Universidade de Brasília - UnB
Faculdade UnB Gama - FGA
Engenharia de Software

Grata - Gerenciamento de Reuniões e ATAs

Autor: Victor Hugo Lopes Mota
Orientador: Dr. Wander Cleber Maria Pereira da Silva

Brasília, DF
2019

Victor Hugo Lopes Mota

Grata - Gerenciamento de Reuniões e ATAs

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software .

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Dr. Wander Cleber Maria Pereira da Silva

Brasília, DF

2019

Victor Hugo Lopes Mota
Grata - Gerenciamento de Reuniões e ATAs/ Victor Hugo Lopes Mota. – Brasília, DF, 2019-
40 p. : il. (algumas color.) ; 30 cm.

Orientador: Dr. Wander Cleber Maria Pereira da Silva

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2019.

1. Gerenciamento de Reuniões. 2. Otimização. I. Dr. Wander Cleber Maria Pereira da Silva. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Grata - Gerenciamento de Reuniões e ATAs

CDU 02:141:005.6

Victor Hugo Lopes Mota

Grata - Gerenciamento de Reuniões e ATAs

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software .

Trabalho aprovado. Brasília, DF, 14 de abril de 1912:

Dr. Wander Cleber Maria Pereira da Silva
Orientador

Membro convidado 1
Convidado 1

Membro convidado 2
Convidado 2

Brasília, DF
2019

Dedicatória ficara aqui

Agradecimentos

AQUI VÃO FICAR OS AGRADECIMENTOS

Resumo

Resumo do projeto

Palavras-chaves: Gerenciamento de Reuniões, Otimização.

Abstract

Here go to abstract

Key-words: Meeting management,Optimization.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

GRATA, *Gerenciamento de Reuniões e ATAs*

MAS, *Síndrome de Aceitação Sem Sentido*

NMIL, *Núcleo de Modernização da Informação Legislativa*

TCC, *Trabalho de Conclusão de Curso*

MVC, *Model-View-Controller*

RUP, *Rational Unified Process*

SGM, *Secretária Geral da Mesa*

DGER, *Diretoria Geral*

SINFLEG, *Secretaria de Informação Legislativa*

PRODASEN, *Secretaria de Tecnologia da Informação*

TAP, *Termo de Abertura do Projeto*

PO, *Product Owner*

JAD, *Joint Application Development*

Sumário

1 Introdução

Um dos instrumentos mais fundamentais e que são cada vez mais crescentes na vida organizacional de uma empresa são as reuniões. Segundo (ALLEN, 2016), uma instituição utiliza em média 15% do tempo coletivo da organização com reuniões.

1.1 Contexto

Encontros institucionais que são improdutivas e sem sentido, é o que (DAVID, 2013) chama de "Síndrome de Aceitação Sem Sentido"(MAS). David define o MAS como "um reflexo involuntário em que uma pessoa aceita um convite de reunião sem sequer saber o porquê. Uma doença comum entre o escritório e trabalhadores em todo mundo". Atividades que deveriam ser simples e rápidas se tornam complicadas com várias reuniões para a execução completa delas. Reuniões não são nenhum ponto de prazer dentro de uma instituição, contudo se os próprios funcionários não conseguem visualizar o sentido da reunião e os tópicos abordados, mostra que a empresa como um todo está fadada ao fracasso.

1.2 Justificativa

As reuniões são criadas para promover o compartilhamento de informações, melhorar a tomada de decisões, promover a resolução de problemas, construir a coesão da equipe e reforçar a cultura organizacional (LEACH, 2015). Por se tratar de um encontro dentro da empresa, segundo (LEACH, 2015), as reuniões podem gerar emoções tanto positivas quanto negativas dentre os participantes da reunião. É possível sair de uma reunião sentindo-se energizado e inspirado, ou afastar-se de uma reunião sentindo-se esgotado e desmoralizado.

(MACLEOD, 2011) estimou que entre 30% a 60% do tempo gasto em uma reunião é desperdiçado. Um gerente pode passar onze horas por semanas em reuniões e metade desse tempo é improdutivo. Ao realizar uma pesquisa sobre a produtividade das reuniões, (PERLOW, 2017) constatou que 65% dos gerentes entrevistados indicaram que reuniões os impedem de realizar suas próprias atividades no trabalho e 71% alegam que as reuniões são improdutivas e ineficazes. Uma reunião pode render comentários positivos e negativos, contudo segundo (LEACH, 2015), os comentários negativos são os mais pertinentes e são relacionados a estrutura da reunião. Problemas como falta de planejamento, informações de baixa relevância e impacto pouco claro de participação são os fatores que mais compõem negativamente uma reunião.

(ROGELBERG, 2005) examinou as reuniões a partir de duas teorias: capacidade atencional e teoria da ação. Ao aplicar essas práticas, foi constatado que a fadiga diária e a carga de trabalho subjetiva estão relacionadas diretamente as reuniões atendidas. O estudo sugere ainda que a frequência de reuniões é mais importante do que o tempo gasto em reuniões ao longo do dia. Ainda segundo (ROGELBERG, 2005), a "natureza disruptiva dos resultados das reuniões na drenagem recursos emocionais ou mentais e fadiga subsequente". A conclusão do estudo foi que tanto a qualidade quanto a quantidade das reuniões são importantes para o bem-estar do funcionário.

As reuniões por mais importantes que sejam, não apresentam de fato o verdadeiro sentido que elas possuem. Como apresentado nos tópicos anteriores, estudos nessa área apresentam diversos problemas nas reuniões com níveis de improdutividade elevados, contudo ao analisar esses estudos é possivel elaborar uma solução computacional que auxilie gerentes e funcionários a não gastarem tanto tempo em reuniões e atingir um maior nível de produtividade. A proposta computacional visa aumentar o engajamento dos participantes, aumentar no planejamento das reuniões, informações com maiores níveis de relevância e por fim diminuir os altos níveis de improdutividade apresentados pelos autores nos tópicos anteriores.

Ao realizar uma pesquisa de mercado, foi encontrado quatro *softwares* semelhantes ao proposto nesse projeto, sendo eles:

- *Meeting* (gestão de projetos)
- Qualiex
- Eata
- *Google calendar*

Desses quatro *softwares*, os três primeiros são pagos, contendo uma versão trial de 30 dias. O *Google calendar* é um *software* gratuito, contudo não é tão completo os outros três.

1.3 Problema de Pesquisa

O crescente problema com reuniões mal gerenciadas seja por gerentes não capacitados, ou por falta da especificação prévia dos tópicos a serem abordados, levam diretamente a reuniões mal sucessididas e com isso desperdício de tempo e dinheiro. Estima-se que empresas gastam em média US $ 37 bilhões anualmente em reuniões (DRAKE, 2014). O custo real desses encontros impulsionou a (HARVARD, 2016) a criar uma calculadora que ajuda gerentes a calcularem o verdadeiro custo de um encontro.

Nessa viés e utilizando a justificativa desenvolvida no tópico 1.2, é possível se ter o problema de pesquisa. A problemática a ser resolvida neste projeto é: *Como desenvolver um sistema para auxiliar a condução de reuniões em organizações?*

1.4 Objetivos

1.4.1 Objetivos Gerais

Os objetivos do projeto são de propor novos processos de reuniões e gerenciamento dos documentos gerados no ciclo de desenvolvimento dos projetos, contando com o suporte de um *software* gratuito e de código aberto para automatizar trabalhos manuais, tornando os processos mais ágeis e com armazenamentos seguros.

1.4.2 Objetivos Específicos

- Criar um *software* que auxilie gerentes e líderes a terem reuniões mais objetivas;
- implementar mecanismos que permita o controle gerencial das reuniões;
- permitir o controle de todas as informações provindas das reuniões;
- desenvolver um *software* de código aberto, gratuito e que atenda a demanda de gerenciar as reuniões.

2 Fundamentação Teórica

2.1 Gerenciamento de Projetos

2.1.1 Modelo Tradicional

Uma metodologia de gerenciamento de projetos no modelo tradicional, de acordo com (KERZNER, 2009) é o alcance da excelência no gerenciamento de projetos se torna impossível sem um processo repetitivo que possa ser utilizado em cada projeto.

No modelo tradicional, um dos mais modelos mais utilizados é o RUP *(Rational Unified Process)*. O RUP oferece uma metodologia responsável por responder questões como boas práticas para o gerenciamento de projetos, com o objetivo de estruturar e formatar os processos associados às atividades que envolvem a tecnologia de informação.

As 4 fases principais do RUP podem ser vistas na figura 1:

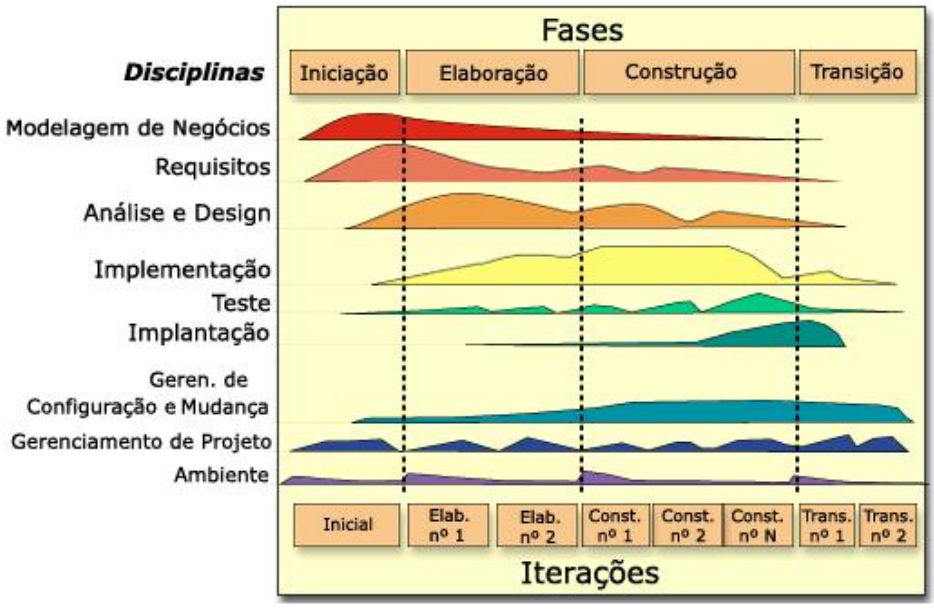

Figura 1 – Fases do RUP. Fonte: (RUP, 1980).

O desenvolvimento do plano de gerenciamento do projeto é uma atividade iterativa ao longo do ciclo de vida do projeto, sempre pronto para melhoria contínua e permitindo à equipes do projeto definir e trabalhar com maior nível de detalhes. De acordo com o (PMBOK, 2012), as fases do (RUP, 1980) são sobrepostas, ou seja, o início de uma fase é ao término de uma outra, isso leva a algumas atividades ocorrerem de forma paralela. A maneira como este tipo de projeto aumenta os riscos, retrabalhos, e exigir recursos adicionais para permitir as atividades em paralelo, como mostrado na figura 1.

Nesta perspectiva, se tem o papel do gerente de projeto como um líder responsável por liderar a equipe para alcançar os objetivos previstos no planejamento do projeto. Entre as funções destes líderes se tem:

- Conhecimento acerca do gerenciamento de projetos;
- Desempenho para aplicar seus conhecimentos na prática;
- Comportamento pessoal de liderança, atingindo objetivos e equilibrando restrições.

Este tipo de gerenciamento de projeto é mais utilizado em empresas já consolidadas, de ramo mais formal, que possui mais burocracia em seus projetos e por tanto maior rigor de documentação e de liderança dos gerentes de projeto. Essa hierarquia pode ser vista na Figura 2.

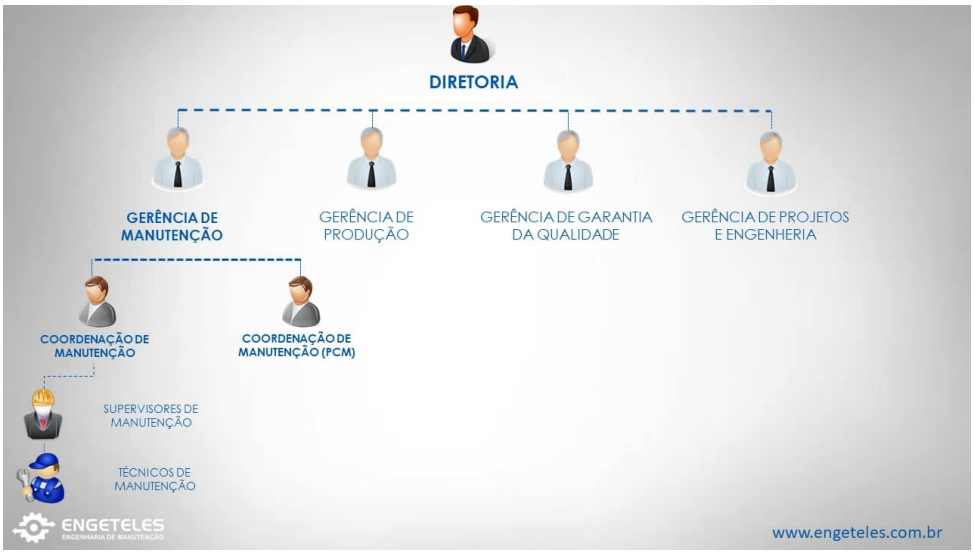

Figura 2 – Hierarquia em projetos tradicionais. Fonte: (JHONATAS, 2018).

2.1.1.1 Casos de Uso

Para o auxilio no desenvolvimento, o RUP utiliza um diagrama conhecido como Casos de Uso. Segundo (VIEIRA, 2015), este modelo é focado em auxiliar a comunicação entre cliente e analistas, apresentar as principais funcionalidades do sistema com foco no cliente e é muito utilizado na fase de levantamento dos requisitos.

Um caso de uso é composto por atores e relacionamentos. Na Figura 3 é possível visualizar um exemplo de caso de uso:

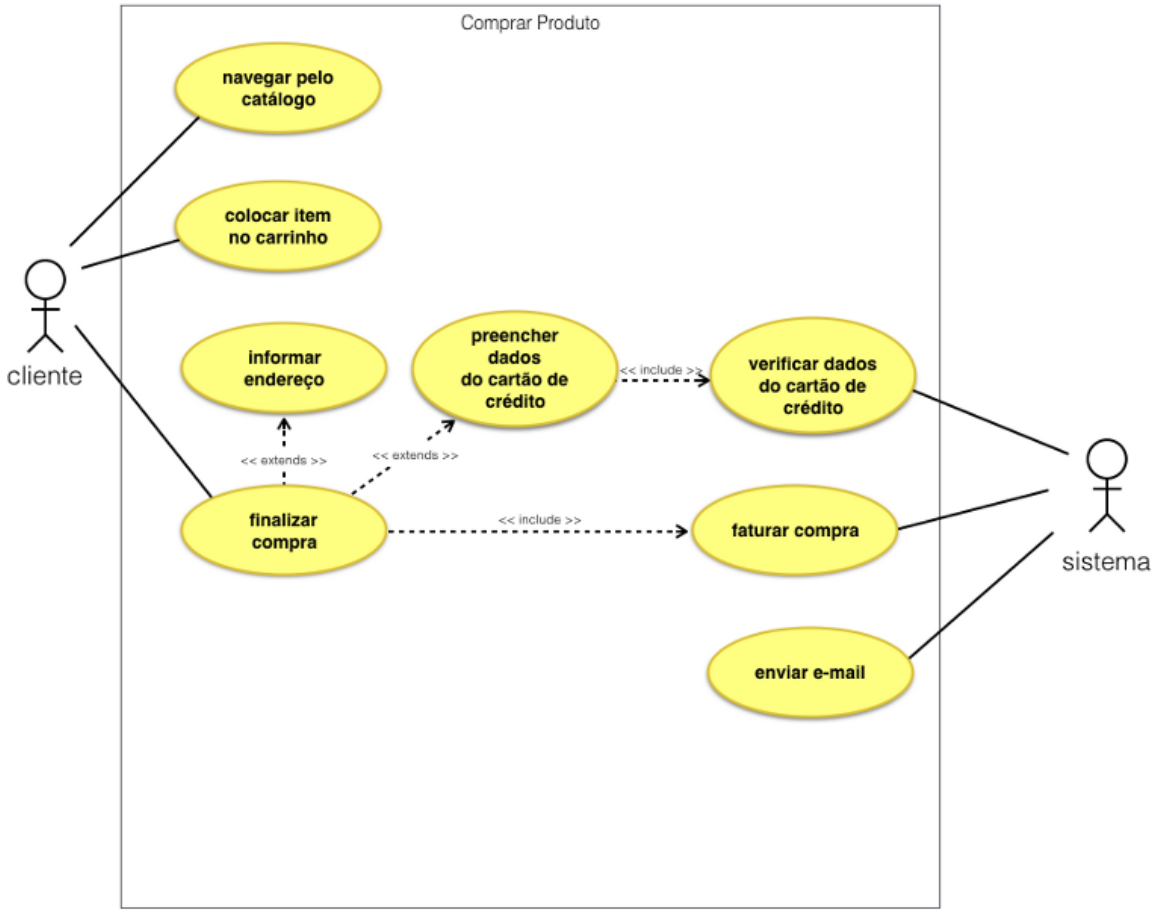

Figura 3 – Exemplo de Caso de Uso. Fonte: (VIEIRA, 2015).

2.1.2 Modelo Ágil

Em contraposição ao modelo tradicional, surte o manifesto ágil como uma reação contra o processo burocrático presente no modelo tradicional, que possuem por característica atividades sequências em modelo cascata. Segundo a (STANDISH, 2014) apenas 16,2% dos projetos entregues por companhias americanas foram entregues respeitando prazos, custos previamente acordados e objetivos determinados. Segundo a própria (STANDISH, 2014), as principais causas destes problemas estavam relacionadas com o modelo sequencial tradicional.

O modelo ágil, segundo (SOARES, 2009), ela deve primeiro aceitar as mudanças em vez de tentar prevê-las, agir de maneira rápida sabendo receber, avaliar e responder como elas devem ser respondidas. As principais características da metodologia ágil são:

- Desenvolvimento iterativo e incremental;
- Comunicação;

- Documentação extensiva;

Em 2001, membros da comunidade de *software* se reuniram e criaram o (AGILE, 2001). O objetivo deste manifesto é utilizar as melhores práticas observadas em projetos anteriores que obtiveram sucessos.

Os principais conceitos do manifesto ágil são:

- Indivíduos e interações ao invés de processos e ferramentas;
- *Software* executável ao invés de documentação;
- Colaboração do cliente ao invés de negociação de contratos;
- Resposta rápida a mudanças ao invés de seguir planos pré-estabelecidos.

No modelo ágil os requisitos dos clientes podem ser mudados a qualquer momento, e o time de gerência e desenvolvimento devem estar preparados para conversar com o cliente a fim de resolver as alterações de requisitos da melhor maneira possível. Este tipo de pensamento no modelo tradicional é mais difícil de acontecer, pois ao observar a figura 1, é possível notar ao iniciar uma fase, essa mesma fase não é retornada mais tarde, ou seja, no modelo tradicional uma troca de requisitos pode levar ao reinicio do projeto.

Este modelo é mais focado para empresas emergentes, que não são muito rigorosas em seus processos e aceita que mudanças nos requisitos ou na visão do produto são sempre bem vindas, desde que melhore o projeto final.

2.1.2.1 Scrum

Uma das boas práticas adotadas ao modelo ágil é o *Scrum*. O *Scrum*, é um *framework* que se refere ao jogo *Rugby*, que é a ação dos jogadores se organizarem em círculo para planejar a próxima jogada. Um dos principais pontos de vista do *Srum* é mostrar um projeto com pequenos ciclos, aumentando as iterações entre os participantes, mas com visão a longo prazo.

O ciclo de vida pode ser visto na figura 4:

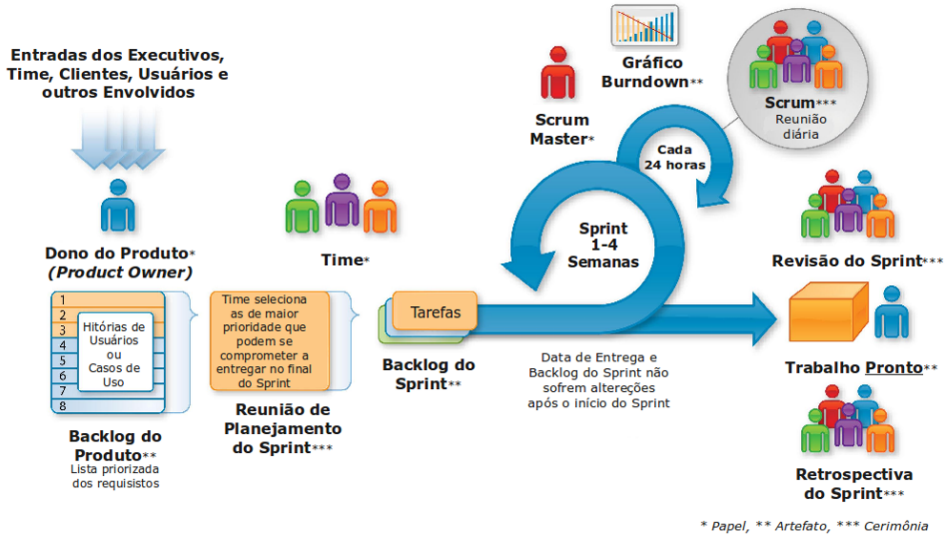

Figura 4 – Ciclo de Vida SCRUM. Fonte: (FABIANE, 2016).

Como visto na figura 4, o *Scrum* é um ciclo progressivo de várias iterações bem definidas, denominadas *Sprints*. As *Sprints* podem ter duração de uma a quatro semanas. Antes de cada *Sprint*, deve ser realizada a reunião de planejamento da *Sprint*, chamada de *Sprint Planning Meeting*. A *Sprint Planning Meeting* é uma reunião de planjemanto em que o *Product Owner* prioriza os itens do *Product Backlog* e a equipe seleciona as atividades que serão implementadas ao longo da *Sprint*. No *Product Backlog* são registradas as funcionalidades que serão implementadas pedidas pelo *Product Backlog*.

Com o objetivo de saber o progresso de cada equipe dentro da *Sprint*, ocorrem as reuniões diárias, denominadas *Daily Meetings*, que tem duração de no máximo 15 minutos e ocorrem com todos os participantes em pé, respondendo perguntas como: "O que você fez ontem?", "O que você fez hoje?"e "O que você vai fazer amanhã?".

Ao final de uma *Sprint* é feita uma análise gráfico do progresso do projeto atráves do *Sprint Backlog* durante a *Sprint Review*. Após a *Sprint Review* ocorre a *Sprint Retrospective* que é a análise de experiências que ocorreram durante a *Sprint* sejam boas ou não a fim de melhora-las.

Segundo (FOWLER, 2005), as equipes devem possuir um quadro para registro das atividades, denominado *Kanban*. O *Kanban* possui o objetivo de auxiliar as equipes em relação ao progresso da *Sprint*, esse quadro pode ser dividido em 4 fases:

- Para fazer;
- Em andamento (com o nome do responsável pela atividade);
- Em revisão;
- Feito.

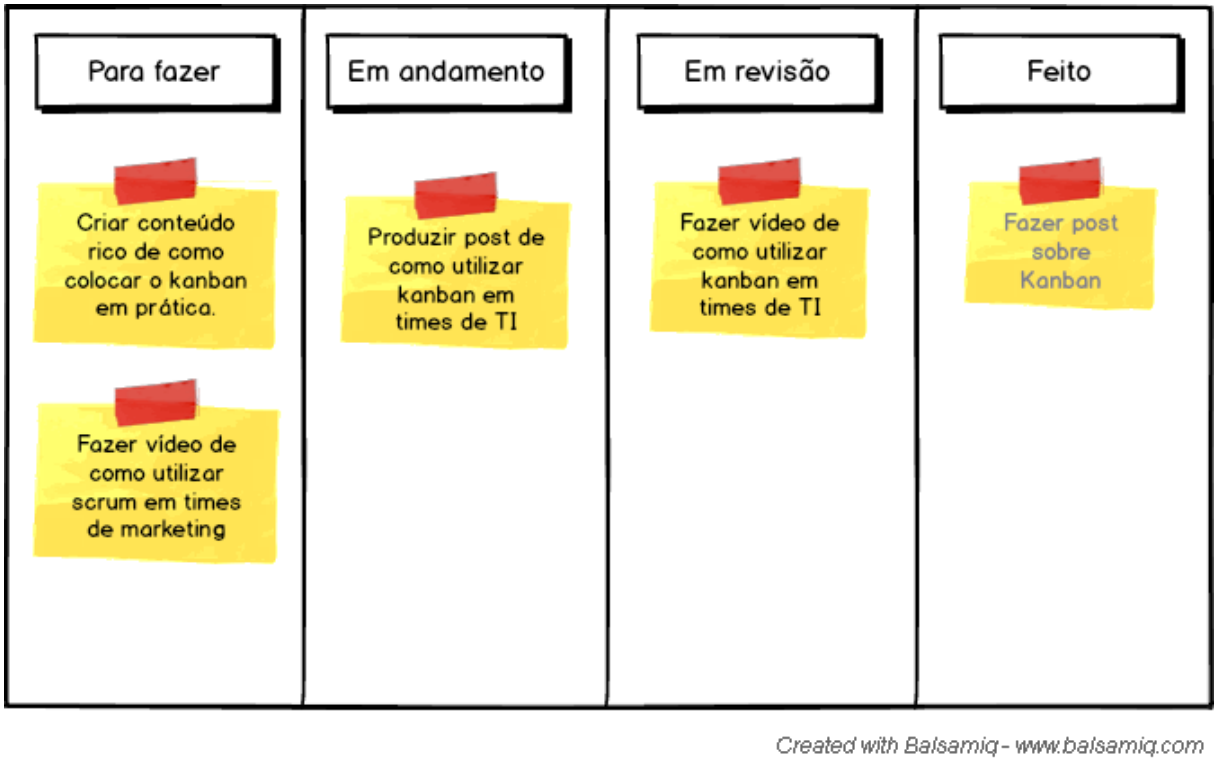

Figura 5 – Quadro Kanban. Fonte: (LAMECK, 2016).

O *Scrum* possui seus papéis bem definidos, podendo ser alterados ao longo do desenvolvimento do projeto. Esses papéis podem ser vistos na figura 6.

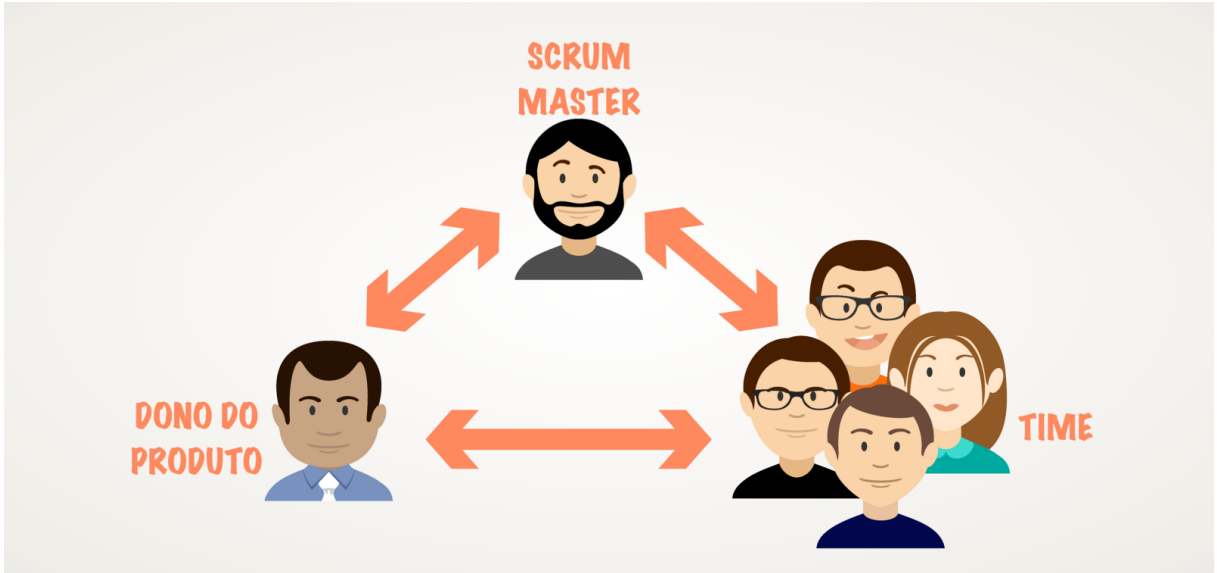

Figura 6 – Papéis Scrum. Fonte: (GONçALVEZ, 2016).

Como observado na figura 6, o *Scrum* possui três papéis bem definidos: o *Product*

Owner, conhecido como PO, o *Scrum Master* e por fim temos o *Dev Team*.

2.1.2.1.1 Product Owner

O *Product Owner*, é o dono do produto. É ele que fornece o conhecimento do negócio em forma de requisitos para a equipe e na forma pratíca são os patrocionadores/-clientes do produto. O PO organiza e prioriza o *Product Backlog* (que são os itens que devem ser desenvolvidos), esse papel deve ser assumido por pessoas que sejam boas em se comunicar, pois esse papel é responsável por trabalhar e tirar dúvidas da *Dev Team*.

2.1.2.1.2 Scrum Master

Ao analisar a figura 6, a figura do *Scrum Master* parece ser superior aos outros papéis, o que não é verdade. O *Scrum Master* possui o dever de ajudar a comunicação entre o PO e o *Dev Team* além de remover todos os impedimentos que estão prejudicando o desenvolvimento, tem a função de auxiliar o amadurecimento da *Dev Team* e promover as cerimônias que o *Scrum* preza, como *Daily Meetings*, *Sprint Review* e *Sprint Retrospective*.

2.1.2.1.3 Dev Team

Esse papel é voltada para as pessoas que de fato irão desenvolver o produto. O *Dev Team* é auto-organizável e decide entre si como implementar os itens priorizados pelo PO.

2.1.3 Processo de Desenvolvimento de Software

Segundo (SOMMERVILLE, 2011), esse processo pode ser definido como "Um processo de *software* é um conjunto de atividades relacionadas que levam à produção de um produto de *software*."

Neste trabalho, foram definidas as principais atividades a serem realizadas para alcançar o objetivo final de ter um *software* gratuito,código aberto e que auxilie os gerentes a otimizar suas reuniões por meio computacional são:

- Especificação do *software*: funcionalidades e restrições do *software*;
- Projeto e implementação do *software*: as especificações que o *software* deve atender;
- Validação de *software*: para que atenda as expectativas do cliente, o *software* deve ser validado pelo mesmo;
- Evolução do *software*: o *software* deve ser capaz de ser extensível a mudanças, tendo assim seu código aberto.

2.1.4 Linguagem de Software

Linguagem de programação são instruções passadas de maneira que o computador entenda e apresente um retorno. Existem diversas linguagens de programação, desde a mais baixo a alto nível.

Linguagens de *software*, como também podem ser chamadas, são divididas em duas frentes: *front-end* e *back-end*. Ambas serão explicadas nos tópicos a seguir.

2.1.4.1 Front-end

A programação de um *software* pelo ponto de vista do *front-end* é a visão final do usuário com o sistema. *Front-end* é a responsável pela interação do usuário com o sistema e essa interação é dada a partir de telas/páginas. Existem diversos tipos de *frameworks* que auxiliam os desenvolvedores a trabalhar com essa frente, como:

- *Bootstrap*
- *Materialize*
- *React*
- *Angular 4*

2.1.5 Back-end

A programação *back-end* possui as responsabilidades de receber os dados pelo *React*, que é o *front-end* deste projeto, possui o dever de tratar os dados, valida-los e fomentá-los a visão do usuário.

Existem diversas linguagens *back-end* que auxiliam os desenvolvedores a trabalhar em uma linguagem que o computador entende, como:

- *Python Django-Rest*
- *Java*
- *Ruby on Rails*
- *PHP*

2.1.6 Arquitetura de Software

A arquitetura de *software* é como o sistema deve ser organizado com a estrutura geral do projeto. A arquitetura possui um valor alto dentro da construção de um *software*, pois nela se tem o elo entre o projeto e a engenharia de requisitos. Possui o dever identificar os principais componentes estruturais no sistema e o relacionamento entre eles.

2.1.6.1 Model-View-Controller

O padrão arquitetural MVC é responsável de responsabilidades em camadas. A primeira é *Model*(modelo), que é responsável pela manipulação de dados, ou seja, leitura, escrita de dados e também suas validações é de responsabilidade da Model. A segunda camada é a *View*(visão), que possui a responsabilidade de interação com o usuário. Por último se tem a *Controller*(controladora), responsável por receber as aquisições do usuário. A controller também tem o dever de disponibilizar os dados para a *view* e assim ocorrer a interação com o usuário.

2.1.7 Definição dos Requisitos

Requisito não é um termo usado apenas pela Engenharia de Software . Há casos em que requisitos são apenas uma declaração abstrata em alto nível de um serviço ou restrição que um sistema deve oferecer.

(SOMMERVILLE, 2011) os define como: "Os requisitos de um sistema são as descrições do que o sistema deve fazer, os serviços que oferece e as restrições a seu funcionamento. Esses requisitos refletem as necessidades dos clientes para um sistema que serve a uma finalidade determinada, como controlar um dispositivo, colocar um pedido ou encontrar informações."

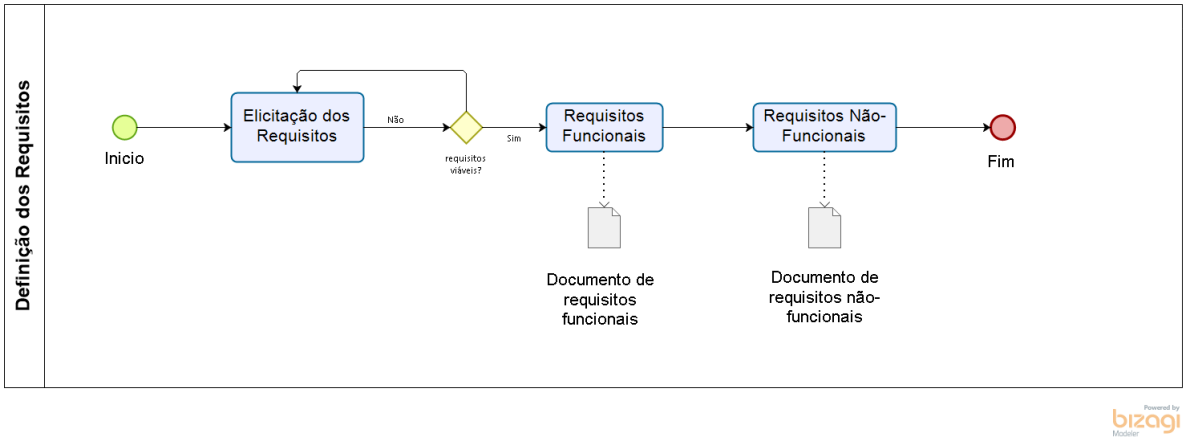

Figura 7 – Definição dos Requisitos. Fonte: Própria.

2.1.7.1 Elicitação dos Requisitos

(SOMMERVILLE, 2011) define a elicitação de requisitos como uma fase do projeto onde são extraídas informações do cliente sobre o que ele deseja que seja construído. É a fase em que o profissional de TI entende a necessidade do cliente e o orienta. É o momento de conversa com o usuário, de sentimento sobre o que este espera que seja entregue a ele.

Na elicitação de requisitos são percebidas as necessidades do sistema e as características que esse sistema deve ter.

Para este projeto, foram escolhidas as seguites técnicas de elicitação de requisitos:

- JAD - *Joint Application Development*
- Entrevista
- Observação

2.1.7.1.1 JAD - Joint Application Development

2.1.7.1.2 Entrevista

2.1.7.1.3 Observação

2.1.7.2 Requisitos Funcionais

Os requisitos funcionais descreve o que o sistema deve de fato ser. Requisitos funcionais podem ser tão específicos quanto necessário,por exemplo, podem ter sistemas com requisitos funcionais gerais e outros que além de refletir os sistemas, também abrangem as formas de trabalho de uma organização. Requisitos funcionais de um sistema deve ser completo, isso quer dizer que todos os serviços requisitados pelo usuário devem ser definidos.

2.1.7.3 Requisitos Não-Funcionais

Requisitos não-funcionais são requisitos que são relacionados as propriedades do sistema como confiabilidade, tempo de espera, desempenho, segurança e até restrições do sistema. Requisitos não-funcionais podem possui tanta relevância quanto os requisitos funcionais, pois em uma reunião de levantamento de requisitos, o cliente sonha o mundo e não está atento se os recursos os próprios recursos e os recursos da emprega conseguem atender ao requisito. Um requisito não-funcional não atendido pode inclusive inutilizar um projeto. Exemplo disso é caso um sistema de uma aeronave não consiga atingir a confiabilidade necessária, não será dado o certificado de segurança para operar, sendo assim a aeronave não poderá voar.

2.2 A Instituição

Tendo a justificativa para o projeto no tópico 1.2, seguida do problema de pesquisa (1.3) e os objetivos descritos no tópico 1.4, se tem a necessidade de escolher alguma empresa que será usada como caso de estudo para o projeto, no caso foi definido o NMIL

(Núcleo de Modernização da Informação Legislativa), um setor localizado no Senado Federal Brasileiro.

2.2.1 Senado Federal

As funções do Senado Federal são exercidas pelos senadores da República, que são eleitos segundo o princípio majoritário para representarem os estados e o Distrito Federal. Cada estado e o Distrito Federal elegem três senadores para um mandato de oito anos. A renovação da representação se dá a cada quatro anos, alternadamente, por um e dois terços. Cada senador é eleito com dois suplentes.

A Estrutura Administrativa compreende a formação das unidades do Senado, suas atribuições, responsáveis e formas de contato.

A Administração tem como ênfase os compromissos com o Parlamento; com excelência na prestação de serviços públicos; com qualidade de vida dos colaboradores; com a igualdade; com a livre disseminação de ideias; com a transparência; com a responsabilidade na utilização de recursos públicos; com a ustentabilidade; com a acessibilidade; com a memória do Senado; e com a comunidade. Na figura 8 pode ser visualizado o organograma organizacional do Senado Federal com suas casas e secretárias.

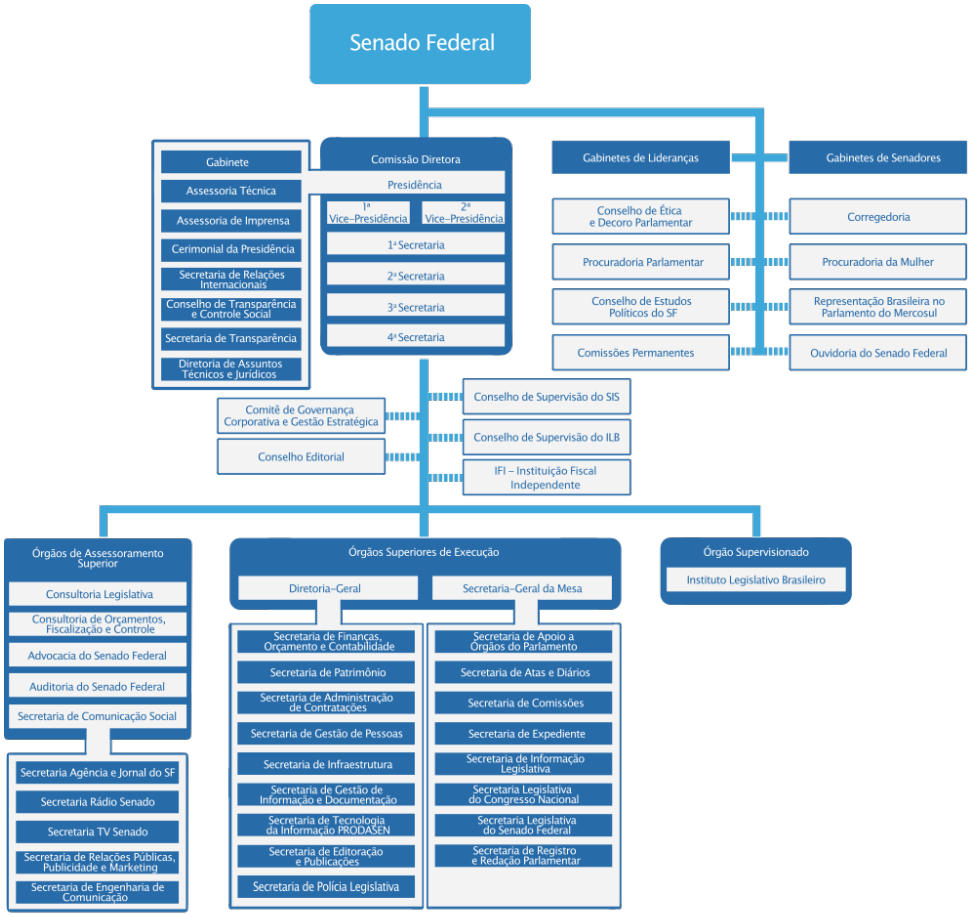

Figura 8 – Organograma Senado Federal. Fonte: (SENADO, 2019).

2.2.2 NMIL

A Comissão Diretora é composta pelo Presidente, dois Vice-Presidentes e quatro Secretários. A composição muda a cada dois anos, correspondentes a uma legislatura. É de responsabilidade da Comissão a direção da casa, designando atividades às unidades que dão suporte.

Essas unidades são: Secretaria Geral da Mesa (SGM), representante da atividade fim da casa; e Diretoria Geral (DGER), que, representa as atividades meio da casa. As duas contam com secretarias, às quais delegam atividades exigidas pelo Presidente.

Quando o Presidente da Comissão Diretora necessita de apoio tecnológico, delega esta atividade à SGM, que, ao receber o problema, começa a definir diretrizes estratégicas para a solução do problema. Após o término da definição das diretrizes, encaminha-as à Secretaria de Informação Legislativa (Sinfleg).

O diretor da Sinfleg atua como Gerente do Projeto, e conta com o apoio da equipe

do NMIL na administração do projeto.

A equipe do NMIL realiza reuniões com as áreas afetadas pelo projeto até conseguir definir todos os requisitos para o produto que será gerado, após a definição, convoca uma reunião com o Gerente para entrega dos requisitos definidos. O Gerente analisa esses requisitos para saber se são viáveis. Caso não sejam, pede ao NMIL novos requisitos e, só após receber requisitos viáveis, aprova a proposta de solução.

O próximo passo é dado pelo NMIL, convocando reunião com a Secretaria de Tecnologia da Informação (Prodasen). Nesta reunião discute-se os requisitos aprovados e prepara-se o Termo de Abertura do Projeto (TAP). O TAP, deve ser encaminhado pelo NMIL ao Gerente para que este aprove o documento; caso não aprove, pede-se um novo, até que seja aprovado.

Após o Gerente aprovar o projeto, ele o apresenta ao Secretário Geral da Mesa, que é o representante da SGM, para uma aprovação final. O Secretário Geral da Mesa também pode pedir um novo projeto, mas se não for o caso, apenas o autoriza.

Dada a aprovação do Secretário Geral da Mesa, a equipe do Prodasen, responsável pela construção do produto, dá início à construção do produto, fazendo as entregas de ambiente de homologação (definidas no TAP) ao NMIL, a fim de que este realize testes. Se forem encontrados erros, estes são listados e repassados ao Prodasen para que sejam reparados. Quando não há mais erros, o NMIL dá sua aprovação do produto. Em seguida o Prodasen termina sua parte do projeto e entrega o produto finalizado ao NMIL. O NMIL encaminha o produto ao Gerente que autoriza o produto e apresenta-o ao Secretário Geral da Mesa.

O Secretário Geral da Mesa, após receber o produto, pode solicitar alterações ao NMIL, que em seguida encaminha esta solicitação ao Prodasen. A equipe do Prodasen responsável pelo produto faz as alterações necessárias o encaminha de volta ao NMIL, passando pelo processo de teste e aprovação novamente até que o Secretário Geral da Mesa autorize a implantação.

Quando o Secretário Geral da Mesa autorizar a implantação, cabe ao NMIL apresentar aos usuários o novo Sistema ou as atualizações em sistemas já existentes.

As figuras relacionadas ao mapeamento do processo que ocorre atualmente no NMIL, pode ser vistos no apêndice A, sendo as figura 12 e 13 como o processo de pedido da comissão diretora para um novo sistema passando pela SGM, Sinfleg, NMIL e por fim ao Prodasen. As figuras 14 e 15 se referem ao processo que o NMIL passa até conseguir atingir um sistema estável e que atenda ao pedida da comissão diretora.

3 Metodologia

3.1 Metodologia de Desenvolvimento

Por conta de possuir um maior conhecimento sobre a metodologia e pela proposta em entregas mais frequentes em períodos menores, foi escolhida a metodologia ágil, explicada no tópico 2.1.2 como metodologia de desenvolvimento do *software* juntamente com algumas práticas do *Scrum*.

3.1.1 Scrum

Uma das principais vantagens do *Scrum* é a adaptação dele a projetos menores e que não são rigorosos a processos, e após ser feita uma análise dos tipos de gerenciamento de projetos no tópicos 2.1, foi escolhido esse *framework* como metodologia de desenvolvimento deste TCC.

3.1.1.1 Papéis

Este projeto será desenvolvido de forma individual, os papéis do *framework Scrum* foram distribuídos de forma com que cada ator tenha a seguinte responsabilidade: o responsável pelo desenvolvido do sistema, Victor Mota, assume os papéis de Time de Desenvolvimento e *Scrum Master*. O papel de *Product Owner* será assumido por Pedro Marques, servidor do Senado Federal e do NMIL, que é o setor de caso de estudo deste trabalho.

3.1.1.2 Sprints

Para este projeto, as *Sprints* foram definidas com duração máxima de duas semanas. Assim como define no *Scrum*, as *Sprints* devem ter as atividades de planejamento e revisão da mesma, para que seja constatado se o que foi planejado foi entregue ao longo das duas semanas.

- *Sprint Planning*: Esta atividade é realizada no primeiro dia de *Sprint* e é nela que são selecionados os itens do *Product Backlog* que serão desenvolvidos ao longo da *Sprint*;
- *Sprint Review*: Esta atividade é realizada no último dia de *Sprint* e é nela são discutidas com *Product Owner* as histórias de usuário desenvolvidas ao longo da *Sprint*.

4 Processo de Desenvolvimento de Software

Como definido o tópico 2.1.3, o processo de desenvolvimento de *software* deste projeto pode ser visualizado na Figura 9. A descrição das atividades do processo serão definididas a seguir:

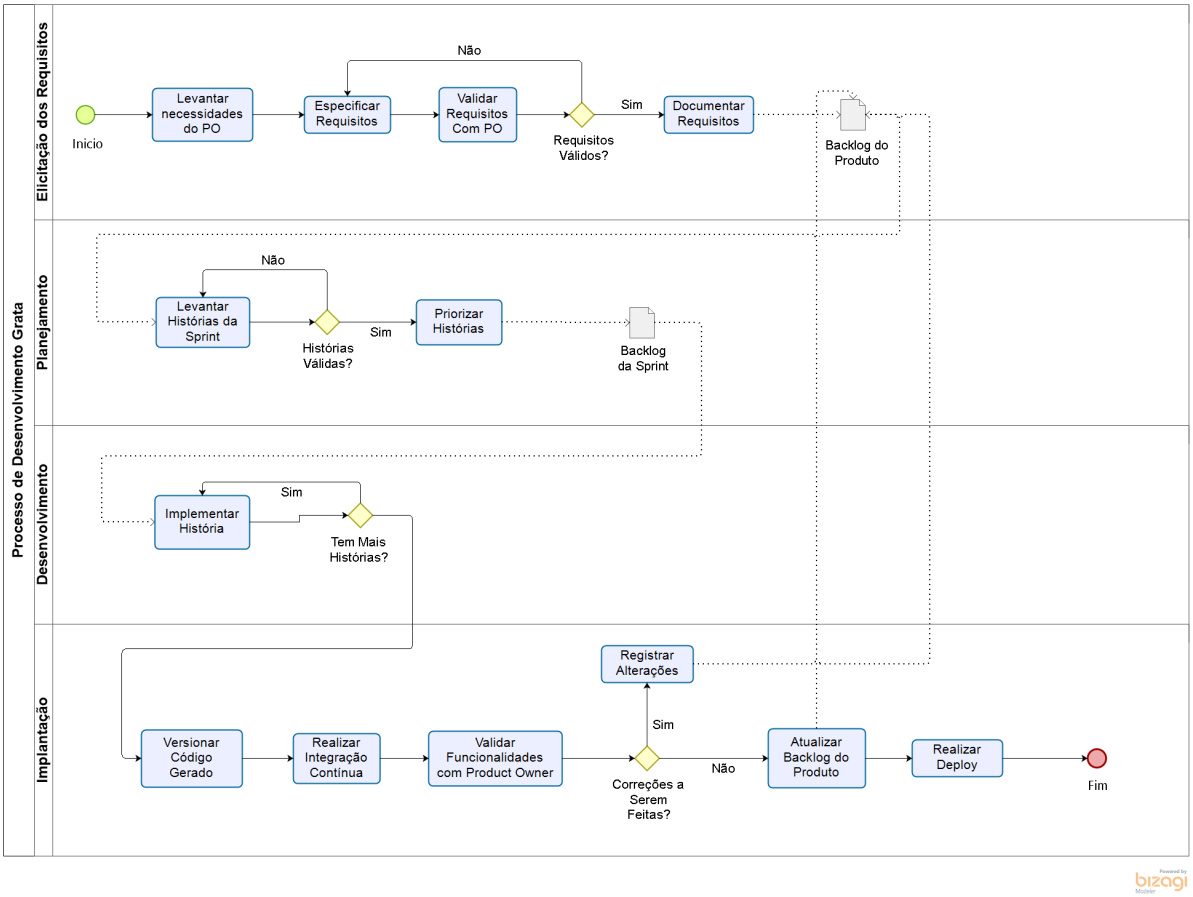

Figura 9 – Processo de Desenvolvimento do Grata

A seguir será mostrado como funciona separadamente o *React*,relacionado ao *front-end* que engloba a *View*, enquanto o *Python Django-Rest* é responsável pelo *back-end* e que engloba a *Model* e a *Controller*:

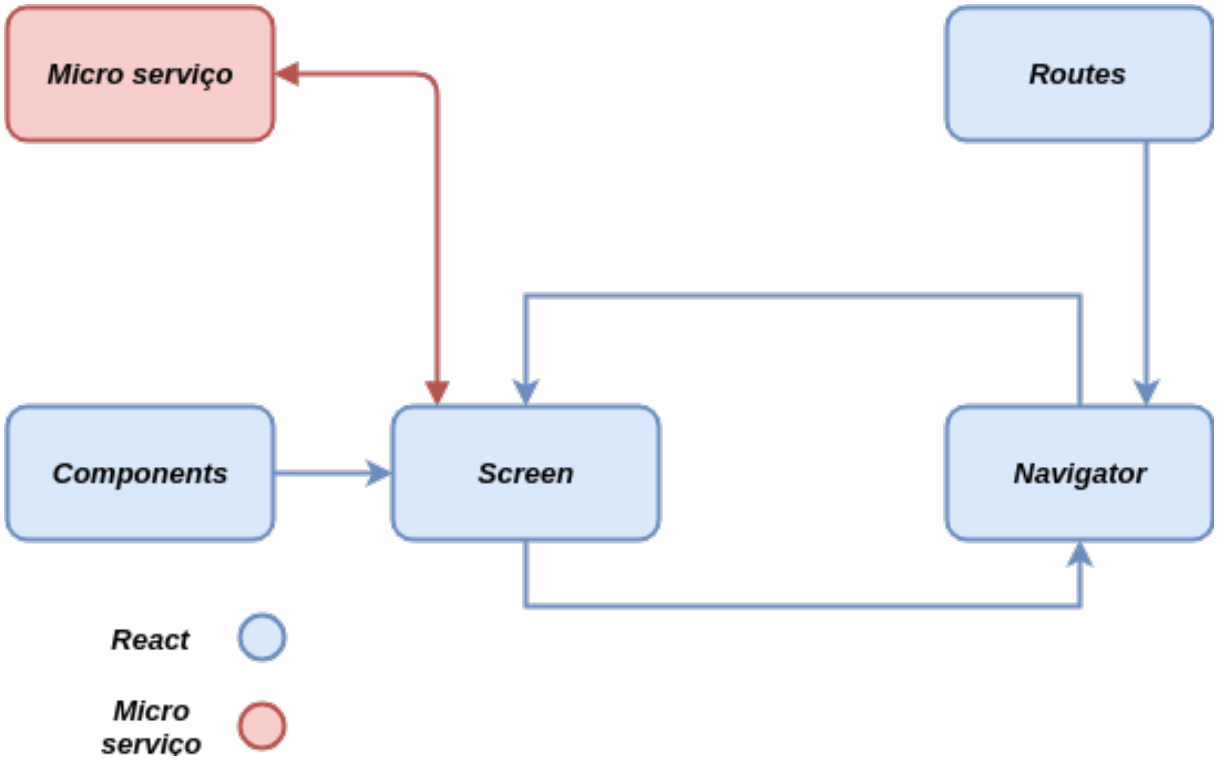

Figura 10 – Diagrama React/Microsserviços

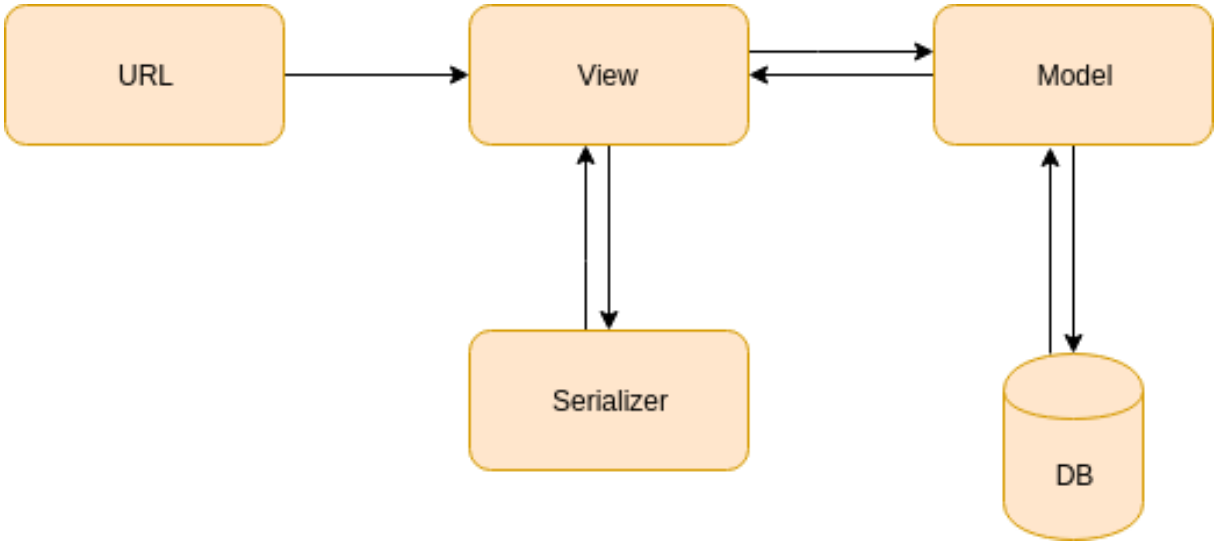

Figura 11 – Diagrama Django REST Framework

4.1 Cronograma TCC

5 Proposta de Solução

6 Conclusão

Referências

AGILE, M. *Manifesto for Agile Software Development*. 2001. Disponível em: <http://www.agilemanifesto.org>. Acesso em: 05.05.2019. Citado na página 19.

ALLEN, L.-W. Meetings as a positive boost? how and when meeting satisfaction impacts employee empowerment. *Journal of Business Research*, 2016. Acesso em: 25.04.2019. Citado na página 13.

DAVID, G. *How to save the world (or at least yourself) from bad meetings*. 2013. Disponível em: <https://www.ted.com/talks/david_grady_how_to_save_the_world_or_at_least_yourself_from_bad_meetings>. Acesso em: 22.04.2019. Citado na página 13.

DRAKE, B. 37 billion is lost every year on these 12 meeting mistakes. *Business Insider*, 2014. Disponível em: <https://www.businessinsider.com/37-billion-is-lost-every-year-on-these-meeting-mistakes-2014-4>. Acesso em: 29.04.2019. Citado na página 14.

FABIANE, S. *Ciclo de Vida do Scrum*. 2016. Disponível em: <https://br.pinterest.com/pin/417357090459098830/>. Acesso em: 05.05.2019. Citado 2 vezes nas páginas 8 e 20.

FOWLER, M. The new methodology. 2005. Disponível em: <https://moodle2016-17.ua.es/moodle/pluginfile.php/69142/mod_resource/content/1/martin-fowler-the-new-methodology.pdf>. Acesso em: 05.04.2019. Citado na página 20.

GONçALVEZ, G. *Papeis Scrum*. 2016. Disponível em: <https://guildadocodigo.atelie.software/como-cada-um-dos-pap%C3%A9is-do-scrum-contribui-para-o-sucesso-do-seu-projeto-b6e8b5f01e57>. Acesso em: 03.06.2019. Citado 2 vezes nas páginas 8 e 21.

HARVARD, B. *Estimate the Cost of a Meeting with This Calculator*. 2016. Disponível em: <https://hbr.org/2016/01/estimate-the-cost-of-a-meeting-with-this-calculator>. Acesso em: 29.04.2019. Citado na página 14.

JHONATAS, T. *PCM Descomplicado – Planejamento e Controle de Manutenção*. 2018. Disponível em: <https://engeteles.com.br/pcm-descomplicado/>. Acesso em: 04.05.2019. Citado 2 vezes nas páginas 8 e 17.

KERZNER, H. Gestão de projeto 2ª edição. *Bookman Editora*, 2009. Acesso em: 03.05.2019. Citado na página 16.

LAMECK, O. *Uma Breve Introdução ao Kanban*. 2016. Disponível em: <https://blog.diferencialti.com.br/uma-breve-introducao-ao-kanban/>. Acesso em: 05.05.2019. Citado 2 vezes nas páginas 8 e 21.

LEACH, G. Meetings at work: Perceived effectiveness and recommended improvements. *Journal of Business Research*, 2015. Disponível em: <https://www.sciencedirect.com/science/article/abs/pii/S0148296315000879>. Acesso em: 02.06.2019. Citado na página 13.

MACLEOD, L. Conducting a well-managed meeting. *Physician Executive*, 2011. Disponível em: <http://dev.orgwise.ca/sites/osi.ocasi.org.stage/files/Conducting%20a%20Well-Managed%20Meeting.pdf>. Acesso em: 02.06.2019. Citado na página 13.

PERLOW, H. Stop the meeting madness: How to free up time for meaningful work. *Harvard Business Review*, 2017. Disponível em: <https://hbr.org/2017/07/stop-the-meeting-madness>. Acesso em: 02.06.2019. Citado na página 13.

PMBOK, P. *Um Guia do Conhecimento em Gerenciamento de Projetos 5ª edição*. [S.l.]: Saraiva, 2012. Acesso em: 30.04.2019. Citado na página 16.

ROGELBERG, S. Meetings and more meetings: The relationship between meeting load and the daily well-being of employees. *Group Dynamics: Theory, Research, and Practice*, 2005. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.74.2962&rep=rep1&type=pdf>. Acesso em: 02.06.2019. Citado na página 14.

RUP, I. *RUP - Rational Unified Process*. 1980. Disponível em: <http://www.anisio.eti.br/index.php/sistemas-de-informacao-menuvertical/conceito-de-sistema/item/47-rup-rational-unified-process>. Acesso em: 04.05.2019. Citado 2 vezes nas páginas 8 e 16.

SENADO, F. *Estrutura Senado*. 2019. Disponível em: <https://www12.senado.leg.br/institucional/estrutura>. Acesso em: 06.05.2019. Citado 2 vezes nas páginas 8 e 27.

SOARES, S. Metodologias Ágeis extreme programming e scrum para o desenvolvimento de software. 2009. Acesso em: 05.04.2019. Citado na página 18.

SOMMERVILLE, I. *Engenharia de Software 9ª edição*. [S.l.]: Pearson Education do Brasil, 2011. Acesso em: 01.04.2019. Citado 2 vezes nas páginas 22 e 24.

STANDISH, G. Failure record. *Standish Group Report Chaos*, 2014. Disponível em: <https://www.projectsmart.co.uk/white-papers/chaos-report.pdf>. Acesso em: 05.04.2019. Citado na página 18.

VIEIRA, R. *Casos de Uso*. 2015. Disponível em: <https://medium.com/operacionalti/uml-diagrama-de-casos-de-uso-29f4358ce4d5>. Acesso em: 05.06.2019. Citado 3 vezes nas páginas 8, 17 e 18.

Apêndices

APÊNDICE A – Figuras

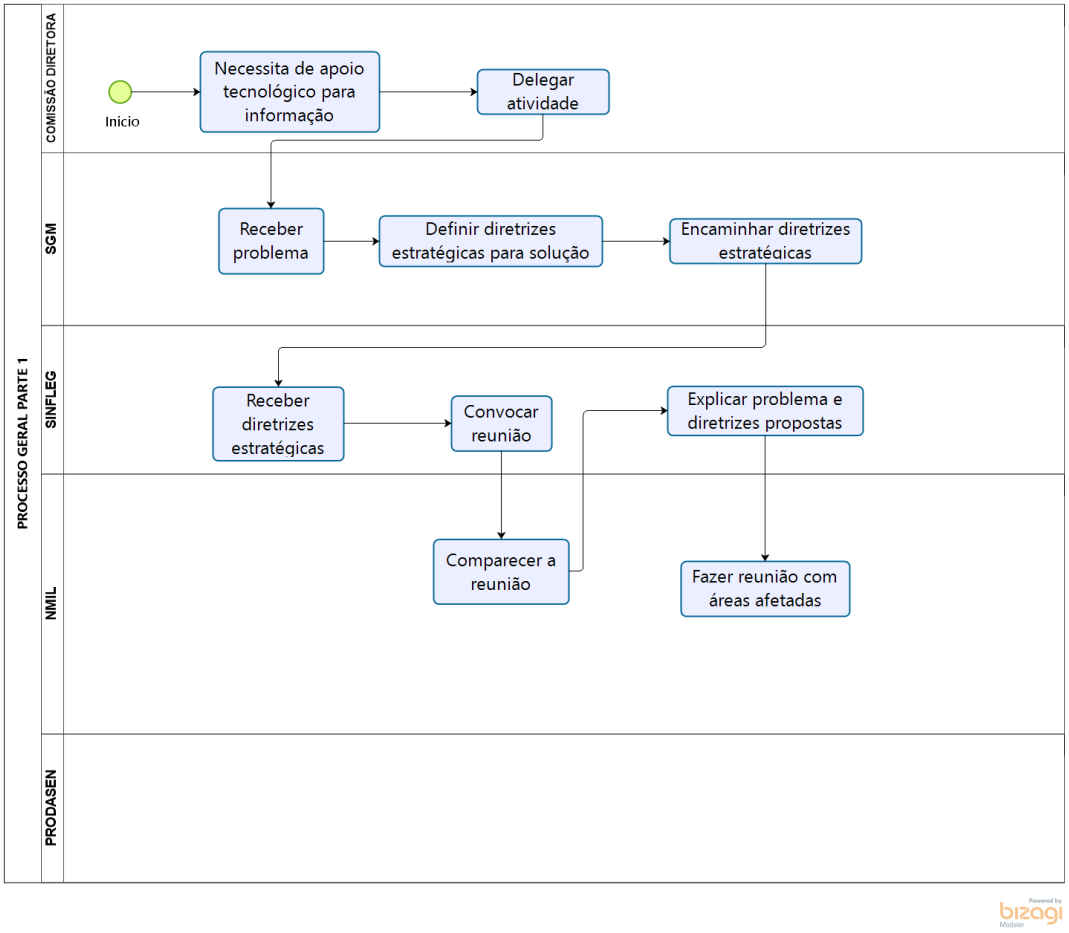

Figura 12 – Modelagem de Processo Geral 1 Parte 1

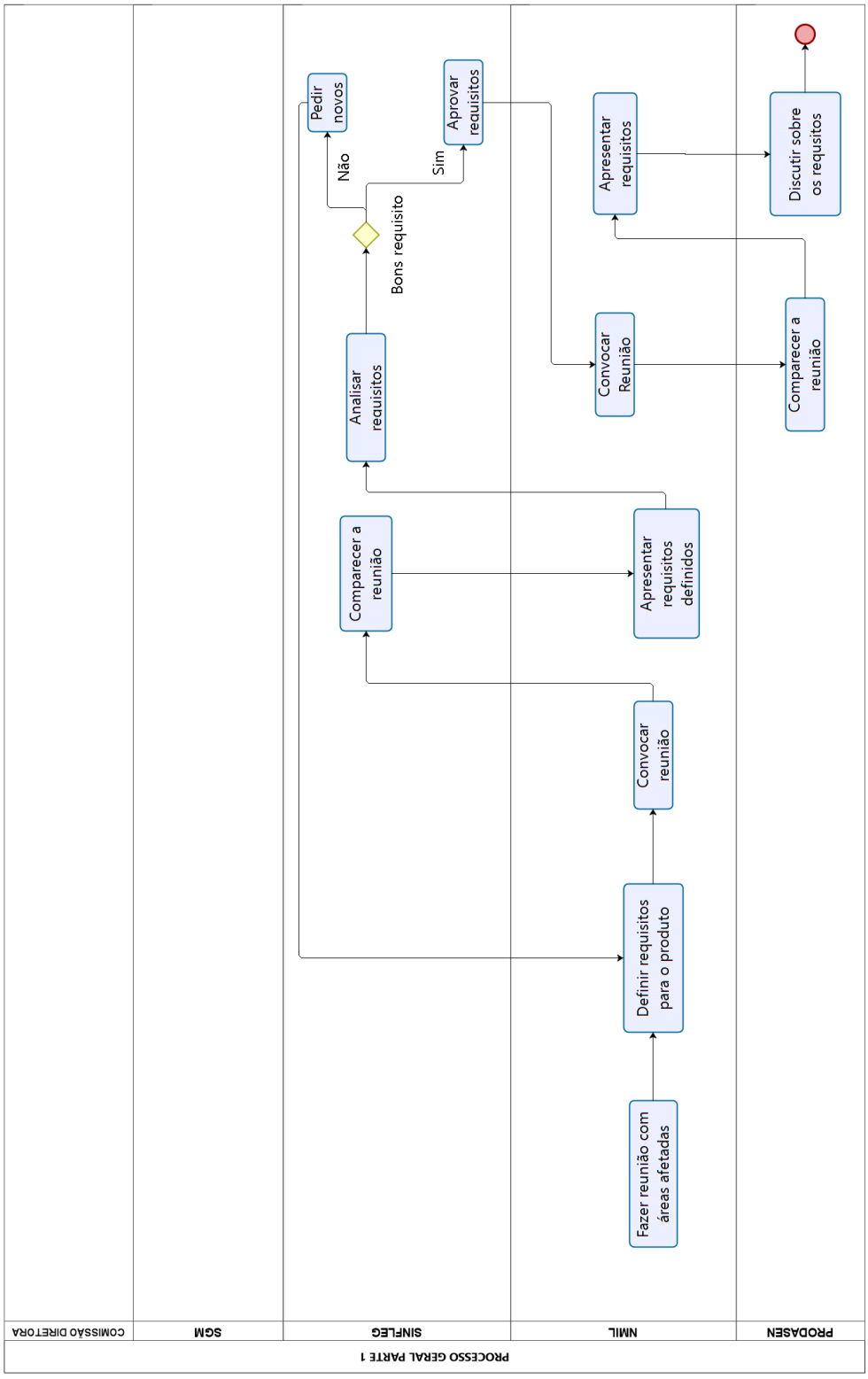

Figura 13 – Modelagem de Processo Geral 1 Parte 2

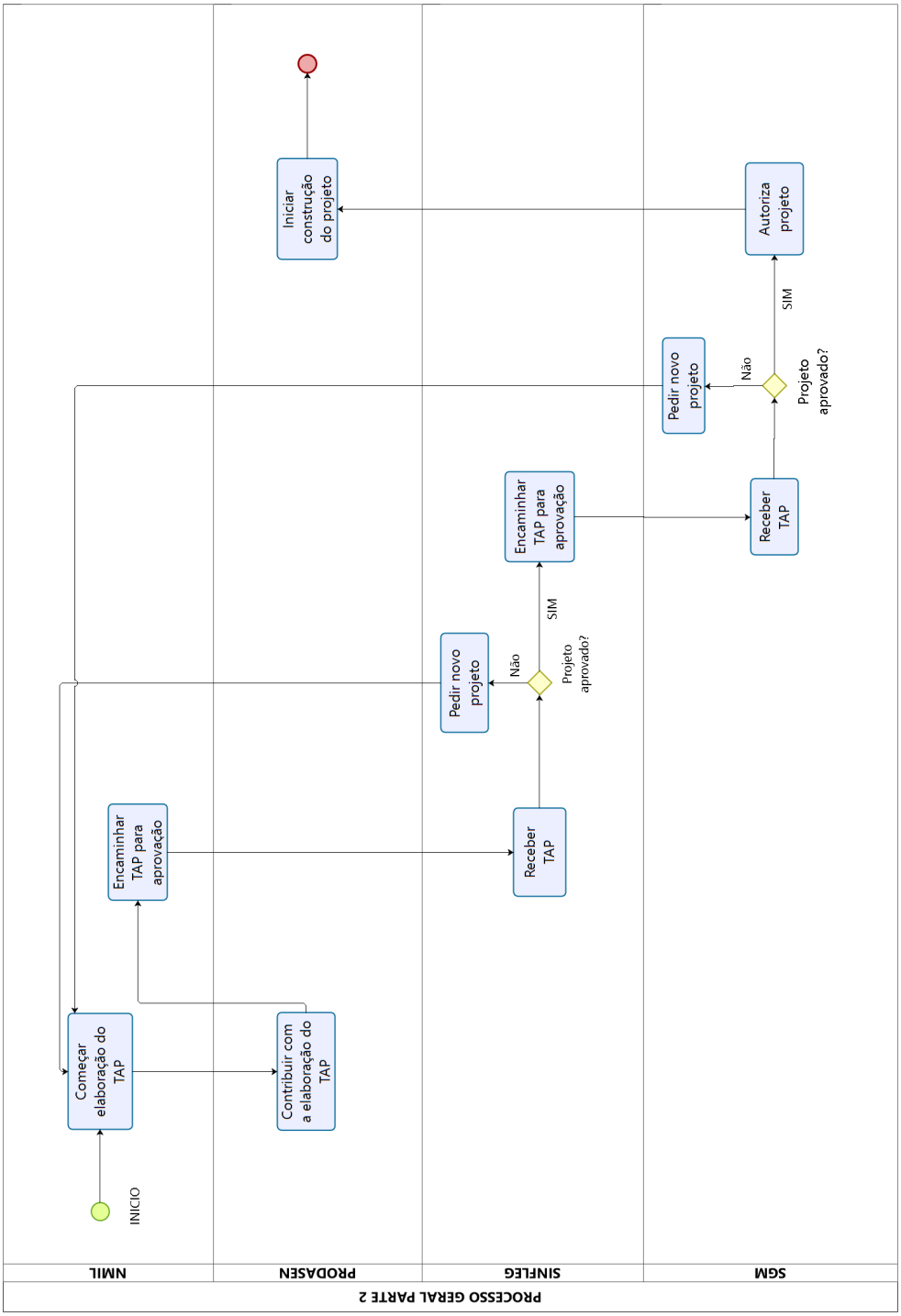

Figura 14 – Modelagem de Processo Geral 2 Parte 1

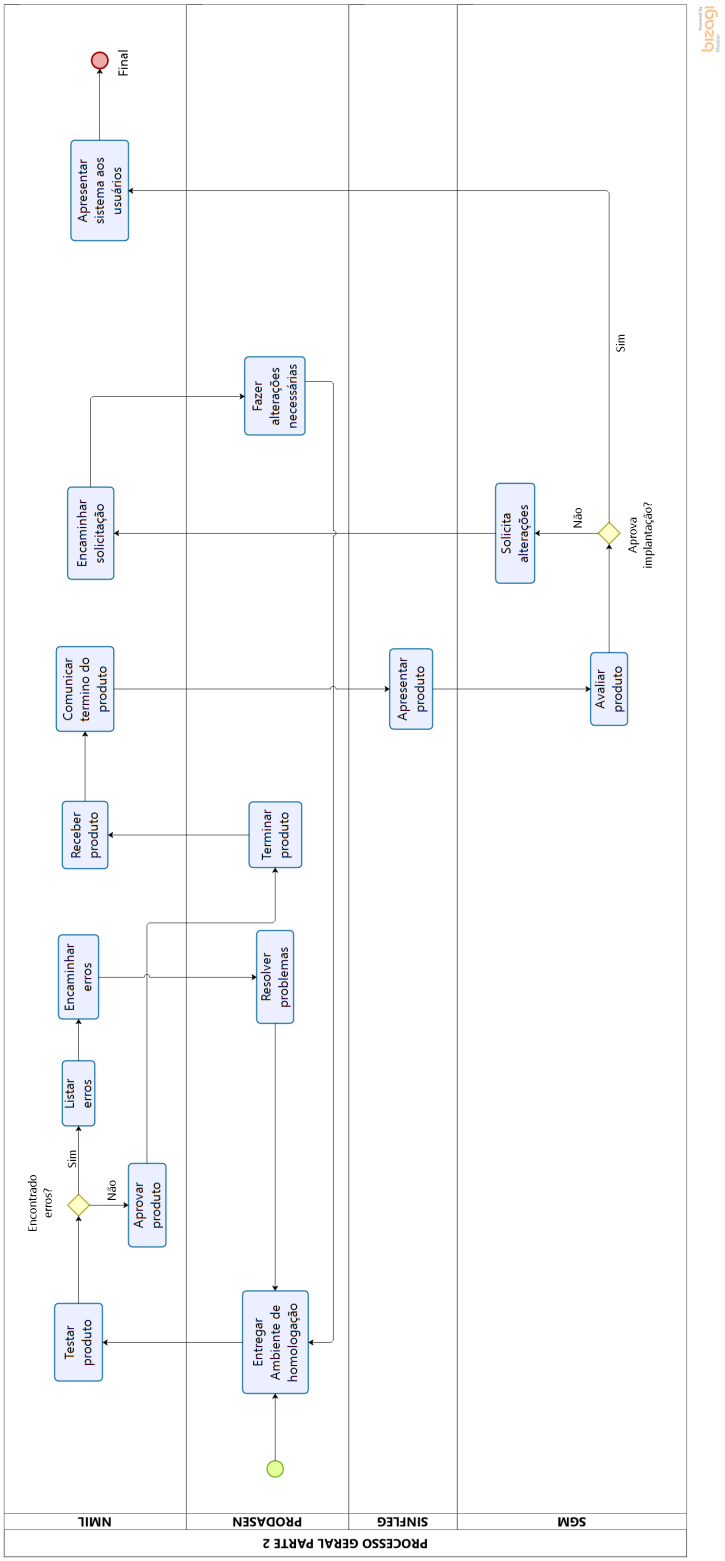

Figura 15 – Modelagem de Processo Geral 2 Parte 2