UNIVERSIDADE FEDERAL DE GOIÁS
Instituto de Informática
Engenharia de Software

Fernando Henrique Callata
Gustavo Batista
Lucas Sampaio
Pedro Basílio
Saulo Calixto

DOCUMENTAÇÃO DE PROCESSO - PROMOSHARE

Goiânia, 30 de Setembro

Controle do Documento

Versão	Data	Autor/Revisor	Descrição
0.1	25/09/2017	Saulo Calixto / Gustavo Batista / Fernando Henrique / Lucas Sampaio / Pedro Basílio	Versão inicial
0.2	15/11/2017	Saulo Calixto / Gustavo Batista / Fernando Henrique / Lucas Sampaio	Versão atualizada

Localização e Distribuição:

Este documento pode ser encontrado no repositório do github:

1. Apresentação:

Essa documentação se refere ao processo para criação do software PromoShare. Seu principal foco é a 1ª fase do Ciclo de Vida de Software que engloba a construção do produto, desde a aprovação da demanda até sua implementação.

Esse processo tem como principais objetivos: ser aderente aos padrões de desenvolvimento utilizando metodologia ágil; ser claro na definição de atividades definindo bem os responsáveis por cada uma. Além de tudo é o foco aprimorar a qualidade do software a ser entregue, garantindo que o escopo da demanda seja plenamente atendido.

2. Escopo:

Esse processo engloba todas as etapas necessárias para o cumprimento da 1ª fase do Ciclo de vida de Software, que são:

- Concepção;
- Especificação;
- Projeto Arquitetônico;
- Construção;
- Testes;
- Homologação;
- Implantação;

Salientamos que estão englobadas ao escopo apenas as atividades da 1ª fase do ciclo de vida do software, não contemplando os demais ciclos.

3. Papéis e Responsabilidades:

Os papéis e suas responsabilidades estão definidos de acordo com o escopo do software que está sendo desenvolvido.

3.1 ***Usuário:*** entre suas responsabilidades está a de representar a área de negócios demandante do software em todas as fases do processo. Além de prover e esclarecer dúvidas sobre os requisitos de software. Em suma, cabe ao usuário mostrar qual sua necessidade e ajudar a entender as regras de negócio envolvidas.

Além disso ele que homologa as funcionalidades e autoriza a implantação do software em produção.

3.2 Gerente de Projetos: Dentre suas responsabilidades estão: planejar, acompanhar e controlar o desenvolvimento do software, elaborar o cronograma, orçamento e plano de trabalho além de gerenciar e controlar as mudanças do projeto.

3.3 Analista de Requisitos: Responsável por auxiliar o usuário na definição do escopo do software, elaborar o documento de visão, detalhar requisitos, elaborar documento de requisitos, apoiar a preparação dos testes, executar a homologação do software em conjunto com o usuário, registrar ocorrências de defeito durante a homologação e participar da publicação do software.

3.4 Analista de Qualidade: Elabora planilha com casos de teste, executa testes integrados para validar o código, registra ocorrências de defeitos encontrados;

3.5 Arquiteto de Software: Define a arquitetura da solução, incluindo a modelagem dos componentes e plataforma tecnológica, elabora diagramas que compõem o Desenho Arquitetônico, provê esclarecimento a eventuais dúvidas dos construtores sobre a arquitetura definida.

3.6 Desenvolvedor: Ele segue a arquitetura definida, tirando dúvidas quando necessário. Elabora o código da solução, participa da construção do código de testes unitários além de prover a publicação nos ambientes de desenvolvimento, homologação e produção. Lembrando que essa lista não é exaustiva.

3.7 Analista de Infraestrutura: Disponibiliza e mantém os ambientes de desenvolvimento, homologação e produção. Implanta o software em produção, executando os scripts fornecidos pelo Construtor.

3.8 ***Scrum Team*****:** Scrum Master, Analista de Requisitos, Analista de Testes, *Desenvolvedor*;

4. Ciclo de Vida

Nesse processo foi utilizado dois ciclos de vida distintos para atender a necessidade do projeto a ser desenvolvido. Os modelos usados foram o Cascata e o Ágil, que são os modelos mais utilizados atualmente. Sendo que cada atividade usou um ou outro modelo de acordo com suas características.

O modelo ágil é iterativo, geralmente todas as fases do ciclo de vida de um software acontecem várias vezes em pequenas iterações, até todo o sistema ser criado. Tudo é feito por *baby-steps*.

Já o modelo cascata tem uma filosofia diferente, pois cada fase nesse é modelo é bem dividida, depois que se termina de levantar requisitos, por exemplo, não voltamos neles para fazer retificações. O desenvolvimento acontece de uma só vez, não de forma iterativa como no modelo ágil.

Em nosso processo decidimos utilizar o modelo cascata na Definição do Projeto, na fase de Concepção, Homologação e Implantação. Essas fases acontecem apenas uma vez, apesar dos requisitos evoluírem com o tempo, isso é feito na sprint backlog.

Os outros processos foram feitos utilizando a metodologia ágil juntamente com o scrum que divide as iterações em sprints curtas o que permite reavaliar o que está sendo desenvolvido, os requisitos e assim se houver mudanças há tempo hábil para corrigir.

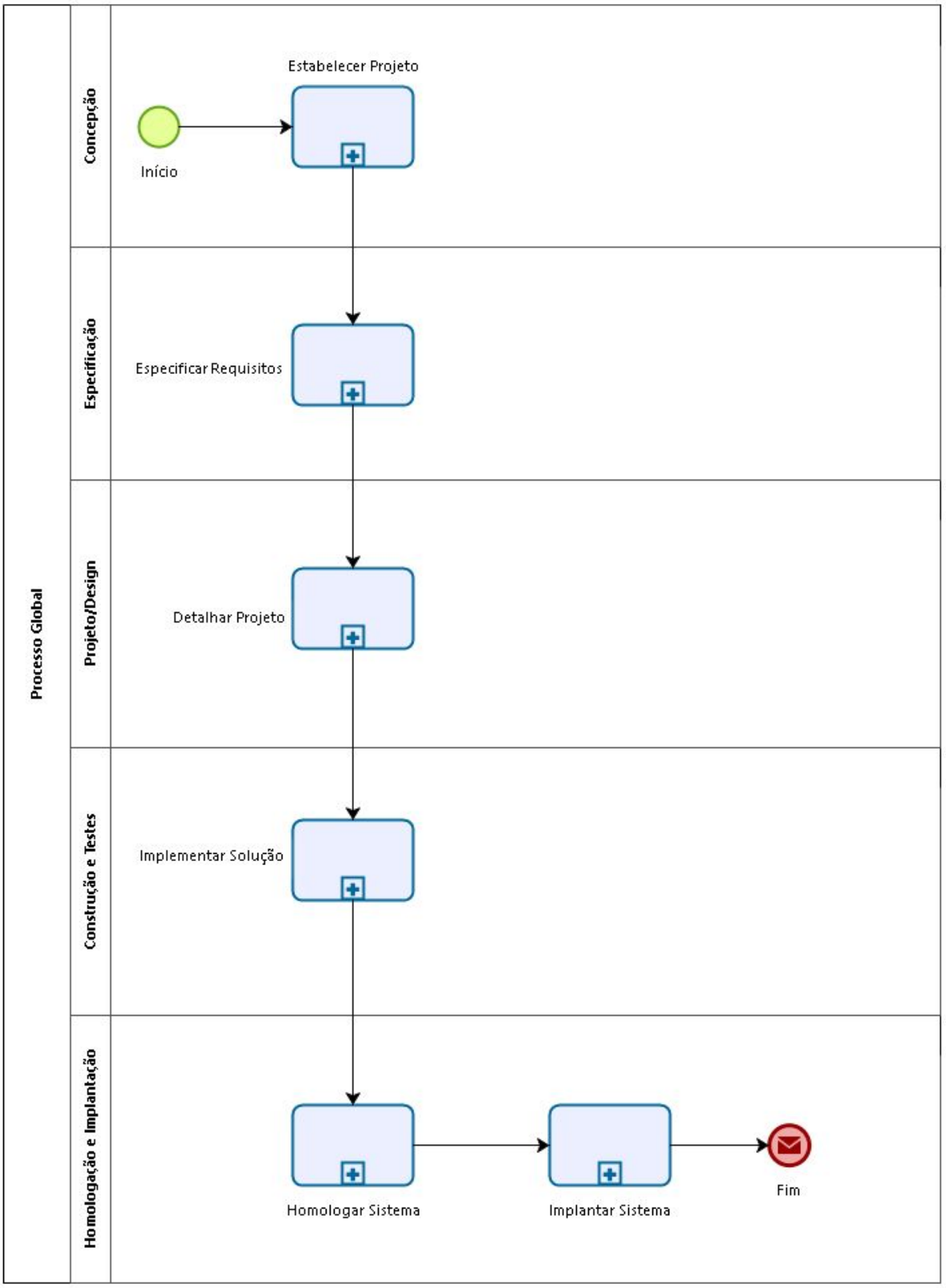
5. Descrição das Entregas:

Nome	Descrição	Opc?
Documento de Visão	Um documento, escrito em linguagem corrente, com base em entrevistas com o Usuário, que define a abrangência do software a ser desenvolvido.	
Cronograma Preliminar	Primeira versão dos prazos esperados para cada atividade prevista no projeto. O cronograma é constantemente revisto no decorrer do projeto, podendo sofrer alterações, controladas pelo Gerente de Projeto.	
Documento de Requisitos	Um documento, escrito em linguagem corrente, com base em entrevistas com o Usuário, que especifica detalhadamente os requisitos do software a ser desenvolvido.	
Diagrama de caso de uso	Diagramas UML que permitem visualizar, de maneira gráfica, as funcionalidades necessárias da aplicação, bem como seus cenários de execução, operações efetuadas e resultados esperados.	
Protótipo de Interfaces	Esboço das interfaces com o usuário a serem implementadas pela aplicação, bem como a seqüência de navegação entre as mesmas. Guia o trabalho de desenvolvimento das interfaces, bem como minimiza um eventual esforço de adaptação das mesmas ao cliente.	Sim
Diagrama da Arquitetura do Software	Diagrama desenhando a solução a ser implementada, com os elementos conceituais e as relações em uma representação esquemática.	
Diagrama de Classes	Diagrama UML contendo as entidades a serem implementadas para a solução do problema e seus relacionamentos, bem como seus atributos e métodos.	
Documentação da Arquitetura	Descrição detalhada do Diagrama da Arquitetura, incluindo sua estrutura estática, e comportamento esperado em tempo de execução.	
Documentação das Classes	Descrição detalhada das classes do Diagrama e seu comportamento.	

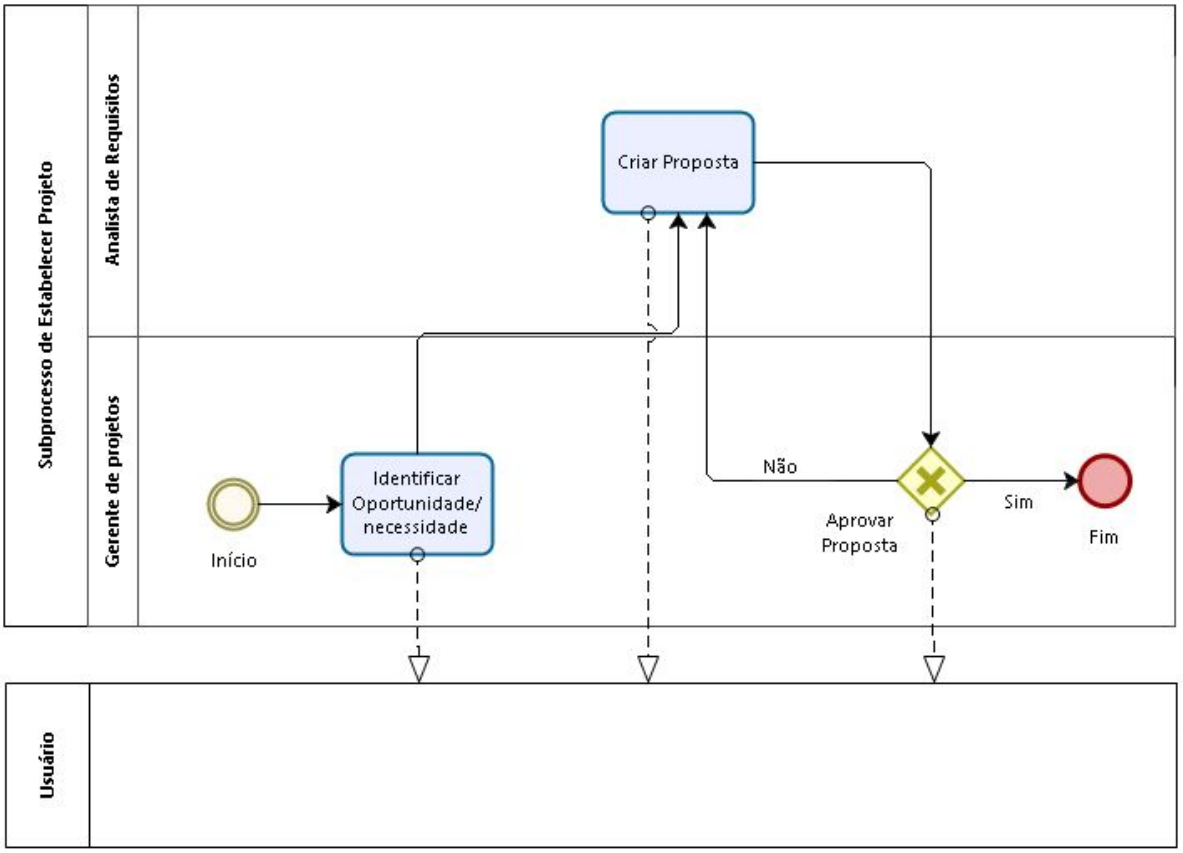
Documentação de Caso de Teste	Contém a especificação dos testes unitários e integrados a que o software é submetido.	
Código Fonte	Conjunto de programas codificados que compõem o software.	
Evidências de Testes de Casos de Uso	Relatório onde constam os dados dos testes e os resultados obtidos.	
Software Verificado	Representa o código fonte testado e aprovado pelo Analista de Qualidade.	
Software Validado	Representa o código fonte testado e homologado pelo Analista de Requisitos e Usuário.	
Termo de Homologação	Representa o aceite final do software por parte do Usuário.	
Software em Produção	Representa o código fonte instalado e operacional no ambiente de produção.	

6. Fluxos do Processo

6.1 Visão Geral:

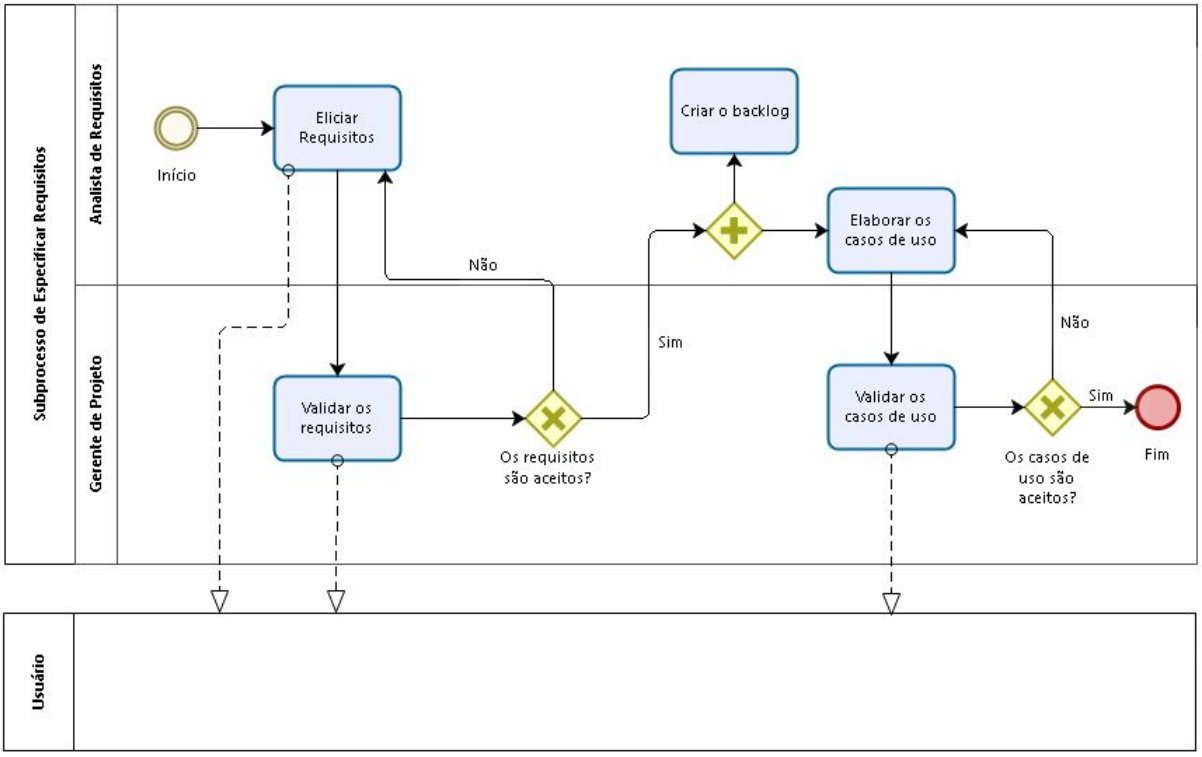


6.2 Fase de Concepção:



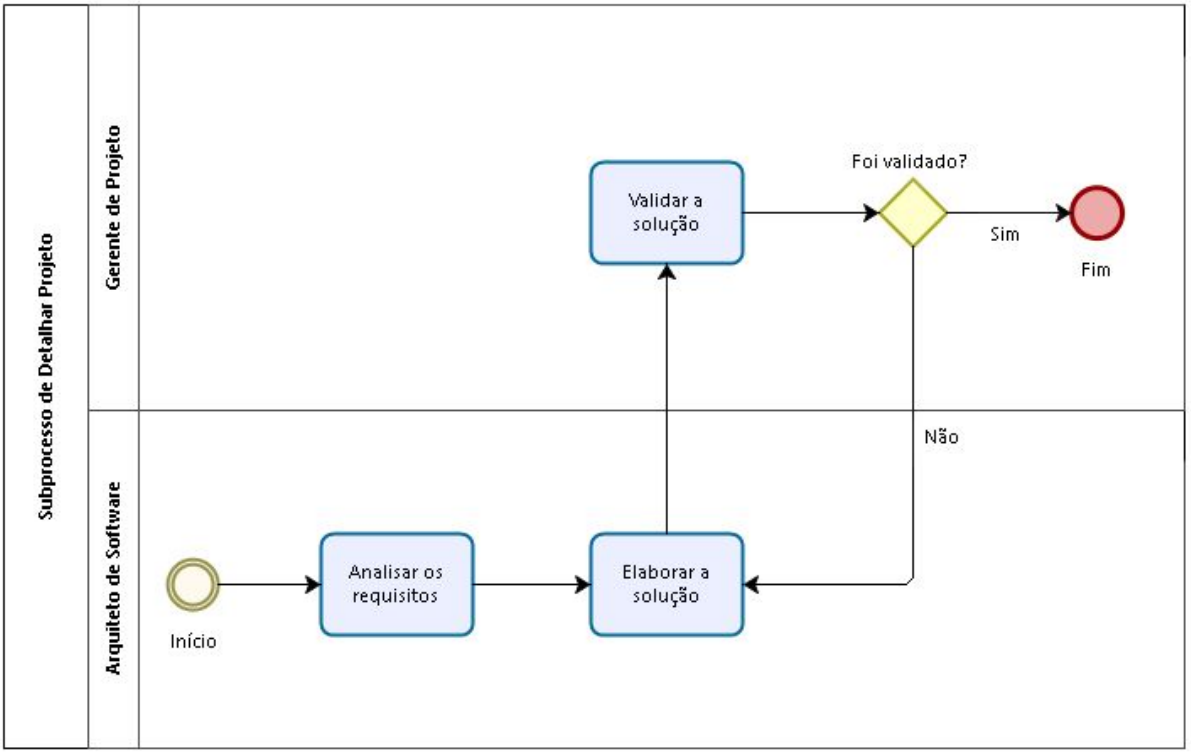
Objetivo	● Identificar os requisitos do usuário por meio do entendimento da demanda ● Definir o escopo da solução a ser desenvolvida ● Elaborar cronograma preliminar do projeto, com estimativa de custos
Papéis	● Usuário ● Analista de Requisitos ● Gerente de projetos
Entradas	● Demanda Aprovada
Saídas	● Documento de Visão ● Cronograma Preliminar

6.3 Fase de Especificação:



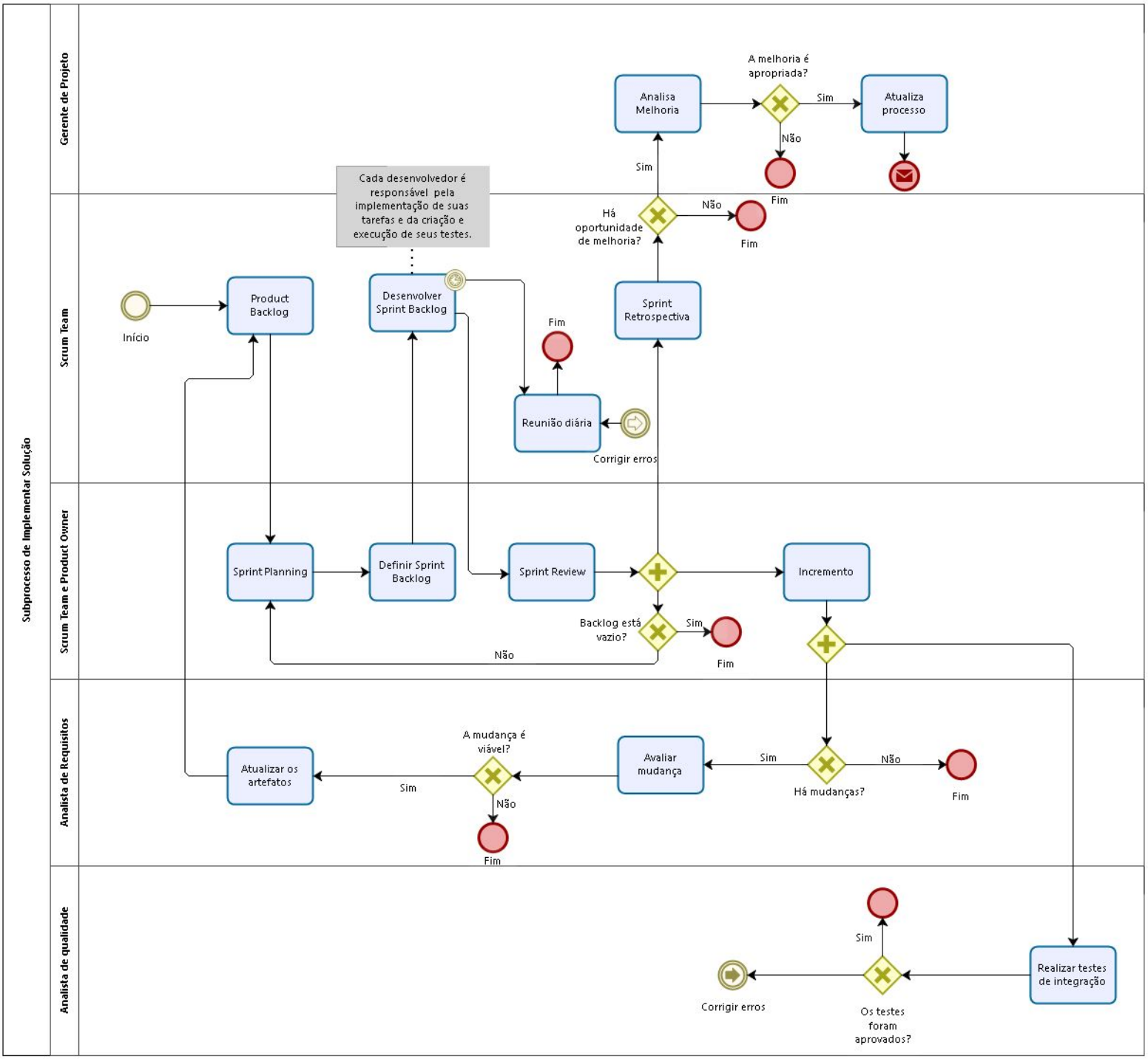
Objetivo	• Obter maior detalhamento dos requisitos e suas particularidades, de modo que se possa projetar a solução. • Ao final dessa fase, todos os documentos gerados devem ser validados pelo usuário de modo a padronizar o entendimento e expectativas em relação à solução.
Papéis	• Analista de Requisitos • Usuário • Gerente de projetos
Entradas	• Documento de Visão
Saídas	• Documento de Requisitos • Diagrama de Casos de Uso • Protótipo de Interfaces • Documentação de Caso de Teste

6.4 Fase de Definição do Projeto:



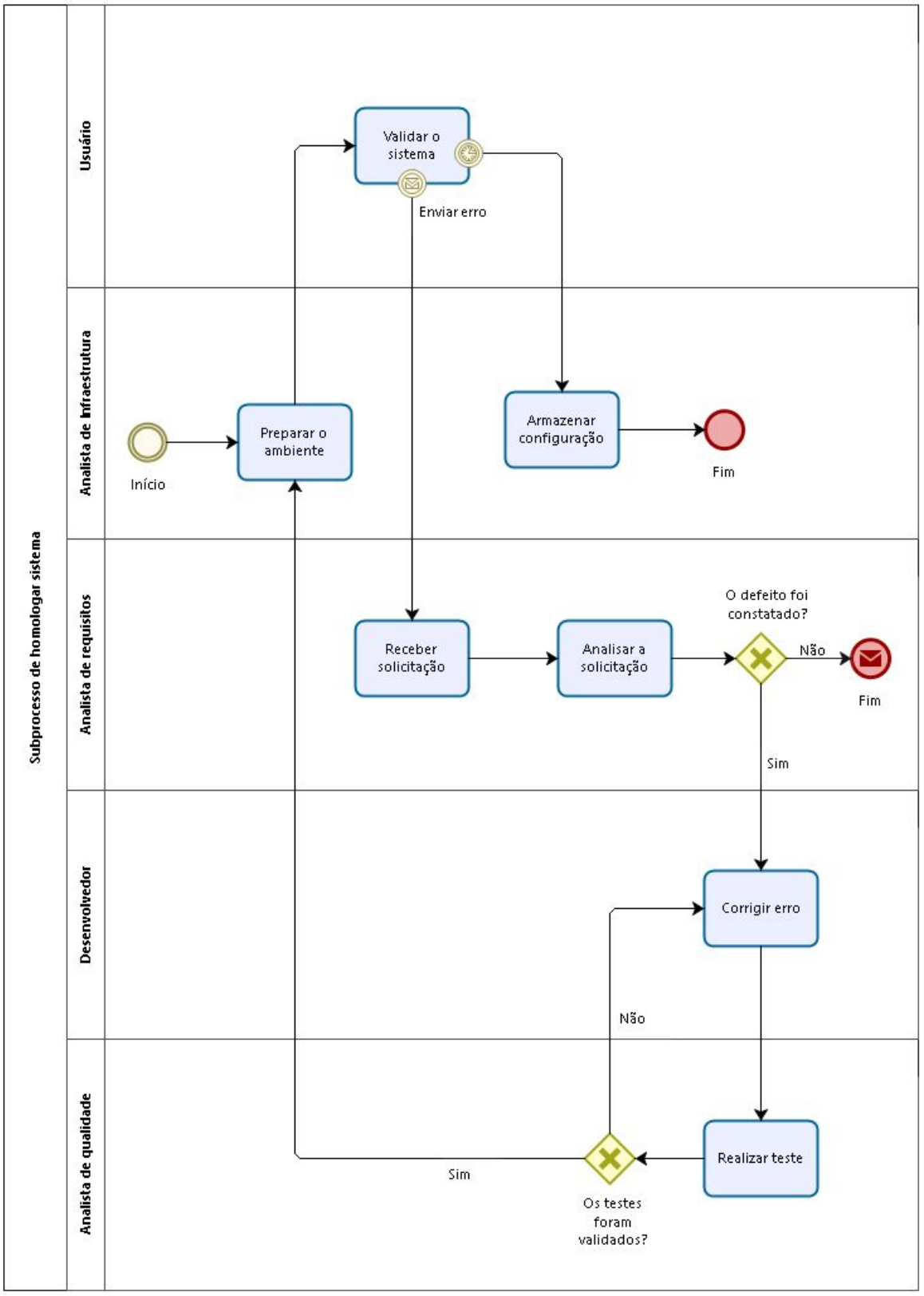
Objetivo	● Definir a arquitetura da aplicação, com base nas informações colhidas na fase anterior. ● A documentação a ser gerada exige, em sua confecção, o planejamento detalhado de toda a lógica da aplicação. ● Desse modo, o desenvolvedor recebe um esquema bem definido de como deve se comportar cada componente a ser codificado, preocupando-se com interação entre componentes.
Papéis	● Arquiteto de Software ● Gerente de projeto
Entradas	● Documento de Requisitos ● Diagrama de Casos de Uso ● Protótipo de Interfaces
Saídas	● Diagrama de Classes, Diagrama de arquitetura de Software ● Documentação da Arquitetura de Software ● Especificação de ferramentas como: Linguagem, ambiente de desenvolvimento, frameworks e etc.

6.5 Fase de Construção e Testes:



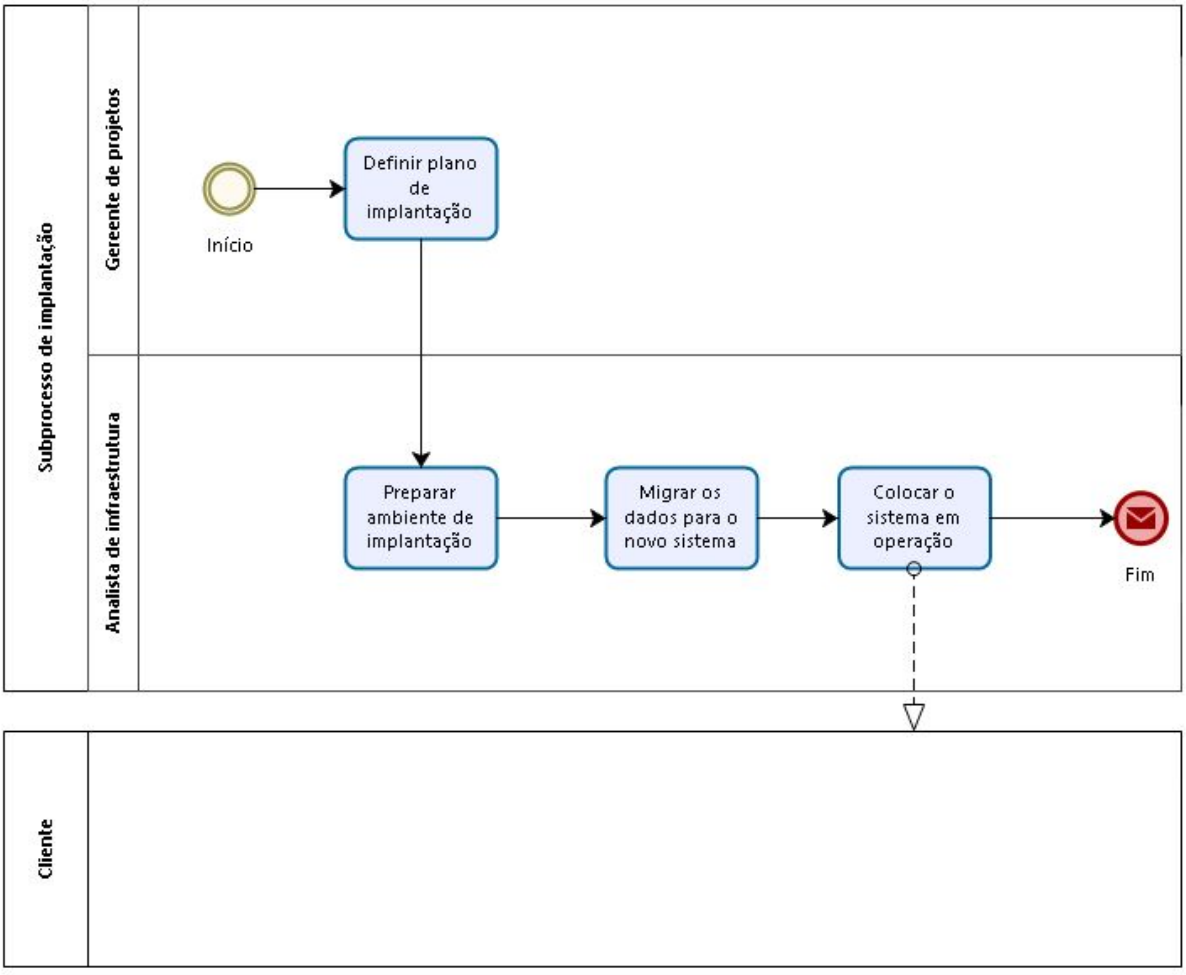
Objetivo	● Realizar a codificação e os testes unitários e de integração no software em desenvolvimento.
Papéis	● Construtor ● Analista de Qualidade ● Analista de Requisitos
Entradas	● Diagrama de Classes e Arquitetura ● Documentação de Caso de Teste
Saídas	● Código fonte ● Evidências de Testes de Casos de Uso

6.6 Fase de Homologação:



Objetivo	• Disponibilizar a aplicação em ambiente de homologação, o mais próximo possível do ambiente de produção, onde um grupo de usuários deverá verificar se o software atende às expectativas geradas na fase de especificação.
Papéis	• Usuário • Analista de Requisitos • Construtor
Entradas	• Software Verificado • Documentação de Caso de Teste • Evidências de Testes de Casos de Uso
Saídas	• Software Validado • Termo de Homologação

6.7 Fase de Implantação:



Objetivo	● Disponibilizar o software em ambiente de produção para uso da área de negócios demandante.
Papéis	● Analista de Requisitos ● Usuário ● Gerente de Projetos ● Construtor ● Analista de Infraestrutura
Entradas	● Objetos a serem copiados
Saídas	● Software em Produção

7. Políticas

As ferramentas que serão utilizadas para a construção do software poderá ser substituída por outra equivalente e que atenda às necessidades da implementação. Boa parte da equipe deve ter aderência às ferramentas definidas. Caso não haja um consenso, será feito um levantamento de nível de conhecimento e habilidades de ferramentas padrões que os desenvolvedores já possuem conhecimento e destreza e assim será estabelecido um plano de implementação a ser seguido.

Alterações na construção do software a nível arquitetural devem ser notificadas à todos os stakeholders, por mais simples que seja. É importante explicitar o que era antes e o que passou a ser diferente para que os envolvidos tenham conhecimento do andamento do processo de desenvolvimento. Isso faz com que exista uma linha do tempo com as principais modificações, quais foram e quais razões motivaram tais alterações.

6.1 Ferramentas: Recomendamos as seguintes ferramentas:

Ferramenta	**Finalidade**
LibreOffice	Documentação, planilha
Bizagi	Modelagem de processo
Astah	Diagrama de Classe, caso de uso e casos de teste
Mantis	Gerenciamento de defeitos
GitHub	Controle de versão de software
TeamCity	Integração Contínua
Artifactory	Automatização de tarefas
Web Driver	Testes automatizados na Web

Atividades informais não serão documentadas, desde que as ações não prejudiquem o andamento do processo. Todavia, se a atividade informal for caracterizada como correção de erro, por mínimo que seja, deverá ser realizado a descrição do problema e a solução proposta para posteriormente ser validada pelos revisores da equipe.

6.2 Modelo de Documentos

Documento	**Localização**
Documento de Requisitos	https://goo.gl/exfFVV
Diagrama de caso de uso	https://goo.gl/exfFVV
Protótipo de Interfaces	https://github.com/saulocalixto/ProcessoDeSoftware-ES-2017-2
Diagrama da Arquitetura do Software	https://goo.gl/b8Wr78
Documentação das Classes	https://goo.gl/b8Wr78
Documentação de Caso de Teste	https://github.com/saulocalixto/ProcessoDeSoftware-ES-2017-2

O protótipo de interface deve ser evoluído de acordo com o modelo de metodologia ágil adotado pela equipe. Caso o layout mude significativamente, deverá ser feito um documento de transição, evidenciando como era antes e como passou a ser de acordo com as mudanças sugeridas pelo dono do produto.

O diagrama de classes poderá ser modificado a qualquer momento sem que haja uma documentação específica indicando como era antes. A equipe apenas terá acesso ao diagrama e deverá sugerir mudanças, para que o arquiteto realize as adaptações necessárias ao negócio.