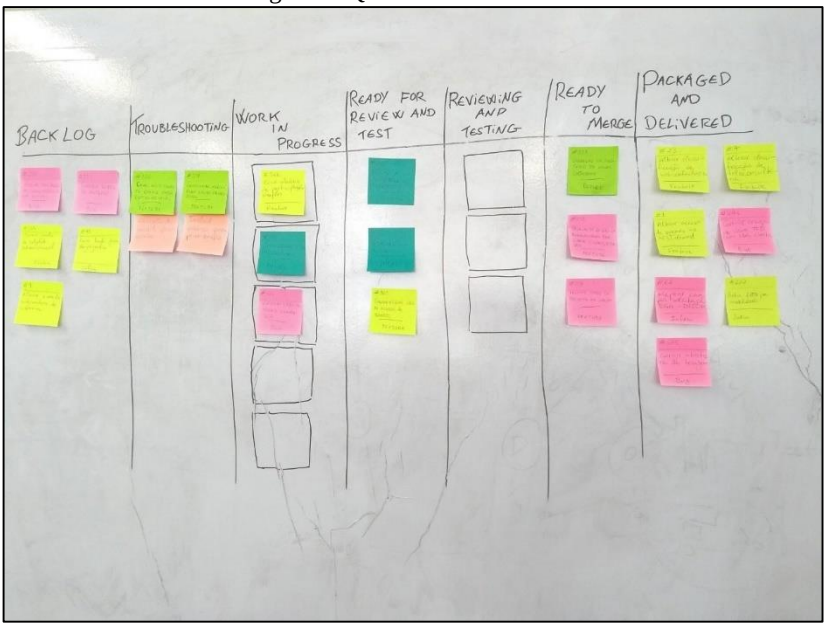
Com foco no trabalho que atinge as necessidades do cliente, o Kanban define 10 práticas gerais:

- Visualização do fluxo de trabalho;
- Limitação do trabalho em andamento;
- Definir políticas de gestão de qualidade;
- Ajustar cadência;
- Medir fluxo de trabalho;
- Definir prioridades;
- Identificar classes de serviço;
- Gerência de fluxo;
- Estabelecer acordos de nível de serviço e;
- Foco na melhoria contínua.

A operacionalização desses conceitos é atingida por intermédio de um quadro Kanban (BOEG, 2012). Esse quadro exibe um conjunto de cartões que descrevem uma tarefa a ser resolvida. Esses cartões são organizados no quadro por colunas: uma coluna representa o estado de

desenvolvimento em que uma tarefa se encontra. Por exemplo, o quadro Kanban ilustrado na Figura 2 apresenta 25 cartões organizados em 7 colunas.

Figura 2: Quadro Kanban físico



Fonte: autor

Interpretando o quadro, pode-se notar que as tarefas na coluna *Backlog* estão definidas, mas o seu desenvolvimento ainda não começou.

As tarefas na coluna *Troubleshooting* estão esperando a resolução de algum problema que impede sua continuidade.

As tarefas na coluna *Work in progress* estão em desenvolvimento, portanto, estão associadas a um desenvolvedor. Nota-se que há cinco retângulos, com três deles preenchidos com tarefas, esses retângulos são a limitação do trabalho em andamento. Os retângulos informam que é possível no máximo 5 trabalhos em andamento paralelamente. Essa limitação pode ter inúmeros motivos como, número limitado de computadores disponíveis, espaço físico limitado ou ainda a quantidade de desenvolvedores envolvidos no projeto não possibilita que uma sexta tarefa seja desenvolvida.

As tarefas na coluna *Ready for review and test* estão prontas para revisão e na coluna *Reviewing and testing* estão sendo revisadas.

Na coluna *Ready to merge* estão as tarefas prontas para serem adicionadas ao sistema.

E por fim, na coluna *Packaged and delivered* localizam-se as tarefas a serem empacotadas e entregues para o cliente.

2.4 UMA FERRAMENTA KANBAN WEB

Apesar do Kanban ser uma abordagem ágil, incremental, evolutiva e de fácil compreensão e manutenção, é importante ressaltar algumas limitações típicas desse tipo de prática.

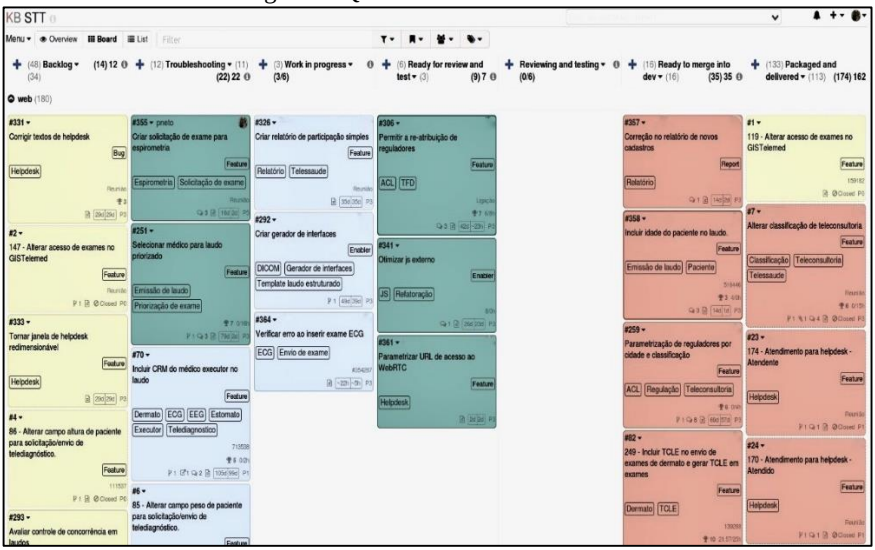
Ferramentas como quadro Kanban físico, instalado em uma parede, por exemplo, auxiliam na visualização do fluxo de trabalho, contudo há a desvantagem do uso do espaço físico. Considerando membros da mesma equipe geograficamente dispersos, há uma dificuldade em se manter a consistência entre quadros Kanban diferentes. Outra desvantagem de se utilizar meios físicos é a dificuldade de manter um histórico completo (necessário para resgatar pacotes de trabalho já terminados) e *backup* em caso de sinistros (como uma faxineira motivada pelo desejo de limpar uma parede coberta de papéis).

Com o objetivo de resolver essas limitações, atualmente há uma demanda para ferramentas automatizadas de planejamento, monitoração e controle.

Uma dessas ferramentas é o Kanboard[4], que é utilizado na gerência de projetos do laboratório de Telemedicina. O foco em simplicidade e minimalismo pode dar uma aparência amadora para esta ferramenta *web*, porém suas funcionalidades revelam um sistema poderoso e estável. A versão da ferramenta utilizada no laboratório de Telemedicina permite a visualização objetiva das tarefas, facilidade em arrastar tarefas entre colunas, diferentes formas de visualização, ações automáticas, gráficos de Gantt, relatórios de produtividade, internacionalização para 26 idiomas entre outras funcionalidades. A Figura 3 apresenta um quadro Kanban similar ao da Figura 2, porém usando o Kanboard. Nota-se que não há perda de informações; pelo contrário, o uso do Kanboard permite que seja visualizado várias informações como prioridade, complexidade e prazo com mais clareza.

[4] https://kanboard.net/

Figura 3: Quadro Kanban virtual



Fonte: autor

3 ESTADO DA ARTE

Neste capítulo são apresentadas as etapas de definição, execução e discussão da síntese do estado da arte para a abordagem *Use-Case* 2.0.

A pesquisa e avaliação de estudos primários que corroborem ou apresentem um contraponto ao estudo sendo realizado é importante para verificar se há inconsistências no assunto estudado. Assim, tendo como base a área médica, onde iniciou-se a demanda por estudos mais detalhados, a revisão sistemática da literatura define conceitos para direcionar a pesquisa.

Revisões e mapeamentos sistemáticos de literatura têm obtido respaldo da comunidade científica na computação, principalmente pela capacidade de prover um estudo detalhado e abrangente sobre o assunto pesquisado (HERNANDES et al., 2010).

Kitchenham (2004), define que um mapeamento sistemático da literatura (MSL) é o ato de identificar, valorar e interpretar todas as possíveis pesquisas relevantes a questões, áreas ou fenômenos de interesse pertinentes à pesquisa em andamento. Estudos individuais que contribuem para uma revisão sistemática são chamados de estudos primários e a revisão sistemática é considerada um estudo secundário.

Neste capítulo é apresentado um MSL simplificado e adaptado a um trabalho de conclusão de curso, portanto, não são seguidos todos os passos, porém a abordagem definida por Kitchenham (2004) é utilizada como um guia para sistematizar o levantamento do estado da arte.

Assim, o objetivo deste mapeamento consiste em identificar ferramentas web que implementam o *Use-Case* 2.0.

3.1 DEFINIÇÃO DAS FONTES DE PESQUISA

Inicialmente, para minimizar a ameaça a validade desta pesquisa, foram selecionadas importantes bibliotecas e bases de dados eletrônicas através do portal CAPES. As plataformas de busca selecionadas foram: IEEE Xplore, ACM Digital Library, ScienceDirect, Springer e Google.

Para a pesquisa de possíveis ferramentas que possam implementar e/ou integrar a abordagem *Use-Case* 2.0, o motor de busca Google foi adicionado às plataformas de pesquisa.

3.2 QUESTÕES DA PESQUISA

As questões sobre a pesquisa são definidas para capturar os pontos mais importantes e verificar a existência de trabalhos correlacionados ou similares a este. Abaixo, as perguntas formuladas:

Q1. Quais ferramentas *web* existem para modelar *Use-Case* 2.0?

Q2. Quão alinhadas estas ferramentas estão com *Use-Case* 2.0?

Q3. Essas ferramentas estão ou podem ser integradas a uma ferramenta de gerenciamento de projeto?

Nesse contexto, com base nos conceitos apresentados por Jacobson, Spence e Bittner (2011), foram identificadas as principais características que uma ferramenta deve ter para ser considerada uma modelagem da abordagem *Use-Case* 2.0. Essas características são mostradas no Quadro 1.

Quadro 1: Características para a implementação do *Use-Case* 2.0.

ID	Característica
F00	Conter a representação de casos de uso em qualquer complexidade.
F01	Conter a modelagem ou desenho de um diagrama de casos de uso de acordo com a UML 2.5
F02	Conter a modelagem ou desenho dos *slices* dos casos de uso.
F03	Conter a modelagem das *stories* nos casos de uso e nos *slices*.

Fonte: autor

Algumas características precisam existir em uma ferramenta que implementa a abordagem *Use-Case* 2.0. As características identificadas são apresentadas no Quadro 2.

Quadro 2: Características em ferramentas que implementam *Use-Case* 2.0.

ID	Características
F10	A ferramenta é código aberto. (Não é uma característica necessária se a ferramenta possibilitar integração com ferramentas de gerência de projetos)

F11	A ferramenta é gratuita
F12	A ferramenta tem suporte para a integração com ferramentas de gerência de projetos (por exemplo com *plug-in*s)

Fonte: autor

3.3 TERMOS E *STRINGS* DE BUSCA

Para execução dessa pesquisa, foram considerados os termos ilustrados no Quadro 3. Os termos definidos compõem as *strings* de busca, utilizadas para a pesquisa nas bases de dados.

Quadro 3: Termos de busca.

Termo	**Sinônimo**
use case	use-case, use case, use-case 2.0, use case 2.0
tool	tool, engine, web tool, system, platform, plugin
modeling	modeling, implemente
diagram	diagram
web	Web
desktop	Desktop
mobile	mobile
free	Free
online	Online
gui	gui, ui, interface

Fonte: autor

Baseado nos termos de busca do Quadro 3 é possível construir *strings* de busca que são a entrada para os mecanismos de busca. Os resultados das buscas auxiliam na verificação de possíveis ferramentas

que implementem a abordagem *Use-Case* 2.0. As *strings* de busca são explicitadas no Quadro 4.

Quadro 4: *Strings* de busca para *Use-Case* 2.0.

ID	**Base de dados**	***String* de busca**
A0	IEEE Xplore	"use case 2.0" AND tool OR "use case 2.0" AND engine OR "use case 2.0" AND modeling OR "use case 2.0" AND system OR "use case 2.0" AND platform OR "use case 2.0" AND plugin
A1	ACM Digital Library	"use case 2.0" AND tool OR "use case 2.0" AND engine OR "use case 2.0" AND modeling OR "use case 2.0" AND system OR "use case 2.0" AND platform OR "use case 2.0" AND plugin
A2	ACM Digital Library	"use case 2.0" OR "use cases 2.0" OR "use-cases 2.0"
A3	Springer	"use case 2.0" OR "use cases 2.0" OR "use-cases 2.0"
A4	ScienceDirect	"use case 2.0" OR "use cases 2.0" OR "use-cases 2.0"
A5	Scopus	ALL("use case 2.0") OR ALL("use cases 2.0") OR ALL("use-case 2.0")
A6	Google	"use case 2.0" tool web uml diagram modeling implement gui

Fonte: autor

3.4 EXECUÇÃO DA PESQUISA E RESULTADOS

A pesquisa foi executada no final de 2017 por um graduando em ciência da computação e revisada por um mestre em ciência da computação com experiência no uso de casos de uso no contexto de modelagem de sistemas de Telemedicina. O Quadro 5 apresenta um sumário dos resultados da pesquisa, exibindo o número de artigos

encontrados em cada base de dados. Apenas os 100 primeiros resultados foram analisados na pesquisa do Google, pois foi observado pouca relevância nos resultados seguintes.

Quadro 5: Resultados da busca para *Use-Case* 2.0

Base de dados	**ID *String* de busca**	**Número de resultados**	**Resultados analisados**	**Resultados relevantes**
IEEE Xplore	A0	4	4	0
ACM Digital Library	A1	0	0	0
ACM Digital Library	A2	2	2	0
Springer	A3	5	5	0
ScienceDirect	A5	0	0	0
Google	A6	150	100	0

Fonte: autor

Foi observado sobre a pesquisa realizada, com o escopo definido sobre implementações da abordagem *Use-Case* 2.0, um retorno insatisfatório de resultados relevantes. Logo, optou-se por expandir o escopo, para ferramentas que consideram a modelagem clássica de casos de uso.

No Quadro 6 foram definidas novas *strings* de busca considerando a modelagem clássica de casos de uso.

Quadro 6: *Strings* de busca para casos de uso.

ID	**Base de dados**	***String* de busca**
B0	IEEE Xplore	"use case" AND interface AND modeling
B1	IEEE Xplore	"use case" AND tool AND modeling

B2	ACM Digital Library	"use case" AND tool OR "use case" AND engine OR "use case" AND modeling OR "use case" AND system OR "use case" AND platform OR "use case" AND plugin
B3	ACM Digital Library	"use case" tool AND modeling AND UML
B4	ScienceDirect	"use case" AND tool OR "use case" AND engine OR "use case" AND modeling OR "use case" AND system OR "use case" AND platform OR "use case" AND plugin
B5	Google	"use case" tool web online desktop mobile free uml diagram "open code" interface

Fonte: autor

A pesquisa aplicada para a procura de uma ferramenta que implementasse os casos de uso retornou resultados relevantes. Os resultados podem ser observados no Quadro 7.

Quadro 7: Resultados da busca para os casos de uso.

Base de dados	**ID *String* de busca**	**Número de resultados**	**Resultados analisados**	**Resultados relevantes**
IEEE Xplore	B0	213	100	0
IEEE Xplore	B1	424	100	0
ACM Digital Library	B2	1	1	0
ACM Digital Library	B3	1.076	100	0
ScienceDirect	B4	949	100	0

Google	B5	956	100	5

Fonte: autor

Os resultados da pesquisa estão no Quadro 8 e mostram que há ferramentas que implementam a modelagem de casos de uso.

Quadro 8: Referências relevantes encontradas na pesquisa.

Fonte	Referências relevantes
Google	www.visual-paradigm.com/
Google	https://dashboard.genmymodel.com/
Google	https://www.lucidchart.com/
Google	http://staruml.io/
Google	https://www.draw.io/

Fonte: autor

Os resultados da pesquisa possibilitam responder as questões da pesquisa. As seções a seguir descrevem em detalhes o resultado da execução da pesquisa por questão de pesquisa.

Q1: quais ferramentas informatizadas existem para modelar *Use-Case* 2.0?

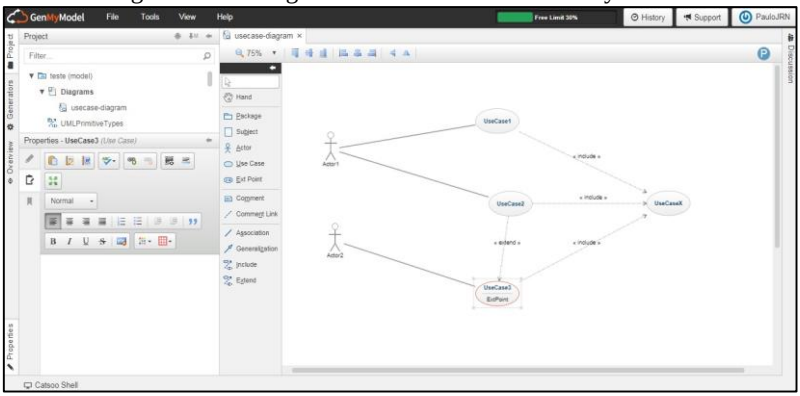
Não foram encontrados nas bases científicas artigos que abordam o projeto, desenvolvimento ou avaliação de ferramentas implementadas com base na abordagem *Use-Case* 2.0. Contudo, ao estender a busca para os casos de uso clássicos, foram encontradas ferramentas *web* e *desktop* que permitem a modelagem gráfica de casos de uso tradicionais. Ao todo, foram encontradas 5 ferramentas que implementam os casos de uso.

A ferramenta *web* GenMyModel[5] implementa os componentes clássicos de um modelo de casos de uso, como pode ser visto na Figura 4. Esta ferramenta permite o detalhamento dos casos de uso e a criação de pontos de extensão. Também é possível exportar os casos de uso como

[5] https://dashboard.genmymodel.com/

uma figura ou gerar código Java ou SQL, porém estas funcionalidades são pagas.

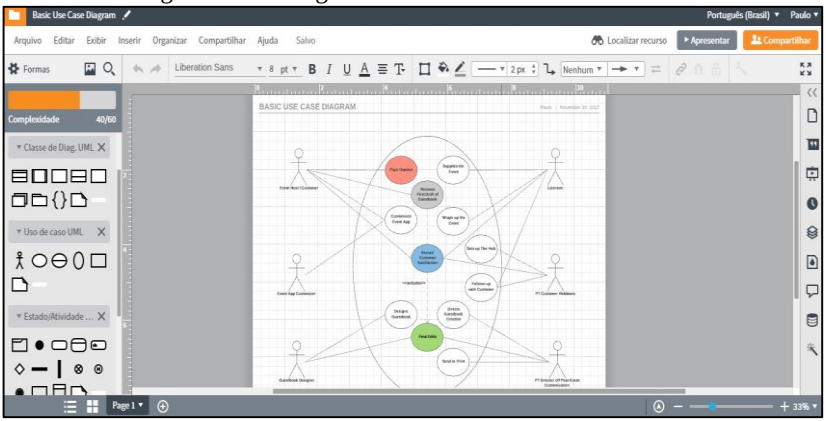
Figura 4: Modelagem de caso de uso com GenMyModel.



Fonte: autor

A ferramenta Lucidchart[6] implementa todos os componentes básicos para a construção de um modelo de casos de uso e ainda possui vários componentes UML adicionais para serem utilizados, como componentes visuais de estado, fluxos alternativos, entre outros. A Figura 5 contém um exemplo de modelo de caso de uso.

Figura 5: Modelagem de caso de uso com Lucidchart

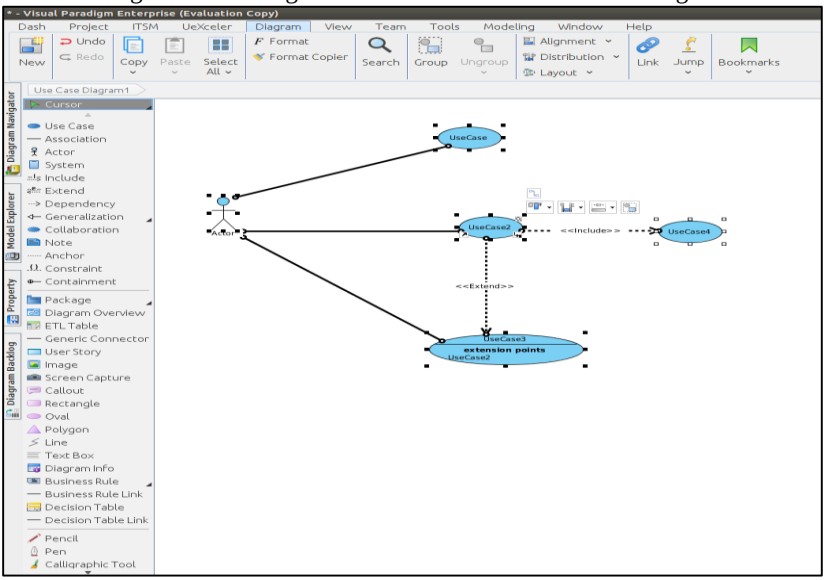


Fonte: autor

[6] https://www.lucidchart.com/

Visual Paradigm[7] é uma ferramenta *desktop* que contém suporte a gerência de projeto e implementa os componentes básicos dos casos de uso (Figura 6). Esta ferramenta permite integrar os casos de uso à ferramenta de gerência contida no próprio Visual Paradigm.

Figura 6: Modelagem de caso de uso com Visual Paradigm

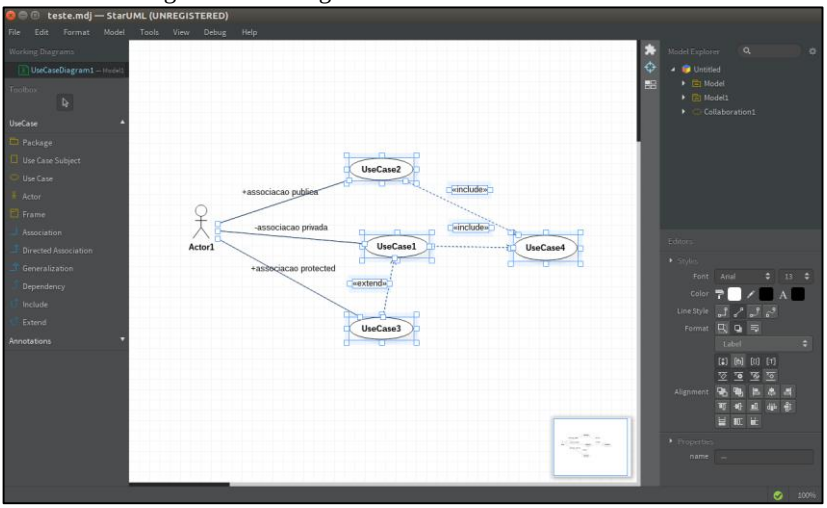


Fonte: autor

StarUML[8] é uma ferramenta *desktop* e gratuita que implementa os componentes básicos dos casos de uso (Figura 7) e permite exportá-los como JSON, PHP, JavaScript, DDL (*Data Definition Language*) entre outras, gerando os respectivos códigos.

[7] https://www.visual-paradigm.com/
[8] http://staruml.io/

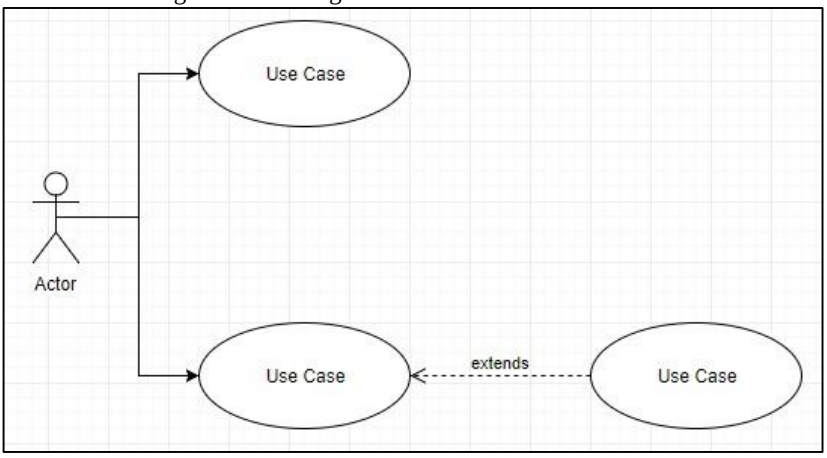
Figura 7: Modelagem de caso de uso com StarUML.



Fonte: autor

A ferramenta DrawIO[9] é uma ferramenta *web* e gratuita que permite a criação de casos de uso. Contém componentes UML adicionais para uso.

Figura 8: Modelagem de caso de uso com DrawIO



Fonte: autor

[9] https://www.draw.io/

Q2: quão alinhadas essas ferramentas são com os *Use-Case* 2.0?

Para responder esta pergunta, uma análise de cada ferramenta foi realizada para verificar quais das ferramentas contém todas as características necessárias para ser classificada como uma implementação da abordagem *Use-Case* 2.0.

A ferramenta GenMyModel é uma ferramenta *web* e gratuita até um certo nível de complexidade do modelo de casos de uso, ou seja, a ferramenta possui um limite de casos de uso a serem criados. Com isso a ferramenta falha na característica F00.

No modo gratuito, não é permitido projetos privados ou salvar imagens contendo o modelo de casos de uso, o que limita esta ferramenta para o contexto deste trabalho. Outra limitação é a não contemplação dos *slices*, falhando nas características F02 e F03.

Esta ferramenta não é código aberto e não possui integração com ferramentas de gerência de projeto, portanto não contém as características F10 e F12 respectivamente.

A ferramenta Lucidchart não permite o detalhamento textual interno de um caso de uso (falha em F03), mas permite comentários e anotações. Lucidchart é *web* e gratuita até um certo nível de complexidade do modelo de casos de uso, falhando em F00. A ferramenta não contempla os *slices* previstos no *Use-Case* 2.0 (falha em F02).

Não é uma ferramenta de código aberto, portanto falha em F10.

A ferramenta Visual Paradigm não é uma ferramenta gratuita (falha em F11), podendo utilizar suas funcionalidades sem custos por apenas 30 dias. Outro ponto negativo é a não contemplação dos *slices* (falha em F02 e F03). Logo, não contempla algumas das características para este trabalho que é a gratuidade e/ou código aberto da ferramenta (falha em F10).

A ferramenta StarUML não contempla a utilização dos *slices* e das *stories* (falha em F02 e F03 respectivamente), não contendo características essenciais da abordagem *Use-Case* 2.0. Outra característica não existente é a do código aberto (falha em F10).

Por fim, a ferramenta DrawIO não contém componentes chave da abordagem *Use-Case* 2.0 como os *slices* ou as *stories* (falha em F02 e F03 respectivamente). Esta ferramenta permite a adição de propriedades ao caso de uso, mas não possibilita a adição de um detalhamento textual. É uma ferramenta de código fechado, não contemplando a característica F10.

Em relação a essa questão de pesquisa, foi identificado uma lacuna no corrente estado da arte e prática. As ferramentas analisadas não

possuem todas as características necessárias que a classifique como uma ferramenta que implementa a abordagem *Use-Case* 2.0.

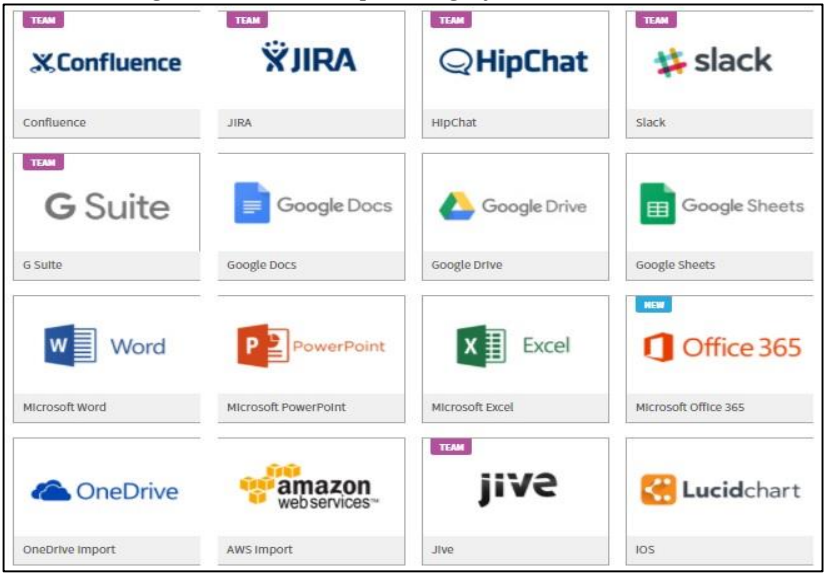
Q3: essas ferramentas estão ou podem ser integradas a uma ferramenta de gerenciamento de projeto?

Nesta parte será verificado a ausência ou presença de integração a ferramentas de gerenciamento de projeto por parte das ferramentas em análise.

Nas ferramentas GenMyModel, StarUML e DrawIO não foi identificado integração destas ferramentas com nenhuma ferramenta de gerência de projeto (falha em F12). As ferramentas Visual Paradigm e Lucidchart possuem suporte à integração com várias ferramentas.

A ferramenta Lucidchart contém várias ferramentas para integração, porém nenhuma delas é uma ferramenta de gerência de projetos como pode ser verificado na Figura 9. Logo, a característica F12 não se aplica.

Figura 9: Ferramentas para integração com Lucidchart.



Fonte: https://www.lucidchart.com/market/

A ferramenta Visual Paradigm contém uma ferramenta de gerência de projeto embutida na qual integra com os casos de uso, porém não foi

encontrada nenhuma integração com uma ferramenta que implementa a abordagem *Use-Case* 2.0.

3.5 DISCUSSÃO

A partir da análise das ferramentas pesquisadas foi possível perceber que nenhuma das ferramentas analisadas contém todas as características necessárias para implementar e/ou integrar o *Use-Case* 2.0 em sua totalidade.

Com os resultados obtidos na busca e na análise das ferramentas, um quadro com as informações foi gerado.

Quadro 9: Características das ferramentas analisadas.

Ferramenta	**Plataforma**	**Integração com ferramentas web?**	**Gratuita?**
GenMyModel	*Web*	Não	Não
Lucidchart	*Web*	Sim	Não
Visual Paradigm	*Desktop*	Sim	Não
StarUML	*Desktop*	Não	Sim
DrawIO	*Web*	Não	Sim

Fonte: autor

Assim, na análise do estado da arte não foi possível encontrar ferramentas que suportem a abordagem de *Use-Case* 2.0, reforçando a necessidade da implementação de um módulo integrado a uma ferramenta de gerenciamento de projetos.

Observou-se que o Kanboard é uma ferramenta de gerência de projetos com as características necessárias para a implementação de um módulo com a abordagem *Use-Case* 2.0. O Kanboard também é utilizado no laboratório de Telemedicina, que será utilizado nas avaliações, o que incentiva o desenvolvimento do módulo para o Kanboard.

O próximo capítulo apresenta, então, a análise e modelagem da solução proposta neste trabalho.

4 MODELAGEM DA SOLUÇÃO

Neste capítulo é descrito o processo de engenharia de requisitos para a modelagem e desenvolvimento da ferramenta proposta. Nas seções seguintes serão descritos o processo de engenharia de requisitos, a prototipação evolutiva da interface com o usuário e a modelagem de componentes do sistema.

4.1 ENGENHARIA DE REQUISITOS

Nesta seção, os principais requisitos para a implementação e o funcionamento do *Use-Case* 2.0 no Kanboard são apresentados. Inicialmente identifica-se os requisitos funcionais e não funcionais para o sistema, seguido da utilização da abordagem ágil *Use-Case* 2.0 para identificar atores, casos de uso e *slices*.

4.1.1 Requisitos funcionais

Um requisito funcional descreve as funções ou tarefas de um sistema a serem implementadas (IEEE-29148, 2011). Os requisitos funcionais apresentados no Quadro 10 foram levantados a partir de reuniões com o orientador e o coorientador deste trabalho, na qual foram identificados os requisitos necessários para uma implementação da abordagem *Use-Case* 2.0. Os requisitos levantados têm como base de estudo o livro escrito por Jacobson, Spence e Bittner (2011), que define o *Use-Case* 2.0 e informa quais funcionalidades são necessárias para alcançar o objetivo de implementar o *Use-Case* 2.0.

Quadro 10: Requisitos funcionais.

ID	Descrição
RF01	A ferramenta deve permitir que o usuário crie um ator (para interagir com os casos de uso), edite um ator (para modificar alguma característica) e exclua um ator (para remover atores não utilizados).
RF02	A ferramenta deve permitir a criação, edição e exclusão de casos de uso e de seus componentes internos como as histórias e fluxos.

RF03	A partir de um caso de uso, a ferramenta deve permitir a criação de *slices*, com suas histórias selecionadas e os fluxos.
RF04	A ferramenta deve permitir a visualização de um diagrama com os atores, casos de uso e *slices* cadastrados no sistema.
RF05	A ferramenta deve permitir a visualização dos casos de uso e *slices* no quadro Kanban.

Fonte: autor

4.1.2 Requisitos não funcionais

Requisitos não funcionais são requisitos na qual o sistema precisa para operar ou existir (IEEE-29148, 2011). A seguir, o quadro com os requisitos não funcionais na qual o sistema não poderia ser acessado e/ou implementado.

Quadro 11: Requisitos não funcionais.

ID	Descrição
RNF01	A ferramenta deve poder ser acessada por um navegador *Web* (Google Chrome versão 65 ou superior, Firefox versão 57 ou superior).
RNF02	A ferramenta deve utilizar a linguagem de programação PHP 7.0, pois a lógica do *backend* do Kanboard foi construída em PHP 7.0.
RNF03	A ferramenta deve utilizar JSON para formatar os dados e serem utilizados pois é padrão do Kanboard.

Fonte: autor

4.1.3 Modelagem do Sistema em *Use-Case* 2.0

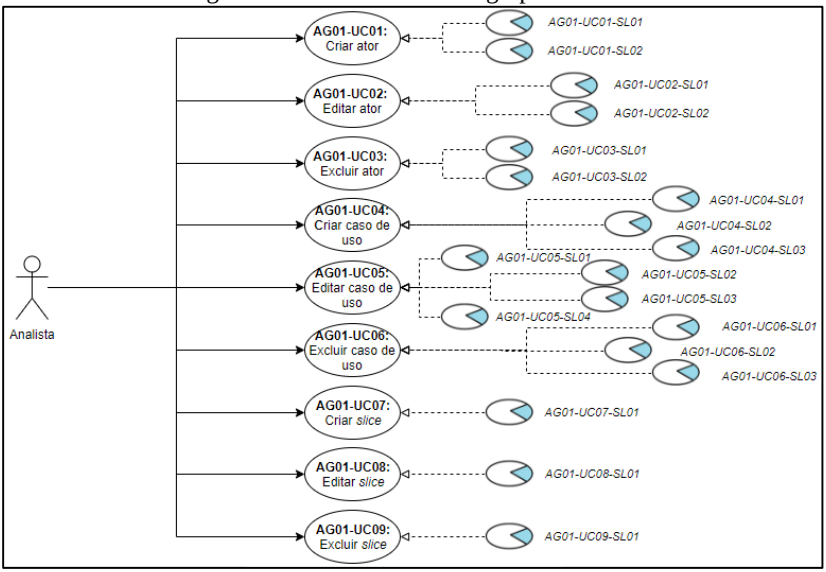
Por meio da análise das ferramentas similares (vide capítulo 3), estudo da abordagem *Use-Case* 2.0 na literatura (vide capítulo 2) e discussões com o orientador, foram obtidos os casos de uso, e posteriormente decompostos em *slices*. A Figura 10 e a Figura 11 ilustram a modelagem de um diagrama de casos de uso (OMG-UML, 2017), sendo estendido com a abordagem *Use-Case* 2.0.

A seguir casos de uso, *slices* e histórias de usuário (*stories*) são estruturados de forma agrupada e apresentados usando a codificação AG## para os agrupamentos, AG##-UC## para os casos de uso, AG##-UC##-SL## para os *slices* e HU## para as histórias de usuário.

Os casos de uso do agrupamento AG01 são referentes à construção de uma instância do *Use-Case* 2.0, enquanto os casos de uso do agrupamento AG02 são referentes ao diagrama de visualização dos componentes do *Use-Case* 2.0.

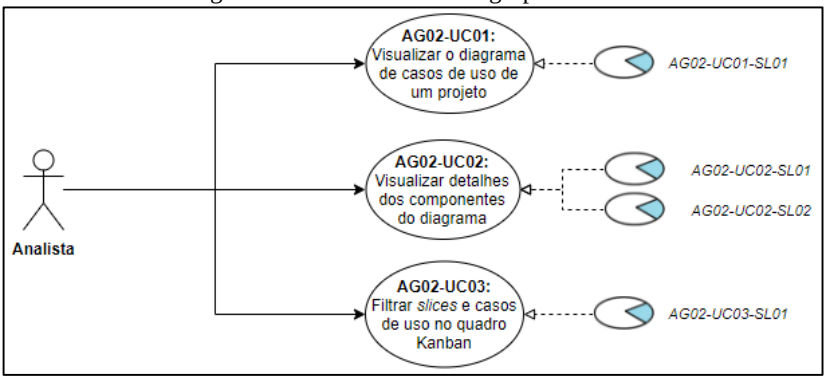
A divisão dos casos de uso em criar, editar e excluir para ator, caso de uso e *slice* foi pensada desta forma pois as consequências indiretas de cada uma das ações pode agregar uma complexidade alta. Por exemplo, ao excluir um *slice*, deve-se tratar adequadamente seu desacoplamento do caso de uso associado, excluir seus fluxos, etc. Portanto, optou-se por utilizar esta divisão dos casos de uso para balancear a complexidade sobre os casos de uso do sistema proposto.

Figura 10: Casos de uso do Agrupamento 1.



Fonte: autor

Figura 11: Casos de uso do Agrupamento 2



Fonte: autor

Como não existe ainda uma notação formal visual UML para identificar *slices*, é adotada, neste trabalho, então, a forma geométrica de uma elipse com uma fatia preenchida para representar os *slices* identificados para os casos de uso no diagrama de casos de uso das figuras 10 e 11.

Cada um dos casos de uso identificados para atendimento dos requisitos propostos, é detalhado seguindo a abordagem clássica de casos de uso, onde são descritos os atores, os fluxos base, os fluxos alternativos, pré-condições, entre outros. (COCKBURN, 2001)

Seguindo a abordagem de *Use-Case* 2.0, cada *slice* inclui ao menos uma história de usuário (*stories*), que por sua vez representa ao menos um fluxo base, alternativo ou de exceção do caso de uso.

Nesta seção é apresentado somente um agrupamento e um de seus casos de uso como exemplo. Os demais são apresentados no Apêndice I.

Agrupamento AG01: permitir criar artefatos de *Use-Case* 2.0

Caso de uso AG01-UC01: criar ator
Ator: analista
Pré-condição:
1. Usuário autenticado no sistema;
2. Acessar um projeto;
3. Acessar “Settings”.

Fluxo principal:
1. Clicar na opção “Actors”;
2. Clicar em “Add new actor”;

3. Digitar o nome do ator;
4. Clicar em “Save”.

Fluxo alternativo 01:

1. Deriva do passo 4
2. Se houver ator com o mesmo nome, mostra mensagem informando que já existe ator com o nome informado.

Pós-condição: há um novo ator cadastrado no sistema, a lista de atores está atualizada e os casos de uso podem acessar os novos atores.
Pós-condição do fluxo alternativo 01: o novo ator não foi cadastrado no sistema.

***Slice* AG01-UC01-SL01**
História de Usuário HU01: como analista eu quero criar um ator para vinculá-lo aos casos de uso.

***Slice* AG01-UC01-SL02**
História de Usuário HU02: como analista eu quero ser alertado se tentar replicar o nome de um ator já existente.

4.2 ESTIMATIVA DOS PONTOS DE CASOS DE USO

Quando um projeto é configurado, é muito importante saber, com antecedência, quantos recursos são necessários para que o projeto seja concluído. Esse tipo de conhecimento ajudará a estimar o custo e o prazo de entrega do projeto. Também ajudará a planejar o uso dos recursos (KARNER, 1993).

Para estimar a complexidade, o tamanho e o esforço a ser utilizado para o desenvolvimento da ferramenta *Use-Case* 2.0, foi utilizado o cálculo de pontos de casos de uso (Use Case Points - UCP) que utiliza o modelo de caso de uso como entrada. Essa estimativa inicial, auxilia no planejamento do cronograma do trabalho e consequentemente na alocação de recursos para o projeto.

Inicialmente, para estimar a complexidade dos itens do modelo de caso de uso, utiliza-se uma contagem chamada *Unadjusted Use Case Point* (UUCP). Para computar os UUCPs, é verificado a complexidade do ator, se é simples, média ou alta. O Quadro 10 auxilia nesta tarefa. O cálculo se realiza da mesma forma para os casos de uso, com a ajuda do Quadro 12.

Quadro 12: *Unadjusted Use Case Point* para ator

Complexidade	Definição	Peso (W)
Simples	Um ator é simples se representar outro sistema com uma interface de programação de aplicativos definida.	1
Média	Um ator é mediano se for: 1. Uma interação com outro sistema através de um protocolo. 2. Uma interação humana com um terminal.	2
Alta	Um ator é complexo se interage por meio de uma interface gráfica.	3

Fonte: autor

Quadro 13: *Unadjusted Use Case Point* para casos de uso.

Complexidade	Definição	Peso (W)
Simples	Tem até 3 transações, incluindo os passos alternativos, e envolve menos de 5 entidades.	5
Média	Tem de 4 a 7 transações, incluindo os passos alternativos, e envolve entre 5 e 10 entidades.	10
Alta	Tem acima de 7 transações, incluindo os passos alternativos, e envolve 10 ou mais entidades.	15

Fonte: autor

O cálculo da UUCP soma o peso (W) dos atores e dos casos de uso, onde *n* é o número de itens e *i* é a variação da complexidade dos atores e dos casos de uso, como pode ser observado na fórmula abaixo.

$$UUCP = \sum_{i=1}^{6} n_i * W_i$$

Para os atores, a complexidade é alta pois os dois atores disponíveis (analista e programador) interagem com a interface do sistema. Para os 12 casos de uso disponíveis, a complexidade é simples, pois há poucas transações para se chegar ao valor de saída desejado e as

únicas entidades envolvidas nas transações são o ator (que pode ser o analista ou o programador) e o sistema Kanboard. O Quadro 14 mostra as informações descritas.

Quadro 14: Cálculo UUCP

Total atores	Total casos de uso	Total UUCP
2*3 = 6	12*5 = 60	66

Fonte: autor

O UUCP é ponderado com o *technical complexity factor* (TCF). Os fatores de complexidade técnica (TCF) variam dependendo do quão difícil será a construção do sistema. Uma tabela com os fatores que contribuem para a complexidade do sistema é informada com os seus devidos pesos. Os fatores são classificados, no Quadro 15, em uma escala 0, 1, 2, 3, 4 e 5, na qual já está preenchido o fator de escala dentro do contexto do sistema a ser desenvolvido. O valor 0 significa que é irrelevante e o 5 significa que é essencial. Se o fator não é importante nem irrelevante, ele terá o valor 3 (KARNER, 1993).

Quadro 15: Fatores de complexidade técnica.

Escala do fator (F)	Fatores que contribuem para a complexidade	Peso (W)
0	Sistema distribuído	2
4	Objetivos de desempenho do aplicativo, em resposta ou desempenho	1
4	Eficiência do usuário final (on-line)	1
2	Complexidade do processamento interno	1
0	Reutilização, o código deve ser reutilizável em outros aplicativos	1
4	Facilidade de instalação	0,5
4	Usabilidade	0,5
0	Portabilidade	2
2	Mutabilidade	1
0	Concorrência	1
0	Recursos especiais de segurança	1
0	Fornece acesso direto para terceiros	1
3	Requer treinamento especial	1

Fonte: autor

A fórmula para o cálculo do TCF observada abaixo contém C1 e C2 como constantes definidas por Karner (1993), onde C1 é igual 0,6 e C2 0,01. A escala F informa a importância do fator no sistema a ser construído e o W é o peso que denota a importância do fator. O valor obtido para o TCF é de 0,79.

$$TCF = C_1 + C_2 \sum_{i=1}^{13} F_i * W_i$$

O *Environmental Factor* (EF) é o último elemento no cálculo dos pontos de casos de uso. Os fatores ambientais podem influenciar no desenvolvimento do projeto, portanto são estimados e incluídos no cálculo do UCP. Os parâmetros utilizados no cálculo são parecidos com os do TCF, com uma escala de números inteiros de 0 a 5 de classificação e fatores que influenciam na eficiência do projeto. A escala e os fatores podem ser observados no Quadro 16.

Quadro 16: Fatores de ambiente.

Escala do fator (F)	**Fatores que contribuem para a eficiência**	**Peso (W)**
4	Familiarização com os objetivos	1,5
0	Trabalhadores em tempo parcial	-1
3	Experiência do analista	0,5
4	Experiência com a aplicação em desenvolvimento	0,5
4	Experiência em orientação a objetos	1
5	Motivação	1
2	Dificuldade com a linguagem de programação	-1
4	Requisitos estáveis	2

Fonte: autor

A fórmula descrita abaixo para o cálculo do EF contém as constantes C1 e C2 com os valores 1,4 e -0,03, respectivamente.

$$EF = C_1 + C_2 \sum_{i=1}^{8} F_i * W_i$$

O cálculo do EF obteve o valor 0,665. Para finalizar o cálculo dos pontos dos casos de uso (UCP), realiza-se a multiplicação de todos os valores encontrados para encontrar o esforço total:

UCP = UUCP*TCF*EF
UCP = 66*0,79*0,665 = 34,6731

O resultado é uma estimativa do tamanho do projeto em pontos de caso de uso. Para fins de estimativa, o número de UCP pode então ser combinado com uma constante de produtividade (denominada *Mean Resources* – MR). A estimativa do esforço, utilizando os pontos de caso de uso, é mais precisa quando o número de projetos concluídos é maior, pois utiliza-se os dados do projeto anterior para estimar o atual. No entanto, como não houve projeto anterior a este, Karner (1993) propõe uma constante de produtividade de 20 horas/homem para valores entre 100 e 600 pontos de caso de uso para realizar o cálculo do esforço total do projeto, porém o valor encontrado no cálculo deste projeto foi de **34,6731** pontos de caso de uso, saindo do intervalo especificado por Karner.

Com consciência que o valor especificado da constante de produtividade é para projetos mais complexos, optou-se por manter o valor da constante sugerida por Karner e trabalhar com uma margem de erro pessimista para a conclusão da implementação proposta neste trabalho.

Portanto, estima-se gastar 20 horas/homem por ponto de caso de uso, o que resulta em 34,6731 * 20 = 693,462 horas/homem.

O resultado final apresenta a estimativa de 693,462 horas/homem para concluir a implementação do *Use-Case* 2.0 no Kanboard.

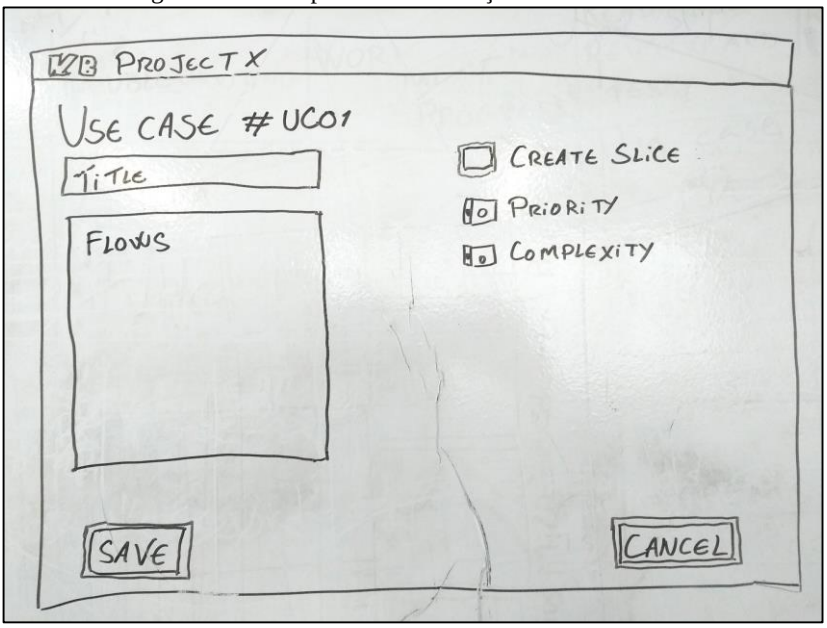
4.3 PROTOTIPAÇÃO

Baseado nos casos de uso do sistema levantados na seção 4.1, foi realizado um processo de prototipação de telas. Os primeiros protótipos de baixa fidelidade criados pelo autor ilustram a proposta de interface com os usuários do sistema *web*. Os protótipos elaborados são organizados por caso de uso.

4.3.1 AG01-UC04: criar caso de uso

Na Figura 12 um protótipo de tela para a criação de um caso de uso é apresentado. Nesta tela é possível criar e salvar um caso de uso e também possibilita acesso à tela de criação do *slice* do caso de uso em questão.

Figura 12: Protótipo da tela de criação de casos de uso.

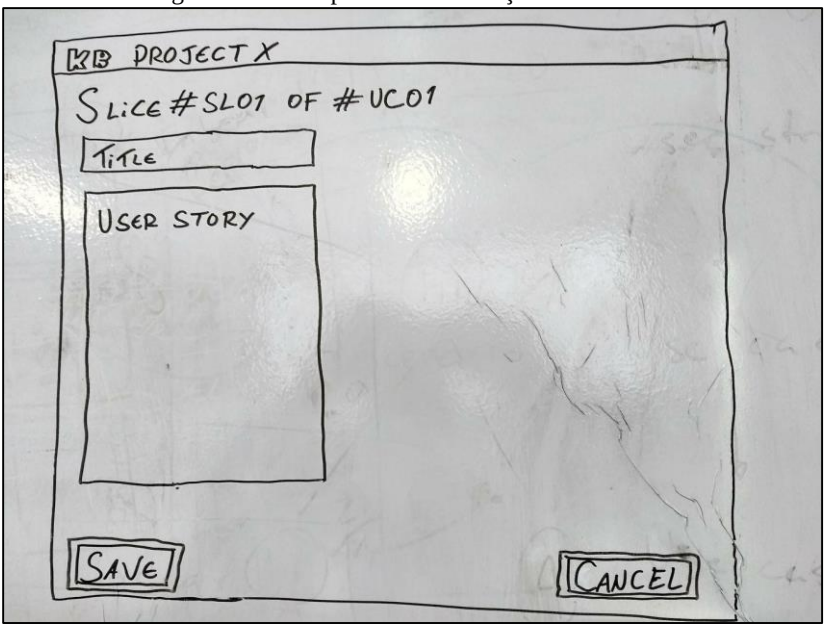


Fonte: autor

4.3.2 AG01-UC07: criar um *slice*

O protótipo para a tela de criação do *slice* é apresentado na Figura 13. Nesta tela é possível elaborar e salvar um *slice*.

Figura 13: Protótipo da tela de criação de um *slice*.

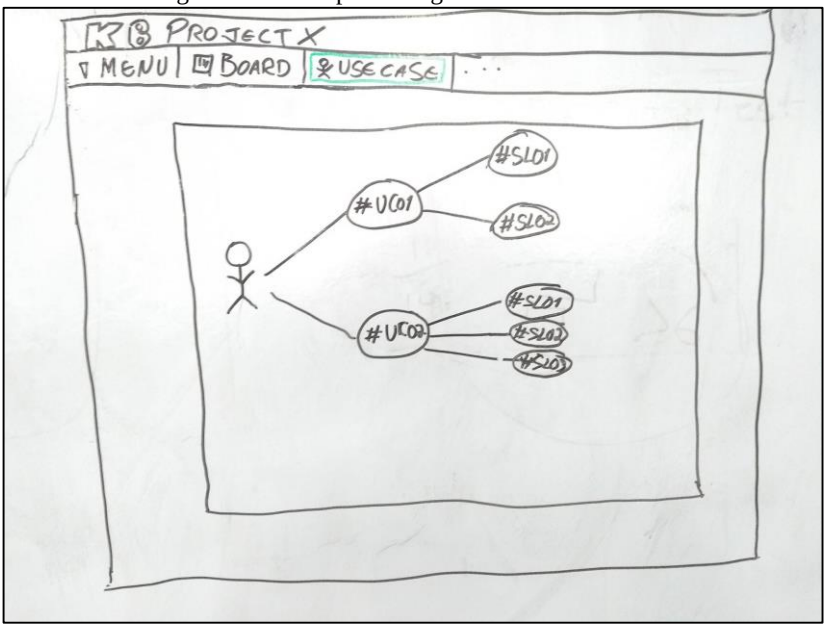


Fonte: autor

4.3.3 AG02-UC01: visualizar o diagrama de casos de uso de um projeto

O próximo protótipo, apresentado na Figura 14, mostra a tela com a interação dos atores, casos de uso e *slices*, apresentando um diagrama com a abordagem *Use-Case* 2.0. Ao clicar nos casos de uso e nos *slices* é possível interagir com os mesmos, arrastando-os conforme for cômodo para o usuário.

Figura 14: Protótipo do diagrama de casos de uso.

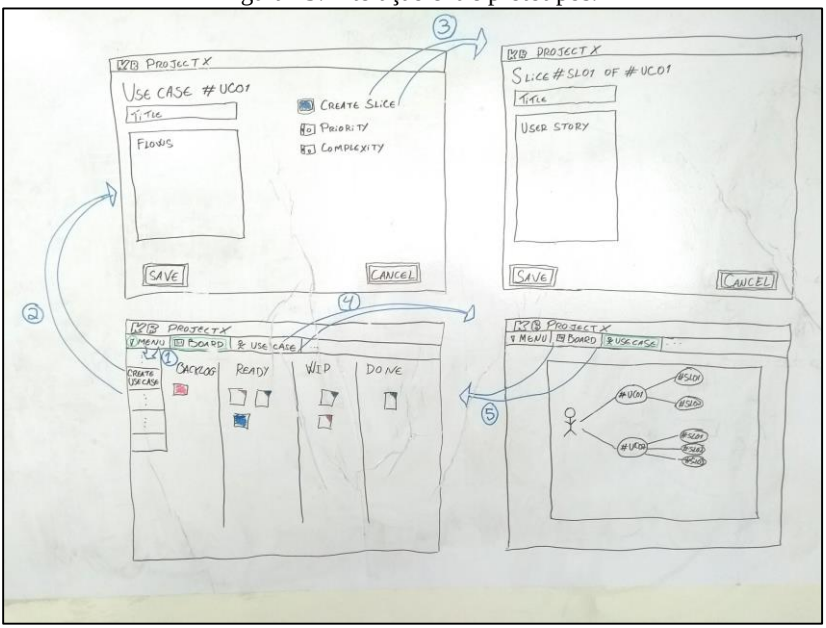


Fonte: autor

A Figura 15 é uma visão geral das telas idealizadas e dá uma noção da interação proposta entre as telas. No passo 1 apresentado na figura, o usuário seleciona na opção “Menu” a opção “Create use case” para visualizar a tela de criação de casos de uso. No passo 2 a tela abre para o cliente poder realizar as modificações necessárias. No passo 3 o cliente pode criar *slices*, que serão associados ao caso de uso sendo criado. No passo 4 o cliente pode clicar na opção “Use case” para acessar o diagrama com a abordagem *Use-Case* 2.0 e visualizar a interação do ator, dos casos de uso e dos *slices*. O passo 5 mostra que ao clicar na opção “Board” é

possível retornar a tela com o quadro de casos de uso e *slices* que interagiram na tela do passo 4.

Figura 15: Interação entre protótipos.



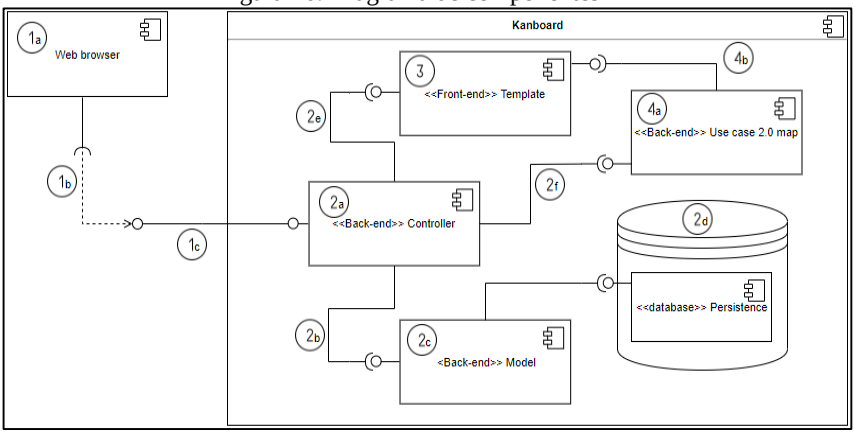
Fonte: autor

4.4 MODELAGEM DOS COMPONENTES

Para modelar os componentes envolvidos na implementação da abordagem *Use-Case* 2.0 no Kanboard, um diagrama de componentes foi construído. Um diagrama de componentes identifica os componentes que fazem parte de um sistema, subsistema ou mesmo componentes internos de um componente individual (GUEDES, 2018).

Na Figura 16 é apresentado o diagrama de componentes baseado na UML 2.5.

Figura 16: Diagrama de componentes



Fonte: autor

No diagrama apresentado, os componentes e as conexões são numerados para auxiliar na compreensão da implementação da abordagem *Use-Case* 2.0. A seguir a explicação para cada um dos pontos elencados na figura acima:

1. Acesso ao Kanboard com *Use-Case* 2.0 implementado.
    a. Através do navegador o usuário pode requisitar os recursos disponibilizados pelo Kanboard.
    b. A porta de interface requerida representa um grupo de mensagens enviadas a outro componente. A dependência é usada para indicar que a interface do navegador é satisfeita por uma interface no Kanboard.
    c. O Kanboard fornece uma porta para receber as mensagens enviadas pelo navegador.

2. Controlador e suas interações
    a. Recebe as requisições do usuário e identifica a ação a ser tomada. No controlador há a possibilidade de manipular uma instância do *Use-Case* 2.0 ou um *post-it* Kanban, para manter o contexto do trabalho, apenas o caso da manipulação do *Use-Case* 2.0 será abordado.
    b. Requisita uma ação ao modelo (atualizar, criar, recuperar ou deletar dados) enviando os dados referentes ao *Use-Case* 2.0.
    c. No modelo é construído um item SQL a partir da ação e dos parâmetros requisitados pelo usuário.
    d. Em seguida, o item SQL é executado por um gerenciador de banco de dados e os resultados (caso haja) são encaminhados ao controlador.
    e. Caso a requisição do usuário seja para visualizar as instâncias existentes do *Use-Case* 2.0 no quadro Kanban, o controlador envia os dados referentes às instâncias para a visão (chamado de *template* no código fonte) do sistema.
    f. Caso a requisição do usuário seja para visualizar o diagrama de casos de uso, o controlador envia os dados para a construção do diagrama de casos de uso no módulo contendo o VisJs.
3. Na visão (*template*), os casos de uso, *slices* e demais dados referentes ao *Use-Case* 2.0 são graficamente construídos para permitir a visualização no quadro Kanban por parte do usuário. Neste ponto, o usuário pode interagir com o quadro Kanban acessando os casos de uso e seus *slices* e modificando o estágio de desenvolvimento de cada instância *Use-Case* 2.0. A visão também disponibiliza uma área para a visualização do diagrama com de casos de uso.
4. Diagrama de casos de uso com *slices*.
    a. Os dados recebidos do controlador são estruturados e organizados para comporem um grafo com os componentes *Use-Case* 2.0 e suas associações.
    b. O grafo criado é enviado à visão para ser visualizado pelo usuário, na qual poderá organizá-lo ou acessar casos de uso e *slices* de maneira ágil.

5 DESENVOLVIMENTO

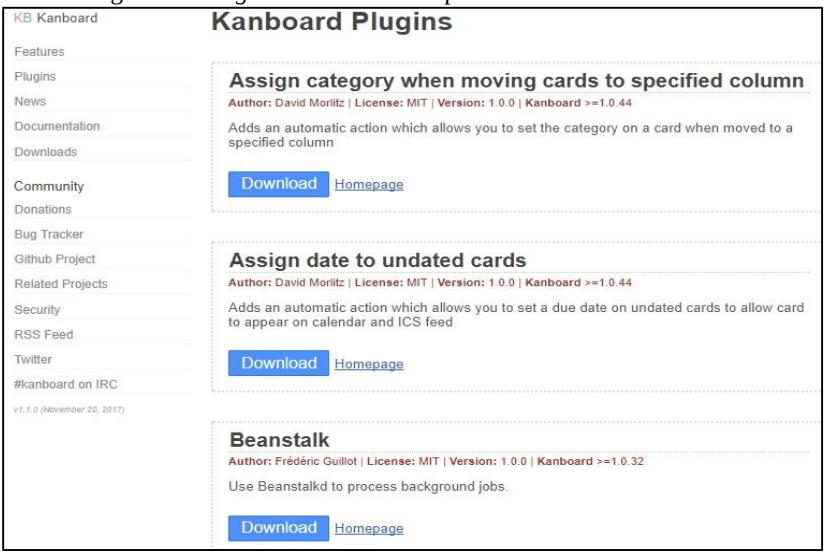
Este capítulo apresenta o desenvolvimento do módulo *Use-Case* 2.0 para a ferramenta Kanboard, as principais dificuldades encontradas e superadas, as tecnologias utilizadas, alguns exemplos de funcionalidades implementadas e, por fim, um comparativo entre a ferramenta implementada e as outras ferramentas similares identificadas no estado da arte.

5.1 DIFICULDADES NA IMPLEMENTAÇÃO

Inicialmente a tentativa de desenvolvimento iniciou-se com o estudo de implementação de *plug-ins* para o Kanboard. A ferramenta Kanboard permite o desenvolvimento de *plug-ins* por meio de *webhooks* — forma de executar uma ação de aplicativos externos (KANBAN PROJECT MANAGEMENT SOFTWARE, 2017).

Vários *plug-ins* foram instalados e verificados sua consistência, confirmando o suporte à criação de *plugi-ins*, porém alguns pontos necessários para o desenvolvimento do *Use-Case* 2.0 como um *plug-in* não foram encontrados

Figura 17: *Plug-ins* desenvolvidos para a ferramenta Kanboard



Fonte: https://kanboard.net

Algumas dificuldades para o desenvolvimento de *plug-ins* no Kanboard foram detectadas. A parte da manipulação do modelo do banco de dados, e em alguns módulos do sistema de controle implementado, o *webhook* não é contemplado. Devido a isso, não é possível acessar esses recursos através do *plug-in*. O mesmo aplica-se a manipulação de algumas telas, para possibilitar a criação das telas referentes a interface com a abordagem *Use-Case* 2.0.

No desenvolvimento percebeu-se a necessidade de explorar o código fonte da ferramenta e realizar alterações de modo a tornar possível a inclusão do *Use-Case* 2.0 e atender os requisitos propostos na seção 4.1.

Com isso, iniciou-se o desenvolvimento de um módulo com o *Use-Case* 2.0 que envolveu modificações no código fonte do Kanboard.

5.2 TECNOLOGIAS

O uso da ferramenta de gerência de projetos Kanboard como base para inclusão do novo módulo impõe restrições nas tecnologias a serem utilizadas no desenvolvimento para *Use-Case* 2.0. O Quadro 17 apresenta a lista das tecnologias utilizadas para o desenvolvimento do novo módulo.

Quadro 17: Tecnologias utilizadas no desenvolvimento.

Tecnologia	Fonte	Uso
PHP 7.0.22	http://www.php.net/	Utilizado no desenvolvimento do *backend* do aplicativo
SQLite 3.1	https://www.sqlite.org	Utilizado para a persistência dos dados do sistema
JavaScript 5.0	https://www.javascript.com	Utilizado no desenvolvimento do *frontend* para possibilitar componentes dinâmicos
HTML5	https://www.w3.org/TR/html5/	Utilizado para formatar os campos das páginas do sistema

Fonte: autor

Será comentado brevemente as tecnologias citadas no Quadro 17 que implementam o *frontend* — parte visual de um programa — e o *backend* — servidor de serviços para o *frontend* — do Kanboard e consequentemente da ferramenta proposta.

O *backend* da ferramenta é desenvolvido em PHP 7.0.22, portanto o módulo proposto manteve-se na mesma linguagem e versão para evitar conflitos. A mesma lógica foi aplicada para o JavaScript e o HTML5. Conflitos estes que poderiam comprometer a ferramenta Kanboard e consequentemente este trabalho.

O PHP é a linguagem de programação utilizada para construir a lógica do *backend* do Kanboard. O PHP é código aberto e versátil, possibilitando a inserção do seu código dentro do HTML.

O SQLite é o sistema de gerenciamento de banco de dados que utiliza SQL como linguagem de consulta para a manipulação dos dados do Kanboard, permitindo armazenar, recuperar ou excluir dados. O SQLite é uma linguagem menos robusta, se comparada a outras linguagens de consulta, porém é leve e atende perfeitamente os requisitos para a implementação da abordagem *Use-Case* 2.0.

O HTML é utilizado para formatar os campos que serão mostrados ao cliente e o JavaScript possibilita a adição de lógica e dinamismo no *frontend*.

Para permitir a modelagem gráfica dos atores, dos casos de uso e dos *slices* no módulo proposto, foi necessário pesquisar uma biblioteca Javascript que permitisse receber os dados do Kanboard e, a partir deles, modelar graficamente os elementos citados.

Foram pesquisadas as bibliotecas:

- http://www.draw2d.org/draw2d
- http://fabricjs.com/
- http://visjs.org/
- http://paperjs.org
- https://jsplumbtoolkit.com/
- https://github.com/jgraph/mxgraph
- https://github.com/d3/d3

As bibliotecas analisadas são bem elaboradas e contém uma variedade muito grande de funcionalidades para auxiliar na elaboração da modelagem proposta, porém a biblioteca VisJS destacou-se pela sua simplicidade e pelo fato de *plug-ins* já implementados para o Kanboard utilizarem esta biblioteca — o *plug-in* Relationgraph utiliza a biblioteca VisJS.

5.3 IMPLEMENTAÇÃO

Algumas partes do código fonte do Kanboard foram modificadas para a inclusão das novas funcionalidades. Estas modificações ocorreram em vários pontos, como no modelo de banco de dados (para a inclusão de tabelas) e na interface da tela (para possibilitar a visualização do componente de acesso ao diagrama de casos de uso).

O código-fonte da ferramenta alterada para suportar a abordagem *Use-Case* 2.0 está disponível em <*https://github.com/PauloJRN/tcc*>.

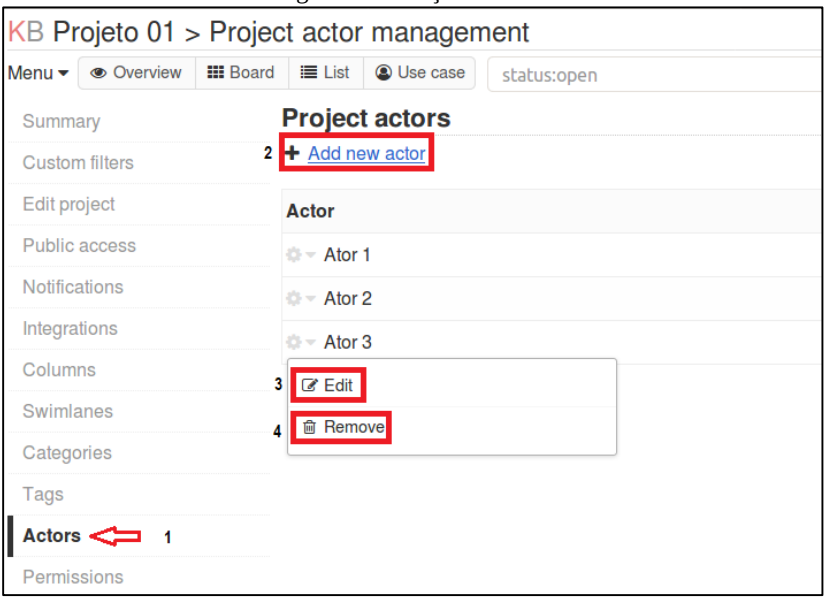
Serão mostrados os casos de uso e as respectivas telas do Kanboard que sofreram as modificações para possibilitar a implementação do *Use-Case* 2.0 e seus respectivos casos de uso e *slices*. É importante frisar que algumas telas se diferem dos protótipos idealizados pois durante a implementação foram realizadas melhorias nas interfaces idealizadas nos protótipos.

O tempo investido na implementação do sistema ao longo deste trabalho é de aproximadamente 25 dias, ou 600 horas. Ao observar o tempo estimado pelos pontos de casos de uso, de aproximadamente 693,462 horas, observa-se que o desenvolvimento foi concluído dentro do tempo estimado no cálculo dos pontos de casos de uso.

5.3.1 Tela de manipulação do ator

Os casos de uso AG01-UC01, AG01-UC02 e AG01-UC03 detalham, respectivamente, a criação, edição e exclusão de um ator. Para a criação de um ator, seleciona-se a opção “Actors” (passo 1) para poder visualizar a tela compreendida na Figura 17. No passo 2, clica-se em “Add new actor” e uma tela para digitar as informações do ator aparecerá. Com as informações sobre o ator preenchidas, há a possibilidade de editar (passo 3) ou remover o ator (passo 4).

Figura 18: Criação do ator



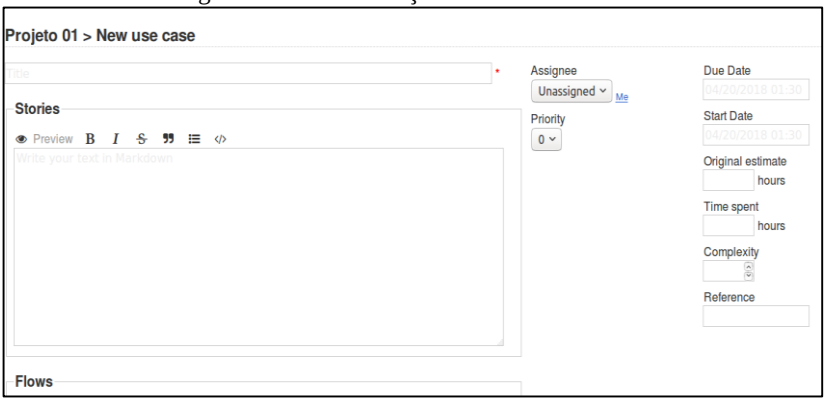
Fonte: autor

5.3.2 Tela de criação de casos de uso

O caso de uso AG01-UC04 explicita a criação do caso de uso. A tela para a criação de um caso de uso é extensa, portanto apenas uma fração dela foi mostrada na Figura 18. Na tela de criação do caso de uso, há campos como nome, histórias, fluxos (básicos, alternativos e de exceção), prioridade do caso de uso, ator, entre outros atributos a serem preenchidos.

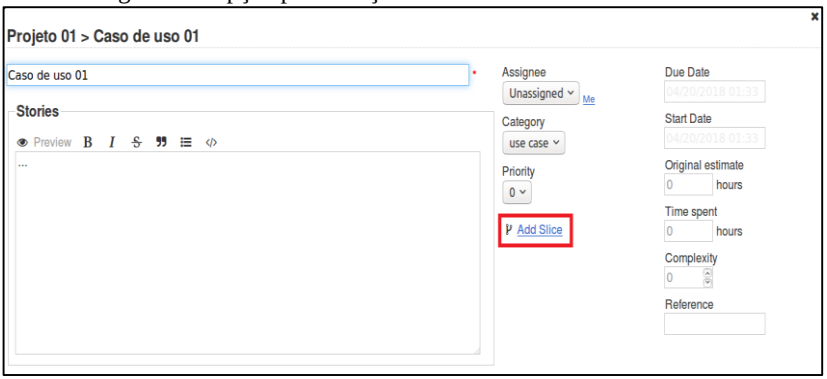
Um caso de uso no Kanboard possui propriedades únicas que o diferenciam de um *post-it* do Kanban. Apenas no caso de uso é possível associar atores, informando as partes interessadas no desenvolvimento daquele caso de uso, e criar *slices* (Figura 19), fragmentando o caso de uso. No diagrama de casos de uso, apenas casos de uso e seus respectivos atores e *slices* aparecem, não mostrando os *post-its* do Kanboard.

Figura 19: Tela de criação de um caso de uso.



Fonte: autor

Figura 20: Opção para criação do *slice* dentro do caso de uso.



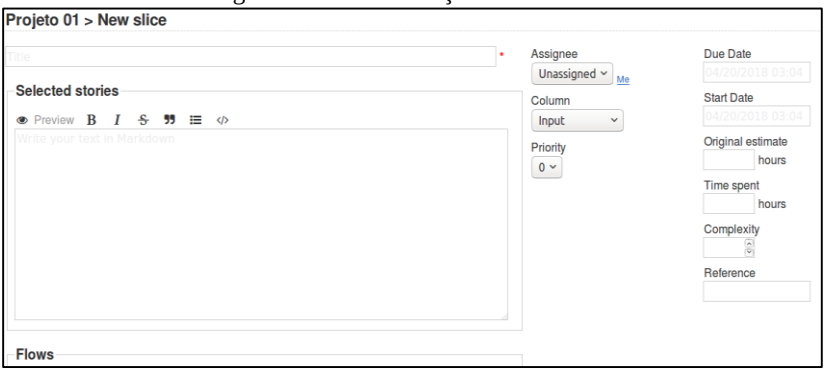
Fonte: autor

5.3.3 Tela de criação do *slice*

O caso de uso AG01-UC07 explana a criação do *slice***.** A tela de criação de um *slice* contém campos parecidos com os da criação do caso de uso, porém há campos diferentes. Os campos que se diferem do caso de uso são as “Selected stories” e os “Test cases”, sendo que o campo para informar o ator não aparece pois já é o mesmo ator do caso de uso.

Nas “Selected stories” são informadas as histórias selecionadas do caso de uso para compor o *slice*. E os “Test case” são os possíveis testes a serem feitos sobre as funcionalidades a serem implementadas pelo *slice*.

Figura 21: Tela de criação de um *slice*.

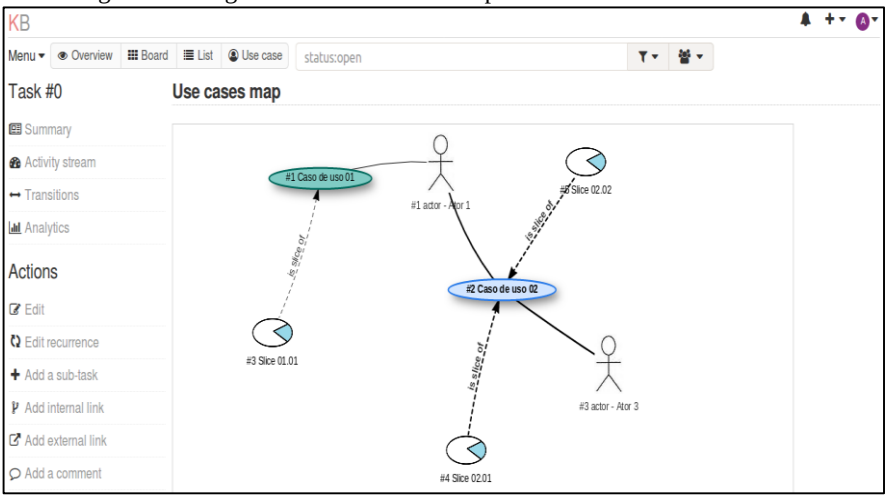


Fonte: autor

5.3.4 Tela de visualização do diagrama de casos de uso

O caso de uso AG02-UC01 detalha a visualização do diagrama de casos de uso de um projeto: na Figura 22, apresenta-se o diagrama com casos de uso e seus atores e *slice* para auxiliar na compreensão das relações existentes. Através do diagrama é possível acessar as informações dos casos de uso e dos *slices* com um duplo clique no *mouse*.

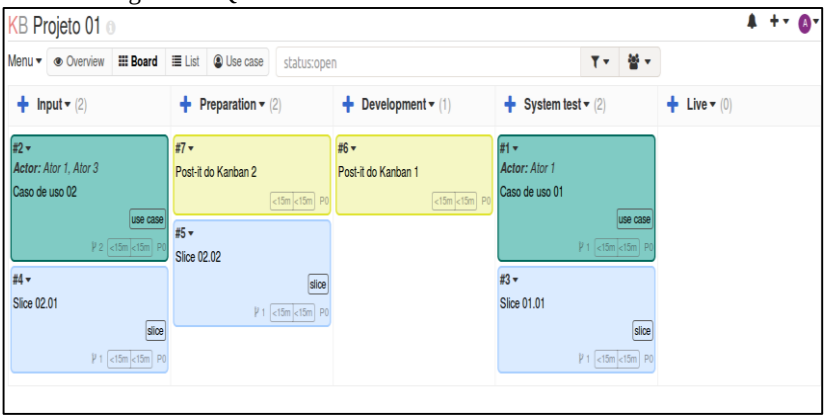
Figura 22: Diagrama de casos de uso implementado no Kanboard



Fonte: autor

Na Figura 23 o quadro Kanban aparece com o suporte ao *Use-Case* 2.0. Os casos de uso e os *slices* podem ser identificados observando-se os rótulos contidos nas caixas. As colunas são as fases na qual um *Use-Case* 2.0 deve percorrer para alcançar o seu objetivo.

Figura 23: Quadro Kanban com instâncias do *Use-Case* 2.0



Fonte: autor

5.4 COMPARAÇÃO DE FUNCIONALIDADES DAS FERRAMENTAS

Finalizada a implementação da abordagem *Use-Case* 2.0 no Kanboard, uma comparação da ferramenta implementada em relação àquelas ferramentas descobertas na análise do estado da arte (vide capítulo 3) é realizada. Nesta comparação contida no Quadro 18 é possível observar quais requisitos foram contemplados de acordo com a ferramenta avaliada.

Na primeira linha do quadro estão os requisitos funcionais (vide seção 4.1.1) requeridos para uma ferramenta que implemente o *Use-Case* 2.0 e na coluna mais à esquerda as ferramentas avaliadas na seção do estado da arte e a ferramenta implementada estão informadas. Para as ferramentas que atendem completamente um requisito, o símbolo (+) é marcado na linha e coluna correspondente, para as ferramentas que atendem parcialmente o símbolo (+/-) é marcado e para as ferramentas que não atendem o símbolo (-) é marcado.

Quadro 18: Comparação de funcionalidades.

	RF01	**RF02**	**RF03**	**RF04**	**RF05**
Visual Paradigm	+	+/-	-	+/-	-
GenMyModel	+	+/-	-	+/-	-
LucidChart	+	+/-	-	+/-	-
Star UML	+	+/-	-	+/-	-
Draw IO	+	+/-	-	+/-	-
Kanboard com a abordagem *Use-Case* 2.0	+	+	+	+	+

Fonte: autor

Ao analisar o Quadro 18, observa-se que as ferramentas Visual Paradigm, GenMyModel, LucidChart, Star UML e Draw IO possuem ênfase na construção e visualização de componentes básicos de casos de uso (RF01), porém não possuem um detalhamento maior como campos para informar as histórias de usuário ou os fluxos (RF02). Com a implementação da abordagem *Use-Case* 2.0 em uma plataforma de gerência de projeto foi possível habilitar um detalhamento maior dos casos de uso, com campos para informar histórias de usuário, fluxos, atores além de possibilitar a divisão dos casos de uso em *slices* (RF03).

As demais ferramentas possibilitam a visualização de um diagrama de casos de uso, porém não comportam a extensão *Use-Case* 2.0, que permite a visualização dos *slices*, componente essencial do *Use-Case* 2.0 (RF04). E por fim, a implementação apresentada permite a utilização direta da abordagem *Use-Case* 2.0 em uma ferramenta de gerência de projeto, o que não é contemplado pelas demais ferramentas (RF05).

A avaliação de comparação foi realizada pelo próprio autor deste trabalho e desenvolvedor da ferramenta, o que pode, naturalmente, representar uma ameaça à validade das conclusões. Entretanto, mesmo assim, a avaliação levanta indícios de que a ferramenta desenvolvida tem maior cobertura dos requisitos propostos do que as ferramentas pesquisadas no estado da arte.

6 AVALIAÇÃO

Neste capítulo é apresentada a avaliação do módulo implementado com a abordagem *Use-Case* 2.0 no intuito de verificar se as funcionalidades do *Use-Case* 2.0, propostas para este trabalho, podem ser atendidas e aceitas pelos potenciais usuários finais.

6.1 PLANEJAMENTO DA AVALIAÇÃO

O módulo de *Use-Case* 2.0 avaliado com dois focos: (i) avaliar a usabilidade do sistema e, (ii) avaliar a aplicação e atendimento da abordagem *Use-Case* 2.0, sob o ponto de vista de analistas de sistema e programadores de duas organizações de desenvolvimento de *software* da cidade de Florianópolis.

O sistema desenvolvido tem como foco os profissionais da área da computação como analistas de sistema, engenheiros de *software*, programadores, entre outros. Nesta avaliação tanto os analistas quanto os programadores, utilizaram todas as funcionalidades disponíveis pois possuem conhecimento sobre ambas as áreas.

Para a avaliação da usabilidade do sistema proposto neste trabalho foi utilizado o SUS (*System Usability Scale*), uma escala de usabilidade confiável e de baixo custo que pode ser usada para avaliações da usabilidade de sistemas. Para especificar a usabilidade de um sistema (adequação à finalidade), deve-se definir os usuários pretendidos do sistema, as tarefas que esses usuários executarão com ele e as características do sistema físico, organizacional e ambiente social em que será usado (BROOKE, et al., 1996).

Para avaliar a aplicação e atendimento da abordagem *Use-Case* 2.0, um questionário foi elaborado, contendo perguntas complementares na tentativa de obter uma acurácia maior na avaliação e com isso determinar mais precisamente os pontos positivos e negativos das funcionalidades da abordagem *Use-Case* 2.0 implementada.

6.1.1 Avaliação da usabilidade com o método SUS (*System Usability Scale*)

Segundo Brooke (1996), as medidas de usabilidade que o SUS cobre são a efetividade, sendo a capacidade dos usuários de concluir tarefas com qualidade usando o sistema, a eficiência, que é a quantidade de esforço e recursos para concluir uma tarefa e a satisfação como reação dos usuários ao uso do sistema.

O SUS é composto por 10 itens (Quadro 19) que abrangem uma variedade de aspectos da usabilidade do sistema, como a necessidade de suporte, treinamento e complexidade, e, portanto, possuem um alto nível de validade para medir a usabilidade de um sistema.

Quadro 19: Itens SUS

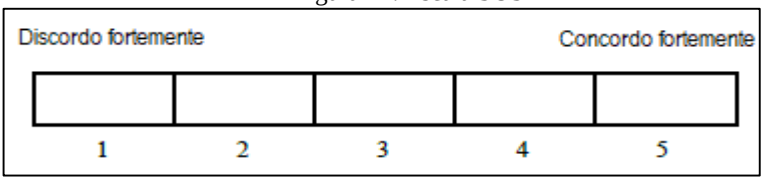
01. Eu gostaria de usar o sistema mais frequentemente.
02. Eu achei o sistema desnecessariamente complexo.
03. Eu achei o sistema fácil de usar.
04. Eu acho que precisaria de ajuda de uma pessoa com conhecimentos técnicos para usar o sistema.
05. Eu acho que as várias funções do sistema estão muito bem integradas.
06. Eu acho que o sistema apresenta muita inconsistência.
07. Eu imagino que a maioria das pessoas aprenderão a utilizar o sistema facilmente.
08. Eu acho o sistema muito atrapalhado/sobrecarregado de se usar.
09. Eu me senti confiante ao usar o sistema.
10. Eu precisei aprender várias coisas novas antes de conseguir usar o sistema.

Fonte: autor

Os itens elencados no SUS utilizam-se de uma escala conhecida como escala Likert (BROOKE, et al., 1996), que permite medir as atitudes e conhecer o grau de conformidade do entrevistado com as afirmações propostas. O SUS utiliza uma escala de 5 alternativas para identificar o grau de concordância com os 10 itens informados no Quadro 19, variando de “Concordo totalmente” até “Discordo totalmente”.

Na utilização do SUS o usuário lê o item e marca a opção correspondente à concordância. No caso de um usuário que não saiba responder sobre um item específico, o ponto central da escala deve ser marcado. A escala utilizada pode ser vista na Figura 24.

Figura 24: Escala SUS



Fonte: BROOKE (1996)

Os 10 itens foram selecionados para que a resposta comum a metade dos itens fosse forte concordância, e para a outra metade, forte discordância. Isso foi feito para evitar vieses de resposta causados pelos entrevistados, não tendo que pensar em cada declaração. Ao alternar itens positivos e negativos, o entrevistado deve ler cada afirmação e esforçar-se para pensar se concordam ou discordam (BROOKE, et al., 1996).

O SUS produz um valor único representando uma medida composta da usabilidade geral do sistema em estudo. Para calcular a pontuação do SUS, primeiro some as contribuições da pontuação de cada item. Para os itens 1,3,5,7 e 9, a contribuição da pontuação é a posição da escala menos 1. Para os itens 2,4,6,8 e 10, a contribuição é de 5 menos a posição da balança. A soma das pontuações deve ser multiplicada por 2,5 para obter o valor global da SUS. As pontuações do SUS têm um intervalo de 0 a 100 (BROOKE, et al., 1996).

6.1.2 Avaliação da aplicação de *Use-Case* 2.0

As questões específicas foram elaboradas para verificar a aceitação por parte dos avaliadores e a concordância com as respostas informadas na escala de usabilidade SUS. Estas questões, compreendidas no Quadro 20, avaliam pontos específicos para identificar possíveis melhorias futuras para o sistema e possibilita que trabalhos correlatos se beneficiem de dados mais específicos para suas análises.

Quadro 20: Questões específicas.

Questão	Motivação
AQ1. Quanto tempo de estudo para compreender o funcionamento do sistema? (em minutos)	Analisar se o tutorial gerado foi efetivo na utilização da ferramenta.
AQ2. Quanto tempo gasto para criar um ator, um caso de uso e um *slice*? (em minutos)	Analisar se a usabilidade da ferramenta proposta permite criar uma instância simples do *Use-Case* 2.0 em tempo hábil.
AQ3. Houve um aumento no tempo de utilização do sistema após a inclusão da abordagem *Use-Case* 2.0 no Kanboard?	Analisar se houve um aumento significativo na complexidade do Kanboard em relação ao Kanboard alterado.

AQ4. Quais foram as principais dificuldades encontradas ao utilizar o Kanboard com a abordagem *Use-Case* 2.0?	Analisar os pontos específicos de dificuldade ao utilizar a ferramenta.
AQ5. A experiência da aplicação do Kanboard com a abordagem *Use-Case* 2.0 na unidade organizacional foi benéfica?	Analisar a aceitação da ferramenta por parte dos avaliadores.
AQ6. Quais as principais técnicas da abordagem *Use-Case* 2.0 obteve maior aceitação?	Analisar quais pontos obtiveram maior aceitação.

Fonte: autor

6.2 EXECUÇÃO DA AVALIAÇÃO

A primeira parte da avaliação da ferramenta foi feita por meio do questionário SUS em conjunto com a segunda parte, contendo as perguntas específicas elencadas no Quadro 20 e executado em duas organizações de *software*.

Analistas e programadores dispuseram-se a avaliar a ferramenta proposta. Ambos exerceram o papel de analista de sistema na utilização da ferramenta de gerenciamento de projetos, pois o módulo implementado tem como foco os processos realizados pelo analista de sistema.

A primeira organização de *software* é o laboratório de Telemedicina da UFSC. O laboratório de Telemedicina foi criado pelo ministério da saúde em 2005 e desenvolve tecnologias de arquivamento, processamento e transmissão de imagens médicas, facilita o acesso à exames e laudos para médicos e pacientes, permite o treinamento via *web* conferências, com cursos e consultorias. Todas as funcionalidades descritas são disponibilizadas *online*. O laboratório de Telemedicina já realizou mais de 5 milhões de exames e contempla a rede pública de saúde de Santa Catarina. As principais tecnologias utilizadas pelo Telemedicina são as linguagens PHP e Javascript e seus respectivos *frameworks* Zend e Dojo. Para o banco de dados é utilizado PostgreSQL.

A avaliação foi executada presencialmente no laboratório de Telemedicina conforme a disponibilidade dos colaboradores, e os mesmos foram selecionados devido a sua familiaridade com a ferramenta Kanboard e consequentemente com sistema Kanban. Ao total, 8 colaboradores do laboratório de Telemedicina participaram da avaliação, testando todas as funcionalidades do *Use-Case* 2.0 implementado no Kanboard.

A segunda organização de *software* desenvolve uma API (*Application Programming Interface*) para auxiliar empresas a criar interações e medir os resultados analisando o comportamento do cliente. A análise possibilita identificar as possíveis ações a serem tomadas para melhorar o relacionamento empresa-cliente. Não foi informado pela empresa o número total de colaboradores, porém 6 dos colaboradores da empresa participaram da avaliação. A principal tecnologia que a empresa utiliza é a linguagem Java. As demais tecnologias não foram informadas no site correspondente à empresa. Por motivos de confidencialidade o nome da empresa será preservado.

As duas organizações de *software* trabalham com a ferramenta Kanboard, o que possibilita um ambiente propício para a avaliação da ferramenta proposta neste trabalho. Na segunda organização a avaliação através do SUS disponibilizado online obteve um total de 6 participantes. Alguns dos participantes (5 participantes) somente responderam às questões do questionário SUS, não avaliando a segunda parte referente à abordagem *Use-Case* 2.0.

Um tutorial (Apêndice II) foi criado para orientar os testes sobre a ferramenta proposta, dando uma sucinta descrição sobre a abordagem *Use-Case* 2.0 e informando como utiliza o sistema implementado, desde a criação dos casos de uso e seus *slices*, até a utilização posterior dos itens criados.

As avaliações ocorreram durante o período de 12/03/2018 até 06/04/2018 e os dados coletados são apresentados na seção seguinte.

6.3 ANÁLISE DOS DADOS

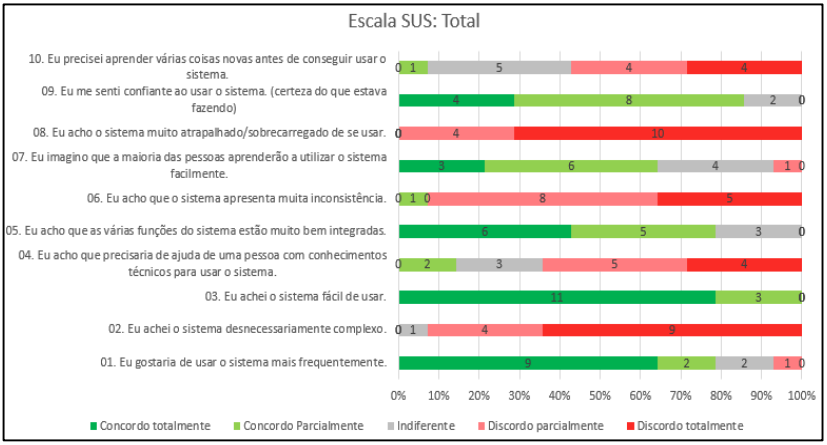
Finalizadas as avaliações, os dados coletados são analisados para verificar se os objetivos estabelecidos foram alcançados.

6.3.1 Análise da avaliação de usabilidade

Nesta etapa, os dados são analisados para verificar a aceitação, por parte dos avaliadores, do sistema implementado. No total, são 14 avaliadores que informaram através do questionário SUS as impressões sobre o sistema.

O Gráfico 1 mostra os dados coletados sobre os itens do SUS. Será realizada uma análise sobre o gráfico contido para identificar a opinião dos avaliadores sobre o sistema proposto. Os gráficos separados por itens SUS que contém as porcentagens indicadas na análise a seguir estão no Apêndice IV.

Gráfico 1: Gráfico com os dados sobre a coleta dos dados com SUS.



Fonte: autor

Com base no item 10 do SUS, observa-se que 57% dos avaliadores obtiveram pouca, ou não obtiveram, dificuldades em compreender o *Use-Case* 2.0. 36% se mantiveram indiferentes em relação ao volume de conhecimento para utilizar o sistema e 7% gastou mais tempo estudando para compreender o *Use-Case* 2.0. Conclui-se para este item que houve um volume médio de conteúdo a ser absorvido pelos avaliadores para compreender o funcionamento do *Use-Case* 2.0.

Os valores observados no item 9 mostram que os avaliadores têm um nível alto de confiança, mas há algum ponto do sistema que pode não ser intuitivo, causando os 57% de concordo totalmente/parcialmente e os 14% de indiferentes. Ao aplicar a avaliação presencialmente no laboratório de Telemedicina, foi presenciado uma quantidade alta de dúvidas sobre como preencher o fluxo de exceção, podendo ser a causa da queda na confiança ao utilizar o sistema.

Observando o item 8 do SUS, conclui-se que a maioria dos avaliadores não achou a usabilidade do sistema poluída (71%), ou seja, sem exagero de informações na tela para confundir ou sobrecarregar o usuário.

Ao observar o item 7 do SUS, nota-se uma correlação com o item 10, pois a facilidade de aprendizagem do sistema está relacionada com o entendimento do *Use-Case* 2.0. Portanto, observa-se que os avaliadores têm alguma dúvida (36%) sobre a facilidade de aprendizagem do sistema em um contexto diferente do seu ambiente de trabalho.

No item 6 do SUS observa-se que a maioria dos avaliadores (93%) percebeu alguma inconsistência técnica ou teórica na implementação do sistema. Como essas inconsistências não foram informadas pelos avaliadores para posterior correção, não é possível identificar se houve de fato alguma inconsistência ou se o avaliador não compreendeu algum ponto da utilização do sistema ou do *Use-Case* 2.0.

No item 5 do SUS, os dados indicam que há uma concordância (79%) por parte dos avaliadores que o sistema possui suas funções bem integradas, porém 21% não souberam opinar ou ficou em dúvida se o sistema é bem integrado.

No item 4 do SUS observa-se que os avaliadores, apesar de conseguirem utilizar o sistema (65%), ainda tem dúvidas (35%) sobre algum procedimento.

Observando o item 3 do SUS nota-se que não houve dificuldade para a utilização do sistema (100% de concordância sobre a facilidade de usa o sistema). Correlacionando o item 3 com o item 4 pode-se concluir que a dificuldade maior dos avaliadores é teórica, pois utilizam o sistema facilmente, mas tem dúvidas sobre a aprendizagem do sistema (item 7) e houve um acréscimo razoável de informações aos avaliadores (item 10).

Segundo o item 2 do SUS, a maioria dos avaliadores não achou o sistema complexo (93%). Esse item possivelmente se justifica devido a familiaridade dos avaliadores com a ferramenta Kanboard.

Analisando o item 1 do SUS, nota-se uma aceitação do sistema implementado por parte dos avaliadores (79%), contudo, dois avaliadores (14%) sentiram-se indiferentes e um avaliador (7%) discorda parcialmente da utilização do sistema, dando a noção de que o sistema proposto pode ser melhorado. Como visto na seção 6.1, o SUS produz um único número representando uma medida composta da usabilidade geral do sistema em estudo. Este valor é calculado para possibilitar a verificação da usabilidade do sistema proposto de forma numérica. Para isto, é necessário valorar a escala utilizada, que recebe um valor numérico como pode ser observa no Quadro 21.

Quadro 21: Escala valorada.

Escala	Valor
Concordo totalmente	5
Concordo parcialmente	4
Indiferente	3
Discordo parcialmente	2
Discordo totalmente	1

Fonte: autor

No Quadro 22 é realizado o cálculo do valor do SUS.

Quadro 22: Cálculo SUS.

Item SUS	Cálculo do itens negativos (par)	Cálculo dos itens positivos (impar)
Item 10 / Item 09	4*(5-1) + 4*(5-2) + 5*(5-3) + 1*(5-4) = 39	4*(5-1) + 8*(4-1) + 2*(3-1) = 44
Item 08 / Item 07	10*(5-1) + 4*(5-2) = 52	3*(5-1) + 6*(4-1) + 4*(3-1) + 1*(2-1) = 39
Item 06 / Item 05	5*(5-1) + 8(5-2) + 1(5-4) = 45	6*(5-1) + 5*(4-1) + 3*(3-1) = 45
Item 04 / Item 03	4*(5-1) + 5*(5-2) + 3*(5-3) + 2*(5-4) = 39	11*(5-1) + 3*(4-1) = 53
Item 02 / Item 01	9*(5-1) + 4*(5-2) + 1*(5-3) = 50	9*(5-1) + 2*(4-1) + 2*(3-1) + 1*(2-1) = 47
Pontuação SUS	((39+52+45+39+50+44+39+45+53+47)/14)*2,5 = 80.89285714285714 ≈ 81	

Fonte: autor

A pontuação SUS varia de 0 a 100, sendo 0 o sistema está ruim e 100 está o melhor possível. O valor médio encontrado em avaliações é 68, indicando que o sistema está sem grandes problemas de usabilidade (BROOKE, 2013).

A pontuação SUS calculada para o módulo implementado, de valor aproximado de 81 pontos SUS, indica que o módulo está em boas condições no que diz respeito a usabilidade.

6.3.2 Análise da avaliação da aplicação de *Use-Case* 2.0

Em relação à avaliação da aplicação da abordagem *Use-Case* 2.0, 14 avaliadores responderam às perguntas sobre o tempo investido em compreender o funcionamento do sistema e sobre o tempo demandado para a construção de uma instância *Use-Case* 2.0 e 8 avaliadores responderam as demais perguntas. A seguir uma análise sobre cada uma das perguntas auxiliares é realizada. No Quadro 23 é possível observar os valores informados pelos avaliadores em relação as perguntas **AQ1** e **AQ2**.

Quadro 23: Tempos observados.

	Tempo de estudo para compreender o funcionamento do sistema. (em minutos)	Tempo gasto para criar um ator, um caso de uso e um *slice*. (em minutos)
Organização 1: Telemedicina	10	15
	15	15
	15	20
	10	15
	15	15
	10	5
	20	15
	4	15
Média	**12,375**	**14,375**
Organização 2	10	10
	5	10
	15	5
	15	10
	15	10
	30	12
Média	**15**	**9,5**
Media total	**13,6875**	**11,9375**

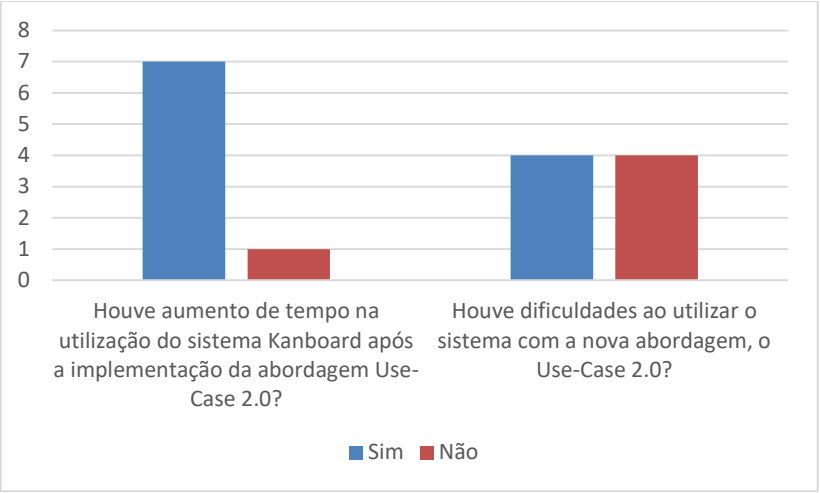
Fonte: autor

Sobre os valores obtidos, pode-se observar que o tempo para compreender o funcionamento do sistema, o que inclui o estudo sobre a abordagem *Use-Case* 2.0 e o funcionamento da abordagem no Kanboard, obteve uma média de aproximadamente 14 minutos. Após compreender o funcionamento do sistema, a criação de uma instância levou aproximadamente 12 minutos para ser concluída. Esses valores mostram que para utilizar o sistema, inicialmente o tempo médio gira em torno de 26 minutos, que possivelmente pode ser diminuído conforme os usuários ficam mais experientes na utilização do sistema.

De um modo geral, foi relatado pelos avaliadores que realizaram a avaliação presencialmente com aplicador da avaliação (7 membros da equipe do Telemedicina e 1 membro da organização 2) que o tempo médio de utilização do sistema com a abordagem implementada foi satisfatório.

As respostas para as questões **AQ3, AQ4, AQ5** e **AQ6** podem ser observadas no Apêndice III.

Gráfico 2: Histograma com as respostas das questões **AQ3** e **AQ4**.



Fonte: autor

Na questão **AQ3** sobre o aumento de tempo na utilização do sistema Kanboard após a implementação da abordagem *Use-Case* 2.0, foi observado que apenas um avaliador relatou que não houve aumento do tempo, os demais avaliadores afirmaram que houve o aumento. Esse aumento ocorreu devido as novas funcionalidades do *Use-Case* 2.0.

Na questão **AQ4** sobre as dificuldades ao utilizar o sistema com a nova abordagem, 4 avaliadores informaram não ter dificuldades. Os outros 4 avaliadores informaram dificuldades, sendo que 1 avaliador informou dificuldade de compreender o funcionamento da abordagem implementada em relação ao Kanboard e 3 avaliadores informaram ter dificuldades na criação dos *slices*.

Na questão **AQ5** sobre os benefícios da implementação para a organização, a maioria (87,5%, 7 avaliadores) dos avaliadores afirmou que a ferramenta implementada pode trazer benefícios para a unidade organizacional, porém informaram que o tempo de utilização da ferramenta foi insuficiente para obter uma análise mais crítica sobre esta pergunta.

E por fim, na questão **AQ6** os participantes informaram que as principais técnicas do *Use-Case* 2.0 que obtiveram maior aceitação foram

os casos de uso (1 voto), o diagrama de casos de uso (6 votos) e os *slices* (6 votos). Importante ressaltar que houveram avaliadores que votaram em mais de uma técnica.

6.4 DISCUSSÃO

Como pode ser observado no Gráfico 1, no cálculo do SUS, os itens com numeração ímpar, que contém afirmações positivas em relação ao sistema, apresentam um alto grau de concordância, já os itens de numeração par do SUS, que contém afirmações de teor negativo em relação ao sistema, apresentam um baixo grau de concordância. Os resultados obtidos através da escala SUS foram agrupados e são apresentados no Gráfico 1.

Ao avaliar a aplicação da abordagem *Use-Case* 2.0, nota-se que os avaliadores não gastaram uma grande quantidade de tempo significativa, para compreender o sistema. Houve uma média aproximada de 14 minutos para compreender o *Use-Case* 2.0 e seu funcionamento no Kanboard e uma média aproximada de 12 minutos para construir uma instância do *Use-Case* 2.0 com o sistema proposto. Essas informações sobre os tempos AQ1 e AQ2 em conjunto com os dados informados nos itens SUS (principalmente item 3), indicam que o tempo gasto pelos avaliadores foi aceitável, portanto, mesmo com o aumento de tempo e de complexidade informado pelos avaliadores em AQ3, as funcionalidades implementadas aparentemente contêm uma característica ágil.

Sobre a complexidade do sistema, observa-se no SUS (principalmente item 2 e 4) que houve uma pequena dificuldade para utilizar o sistema. As perguntas auxiliares (principalmente AQ4) mostram que a dificuldade observada é em relação à compreensão dos *slices* e, segundo observado pelo aplicador da avaliação, essa dificuldade aparece no momento de selecionar os fluxos para compor os *slices.*

Apesar do pouco tempo de utilização da ferramenta indicada em AQ5, é possível perceber que a avaliação obteve um resultado positivo em relação a aceitação do sistema *Use-Case* 2.0 implementado na ferramenta Kanboard por parte dos avaliadores.

6.5 AMEAÇAS A VALIDADE

Alguns pontos deste trabalho podem ter sido influenciados por fatores que podem ameaçar a validade dos resultados. O número de avaliações realizadas, o tempo de utilização do sistema e o perfil dos avaliadores podem influenciar os resultados obtidos.

Devido ao tempo disponível, não foi possível avaliar o sistema proposto com um número maior de avaliadores, impossibilitando uma precisão maior nas afirmações em relação aos dados coletados. O tempo de avaliação realizado pode caracterizar uma ameaça a validade pois os avaliadores que foram observados presencialmente utilizaram o sistema implementado uma única vez em uma faixa de tempo de 30 a 40 minutos.

As perguntas auxiliares foram adicionadas na tentativa de verificar quais pontos específicos da implementação podem ser melhorados. A ameaça a validade por parte das perguntas auxiliares se dá no fato de elas não possuírem embasamento científico comprovado de sua eficácia e no fato de alguns membros da segunda organização não terem respondido estas questões, e portanto, nem todos os membros da segunda organização responderam todas as perguntas.

Em relação a usabilidade do sistema, os componentes utilizados para informar os fluxos dos casos de uso e dos *slices* são componentes de texto. Estes componentes de texto impossibilitam uma seleção, de maneira ágil, dos fluxos do caso de uso para formar um *slice*, influenciando na característica ágil do sistema proposto. Outra possível ameaça a validade encontrada no sistema proposto é que há a possibilidade de modificação das histórias e dos fluxos dos *slices*, o que pode acarretar em *slices* inconsistentes com seu caso de uso.

7 CONCLUSÃO

Neste trabalho é apresentado o desenvolvimento de um módulo para o sistema de gerenciamento de projetos Kanboard, possibilitando a utilização da abordagem *Use-Case* 2.0. O objetivo principal deste trabalho foi desenvolver um módulo que possibilitasse a utilização de casos de uso no Kanboard segundo a ótica da abordagem *Use-Case* 2.0 e avaliar a aceitação por parte dos interessados.

Neste contexto, foi realizado um estudo sobre a teoria por trás da abordagem *Use-Case* 2.0 e feita a análise da fundamentação teórica. Também foi realizada a análise do estado da arte identificando a ausência de uma implementação da abordagem *Use-Case* 2.0 em uma ferramenta para gerência de projeto.

Desse modo, baseando-se na fundamentação teórica e na análise do estado da arte, foi elaborado um modelo conceitual incluindo protótipos de baixa fidelidade e posteriormente a modelagem e implementação do módulo proposto.

O módulo desenvolvido foi avaliado e os dados coletados desta avaliação foram analisados. A partir da análise realizada, conclui-se que o módulo obteve uma recepção positiva entre os avaliadores. A usabilidade mostrou-se intuitiva e de fácil utilização e a aplicação do *Use-Case* 2.0 mostrou-se aceitável de ser utilizado.

Para trabalhos futuros, recomenda-se focar na melhoria da usabilidade no que diz respeito à utilização dos fluxos de um caso de uso para a construção de um *slice*, agilizando a criação de *slices*.

Ainda como trabalho futuro, seria interessante a migração do módulo desenvolvido para a forma de um *plug-in*, que não foi implementado neste trabalho por limitações de tempo, mas tornaria o módulo mais flexível de ser instalado em outros ambientes.

REFERÊNCIAS

BOEG, Jesper. **Priming Kanban**. InfoQ/Trifork, 2012.

BOOCH, Grady et al. **The unified modeling language**. Unix Review, v. 14, n. 13, p. 5, 1996.

BROOKE, John et al. **SUS-A quick and dirty usability scale. Usability evaluation in industry**, v. 189, n. 194, p. 4-7, 1996.

BROOKE, John. **SUS: a retrospective. Journal of usability studies**, v. 8, n. 2, p. 29-40, 2013.

COCKBURN, Alistair. **Writing Effectives Use Cases**. Boston: Addison-Wesley, 2001.

FOWLER, Martin; HIGHSMITH, Jim. **The agile manifesto. Software Development**, v. 9, n. 8, p. 28-35, 2001.

GENMYMODEL. Disponível em <https://dashboard.genmymodel.com>. Acesso em 2 de novembro de 2017.

GERHARDT, Tatiana Engel; SILVEIRA, Denise Tolfo. **Métodos de pesquisa**. Plageder, 2009.

GUEDES, Gilleanes TA.UML 2-**Uma abordagem prática**. Novatec Editora, 2018.

ISO, IEC. **IEEE. 29148: 2011-Systems and software engineering - Requirements engineering.** Technical report, 2011.

JACOBSON, Ivar; SPENCE, Ian; BITTNER, Kurt. **Use Case 2.0: The guide to succeeding with use cases**. Ivar Jacobson International, 2011.

KANBAN PROJECT MANAGEMENT SOFTWARE, 2017. Disponível em <https://kanboard.net>. Acesso em 16 de agosto de 2017.

KARNER, Gustav. **Resource estimation for objectory projects. Objective Systems**. SF AB, v. 17, 1993.

KAUR, Rupinder; SENGUPTA, Jyotsna. **Software Process Models and Analysis on Failure of Software Development Projects**. International Journal of Scientific & Engineering Research. [s.i.], p. 1-4. fev. 2011.

KITCHENHAM, Barbara. **Procedures for performing systematic reviews**. Keele, UK, Keele University, v. 33, n. 2004, p. 1-26, 2004.

LEWIS, James R.; SAURO, Jeff. **The factor structure of the system usability scale. In:International conference on human centered design**. Springer, Berlin, Heidelberg, 2009. p. 94-103.

LUCIDCHART. Disponível em <https://www.lucidchart.com>. Acesso em 2 de novembro de 2017.

OHNO, Taiichi. **Toyota production system: beyond large-scale production**. crc Press, 1988.

POHL, Klaus. **Requirements engineering fundamentals: a study guide for the certified professional for requirements engineering exam-foundation level-IREB compliant**. Rocky Nook, Inc., 2016.

PRESSMAN, Roger S. **Engenharia de software: uma abordagem profissional**. 7ª Edição. Ed: McGraw Hill, 2011.

STARUML. Disponível em <http://staruml.io>. Acesso em 2 de novembro de 2017

SUGIMORI, Y. et al. **Toyota production system and kanban system materialization of just-in-time and respect-for-human system**. The International Journal of Production Research, v. 15, n. 6, p. 553-564, 1977.

UML, OMG. **Unified Modelling Language** version 2.5. Unified Modelling (2017). 2017.

VISUAL PARADIGM - LEADING UML, BPMN, EA, AGILE AND PROJECT MANAGEMENT SOFTWARE. Disponível em <https://www.visual-paradigm.com>. Acesso em 2 de novembro de 2017.

WIEGERS, Karl; BEATTY, Joy. **Software requirements**. Pearson Education, 2013.

APÊNDICE I – Detalhamento dos casos de uso

Agrupamento AG01: permitir criar artefatos de *Use-Case 2.0*

Caso de uso AG01-UC01: criar ator

Ator: analista

Pré-condição:

1. Usuário autenticado no sistema;
2. Acessar um projeto;
3. Acessar “Settings”.

Fluxo principal:

1. Clicar na opção “Actors”;
2. Clicar em “Add new actor”;
3. Digitar o nome do ator;
4. Clicar em “Save”.

Fluxo alternativo 01:

1. Deriva do passo 4
2. Se houver ator com o mesmo nome, mostra mensagem informando que já existe ator com o nome informado.

Pós-condição: há um novo ator cadastrado no sistema, a lista de atores está atualizada e os casos de uso podem acessar os novos atores.

Pós-condição do fluxo alternativo 01: o novo ator não foi cadastrado no sistema.

***Slice* AG01-UC01-SL01**

História de usuário HU01: como analista eu quero criar um ator para vinculá-lo aos casos de uso.

***Slice* AG01-UC01-SL02**

História de usuário HU02: como analista eu quero ser alertado se tentar replicar o nome de um ator já existente.

Caso de uso AG01-UC02: editar ator

Ator: analista

Pré-condição:

1. Usuário autenticado no sistema;
2. Acessar um projeto;
3. Acessar “Settings”;
4. Haver pelo menos um ator cadastrado no sistema.

Fluxo principal:

1. Clicar em “Actors”;
2. Clicar na engrenagem ao lado do nome do ator;
3. Selecionar a opção “Edit”;
4. Modificar o nome do ator;

5. Clicar em “Save”.

Fluxo alternativo 01:

1. Deriva do passo 5:
2. Se houver ator com o mesmo nome, mostra mensagem informando que já existe ator com o nome informado.

Pós-condição: nome do ator modificado.

Pós-condição do fluxo alternativo 01: a modificação do nome do ator não foi concluída.

***Slice* AG01-UC02-SL01**

História de usuário HU01: como analista eu quero alterar o nome de um ator.

***Slice* AG01-UC02-SL02**

História de usuário HU02: como analista eu quero ser alertado se tentar replicar o nome de um ator já existente.

Caso de uso AG01-UC03: excluir ator

Ator: analista

Pré-condição:

1. Usuário autenticado no sistema;
2. Acessar um projeto;
3. Acessar “Settings”;
4. Haver pelo menos um ator cadastrado no sistema.

Fluxo principal:

1. Clicar em “Actors”;
2. Clicar no ícone da engrenagem ao lado do nome do ator;
3. Selecionar a opção “Remove”;
4. Exibir mensagem de confirmação de exclusão.

Fluxo alternativo 01:

1. Sequente ao passo 3;
2. Tratar os casos de uso que estiverem associados ao ator removido.
3. Continua o passo 4.

Pós-condição: ator removido do sistema, dos casos de uso associados e da lista de atores.

Pós-condição do fluxo alternativo 01: ator não removido.

***Slice* AG01-UC03-SL01**

História de usuário HU01: como analista eu quero deletar o nome de um ator.

***Slice* AG01-UC03-SL02**

História de usuário HU02: como analista eu quero que os casos de uso que não tiverem nenhum outro ator que não seja o que está em processo de exclusão também seja excluído.

Caso de uso AG01-UC04: criar caso de uso

Ator: analista

Pré-condição:

1. Usuário autenticado no sistema;
2. Acessar um projeto;

Fluxo principal:

1. Clicar em "+" para adicionar uma tarefa;
2. Digitar o título;
3. Selecionar o ator;
4. Selecionar como categoria a opção "Use case";
5. Clicar em "Save".

Fluxo alternativo 01:

1. Clicar em "Menu";
2. Clicar em "Add a new task";
3. Digitar o título;
4. Selecionar o ator;
5. Selecionar como categoria a opção "Use case";
6. Clicar em "Save".

Fluxo alternativo 02:

1. Entre os passos 1 e 4;
2. Digitar o texto;
3. Continua o passo 5.

Fluxo alternativo 03:

1. Entre os passos 1 e 4;
2. Selecionar um ou mais rótulos;
3. Continua o passo 5:

Pós-condição: novo caso de uso cadastrado no sistema.

***Slice* AG01-UC04-SL01**

História de usuário HU01: como analista gostaria de ter a opção de poder cadastrar um caso de uso.

***Slice* AG01-UC04-SL02**

História de usuário HU02: como analista gostaria de ter a opção de digitar um texto com informações sobre o caso de uso.

***Slice* AG01-UC04-SL03**

História de usuário HU03: como analista gostaria de ter a opção de selecionar um ou mais rótulos (*tag*) para a identificação do caso de uso.

Caso de uso AG01-UC05: editar caso de uso

Ator: analista
Pré-condição:
1. Usuário autenticado no sistema;
2. Acessar um projeto;
3. Acessar um caso de uso.

Fluxo principal:
1. Clicar em “Edit”;
2. Abre tela de edição;
3. Clicar em “Save”.

Fluxo alternativo 01:
1. Sequente ao passo 2;
2. Modificar título;
3. Continua o passo 3.

Fluxo alternativo 02:
1. Sequente ao passo 2;
2. Selecionar ator;
3. Continua o passo 3.

Fluxo alternativo 03:
1. Sequente ao passo 2;
2. Digitar o texto;
3. Continua o passo 3.

Fluxo alternativo 04:
1. Sequente ao passo 2;
2. Selecionar um ou mais rótulos;
3. Continua o passo 3.

Pós-condição: caso de uso teve alterações realizadas e salvas.

***Slice* AG01-UC05-SL01**
História de usuário HU01: como analista gostaria de ter a opção de editar um caso de uso.

***Slice* AG01-UC05-SL02**
História de usuário HU02: como analista gostaria de ter a opção de editar o título de um caso de uso.

***Slice* AG01-UC05-SL03**
História de usuário HU03: como analista gostaria de ter a opção de trocar o ator de um caso de uso.

***Slice* AG01-UC05-SL04**
História de usuário HU04: como analista gostaria de ter a opção de editar um texto com informações sobre o caso de uso.

Caso de uso AG01-UC06: excluir caso de uso
Ator: analista

Pré-condição:
1. Usuário autenticado no sistema;
2. Acessar um projeto;
3. Acessar um caso de uso.

Fluxo principal:
1. Clicar em “Delete”;
2. Abre tela para confirmar exclusão;
3. Clicar em “Yes”.

Fluxo alternativo 01:
1. Sequente ao passo 3;
2. Retirar a referência de atores e casos de uso associados ao caso de uso excluído.

Fluxo alternativo 02:
1. Sequente ao passo 3;
2. Excluir os *slices*.

Pós-condição: o caso de uso é removido do sistema, perde a referência com os atores e os casos de uso. *Slices* associados são excluídos.

***Slice* AG01-UC06-SL01**
História de usuário HU01: como analista gostaria de poder excluir um caso de uso.

***Slice* AG01-UC06-SL02**
História de usuário HU02: como analista gostaria de que os atores e outros casos de uso associados deixem de referenciar o caso de uso deletado.

***Slice* AG01-UC06-SL03**
História de usuário HU03: como analista gostaria de que os *slices* associados a este caso de uso sejam excluídos.

Caso de uso AG01-UC07: criar um *slice*
Ator: analista
Pré-condição:
1. Usuário autenticado no sistema;
2. Acessar um projeto;
3. Acessar um caso de uso.

Fluxo principal:
1. Clicar em “Add slice”;
2. Abre tela para entrada das informações;
3. Digitar título;
4. Digitar *stories* do *slice*;
5. Clicar em “Save”.

Pós-condição: *slice* registrado no sistema e associado ao caso de uso onde o *slice* foi criado. *Slice* na lista de *slices* do caso de uso.

***Slice* AG01-UC07-SL01**
História de usuário HU01: como analista gostaria de adicionar *slices* a um caso de uso.

Caso de uso AG01-UC08: editar um *slice*
Ator: analista
Pré-condição:
1. Usuário autenticado no sistema;
2. Acessar um projeto;
3. Acessar um caso de uso;
4. Acessar um *slice*.

Fluxo principal:
1. Clicar em "Edit";
2. Abre tela para entrada das informações;
3. Modificar título;
4. Clicar em "Save".

Fluxo alternativo 01:
1. Sequente ao passo 2;
2. Modificar *stories*;
3. Continua o passo 3 ou 4.

Pós-condição: alterações no *slice* salvas.

***Slice* AG01-UC08-SL01**
História de usuário HU01: como analista gostaria de alterar o título e as *stories* do *slice*.

Caso de uso AG01-UC09: excluir um *slice*
Ator: analista
Pré-condição:
1. Usuário autenticado no sistema;
2. Acessar um projeto;
3. Acessar um caso de uso;
4. Acessar um *slide*.

Fluxo principal:
1. Clicar em "Delete";
2. Abre tela para confirmar exclusão;
3. Clicar em "Yes".

Pós-condição: *slice* e seu conteúdo deixam de existir no sistema. *Slice* é desassociado e sai de lista de *slices* do caso de uso.

***Slice* AG01-UC09-SL01**

História de usuário HU01: como analista gostaria de excluir o *slice* associado ao caso de uso.

Agrupamento AG02: permitir a visualização do *Use-Case* 2.0 no sistema.

Caso de uso AG02-UC01: visualizar o diagrama de casos de uso de um projeto.

Ator: analista

Pré-condição:

1. Usuário autenticado no sistema;
2. Acessar um projeto.

Fluxo principal:

1. Clicar em "Use case";
2. Abre o diagrama de casos de uso, suas associações e seus *slices*;

Pós-condição: visualização do diagrama de casos de uso, suas associações e seus *slices*.

***Slice* AG02-UC01-SL01**

História de usuário HU01: eu como analista/programador quero visualizar o diagrama dos casos de uso de um projeto. O diagrama deve ter as associações entre atores, casos de uso e *slices*.

Caso de uso AG02-UC02: visualizar detalhes dos componentes do diagrama.

Ator: analista

Pré-condição:

1. Usuário autenticado no sistema;
2. Acessar um projeto;
3. Abrir diagrama de casos de uso.

Fluxo principal:

1. Duplo clique sobre um caso de uso;
2. Abre a tela de edição do caso de uso.

Fluxo alternativo 01:

1. Duplo clique sobre um *slice*;
2. Abre a tela de edição do *slice*.

Pós-condição: tela com as informações do componente selecionado aberta.

***Slice* AG02-UC02-SL01**

História de usuário HU01: eu como analista/programador quero poder acessar as informações de um caso de uso através do diagrama de casos de uso.

***Slice* AG02-UC02-SL02**

História de usuário HU02: eu como analista/programador quero poder acessar as informações de um *slice* através do diagrama de casos de uso.

Caso de uso AG02-UC03: filtrar casos de uso e *slices* no quadro Kanban.

Ator: analista

Pré-condição:

1. Usuário autenticado no sistema;
2. Acessar um projeto;
3. Abrir o quadro Kanban clicando na aba “Board”.

Fluxo principal:

1. Clicar sobre o símbolo do filtro;
2. Selecionar entre os filtros disponíveis a opção desejada para filtrar casos de uso.

Fluxo alternativo 01:

1. Selecionar entre os filtros disponíveis a opção desejada para filtrar *slices*.

Fluxo alternativo 02:

1. Selecionar entre os filtros disponíveis a opção desejada para filtrar casos de uso e *slices*.

Pós-condição: tela com os *slices* a mostra ou escondidos.

***Slice* AG02-UC03-SL01**

História de usuário HU01: eu como analista/programador quero poder filtrar os casos de uso e *slices* existentes.

APÊNDICE II – Tutorial para utilizar o Kanboard com *Use-Case* 2.0

Utilizando o Use Case 2.0 implementado na ferramenta Kanboard

O que é o Use Case 2.0?
Uma prática escalável e ágil que usa casos de uso e slices para capturar um conjunto de requisitos e impulsionar o desenvolvimento incremental de um sistema para atendê-los.

O que é um use case (caso de uso)?
Um caso de uso compreende todas as formas de usar um sistema para atingir uma meta específica para um usuário específico. Um caso de uso contém:
a) **Stories**: as stories cobrem como alcançar com sucesso um objetivo e como lidar com quaisquer problemas que possam ocorrer no caminho.
b) **Basic flow**: é o principal caminho, apresentado pelas stories do caso de uso, para alcançar o objetivo.
c) **Alternative flow**: alternative flows identificam quaisquer outras formas de usar o sistema para atingir o objetivo, quaisquer recursos opcionais que possam ser oferecidos ao usuário, como a distribuição de um recibo.
d) **Exception flow**: exception flow é um caminho não intencional através do sistema, geralmente como resultado da falta de informações ou problemas de disponibilidade do sistema. Problemas que possam ocorrer no caminho, como um cartão ficar preso representam um caminho indesejável para o usuário.
e) **Actors**: um actor define um papel que um usuário pode desempenhar ao interagir com o sistema. Um usuário pode ser um indivíduo ou outro sistema. Os atores estão associados aos casos de uso com os quais eles interagem.
f) **Slice**: um slice é uma fatia de um caso de uso contendo uma ou mais histórias selecionadas (de um caso de uso) para formar um item de trabalho de valor para o cliente.

Um slice contém:
a) **Selected stories**: histórias selecionadas de stories de um caso de uso.
b) **Basic, alternative and exception flow**: mesma descrição do caso de uso.
c) **Test cases**: testes identificados a serem realizados no sistema.

Para utilizar o Use Case 2.0, primeiro é necessário cadastrar um ator. Em todos os passos é necessário estar dentro de um projeto.

1. **Cadastrar ator**:
a. Clique em Menu → Settings → Actors → Add new actor;
b. Digite um nome e salve;

Em seguida já é possível criar um caso de uso e seus respectivos slices.

1. **Criar caso de uso**:
a. Clique em Menu → Add a new use case;
b. Preencha a história e os fluxos;
c. Preencha o nome e adicione um ator (obrigatórios);
d. Salve.

2. **Criar slice**:
a. Clique em cima de um caso de uso existente;
b. Clique em Edit → Add slice;
c. Preencha com a(s) história(s) selecionada(s) do caso de uso, preencha os fluxos e informe os casos de teste;
d. Salve.

Após os cadastros, é possível verificar o diagrama de casos de uso na aba use case.

Há duas formas para **editar um caso de uso ou um slice**:
1. Clicar em um caso de uso ou slice e em seguida clicar em Edit;
2. Ir no diagrama de casos de uso, clicando na aba Use Case, e clicando 2x sobre um caso de uso ou um slice.

Como é possível adicionar tickets Kanban, pode ser que haja a necessidade de **filtrar apenas casos de uso ou slices**, então é possível filtrar clicando no combobox ao lado do filtro escrito e selecionar:
1. Use case filter;
2. Slice filter;
3. Use case and slice filter.

Notas e dicas:
1. Um caso de uso não pode acabar antes de seus slices, então não é possível mover os casos de uso uma coluna a frente de seus próprios slices.

2. Da mesma forma não é possível mover um slice para uma coluna antes do seu caso de uso.
3. Os slices simplificam o caso de uso, dividindo-o para torná-lo mais maleável e ajudar a tarefa a fluir melhor, então sempre use os slices para fatiar os casos de uso.
4. O código com o Kanboard modificado se encontra em *<https://github.com/PauloJRN/tcc>*

APÊNDICE III – Respostas das questões auxiliares

Avaliador 1

AQ3. Não;
AQ4. A nova abordagem foi intuitiva e fácil de compreender, por isso não houveram dificuldades em relação ao antecessor;
AQ5. Sim, pois podemos dividir tarefas maiores em conjuntos de tarefas menores mais fáceis de gerenciar e manter;
AQ6. Slices;

Avaliador 2

AQ3. sim
AQ4. sem dificuldades
AQ5. sim, apesar de pouco tempo
AQ6. Diagrama de casos de uso

Avaliador 3

AQ3. Sim, devido ao processo de preenchimento dos dados dos casos de uso de acordo com a nova abordagem.
AQ4. Entender o funcionamento do plugin e sua relação com o restante da ferramenta de gerência de projetos.
AQ5. Como a abordagem ainda não foi incorporada para uso cotidiano junto ao Kanboard, não é possível avaliar seu real benefício.
AQ6. Casos de uso e diagrama de casos de uso.

Avaliador 4

AQ3. sim, mas pouco tempo
AQ4. a possibilidade de divisão dos casos de usos em slices dentro do kanboard
AQ5. sim
AQ6. slices e diagrama de casos de uso

Avaliador 5

AQ3. Sim, contudo o overhad de tempo não parece relevante
AQ4. Encontrar a granularidade ótima dos slices
AQ5. Sim. Contudo a validade do experimento é limitada devido ao tempo de aplicação e falta de métrica de sucesso.
AQ6. Slices.

Avaliador 6

AQ3. Sim

AQ4. Dificuldade em criar os slices
AQ5. Sim, porém o tempo de utilização pode não ter sido suficiente.
AQ6. slices e diagrama

Avaliador 7

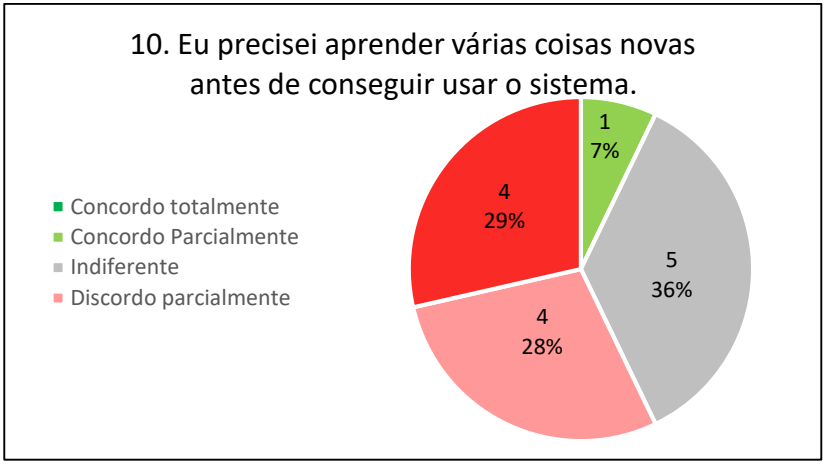
AQ3. Houve um aumento do tempo de utilização, mas foi devido a introdução de novas funcionalidades. Não houve aumento da dificuldade ou complexidade da plataforma.
AQ4. Não tive grandes dificuldades. Após entender os conceitos e a funcionalidade, a utilização foi simples e intuitiva.
AQ5. Foi benéfica pois facilitou a criação de casos de uso, o entendimento dos mesmos e a distribuição de tarefas.
AQ6. As tarefas que mais obtiveram aceitação de minha parte foram os slides e o diagrama de casos de uso.

Avaliador 8

AQ3. Sim
AQ4. nao tive dificuldade
AQ5. sim
AQ6. slices e diagrama

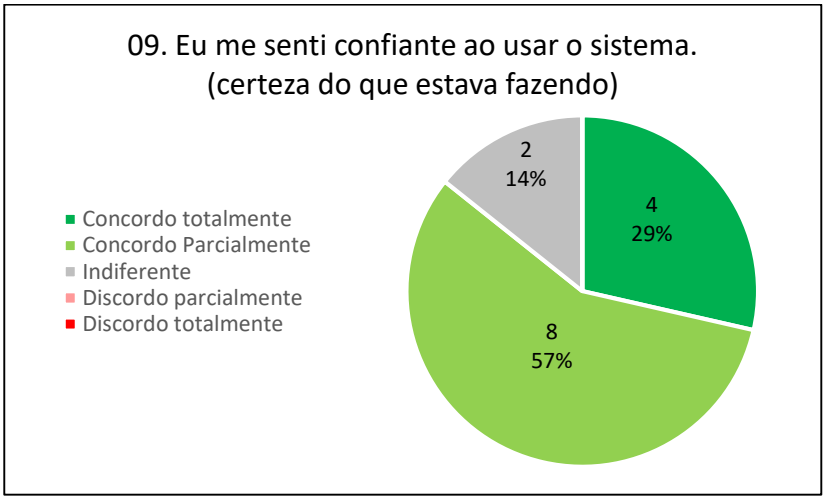
APÊNDICE IV – Gráficos com os resultados da avaliação por item SUS

Gráfico 3: Item 10 do SUS



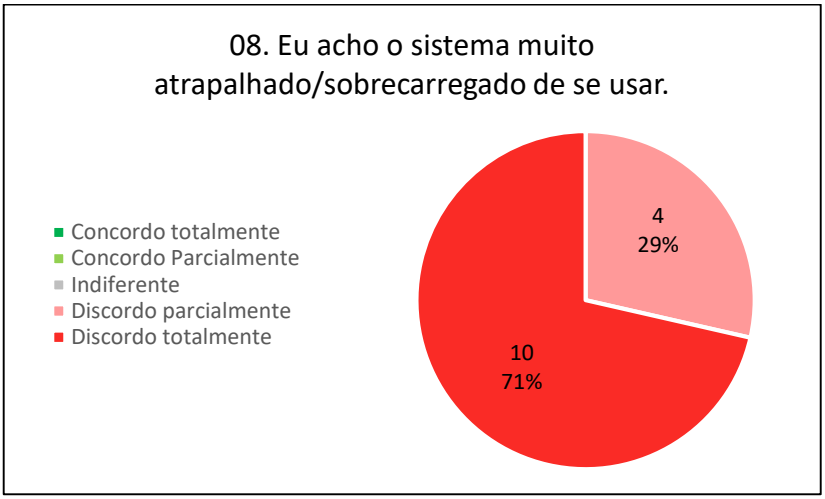
Fonte: autor

Gráfico 4: Item 9 do SUS



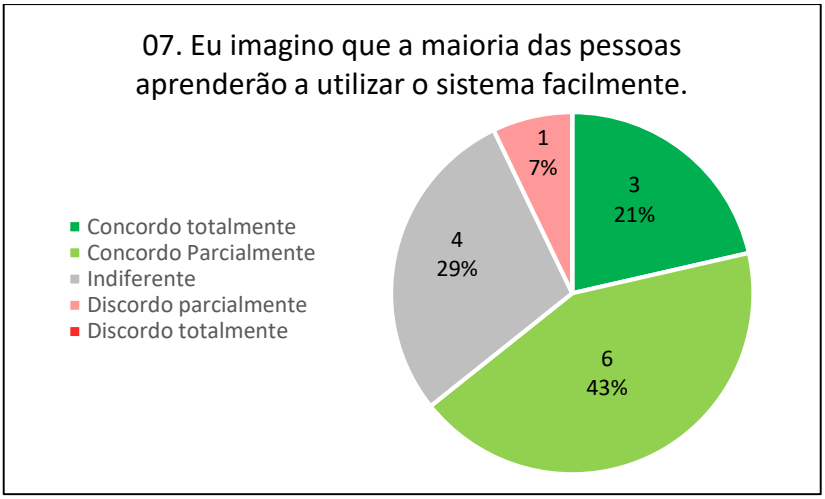
Fonte: autor

Gráfico 5: Item 8 do SUS



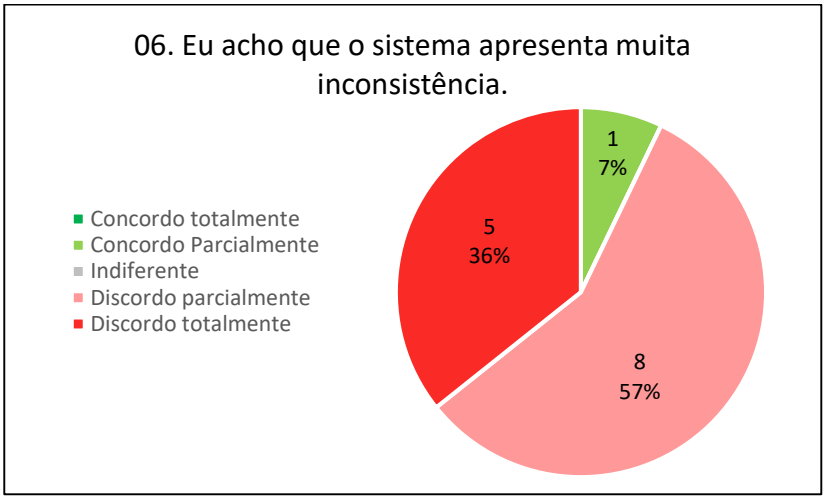
Fonte: autor

Gráfico 6: Item 7 do SUS.



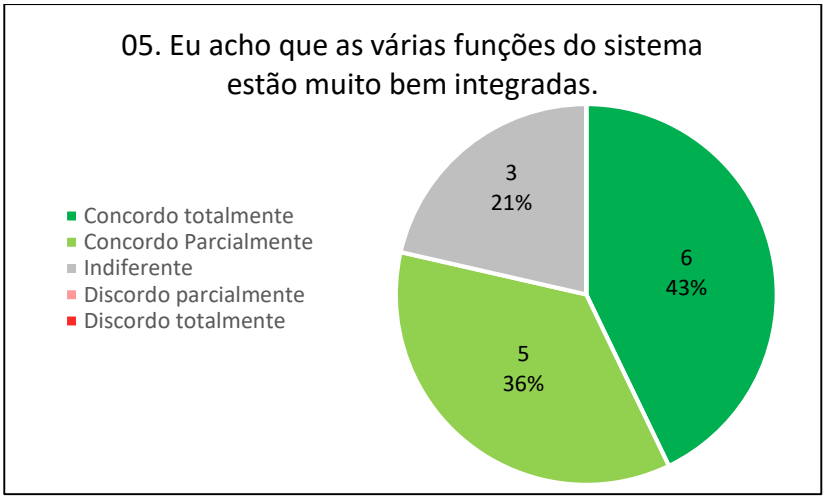
Fonte: autor

Gráfico 7: Item 6 do SUS.



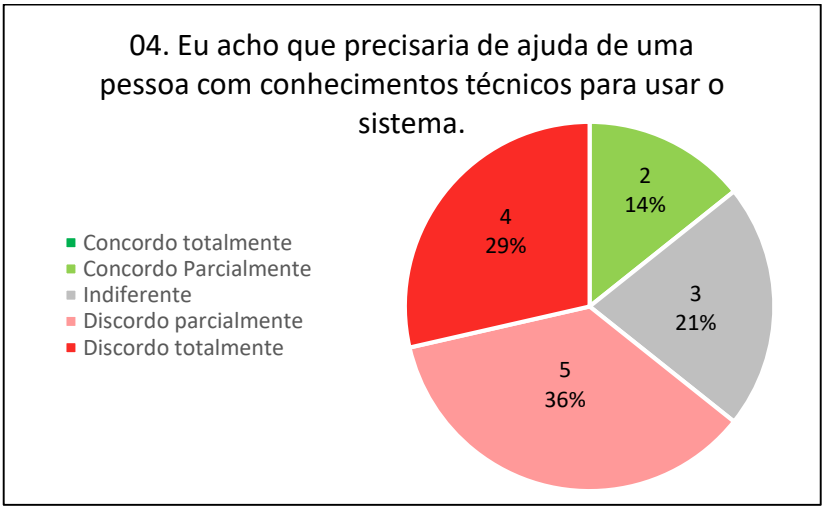
Fonte: autor

Gráfico 8: Item 5 do SUS.



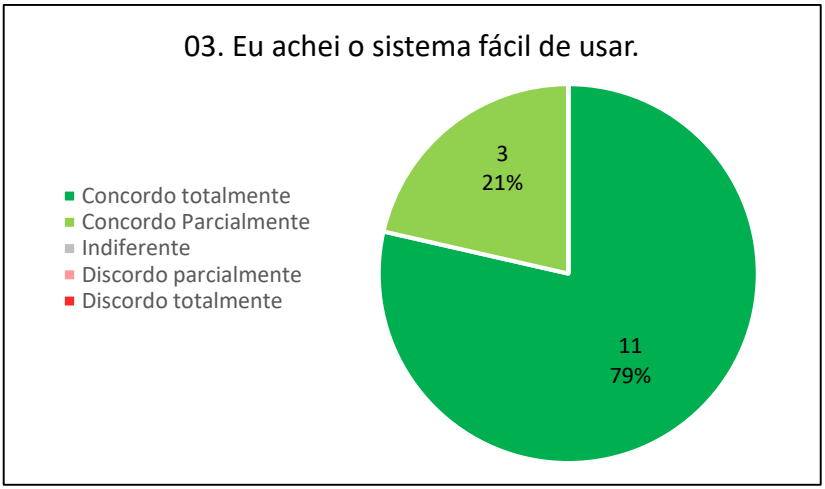
Fonte: autor

Gráfico 9: Item 4 do SUS.



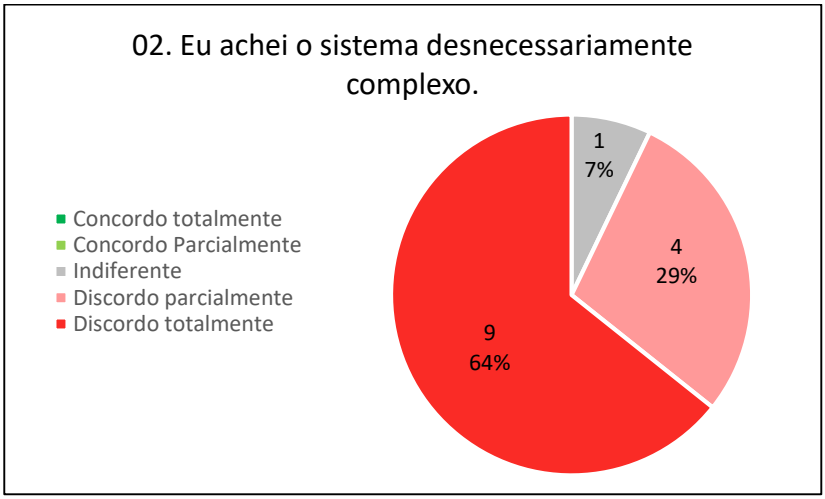
Fonte: autor

Gráfico 10: Item 3 do SUS.



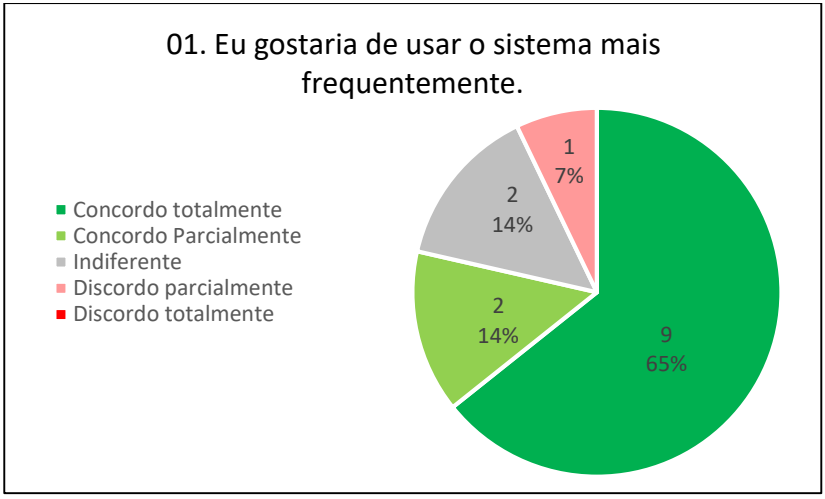
Fonte: autor

Gráfico 11: Item 2 do SUS.



Fonte: autor

Gráfico 12: Item 1 do SUS.



Fonte: autor

APÊNDICE V – Artigo

Implementação de um módulo para a integração da abordagem Use-Case 2.0 com um Sistema de Gerenciamento de Projetos

Paulo João Rodrigues Neto[1], Jean Carlo Rossa Hauck[1], João Marcus Alves[1]

[1]Departamento de Informática e Estatística– Universidade Federal de Santa Catarina (UFSC) – Florianópolis – SC – Brasil

paulo.xcco@gmail.com, jean.hauck@ufsc.br, joaomarcalves@gmail.com

Abstract. *In general, software development companies have difficulty in delivering or completing the scope of the product idealized by the customer. One of the main reasons is related to requirements engineering, affecting costs and deadlines. Agile and iterative practices enable developers to adapt to the changes required by the customer or identify requirements that are not understood quickly. One approach currently used for requirements collection is Use-Case 2.0, which extends classic use cases to an agile and iterative context. Kanboard is a tool that implements Kanban, an approach to activity management that follows agile values complementary to Use-Case 2.0, being agile and iterative. Analyzing this scenario, it is proposed the development of a module that models Use-Case 2.0 and integrates it with the Kanboard project management platform.*

Resumo. *De forma geral, as empresas desenvolvedoras de software têm dificuldade para*

entregar ou concluir o escopo do produto idealizado pelo cliente. Um dos principais motivos é relacionado à engenharia de requisitos, que desencadeia alterações, afetando custos e prazos. As práticas ágeis e iterativas possibilitam às empresas desenvolvedoras adaptar-se às mudanças requisitadas pelo cliente ou identificar requisitos não compreendidos com rapidez. Uma abordagem utilizada atualmente para a coleta de requisitos é a Use-Case 2.0, que estende os casos de uso clássicos para um contexto ágil e iterativo. O Kanboard é uma ferramenta que implementa o Kanban, uma abordagem para a gerência de atividades que segue valores ágeis complementares ao Use-Case 2.0, sendo ágil e iterativo. Analisando este cenário, é proposto o desenvolvimento de um módulo que modele o Use-Case 2.0 e o integre com a plataforma de gerência de projeto Kanboard.

1. Introdução

Aproximadamente 70% dos produtos de *software* falham, e um dos principais motivos é a baixa qualidade na engenharia de requisitos [Kaur e Sengupta 2011]. Requisito, segundo a *Institute of Electrical and Electronics Engineers*, é definido como uma declaração que representa uma necessidade com suas restrições e condições associadas [IEEE-29148 2011].

Nesse contexto, uma engenharia de requisitos de qualidade é necessária para minimizar problemas que implicam em consequências indesejadas no desenvolvimento de *software*. Assim, por meio de atividades como: elicitação, desenvolvimento, análise, verificação, validação, comunicação, documentação e gerência de requisitos, uma hierarquia de requisitos é gerada [IEEE-29148 2011]. Esta hierarquia de requisitos possibilita a validação e a compreensão dos requisitos entre os *stakeholders* — indivíduos ou organizações com

interesse em um sistema de acordo com suas necessidades e expectativas. A validação dos requisitos em relação às necessidades reais dos *stakeholders* reforça a compreensão dos requisitos e fornece uma base que facilita o projeto do sistema [IEEE-29148, 2011].

Existem diversas abordagens e técnicas que auxiliam a engenharia de requisitos. Dentre elas, a abordagem de casos de uso é utilizada para a coleta, modelagem e gerenciamento dos requisitos dos usuários [Pressman, 2007]. Nesse contexto, um caso de uso representa todos os modos de utilização do sistema para atingir os objetivos específicos de um determinado ator, portanto, o conjunto dos casos de uso abrange todos os caminhos possíveis para a utilização do sistema e o valor que esse sistema fornecerá [Jacobson; Spence e Bittner 2011].

Uma modernização da abordagem dos casos de uso, chamada de *Use-Case* 2.0, surgiu grande parte em resposta à necessidade de envolver práticas ágeis na engenharia de requisitos. Com o advento das abordagens ágeis na engenharia de *software*, a forma de coletar, modelar, analisar e gerenciar requisitos tem mudado, focando em *software* eficiente no lugar de uma documentação extensa, colaboração com o cliente e resposta a mudanças [Fowler e Highsmith 2001]. O *Use-Case* 2.0 é uma prática escalável, que usa casos de uso para capturar um conjunto de requisitos, guia o desenvolvimento incremental do sistema, não necessita de uma documentação preditiva extensa e enquadra-se perfeitamente na mentalidade ágil [Jacobson; Spence e Bittner 2011].

Essa mentalidade ágil e incremental vem sendo aplicada na gerência de projetos do laboratório de Telemedicina do Instituto Nacional para Convergência Digital (INCoD) vinculado à Universidade Federal de Santa Catarina (UFSC). O Laboratório de Telemedicina, através do Sistema Integrado Catarinense de Telemedicina e Telessaúde, encurta a distância entre pacientes e provedores de saúde. Desde 2005, esse sistema

já emitiu mais de 5 milhões de laudos de várias modalidades de diagnóstico.

O sistema de gerenciamento de projeto Kanboard, que implementa a abordagem Kanban, vem sendo adotado no contexto do laboratório de Telemedicina e tem proporcionado melhoria significativa nos prazos de entregas das tarefas do laboratório. Como ideia central, idelizou-se a implementação da abordagem *Use-Case* 2.0 na ferramenta de gerenciamento de projetos Kanboard para adicionar um nivel maior de detalhamento sobre as tarefas. Em seguida, avaliações e análises foram realizadas no intuito de verificar se a coleta, a análise e a manutenção dos requisitos obtiveram melhorias significativas e aceitação por parte dos avaliadores.

2. Fundamentação teórica

2.1. Casos de uso

Avaliando mais precisamente, Pohl (2016) define um requisito de *software* de duas formas: uma condição ou capacidade de um usuário resolver um problema ou alcançar um objetivo; ou uma condição que precisa ser atingida, ou possuída, por um sistema (ou módulo do sistema), para satisfazer um contrato, padrão, especificação ou outro documento formal. Consequentemente, Pohl (2016) apresenta o processo de engenharia de requisitos composto de 4 atividades fundamentais:

- Elicitação: diferentes técnicas usadas para obter os requisitos de *stakeholders* e de outras fontes;
- Documentação: descrição adequada dos requisitos obtidos na atividade de elicitação. Diferentes técnicas podem ser usadas para documentar os requisitos, usando linguagem natural ou modelos conceituais.
- Validação e negociação: para garantir coerência, os requisitos devem ser negociados e validados pelos stakeholders.

• Gerenciamento de requisitos: atividades necessárias para estruturar requisitos, prepará-los para que possam ser compreendidos por diferentes pessoas com diferentes responsabilidades, e principalmente manter consistência depois de mudanças para garantir a implementação adequada.

Requisitos de *software* licitados podem ser documentados de diversas formas como, histórias de usuário ou casos de uso por exemplo [Wiegers e Beatty 2013]. Nesse cenário, uma técnica importante para a análise de requisitos que tem sido largamente aplicada no desenvolvimento de *software*, são os casos de uso.

Dentro da elicitação, uma das 4 atividades fundamentais da engenharia de requisitos, os requisitos podem ser obtidos através de *stories* — narrativa que apresenta formas de atingir um objetivo e lidar com os problemas que podem ocorrer no caminho — e inseridas nas narrativas de caso de uso para completar a descrição dos casos de uso, tornando os requisitos acessíveis e testáveis [Jacobson, Spence e Bittner 2011].

O conjunto de casos de uso forma o modelo de casos de uso, que compreende o contexto para que os requisitos do sistema sejam descobertos, compartilhados e compreendidos. No modelo de casos de uso, os *stakeholders* que usam o sistema e contribuem para completar os objetivos, são modelados como atores, e os caminhos para alcançar esses objetivos são modelados como casos de uso [Jacobson, Spence e Bittner 2011].

Classicamente um caso de uso deve descrever um fluxo principal de atividades de um ator, baseado em um conjunto de premissas, possíveis fluxos alternativos e pós-condições esperadas após a conclusão do caso de uso [Cockburn 2001]. Uma ferramenta gráfica para ilustrar casos de uso é o diagrama UML de casos de uso. Neste diagrama, um ator é relacionado a um conjunto de casos de uso que pode incluir ainda mais informações com o uso de notações UML. Por exemplo, a Figura 1 mostra a notação *include*, que implica que o caso de uso

"Inserir TFD" (Tratamento Fora de Domicílio) e "Inserir exame" tem um fluxo de atividades em comum, que é representado no caso de uso "Inserir paciente". Da mesma maneira, há um fluxo de atividades opcionais em "Inserir exame", que é representado no caso de uso "Invalidar exame".

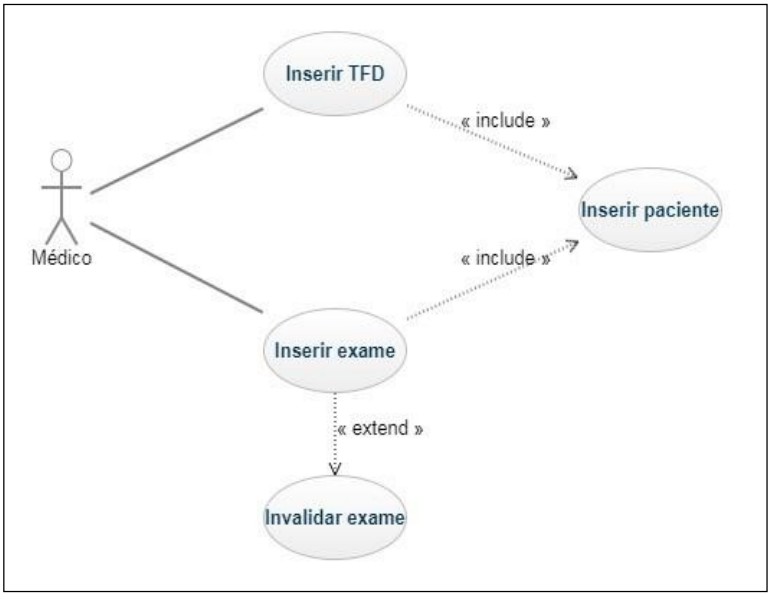


Figura 25: Caso de uso clássico

2.2. *Use-Case* 2.0

Em 2011, Jacobson, Spence e Bittner apresentam uma extensão da técnica de casos de uso, chamada *Use-Case* 2.0, e a define como uma prática leve, escalável, ágil, versátil e fácil de usar. Essa abordagem utiliza-se dos casos de uso para capturar um conjunto de requisitos e direcionar o desenvolvimento incremental de um sistema. [Jacobson, Spence e Bittner 2011].

Como o *Use-Case* 2.0 pode ser utilizado no desenvolvimento de um sistema, as fases do desenvolvimento como análise, design, planejamento, estimativa, rastreamento e teste são suportadas pela abordagem. O *Use-Case* 2.0 não

prescreve como planejar, gerenciar e desenvolver um sistema, mas provê uma estrutura para auxiliar nas práticas de gerenciamento e de desenvolvimento selecionadas [Jacobson, Spence e Bittner 2011].

Para a aplicação do *Use-Case 2.0* com sucesso, 6 princípios são utilizados:

1. Simplificar através de *stories*: as *stories* são utilizadas para descrever os requisitos, englobando como alcançar o objetivo com sucesso e como lidar com problemas que podem ocorrer. O resultado das *stories* está presente como parte da narrativa dos casos de uso que acompanha cada caso de uso;

2. Conhecer o todo: sem entender completamente o sistema não é possível tomar as decisões corretas sobre o que incluir no sistema, o que retirar, seu custo e seus benefícios. O modelo de casos de uso dá a visualização do sistema como um todo e o diagrama de casos de uso pode auxiliar na parte visual para o entendimento do modelo de casos de uso;

3. Foco no valor: priorizar um objetivo específico de um ator específico é importante para obter o fluxo principal que conterá o principal caminho para se alcançar o objetivo. Através das *stories* vários fluxos são identificados, dentre eles o principal;

4. Construir o sistema em *slices*: construir um sistema por inteiro aumenta as chances de erro, para isso o sistema deve ser dividido em *slices* — pedaços menores de um caso de uso;

5. Entregar o sistema em incrementos: auxilia na evolução do sistema, na qual cada incremento adiciona novas funcionalidades ou melhorias;

6. Adaptar para se enquadrar nas necessidades da equipe: diferentes times e diferentes situações requerem diferentes níveis de detalhe.

Os itens a serem trabalhados pelo *Use-Case* 2.0 são os requisitos, o sistema a ser desenvolvido de acordo com os requisitos e os testes, para demonstrar que o sistema está de acordo com os requisitos. Esses itens citados acima são

capturados pelos casos de uso, as *stories* e os *slices*, que são o coração do *Use-Case* 2.0 [Jacobson, Spence e Bittner 2011].

Os casos de uso capturam os requisitos através das *stories*, que são contadas pelos *stakeholders*, e auxiliam a identificar os *slices* e os testes. Os testes por sua vez, verificam a qualidade e completude da implementação dos *slices* e consequentemente do sistema, cujo o escopo e objetivos são modelados pelos casos de uso.

As partes que compõem a estrutura interna do *Use-Case* 2.0 são:

- Casos de uso: conforme o sistema for implementado através dos *slices*, os casos de uso vão progredindo no sistema.
- *Slices*: os *slices* possibilitam a divisão dos casos de uso, permitindo a seleção dos pedaços a serem trabalhados e provendo unidades menores aproximadamente do mesmo tamanho. Como os *slices* são geralmente pequenos e acabam fluindo rápido, o desenvolvimento e o teste tornam-se ágeis. Com isso, os *slices* tem uma maior vazão a cada iteração do sistema, possibilitando um panorama mais preciso do estado de desenvolvimento do caso de uso e do sistema como um todo.
- *Stories*: as stories são a forma de explorar os casos de uso com o auxílio dos *stakeholders*, auxiliando na identificação dos *slices* e dos casos de teste. Um caso de uso pode ser decomposto em várias *stories*. Cada uma com sua importância para atores e outros *stakeholders*. A *story* por sua vez é descrita como parte da narrativa de caso de uso. Um ou mais fluxos formam uma rede que pode ser entendida como o resumo de todas as *stories* necessárias para descrever um caso de uso.

Com a descrição das partes mais importantes do *Use-Case* 2.0, verifica-se a proximidade com técnicas de gerenciamento iterativas e incrementais [Jacobson, Spence e Bittner 2011].

2.3. Kanban

O Kanban é uma abordagem que usa um quadro físico para visualizar tarefas. Dessa forma melhora-se a compreensão e o fluxo de trabalho. A metodologia Kanban também sugere a limitação no progresso de trabalho, assim, reduz o desperdício de tempo e esforço devido a multitarefas e mudanças de contexto, estimulando a colaboração para melhorar o sistema. O método não sugere um número de passos ou procedimentos, contudo estimula mudanças contínuas, incrementais e evolucionárias no sistema. Assim um dos objetivos do Kanban é minimizar resistência à mudança e facilita-la [Boeg 2012].

Como pode ser observado na Figura 2, o quadro Kanban exibe um conjunto de cartões que descrevem uma tarefa a ser resolvida. Esses cartões são organizados no quadro por colunas: uma coluna representa o estado de desenvolvimento em que uma tarefa se encontra.

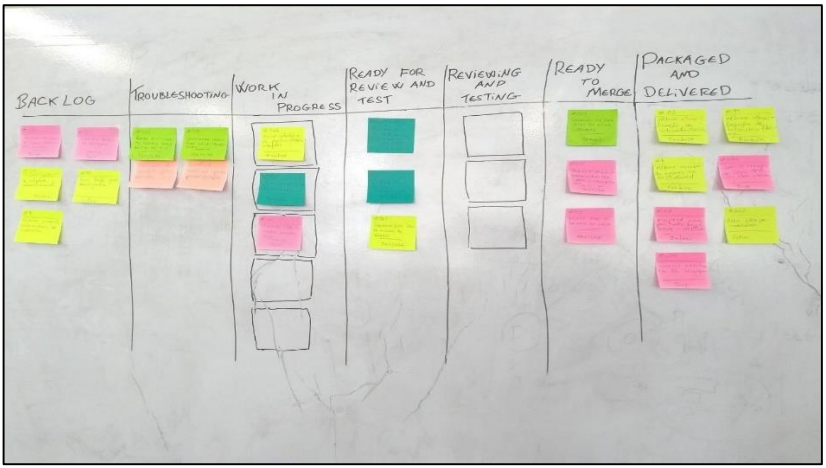


Figura 26: Quadro Kanban físico

2.4. Kanboard, uma ferramenta web

O Kanban físico possui algumas desvantagens. Considerando membros da mesma equipe geograficamente dispersos, há uma dificuldade em se manter a consistência entre quadros Kanban diferentes. Outra desvantagem de se utilizar meios físicos é a dificuldade de manter um histórico completo (necessário para resgatar pacotes de trabalho já terminados) e *backup* em caso de sinistros (como uma faxineira motivada pelo desejo de limpar uma parede coberta de papéis).

Com o objetivo de resolver essas limitações, atualmente há uma demanda para ferramentas automatizadas de planejamento, monitoração e controle.

Uma dessas ferramentas é o Kanboard, que é utilizado na gerência de projetos do laboratório de Telemedicina. A versão da ferramenta utilizada no laboratório de Telemedicina permite a visualização objetiva das tarefas, facilidade em arrastar tarefas entre colunas, diferentes formas de visualização, ações automáticas, gráficos de Gantt, relatórios de produtividade, internacionalização para 26 idiomas entre outras funcionalidades. A Figura 3 apresenta um quadro Kanban similar ao da Figura 2, porém usando o Kanboard. Nota-se que não há perda de informações; pelo contrário, o uso do Kanboard permite que seja visualizado várias informações como prioridade, complexidade e prazo com mais clareza.

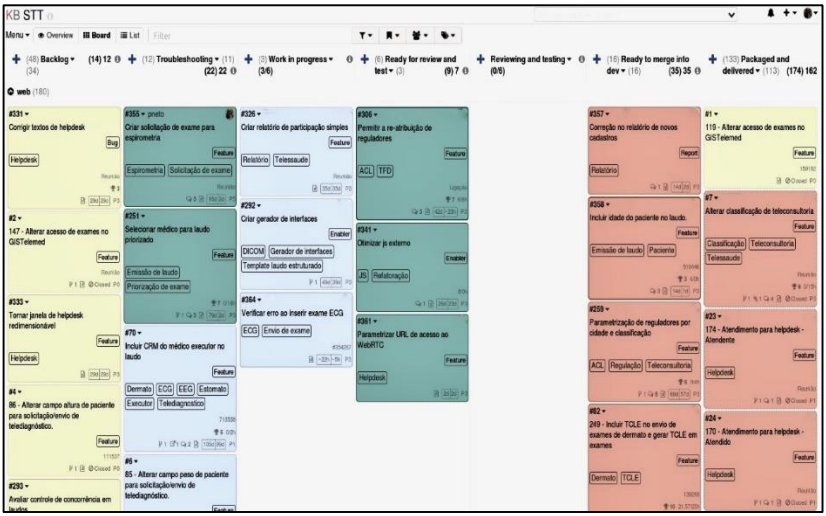


Figura 27: Quadro Kanban virtual

3. Modelagem da solução

A seguir é descrito o processo de engenharia de requisitos para a modelagem e desenvolvimento da ferramenta proposta.

3.1. Requisitos funcionais

Os requisitos funcionais apresentados no quadro a seguir foram levantados a partir de reuniões com o orientador e o co-orientador que instruíram o desenvolvimento da ferramenta, na qual foram identificados os requisitos necessários para uma implementação da abordagem *Use-Case* 2.0. Os requisitos levantados têm como base de estudo o livro escrito por Jacobson, Spence e Bittner (2011), que define o *Use-Case* 2.0 e informa quais funcionalidades são necessárias para alcançar o objetivo de implementar o *Use-Case* 2.0.

Quadro 24: Requisitos funcionais

ID	Descrição
RF01	A ferramenta deve permitir que o usuário crie um ator (para interagir com os casos de

	uso), edite um ator (para modificar alguma característica) e exclua um ator (para remover atores não utilizados).
RF02	A ferramenta deve permitir a criação, edição e exclusão de casos de uso e de seus componentes internos como as histórias e fluxos.
RF03	A partir de um caso de uso, a ferramenta deve permitir a criação de *slices*, com suas histórias selecionadas e os fluxos.
RF04	A ferramenta deve permitir a visualização de um diagrama com os atores, casos de uso e *slices* cadastrados no sistema.
RF05	A ferramenta deve permitir a visualização dos casos de uso e *slices* no quadro Kanban.

3.2. Modelagem de sistema em *Use-Case* 2.0

A seguir casos de uso, *slices* e histórias de usuário (*stories*) são estruturados de forma agrupada e apresentados usando a codificação AG## para os agrupamentos, AG##-UC## para os casos de uso, AG##-UC##-SL## para os *slices* e HU## para as histórias de usuário.

Os casos de uso do agrupamento AG01 são referentes à construção de uma instância do *Use-Case* 2.0 no Kanboard, enquanto os casos de uso do agrupamento AG02 são referentes ao diagrama de visualização dos componentes do *Use-Case* 2.0 no Kanboard.

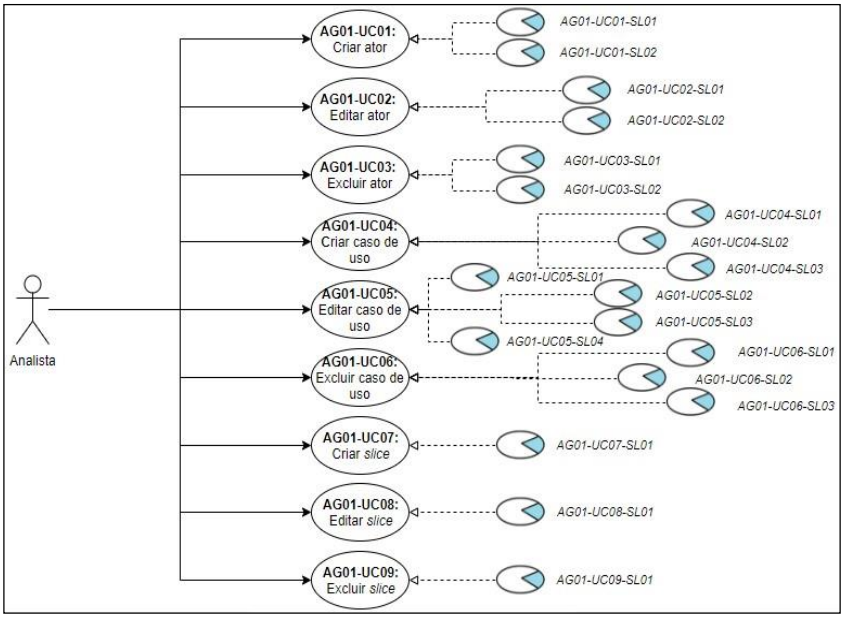


Figura 28: Casos de uso no agrupamento 1

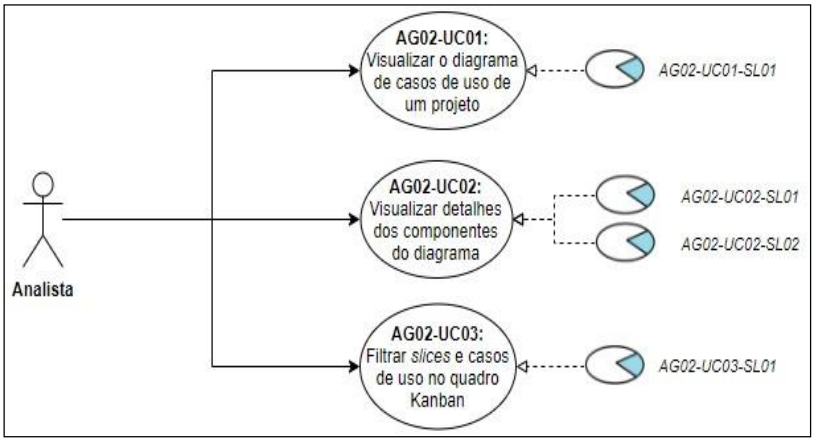


Figura 29: Casos de uso no agrupamento 2

Seguindo a abordagem de *Use-Case* 2.0, cada *slice* inclui ao menos uma história de usuário (*stories*), que por sua vez representa ao menos um fluxo base, alternativo ou de exceção

do caso de uso. No Apêndice I estão as instâncias *Use-Case* 2.0 usadas para modelar a implementação proposta.

3.3. Modelagem dos componentes

Para modelar os componentes envolvidos na implementação da abordagem *Use-Case* 2.0 no Kanboard, um diagrama de componentes foi construído. Um diagrama de componentes identifica os componentes que fazem parte de um sistema, subsistema ou mesmo componentes internos de um componente individual [GUEDES 2018]. Na figura a seguir é apresentado o diagrama de componentes baseado na UML 2.5.

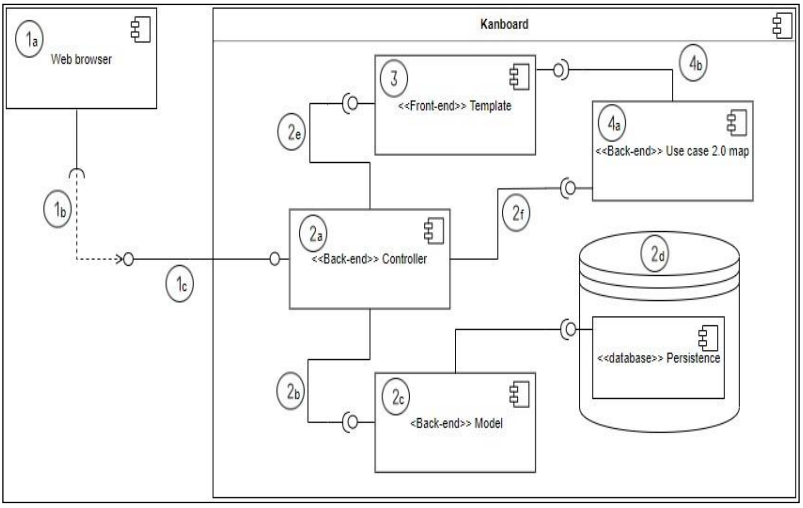


Figura 30: Diagrama de componentes

No diagrama apresentado, os componentes e as conexões são numerados para auxiliar na compreensão da implementação da abordagem *Use-Case* 2.0. A seguir a explicação para cada um dos pontos elencados na figura acima:

1. Acesso ao Kanboard com *Use-Case* 2.0 implementado.
    a. Através do navegador o usuário pode requisitar os recursos disponibilizados pelo Kanboard.

   b. A porta de interface requerida representa um grupo de mensagens enviadas a outro componente. A dependência é usada para indicar que a interface do navegador é satisfeita por uma interface no Kanboard.
   c. O Kanboard fornece uma porta para receber as mensagens enviadas pelo navegador.
2. Controlador e suas interações
   a. Recebe as requisições do usuário e identifica a ação a ser tomada. No controlador há a possibilidade de manipular uma instância do *Use-Case* 2.0 ou um *post-it* Kanban, para manter o contexto do trabalho, apenas o caso da manipulação do *Use-Case* 2.0 será abordado.
   b. Requisita uma ação ao modelo (atualizar, criar, recuperar ou deletar dados) enviando os dados referentes ao *Use-Case* 2.0.
   c. No modelo é construído um item SQL a partir da ação e dos parâmetros requisitados pelo usuário.
   d. Em seguida, o item SQL é executado por um gerenciador de banco de dados e os resultados (caso haja) são encaminhados ao controlador.
   e. Caso a requisição do usuário seja para visualizar as instâncias existentes do *Use-Case* 2.0 no quadro Kanban, o controlador envia os dados referentes às instâncias para a visão (chamado de template no código fonte) do sistema.
   f. Caso a requisição do usuário seja para visualizar o diagrama de casos de uso, o controlador envia os dados para a construção do diagrama de casos de uso no módulo contendo o VisJs.
3. Na visão (template), os casos de uso, slices e demais dados referentes ao *Use-Case* 2.0 são graficamente construídos para permitir a visualização no quadro Kanban por parte do usuário. Neste ponto, o usuário

pode interagir com o quadro Kanban acessando os casos de uso e seus *slices* e modificando o estágio de desenvolvimento de cada instância *Use-Case* 2.0. A visão também disponibiliza uma área para a visualização do diagrama com de casos de uso.

4. Diagrama de casos de uso com *slices*.
    a. Os dados recebidos do controlador são estruturados e organizados para comporem um grafo com os componentes *Use-Case* 2.0 e suas associações.
    b. O grafo criado é enviado à visão para ser visualizado pelo usuário, na qual poderá organizá-lo ou acessar casos de uso e slices de maneira ágil.

4. Desenvolvimento

4.1. Técnologias

O uso da ferramenta de gerência de projetos Kanboard como base para inclusão do novo módulo impõe restrições nas tecnologias a serem utilizadas no desenvolvimento para *Use-Case* 2.0. O quadro a seguir apresenta a lista das tecnologias utilizadas para o desenvolvimento do novo módulo.

Quadro 25: Tecnologias utilizadas no desenvolvimento

Tecnologia	Fonte	Uso
PHP 7.0.22	http://www.php.net/	Utilizado no desenvolvimento do *backend* do aplicativo
SQLite 3.1	https://www.sqlite.org	Utilizado para a persistência dos dados do sistema

JavaScript 5.0	https://www.javascript.com	Utilizado no desenvolvimento do *frontend* para possibilitar componentes dinâmicos
HTML5	https://www.w3.org/TR/html5/	Utilizado para formatar os campos das páginas do sistema

Será comentado brevemente as tecnologias citadas no Quadro 2 que implementam o frontend — parte visual de um programa — e o backend — servidor com serviços para o frontend — do Kanboard e consequentemente da ferramenta proposta.

O backend da ferramenta é desenvolvido em PHP 7.0.22, portanto o módulo proposto manteve-se na mesma linguagem e versão para evitar conflitos. A mesma lógica foi aplicada para o JavaScript e o HTML5. Conflitos estes que poderiam comprometer a ferramenta Kanboard e consequentemente este trabalho.

O PHP é a linguagem de programação utilizada para construir a lógica do backend do Kanboard. O PHP é código aberto e versátil, possibilitando a inserção do seu código dentro do HTML.

O SQLite é a linguagem de consulta utilizada para a manipulação dos dados do Kanboard, permitindo armazenar, recuperar ou excluir dados. O SQLite é uma linguagem menos robusta, se comparada a outras linguagens de consulta, porém é leve e atende perfeitamente os requisitos para a implementação da abordagem *Use-Case* 2.0.

O HTML é utilizado para formatar os campos que serão mostrados ao cliente e o JavaScript possibilita a adição de lógica e dinamismo no frontend.

Para permitir a modelagem gráfica dos atores, dos casos de uso e dos *slices* no módulo proposto, foi utilizada a biblioteca VisJS, que se destacou entre outras pela sua simplicidade e pelo fato de *plug-ins* já implementados para o Kanboard utilizarem esta biblioteca — o *plug-in* Relationgraph utiliza a biblioteca VisJS.

4.2. Implementação

Algumas partes do código fonte do Kanboard foram modificadas para a inclusão das novas funcionalidades. Estas modificações ocorreram em vários pontos, como no modelo de banco de dados (para a inclusão de tabelas) e na interface da tela (para possibilitar a visualização do componente de acesso ao diagrama de casos de uso).

O código-fonte da ferramenta alterada para suportar a abordagem *Use-Case* 2.0 está disponível em <*https://github.com/PauloJRN/tcc*>.

Serão mostrados os casos de uso e as respectivas telas do Kanboard que sofreram as modificações para possibilitar a implementação do *Use-Case* 2.0 e seus respectivos casos de uso e *slices*.

Os casos de uso AG01-UC01, AG01-UC02 e AG01-UC03 detalham, respectivamente, a criação, edição e exclusão de um ator. Para a criação de um ator, seleciona-se a opção “Actors” (passo 1) para poder visualizar a tela compreendida na figura abaixo. No passo 2, clica-se em “Add new actor” e uma tela para digitar as informações do ator aparecerá. Com as informações sobre o ator preenchidas, há a possibilidade de editar (passo 3) ou remover o ator (passo 4).

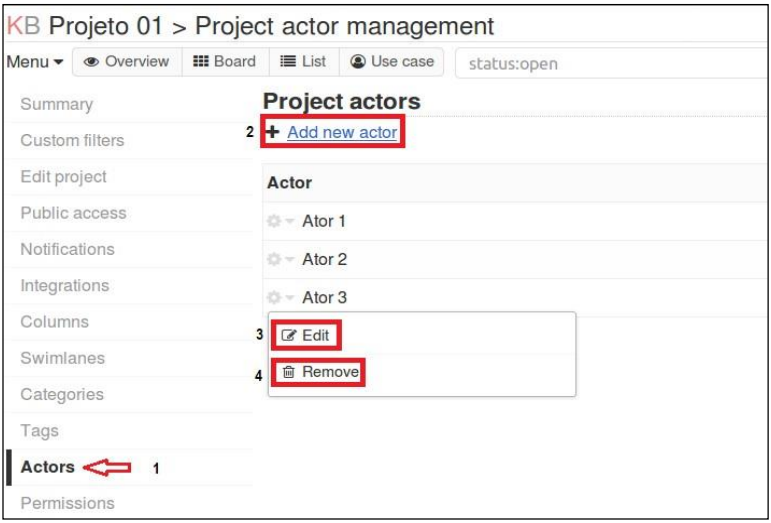


Figura 31: Tela de manipulação do ator

O caso de uso AG01-UC04 explicita a criação do caso de uso. A tela para a criação de um caso de uso é extensa, portanto apenas uma fração dela foi mostrada na Figura 8. Na tela de criação do caso de uso, há campos como nome, histórias, fluxos (básicos, alternativos e de exceção), prioridade do caso de uso, ator, entre outros atributos a serem preenchidos.

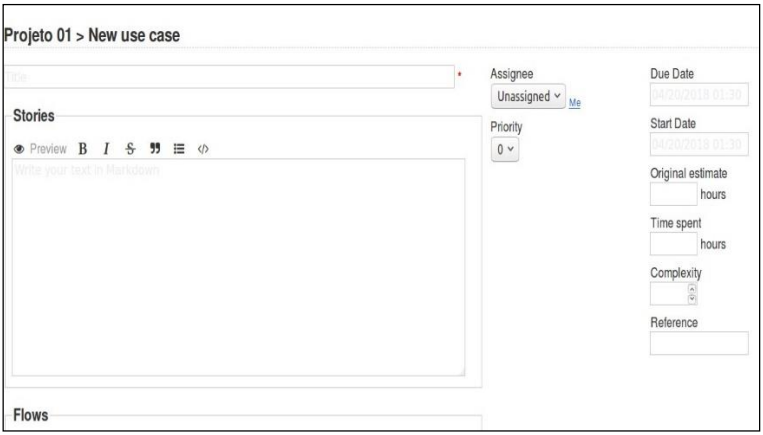


Figura 32: Tela de criação de caso de uso

Um caso de uso no Kanboard possui propriedades únicas que o diferenciam de um *post-it* do Kanban. Apenas no caso de uso é possível associar atores, informando as partes interessadas no desenvolvimento daquele caso de uso, e criar *slices* (Figura 9), fragmentando o caso de uso. No diagrama de casos de uso, apenas casos de uso e seus respectivos atores e *slices* aparecem, não mostrando os *post-its* do Kanboard.

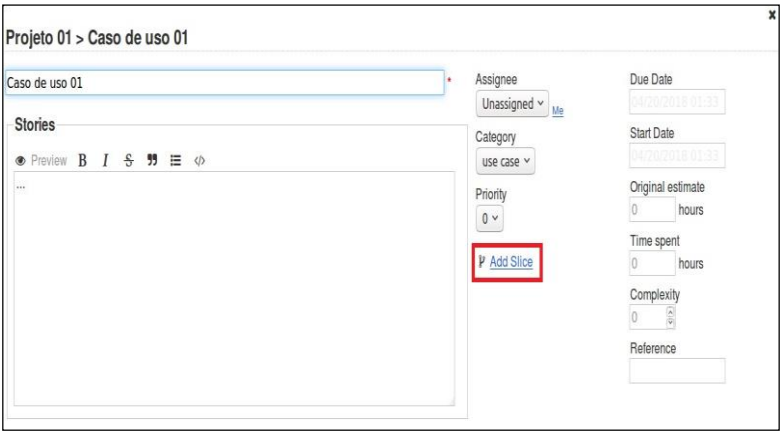


Figura 33: Opção para a criação do slice dentro do caso de uso

O caso de uso AG01-UC07 explana a criação do *slice***.** A tela de criação de um *slice* contém campos parecidos com os da criação do caso de uso, porém há campos diferentes. Os campos que se diferem do caso de uso são as "Selected stories" e os "Test cases", sendo que o campo para informar o ator não aparece pois já é o mesmo ator do caso de uso.

Nas "Selected stories" são informadas as histórias selecionadas do caso de uso para compor o *slice*. E os "Test case" são os possíveis testes a serem feitos sobre as funcionalidades a serem implementadas pelo *slice*.

O caso de uso AG02-UC01 detalha a visualização do diagrama de casos de uso de um projeto: Na Figura 10, apresenta-se o diagrama com casos de uso e seus atores e *slice* para auxiliar na compreensão das relações existentes. Através do

diagrama é possível acessar as informações dos casos de uso e dos *slices* com um duplo clique no *mouse*.

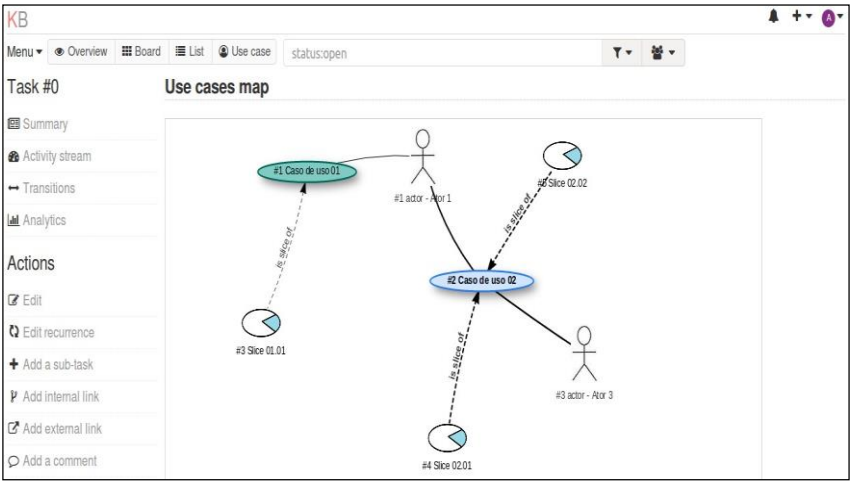


Figura 34: Diagrama de casos de uso implementado no Kanboard

Na Figura 11 o quadro Kanban aparece com o suporte ao *Use-Case* 2.0. Os casos de uso e os *slices* podem ser identificados observando-se os rótulos contidos nas caixas. As colunas são as fases na qual um *Use-Case* 2.0 deve percorrer para alcançar o seu objetivo.

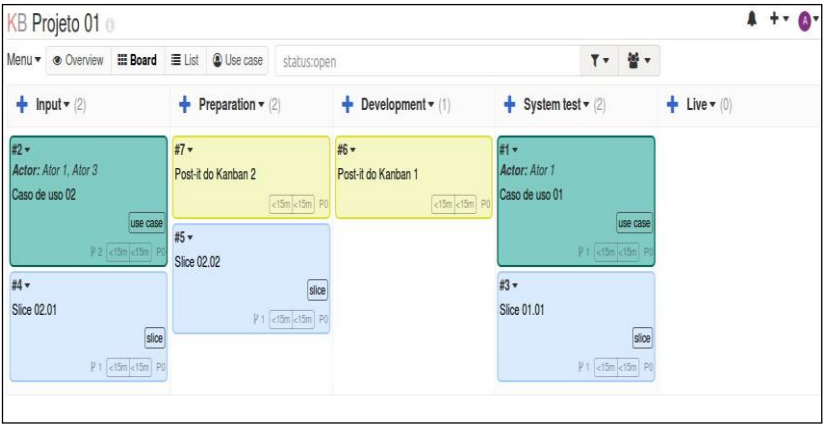


Figura 35: Quadro Kanban com instâncias do Use-Case 2.0

5. Avaliação

Para a avaliação da usabilidade do sistema proposto neste trabalho foi utilizado o SUS (*System Usability Scale*), uma escala de usabilidade confiável e de baixo custo que pode ser usada para avaliações da usabilidade de sistemas. Para especificar a usabilidade de um sistema (adequação à finalidade), deve-se definir os usuários pretendidos do sistema, as tarefas que esses usuários executarão com ele e as características do sistema físico, organizacional e ambiente social em que será usado [Brooke, et al., 1996].

Quadro 26: Questionário SUS

01. Eu gostaria de usar o sistema mais frequentemente.
02. Eu achei o sistema desnecessariamente complexo.
03. Eu achei o sistema fácil de usar.
04. Eu acho que precisaria de ajuda de uma pessoa com conhecimentos técnicos para usar o sistema.
05. Eu acho que as várias funções do sistema estão muito bem integradas.
06. Eu acho que o sistema apresenta muita inconsistência.
07. Eu imagino que a maioria das pessoas aprenderão a utilizar o sistema facilmente.

08. Eu acho o sistema muito atrapalhado/sobrecarregado de se usar.
09. Eu me senti confiante ao usar o sistema.
10. Eu precisei aprender várias coisas novas antes de conseguir usar o sistema.

Para avaliar a aplicação e atendimento da abordagem *Use-Case* 2.0, um questionário foi elaborado, contendo perguntas complementares na tentativa de obter uma acurácia maior na avaliação e com isso determinar mais precisamente os pontos positivos e negativos das funcionalidades da abordagem *Use-Case* 2.0 implementada.

Quadro 27: Questionário específico

Questão	Motivação
AQ1. Quanto tempo de estudo para compreender o funcionamento do sistema? (em minutos)	Analisar se o tutorial gerado foi efetivo na utilização da ferramenta.
AQ2. Quanto tempo gasto para criar um ator, um caso de uso e um *slice*? (em minutos)	Analisar se a usabilidade da ferramenta proposta permite criar uma instância simples do *Use-Case* 2.0 em tempo hábil.
AQ3. Houve um aumento no tempo de utilização do sistema após a inclusão da abordagem *Use-Case* 2.0 no Kanboard?	Analisar se houve um aumento significativo na complexidade do Kanboard em relação ao Kanboard alterado.
AQ4. Quais foram as principais dificuldades encontradas ao utilizar o Kanboard com a abordagem *Use-Case* 2.0?	Analisar os pontos específicos de dificuldade ao utilizar a ferramenta.

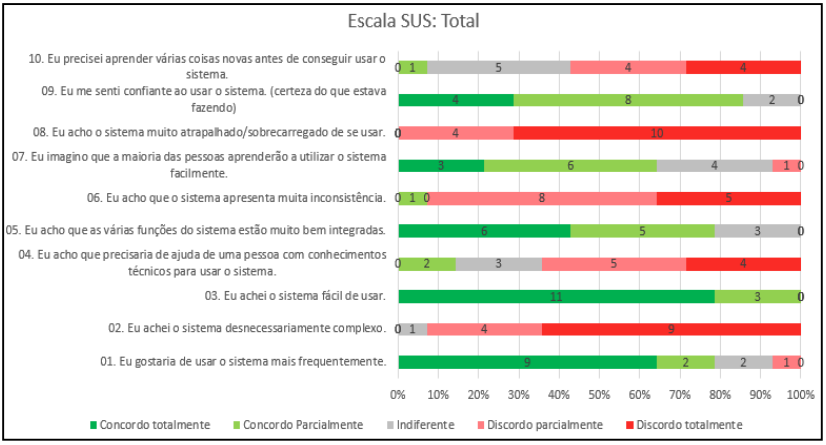
AQ5. A experiência da aplicação do Kanboard com a abordagem *Use-Case* 2.0 na unidade organizacional foi benéfica?	Analisar a aceitação da ferramenta por parte dos avaliadores.
AQ6. Quais as principais técnicas da abordagem *Use-Case* 2.0 obteve maior aceitação?	Analisar quais pontos obtiveram maior aceitação.

5.1. Análise da avaliação da usabilidade

Nesta etapa, os dados são analisados para verificar a aceitação, por parte dos avaliadores, do sistema implementado. No total, são 14 avaliadores que informaram através do questionário SUS as impressões sobre o sistema.

O Gráfico 1 mostra os dados coletados sobre os itens do SUS. Será realizada uma análise sobre o gráfico contido para identificar a opinião dos avaliadores sobre o sistema proposto.

Gráfico 13: Dados coletados com a escala SUS



Com base no item 10 do SUS, observa-se que 57% dos avaliadores obtiveram pouca, ou não obtiveram, dificuldades em

compreender o *Use-Case* 2.0. 36% se mantiveram indiferentes em relação ao volume de conhecimento para utilizar o sistema e 7% gastou mais tempo estudando para compreender o *Use-Case* 2.0. Conclui-se para este item que houve um volume médio de conteúdo a ser absorvido pelos avaliadores para compreender o funcionamento do *Use-Case* 2.0.

Os valores observados no item 9 mostram que os avaliadores têm um nível alto de confiança, mas há algum ponto do sistema que pode não ser intuitivo, causando os 57% de concordo totalmente/parcialmente e os 14% de indiferentes. Ao aplicar a avaliação presencialmente no laboratório de Telemedicina, foi presenciado uma quantidade alta de dúvidas sobre como preencher o fluxo de exceção, podendo ser a causa da queda na confiança ao utilizar o sistema.

Observando o item 8 do SUS, conclui-se que a maioria dos avaliadores não achou a usabilidade do sistema poluída (71%), ou seja, sem exagero de informações na tela para confundir ou sobrecarregar o usuário.

Ao observar o item 7 do SUS, nota-se uma correlação com o item 10, pois a facilidade de aprendizagem do sistema está relacionada com o entendimento do *Use-Case* 2.0. Portanto, observa-se que os avaliadores têm alguma dúvida (36%) sobre a facilidade de aprendizagem do sistema em um contexto diferente do seu ambiente de trabalho.

No item 6 do SUS observa-se que a maioria dos avaliadores (93%) percebeu alguma inconsistência técnica ou teórica na implementação do sistema. Como essas inconsistências não foram informadas pelos avaliadores para posterior correção, não é possível identificar se houve de fato alguma inconsistência ou se o avaliador não compreendeu algum ponto da utilização do sistema ou do *Use-Case* 2.0.

No item 5 do SUS, os dados indicam que há uma concordância (79%) por parte dos avaliadores que o sistema possui suas funções bem integradas, porém 21% não souberam opinar ou ficou em dúvida se o sistema é bem integrado.

No item 4 do SUS observa-se que os avaliadores, apesar de conseguirem utilizar o sistema (65%), ainda tem dúvidas (35%) sobre algum procedimento.

Observando o item 3 do SUS nota-se que não houve dificuldade para a utilização do sistema (100% de concordância sobre a facilidade de usa o sistema). Correlacionando o item 3 com o item 4 pode-se concluir que a dificuldade maior dos avaliadores é teórica, pois utilizam o sistema facilmente, mas tem dúvidas sobre a aprendizagem do sistema (item 7) e houve um acréscimo razoável de informações aos avaliadores (item 10).

Segundo o item 2 do SUS, a maioria dos avaliadores não achou o sistema complexo (93%). Esse item possivelmente se justifica devido a familiaridade dos avaliadores com a ferramenta Kanboard.

Analisando o item 1 do SUS, nota-se uma aceitação do sistema implementado por parte dos avaliadores (79%), contudo, dois avaliadores (14%) sentiram-se indiferentes e um avaliador (7%) discorda parcialmente da utilização do sistema, dando a noção de que o sistema proposto pode ser melhorado.

A pontuação SUS varia de 0 a 100, sendo 0 o sistema está muito ruim e 100 está o melhor possível. O valor médio encontrado em avaliações é 68, indicando que o sistema está sem grandes problemas de usabilidade [Brooke 2013]. A pontuação SUS calculada para o módulo implementado, de valor aproximado de 81 pontos SUS, indica que o módulo está em boas condições no que diz respeito a usabilidade.

7.15.2. Análise da aplicação do *Use-Case* 2.0 na ferramenta Kanboard

Em relação à avaliação da aplicação da abordagem *Use-Case* 2.0, 14 avaliadores responderam às perguntas sobre o tempo investido em compreender o funcionamento do sistema e sobre o tempo demandado para a construção de uma instância *Use-Case* 2.0 (**AQ1** e **AQ2**) e 8 avaliadores responderam as demais perguntas (**AQ3**, **AQ4**, **AQ5** e **AQ6**).

Sobre os valores obtidos em **AQ1** e **AQ2**, pode-se observar que o tempo para compreender o funcionamento do sistema, o que inclui o estudo sobre a abordagem *Use-Case* 2.0 e o funcionamento da abordagem no Kanboard, obteve uma média de aproximadamente 14 minutos. Após compreender o funcionamento do sistema, a criação de uma instância levou aproximadamente 12 minutos para ser concluída.

Esses valores mostram que para utilizar o sistema, inicialmente o tempo médio gira em torno de 26 minutos, que possivelmente pode ser diminuído conforme os usuários ficam mais experientes na utilização do sistema.

Quadro 5: Tempos observados

	Tempo de estudo para compreender o funcionamento do sistema. (em minutos)	Tempo gasto para criar um ator, um caso de uso e um *slice*. (em minutos)
Organização 1: Telemedicina	10	15
	15	15
	15	20
	10	15
	15	15
	10	5
	20	15
	4	15
Média	**12,375**	**14,375**
Organização 2	10	10
	5	10
	15	5
	15	10
	15	10
	30	12
Média	**15**	**9,5**

Media total	**13,6875**	**11,9375**

Na questão **AQ3** sobre o aumento de tempo na utilização do sistema Kanboard após a implementação da abordagem *Use-Case* 2.0, foi observado que apenas um avaliador relatou que não houve aumento do tempo, os demais avaliadores afirmaram que houve o aumento. Esse aumento ocorreu devido as novas funcionalidades do *Use-Case* 2.0.

Na questão **AQ4** sobre as dificuldades ao utilizar o sistema com a nova abordagem, 4 avaliadores informaram não ter dificuldades. Os outros 4 avaliadores informaram dificuldades, sendo que 1 avaliador informou dificuldade de compreender o funcionamento da abordagem implementada em relação ao Kanboard e 3 avaliadores informaram ter dificuldades na criação dos *slices*.

Na questão **AQ5** sobre os benefícios da implementação para a organização, a maioria (87,5%, 7 avaliadores) dos avaliadores afirmou que a ferramenta implementada pode trazer benefícios para a unidade organizacional, porém informaram que o tempo de utilização da ferramenta foi insuficiente para obter uma análise mais crítica sobre esta pergunta.

E por fim, na questão **AQ6** os participantes informaram que as principais técnicas do *Use-Case* 2.0 que obtiveram maior aceitação foram os casos de uso (1 voto), o diagrama de casos de uso (6 votos) e os *slices* (6 votos). Importante ressaltar que houveram avaliadores que votaram em mais de uma técnica.

6. Conclusão

Neste trabalho é apresentado o desenvolvimento de um módulo para o sistema de gerenciamento de projetos Kanboard, possibilitando a utilização da abordagem *Use-Case* 2.0. O objetivo principal deste trabalho foi desenvolver um módulo que possibilitasse a utilização de casos de uso no Kanboard segundo a ótica da abordagem *Use-Case* 2.0 e avaliar a aceitação por parte dos interessados.

Neste contexto, foi realizado um estudo sobre a teoria por trás da abordagem *Use-Case* 2.0 e feita a análise da fundamentação teórica. Também foi realizada a análise do estado da arte identificando a ausência de uma implementação da abordagem *Use-Case* 2.0 em uma ferramenta para gerência de projeto.

O módulo desenvolvido foi avaliado e os dados coletados desta avaliação foram analisados, concluindo que o módulo foi bem aceito entre os avaliadores e que não houve maiores dificuldades em sua utilização.

Referências

BOEG, Jesper. **Priming Kanban**. InfoQ/Trifork, 2012.

BOOCH, Grady et al. **The unified modeling language**. Unix Review, v. 14, n. 13, p. 5, 1996.

BROOKE, John et al. **SUS-A quick and dirty usability scale. Usability evaluation in industry**, v. 189, n. 194, p. 4-7, 1996.

BROOKE, John. **SUS: a retrospective. Journal of usability studies**, v. 8, n. 2, p. 29-40, 2013.

COCKBURN, Alistair. **Writing Effectives Use Cases**. Boston: Addison-Wesley, 2001.

GUEDES, Gilleanes TA.UML 2-**Uma abordagem prática**. Novatec Editora, 2018.

ISO, IEC. **IEEE. 29148: 2011-Systems and software engineering - Requirements engineering.** Technical report, 2011.

JACOBSON, Ivar; SPENCE, Ian; BITTNER, Kurt. **Use Case 2.0: The guide to succeeding with use cases**. Ivar Jacobson International, 2011.

KARNER, Gustav. **Resource estimation for objectory projects. Objective Systems**. SF AB, v. 17, 1993.

KAUR, Rupinder; SENGUPTA, Jyotsna. **Software Process Models and Analysis on Failure of Software Development Projects**. International Journal of Scientific & Engineering Research. [s.i.], p. 1-4. fev. 2011.

POHL, Klaus. **Requirements engineering fundamentals: a study guide for the certified professional for requirements engineering exam-foundation level-IREB compliant**. Rocky Nook, Inc., 2016.

UML, OMG. **Unified Modelling Language** version 2.5. Unified Modelling (2017). 2017.

Apêndice I

Agrupamento AG01: permitir criar artefatos de *Use-Case 2.0*
Caso de uso AG01-UC01: criar ator **Ator:** analista **Pré-condição:** 1. Usuário autenticado no sistema; 2. Acessar um projeto; 3. Acessar “Settings”. **Fluxo principal:** 1. Clicar na opção “Actors”; 2. Clicar em “Add new actor”; 3. Digitar o nome do ator; 4. Clicar em “Save”.

Fluxo alternativo 01:

1. Deriva do passo 4
2. Se houver ator com o mesmo nome, mostra mensagem informando que já existe ator com o nome informado.

Pós-condição: há um novo ator cadastrado no sistema, a lista de atores está atualizada e os casos de uso podem acessar os novos atores.

Pós-condição do fluxo alternativo 01: o novo ator não foi cadastrado no sistema.

***Slice* AG01-UC01-SL01**

História de usuário HU01: como analista eu quero criar um ator para vinculá-lo aos casos de uso.

***Slice* AG01-UC01-SL02**

História de usuário HU02: como analista eu quero ser alertado se tentar replicar o nome de um ator já existente.

Caso de uso AG01-UC02: editar ator

Ator: analista

Pré-condição:

1. Usuário autenticado no sistema;
2. Acessar um projeto;
3. Acessar “Settings”;
4. Haver pelo menos um ator cadastrado no sistema.

Fluxo principal:

1. Clicar em “Actors”;
2. Clicar na engrenagem ao lado do nome do ator;
3. Selecionar a opção “Edit”;
4. Modificar o nome do ator;
5. Clicar em “Save”.

Fluxo alternativo 01:

1. Deriva do passo 5.
2. Se houver ator com o mesmo nome, mostra mensagem informando que já existe ator com o nome informado.

Pós-condição: nome do ator modificado.

Pós-condição do fluxo alternativo 01: a modificação do nome do ator não foi concluída.

***Slice* AG01-UC02-SL01**
História de usuário HU01: como analista eu quero alterar o nome de um ator.

***Slice* AG01-UC02-SL02**
História de usuário HU02: como analista eu quero ser alertado se tentar replicar o nome de um ator já existente.

Caso de uso AG01-UC03: excluir ator
Ator: analista
Pré-condição:
1. Usuário autenticado no sistema;
2. Acessar um projeto;
3. Acessar "Settings";
4. Haver pelo menos um ator cadastrado no sistema.

Fluxo principal:
1. Clicar em "Actors";
2. Clicar no ícone da engrenagem ao lado do nome do ator;
3. Selecionar a opção "Remove";
4. Exibir mensagem de confirmação de exclusão.

Fluxo alternativo 01:
1. Sequente ao passo 3;
2. Tratar os casos de uso que estiverem associados ao ator removido.
3. Continua o passo 4.

Pós-condição: ator removido do sistema, dos casos de uso associados e da lista de atores.
Pós-condição do fluxo alternativo 01: ator não removido.

***Slice* AG01-UC03-SL01**
História de usuário HU01: como analista eu quero deletar o nome de um ator.

***Slice* AG01-UC03-SL02**
História de usuário HU02: Como analista eu quero que os casos de uso que não tiverem nenhum outro ator que não seja o que está em processo de exclusão também seja excluído.

Caso de uso AG01-UC04: criar caso de uso

Ator: analista
Pré-condição:
1. Usuário autenticado no sistema;
2. Acessar um projeto;

Fluxo principal:
1. Clicar em "+" para adicionar uma tarefa;
2. Digitar o título;
3. Selecionar o ator;
4. Selecionar como categoria a opção "Use case";
5. Clicar em "Save".

Fluxo alternativo 01:
1. Clicar em "Menu";
2. Clicar em "Add a new task";
3. Digitar o título;
4. Selecionar o ator;
5. Selecionar como categoria a opção "Use case";
6. Clicar em "Save".

Fluxo alternativo 02:
1. Entre os passos 1 e 4;
2. Digitar o texto;
3. Continua o passo 5.

Fluxo alternativo 03:
1. Entre os passos 1 e 4;
2. Selecionar um ou mais rótulos;
3. Continua o passo 5:

Pós-condição: novo caso de uso cadastrado no sistema.

***Slice* AG01-UC04-SL01**
História de usuário HU01: como analista gostaria de ter a opção de poder cadastrar um caso de uso.

***Slice* AG01-UC04-SL02**
História de usuário HU02: como analista gostaria de ter a opção de digitar um texto com informações sobre o caso de uso.

***Slice* AG01-UC04-SL03**

História de usuário HU03: como analista gostaria de ter a opção de selecionar um ou mais rótulos (*tag*) para a identificação do caso de uso.

Caso de uso AG01-UC05: editar caso de uso

Ator: analista

Pré-condição:

1. Usuário autenticado no sistema;
2. Acessar um projeto;
3. Acessar um caso de uso.

Fluxo principal:

1. Clicar em “Edit”;
2. Abre tela de edição;
3. Clicar em “Save”.

Fluxo alternativo 01:

1. Sequente ao passo 2;
2. Modificar título;
3. Continua o passo 3.

Fluxo alternativo 02:

1. Sequente ao passo 2;
2. Selecionar ator;
3. Continua o passo 3.

Fluxo alternativo 03:

1. Sequente ao passo 2;
2. Digitar o texto;
3. Continua o passo 3.

Fluxo alternativo 04:

1. Sequente ao passo 2;
2. Selecionar um ou mais rótulos;
3. Continua o passo 3.

Pós-condição: caso de uso teve alterações realizadas e salvas.

***Slice* AG01-UC05-SL01**

História de usuário HU01: como analista gostaria de ter a opção de editar um caso de uso.

***Slice* AG01-UC05-SL02**

História de usuário HU02: como analista gostaria de ter a opção de editar o título de um caso de uso.
***Slice* AG01-UC05-SL03** **História de usuário HU03:** como analista gostaria de ter a opção de trocar o ator de um caso de uso.
***Slice* AG01-UC05-SL04** **História de usuário HU04:** como analista gostaria de ter a opção de editar um texto com informações sobre o caso de uso.
Caso de uso AG01-UC06: excluir caso de uso **Ator:** analista **Pré-condição:** 1. Usuário autenticado no sistema; 2. Acessar um projeto; 3. Acessar um caso de uso. **Fluxo principal:** 1. Clicar em “Delete”; 2. Abre tela para confirmar exclusão; 3. Clicar em “Yes”. **Fluxo alternativo 01:** 1. Sequente ao passo 3; 2. Retirar a referência de atores e casos de uso associados ao caso de uso excluído. **Fluxo alternativo 02:** 1. Sequente ao passo 3; 2. Excluir os *slices*. **Pós-condição:** o caso de uso é removido do sistema, perde a referência com os atores e os casos de uso. *Slices* associados são excluídos.
***Slice* AG01-UC06-SL01** **História de usuário HU01:** como analista gostaria de poder excluir um caso de uso.
***Slice* AG01-UC06-SL02**

História de usuário HU02: como analista gostaria de que os atores e outros casos de uso associados deixem de referenciar o caso de uso deletado.

***Slice* AG01-UC06-SL03**
História de usuário HU03: como analista gostaria de que os *slices* associados a este caso de uso sejam excluídos.

Caso de uso AG01-UC07: criar um *slice*
Ator: analista
Pré-condição:
1. Usuário autenticado no sistema;
2. Acessar um projeto;
3. Acessar um caso de uso.

Fluxo principal:
1. Clicar em "Add slice";
2. Abre tela para entrada das informações;
3. Digitar título;
4. Digitar *stories* do *slice*;
5. Clicar em "Save".

Pós-condição: *Slice* registrado no sistema e associado ao caso de uso onde o *slice* foi criado. *Slice* na lista de *slices* do caso de uso.

***Slice* AG01-UC07-SL01**
História de usuário HU01: como analista gostaria de adicionar *slices* a um caso de uso.

Caso de uso AG01-UC08: editar um *slice*
Ator: analista
Pré-condição:
1. Usuário autenticado no sistema;
2. Acessar um projeto;
3. Acessar um caso de uso;
4. Acessar um *slice*.

Fluxo principal:
1. Clicar em "Edit";
2. Abre tela para entrada das informações;
3. Modificar título;

4. Clicar em “Save”.

Fluxo alternativo 01:

1. Sequente ao passo 2;
2. Modificar *stories*;
3. Continua o passo 3 ou 4.

Pós-condição: alterações no *slice* salvas.

***Slice* AG01-UC08-SL01**

História de usuário HU01: como analista gostaria de alterar o título e as *stories* do *slice*.

Caso de uso AG01-UC09: excluir um *slice*

Ator: analista

Pré-condição:

1. Usuário autenticado no sistema;
2. Acessar um projeto;
3. Acessar um caso de uso;
4. Acessar um *slide*.

Fluxo principal:

1. Clicar em “Delete”;
2. Abre tela para confirmar exclusão;
3. Clicar em “Yes”.

Pós-condição: *slice* e seu conteúdo deixam de existir no sistema. *Slice* é desassociado e sai de lista de *slices* do caso de uso.

***Slice* AG01-UC09-SL01**

História de usuário HU01: como analista gostaria de excluir o *slice* associado ao caso de uso.

Agrupamento AG02: permitir a visualização do *Use-Case* 2.0 no sistema.

Caso de uso AG02-UC01: visualizar o diagrama de casos de uso de um projeto.

Ator: analista

Pré-condição:

1. Usuário autenticado no sistema;
2. Acessar um projeto.

Fluxo principal:

1. Clicar em “Use case”;
2. Abre o diagrama de casos de uso, suas associações e seus *slices*;

Pós-condição: visualização do diagrama de casos de uso, suas associações e seus *slices*.

***Slice* AG02-UC01-SL01**

História de usuário HU01: eu como analista/programador quero visualizar o diagrama dos casos de uso de um projeto. O diagrama deve ter as associações entre atores, casos de uso e *slices*.

Caso de uso AG02-UC02: visualizar detalhes dos componentes do diagrama.

Ator: analista

Pré-condição:

1. Usuário autenticado no sistema;
2. Acessar um projeto;
3. Abrir diagrama de casos de uso.

Fluxo principal:

1. Duplo clique sobre um caso de uso;
2. Abre a tela de edição do caso de uso.

Fluxo alternativo 01:

1. Duplo clique sobre um *slice*;
2. Abre a tela de edição do *slice*.

Pós-condição: tela com as informações do componente selecionado aberta.

***Slice* AG02-UC02-SL01**

História de usuário HU01: eu como analista/programador quero poder acessar as informações de um caso de uso através do diagrama de casos de uso.

***Slice* AG02-UC02-SL02**

História de usuário HU02: eu como analista/programador quero poder acessar as informações de um *slice* através do diagrama de casos de uso.

Caso de uso AG02-UC03: filtrar casos de uso e *slices* no quadro Kanban.

Ator: analista
Pré-condição:
1. Usuário autenticado no sistema;
2. Acessar um projeto;
3. Abrir o quadro Kanban clicando na aba “Board”.

Fluxo principal:
1. Clicar sobre o símbolo do filtro;
2. Selecionar entre os filtros disponíveis a opção desejada para filtrar casos de uso.

Fluxo alternativo 01:
1. Selecionar entre os filtros disponíveis a opção desejada para filtrar *slices* .

Fluxo alternativo 02: Selecionar entre os filtros disponíveis a opção desejada para filtrar casos de uso e *slices*.

Pós-condição: tela com os *slices* a mostra ou escondidos.

***Slice* AG02-UC03-SL01**
História de usuário HU01: eu como analista/programador quero poder filtrar os casos de uso e *slices* existentes.