

# CENTRO PAULA SOUZA

---

**FACULDADE DE TECNOLOGIA DE AMERICANA  
ANÁLISE E DESENVOLVIMENTO DE SISTEMA**

LURAM ARCHANJO

**SOFTWARE PARA GERENCIAMENTO UTILIZANDO SCRUM**

**AMERICANA-SP**

**2015**

LURAM ARCHANJO

## **SOFTWARE PARA GERENCIAMENTO UTILIZANDO SCRUM**

Trabalho de Conclusão de Curso desenvolvido em cumprimento à exigência curricular do Curso Analise e Desenvolvimento de Sistema, sob a orientação do Prof. Me. Wladimir da Costa.

Área de concentração: Engenharia de Software

**AMERICANA-SP**

**2015**

## DEDICATÓRIA

Aos meus pais, irmãos, minha namorada  
Caroline Teixeira, e todos meus amigos  
que me apoiaram neste trabalho.

## **AGRADECIMENTOS**

Agradeço a todos os professores por me proporcionar o conhecimento não apenas racional, mas a manifestação do caráter e afetividade da educação no processo de formação profissional, por tanto que se dedicaram a mim, não somente por terem me ensinado, mas por terem me feito aprender. A palavra mestre, nunca fará justiça aos professores dedicados aos quais sem nominar terão os meus eternos agradecimentos.

## RESUMO

Objetivo deste trabalho é o desenvolvimento de um software para gerenciamento de projetos que utilizam a metodologia Scrum, metodologia a qual visa há entrega rápida, portanto este trabalho agrega todos os elementos da metodologia Scrum para o acompanhamento do projeto, tais como controle de *Product Backlog*, *Sprint Backlog*, *Sprint*, Tarefas e funcionários e também proporciona o quadro Kanban, calendário e relatório. A metodologia utilizada para o desenvolvimento deste trabalho foi a metodologia de Engenharia Reversa, ou seja, primeiro se desenvolveu o software em seguida a partir das informações do software se desenvolveu a documentação necessária, tais como diagrama de classe, atividade, sequencia, caso de uso, diagrama entidade relacionamento e um manual sobre o software para melhor compreensão dos novos usuários. Os resultados esperados deste trabalho foram alcançados com êxito, tais como a implementação de todos os elementos da metodologia Scrum para o acompanhamento do projeto e a integração dos mesmo.

**Palavras Chave:** Scrum, Diagramas, Programa.

## **ABSTRACT**

*This study aims to develop a software for managing projects that use the methodology Scrum methodology which aims for fast delivery, so this work combines all the elements of the methodology Scrum for monitoring the project, such as control product backlog, Sprint Backlog, Sprint, Tasks and employees and also provides the kanban board, schedule and report. The methodology used to develop this work was the methodology of reverse engineering, that is, first developed the software then from the software information has developed the necessary documentation, such as a class diagram, activity, sequence, use case , entity relationship diagram and a manual on the software to better understand the new users. The expected results of this work have been met with success, such as the implementation of all elements of the Scrum methodology for monitoring the project and the integration of it.*

**Keywords:** *Scrum, Diagrams, Software.*

## LISTA DE FIGURAS

Figura 1 - Esquema genérico de um subsistema.....	3
Figura 2 - Ciclo de um desenvolvimento de software.....	4
Figura 3 - Desenvolvimento Cascata.....	6
Figura 4 - Desenvolvimento Evolucionário.....	7
Figura 5 - Desenvolvimento Espiral.....	8
Figura 6 - Desenvolvimento Incremental.....	9
Figura 7 - Ciclo de uma Sprint.....	15
Figura 8 - Kanban.....	17
Figura 9 - Gráfico Burndown.....	18
Figura 10 - Gráfico Chaos.....	22
Figura 11 - Exemplo de Caso de Uso.....	27
Figura 12 - Exemplo Diagrama de Classe.....	28
Figura 13 - Exemplo Diagrama de Atividade.....	28
Figura 14 - Exemplo Diagrama de Sequência.....	29
Figura 15 - Exemplo Diagrama Entidade Relacionamento.....	30
Figura 16 – Tela Carregamento.....	62
Figura 17 – Tela Login.....	63
Figura 18 – Tela Recuperar Senha.....	64
Figura 19 – E-mail recuperar senha.....	64
Figura 20 –Tela Kanban.....	65

Figura 21 –Tela Kanban Tarefa.....	67
Figura 22 –Tela Burndown.....	68
Figura 23 – Tela Conta.....	70
Figura 24 – E-mail alterar conta.....	71
Figura 25 – Tela Validação.....	71
Figura 26 – Tela Lista de Funcionário.....	72
Figura 27 –Tela Funcionário.....	74
Figura 28 – E-mail criar conta.....	75
Figura 29 – Tela Alterar Senha.....	75
Figura 30 – Tela Lista Product Backlog.....	76
Figura 31 – Tela Product Backlog.....	77
Figura 32 – Tela Lista de Sprint Backlog.....	79
Figura 33 – Tela Sumário.....	80
Figura 34 –Tela Backlog.....	81
Figura 35 – Tela Confirmar Escolha.....	82
Figura 36 – Tela Tarefa.....	83
Figura 37 –Tela Sprint.....	84
Figura 38 –Tela Cor.....	85
Figura 39 – Legenda.....	85
Figura 40 – Tela Calendário.....	86
Figura 41 –Tela Configuração.....	87



<b>Figura 42 –Tela Histórico.....</b>	<b>88</b>
<b>Figura 43 – Tela Relatório.....</b>	<b>89</b>
<b>Figura 44 – Relatório.....</b>	<b>90</b>

## **LISTA DE TABELAS**

<b>Tabela 1 - Comparação de ferramentas de Scrum.....</b>	<b>19</b>
---	-----------

## **LISTA DE ABREVIATURAS E SIGLAS**

**CMMI** Modulo de Maturidade em Capacitação

**FAQ** Frequently Asked Questions

**MER** Modelo Entidade Relacionamento

**SGBD** Sistema de Gerenciamento de Banco de Dados

**UML** Unified Modeling Language

**XP** Extreme Programming

## SUMÁRIO

INTRODUÇÃO	1
1 SISTEMA DE INFORMAÇÃO	3
1.2 PROCESSO DE DESENVOLVIMENTO DE SISTEMA	4
1.2.1 CASCATA	5
1.2.2 EVOLUCINÁRIO	6
1.2.3 ESPIRAL	7
1.2.4 INCREMENTAL	8
1.3 METODOLOGIA ÁGIL	10
1.3.1 EXTREME PROGRAMMING	11
1.3.2 SCRUM	14
1.3.2.1 COMPONENTES DO MÉTODO SCRUM	15
1.4 FERRAMENTAS SCRUM	18
1.5 GESTÃO DE PROJETOS	19
2 LEVANTAMENTO DE REQUISITOS	23
2.1 REQUISITOS FUNCIONAIS	23
2.2 REQUISITOS NÃO FUNCIONAIS	23
2.2.1 CONFIABILIDADE	23
2.2.2 PORTABILIDADE	24
2.2.3 SEGURANÇA	24
2.2.4 DESEMPENHO	24
2.2.5 USABILIDADE	24
2.2.6 TIPO DE INTERFACE	24
2.2.7 DOCUMENTAÇÃO NECESSÁRIA	24
2.2.8 REQUISITO DE INSTALAÇÃO	25
3 FERRAMENTAS PARA DOCUMENTAÇÃO DE PROJETO	26
3.1 UML	26
3.2 DIAGRAMA ENTIDADE RELACIONAMENTO	29
4 ESTUDO DE CASO	31
4.1 CASO DE USO	32
4.2 CLASSE	33
4.2.1 FUNCIONÁRIO	33
4.2.2 USUÁRIO	34

<b>4.2.3</b>	<b>PRODUCT BACKLOG</b>	<b>35</b>
<b>4.2.4</b>	<b>SPRINT BACKLOG</b>	<b>36</b>
<b>4.2.5</b>	<b>TAREFA</b>	<b>37</b>
<b>4.2.6</b>	<b>SPRINT</b>	<b>38</b>
<b>4.3</b>	<b>DIAGRAMA DE SEQUENCIA</b>	<b>39</b>
<b>4.3.1</b>	<b>CONTROLE DE FUNCIONÁRIO</b>	<b>39</b>
<b>4.3.2</b>	<b>CONTROLE DE CONTA</b>	<b>40</b>
<b>4.3.3</b>	<b>CONTROLE DE PRODUCT BACKLOG</b>	<b>41</b>
<b>4.3.4</b>	<b>CONTROLE DE SPRINT BACKLOG</b>	<b>42</b>
<b>4.3.5</b>	<b>CONTROLE DE SPRINT</b>	<b>43</b>
<b>4.3.6</b>	<b>CONTROLE DE KANBAN</b>	<b>44</b>
<b>4.3.7</b>	<b>CONTROLE DE TAREFA</b>	<b>45</b>
<b>4.3.8</b>	<b>VISUALIZAR BURNDOWN</b>	<b>46</b>
<b>4.3.9</b>	<b>VISUALIZAR HISTÓRICO</b>	<b>47</b>
<b>4.4</b>	<b>DIAGRAMA DE ATIVIDADE</b>	<b>48</b>
<b>4.4.1</b>	<b>CONTROLE DE CONTA</b>	<b>48</b>
<b>4.4.2</b>	<b>CONTROLE DE FUNCIONÁRIO</b>	<b>50</b>
<b>4.4.3</b>	<b>CONTROLE DE PRODUCT BACKLOG</b>	<b>52</b>
<b>4.4.4</b>	<b>CONTROLE DE SPRINT BACKLOG</b>	<b>54</b>
<b>4.4.5</b>	<b>CONTROLE DE SPRINT</b>	<b>56</b>
<b>4.4.6</b>	<b>CONTROLE DE TAREFA</b>	<b>57</b>
<b>4.4.7</b>	<b>CONTROLE DE KANBAN</b>	<b>59</b>
<b>4.5</b>	<b>MODELO ENTIDADE RELACIONAMENTO (MER)</b>	<b>61</b>
<b>4.6</b>	<b>SOFTWARE</b>	<b>62</b>
<b>4.6.1</b>	<b>TELA CARREGAMENTO</b>	<b>62</b>
<b>4.6.2</b>	<b>TELA LOGIN</b>	<b>63</b>
<b>4.6.3</b>	<b>TELA RECUPERAR SENHA</b>	<b>64</b>
<b>4.6.5</b>	<b>TELA KANBAN TAREFA</b>	<b>67</b>
<b>4.6.6</b>	<b>TELA BURNDOWN</b>	<b>68</b>
<b>4.6.7</b>	<b>TELA CONTA</b>	<b>70</b>
<b>4.6.8</b>	<b>TELA LISTA DE FUNCIONÁRIO</b>	<b>72</b>
<b>4.6.9</b>	<b>TELA FUNCIONÁRIO</b>	<b>74</b>
<b>4.6.10</b>	<b>TELA LISTA PRODUCT BACKLOG</b>	<b>76</b>
<b>4.6.11</b>	<b>TELA PRODUCT BACKLOG</b>	<b>77</b>

<b>4.6.12 TELA LISTA DE SPRINT BACKLOG</b>	<b>79</b>
<b>4.6.13 TELA SUMÁRIO</b>	<b>80</b>
<b>4.6.14 TELA BACKLOG</b>	<b>81</b>
<b>4.6.15 TELA TAREFA</b>	<b>83</b>
<b>4.6.16 TELA SPRINT</b>	<b>84</b>
<b>4.6.17 TELA CALENDÁRIO</b>	<b>86</b>
<b>4.6.18 TELA CALENDÁRIO CONFIGURAÇÕES</b>	<b>87</b>
<b>4.6.19 TELA HISTÓRICO</b>	<b>88</b>
<b>4.6.20 TELA RELATÓRIO</b>	<b>89</b>
<b>CONSIDERAÇÕES FINAIS</b>	<b>91</b>
<b>REFERÊNCIAS BIBLIOGRÁFICAS</b>	<b>92</b>

## INTRODUÇÃO

Com o crescimento da tecnologia na última década, e sua implantação nas corporações como um meio de alavancar seus negócios, cresce a necessidade de buscar por melhores resultados em seus processos, tais como, processo produtivo, qualitativo e organizacional.

Os principais problemas enfrentados pelas empresas desenvolvedoras de softwares são produto de baixa qualidade, devido os atrasos nas entregas, aumento nos orçamentos, os quais, talvez, são gerados pela falta de gerenciamento de processos ou pelo mal gerenciamento do mesmo, portanto, surge uma necessidade no mercado para melhoria de processo, surgindo várias metodologias da área de engenharia de software, metodologia estruturada, orientada a objetos e ágil, porém, a aplicação dos mesmo é cara e complexa.

Com o passar dos anos, foi se popularizando as metodologias ágeis, como Scrum e o *Extreme Programming* (TELES,2014), devidas as aplicações de grande importância para as corporações, tais como, adaptabilidade as mudanças do projeto, interação com o cliente, interação com os membros da equipe de desenvolvimento, desenvolvimento incremental com a validação do cliente e entregas constantes de pequenas partes funcionais do software.

O objetivo deste trabalho é o desenvolvimento de uma aplicação desktop, específica para gerenciamento de projetos que utilizam a metodologia Scrum.

Este trabalho procura solucionar os problemas relacionados ao gerenciamento de projetos, se especificando em projetos relacionados a Scrum, auxiliando todo os membros envolvidos *Product Owner*, *Scrum Master* e *Scrum Team*, por meios de princípios que o Scrum segue como: *Burndown*, Kanban, *Daily Scrum* e *Sprint*.

Para realização deste trabalho utilizei como fonte de pesquisa livros e sites referente ao assunto Scrum, Engenharia de Software, UML e Sistema de Informação. Para concluir o mesmo utilizei a Engenharia Reversa, utilizando como plataforma de desenvolvimento a IDE NetBeans 8.0.2 o qual utilizei a linguagem de programação Java em sua versão 8u65, para integração do banco de dados utilizei o banco de dados relacional Firebird em sua versão 2.5 para o gerenciamento do mesmo utilizei

o SGBD IBExpert em sua versão 2013.2.15.1, para melhor desempenho da aplicação e manutenção do mesmo utilizei duas técnicas de programação a *Singleton* o qual tem como objetivo apenas ter uma instancia por classe em memória e para fins de manutenção e compreensão a programação em camadas.

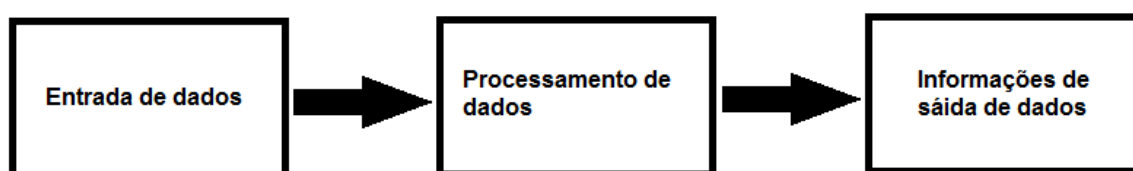
Este trabalho foi estruturado em quatro capítulos, sendo que o primeiro conceitua o sistema de informação e os principais processos de desenvolvimento de software cuja abrange o desenvolvimento ágil, o qual a metodologia Scrum se baseia, as ferramentas Scrum e conceitua gestão de projetos e suas dificuldades, o segundo conceitua os levantamentos de requisitos tais como requisitos funcionais e não funcionais que devem ser implementados no desenvolvimento deste trabalho, o terceiro conceitua ferramentas para documentação de software tais como UML e seus principais diagramas e o diagrama entidade relacionamento responsável pela documentação da estrutura do banco de dados, no quarto se descreve todas as documentações deste trabalho, tais como diagrama de caso de uso, classe, sequencia, atividade e diagrama entidade relacionamento e também um manual referente a este trabalho.



## 1 SISTEMA DE INFORMAÇÃO

Segundo L. Von Bertalanffy (1977), “Um sistema pode ser definido como um complexo de elementos em interação”, visando os setores empresariais, vendas, contabilidade, financeiro, produção, podemos dizer que sistema de informação dentro de uma corporação, são subsistemas que se interagem para um determinado fim, tal como controle de estoque, o qual necessita de um subsistema de venda, subsistema de cadastro de produto, cada qual compostos por três etapas (Figura 1), entrada de dados (venda de um produto ou cadastro de um produto), processamento de dados (alocar produto a contas a receber, retirar o produto do estoque), saída de dados (nota fiscal , balanço de estoque).

**Figura 1: Esquema genérico de um subsistema.**



**Fonte: Próprio autor**

Segundo Sérgio Rodrigues Bio (1991)

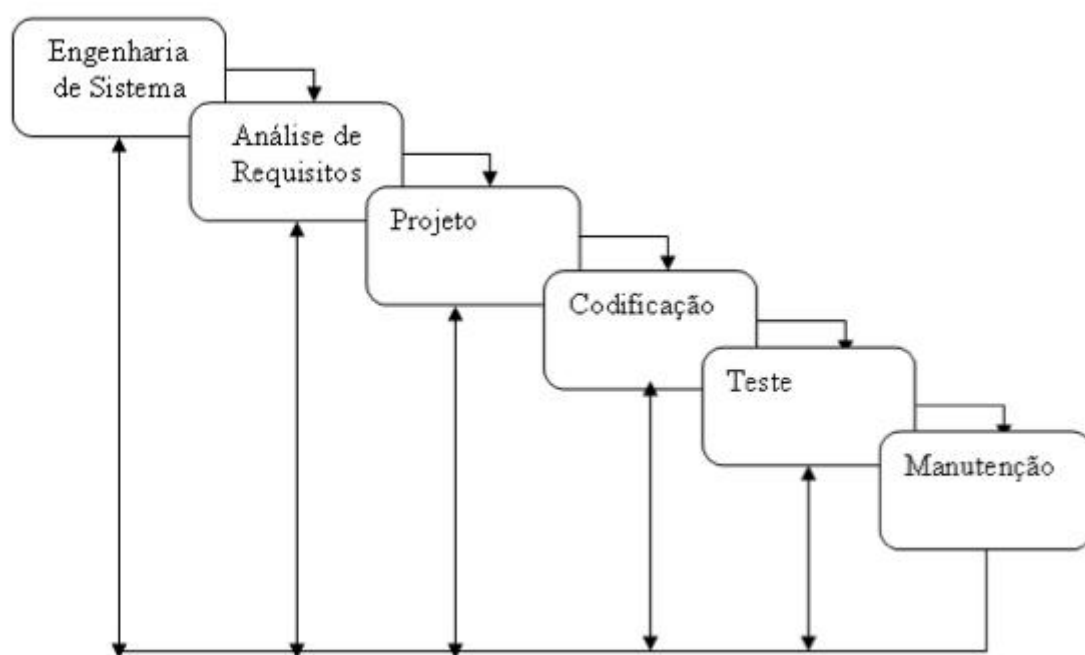
Os dados de entrada, uma vez coletados, são transportados (com base em procedimentos) até o ponto de processamento. Esse processamento pode utilizar meios manuais, mecânicos ou eletrônicos. De qualquer forma, o processamento será sempre um série de operações necessárias para registrar dados e convertê-los em todas as informações de saída desejáveis

Portando um sistema de informações é um conjunto de procedimentos (subsistemas) lógicos, os quais são repetitivas, iniciadas, executadas e controladas e finalizadas, como uma loja, a qual tem seu fluxo, compra de produto, estoca o produto, vende o produto, verifica o estoque, verifica o preço, emite nota e balance mensal/semanal de vendas.

## 1.2 Processo de Desenvolvimento de Sistema

Com o crescimento da demanda por sistemas, e o desenvolvimento do mesmo surgiram inúmeros meios de processos de desenvolvimento de sistema, os quais são um conjunto de práticas adotadas para um bom desenvolvimento de um sistema, os quais são divididas em seis principais etapas, como, engenharia do sistema, extração de requisito, projeto, desenvolvimento, fase de teste, implantação e manutenção/evolução do software (Figura 2).

**Figura 2: Ciclo de um desenvolvimento de software**



Fonte: Rogers S. Pressman, 2007, p.33.

Cada processo gerencia essas atividades de acordo com suas respectivas regras as quais os diferenciam umas das outras, cada qual para um específico modo de processo e de uma perspectiva particular.

Segundo Ian Sommerville (2007), “Diferentes processos de software organizam essas atividades de maneiras diversas e são descritos em diferentes níveis de detalhes[...]”.

Porem genericamente a etapa de especificação têm como objetivo extrair as funcionalidades esperadas/exigidas e suas restrições e as especificarem para melhor compreensão pela parte dos integrantes o que se deve ser desenvolvido o qual é a

fase de desenvolvimento, após o termino de desenvolvimento as especificações serão validas pelo cliente e logo após esta etapa o produto final é implantado assim entrando na última fase do ciclo, manutenção e evolução do sistema o qual tem o objetivo de aprimorar o sistema e manter ele funcionando.

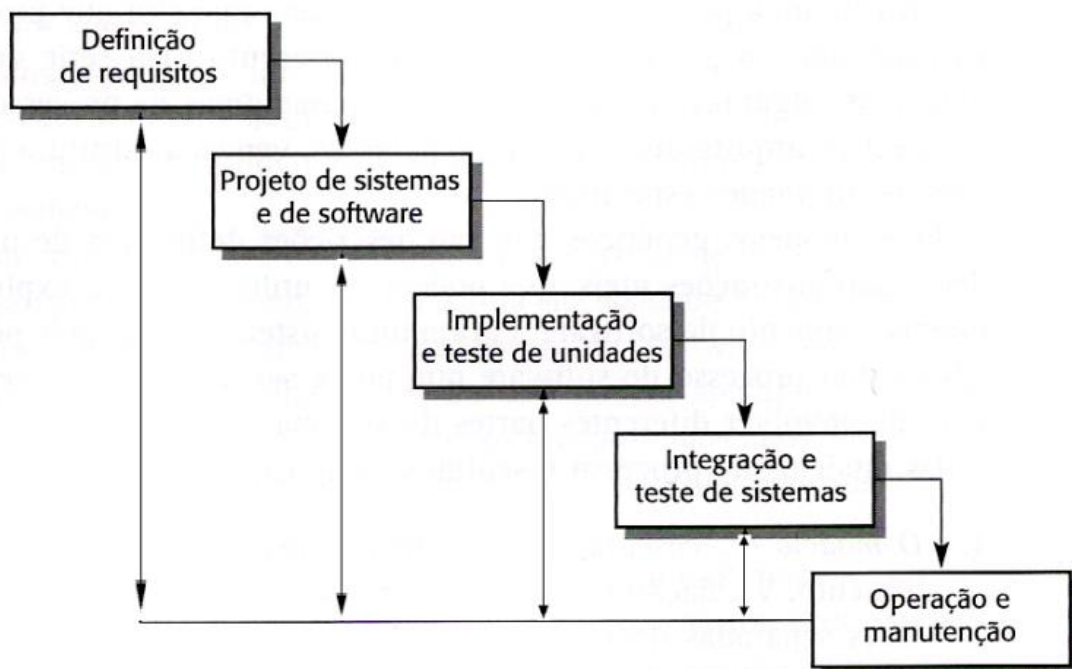
Os processos mais conhecidos dentro da área de engenharia de software são o cascata, evolucionário, incremental e espiral.

### 1.2.1 Cascata

O modelo cascata separa as principais etapas de desenvolvimento de software por ordem de ciclo (sequencial), onde somente é possível iniciar uma nova etapa quando a anterior é finalizada (Figura 3).

Segundo Rogers S. Pressman (2007), “O Modelo Cascata nasceu de um visão industrial de construção. Ou seja, foi baseado em uma ordem sequencial e de cima para baixo de suas atividades”.

**Figura 3: Desenvolvimento Cascata**



**Fonte:** Ian Sommerville, 2007, p.44.

O modelo cascata não vêm sendo muito utilizado, devido sua metodologia ser sequencial, ou seja, caso na primeira fase do processo, a fase de definição de requisitos o cliente omite um requisito, devido ele estar acostumado com o processo

em sua empresa, o mesmo somente será abordado na fase de desenvolvimento e de teste, portanto é necessário retomar do início, gerando problemas relacionados a prazo.

Segundo Rogério Magela (2006),

Difícilmente é possível resolver um problema somente no nível de Análise de Sistema. Ou seja, as maiores contradições e inconsistências são descobertas na programação. Sendo assim, há uma grande chance de que, após meses de trabalho em análise, e até mesmo após o seu término, já iniciada a codificação, venhamos a descobrir sérios problemas que forcem uma retomada e uma reanálise do problema.

### 1.2.2 Evolucionário

O modelo evolucionário visa desenvolver um software inicialmente com base em requisitos abstratos (Figura 4), ou seja, requisitos genéricos tais como, tela de login, cadastro de pessoas; Após o desenvolvimento dos requisitos abstratos é criado um versão e exibida ao cliente com o objetivo de extrair novos requisitos, e assim a cada nova versão, ou seja, a cada desenvolvimento de um requisito e mostrado um nova versão do produto, até chegar o produto final, o qual é o produto desejável pelo cliente.

Segundo Ian Sommerville (2007),

O desenvolvimento evolucionário baseia-se na ideia de desenvolvimento de uma implementação inicial, expondo o resultado aos comentários do usuário e refinando esse resultado por meio de várias versões até que seja desenvolvido um sistema adequado.

**Figura 4: Desenvolvimento Evolucionário**



Fonte: Ian Sommerville, 2007, p.46.

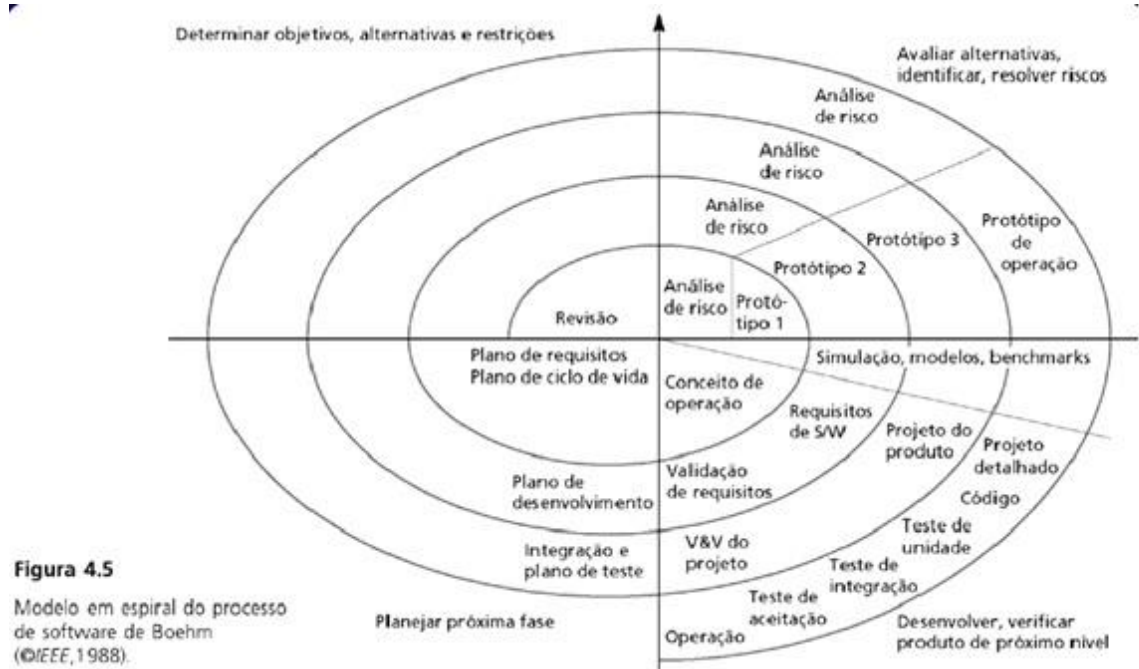
Existem dois meios no modelo evolucionário de se extrair requisitos um denominado de desenvolvimento exploratório que tem como objetivo, desenvolver o software juntamente com o usuário afim de aprimorar e extrair novos requisitos, outro meio é conhecido como Prototipação *throwaway* (descartável), o qual o objetivo é a compreensão dos requisitos do cliente e desenvolver e especificar melhor os requisitos requeridos, o protótipo serve para compreender os requisitos omitidos ou mal compreendidos.

A consequência de se utilizar o processo evolucionário, é a mal estruturação da codificação e documentação devido as implementações de novos requisitos, porém o processo evolucionário tende a ser rápido e satisfatório.

### 1.2.3 *Espiral*

O processo espiral dentro de um desenvolvimento é semelhante ao cascata, pois, seu desenho de espiral é dividido em quatro etapas e consequentemente segue uma ordem, porém sua espiral é um loop, o primeiro quadrante da espiral tem como objetivo definir o objetivo do projeto, em seguida é feita a avaliação dos riscos que podem ocorrer durante o projeto e a redução dos mesmos, em seguida tem-se a fase de desenvolvimento, ao final da espiral o projeto é revisto e se verifica se o projeto prossegue (Figura 5).

**Figura 5: Desenvolvimento Espiral**



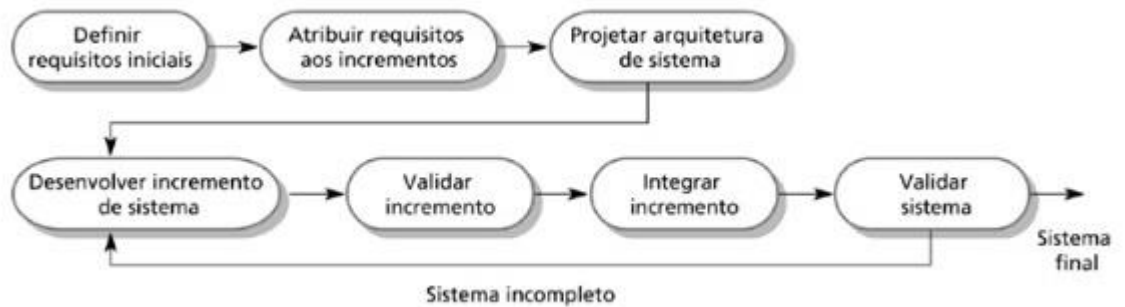
Fonte: Ian Sommerville, 2007, p.49.

#### 1.2.4 Incremental

O processo incremental utiliza princípios do modelo cascata onde cada funcionalidade requerida pelo cliente em nível de prioridade é entregue, e também utiliza princípios do modelo evolucionário devido cada funcionalidade desenvolvida ser colocada em produção, ou seja, por meios de versões o software vai se incrementando até todas as funcionalidades requeridas serem incrementadas (Figura 6) até chegar no produto final.

Segundo Rogério Magela (2006), “Essa abordagem está baseada no incremento contínuo de funcionalidades do software ao longo do desenvolvimento”

**Figura 6: Desenvolvimento incremental**



**Fonte:** Ian Sommerville, 2007, p.47.

O processo incremental tem um ganho referente aos processos abordados anteriores (cascata, evolucionário, espiral), devido de forma gradativamente reduzir os riscos de um projeto e os constantes contatos com o usuário em suas versões, porém a falta de prioridade em suas funcionalidades podem levar o desenvolvimento ao desastre, pois uma funcionalidade pode interferir no fluxo de outras funcionalidades.

### 1.3 Metodologia Ágil

Com a globalização os negócios em qualquer âmbito tendem a mudar de requisitos para inúmeros fins, devido as práticas de desenvolvimento de software visando a especificação de requisitos, documentação, planejamento e testes serem conhecidas e adotadas no mercado mundialmente, ao se mudar um requisito é necessário refazer toda a especificação de requisitos, documentação, planejamento e teste, gerando irritabilidade aos desenvolvedores envolvidos, surgindo a necessidade de um processo que visa a entrega rápida de um software, assim surge as metodologias ágeis.

Segundo Ian Sommerville (2007),

Os processos de desenvolvimento de software baseados em especificações completas de requisitos, projeto, construção e teste de sistema não estão voltados para desenvolvimento rápido de software. Quando os requisitos mudam ou quando os problemas com requisitos são descobertos, o projeto de sistema ou a implementação precisa ser retrabalhada e retestada. Como consequência, um processo em cascata convencional ou baseado em especificações é geralmente prolongado e o software final é entregue para o cliente muito depois de ter sido originalmente especificado.

Em um ambiente de negócios de rápidas mudanças, isso pode causar problemas reais. No momento em que o software estiver disponível para uso, a razão original para sua aquisição pode ter mudado tão radicalmente que o software será inútil. Portanto, para sistemas de negócios específicos, processos de desenvolvimento voltados para desenvolvimento e entrega rápidos são essenciais.

Com a insatisfação gerada pelos processos adotados para desenvolvimento de software um grupo de desenvolvedores decidiram se reunir, para abordar e discutir sobre as práticas utilizadas pelos mesmos para um bom desenvolvimento de um software, chegando a princípios a serem seguidos para se obter sucesso em um projeto os quais são, envolvimento do cliente, entrega incremental, reconhecimento e exploração de habilidades, estar ciente a mudanças e manter a simplicidade visando sua usabilidade.



Segundo Ian Sommerville (2007),

A insatisfação com essas abordagens pesadas levou um número de desenvolvedores de software na década de 1990 a propor novos métodos ágeis. Estes permitiram que a equipe de desenvolvimento se concentrassem no software somente, em vez de em seu projeto e documentação.

Esta ação dos desenvolvedores é conhecida como Manifesto Ágil, após o manifesto se criou vários métodos ágeis, os quais se popularizaram *Extreme Programming* (XP) e Scrum.

Segundo Vinicius Manhães Teles (2008),

Há alguns anos, um grupo de profissionais veteranos na área de software decidiram se reunir em uma estação de esqui, nos EUA, para discutir formas de melhorar o desempenho de seus projetos.

Embora cada envolvido tivesse suas próprias práticas e teorias sobre como fazer um projeto de software ter sucesso, cada qual com as suas particularidades, todos concordavam que, em suas experiências prévias, um pequeno conjunto de princípios sempre parecia ter sido respeitado quando os projetos davam certo.

Com base nisso eles criaram o Manifesto para o Desenvolvimento Ágil de Software, freqüentemente chamado apenas de Manifesto Ágil, e o termo Desenvolvimento Ágil [...].

### 1.3.1 *Extreme Programming*

XP é um processo de desenvolvimento ágil, portanto ele se baseia nos princípios da metodologia ágil citado no tópico anterior, e suas características são ciclos curtos, constantes *feedbacks*, o qual adota o modelo incremental, com o objetivo de desenvolver o software em partes, de acordo com os *feedbacks*, assim reduzindo os riscos de extração de requisitos errados e atrasos em entregas.

Para se desenvolver em XP, é preciso ter coragem, devido o desenvolvimento ágil visar entregas rápidas, seguras e de qualidade, portanto é preciso para enfrentar os desafios de manter a qualidade, segurança e agilidade para se entregar o produto a ser desenvolvido.

O *feedback* faz com que os desenvolvedores e cliente estejam cientes do que se espera para aprimorar o produto, ou seja, constantemente o cliente está ligado ao projeto, assim agilizando suas entregas, devido especificar melhor as funcionalidades requeridas.

A comunicação em XP, e em outras metodologias é de extrema importância, porém em desenvolvimentos ágeis é essencial, devido o cliente estar diretamente ligado ao projeto, e em XP, para reduzir as chances de erros na codificação se criou-se a programação em par, onde um desenvolve e o outro companheiro verifica a codificação, portanto a comunicação e ciência do que deve ser desenvolvido é de extrema importância.

XP têm suas variáveis de controle as quais são, custo, prazo, qualidade, escopo. Os custos têm dois lados da moeda, dinheiro em excesso traz problemas para um projeto e dinheiro em falta pode fazer o seu projeto nem ser inicializado, portanto o XP visa controlar esta variável.

Projetos longos tendem a acumular erros e baixa a qualidade do seu produto, portanto XP, visa prazos pequenos com constantes *feedbacks* do cliente para reduzir os prazos e eliminar os erros consequentemente aumentar a qualidade do produto final.

O escopo, quanto menor o escopo menor o custo e de melhor qualidade ficará o produto final.

Qualidade, para todos os processos de desenvolvimento de software a qualidade é o aspecto mais esperado, porém, um produto de alta qualidade requer tempo e dinheiro. O XP tenta reduzir o custo, tempo e manter a alta qualidade, portanto a metodologia XP visa reduzir os principais erros de um desenvolvimento de software os quais são as variáveis de controle, custos, prazos, qualidade, escopo e para isto criou-se as práticas de XP. Segundo Ian Sommerville (2008), esse processo é constituído das seguintes etapas.

**1. Jogo de Planejamento:** o objetivo desta prática é determinar o escopo da próxima versão de acordo com os *feedbacks* do cliente, e determinar o tempo para se desenvolver os requisitos.

**2. Pequenas Versões:** Quanto menor a versão, mais rápido ela será desenvolvida, gerando *feedbacks* rápidos, e redução de erros.

**3. Metáfora:** Utilizando metáforas permite interligar a equipe de desenvolvimento ao cliente, e reforçar o que se deve ser feito, reduzindo os erros de mal compreensão.

**4. Projeto Simples:** Com projetos simples e possível efetuar testes efetivos e manter a integridade do produto facilitando a manutenção e evolução do mesmo.

**5. Testes Constantes:** Com projetos simples, versões pequenas, é preciso a cada nova versão efetuar testes, portanto é preciso testar constantemente assim reduzindo os erros de fluxos e mal compreensão de requisitos.

**6. Reestruturação:** Com muitas mudanças devido a várias versões é preciso reestruturar a codificação assim facilitando a manutenção e evolução do software.

**7. Programação em Duplas:** Programação em duplas reduz o erro de codificação e mal compreensão do que se deve ser desenvolvido.

**8. Propriedade Coletiva:** Todos os desenvolvedores partilham o mesmo código, assim aumentando a interação e desenvolvimento em equipe e reduz o risco de dependência de desenvolvedor.

**9. Interação Contínua:** Integrando as novas versões é preciso testa-las assim obtém-se mais *feedbacks* e constantes testes.

**10. Semana de 40 horas:** Determinar um total de horas a se trabalhar semanalmente mantém a equipe em produtividade, assim reduzindo o prazo de entrega e a interação completa da equipe no projeto.

**11. Cliente Local:** Mantendo um cliente local na área de desenvolvimento garante o que está sendo desenvolvido é o que realmente ele deseja, assim reduzindo os erros de mal compreensão de requisitos.

**12. Padrão de Codificação:** Manter regras de codificação em um projeto e de extrema importância pois elimina a dependência de um programador específico e facilita a manutenção e evolução do software.

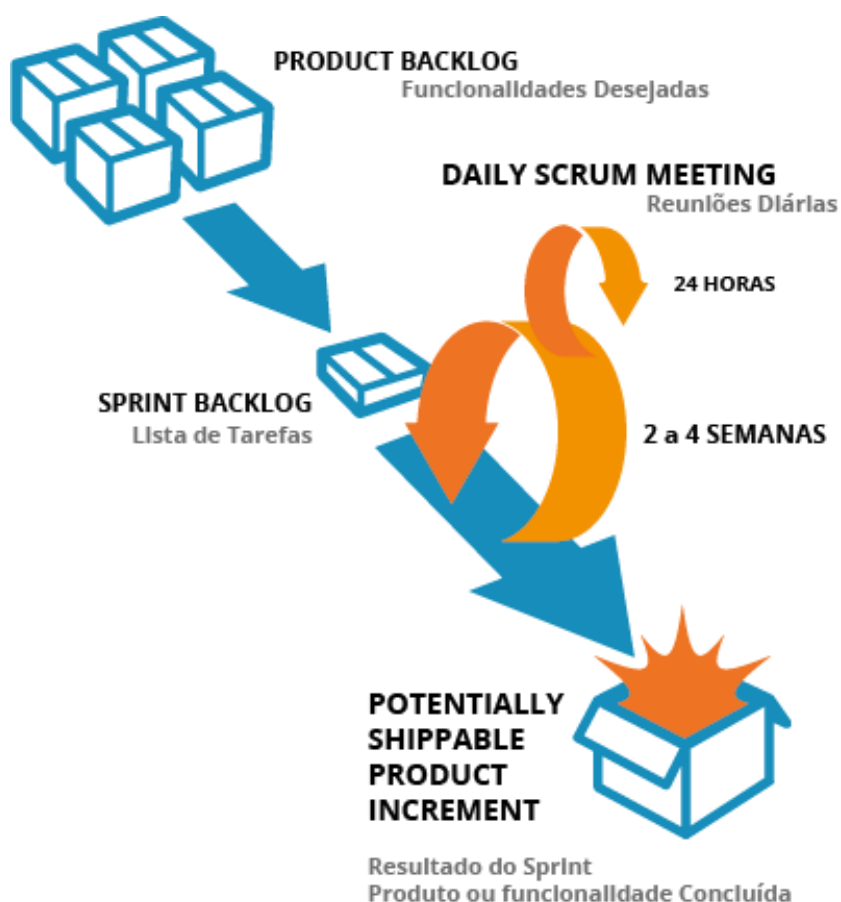
### 1.3.2 Scrum

Scrum é uma metodologia ágil o qual é dividida em ciclos (figura 7) que são nomeadas como *Sprint*, geralmente uma *Sprint* dura entre duas a quatro semanas visando a agilidade, gerenciamento e planejamento do projeto, pois cada *Sprint* são quebradas em tarefas o qual um conjunto de tarefas concluídas chega-se a uma funcionalidade requerida pelo cliente, portanto uma *Sprint* pode ter como objetivo desenvolver uma ou inúmeras funcionalidades.

Segundo Vinícius Manhães Teles (2014),

No Scrum, os projetos são divididos em ciclos (tipicamente mensais) chamados de Sprints. O Sprint representa um Time Box dentro do qual um conjunto de atividades deve ser executado. Metodologias ágeis de desenvolvimento de software são iterativas, ou seja, o trabalho é dividido em iterações, que são chamadas de Sprints no caso do Scrum.

Figura 7: Ciclo de uma Sprint



Fonte: <http://14268.http.cdn.softlayer.net/8014268/www.brq.com//wp-content/uploads/2013/10/scrum-por.png?d04945>

Como Scrum visa a agilidade, gerenciamento e planejamento de um projeto, tem-se três papéis definidos. Segundo Vinícius Manhães Teles (2014), os três papéis dentro do Scrum são:

*Product Owner*: Responsável por extrair requisitos os quais compõem o *Product Backlog* e são ordenados por prioridade, definidos na *Sprint Planning Meeting*, após o início de uma *Sprint* o *Product Owner* não pode trazer novos requisitos apenas para as próximas *Sprints*, ou seja, *Product Owner* é o vínculo entre o cliente e a equipe.

*Scrum Master*: Responsável por gerenciar a equipe de desenvolvimento, ou seja, um gerente de equipe, porém esta função pode ser destinada a qualquer membro da equipe, pois o *Scrum Master* visa assegurar que a equipe siga os princípios do Scrum e também realiza o *Daily Scrum*, com o objetivo de identificar e remover os problemas relacionados ao projeto.

*Scrum Team*: Responsáveis pelo desenvolvimento, ou seja os desenvolvedores, porém em Scrum não há uma designação de funções tais como *designer*, analistas, "testers", ou seja, *Scrum Team* são membros de uma equipe de desenvolvimento os quais se comprometem a entregar funcionalidades de um *Sprint* e quebrá-las em tarefas os quais são definidas na *Sprint Planning Meeting*.

#### 1.3.2.1 Componentes do Método Scrum

Segundo Vinícius Manhães Teles (2014), o método Scrum é constituído pelas seguintes partes:

### **PRODUCT BACKLOG**

*Product Backlog* é composto pelas funcionalidades extraídas pelo *Product Owner*, o qual é uma lista de funcionalidades a serem desenvolvidas para se chegar ao produto final, o qual é ordenada por prioridade os quais são definidas pelo *Scrum Team* na *Sprint Planning Meeting*.

## **SPRINT PLANNING MEETING**

É uma reunião onde o *Product Owner* lista todas as funcionalidades requeridas pelo cliente para o *Scrum Master* e os membros do *Scrum Team*, os quais fazem perguntas referente as funcionalidades com o objetivo de quebra-las em tarefas técnicas, estas tarefas geram o *Sprint Backlog*, após as quebras de funcionalidades em tarefas o *Product Owner* junto com *Scrum Team* definem o objetivo da próxima *Sprint* e quanto eles podem se comprometer ao projeto, porém a responsabilidade de definir o tempo para se desenvolver uma funcionalidade é por parte dos membros do *Scrum team*, pois serão eles que vão se comprometer a desenvolver as tarefas necessárias para se obter a funcionalidade requerida.

## **SPRINT BACKLOG**

*Sprint Backlog* é uma lista de tarefas extraídas a partir das funcionalidades a serem desenvolvidas na *Sprint*, onde os membros do *Scrum Team* determina o tempo necessário para se desenvolver as tarefas. O *Scrum Master* se compromete a manter atualizado a *Sprint*, para isto o Scrum traz técnicas para gerenciar uma *Sprint* os quais são:

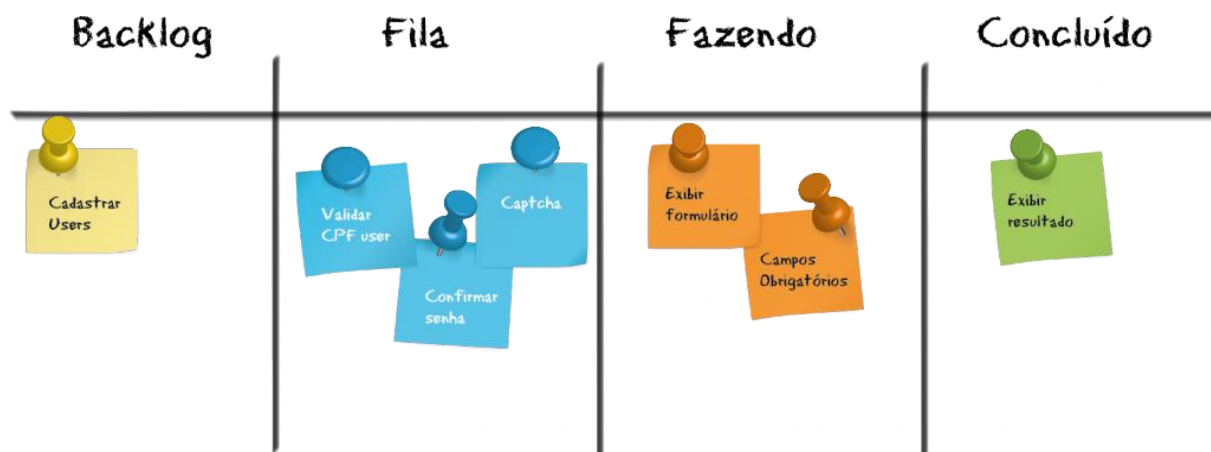
### **Daily Scrum**

Com objetivo de manter os membros do *Scrum Team* em conformidade com o planejamento da *Sprint*, é preciso ter ciência sobre ela, portanto o *Scrum Master* faz reuniões diárias com os membros e questiona três perguntas diretas, oque você fez ontem, oque você fará hoje, há algum impedimento em seu caminho. De acordo com as resposta o *Scrum Master* têm a responsabilidade de remover os impedimentos do projeto e manter a *Sprint* conforme o planejado.

### **Kanban**

O Kanban tem como objetivo separar as tarefas em estágios com um fluxo estabelecido, para o acompanhamento do mesmo, como podemos ver na Figura 8, quando uma tarefa é iniciada, ela passa do estágio de fila e vai para o estágio de fazendo, logo quando é finalizada passa para o estágio de concluído.

Figura 8: Kanban



Fonte: <http://14268.http.cdn.softlayer.net/8014268/www.brq.com//wp-content/uploads/2013/10/kanbam-1024x382.png?d04945>

### Gráfico *Burndown*

Para monitoramento da *Sprint*, o Scrum traz um gráfico denominado *Release Burndown Chart* (Figura 9), onde a responsabilidade de atualiza-lo é do *Scrum Master* de acordo com os dados extraídos do quadro Kanba.

Figura 9: Gráfico Burndown



Fonte: <http://14268.http.cdn.softlayer.net/8014268/www.brq.com//wp-content/uploads/2013/10/Imagem-3-Burn-Down-1024x807.png?d04945>

## ***SPRINT REVIEW MEETING***

Após o término de um *Sprint*, se faz uma reunião denominada de *Sprint Review Meeting*, onde participam o *Product Owner*, *Scrum Master*, *Scrum Team* e todas as partes interessadas no projeto, tais como gerência e clientes.

Durante a reunião avalia-se o objetivo da *Sprint* e as funcionalidades a serem desenvolvidas definidas na *Sprint Planning Meeting*, após as avaliações a uma retrospectiva, com o objetivo de extrair os pontos positivos e negativos referente a *Sprint* e verifica-se o que pode ser melhorado para a próxima *Sprint*.

### **1.4 Ferramentas Scrum**

Atualmente no mercado há inúmeras ferramentas para auxiliar o Scrum, as mais conhecidas e mais completas segundo Denisson Vieira (2014), são:

O ScrumHalf é uma ferramenta gratuita brasileira, que facilita muito o uso do Scrum, com quadro de Kanban virtual, facilitando muito a colaboração e o acompanhamento do trabalho de equipe. Permitindo também que a equipe trabalhe de uma forma distribuída simples.

A ferramenta também dispõe das funcionalidades de facilidade de manutenção e priorização do Product Backlog, geração automática do Burndown Chart e vários outros gráficos de controles.

O Pango Scrum também é uma ferramenta brasileira gratuita, onde é possível fazer o gerenciamento do Product Backlog, planejamento de Sprints, monitoramento de progresso, monitoramento de eventos por calendário, reuniões, planejamentos etc.

O Trello é uma ferramenta gratuita podendo optar por pagar para adquirir algumas vantagens, ela é simples mais de muita otimização, pois ela automatiza o quadro Kaban facilitando muito o uso Scrum.

Para ter um comparativo das ferramentas citadas acima, criei uma tabela de comparação (Tabela 1), onde compara os elementos do Scrum que as ferramentas possuem, tais como Kanban (Quadro de trabalho), *Product Backlog*, *Sprint* e demonstração de progresso.



**Tabela 1: Comparação de ferramentas de Scrum**

Ferramenta	Kanban	Product Backlog	Sprints	Progresso
Scrumhalf	+	+	-	-
Pangoscrum	-	-	+	+
Trello	+	-	-	-

Fonte: Autoria do autor.

## 1.5 Gestão de Projetos

Segundo CMMI[CMMI06], projeto é “um conjunto gerido de recursos inter-relacionados, que entrega um ou mais produtos a um cliente ou usuário, com início definido e que, tipicamente opera conforme o plano”, portanto definimos que projeto é um esforço temporário para se chegar a um objetivo, objetivo este que pode ser uma implantação de um novo sistema, melhoria de um novo sistema.

O gerenciamento de projeto de desenvolvimento de software têm como foco gerir recursos humanos, recursos de hardware, recursos financeiros. Para se chegar no objetivo de um projeto é necessário ter mão de obra qualificado, e para ter mão de obra qualificada é preciso ter recursos financeiros, onde muitas vezes é complicado gerir, pois mão de obra qualificada e especializada têm um custo alto, contudo os desenvolvedor precisam ter uma infraestrutura adequada para se desenvolver.

Projeto é um esforço temporário, ou seja, têm um início, meio e fim, onde é preciso gerir o tempo, portanto é preciso gerir especificamente cada parte do projeto para se planejar todo o andamento do projeto. Segundo Wilson de Pádua Paula Filho (2009), um plano de projeto é constituído das seguintes partes:

### Gestão de Escopo

Um conjunto de funcionalidades que descrevem o trabalho que deve ser desenvolvido para se chegar no objetivo final, o escopo e desenvolvido a partir dos requisitos extraídos do cliente, isto varia de acordo com o processo de desenvolvimento de software adotado.

## **Gestão de Tempo**

Gestão de tempo em desenvolvimento de software é complexo, devido a muitas variáveis, o cliente que está investindo quer uma data para se obter o produto e se propõem a investir e esperar o produto requisitado, porém o prazo determinado pode ser afetado por inúmeros fatores, tais como a saída de um membro da equipe, mudanças de requisitos, falta de mão de obra qualificado entre outros fatores, portanto é preciso planejar detalhadamente seu projeto.

## **Gestão de Qualidade**

A qualidade de um software é medido pelo grau de conformidade do software, componentes e processos, para se medir o software precisa passar por uma auditoria de qualidade, onde o intuito é saber se é possível melhorar e como melhorar, os processos de desenvolvimento de software utilização as validações do cliente como um meio de manter a alta qualidade do produto porém setores como infraestrutura somente é possível medir com pessoas especializadas.

## **Gestão de Pessoas**

Sem pessoas, não existe projeto, portanto é necessário manter os membros envolvidos no projeto motivados, uma saída de um desenvolvedor, analista, gerente pode acarretar atrasos enormes no projeto, conseqüentemente gerando pressão por parte do cliente e podendo levar o projeto a ruínas.

## **Gestão de Riscos**

Gestão de riscos é uma parte indispensável na área de gerenciamento de projeto, pois um desenvolvimento de software têm riscos, tais como perda de um integrante da equipe á perda do banco de dados, todos os riscos de um projeto devem ser analisados e calculados seus impactos e riscos no projeto caso ocorra, para tais riscos existem quatro respostas, eliminação, mitigação (redução da probabilidade de ocorrer o risco), transferência e aceitação.

### **Gestão dos Custos**

Gerenciar custos é de extrema importância, porém altamente complexo, devido inúmeras variáveis tais como salários dos integrantes do projeto, custos de ferramentas, terceiros, horas extras. Para isto deve-se gerencia-los para não ultrapassar o orçamento proposto pelo cliente.

### **Gestão de Aquisições**

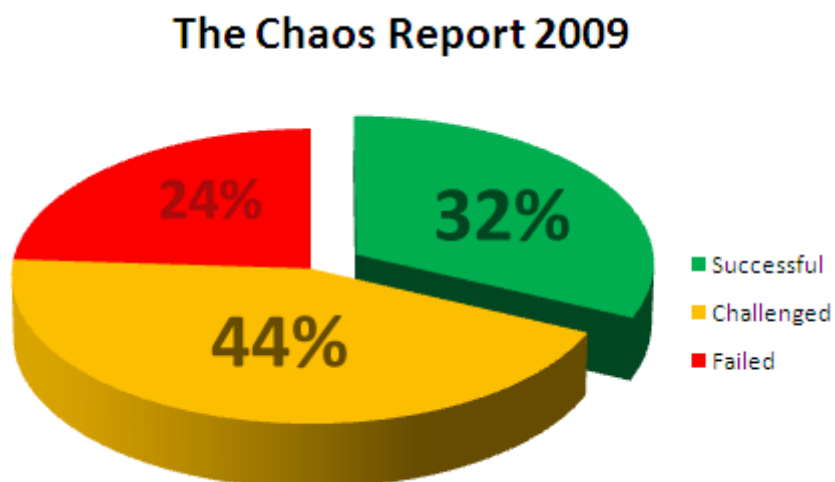
Durante o desenvolvimento de um software, as vezes é preciso qualificar os envolvidos para o desenvolvimento, contratando serviços de terceiros tais como provedores de internet, servidor, enviar funcionários para visitas ao cliente, compras de novos equipamentos, portanto é preciso gerir as aquisições dentro de um projeto para isto cria-se um processo de aquisição.

### **Gestão de Comunicação**

Para entrega de qualidade, é preciso que os envolvidos no projeto estejam em sintonia e motivados, para isto é preciso gerenciar a comunicação por inúmeros meios tais como, material de treinamento, publicações em conferencias, propostas, boletins internos tanto para funcionários quanto para o cliente, listagem de perguntas frequentes, fóruns de discussões e reuniões.

O principal desafio em gerenciar um projeto é devido a complexidade de acompanhar e gerenciar o andamento do projeto, segundo uma pesquisa realizada em 2009 pelo *Standish Group* (Figura 10), descobriu-se que apenas 32% dos projetos de TI são concluídos com sucesso, 44% são desafiadores e 24% falham.

Figura 10: Gráfico Chaos



Fonte: <http://blog.casagrande.la/frederic/2009/06/the-chaos-report-2009.html>

## 2 LEVANTAMENTO DE REQUISITOS

O desenvolvimento de uma aplicação desktop, específica para gerenciamento de sistemas, utilizando a metodologia Scrum, requer os requisitos funcionais e não funcionais que devem ser implementados no software para que possa chegar no objetivo deste trabalho. Segundo Rogers S. Pressman (2007), o levantamento de requisito é constituídos pelas seguintes partes: Requisitos funcionais e requisitos não funcionais.

### 2.1 Requisitos Funcionais

Requisito funcionais são as funcionalidades que o software deverá ter:

- O usuário deve ser capaz de manipular dados, inserir, alterar, exibir e deletas.
- O usuário deve ser capaz de recuperar sua senha de acesso.
- O sistema deve oferecer telas apropriadas para o usuário compreender o que se pede.
- O sistema deve ser uma autenticação por meio de login.
- O sistema deve ser consistente em seus dados.
- O sistema deve ter Burndown
- O sistema deve ter Kanban
- O sistema deve ter controle de prazos por Spring
- O sistema deve ter cadastro de funcionários e controle dos mesmos.
- O sistema deve ter cadastro de *Product Backlog* e controle dos mesmos.
- O sistema deve ter cadastro de Tarefas Técnicas e controle dos mesmos.
- O sistema deve disponibilizar relatórios para acompanhamentos.
- O sistema deve ter calendário para acompanhamento das *Sprints*.

### 2.2 Requisitos Não Funcionais

Os requisitos não funcionais incluem atributos de qualidade para o software.

#### 2.2.1 Confiabilidade

O sistema terá medidas quantitativas de confiabilidade do software, como otimização na memória por meio de técnicas de programação, caso ocorra erro

durante a aplicação o software exibirá mensagens claras e antes da exibição tentará recuperar as falhas.

### *2.2.2 Portabilidade*

O sistema será portátil, ou seja, é fácil de mudar de sistema ou de plataforma, pois foi desenvolvido utilizando a linguagem de programação Java, o qual tem o conceito de “*WRITE ONCE, RUN ANYWHERE*”, ou seja, escreva uma vez, rode em qualquer lugar (qualquer plataforma).

### *2.2.3 Segurança*

O acesso do sistema passa por uma etapa de segurança, só é possível conectar ao sistema quem possuir um cadastro, e efetuar o login inserindo o e-mail e a senha, caso esqueça um dos itens de segurança para acessar o sistema, ele possibilitará a recuperação dos mesmo.

### *2.2.4 Desempenho*

Ao fazer qualquer operação do tipo cadastras/salvar, alterar/modifica e excluir/deletar o sistema automaticamente o faz em questão entre 0 – 3 segundos.

### *2.2.5 Usabilidade*

O usuário novo deverá ser capaz de fazer as operações no qual seu acesso permite, cadastrar, exibir, alterar e excluir, após não mais de trinta minutos de orientação.

### *2.2.6 Tipo de Interface*

O software deve ser exclusivamente para desktop, com fontes no tamanho mínimo doze, deve ser legível e desenvolvido somente na versão em português as cores devem ser claras, os botões com sua função escrita e com ícones que usuários possam compreender.

### *2.2.7 Documentação Necessária*

O software terá a documentação para compreensão dos fluxos e telas do software integrado ao mesmo.

### 2.2.8 *Requisito de Instalação*

O software será distribuído para instalação por um instalador disponibilizado para download.

### **3 FERRAMENTAS PARA DOCUMENTAÇÃO DE PROJETO**

É preciso documentar um software pois assim elimina o caso de dependência de desenvolvedor, e também facilita a análise de um problema ou a projeção de uma melhoria, somente olhando para a documentação do software, para isto existem ferramentas que auxiliam este trabalho e praticas globais para compreensão de todos os estudiosos.

Para documentação deste trabalho utilizarei a ferramenta UML, onde vou utilizar os diagramas mais importantes, diagrama de caso de uso, classe, sequencia e de atividades, para modelagem do banco de dados, utilizarem o diagrama de entidade e relacionamento mais conhecida como DER, porém não faz parte da UML.

#### **3.1 UML**

Atualmente a ferramenta mais utilizada para documentar projetos é a linguagem de modelagem unificada que mundialmente denominado por UML, que provêm um conjunto de diagramas para compreensão do sistema.

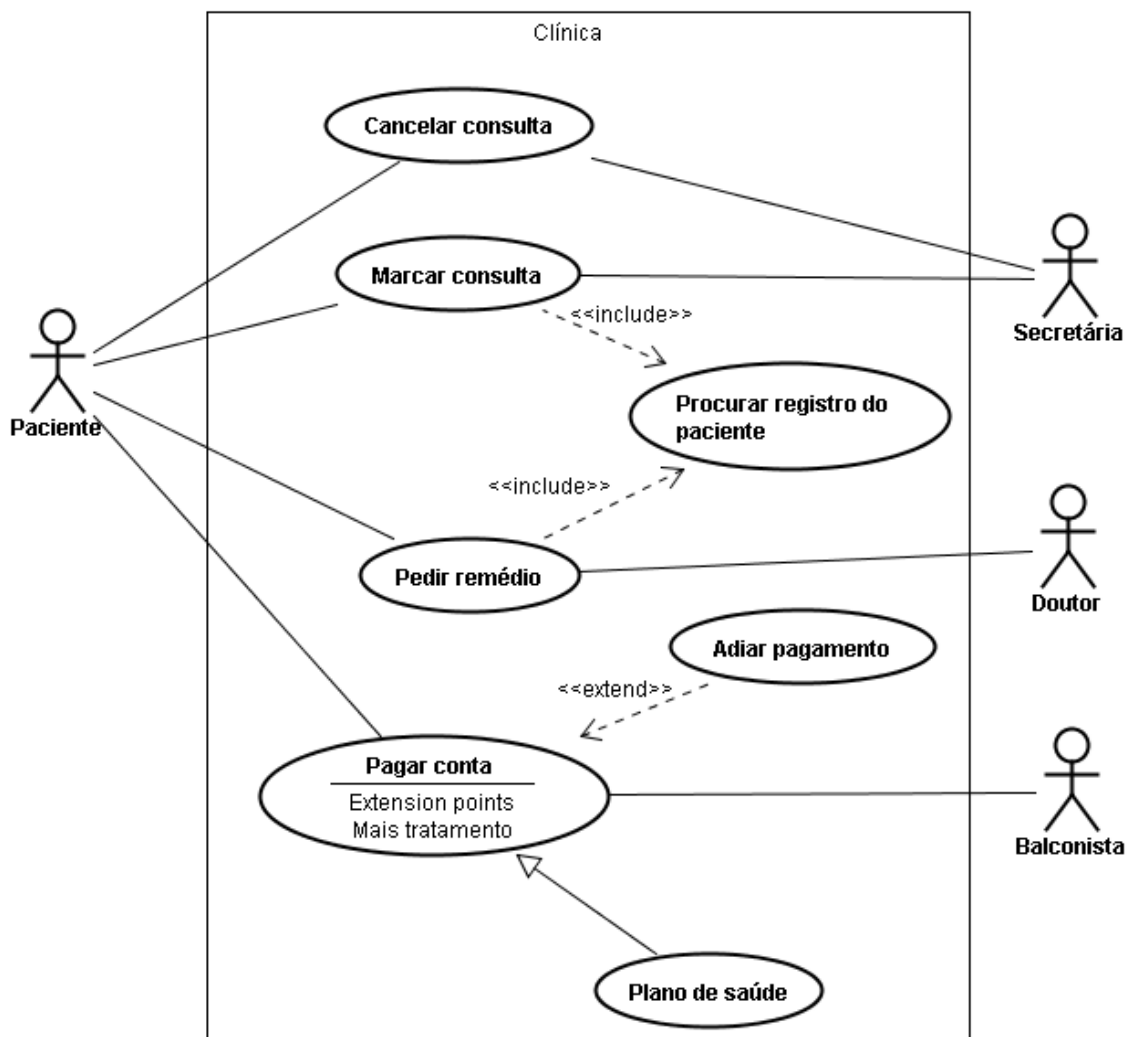
Segundo Gilleanes T.A. Guedes (2009), “Cada diagrama da UML analisa o sistema, ou parte dele, sob uma determinada óptica”, porém neste trabalho abordaremos os diagramas abaixo:

#### **CASO DE USO**

O diagrama de caso de uso têm como objetivo levantar os requisitos do sistema. É um diagrama de fácil compreensão, onde se descreve os atores do sistema e as funcionalidades do sistema, onde são destinadas para os atores de acordo com as especificações do cliente, veja um exemplo deste diagrama na figura 11.



Figura 11: Exemplo de Caso de Uso

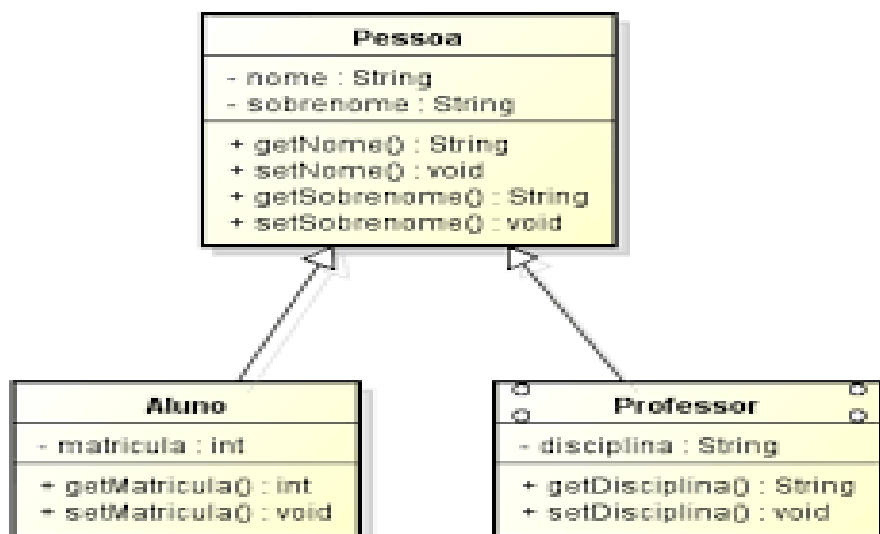


Fonte: <http://www.purainfo.com.br/artigos/o-que-e-uml/>

## CLASSE

Diagrama de classe provêm as classes do sistemas, separando as classes em atributos e métodos, e descreve o relacionamento com as demais classes do sistema. O diagrama de classe é uns dos principais diagramas, pois serve de base para os demais diagramas, veja um exemplo deste diagrama na figura 11.

Figura 12: Exemplo Diagrama de Classe

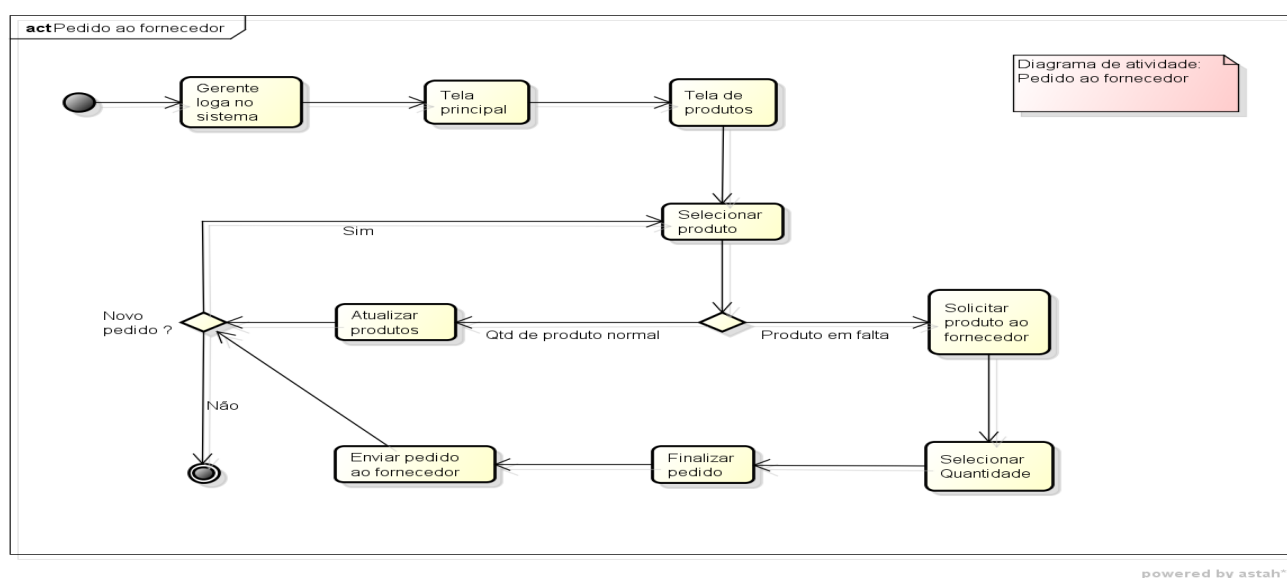


Fonte: [http://lodesenvolvedor.blogspot.com.br/2012\\_06\\_01\\_archive.html](http://lodesenvolvedor.blogspot.com.br/2012_06_01_archive.html)

## ATIVIDADE

O Diagrama de Atividade têm como objetivo descrever o processo de uma atividade no sistema tais como um acesso ao sistema, onde a primeira etapa é inserir dados (usuário, senha), validar usuário, validar senha, validar acesso etc. A figura 12, representa um exemplo do diagrama de atividade.

Figura 13: Exemplo Diagrama de Atividade

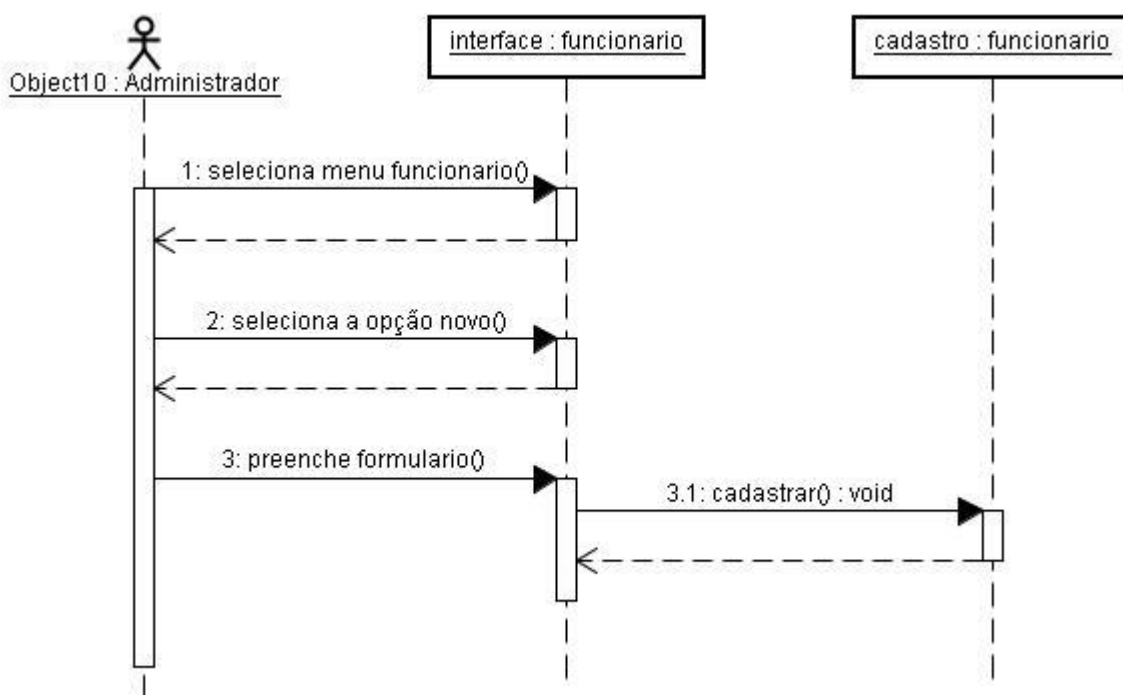


Fonte: <http://javafree.uol.com.br/topic-890052-Duvida-em-Diagrama-de-Atividades-UML-2.html>

## SEQUENCIA

Diagrama de sequencia têm como objetivo descrever um processo do sistema, detalhadamente envolvendo todas as classes do sistema e respectivamente seus métodos invocados para a realização do processo e o comportamento do mesmo, a figura 13 representa um exemplo de diagrama de sequência.

**Figura 14: Exemplo Diagrama de Sequência**

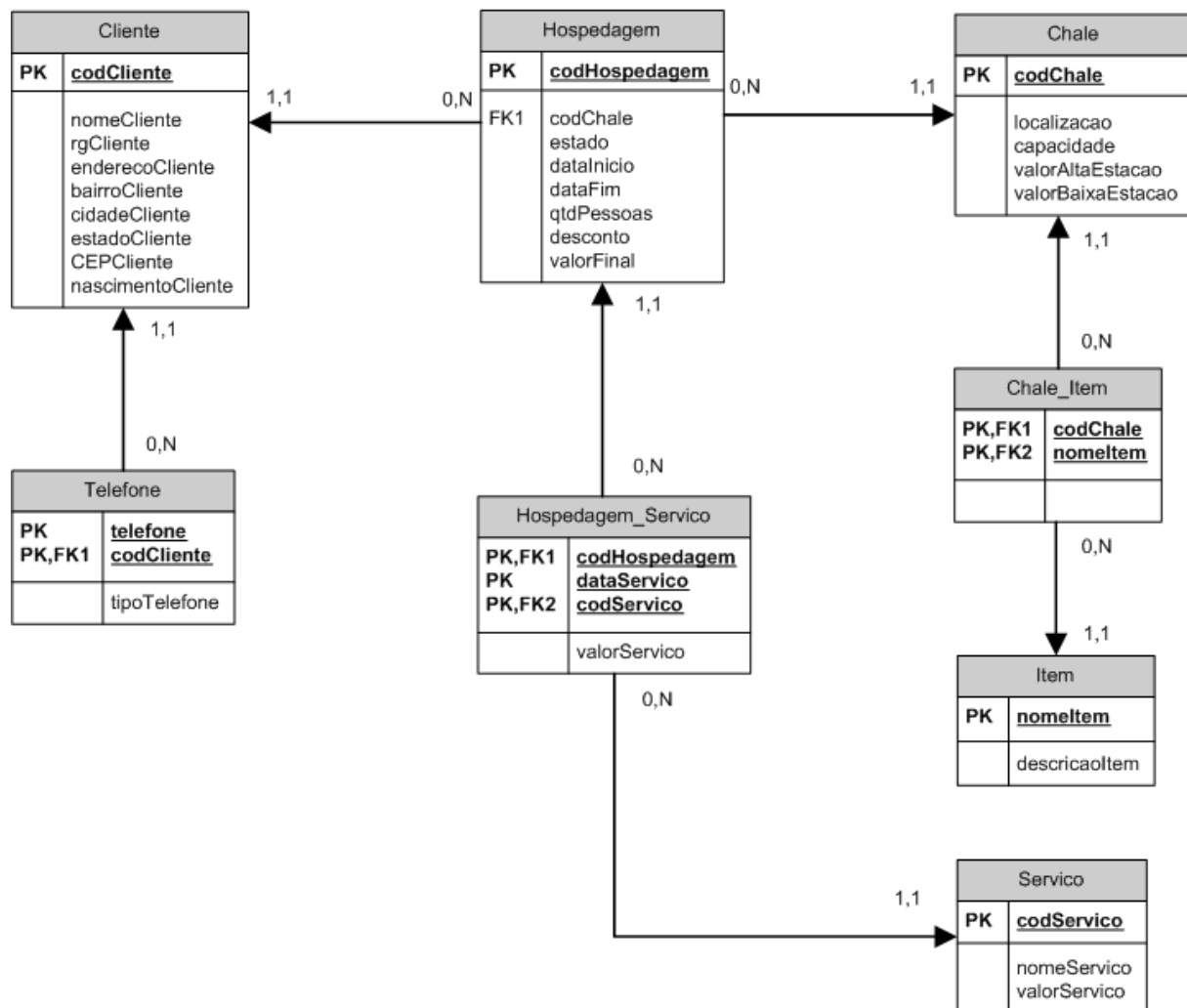


Fonte: <http://br.monografias.com/trabalhos3/nutrifree-solucao-apoio-nutricao-clinica/nutrifree-solucao-apoio-nutricao-clinica2.shtml>

### 3.2 Diagrama Entidade Relacionamento

Diagrama entidade relacionamento é utilizado pelos engenheiros de software para descrever as entidades envolvidas no sistemas, seus atributos e seus relacionamentos, com base no diagrama entidade relacionamento pode-se ter noção da estrutura do banco de dados que o sistema terá. Veja a Figura 14 que representa um exemplo de um diagrama entidade relacionamento.

Figura 15: Exemplo Diagrama Entidade Relacionamento



Fonte: <https://cursobsi.wordpress.com/2008/04/28/diagrama-entidade-relacionamento/>

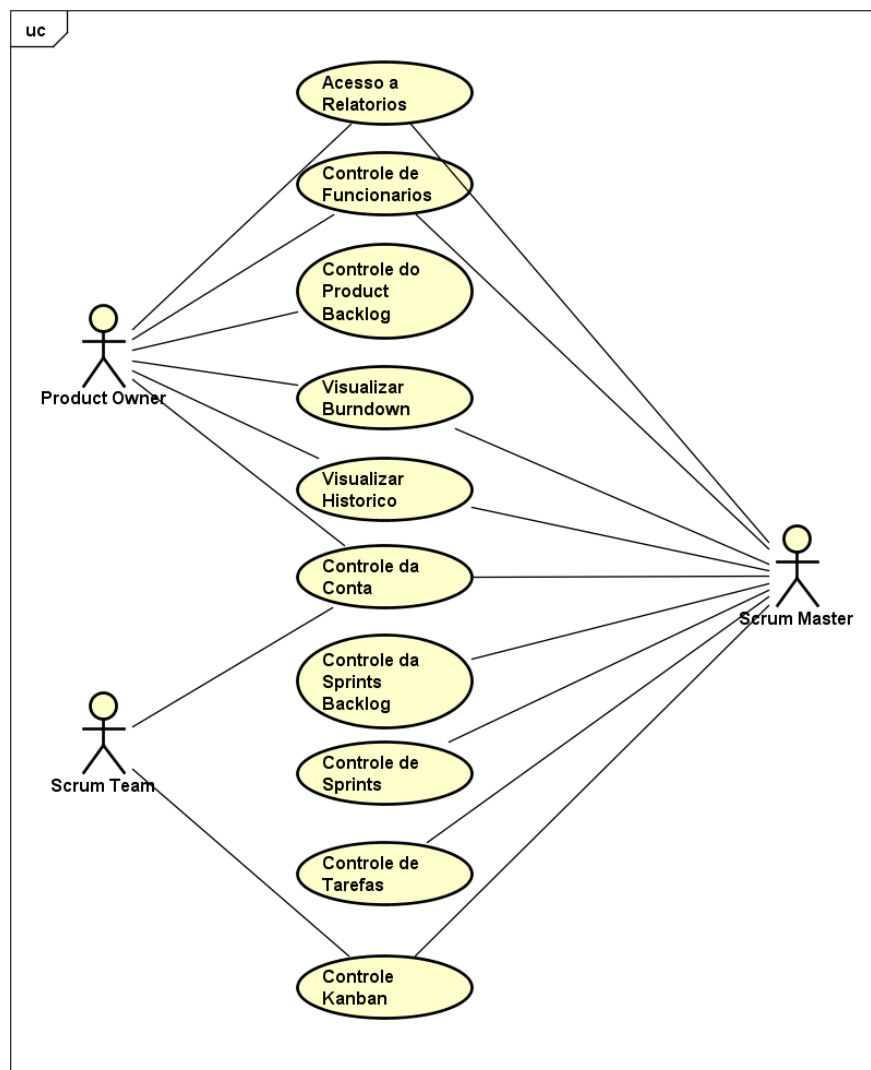
## 4 ESTUDO DE CASO

Neste estudo de caso iremos desenvolver uma aplicação para gerenciar o ciclo de vida de um projeto, sobre o método Scrum. Essa aplicação tem como objetivo gerenciar uma *Sprint*, onde é possível cadastrar *product backlog*, tarefas técnicas, *sprint backlog*. Para gerenciamento de uma Sprint este trabalho provê gráfico de acompanhamento, históricos diários de ações dos usuários com base no kanban, tais como inicialização de uma tarefa, finalização e também provê um relatório.

A seguir iremos utilizar a UML para poder documentar e representas o sistema que será desenvolvido. Para isso iremos utilizar os diagramas de caso de uso, classe, sequencia e de atividade.

Para modelagem do banco de dados iremos utilizar o diagrama entidade e relacionamento (DER).

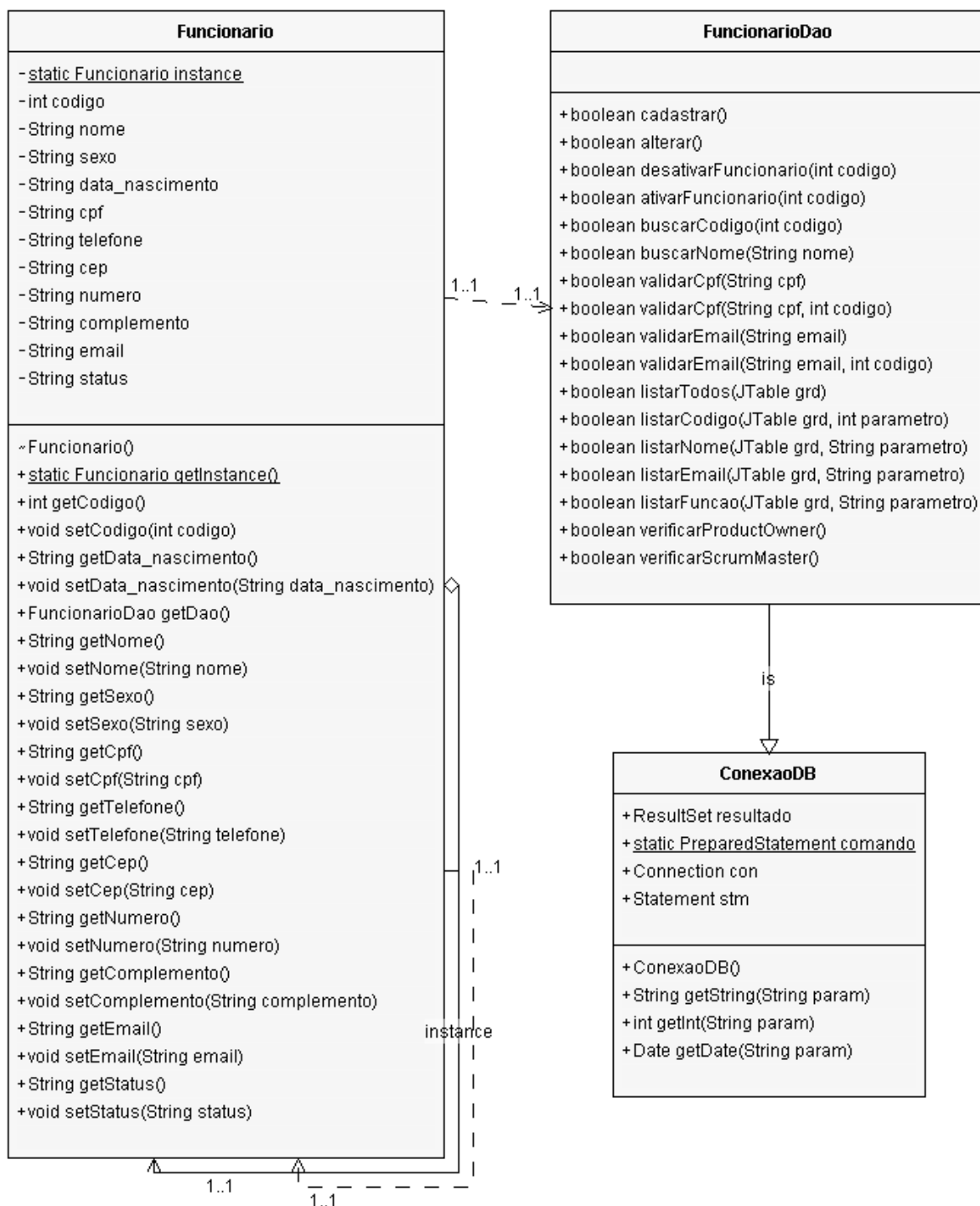
## 4.1 Caso de Uso



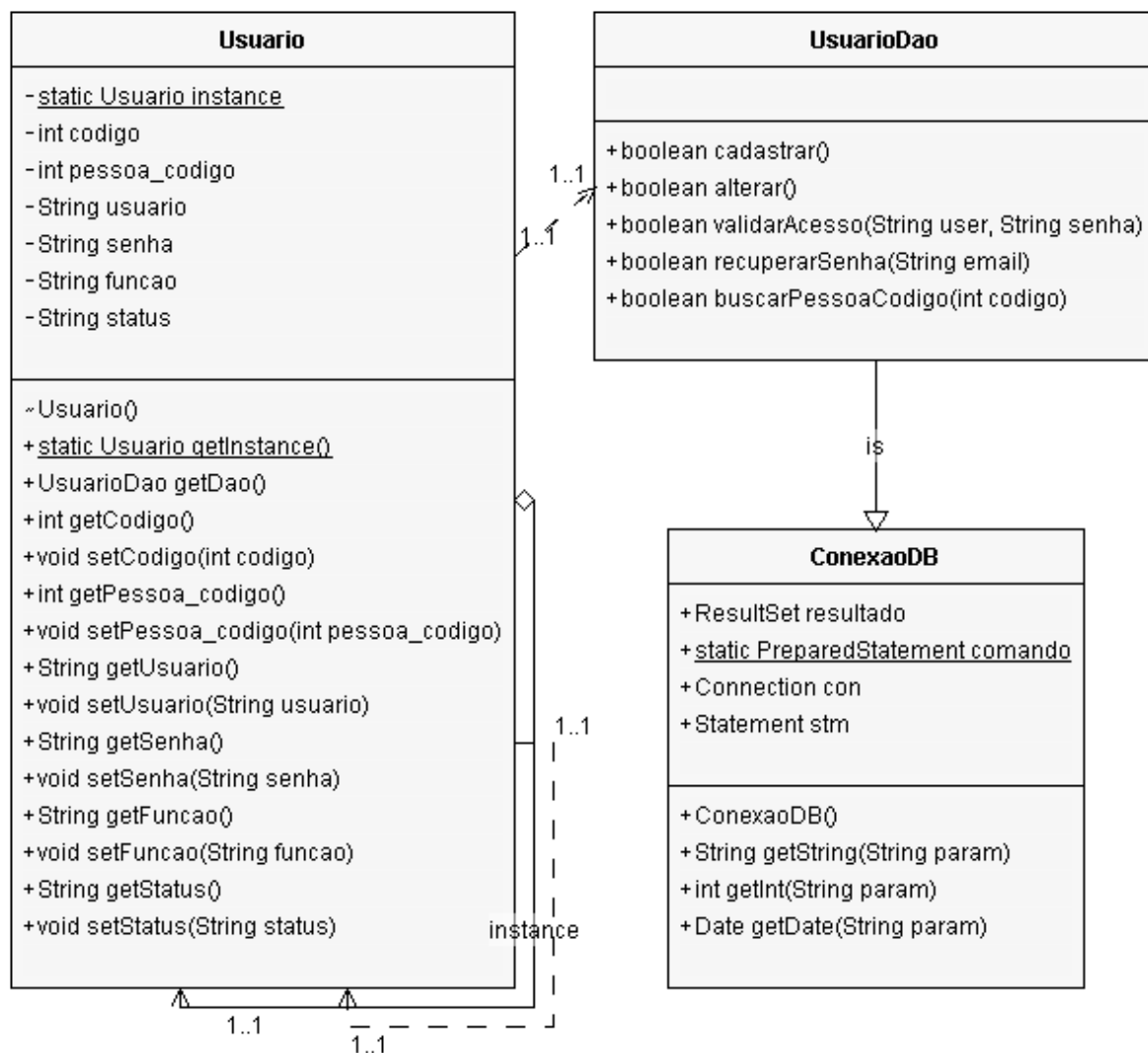
## 4.2 Classe

A seguir iremos representar o diagramas de classe das principais classes que compõem o projeto.

### 4.2.1 Funcionário

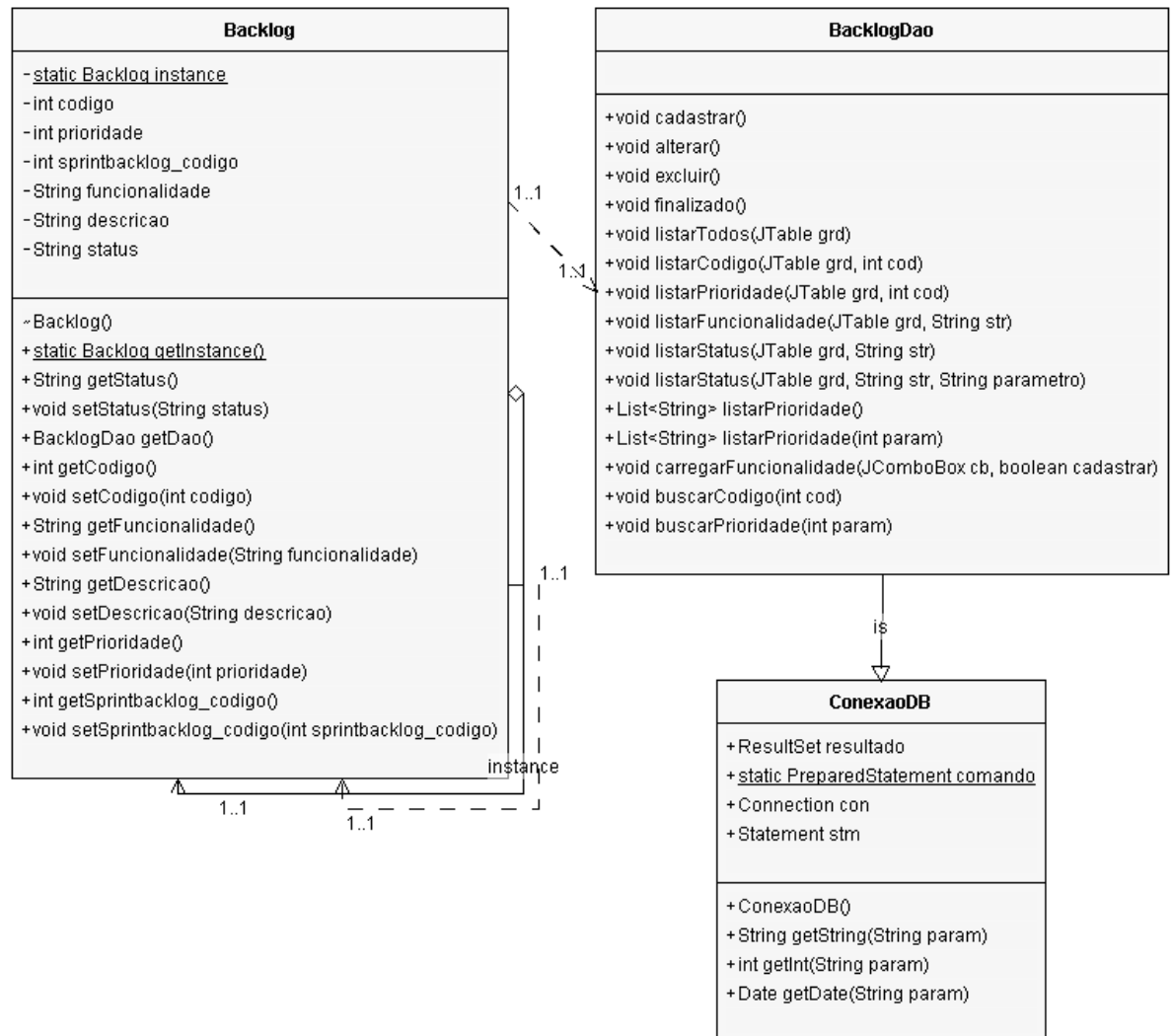


## 4.2.2 Usuário

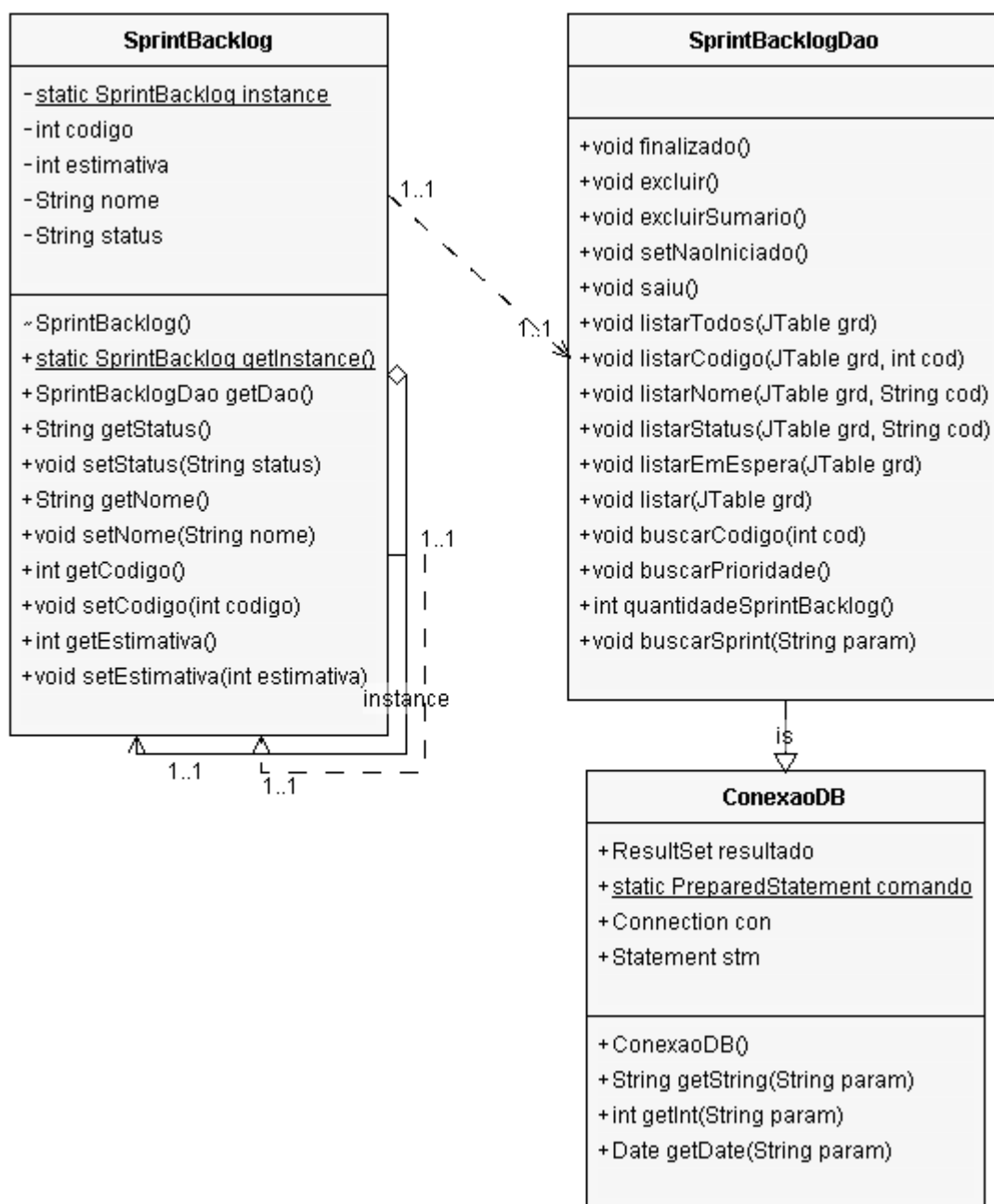




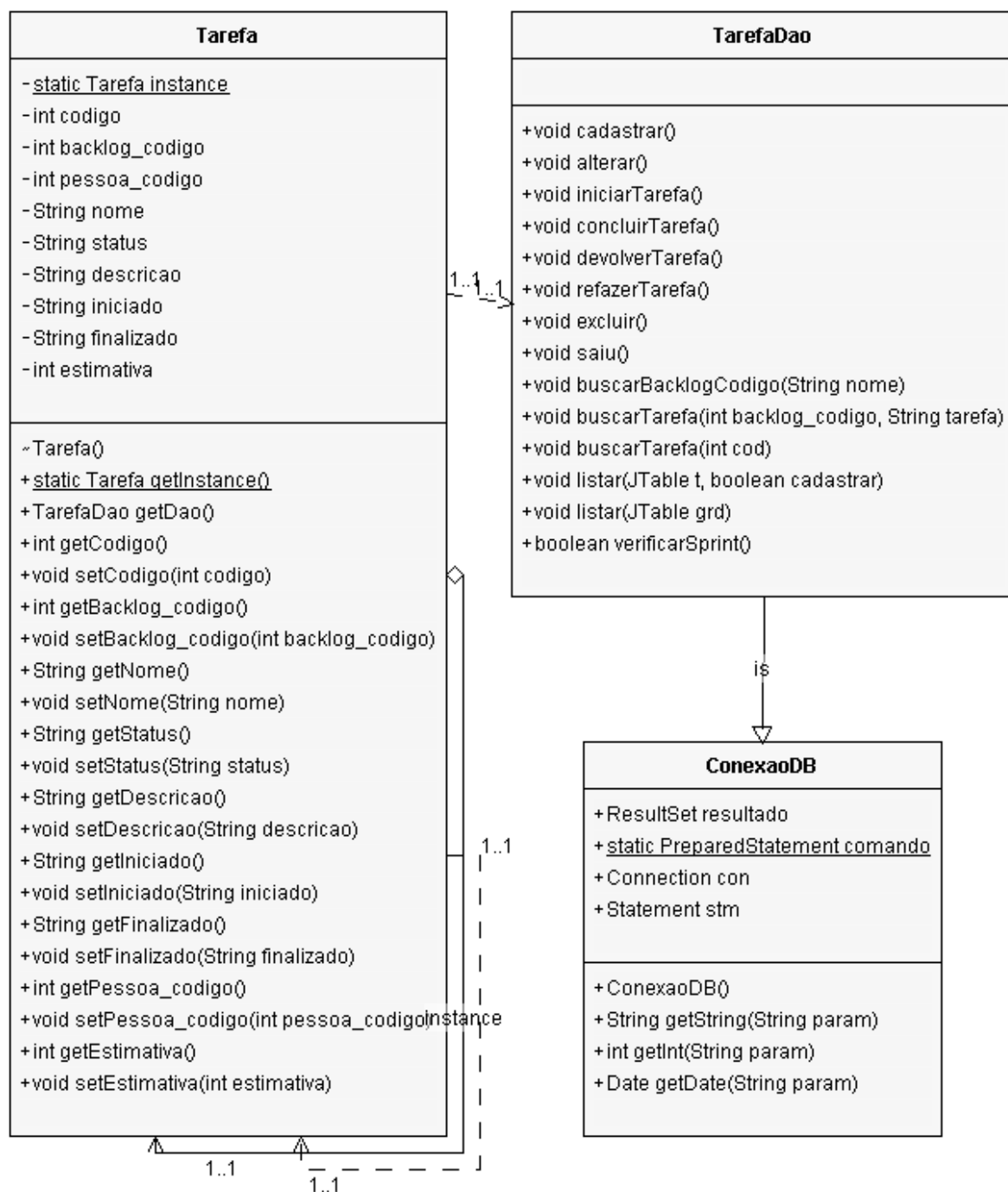
### 4.2.3 Product Backlog



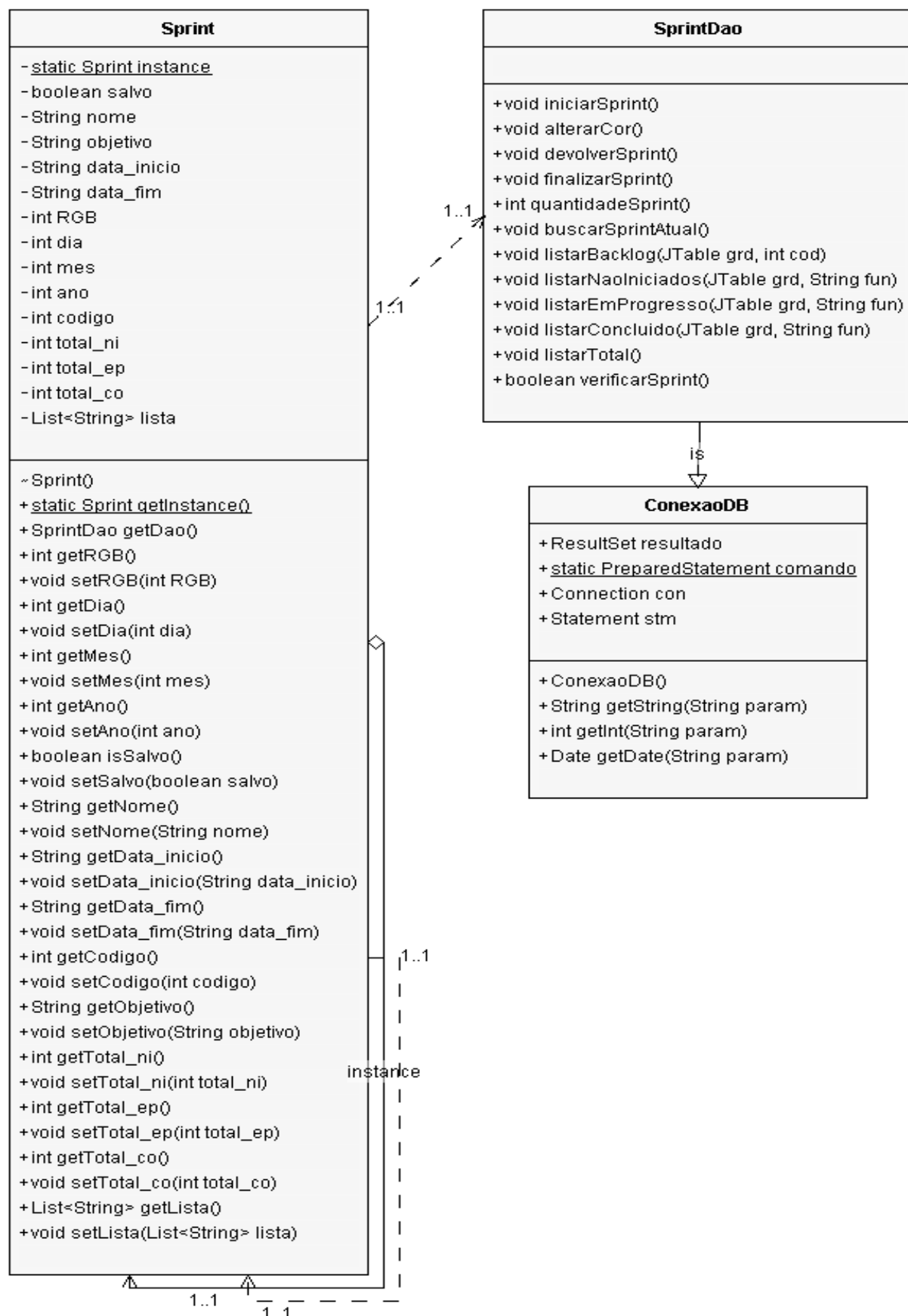
## 4.2.4 Sprint Backlog



## 4.2.5 Tarefa



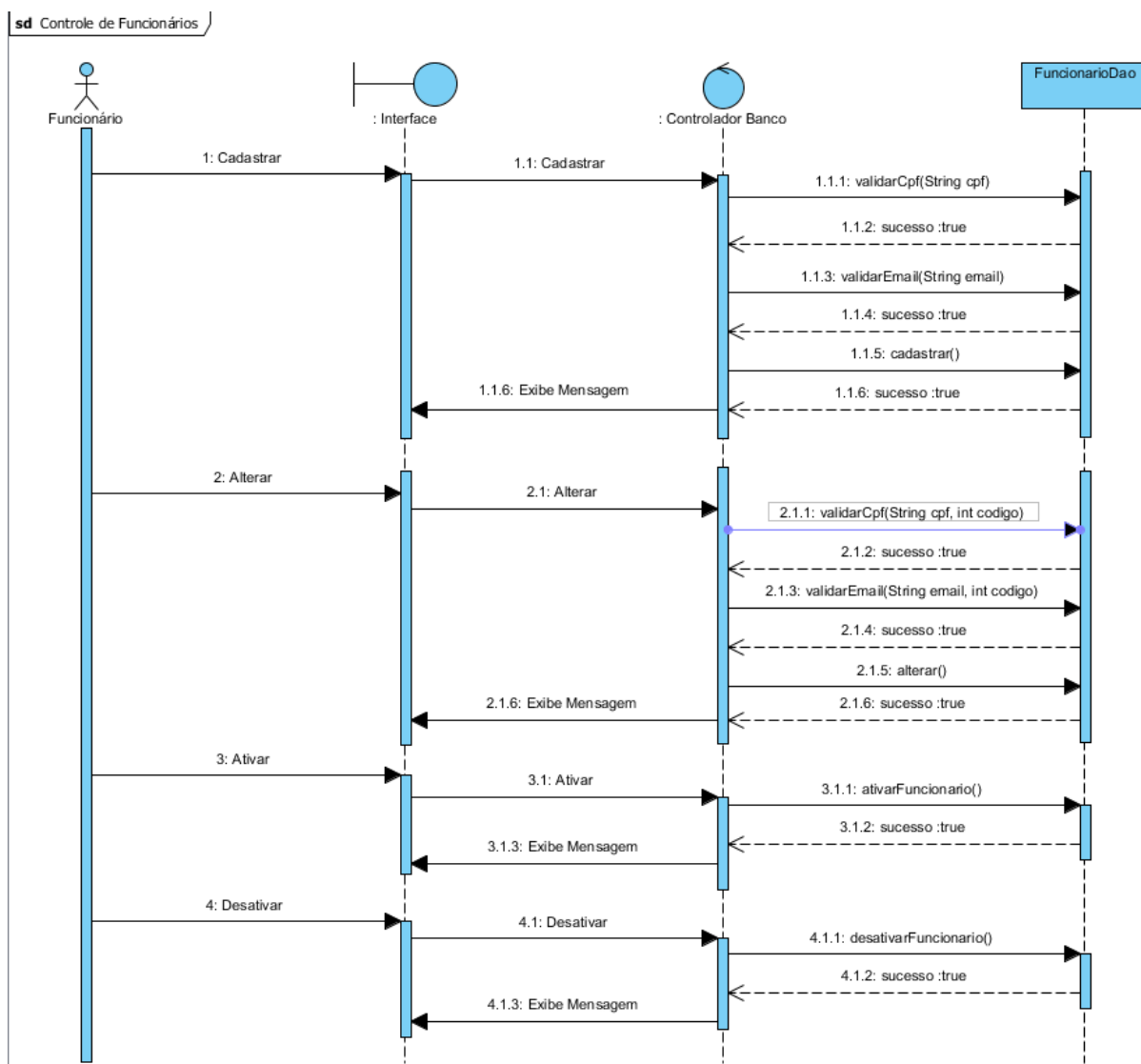
## 4.2.6 Sprint



### 4.3 Diagrama de Sequencia

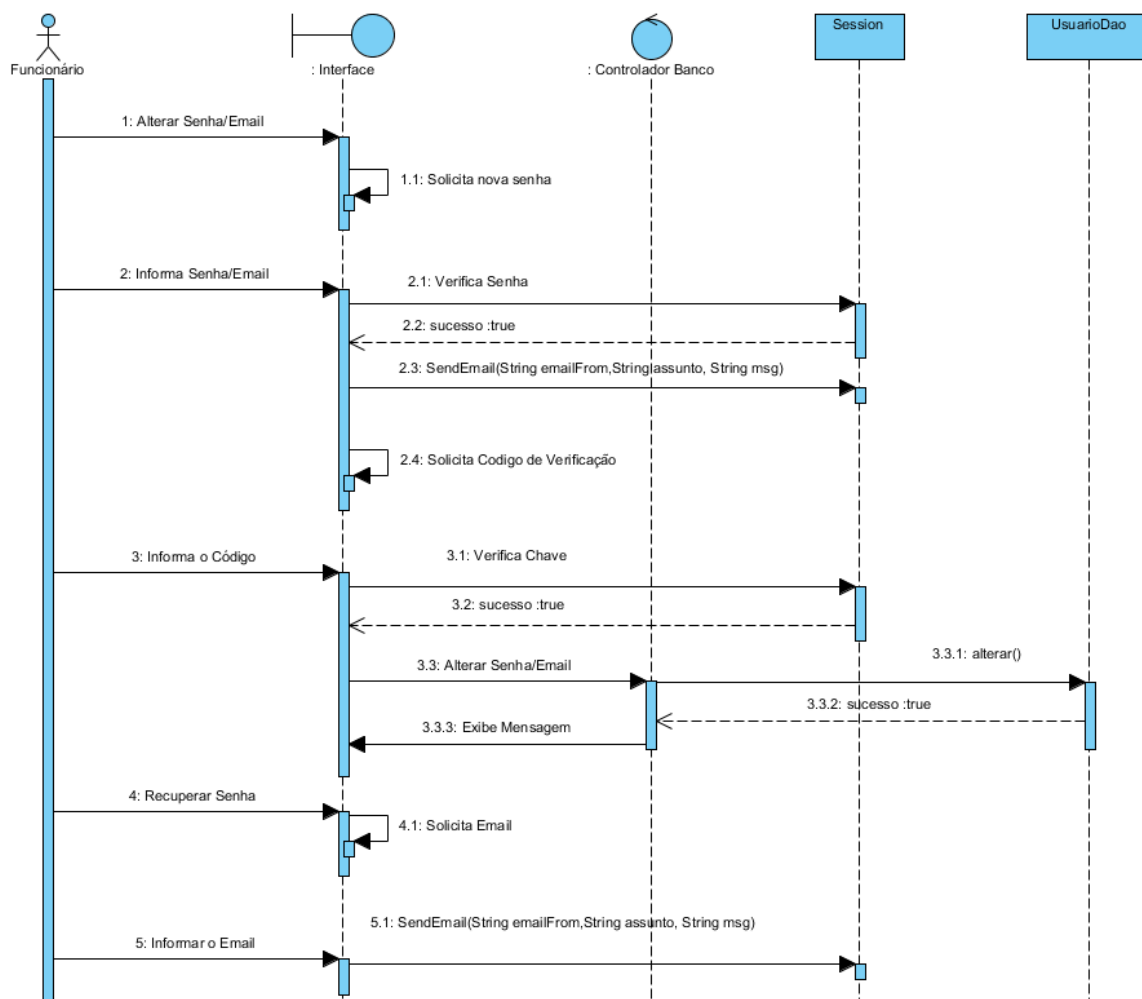
A seguir iremos representar o diagramas de sequencia referente a cada quadro do estudo de caso.

#### 4.3.1 Controle de Funcionário

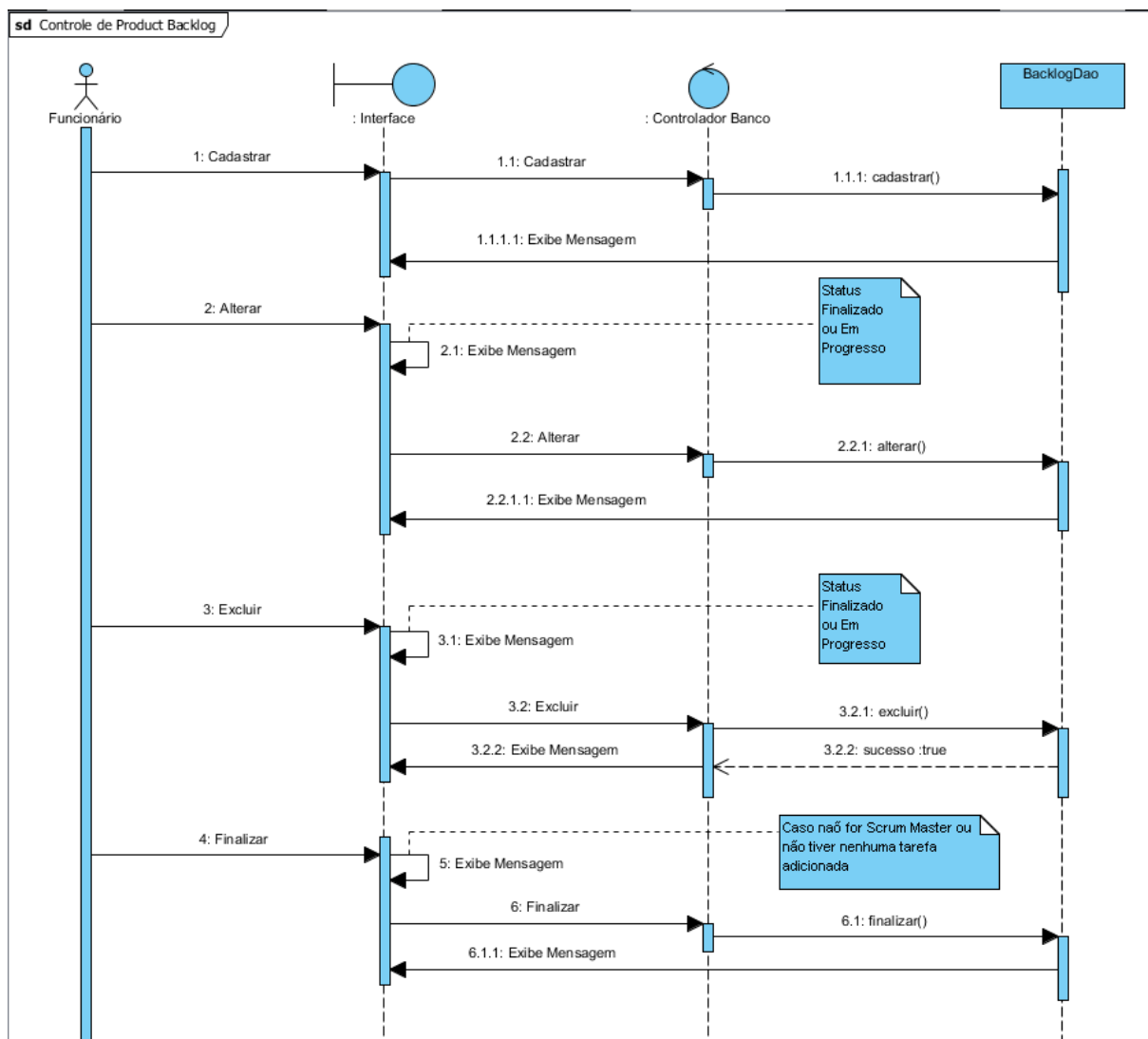


### 4.3.2 Controle de Conta

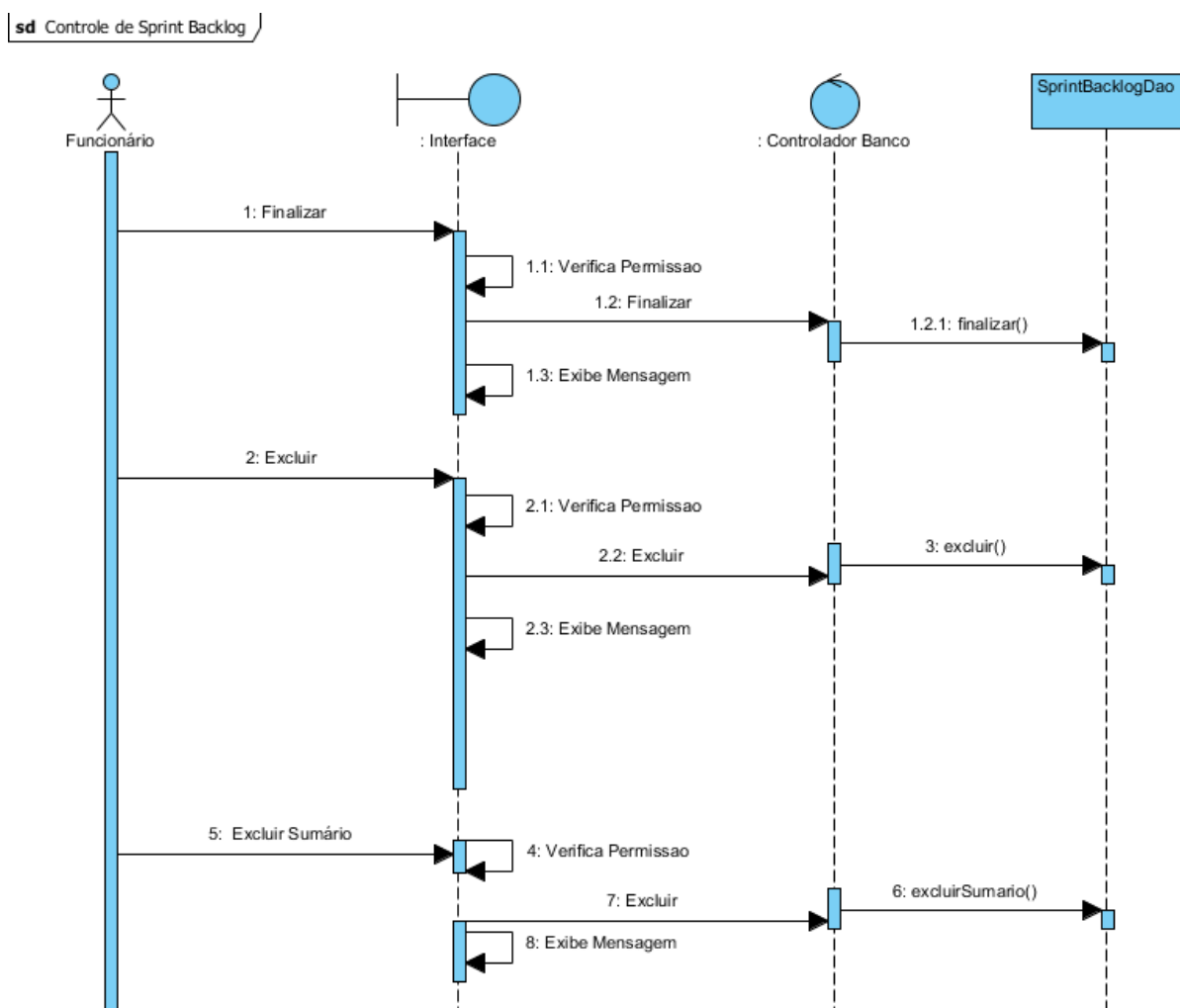
sd Controle de Conta



### 4.3.3 Controle de Product Backlog

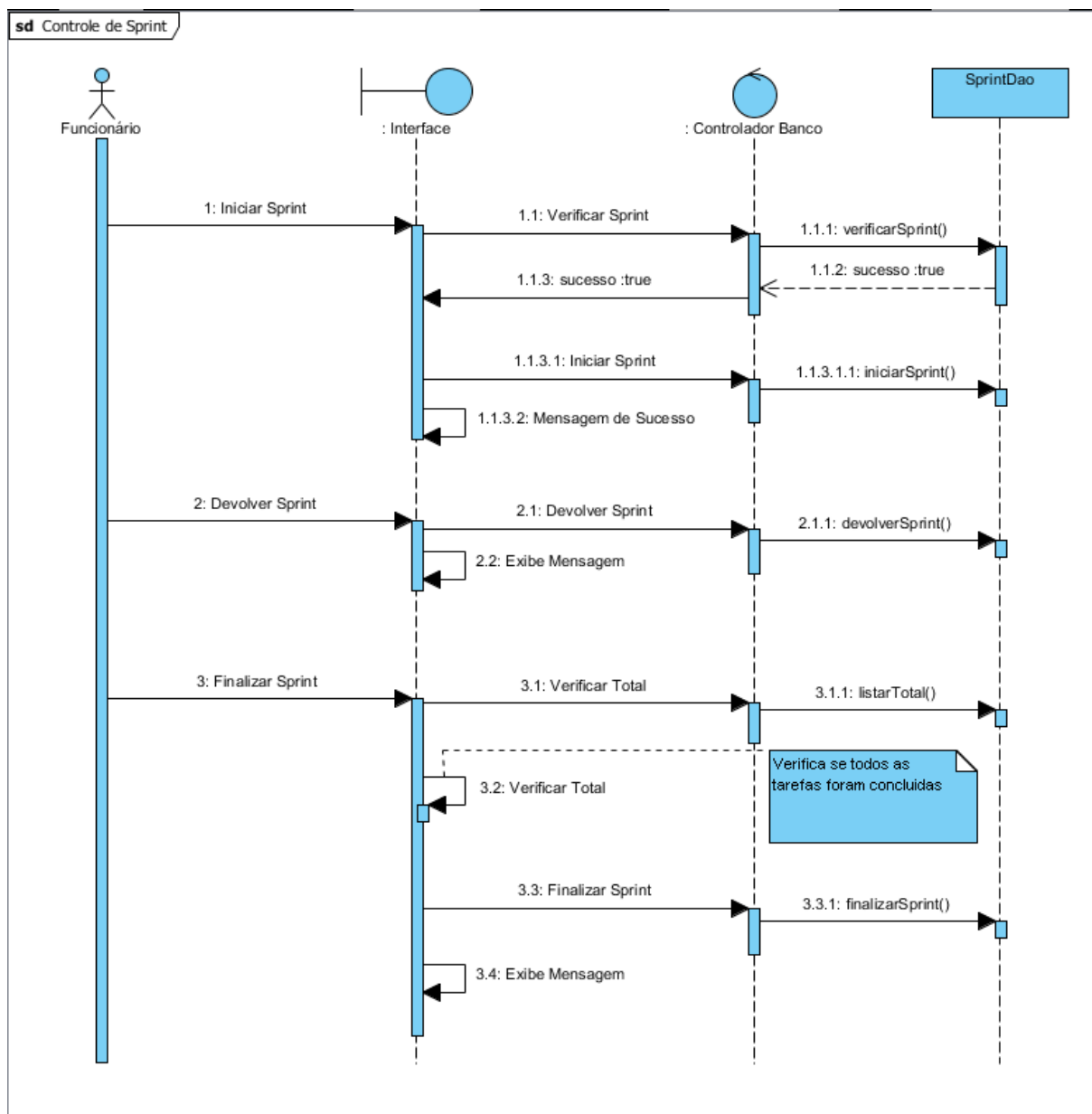


#### 4.3.4 Controle de Sprint Backlog

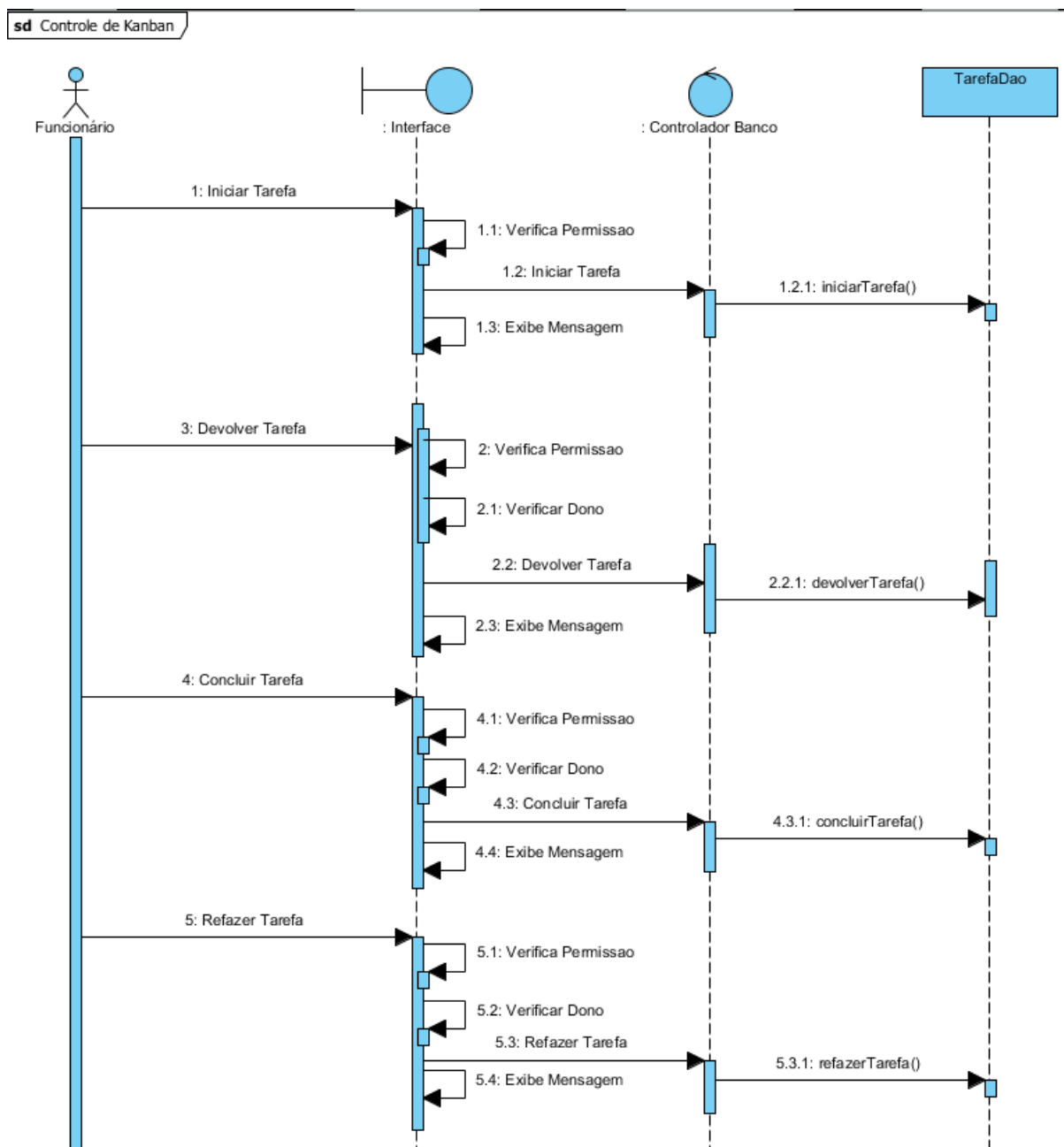




## 4.3.5 Controle de Sprint

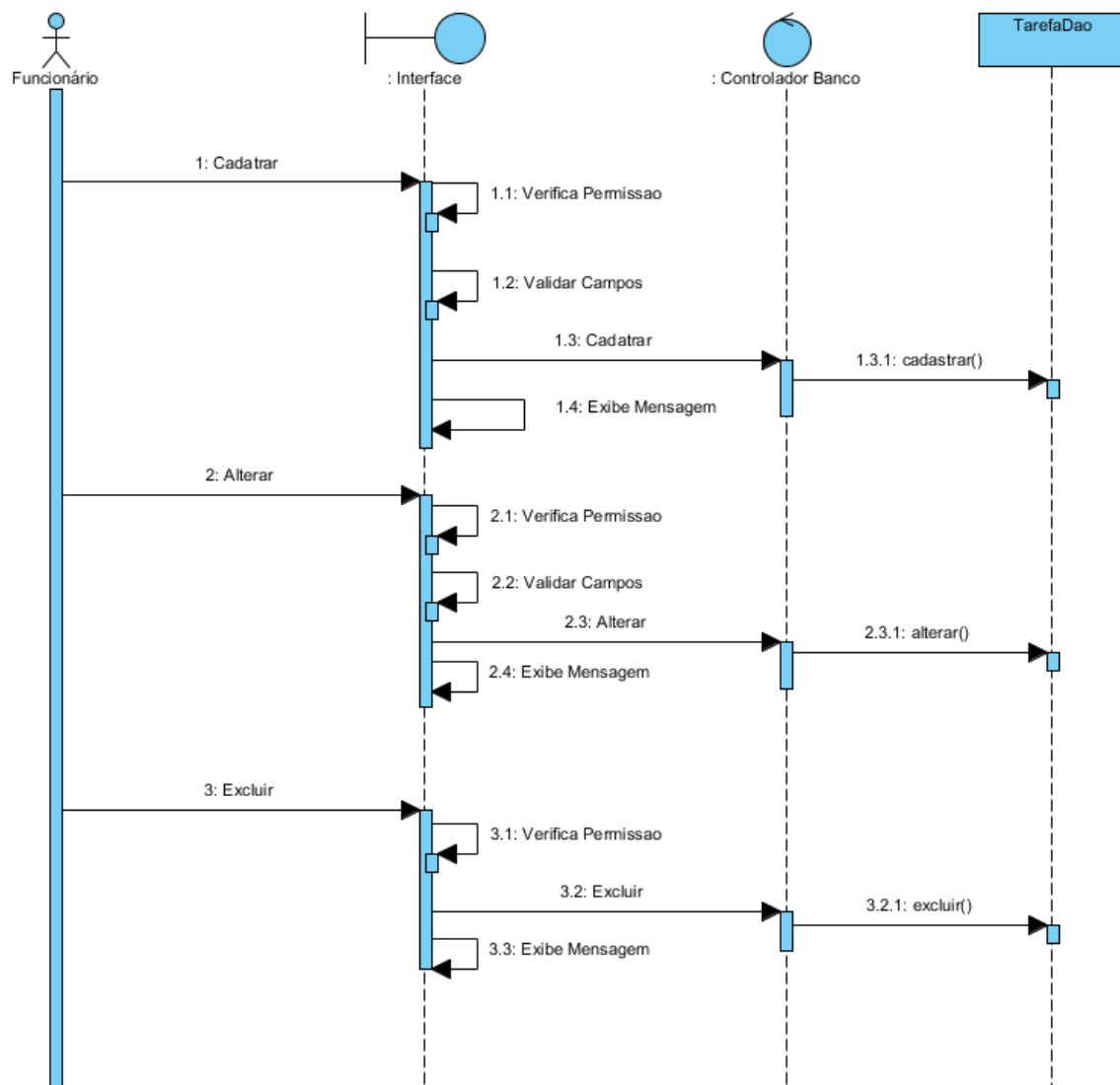


## 4.3.6 Controle de Kanban

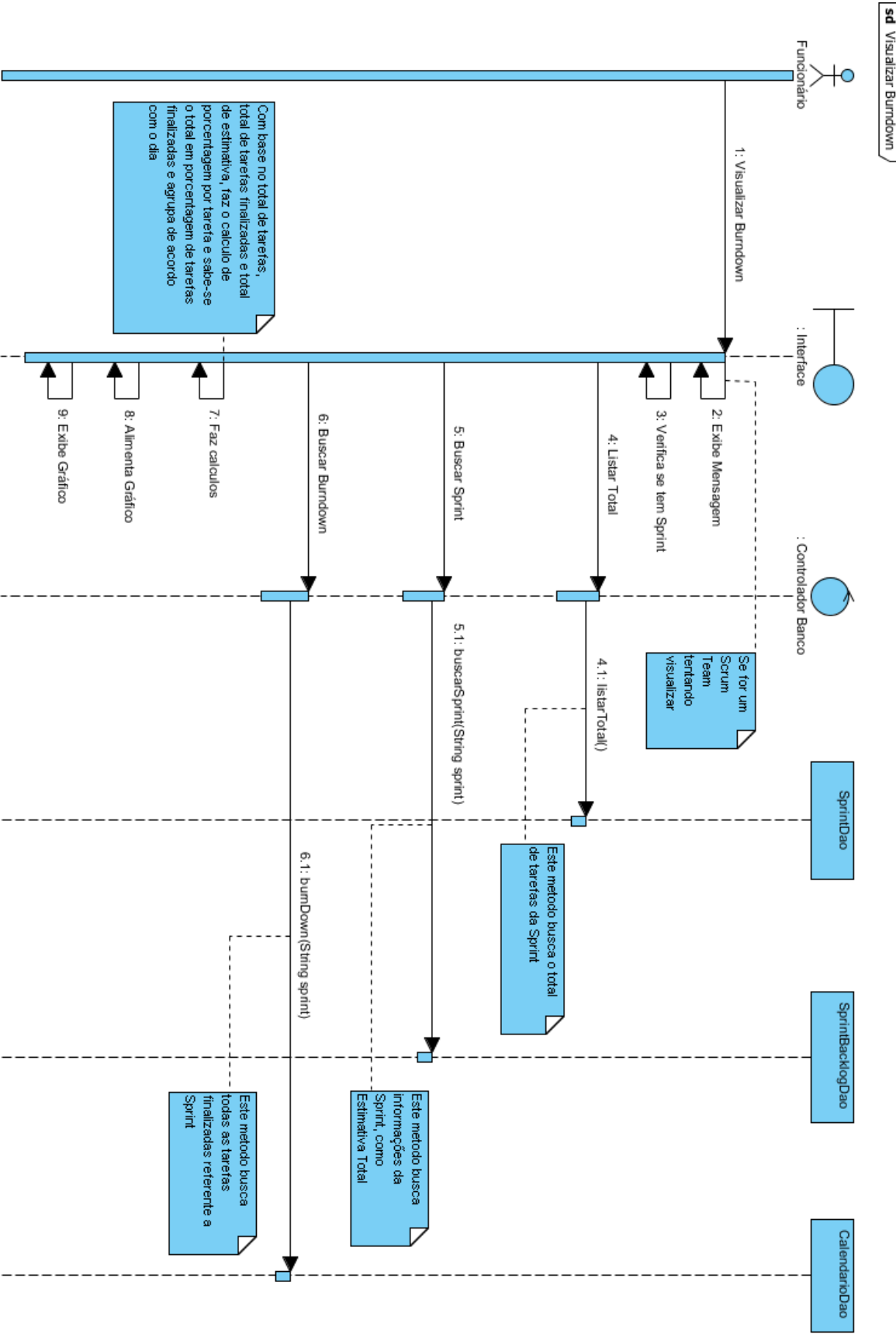


## 4.3.7 Controle de Tarefa

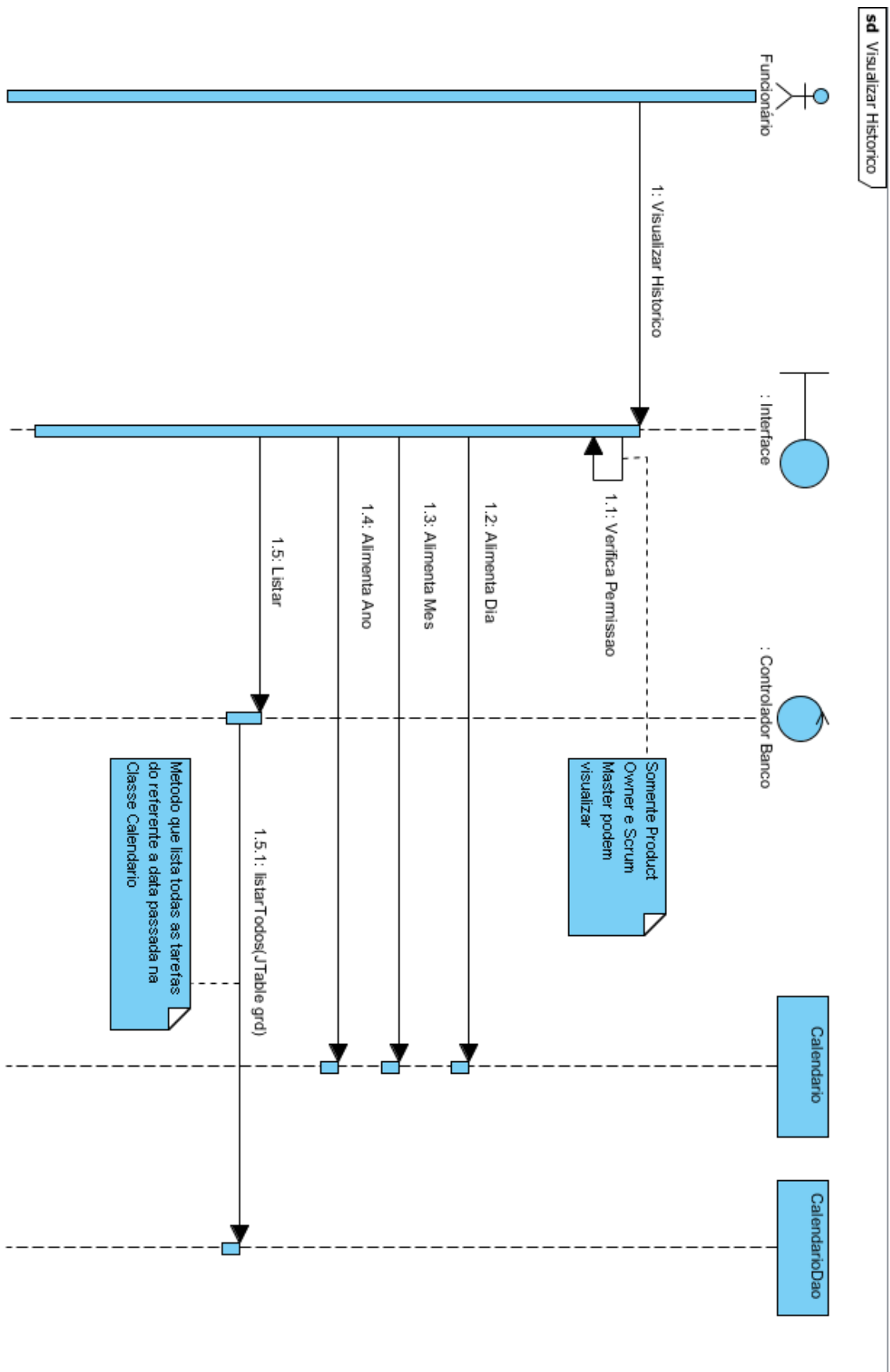
sd Controle de Tarefa



4.3.8 Visualizar Burndown



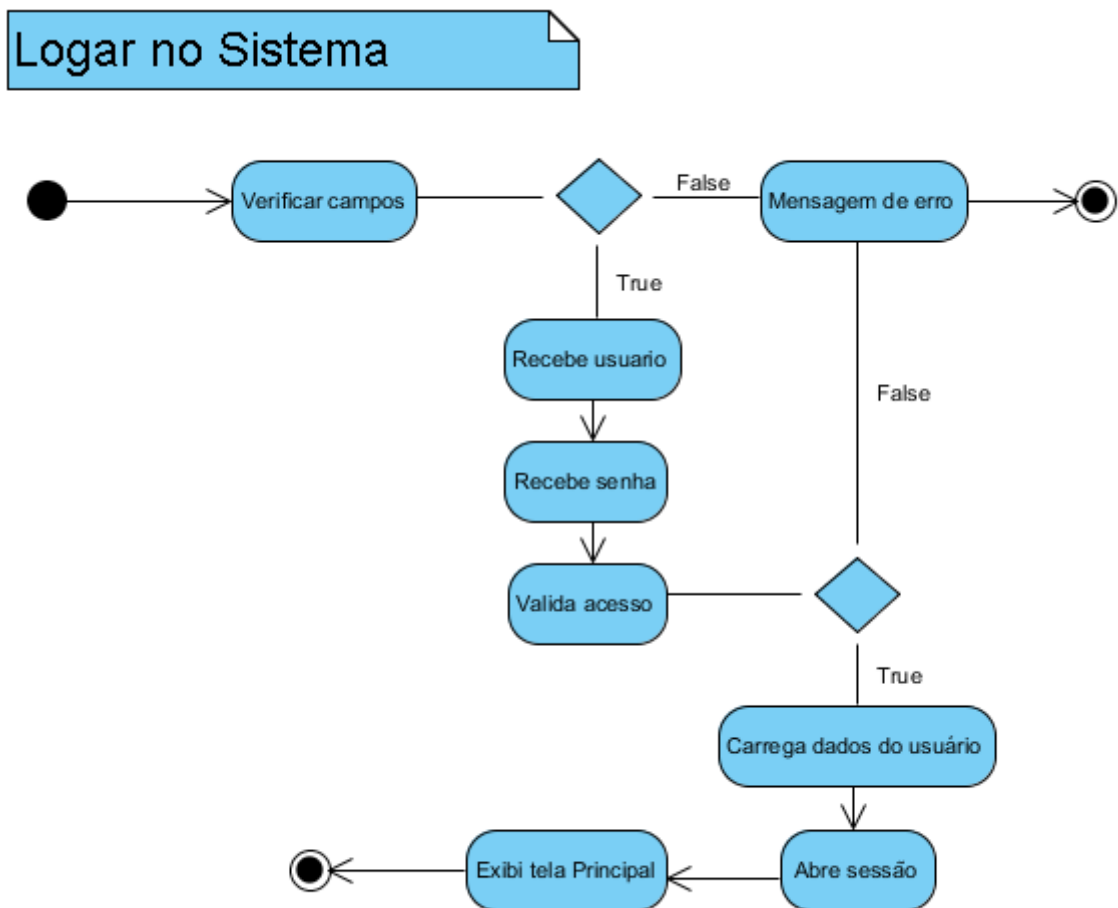
## 4.3.9 Visualizar Histórico



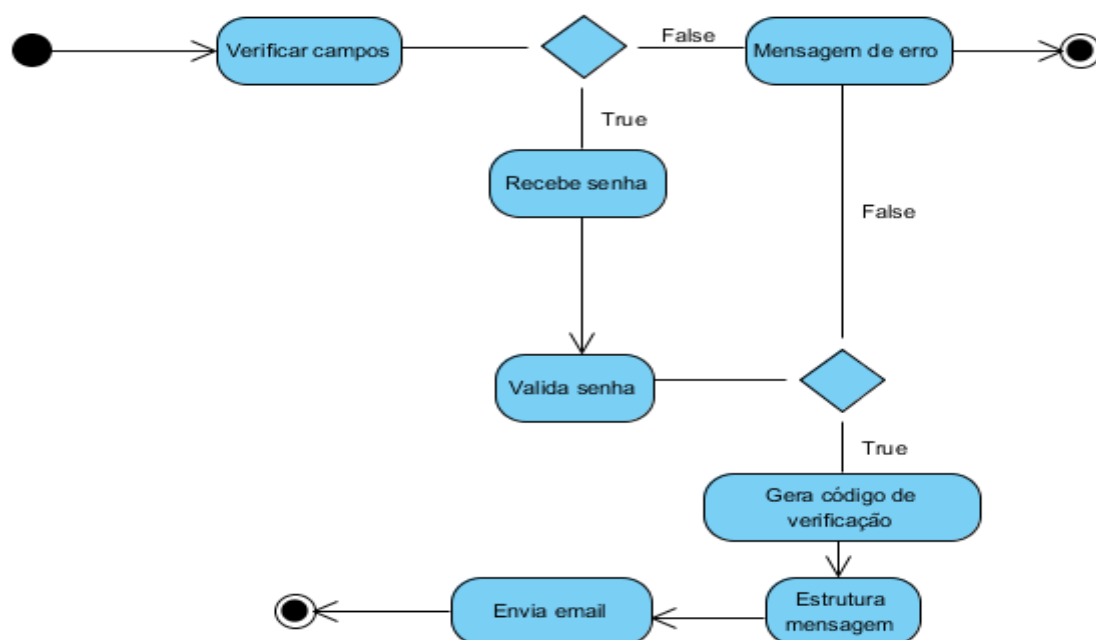
#### 4.4 Diagrama de Atividade

A seguir iremos representar o diagramas de atividade referente a cada ação que o usuário pode efetuar dentro do sistema.

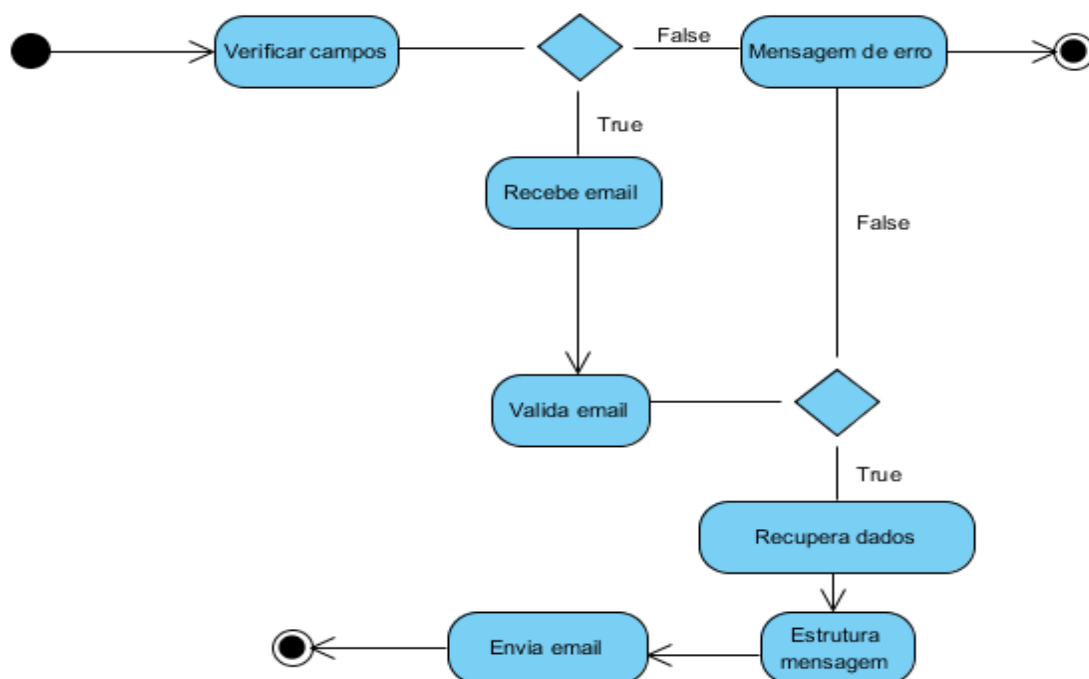
##### 4.4.1 Controle de Conta



## Alterar senha

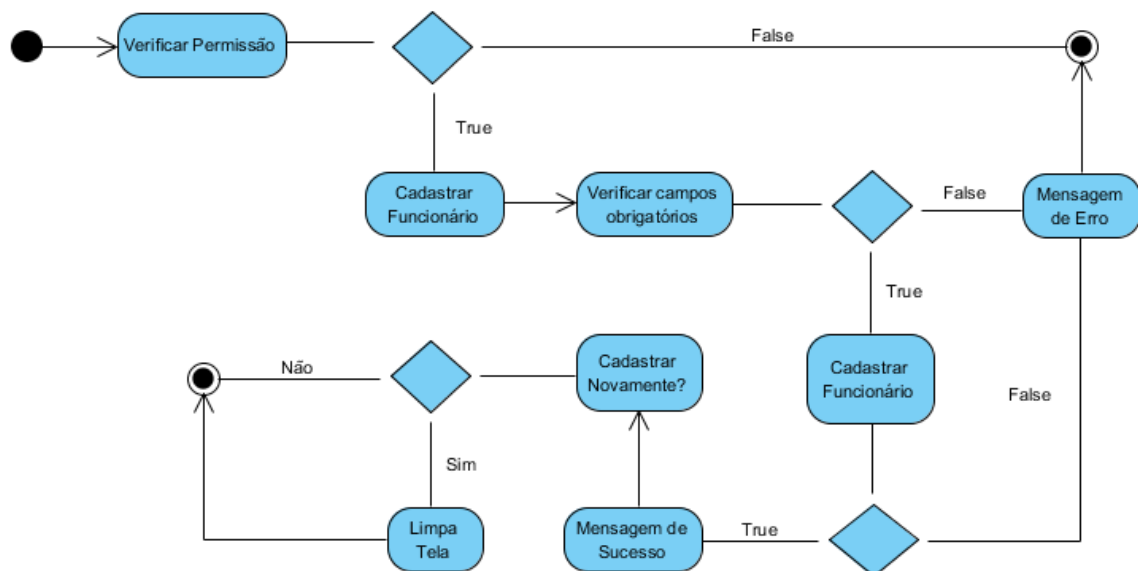


## Recuperar senha

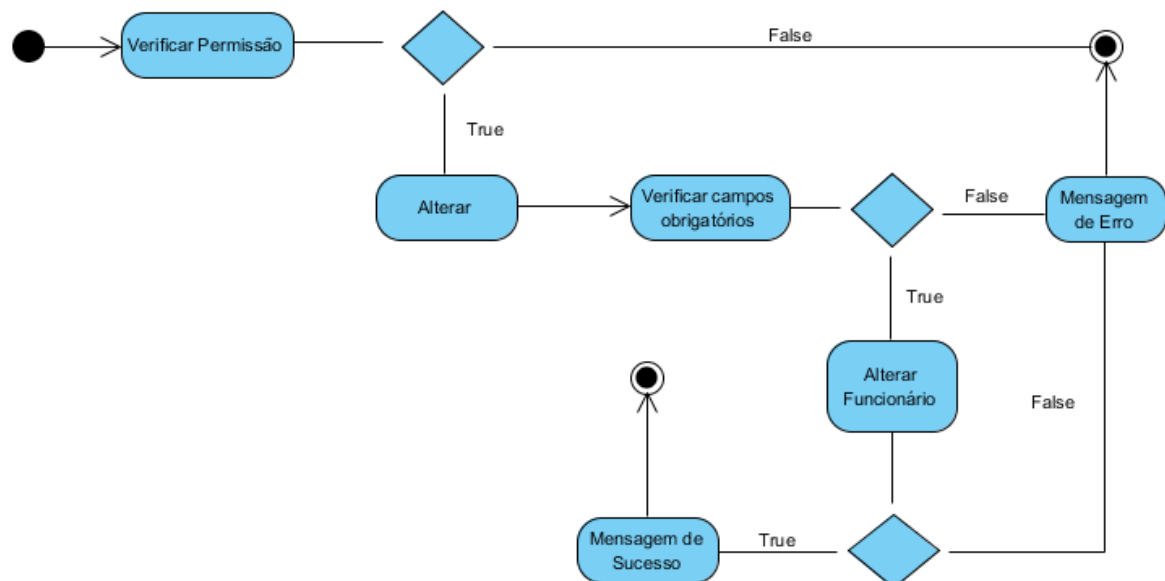


#### 4.4.2 Controle de Funcionário

##### Cadastro de Funcionário

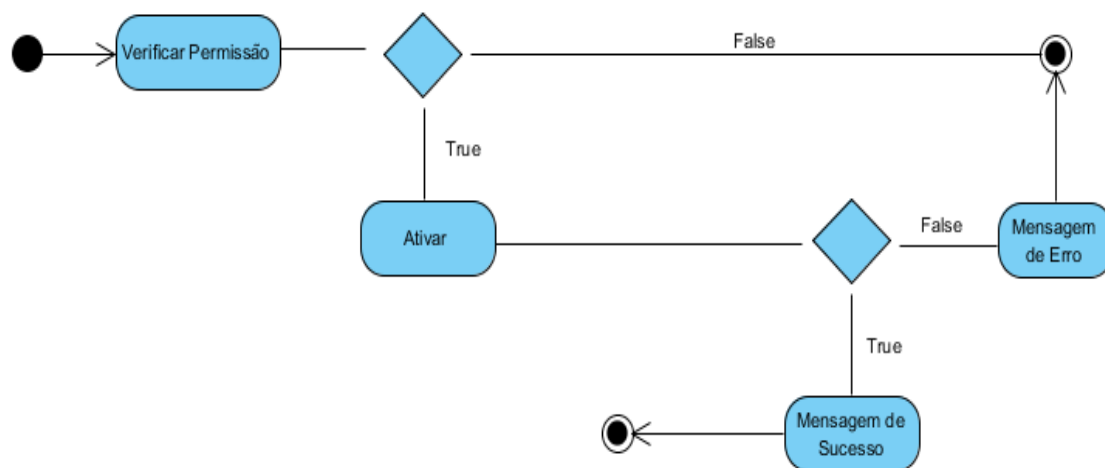


##### Alterar Funcionário

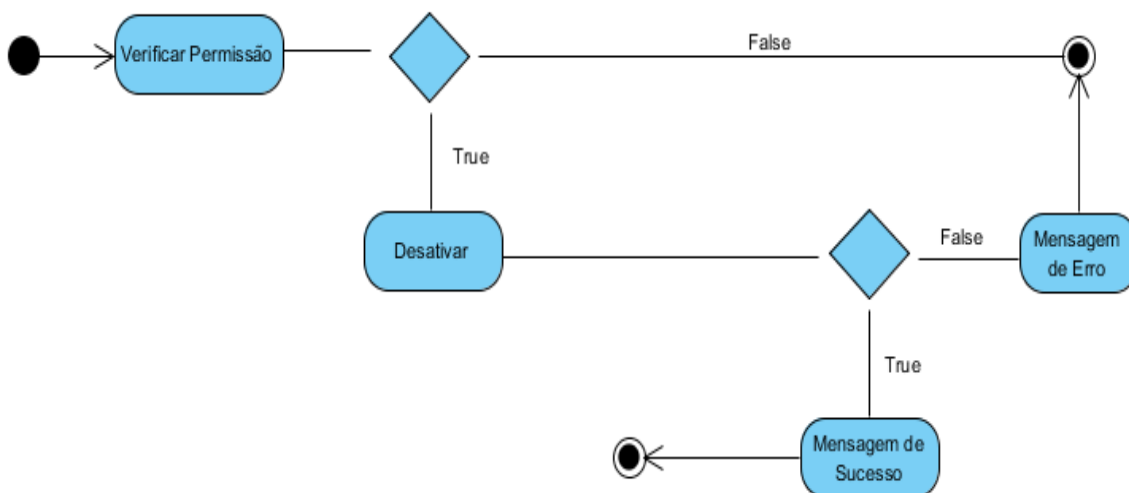




## Ativar Funcionário

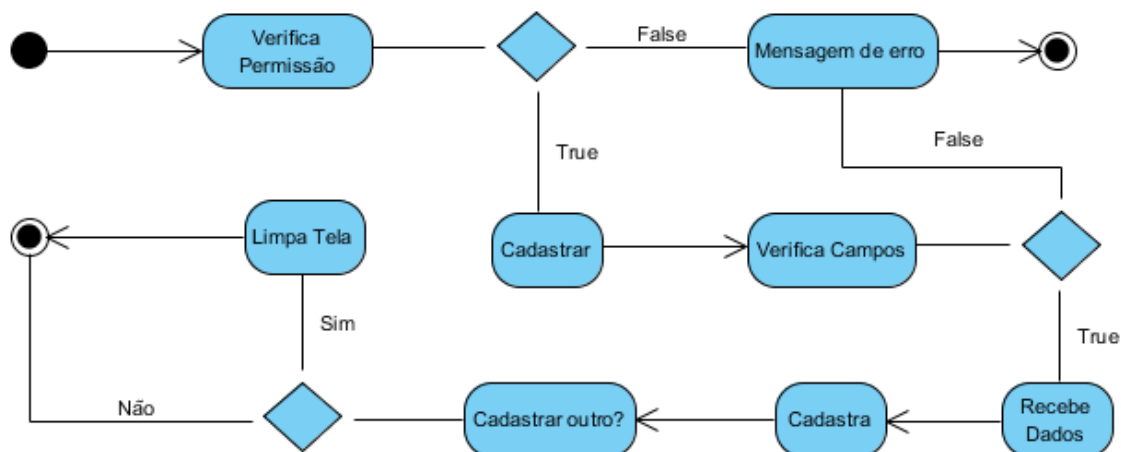


## Desativar Funcionário

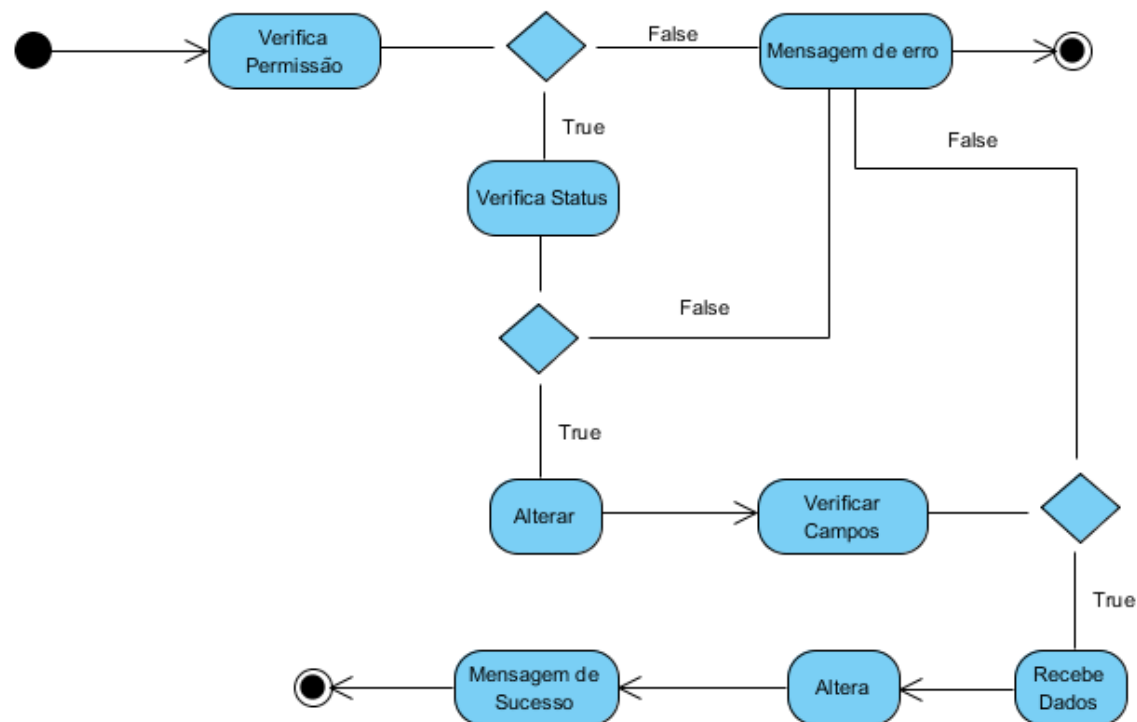


## 4.4.3 Controle de Product Backlog

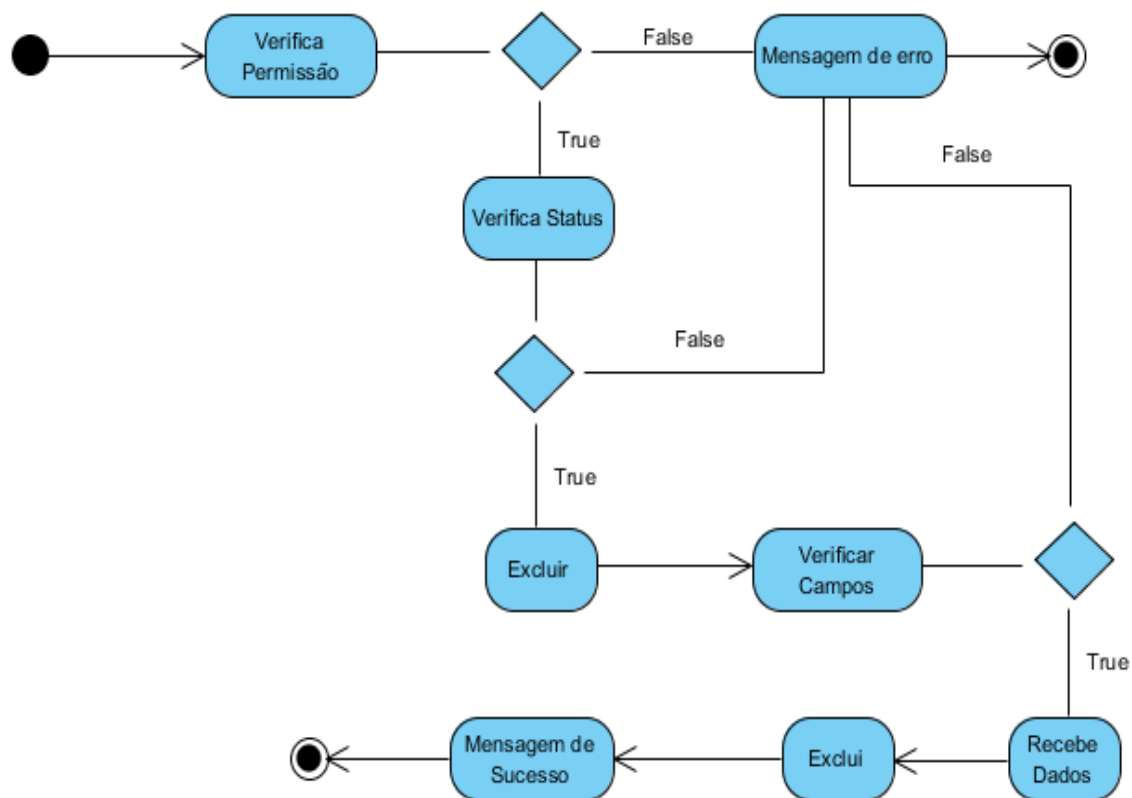
## Cadastrar Product Backlog



## Alterar Product Backlog

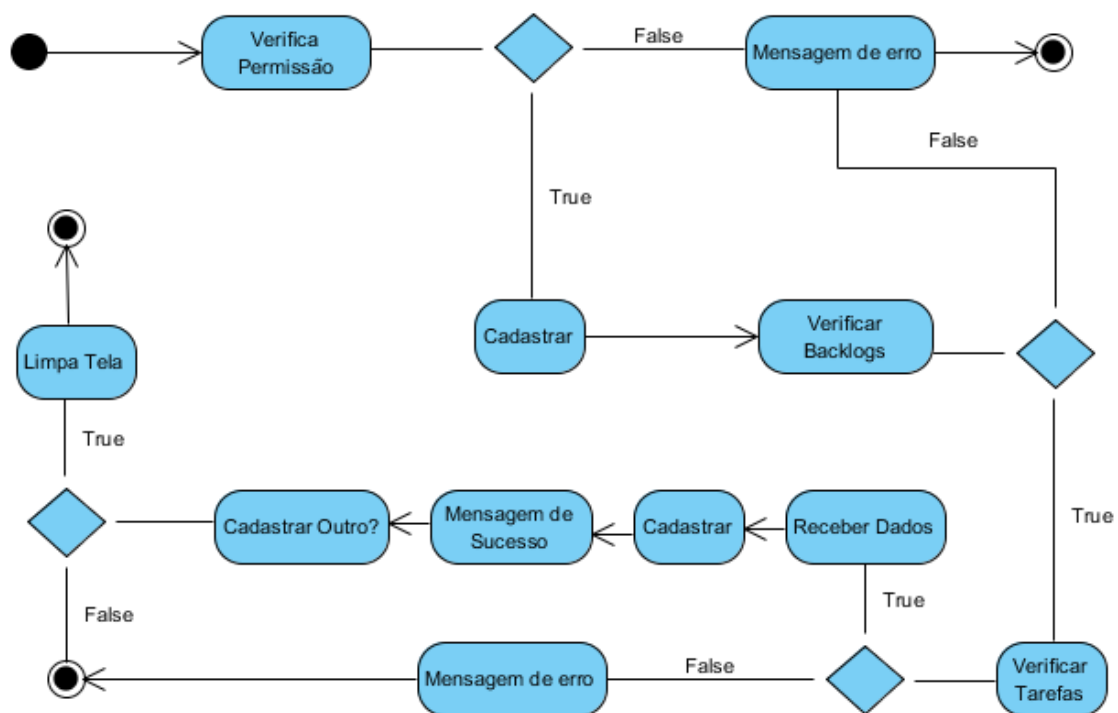


## Excluir Product Backlog

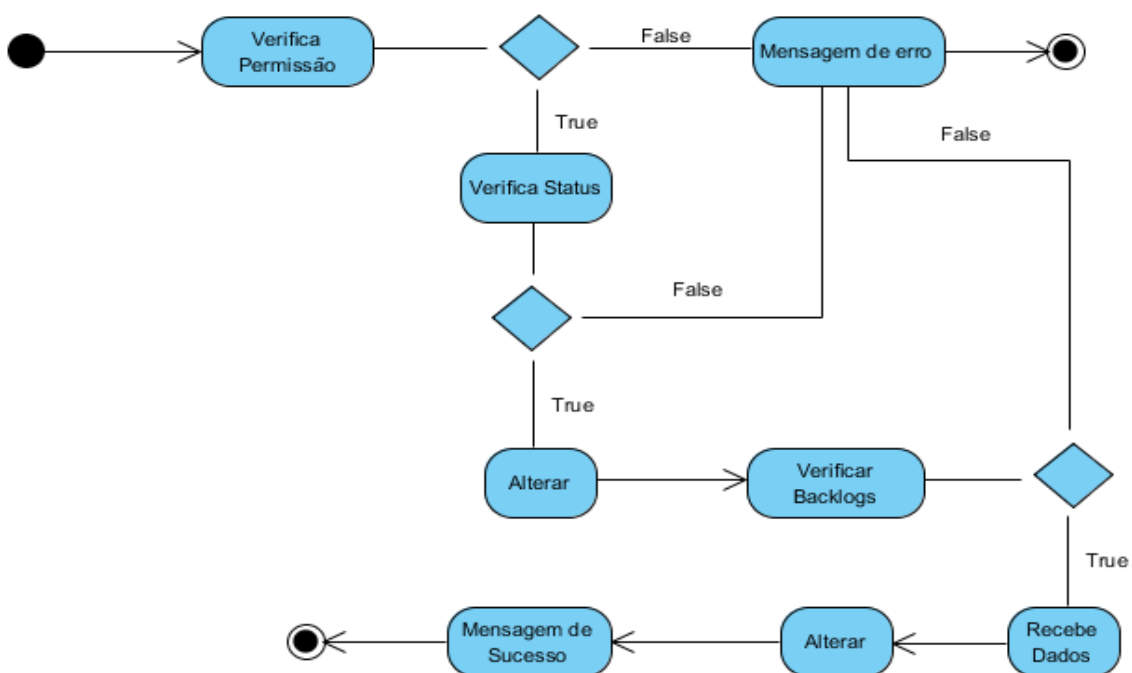


## 4.4.4 Controle de Sprint Backlog

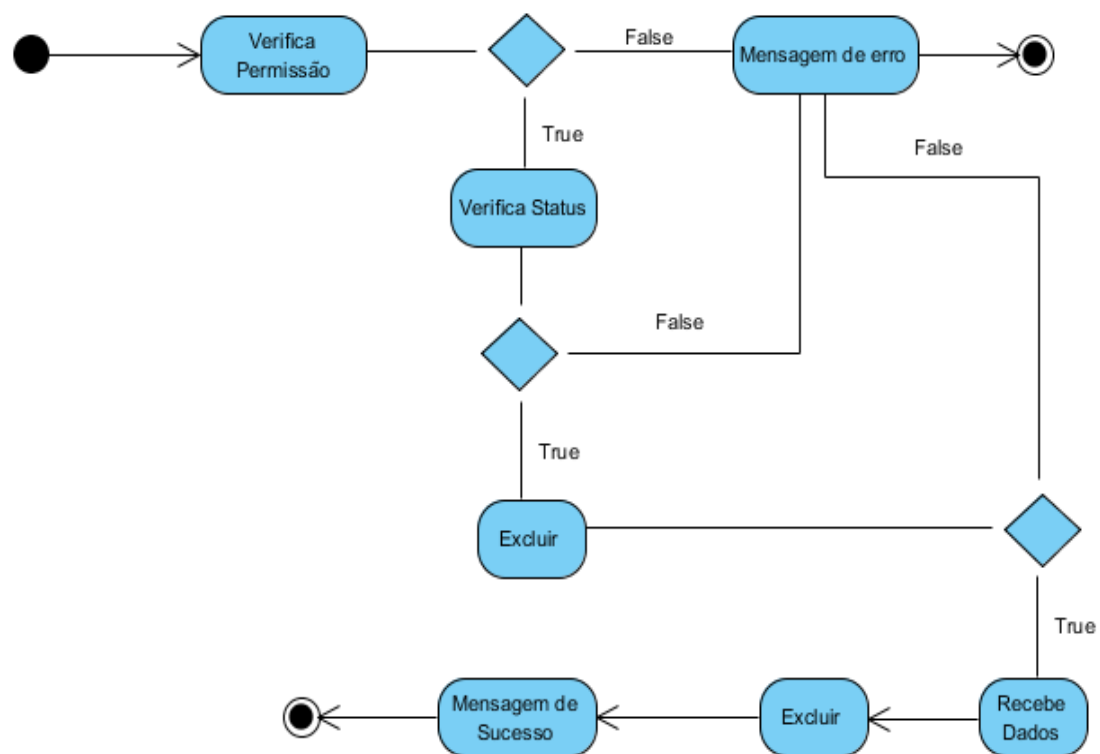
## Cadastrar Sprint Backlog



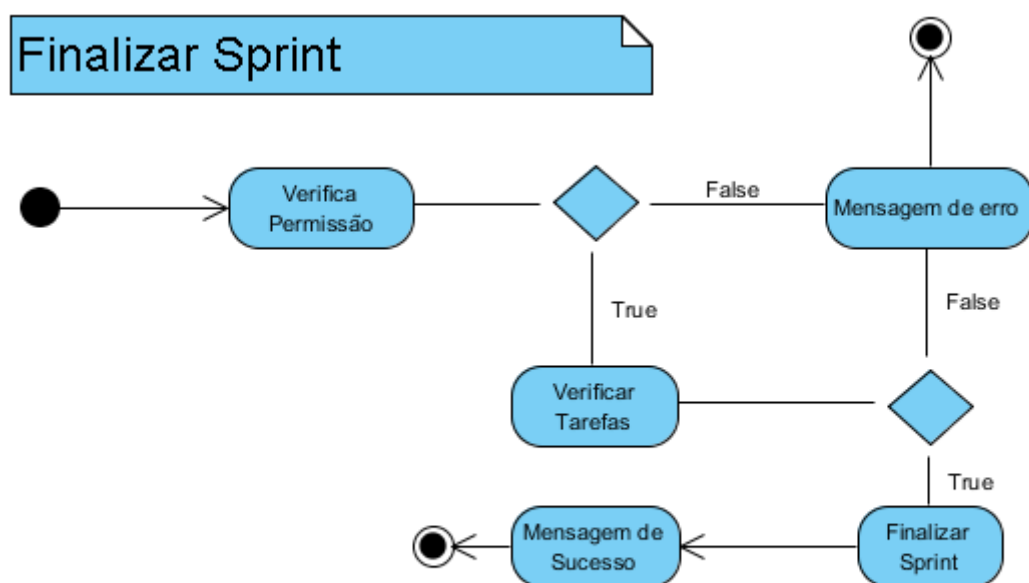
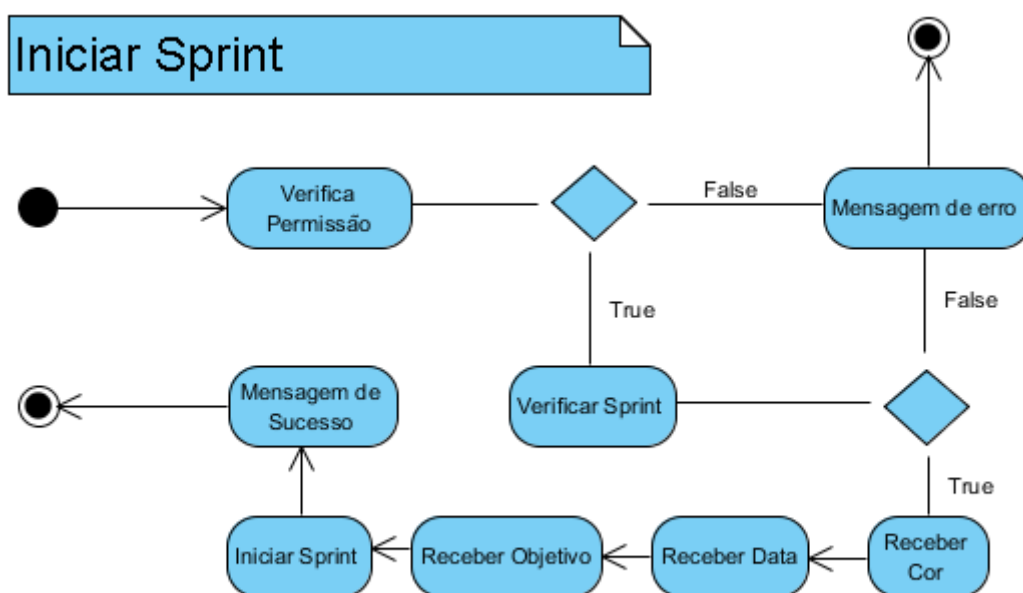
## Alterar Sprint Backlog



## Excluir Sprint Backlog

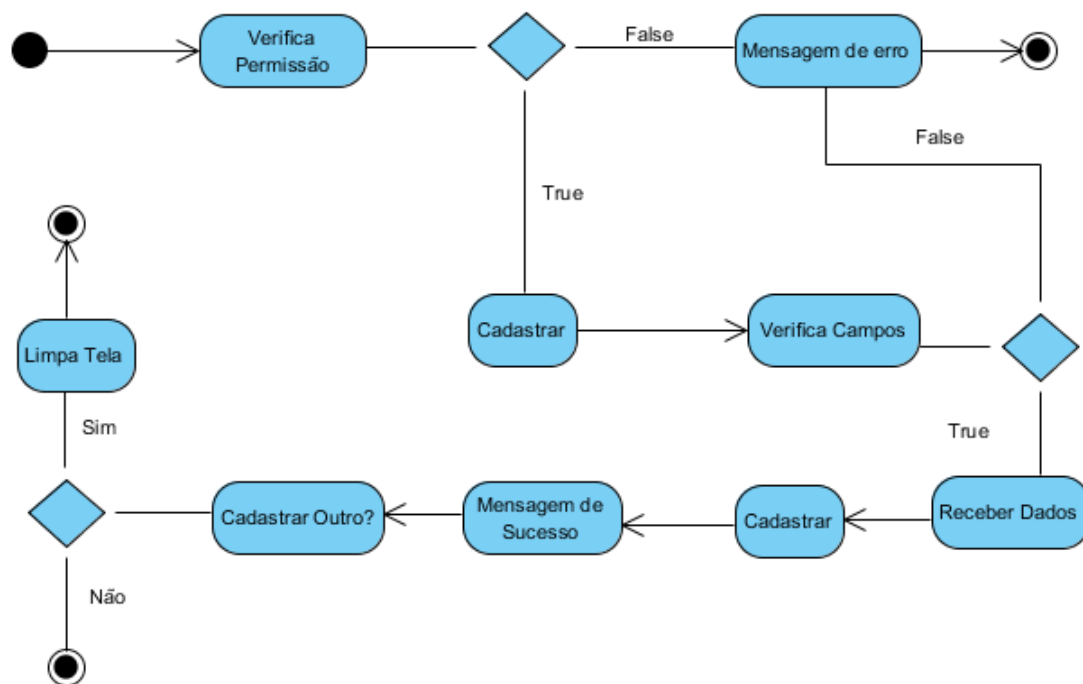


#### 4.4.5 Controle de Sprint

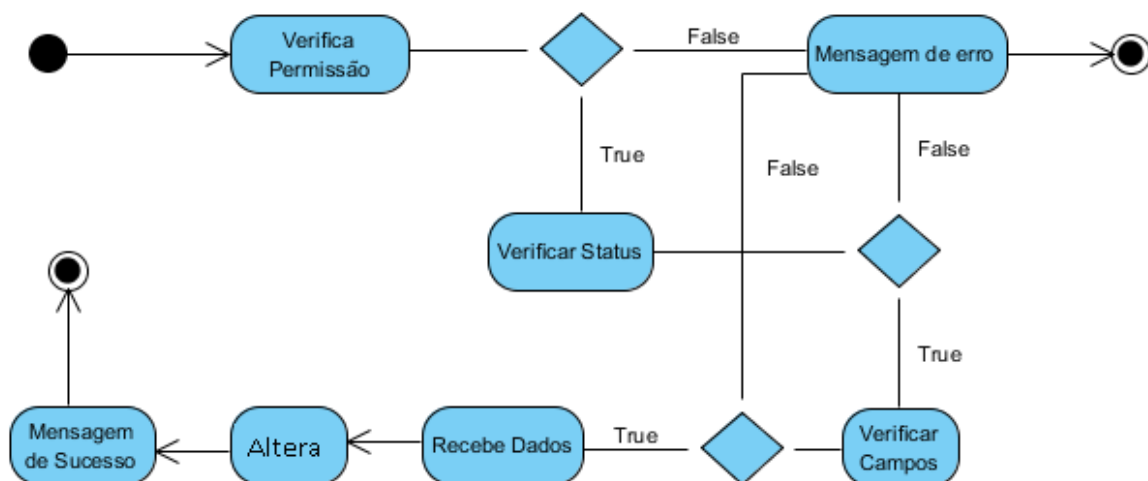


## 4.4.6 Controle de Tarefa

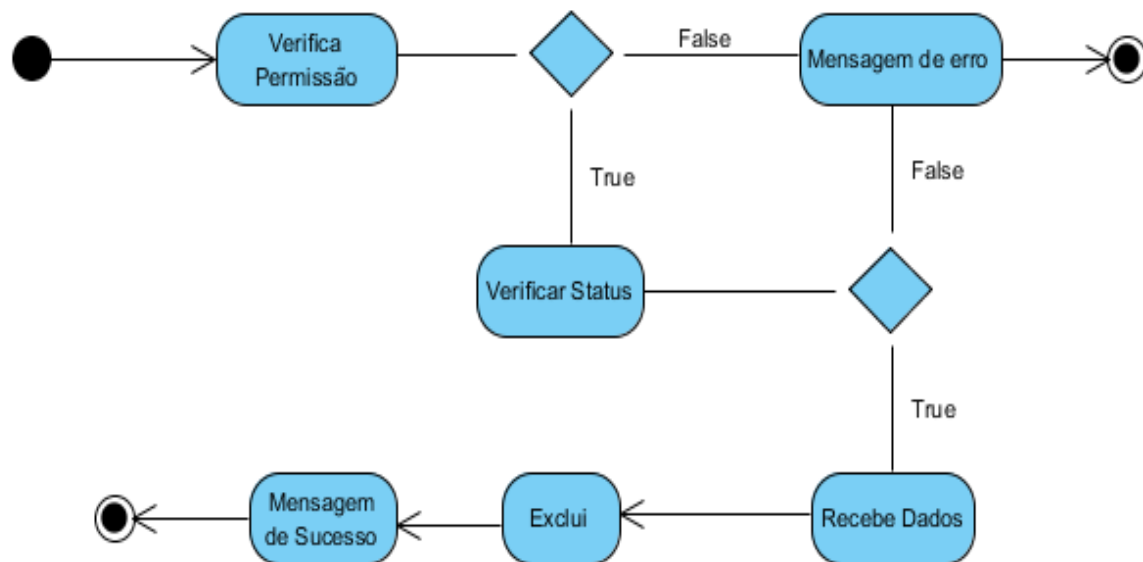
## Cadastrar Tarefa



## Alterar Tarefa



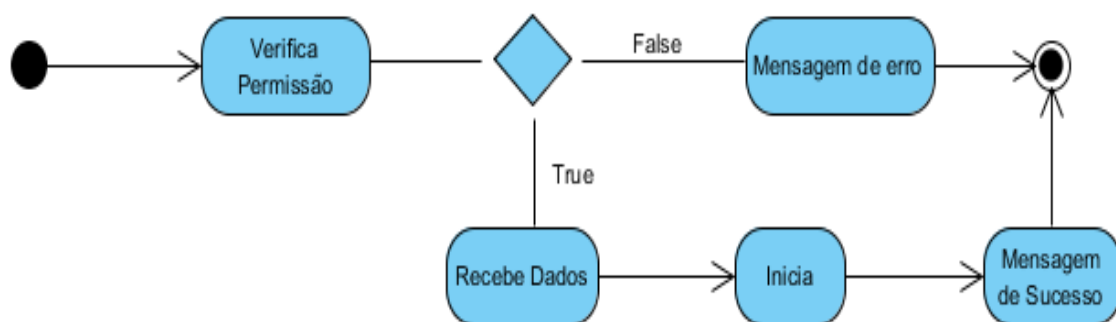
## Excluir Tarefa



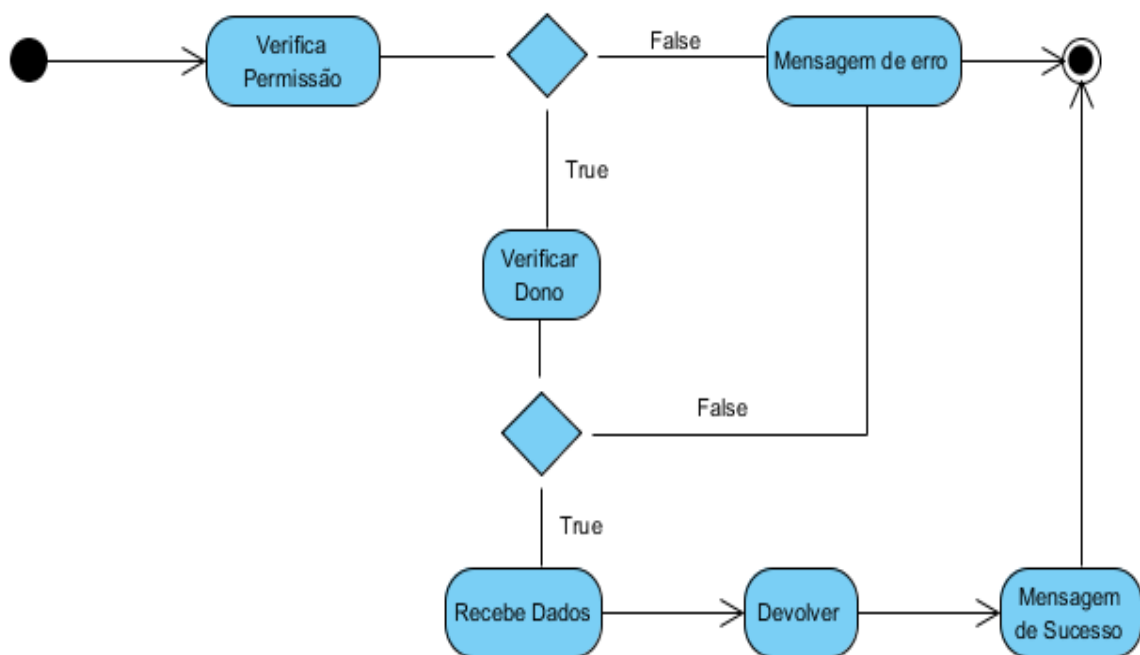


#### 4.4.7 Controle de Kanban

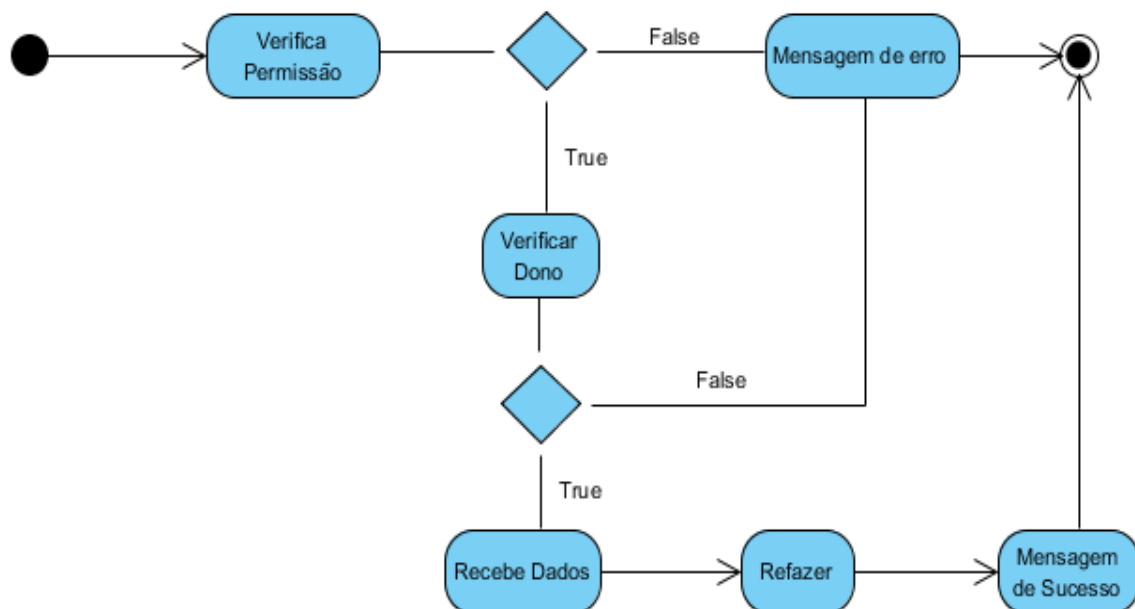
##### Iniciar Tarefa



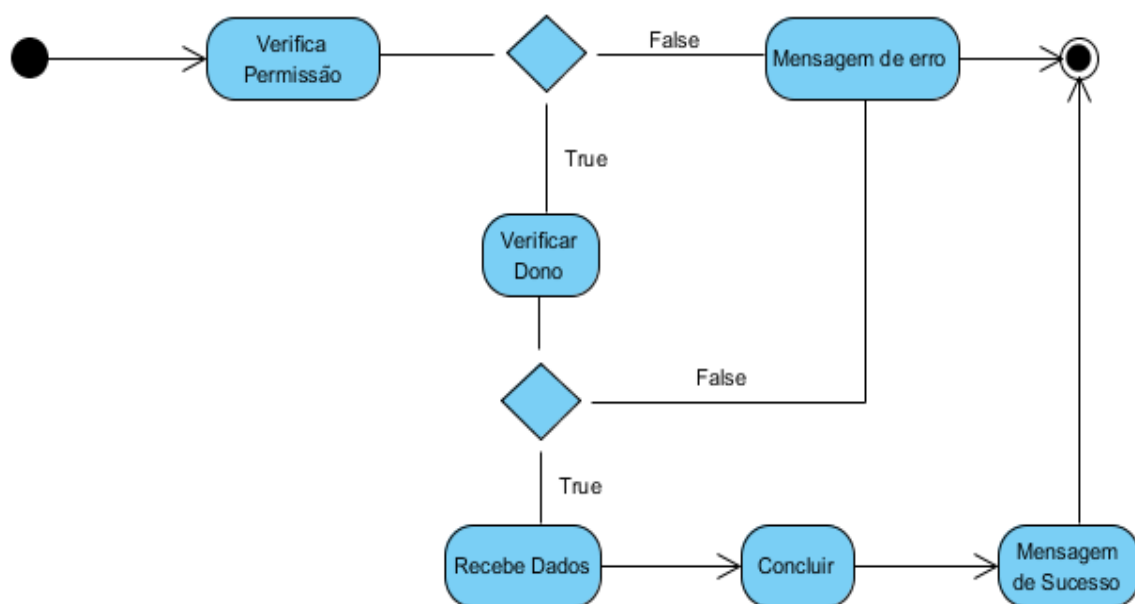
##### Devolver Tarefa



## Refazer Tarefa

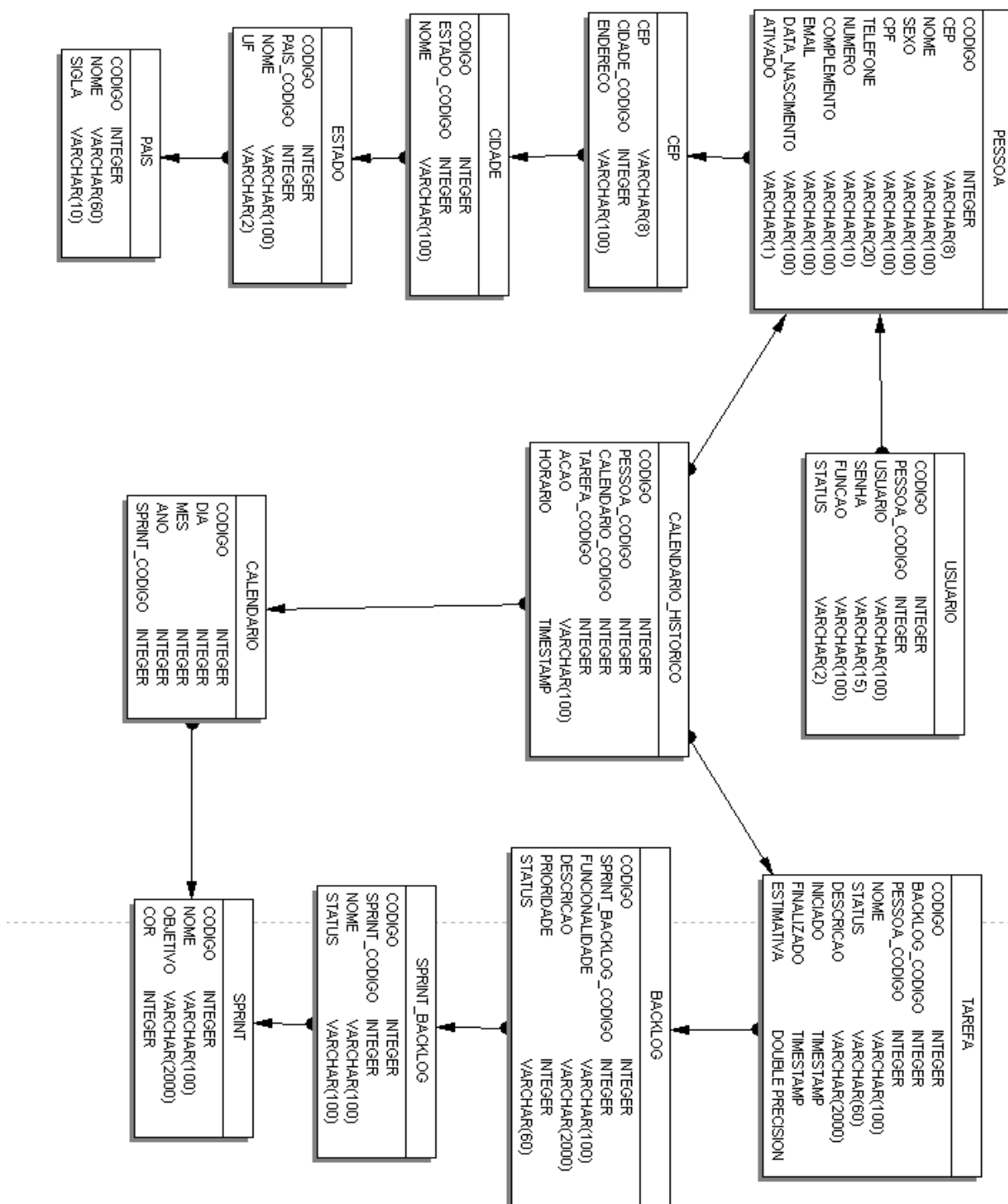


## Concluir Tarefa



#### 4.5 Modelo Entidade Relacionamento (MER)

A seguir iremos representar o diagrama entidade relacionamento (DER) referente à estrutura do banco de dados utilizado para manter a integridade dos dados deste trabalho.

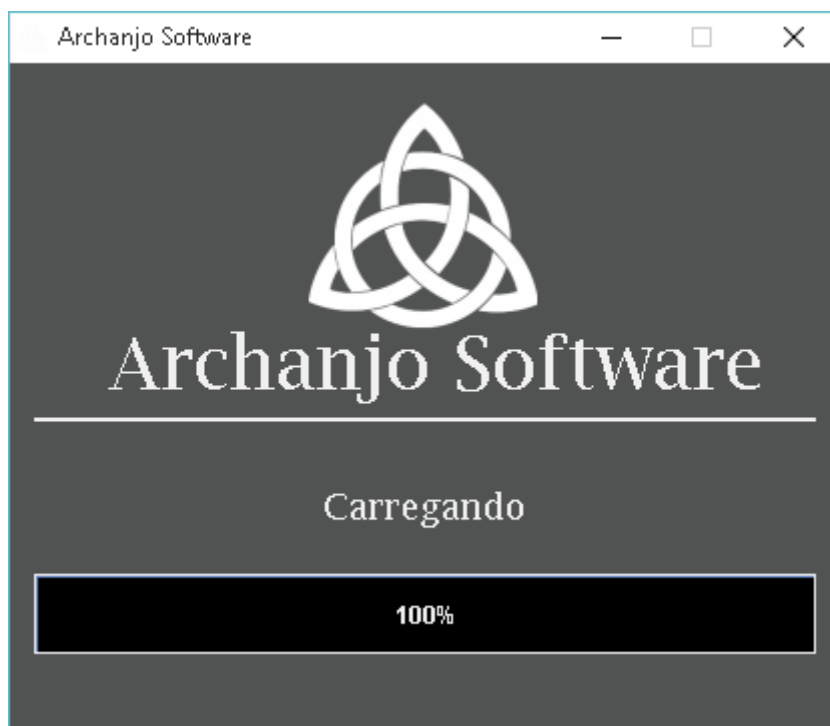


## 4.6 Software

A seguir será demonstrada as principais telas do sistema Archanjo Software. Essa documentação será de grande importância para que o usuário possa entender e se familiarizar com o ambiente do sistemas e suas funcionalidades.

### 4.6.1 Tela Carregamento

**Figura 16: Tela Carregamento**

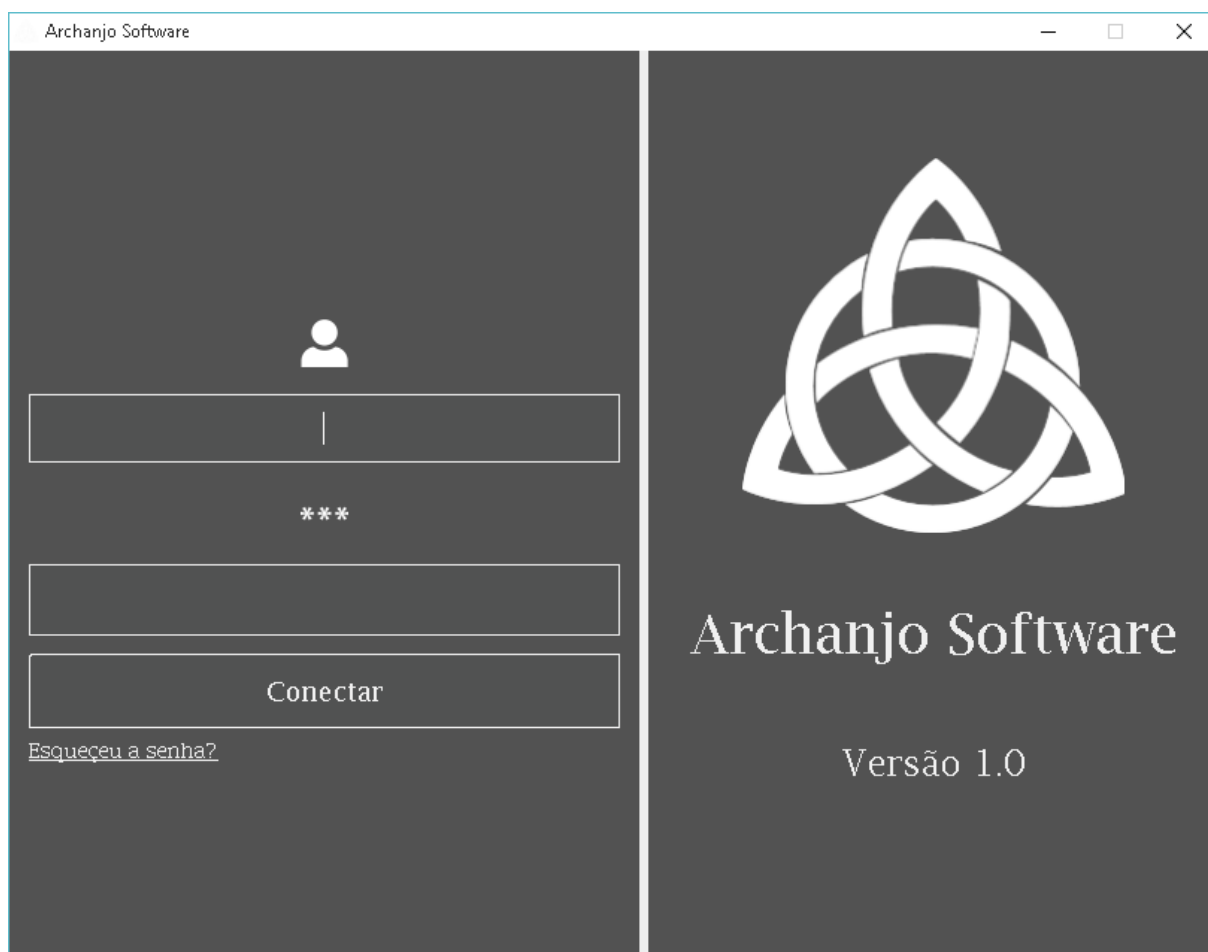


**Fonte: Autoria do autor.**

Primeira tela do sistema, onde sua funcionalidade é configurar os componentes necessários para inicialização do software, tais como bibliotecas e configurações de banco de dados.

#### 4.6.2 Tela Login

**Figura 17: Tela Login**

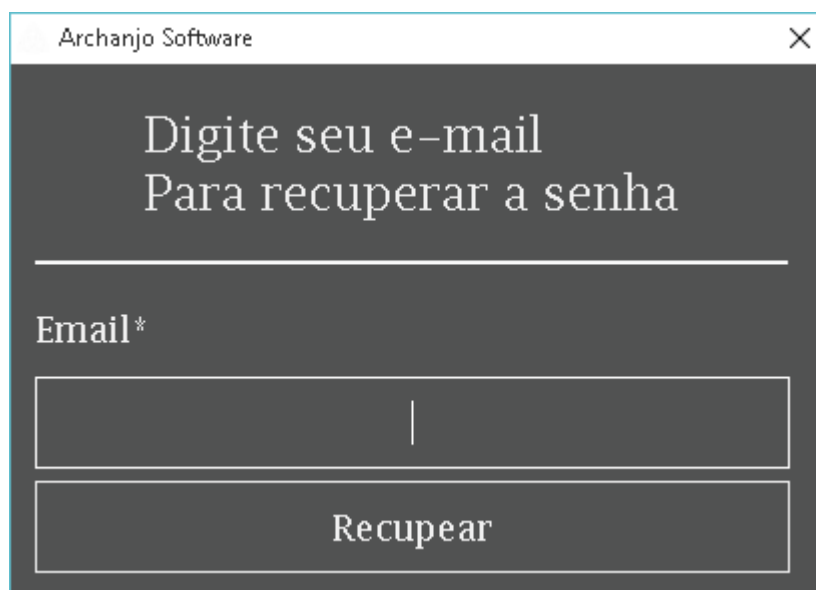


**Fonte: Autoria do autor.**

Tela de acesso ao software onde é possível se conectar com o e-mail referente ao seu cadastro e sua senha, caso for a primeira vez que se conecta ao sistema, o sistema pedirá que troque a senha e logo em seguida lhe redireciona para a Tela Kanban (Item 4.6.4).

#### 4.6.3 Tela Recuperar Senha

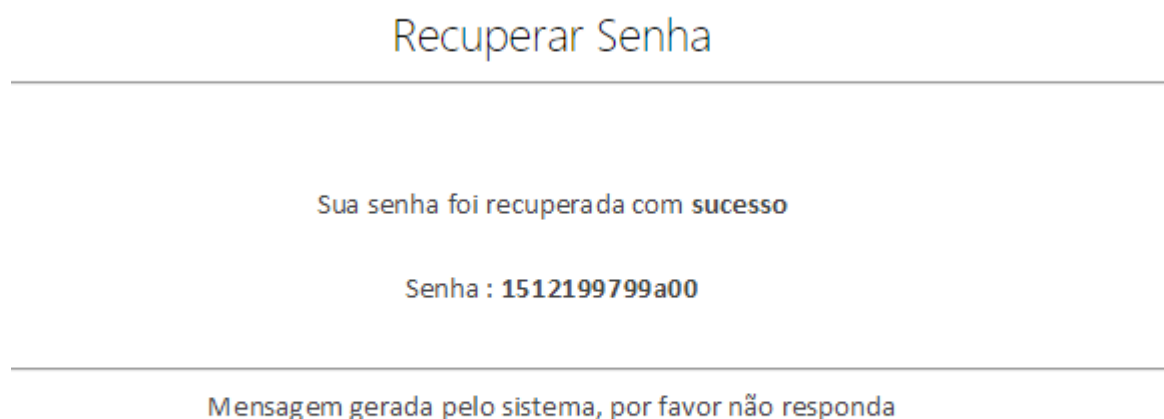
**Figura 18: Tela Recuperar Senha**

A screenshot of a web application window titled "Archanjo Software". The window has a dark gray background. At the top, it says "Digite seu e-mail Para recuperar a senha". Below this is a horizontal line. Under the line, the label "Email\*" is followed by a text input field. Below the input field is a button labeled "Recupear".

**Fonte: Autoria do autor.**

Caso o usuário esqueceu sua senha, o software proporciona a recuperação do mesmo, basta inserir o seu usuário (e-mail) e clicar em recuperar, dentro de segundos o software enviará um e-mail contendo sua senha (Figura 19).

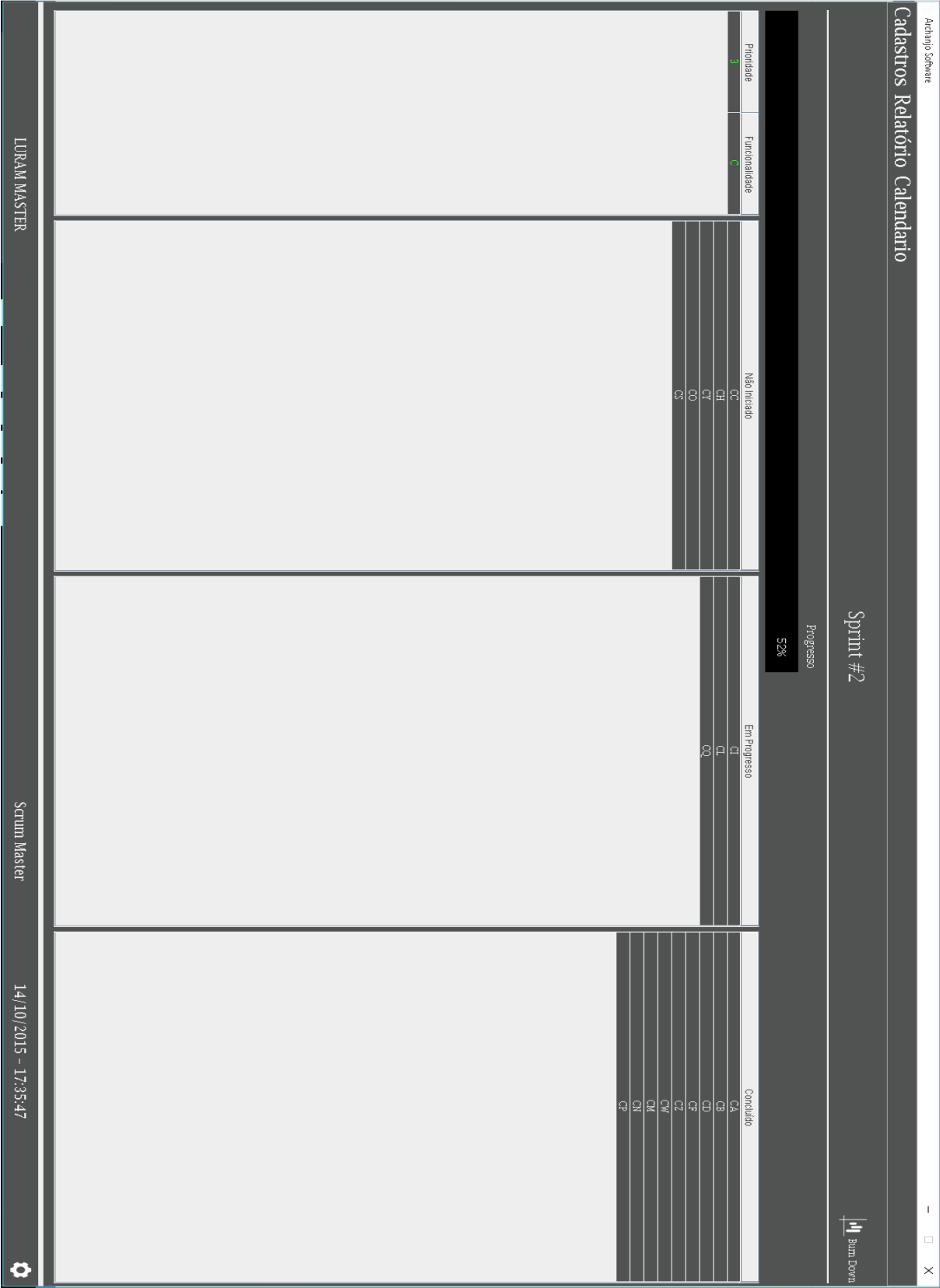
**Figura 19: E-mail recuperar senha**

A screenshot of an email interface. At the top, it says "Recuperar Senha". Below this is a horizontal line. Under the line, it says "Sua senha foi recuperada com sucesso". Below this, it says "Senha : 1512199799a00". At the bottom, it says "Mensagem gerada pelo sistema, por favor não responda".

**Fonte: Autoria do autor.**

4.6.4 Tela Kaban

Figura 20: Tela Kanban



Fonte: Autoria do autor.

Tela principal do sistema, onde dá acesso a todas as outras funcionalidades do software, nela contem quatro tabelas principais, onde a primeira lista as funcionalidades a serem desenvolvidas na *Sprint*, na segunda as tarefas não iniciadas, na terceira as tarefas em progresso e na última as tarefas concluídas.

Ao clicar duas vezes nas tarefas listadas na segunda, terceira e quarta tabela abre-se a tela kaban tarefa (Item 4.6.5).

Na parte superior entre o menu principal e as tabelas se encontra informações referente a *Sprint* e seu progresso, também dispõem a visualização do gráfico *Burndown* referente a *Sprint*, localizado no canto superior direito.

Na parte inferior encontra-se informações referente ao usuário conectado, sua função (*Product Owner*, *Scrum Master*, *Scrum Team*), data e hora e uma figura de engrenagem que dá acesso há Tela Conta (Item 4.6.7).

O menu principal localizado na parte superior, dá acesso a todas outras funcionalidades, lista de funcionário (Item 4.6.8), lista de *product backlog* (Item 4.6.10), lista de *sprint backlog* (Item 4.6.12), acesso a tela de relatório (Item 4.6.20) e acesso ao calendário (Item 4.6.17).



#### 4.6.5 Tela Kanban Tarefa

**Figura 21: Tela Kanban Tarefa**

Archanzo Software

Tarefa Técnica

Tarefa Status Não Iniciado

CC

Descrição Estimativa de Esforço 5 Dia(s)

asasa

5 / 2000

Iniciado Finalizado

Membro

Iniciar Tarefa

Sair Devolver Tarefa

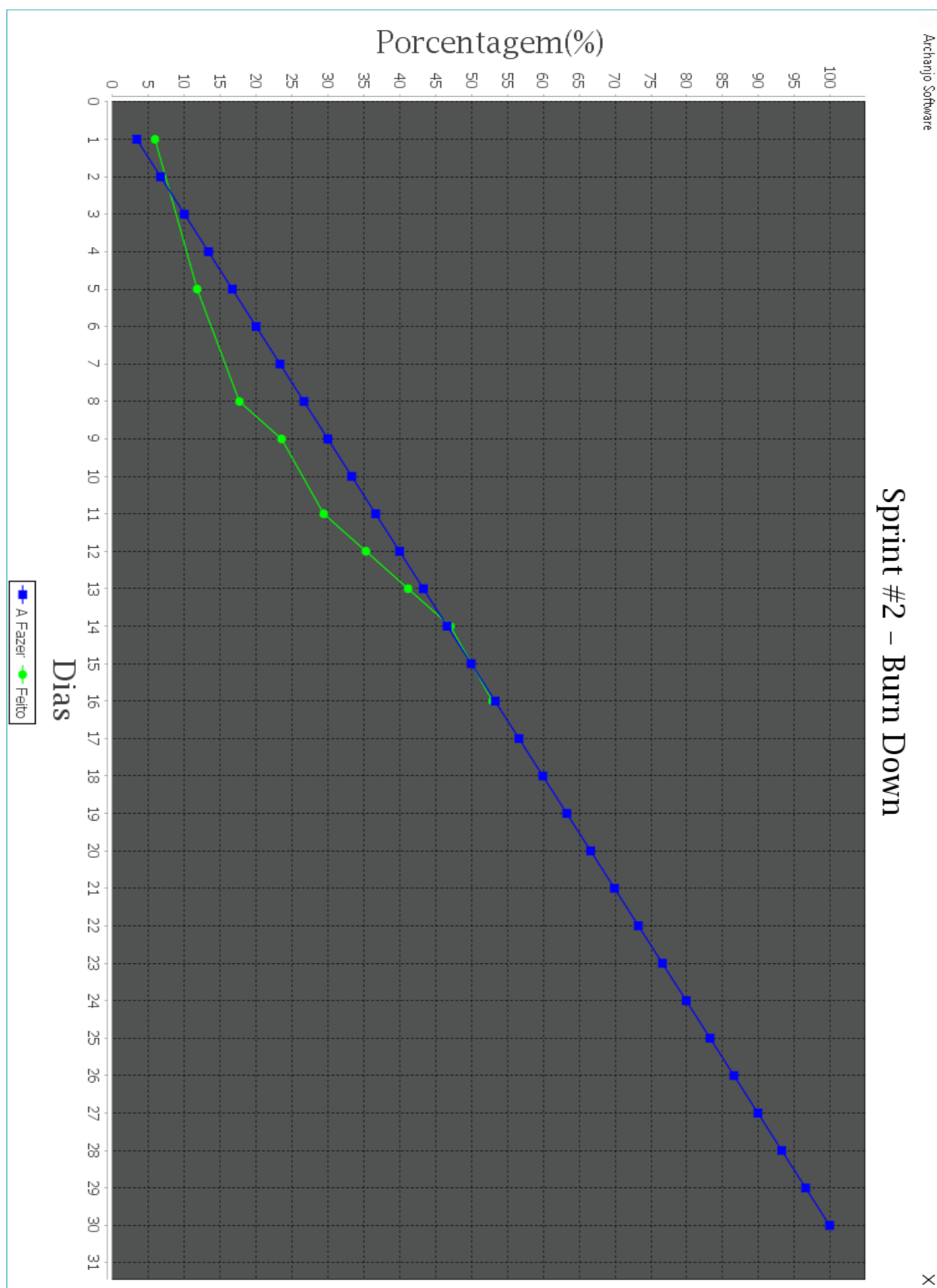
**Fonte: Autoria do autor.**

Está tela se abre quando uma tarefa for clicada duas vezes na tela kanban (Item 4.6.4) , nela é possível iniciar uma tarefa, devolver uma tarefa, refazer uma tarefa e concluir uma tarefa, onde é possível visualizar o objetivo da tarefa, tempo estimado para desenvolvimento do mesmo, data de inicio, data de termino e o membro que desenvolveu a tarefa.

Somente *Scrum Master* e *Scrum Team* podem efetuar as ações referente a está tela devido suas funções referente há metodologia Scrum.

#### 4.6.6 Tela Burndown

Figura 22: Tela Burndown



Fonte: Autoria do autor.

Tela *Burndown* provêm o desempenho referente a estimativa para o desenvolvimento da *Sprint* atual ou uma *Sprint* finalizada, pode ser aberto em duas telas, na tela sumário (Item 4.6.13), onde somente aparecerá a opção de visualizar *burndown* quando o mesmo estiver finalizado ou na tela kanban (item 4.6.4).

A linha azul diz a respeito oque se deve ser feito em porcentagem por dia estimado, logo a linha verde refere-se ao que foi desenvolvido no dia, ou seja, se a linha verde estiver abaixo da linha azul o projeto está atrasado logo se estiver acima da linha azul o projeto está adiantado.

Somente *Scrum Master* e *Product Owner* podem efetuar as ações referente a está tela devido suas funções referente há metodologia Scrum.

#### 4.6.7 Tela Conta

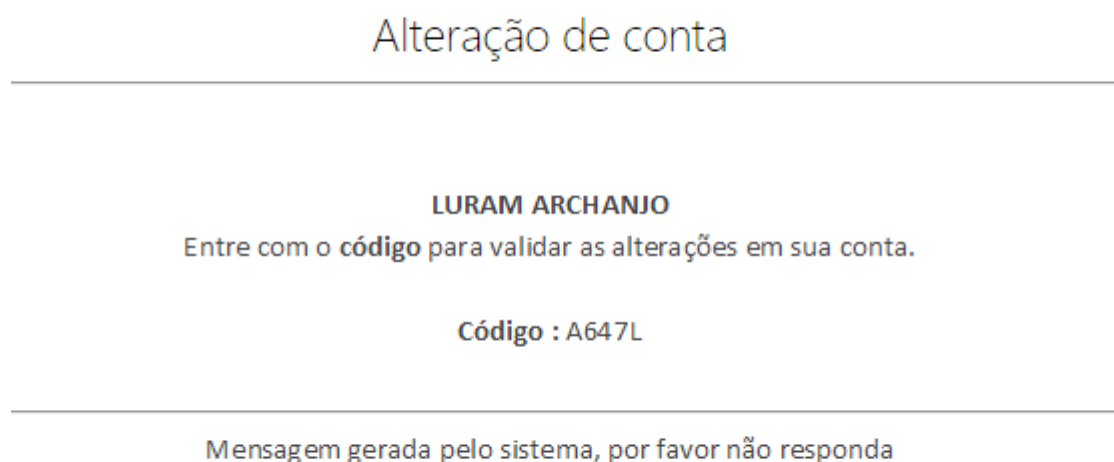
**Figura 23: Tela Conta**

The image shows a mobile application screen titled 'Tela Conta' (Account Page). It features a dark gray background with white text and input fields. The page is divided into two main sections by a horizontal line. The top section is for changing the password, with the title 'Nova senha' (New password) above a text input field. Below this is the title 'Confirmar nova senha' (Confirm new password) above another text input field. A button labeled 'Alterar' (Change) is positioned below the second input field. The bottom section is for changing the user name, with the title 'Usuário' (User) above a text input field. A button labeled 'Alterar' (Change) is positioned below this input field. At the very bottom of the screen, there is a button labeled 'Desconectar' (Disconnect).

**Fonte: Autoria do autor.**

Tela conta pode ser acessado na tela kanban (Item 4.6.4), onde dispõem alterações da conta tais como alteração de senha e alteração de usuário. Logo após a confirmação de alteração o software enviará um código de verificação em seu e-mail de cadastro.

Figura 24: E-mail alterar conta



Fonte: Autoria do autor.

Figura 25: Tela Validação

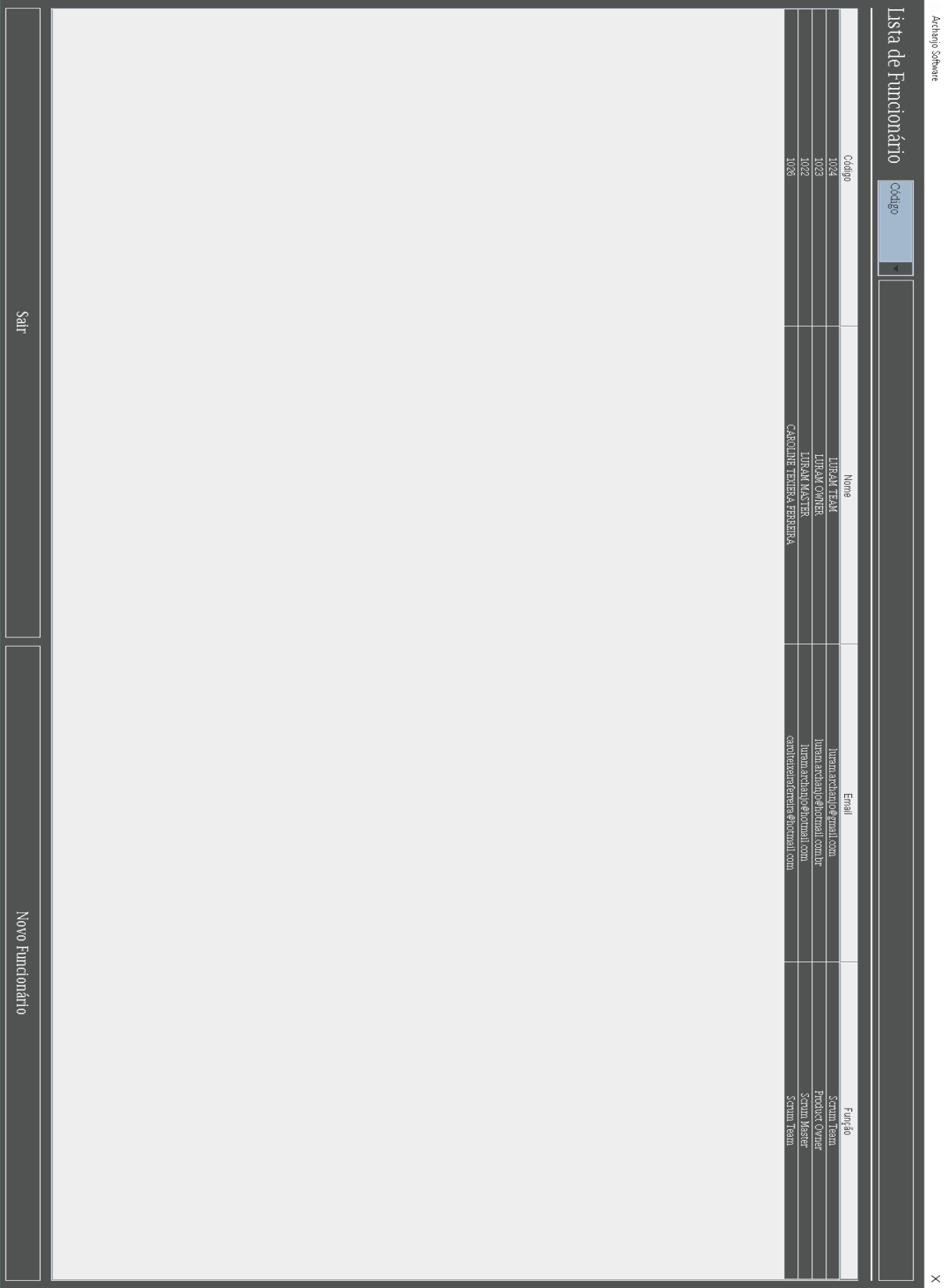


Fonte: Autoria do autor.

Após a verificação do mesmo o sistema trocará sua senha ou seu usuário (e-mail).

4.6.8 Tela Lista de Funcionário

Figura 26: Tela Lista de Funcionário



Fonte: Autoria do autor.

Tela lista de funcionário lista todos os funcionários cadastrados, tanto os funcionários ativados e desativados, onde provêm uma busca mais precisa com opção de filtrar por código, nome, e-mail e função na parte superior, na parte inferior a tela provêm a opção de cadastrar um novo funcionário e sair do mesmo, caso clique em novo funcionário a tela funcionário (Item 4.6.9) será exibida, no centro é possível ver todas as informações referente ao usuário, basta dar dois cliques na linha referente ao funcionário desejável.

A listagem é ordenada por nome decrescente, somente *Scrum Master* e *Product Owner* podem efetuar as ações referente a está tela devido uma regra que adotei, pois considero *Scrum Team* o usuário de mais baixo nível do software, portanto ele não pode ter controle referente há funcionários.

#### 4.6.9 Tela Funcionário

**Figura 27: Tela Funcionário**

Archanjo Software

### Funcionário

Nome completo\*

Sexo\* ☐ Feminino ☐ Masculino

Data de Nascimento\* / /

CPF\* . . -

Telefone\* ( ) -

CEP -

Endereço

Número

Complemento

Estado AC

Cidade Acrelândia

Email\*

Função Scrum Team

Sair Cadastrar

**Fonte: Autoria do autor.**

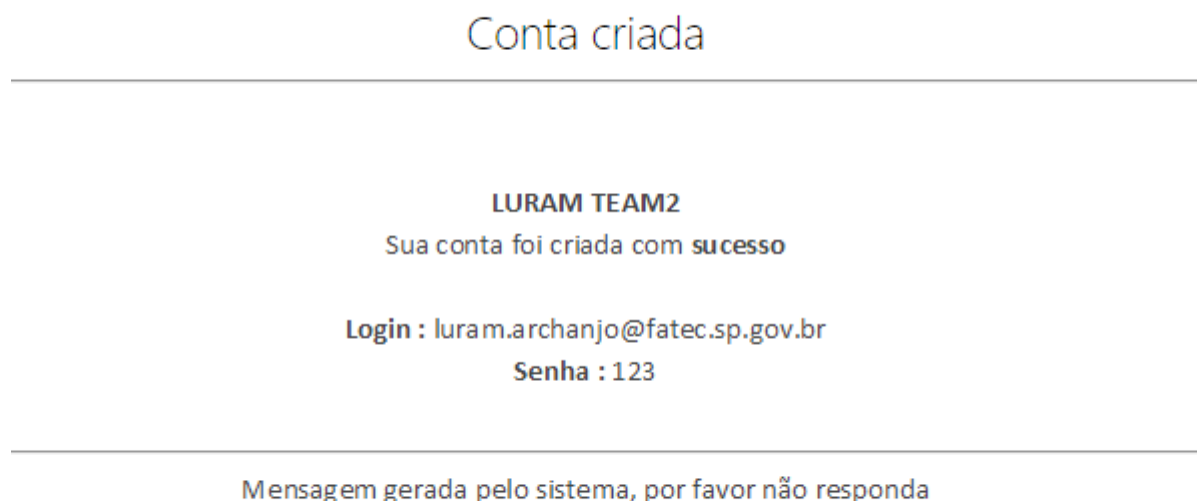
Tela funcionário provêm o cadastro de novos funcionários no software, os campos obrigatórios são os campos com o símbolo (\*) na frente, caso estes não forem preenchidos o software exibirá uma mensagem de erro.

O software não permite cadastrar dois e-mails iguais devido o fato do e-mail ser o usuário para acessar o sistema, e dois CPFs iguais.

O software não permite cadastrar dois *Product Owner* e nem dois *Scrum Master*, devido o software ser específico para gerenciar somente uma equipe.

Logo após o cadastro do novo funcionário, o software enviará um e-mail, com os dados de login e senha.



**Figura 28: E-mail criar conta**

**Fonte: Autoria do autor.**

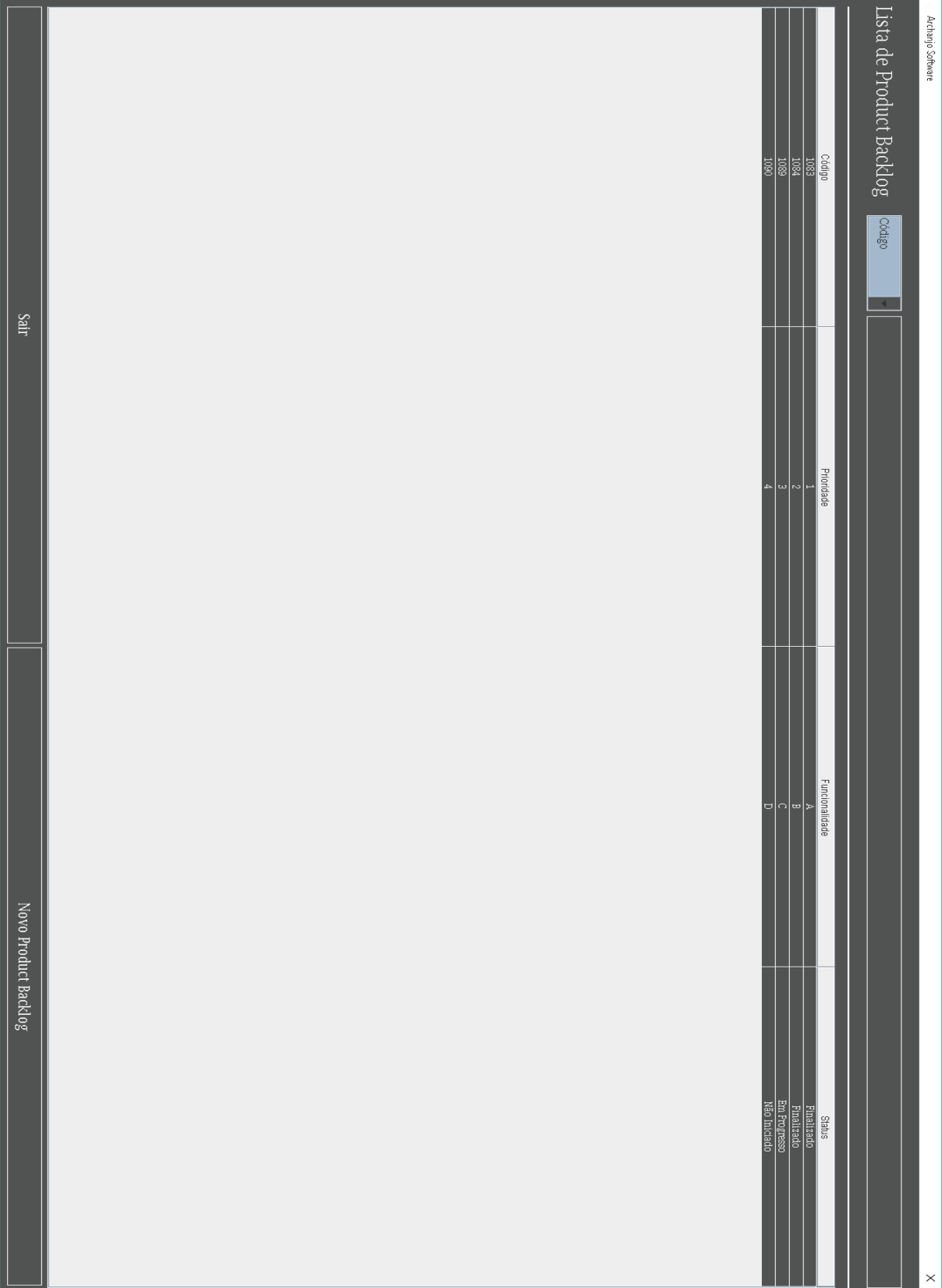
No primeiro acesso do novo usuário o software pedirá para trocar a senha.

**Figura 29: Tela Alterar Senha**

**Fonte: Autoria do autor.**

4.6.10 Tela Lista Product Backlog

Figura 30: Tela Lista Product Backlog



Fonte: Autoria do autor.

Tela lista de *product backlog* lista todos os *backlogs* cadastrados, onde provêm uma busca mais precisa com opção de filtrar por código, prioridade, funcionalidade e status na parte superior, na parte inferior a tela provêm a opção de cadastrar um novo *product backlog* e sair do mesmo, caso clique em novo *product backlog* a tela *product backlog* (Item 4.6.11) será exibida, no centro é possível ver todas as informações referente ao *product backlog*, basta dar dois cliques na linha referente ao *product backlog* desejável.

#### 4.6.11 Tela Product Backlog

**Figura 31: Tela Product Backlog**

Archanjo Software

## Product Backlog

Funcionalidade

Status Não Iniciado

Descrição

0 / 2000

Prioridade 5

Sair Cadastrar

**Fonte: Autoria do autor.**

Tela *product backlog*, provêm cadastrar novos *product backlog*, alterações e exclusões do mesmo.

Todos os campos funcionalidade, descrição e prioridade são obrigatórios, portanto é preciso preencher todos para que o software cadastre o novo *product backlog*, caso contrário o software exibirá uma mensagem de erro.

Não é possível cadastrar dois *product backlog* com a mesma prioridade devido o software ser para somente uma equipe.

Somente o *Product Owner* pode efetuar as ações referente há está tela devido suas funções dentro da metodologia Scrum.

4.6.12 Tela Lista de Sprint Backlog

Figura 32: Tela Lista de Sprint Backlog



Fonte: Autoria do autor.

Tela lista de *sprint backlog* lista todos as *sprint backlogs* cadastradas, onde provêm uma busca mais precisa com opção de filtrar por código, nome e status na parte superior, na parte inferior a tela provêm a opção de cadastrar um novo *sprint backlog* e sair do mesmo, caso clique em novo *sprint backlog* a tela sumário (Item 4.6.13) será exibida, no centro é possível ver todas as informações referente há *sprint backlog*, basta dar dois cliques na linha referente há *sprint backlog* desejável.

#### 4.6.13 Tela Sumário

### Figura 33: Tela Sumário

[illegible]

**Fonte: Autoria do autor.**

Tela sumário provêm adicionar os *product backlog* (item 4.6.14) a nova *sprint backlog* e respectivamente suas tarefas técnicas (item 4.6.15).

Exibindo a estimativa total da *sprint backlog* no canto inferior direita. Após adicionar os *product backlogs* e suas respectivas tarefas técnicas é possível a alteração dos mesmos e a finalização da nova *sprint backlog*.

Nesta tela é possível iniciar uma nova *Sprint* e a finalização do mesmo, é possível acessar está opção clicando duas vezes em uma *sprint backlog* já cadastrada na tela lista de *sprint backlog* (item 4.6.13).

Caso clique em iniciar *Sprint* o software exibirá há tela *Sprint* (item 4.6.16), caso há *sprint backlog* estiver finalizada é possível visualizar o gráfico *burndown* (item 4.6.6) do mesmo.

Somente o *Scrum Master* pode efetuar ações referente há está tela, devido suas funções dentro da metodologia Scrum.

#### 4.6.14 Tela Backlog

**Figura 34: Tela Backlog**

Archanjo Software

Adicione tarefas referente ao Backlog

Backlog 4 - D

Adicionar Tarefa

Tarefa	Estimativa
--------	------------

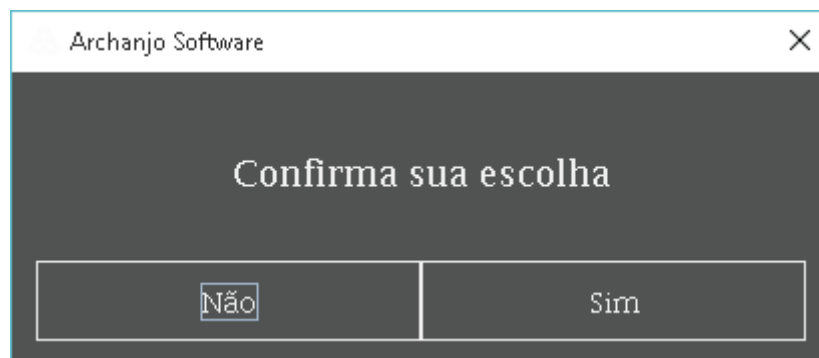
Estimativa Total : 0 Dias

Sair Finalizar Backlog

**Fonte: Autoria do autor.**

Tela *backlog* provêm adicionar as tarefas técnicas referente ao *product backlog* selecionado, para isto o software exibi uma tela de confirmação do mesmo.

**Figura 35: Tela Confirmar Escolha**



**Fonte: Autoria do autor.**

Após a confirmação, não é possível selecionar outro *product backlog*.

Ao clicar em adicionar no tarefa, localizado no canto superior direito, o software exibirá a tela tarefa (item 4.6.15), onde é possível cadastrar novas tarefas técnicas.

No centro é possível acessar as tarefas técnicas e efetuar alterações ou exclusão da mesma, basta clicar duas vezes na tarefa desejável. Após cadastrar as tarefas é preciso finalizar, clicando no botão localizado no canto inferior direito.



#### 4.6.15 Tela Tarefa

**Figura 36: Tela Tarefa**

Archanjo Software

### Tarefa Técnica

Tarefa Status Em Espera

Descrição Estimativa de Esforço 1 Dia(s)

0 / 2000

Sair Cadastrar

**Fonte: Autoria do autor.**

Tela tarefa, provêm cadastrar, alterar e excluir. Todos os campos são obrigatórios tarefa, descrição e estimativa de esforço.

Caso os campos não forem preenchidos o software exibirá uma mensagem de erro, portanto é preciso preencher todos os campos para efetuar as ações de alteração e de cadastro.

Para acessar a opção de alteração e exclusão, é preciso clicar duas vezes em uma tarefa já cadastrada na tela *backlog* (item 4.6.14).

Somente o *Scrum Master* pode efetuar ações referente há está tela devido suas funções dentro da metodologia Scrum.

#### 4.6.16 Tela Sprint

**Figura 37: Tela Sprint**

Titulo

Objetivo da Sprint

Texto / 2000

Estimativa Total      Dias      Data de Início

Sair      Iniciar Sprint

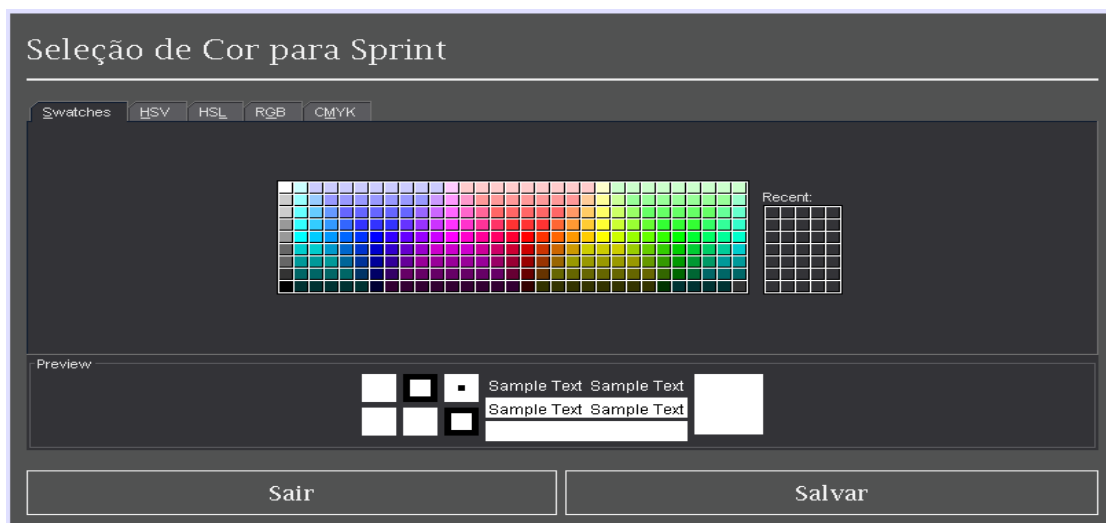
**Fonte: Autoria do autor.**

Tela *Sprint* provêm iniciar uma *sprint*, porém é preciso que todas as anteriores estejam finalizadas.

Os campos obrigatórios objetivo da *Sprint* e data de início é preciso serem preenchido, caso contrário o software exibirá uma mensagem de erro.

Ao clicar no ícone de calendário o software exibirá uma tela para escolher a cor referente há *sprint*.

**Figura 38: Tela Cor**



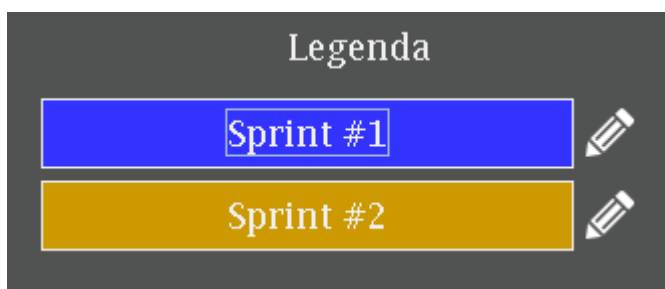
**Fonte: Autoria do autor.**

Após escolher a cor o software exibirá a tela calendário (item 4.6.17).

Não é possível selecionar um data que já tenha uma *sprint* agendada.

É possível alterar a cor da *Sprint*, indo na tela calendário (item 4.6.17), e clicando no ícone semelhante há um lápis.

**Figura 39: Legenda**

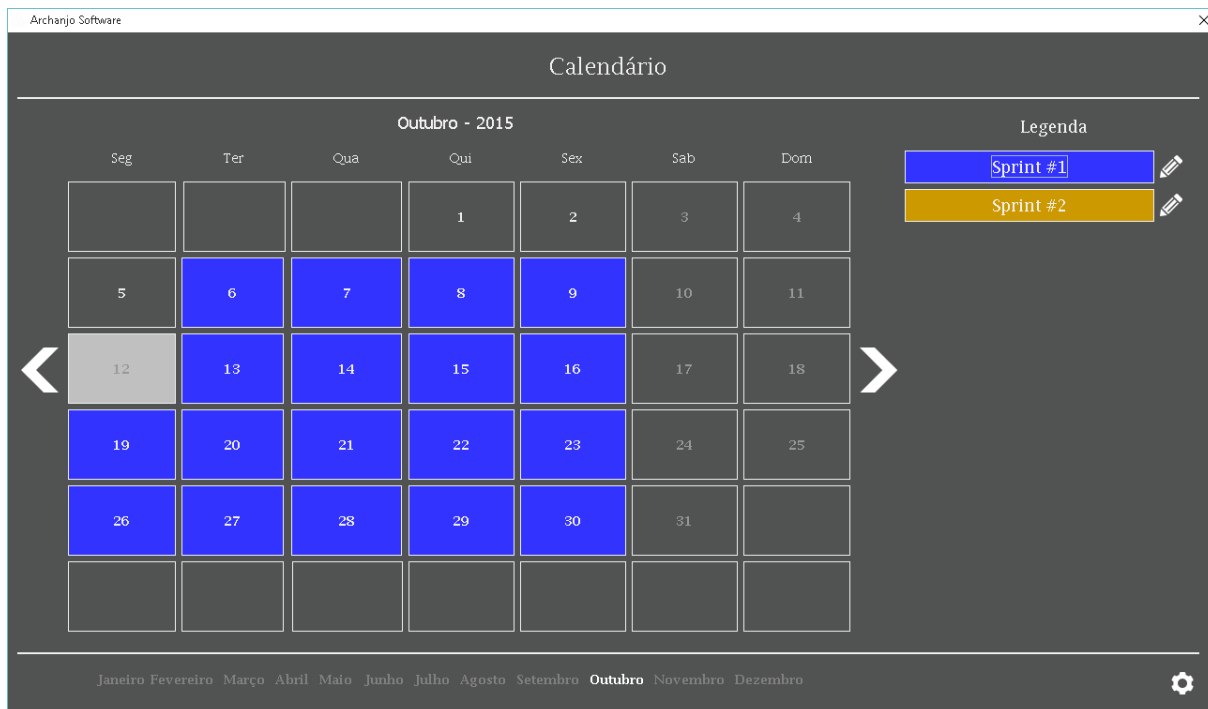


**Fonte: Autoria do autor.**

Somente o *Scrum Master* pode efetuar ações referente há está tela devido suas funções dentro da metodologia Scrum.

#### 4.6.17 Tela Calendário

**Figura 40: Tela Calendário**



**Fonte: Autoria do autor.**

Tela calendário provêm um calendário até o ano de dois mil e vinte, onde é possível acessar informações do dia clicando no dia desejável o software exibirá a tela histórico (item 4.6.19).

No canto superior direito são listados todas as *sprints*, caso queria saber o inicio da mesma clique nela e a tela se desloca para o primeiro dia da *sprint* selecionada.

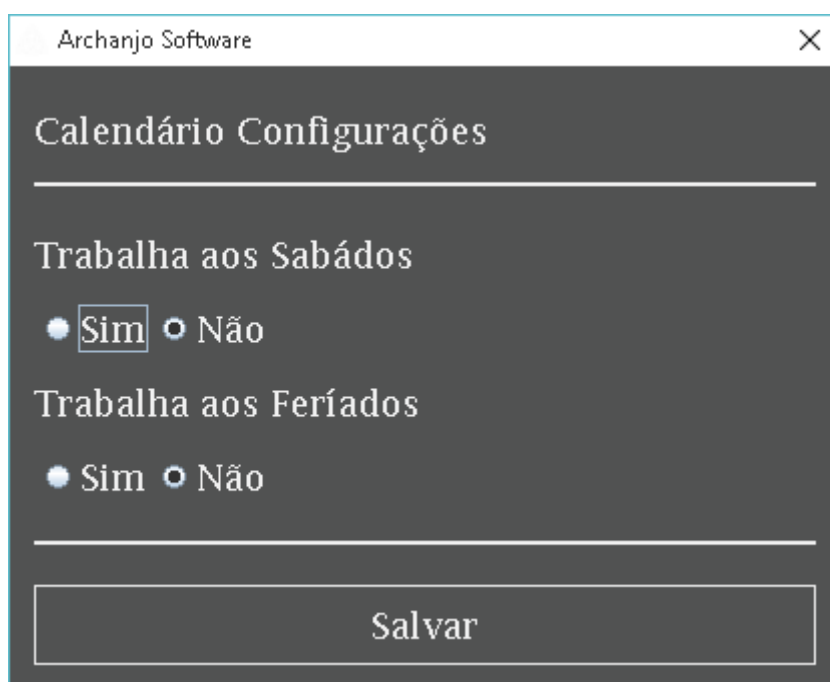
É possível alterar a cor das sprints listadas clicando no ícone de lápis.

É possível configurar o calendário (item 4.6.18), podendo liberar trabalhar aos sábados e aos feriados.

Somente *Scrum Master* e *Product Owner* pode efetuar ações referente há está tela.

#### 4.6.18 Tela Calendário Configurações

**Figura 41: Tela Configuração**



Archanjo Software

### Calendário Configurações

---

Trabalha aos Sabádos

☒ Sim ☐ Não

Trabalha aos Feriados

☐ Sim ☐ Não

---

Salvar

**Fonte:** Autoria do autor.

Está tela permite a configuração da jornada de trabalho, caso a corporação trabalha aos sábados basta selecionar há opção sim e salvar, caso trabalhem nos feriados seleciona a opção sim e salva.

Somente *Scrum Master* e *Product Owner* pode efetuar ações referente há está tela.

#### 4.6.19 Tela Histórico

**Figura 42: Tela Histórico**

Archange Software

Histórico - 07/10/2015

Membro

Membro	Ano	Tarefa	Horário
LURAM TEAM	Início	BC	11:22:39
LURAM TEAM	Início	BG	11:22:42
LURAM TEAM	Início	BA	11:22:44
LURAM TEAM	Concluiu	BG	11:22:47
LURAM TEAM	Concluiu	BD	11:22:51
LURAM TEAM	Concluiu	BC	11:22:53
LURAM TEAM	Concluiu	BF	11:22:55
LURAM TEAM	Concluiu	BB	11:23:02
LURAM TEAM	Concluiu	BA	11:23:05
LURAM TEAM	Início	AD	11:23:10
LURAM TEAM	Início	AB	11:23:12
LURAM TEAM	Início	AJ	11:23:14
LURAM TEAM	Concluiu	AG	11:23:16
LURAM TEAM	Concluiu	AH	11:23:17
LURAM TEAM	Concluiu	AI	11:23:18
LURAM TEAM	Concluiu	AJ	11:23:20
LURAM TEAM	Concluiu	AF	11:23:22
LURAM TEAM	Concluiu	AD	11:23:23
LURAM TEAM	Concluiu	AC	11:23:25
LURAM TEAM	Concluiu	AB	11:23:30

Sair

**Fonte: Autoria do autor.**

Está tela provêm uma listagem de todas as ações referente ao dia selecionado e informações, membro, ação, tarefa, horário.

Está tela permite fazer uma busca precisa basta selecionar umas das opções (Membro, Ação, Tarefa, Horário) localizada no canto superior esquerdo.

É possível ver mais detalhes referente a tarefa desejada, clicando duas vezes na mesma, o software exibirá a tela tarefa (item 4.6.15) com todas as informações referente a mesma.

Somente *Scrum Master* e *Product Owner* pode efetuar ações referente há está tela.

#### 4.6.20 Tela Relatório

**Figura 43: Tela Relatório**



Archanjo Software

## Relatório Geral

Sprint

Tarefa Status

**Fonte:** Autoria do autor.

Está tela provêm gerar um relatório geral com informações referente a sua escolha, tais como sprint e tarefa status.

Figura 44: Relatório

1

Archanjo Software

Relatório Geral

13/10/2015 - 19:46

Filtro Sprint : Todos Filtro Tarefa Status : Todos

Sprint #1

A

Status	Tarefa	Estimativa (Dia)	Iniciada	Finalizada
Concluído	AA	2	07/10/2015	08/10/2015
Concluído	AB	2	07/10/2015	20/10/2015
Concluído	AC	1	07/10/2015	18/11/2015
Concluído	AD	3	07/10/2015	06/10/2015
Concluído	AF	1	07/10/2015	09/11/2015
Concluído	AG	3	07/10/2015	09/10/2015
Concluído	AH	1	07/10/2015	01/11/2015
Concluído	AI	2	07/10/2015	14/10/2015
Concluído	AJ	1	07/10/2015	24/10/2015
Total Tarefa	9	Total Estimativa	16	

B

Status	Tarefa	Estimativa (Dia)	Iniciada	Finalizada
Concluído	BA	1	07/10/2015	14/10/2015
Concluído	BB	2	07/10/2015	26/10/2015
Concluído	BC	4	07/10/2015	13/10/2015
Concluído	BD	3	07/10/2015	07/10/2015
Concluído	BF	2	07/10/2015	19/10/2015
Concluído	BG	2	07/10/2015	21/10/2015
Total Tarefa	6	Total Estimativa	14	
Total Sprint Tarefa	15	Total Sprint Estimativa	30	

Fonte: Autoria do autor.



## CONSIDERAÇÕES FINAIS

O objetivo principal deste trabalho foi desenvolver um software que abrange-se todas as partes da metodologia Scrum, pois as ferramentas pesquisadas somente tinham algumas partes do Scrum tais como, kanban, *product backlog*, *sprint* e algumas são adaptadas para o uso do Scrum. A ferramenta proporciona a oportunidade de juntar todos os componentes do Scrum, *product backlog*, *sprint backlog*, *sprint*, *burndown*, tarefas, kanban e a interação de todos os membros *Product Owner*, *Scrum Master* e *Scrum Team* em somente um software.

Não ocorreram problemas na fase de análise de projeto e programação, porem na contextualização do Scrum, falta de experiência e conhecimento nesta metodologia, tornaram a modelagem do software complexa e trabalhosa, efetuado inúmeros testes, obtive com êxito o objetivo deste trabalho e conhecimento sobre Scrum e pude ver o quão poderoso é está metodologia no quesito gerenciamento de tempo, pois os elementos *burndown*, *kanban*, *sprint* provêm estimativas e acompanhamento do mesmo.

Em sua primeira versão implementou-se todos os componentes Scrum, em possíveis versão pode se implementar controle de equipes, melhores precisões de tempo nas estimativas de tarefas, agendamento de *Sprint Planning* e *Sprint Review* na tela calendário, aprimoramento do kanban adicionando campo de anotação nas tarefas, alocação de demais integrantes para a mesma, exibição de tempo restando para o termino da tarefa, novos relatórios de acompanhamento, tais como um relatório que exiba um *burndown* completo de todas as *sprints*, implementar tutorial, implementar FAQ, implementar o envio de mensagens para o desenvolvedor com possíveis melhorias e erros, implementar um chat online para troca de mensagens entre os integrantes da equipe, implementação de anotações na tala de histórico, implementação de perguntas referente ao *daily scrum* agendada para uma determinada hora e armazenamento do mesmo, para que o *Scrum Master* possa ler e analisar e efetuar possíveis mudanças, possibilitar enviar e-mail dentro do software para os integrantes do software como comunicados, implementar envio automático de e-mail quando uma tarefa estiver atrasada, quando a *sprint* estiver totalmente concluída, quando estiver próximo das reuniões marcadas.

## REFERÊNCIAS BIBLIOGRÁFICAS

BERTALANFFY, L. Von. **Teoria geral dos sistemas**. Petrópolis, Vozes, 1977.

BIO, Sérgio Rodrigues. **Sistema de Informação**: Um enfoque gerencial. São Paulo/SP: Editora Atlas S.A, 1991.

GUEDES, Gilleanes T.A. **UML2**: uma abordagem prática. São Paulp/SP: Novatec, 2009.

PÁDUA PAULA FILHO, Wilson de. **Engenharia de software**: Fundamentos, Métodos e Padrões. 3ºed. Rio de Janeiro/RJ: LTC, 2009.

PRESSMAN, Roger S. **Engenharia de Software**. Trad. José Carlos Barbosa dos Santos. São Paulo/SP: Pearson, 2007.

SOMMERVILLE, Ian. **Engenharia de Software**. Trad. Selma Shin Shimizu Melnikoff. 8ºed. São Paulo/SP: Pearson, 2007.

TELES, Vinícius Manhães. **Manifesto Ágil**. 2008. Disponível em: [http://www.desenvolvimentoagil.com.br/xp/manifesto\\_agil](http://www.desenvolvimentoagil.com.br/xp/manifesto_agil). Acessado em: 18 set. 2015, às 13h39min.

TELES, Vinicius Manhães. **Scrum**. 2015. Disponível em: <http://www.desenvolvimentoagil.com.br/scrum>. Acessado em: 9 jun. 2015, às 18h05min.

VIEIRA, Denisson. **Ferramentas SCRUM**. 2014. Disponível em: <http://www.mindmaster.com.br/ferramentas-scrum>. Acessado em 11 jun. 2015, às 14h54min.