



INSTITUTO FEDERAL DE  
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA  
PARAÍBA

**INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA**  
**UNIDADE ACADÊMICA DE INFORMAÇÃO E COMUNICAÇÃO**  
COORDENAÇÃO DO CURSO SUPERIOR DE TECNOLOGIA EM SISTEMAS PARA INTERNET

# RELATÓRIO DE ESTÁGIO

Veículos: Sistema de Gestão de Frotas do TRE-PB

José Ricardo Bettini Pacola

João Pessoa - PB

04/2019

**Instituto Federal de Educação, Ciência e Tecnologia da Paraíba**  
**Unidade Acadêmica de Informação e Comunicação**  
**Coordenação do Curso Superior de Tecnologia em Sistemas para Internet**

## **Veículos: Sistema de Gestão de Frotas do TRE-PB**

**José Ricardo Bettini Pacola**

Relatório de Estágio Supervisionado apresentado à disciplina Estágio Supervisionado do Curso Superior de Tecnologia em Sistemas para Internet do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba, como requisito parcial para a obtenção do grau de Tecnólogo em Tecnologia de Sistemas para Internet.

<b>Orientador:</b>	Fausto Véras Maranhão Ayres
<b>Supervisor:</b>	Francisco José Rodrigues Gomes
<b>Coordenador do Curso:</b>	Cândido José Ramos do Egypto
<b>Empresa:</b>	Tribunal Regional Eleitoral da Paraíba
<b>Período:</b>	01/04/2017 à 01/04/2019

## **APROVAÇÃO**

**Esta folha deverá ser substituída pela  
cópia digitalizada da folha de aprova-  
ção fornecida.**

---

**Francisco José Rodrigues Gomes**

Supervisor da Empresa

---

**Cândido José Ramos do Egypto**

Coordenador do CST de Sistemas para Internet

---

**Prof. Dr. Fausto Vêras Maranhão Ayres**

Professor Orientador

---

**José Ricardo Bettini Pacola**

Estagiário

A Deus, a família e aos amigos que são a base e o caminho para um futuro sólido e construtivo.

## **AGRADECIMENTOS**

Edite e coloque aqui os agradecimentos às pessoas e/ou instituições que contribuíram para a realização do trabalho.

É obrigatório o agradecimento às instituições de fomento à pesquisa que financiaram total ou parcialmente o trabalho, inclusive no que diz respeito à concessão de bolsas.

*Eu denomino meu campo de Gestão do Conhecimento, mas você não pode gerenciar conhecimento. Ninguém pode. O que pode fazer - o que a empresa pode fazer - é gerenciar o ambiente que otimize o conhecimento. (PRUSAK, Laurence, 1997).*

## RESUMO

PACOLA, José Ricardo Bettini. Veículos: Sistema de Gestão de Frotas do TRE-PB. 04/2019. 39 f. Relatório de Estágio – Curso Superior de Tecnologia em Sistemas para Internet, Instituto Federal de Educação, Ciência e Tecnologia da Paraíba. João Pessoa - PB, 04/2019.

Nesse relatório de estágio, requisito para conclusão do Curso Superior de Tecnologia em Sistemas para Internet, são apresentadas atividades referentes ao planejamento, desenvolvimento e execução do Sistema de Gestão de Frotas denominado Veículos, para o Tribunal Regional Eleitoral da Paraíba. O projeto tem previsão para ser concluído em até seis meses, e será supervisionado pelo chefe da SEDES, Francisco Gomes, bem como acompanhado pelo supervisor técnico, que é o desenvolvedor responsável pelo projeto, efetivo do Tribunal. A equipe no projeto fica então composta por 1 gerente e 3 desenvolvedores, sendo 1 supervisor técnico e 2 estagiários, 1 Product Owner e 1 DBA responsável pelo gerenciamento do banco de dados, este lotado na SISBAN. Esse projeto colaborou profundamente para a fixação dos conhecimentos adquiridos no decorrer do curso, principalmente nas tecnologias: Java, JSF, SQL e SVN. O principal objetivo do estágio foi utilizar, na prática, os conhecimentos obtidos ao longo do curso, bem como aprender conceitos e tecnologias extras, adquirir experiência com a rotina de trabalho e principalmente aprender a codificar em equipe utilizando versionamento de código, seguindo todas as etapas previstas no PDS denominado MODUS.

**Palavras-chave:** Java. JSF. SQL. Gestão. Desenvolvimento.

## ABSTRACT

PACOLA, José Ricardo Bettini. Title in English. 04/2019. 39 f. Relatório de Estágio – Curso Superior de Tecnologia em Sistemas para Internet, Instituto Federal de Educação, Ciência e Tecnologia da Paraíba. João Pessoa - PB, 04/2019.

Elemento obrigatório em tese, dissertação, monografia e TCC. É a versão do resumo em português para o idioma de divulgação internacional. Deve ser antecedido pela referência do estudo. Deve aparecer em folha distinta do resumo em língua portuguesa e seguido das palavras representativas do conteúdo do estudo, isto é, das palavras-chave. Sugere-se a elaboração do resumo (Abstract) e das palavras-chave (Keywords) em inglês; para resumos em outras línguas, que não o inglês, consultar o departamento / curso de origem.

**Keywords:** Java. JEE. JSF, Gerenciamento, Projeto, Veículos.



## LISTA DE FIGURAS

Figura 1 – Exemplo do ciclo SCRUM . . . . .	5
Figura 2 – Catálogo técnico de sistemas - Veículos . . . . .	8
Figura 3 – Ata de Reunião - 21/03/2017 . . . . .	10
Figura 4 – Planejamento das versões . . . . .	12
Figura 5 – Atividade 8940 . . . . .	13

## LISTA DE ABREVIATURAS E SIGLAS

ABNT	Associação Brasileira de Normas Técnicas
DECOM	Departamento de Computação
PDS	Plano de Desenvolvimento de Software
MODUS	Modelo de Desenvolvimento de Software
SEDES	Seção de Desenvolvimento de Sistemas
SISBAN	Seção de Implantação de Sistemas e Banco de dados
TRE-PB	Tribunal Regional Eleitoral da Paraíba
IFPB	Instituto Federal de Educação da Paraíba
SQL	Structured Query Language
DBA	DataBase Administrator
JSF	JavaServer Faces
SVN	Apache Subversion

## SUMÁRIO

<b>1 – Introdução</b>	<b>1</b>
1.1 Objetivo	1
1.1.1 Objetivo geral	1
1.1.2 Objetivos específicos	1
1.2 A Empresa	1
1.3 Descrição geral das atividades	2
1.4 Organização do relatório	2
<b>2 – Embasamento teórico</b>	<b>4</b>
2.1 SCRUM	4
2.2 Java	5
2.3 JSF	6
2.4 PrimeFaces	6
2.5 JPA	6
<b>3 – Veículos: Sistema de Gestão de Frotas do TRE-PB</b>	<b>8</b>
3.1 A análise do problema	9
3.2 Planejamento	9
3.3 Desenvolvimento	10
<b>4 – CONCLUSÃO</b>	<b>14</b>
4.1 TRABALHOS FUTUROS	14
4.2 CONSIDERAÇÕES FINAIS	14
<b>Referências</b>	<b>15</b>
 <b>Apêndices</b>	 <b>16</b>
<b>APÊNDICE A–Nome do apêndice</b>	<b>17</b>
<b>APÊNDICE B–Nome do outro apêndice</b>	<b>18</b>
 <b>Anexos</b>	 <b>19</b>
<b>ANEXO A–Modus 3.1</b>	<b>20</b>

**ANEXO B–Portaria 37/2017 . . . . . 36**

## 1 Introdução

Este relatório descreve as atividades realizadas, no que diz respeito ao planejamento, desenvolvimento e execução de um dos projetos desenvolvidos pela SEDES, denominado de Veículos – Sistema de gestão de frotas do TRE-PB. O mesmo foi executado no exercício do estágio supervisionado no Tribunal Regional Eleitoral da Paraíba, localizado no Centro em João Pessoa. Serão explanados, tanto os conhecimentos teóricos, quanto os práticos, descrevendo todas as etapas do projeto. Também estarão presentes neste relatório informações referentes à empresa e sobre a unidade da empresa onde o projeto foi executado, além de informações sobre infra-estrutura. Por fim, serão mencionadas as correlações entre o conteúdo visto em sala de aula e o que foi feito na prática, bem como as principais dificuldades encontradas durante a execução do projeto.

### 1.1 Objetivo

#### 1.1.1 Objetivo geral

Auxiliar no desenvolvimento dos ativos, soluções web desenvolvidas pela SEDES, no TRE-PB. Para vistas deste relatório será utilizado como referência o projeto Veículos: Sistema de Gestão de Frotas do TRE-PB. O sistema deverá ser capaz de receber solicitações dos usuários e exibi-las em um painel onde o gestor deverá analisar os pedidos e montar as viagens adequando: rotas, disponibilidades dos motoristas e passageiros. Nas atividades são contempladas todas as etapas descritas no Modelo de Desenvolvimento de Software adotado pelo TRE-PB.

#### 1.1.2 Objetivos específicos

Contribuir na implementação do Sistema de Gestão de Frotas, quando possível oferecendo alternativas para a implementação mediante aos conhecimentos adquiridos na graduação. Ganhar experiência com o fluxo de trabalho da uma equipe de desenvolvimento incorporando conceitos de versionamento de código, sendo capaz de resolver conflitos de códigos, quando houver, e evita-los. Trabalhar com reuniões diárias utilizadas pela abordagem da metodologia SCRUM, planejamento das soluções com a modelagem do negócio e com entregas frequentes sempre buscando feedback das releases pelo cliente.

### 1.2 A Empresa

<sup>1</sup> A Justiça Eleitoral é o ramo especializado do Poder Judiciário que visa garantir a lisura, a eficiência e a eficácia do processo eleitoral, contribuindo para o fortalecimento da democracia e a consolidação do Estado de Direito. Compete à Justiça Eleitoral preparar, realizar

---

<sup>1</sup><http://www.tre-pb.jus.br/institucional/conheca-o-tre-pb/conheca-o-tre-pb>

e apurar as eleições, além de administrar o Cadastro Nacional de Eleitores. O principal objetivo da Justiça Eleitoral é o gerenciamento do processo eleitoral, através de diretrizes claras e firmes, evitando vícios, abusos e fraudes. O Tribunal Regional Eleitoral da Paraíba - TRE-PB, órgão máximo da Justiça Eleitoral no Estado, tem como instância superior, em matéria eleitoral, o Tribunal Superior Eleitoral, sediado em Brasília - Distrito Federal. A finalidade do TRE-PB é planejar e coordenar o processo eleitoral nas eleições federais, estaduais e municipais, no âmbito do Estado da Paraíba. Compete, também, ao Tribunal, julgar os recursos interpostos das decisões dos Juízes e Juntas Eleitorais do Estado, bem como, os processos originários e administrativos do próprio Tribunal; registrar os partidos e candidatos a cargos eletivos de Governador, Senador, Deputado Federal e Estadual, assim como, receber e analisar a prestação de contas dos mesmos, prestadas ao final de cada campanha estadual; analisar as prestações de contas anuais dos órgãos regionais dos partidos políticos; elaborar e fiscalizar o calendário estadual de propaganda eleitoral; proceder à anotação e cancelamento dos diretórios estaduais e municipais dos partidos políticos; julgar as impugnações relativas aos pedidos de registros de candidaturas e as arguições de inelegibilidade; designar os Juízes Titulares das Zonas Eleitorais do Estado da Paraíba e administrar o Cadastro de Eleitores.

### 1.3 Descrição geral das atividades

Durante o processo foram utilizadas as metodologias Scrum para gestão e planejamento dos projetos e o Kanban para o controle de fluxo do desenvolvimento. As atividades desenvolvidas no período do estágio foram as seguintes de acordo com as fases:

1. Imersão:
  - a) Elicitação das histórias de usuário.
  - b) Definição do escopo do produto.
  - c) Criar estrutura do projeto.
  - d) Modelagem de dados.
2. Construção:
  - a) Refinar histórias de usuário.
  - b) Estimar histórias.
  - c) Codificação de funcionalidades.
  - d) Preparar versão para homologação.
  - e) Codificar ajustes.
  - f) Gerar documentação.
  - g) Implantar versão.

### 1.4 Organização do relatório

Além desse capítulo, o relatório está dividido em outros quatro capítulos brevemente descritos abaixo:

Capítulo 2 - Embasamento Teórico: Aborda as tecnologias e linguagens que foram utilizadas durante o estágio, bem como as definições necessárias para a compreensão do processo.

Capítulo 3 - Veículos: Sistema de Gestão de Frotas do TRE-PB: apresenta o projeto, no qual as atividades do estagiário foram realizadas, descrevendo os principais conceitos, arquitetura e funcionalidades.

Capítulo 4 – Atividades Realizadas: Relata as atividades realizadas ao longo do período de estágio; descreve o fluxo e o processo de desenvolvimento dessas atividades e detalha a implementação das funcionalidades.

Capítulo 5 – Conclusão: São feitas as considerações finais, ou seja, um relato sobre as experiências adquiridas, metas e objetivos alcançados, o que poderia ter sido feito para obter um melhor resultado final, erros cometidos e trabalhos futuros.

## 2 Embasamento teórico

Este capítulo apresenta uma breve fundamentação dos assuntos e conceitos necessários para a melhor compreensão deste trabalho, dando ênfase aos pontos mais relevantes para a compreensão das atividades realizadas.

A SEDES é uma seção responsável por implementar e realizar manutenções em ativos de TI do TRE-PB. Dessa forma ela se comporta como uma fábrica de software, seguindo rigorosos critérios no desenvolvimento de seus produtos, todos documentados em um modelo de desenvolvimento, visando a qualidade e segurança de seus produtos. Técnicas e tecnologias reconhecidas do mercado são utilizadas, dentre elas o algumas metodologias ágeis como SCRUM e KAMBAM, JAVA, ORACLE SQL.

### 2.1 SCRUM

SCRUM é uma metodologia (ou processo) de desenvolvimento iterativo e incremental, utilizada também no gerenciamento e desenvolvimento de software de forma ágil.

O SCRUM assume-se como uma metodologia extremamente ágil e flexível, que tem por objetivo definir um processo de desenvolvimento iterativo e incremental podendo ser aplicado a qualquer produto ou no gerenciamento de qualquer atividade complexa. (BISSI, 2007).

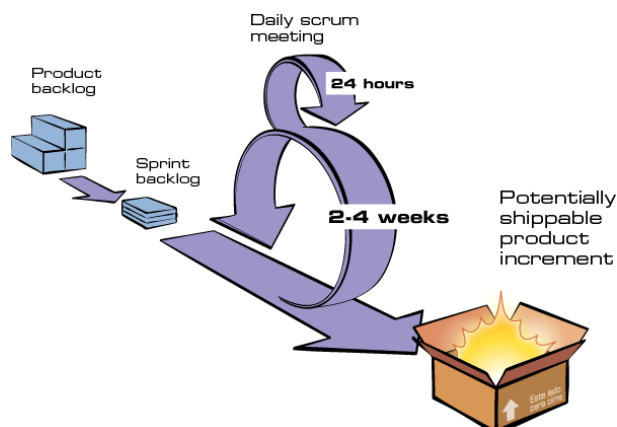
No SCRUM, os projetos são divididos em Sprints, que são ciclos, geralmente mensais, que representam o tempo no qual um determinado conjunto de atividades deve ser executado. Cada atividade ou funcionalidade a ser implementada no projeto são mantidas em uma lista, chamada de Product Backlog. Ao iniciar cada sprint é realizada uma reunião de planejamento, conhecida como Sprint Planning Meeting. Nessa reunião, o Product Owner, que é a pessoa que define o que está no Product Backlog, irá determinar as prioridades e a equipe irá discutir quais atividades ela será capaz de executar naquela sprint. A partir daí, a cada dia dessa sprint é feita uma reunião diária, geralmente realizada no início do dia, denominada de Daily Scrum. Nessa reunião, cada membro da equipe informa o que fez no dia anterior, e são identificados impedimentos e são estabelecidas novas prioridades para o dia atual.

Ao término de uma sprint, a equipe apresenta o que foi implementado, em uma pequena reunião, chamada Sprint Review Meeting. Logo após, é feito um novo planejamento para a próxima Sprint, reiniciando o ciclo, conforme é ilustrado na [Figura 1](#).

Outro conceito do SCRUM é o Planning Poker, que é uma técnica utilizada para estimar o prazo de um projeto. Resumidamente, nessa técnica, é usado um conjunto de cartas. Cada carta contém um número que representa pontos. Os números vão de 1 a 100, seguindo a sequência 0, 1, 2, 3, 5, 8, 13, 20, 40, 100. Cada membro da equipe recebe o conjunto de cartas. Escolhido um ticket (tarefa), os usuários ao mesmo tempo lançam à mesa a carta com



Figura 1 – Exemplo do ciclo SCRUM



Fonte: teste..

a quantidade de pontos que consideram que vale aquele ticket (geralmente, e esse foi o nosso caso, a quantidade de pontos remetia à quantidade de horas que levaríamos para executar aquela tarefa). O processo é repetido para todos os tickets. A ideia é instigar a discussão, pois, dificilmente, todos os membros da equipe irão jogar a mesma carta. Somente após um consenso é que o valor final é atribuído à tarefa (SABBAGH, 2014).

## 2.2 Java

Java<sup>1</sup> é uma linguagem de programação orientada a objetos, lançada em 1995, pela empresa Sun Microsystems, mas que, atualmente, pertence a Oracle (DEITEL, 2009). A linguagem Java permite o desenvolvimento de software para as plataformas desktop (Standard Edition), mobile (Micro Edition) e web (Enterprise Edition). O código em Java não é compilado para código nativo, mas sim para um bytecode, que é executado por uma máquina virtual, a JVM (Java Virtual Machine). O Java é amplamente utilizado ao redor do mundo, principalmente em ambientes corporativos. No Brasil, o Java é uma das principais linguagens de programação, sendo base para o ensino da Programação Orientada à Objeto na grande maioria dos cursos de programação. Além de ser bastante utilizada em organizações públicas. Isso se deve justamente pelo fato de que o foco do Java é em aplicações de médio a grande porte. Se forem bem usadas as recomendações e práticas do paradigma orientado à objeto, torna-se fácil a manutenção de uma aplicação Java, mesmo sendo de grande porte. Outra característica, que faz do Java uma ótima opção para esse escopo, é o suporte dela as diversas bibliotecas (ou APIs) para os mais diversos trabalhos como relatórios, persistência, gráficos, entre outras.

<sup>1</sup><http://www.oracle.com/technetwork/pt/java/index.html>

## 2.3 JSF

De acordo com [Cordeiro \(2014\)](#), JavaServer Faces<sup>2</sup>, ou, mais comumente, JSF, é um framework MVC para desenvolvimento web com Java, que veio para facilitar a construção de interfaces de usuário. A sua interface de usuário é baseada em componentes e orientada a eventos, sendo assim, os detalhes de manipulação dos eventos e a organização dos componentes são abstraídas. Com isso, o programador pode se concentrar bem mais na lógica do negócio. O JSF usa como sistema de template padrão o Facelets.

O JSF estabelece um conjunto de componentes pré-definidos para o desenvolvimento da interface de usuário. Para acessar esses componentes, ele fornece tags JSP. Outra característica interessante é que o JSF permite a reutilização dos componentes em uma página, aumentando a performance de carregamento da mesma. O JSF também faz uso do AJAX em alguns componentes, fazendo com que os processos sejam mais rápidos.

## 2.4 PrimeFaces

PrimeFaces<sup>3</sup> é uma biblioteca Open Source de componentes para o JSF. Essa biblioteca contribui para que o software tenha, o que chamamos de interface rica, devido ao grande conjunto de componentes. O PrimeFaces utiliza o jQuery2 e jQuery UI. Assim como outros frameworks para a parte visual do sistema, ele também se preocupa com a responsividade do layout. Os componentes do PrimeFaces foram construídos para usar AJAX por padrão, com isso o desenvolvedor não precisa ter a preocupação de realizar chamadas assíncronas para o servidor. O PrimeFaces também conta com um conjunto de temas (skins), que permite, de forma fácil mudar a aparência das aplicações. A grande característica do PrimeFaces é a sua simplicidade. Não é necessário configurar nenhum XML. Para utilizá-lo, basta colocar a biblioteca no projeto. Tudo isso está muito bem documentado no site do PrimeFaces, que também possui vários exemplos de código ([CIVICI, 2015](#)).

## 2.5 JPA

Java Persistence API<sup>4</sup> ou JPA é uma especificação criada em 2006 a partir do Hibernate<sup>5</sup>, que é um framework de mapeamento objeto-relacional (ORM), tendo em vista que outros frameworks estavam surgindo foi necessário criar um padrão com o intuito de resolver o famoso vendor lock-in, ou seja, uma vez usando determinada distribuição, ficava-se preso à mesma ([CORDEIRO, 2014](#), p. 12).

A abstração do JPA é a sua maior característica, permitindo que o programador troque o banco de dados sem maiores dificuldades. No projeto Veículos: Sistema de Gestão

---

<sup>2</sup><https://javaee.github.io/javaxserverfaces-spec/>

<sup>3</sup><https://primefaces.org/>

<sup>4</sup><http://www.oracle.com/technetwork/java/javaee/tech/persistence-jsp-140049.html>

<sup>5</sup><http://hibernate.org/>

de Frotas do TRE-PB foi usado o Hibernate através da especificação JPA. O uso da JPA é feito através de anotações na classe que representa o objeto que será persistido. Esse processo de anotar/configurar classes chama-se mapeamento. As classes depois de mapeadas são reconhecidas pelo Hibernate, que faz o seu processo natural de converter esse objeto para uma tabela no banco de dados ([CORDEIRO, 2014](#)).

### 3 Veículos: Sistema de Gestão de Frotas do TRE-PB

O projeto Veículos: Sistema de Gestão de Frotas do TRE-PB foi concebido com o objetivo de controlar as solicitações de veículos e as viagens realizadas, promovendo uma melhor gestão da frota de veículos do TRE-PB bem como possibilitando a realização de auditorias com base nos dados contidos no sistema. O sistema é hoje um ativo de TI que foi desenvolvido pela SEDES e está descrito no catálogo técnico de sistemas da COSIS conforme [Figura 2](#).

Figura 2 – Catálogo técnico de sistemas - Veículos

**Ativo de TI #9788**

**Veículos**  
Adicionado por Francisco José 5 meses atrás. Atualizado 3 meses atrás.

**Situação:** Em execução  
**Prioridade:** Normal  
**Atribuído para:** SEDES  
**Categoria:** Sistema de informação  
**Autenticação:** AD  
**Gerenciamento de permissões:** Própria aplicação

**Início:**  
**Data prevista:**  
**% Terminado:** 100%  
**Tempo estimado:**  
**Suporte:** SEDES  
**Manutenção de código:** SEDES

**Descrição**  
O sistema Veículos foi concebido com o objetivo de controlar as solicitações de veículos e as viagens realizadas, promovendo uma melhor gestão da frota de veículos bem como possibilitando a realização de auditorias com base nos dados contidos no sistema.

**Gestor do sistema**  
• SETRAN / COSEG (apesar de ainda não regulamentado em normativo interno)

**Principais usuários**  
• SETRAN - gestão de solicitação e liberação de veículos  
• Servidores do tribunal - solicitação de veículos

**Página do sistema em produção**  
• <http://portal.tre-pb.gov.br/intranet/servicos-ou-sistemas/veiculos>

**Links com documentação**  
• [Guia rápido de solicitação de veículos](#)  
• [Documentação de requisitos e evolução do produto](#)

**Gestão de acessos**  
• A SETRAN é a gestora das permissões de acesso, apesar de não haver normativo sobre a política de permissões do sistema.  
• Para acessar o sistema, o usuário deve possuir login na rede do TRE-PB.  
• Os seguintes perfis de acesso são tratados pelo sistema:  
  • Solicitante: responsável por elaborar e acompanhar a solicitação de veículo.  
    • Todos os servidores do tribunal que possuam cadastro no SGRRH são considerados solicitantes.  
  • Administrador: responsável por analisar as solicitações (deferir ou indeferir) e montar e gerir as viagens.  
    • Os servidores que possuam cadastro no SGRRH e que estejam no grupo veículos/Admin do Active Directory são considerados administradores.

**Tecnologias usadas**  
• Tecnologias de desenvolvimento:  
  • Java 8  
  • Maven 3  
  • JSF 2.2 com Primefaces 6.1  
  • Spring 4.3  
  • JPA 2 com Hibernate 5.2  
• Banco de dados: Oracle 12g (schema veículos)  
• Container de aplicação: Tomcat 8

**Conhecimento do negócio** (regras do negócio, processos de trabalho etc)  
• **Avançado** - SETRAN: Mônica e Mário - SEDES: Chico, Rômulo, Pedro e Ricardo  
• **Básico** - SEDES: Márcia e Paoli.

**Conhecimento técnico** (conhecimento do modelo de dados, servidor de aplicações, código-fonte etc)  
• **Avançado** - SEDES: Chico, Rômulo, Pedro e Ricardo  
• **Básico** - SEDES: Márcia e Paoli.

**Histórico**  
Atualizado por Márcia há 4 meses #1  
• Descrição atualizado(a) (diff)  
Atualizado por Márcia há 4 meses #2  
• Descrição atualizado(a) (diff)  
Atualizado por Márcia há 4 meses #3  
• Descrição atualizado(a) (diff)  
Atualizado por Rômulo há 3 meses #4  
• Descrição atualizado(a) (diff)

Exportar para Atom | PDF

Fonte: COSIS

O desenvolvimento do sistema passou por todas as etapas de um projeto de software conforme descrição no modelo de desenvolvimento de software no Anexo A. Esse modelo foi

construído a partir de definições estabelecidas em portarias, conforme exemplo, no Anexo B da última publicada pela Diretoria Geral do TRE-PB no que se diz respeito aos padrões de governança em Tecnologia da Informação. O processo de desenvolvimento e manutenção de software se inicia com a autorização de análise do problema, visando à elaboração de proposta de solução (CAVALCANTI, 2017, p. 2).

Nesse capítulo será explanada minha participação no processo de desenvolvimento do projeto o qual tive a oportunidade de participar desde a concepção, homologação, correção de problemas, criação de uma nova versão com expressiva mudança no modelo de dados e implementação da aplicação em produção.

### 3.1 A análise do problema

Uma reunião inicial é realizada entre todos os interessados. A COSIS reúne o gestor do sistema, o time de desenvolvimento e o cliente.

Para o time já se inicia um processo de imersão, onde após ouvir as necessidades descritas pelo cliente começa a imaginar e descrever os possíveis requisitos necessários para a solução. Toda reunião realizada entre a equipe e o cliente é descrita cronologicamente e fica publicada na categoria de Atas/Reuniões da SEDES na sitio Wiki do TRE-PB, com a data, hora de início e fim e o nome de todos os participantes, conforme Figura 3. Dessa forma todos tem acesso ao que realmente foi solicitado.

Posteriormente o time volta a se reunir e descrever as histórias de usuário na forma de requisitos a serem implementados. Com os requisitos descritos e cadastrados no sistema de gestão de projetos, o Redmine, na forma de tarefas e atividades, é possível mensurar o tamanho de cada atividade e assim estimar o tempo necessário para conclusão do projeto.

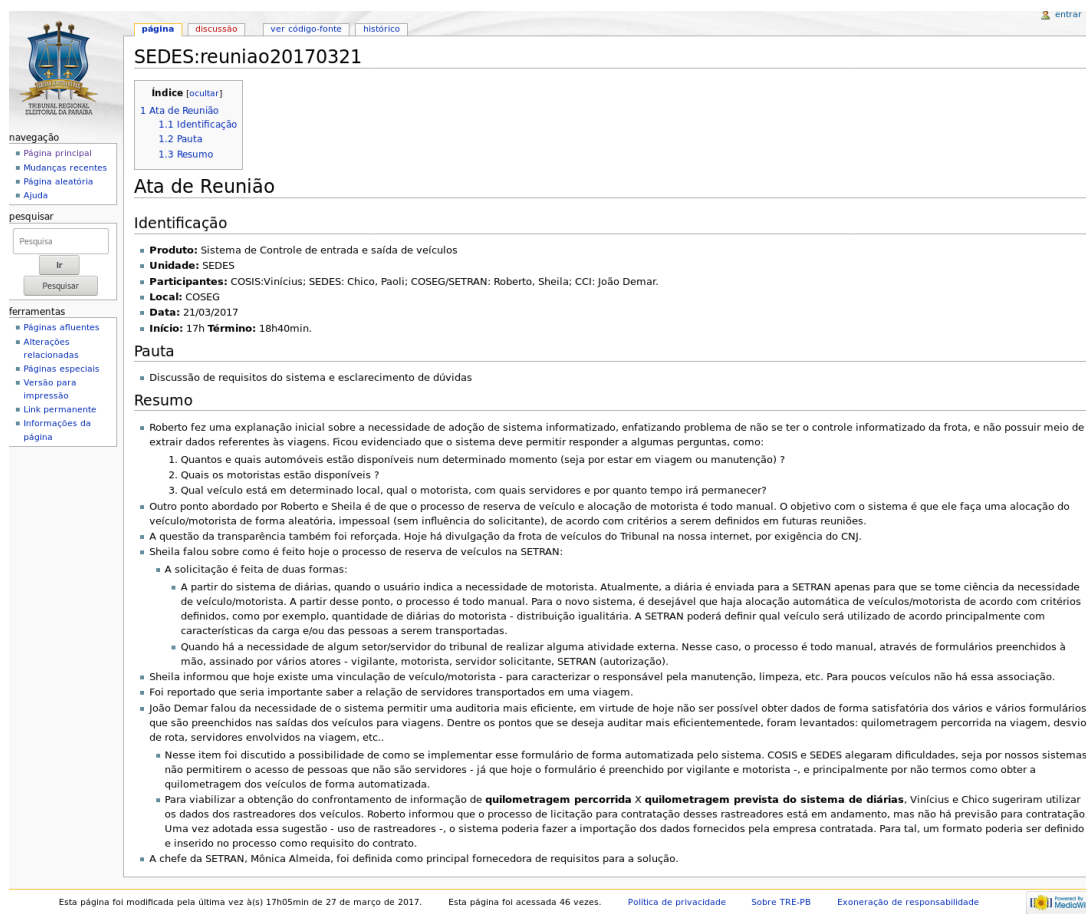
### 3.2 Planejamento

Todos os membros do time se reúnem para medir o tamanho de cada atividade. Uma a uma é analisada e discutida por cada membro do time onde cada um expõe e justifica o tamanho dado através da técnica do Planning Poker. Após um consenso entre todos fica definido o tamanho da atividade. Assim o gerente com esses dados em mãos formaliza o projeto como uma demanda de serviço, contendo o prazo o time necessário para concluir as releases.

Após a autorização, o gestor do sistema promoverá a reunião de partida entre o time de desenvolvimento e as partes interessadas, momento em que será esclarecido o problema de negócio, definidos papéis, explicado o processo de desenvolvimento e distribuídas responsabilidades.(CAVALCANTI, 2017, p. 2)

As releases definidas são apresentadas como as versões que serão entregues e são separadas conforme ordem de prioridade das atividades e requisitos como mostra a Figura 4.

Figura 3 – Ata de Reunião - 21/03/2017



**SEDES:reuniao20170321**

**Índice** [ocultar]

- 1 Ata de Reunião
  - 1.1 Identificação
  - 1.2 Pauta
  - 1.3 Resumo

**Ata de Reunião**

**Identificação**

- Produto:** Sistema de Controle de entrada e saída de veículos
- Unidade:** SEDES
- Participantes:** COSIS: Vinícius; SEDES: Chico, Paoli; COSEG/SETRAN: Roberto, Sheila; CCI: João Demar.
- Local:** COSEG
- Data:** 21/03/2017
- Início:** 17h **Término:** 18h40min.

**Pauta**

- Discussão de requisitos do sistema e esclarecimento de dúvidas

**Resumo**

- Roberto fez uma explanação inicial sobre a necessidade de adoção de sistema informatizado, enfatizando problema de não se ter o controle informatizado da frota, e não possuir meio de extrair dados referentes às viagens. Ficou evidenciado que o sistema deve permitir responder a algumas perguntas, como:
  - Quanto e quais automóveis estão disponíveis num determinado momento (seja por estar em viagem ou manutenção) ?
  - Quais os motoristas estão disponíveis ?
  - Qual veículo está em determinado local, qual o motorista, com quais servidores e por quanto tempo irá permanecer?
- Outro ponto abordado por Roberto e Sheila é de que o processo de reserva de veículo e alocação de motorista é todo manual. O objetivo com o sistema é que ele faça uma alocação do veículo/motorista de forma aleatória, impessoal (sem influência do solicitante), de acordo com critérios a serem definidos em futuras reuniões.
- A questão da transparência também foi reforçada. Hoje há divulgação da frota de veículos do Tribunal na nossa internet, por exigência do CNJ.
- Sheila falou sobre como é feito hoje o processo de reserva de veículos na SETRAN:
  - A solicitação é feita de duas formas:
    - A partir do sistema de diárias, quando o usuário indica a necessidade de motorista. Atualmente, a diária é enviada para a SETRAN apenas para que se tome ciência da necessidade de veículo/motorista. A partir desse ponto, o processo é todo manual. Para o novo sistema, é desejável que haja alocação automática de veículos/motorista de acordo com critérios definidos, como por exemplo, quantidade de diárias do motorista - distribuição igualitária. A SETRAN poderá definir qual veículo será utilizado de acordo principalmente com características da carga e/ou das pessoas a serem transportadas.
    - Quando há a necessidade de algum setor/servidor do tribunal de realizar alguma atividade externa. Nesse caso, o processo é todo manual, através de formulários preenchidos à mão, assinado por vários atores - vigilante, motorista, servidor solicitante, SETRAN (autorização).
  - Sheila informou que hoje existe uma vinculação de veículo/motorista - para caracterizar o responsável pela manutenção, limpeza, etc. Para poucos veículos não há essa associação.
  - Foi reportado que seria importante saber a relação de servidores transportados em uma viagem.
  - João Demar falou da necessidade de o sistema permitir uma auditoria mais eficiente, em virtude de hoje não ser possível obter dados de forma satisfatória dos vários e vários formulários que são preenchidos nas saídas dos veículos para viagens. Dentre os pontos que se deseja auditar mais eficientemente, foram levantados: quilometragem percorrida na viagem, desvio de rota, servidores envolvidos na viagem, etc..
  - Nesse item foi discutido a possibilidade de como se implementar esse formulário de forma automatizada pelo sistema. COSIS e SEDES alegaram dificuldades, seja por nossos sistemas não permitirem o acesso de pessoas que não são servidores - já que hoje o formulário é preenchido por vigilante e motorista -, e principalmente por não termos como obter a quilometragem dos veículos de forma automatizada.
  - Para viabilizar a obtenção do confronto de informação de **quilometragem percorrida X quilometragem prevista do sistema de diárias**, Vinícius e Chico sugeriram utilizar os dados dos rastreadores dos veículos. Roberto informou que o processo de licitação para contratação desses rastreadores está em andamento, mas não há previsão para contratação. Uma vez adotada essa sugestão - uso de rastreadores -, o sistema poderia fazer a importação dos dados fornecidos pela empresa contratada. Para tal, um formato poderia ser definido e inserido no processo como requisito do contrato.
  - A chefe da SETRAN, Mônica Almeida, foi definida como principal fornecedora de requisitos para a solução.

Esta página foi modificada pela última vez às(s) 17h05min de 27 de março de 2017. Esta página foi acessada 46 vezes. [Política de privacidade](#) [Sobre TRE-PB](#) [Exoneração de responsabilidade](#)

Fonte: SEDES

### 3.3 Desenvolvimento

No início da fase de desenvolvimento são gerados os primeiros artefatos em código que serão comuns para todos os membros do time. O TRE-PB utiliza para controle de versões de seus códigos o sistema de versionamento denominado Subversion (SVN). Um novo repositório é criado para o projeto e um novo projeto com um arcabouço contendo as bibliotecas padrões e um template para as páginas já utilizadas pela SEDES é criado disponibilizado no repositório para todos os desenvolvedores. A SISBAN através de um chamado gera um novo schema no banco de dados Oracle 12g e fornece acesso a todos os desenvolvedores.

Todo esse processo é inicialmente implementado pelo supervisor técnico que geralmente é o desenvolvedor responsável pelo projeto e com mais habilidades e experiência do time.

Nesse momento todas as histórias já estão descritas com uma riqueza maior de detalhes, então são criados tickets e colados em um taskboard na forma de um quadro Kanban visível para todos. Esse quadro contém todas as atividades necessárias que atendem ao escopo do projeto para a release em questão.

Diariamente são realizadas pequenas reuniões denominadas "daily" entre o gestor do

projeto e membros do time com o intuito de detalhar o andamento das atividades, conduzir novas atividades para o time bem como relatar as dificuldades encontradas.

Na [Figura 5](#) a seguir é possível ver uma tarefa específica que foi atribuída a mim. A tarefa cadastrada no Redmine possui uma ligação com o repositório SVN, assim é possível ver e analisar trechos de códigos que foram implementados em cada "commit"<sup>1</sup>.

---

<sup>1</sup>No contexto de ciência da computação e gerenciamento de dados, commit refere-se à ideia de fazer permanentes um conjunto de mudanças experimentais. Uma utilização popular está no fim de uma transação. Um commit é o ato de enviar. <https://pt.wikipedia.org/wiki/Commit>

Figura 4 – Planejamento das versões

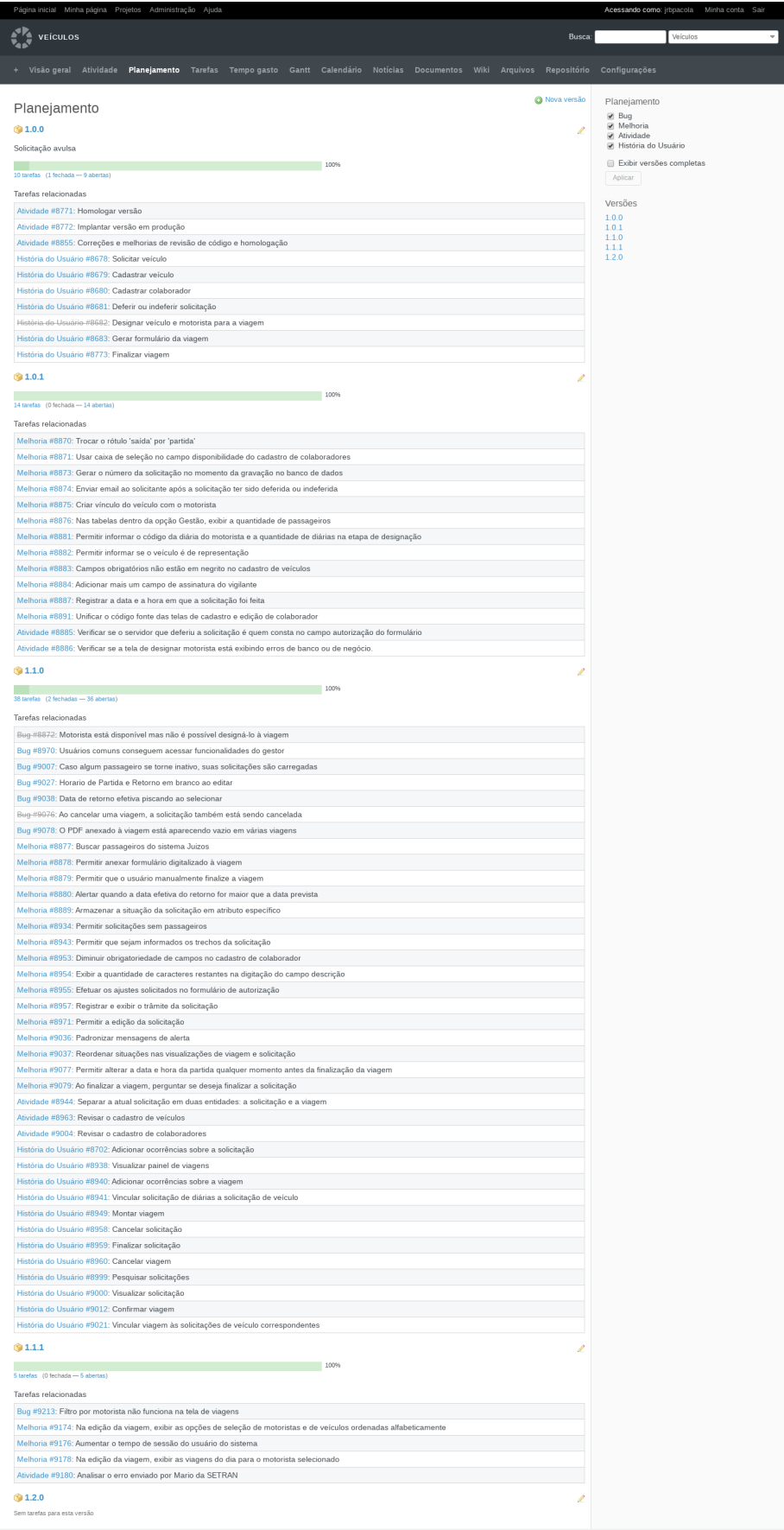




Figura 5 – Atividade 8940

VEÍCULOS

Busca

Veículos

Visão geral

Atividade

Planejamento

Tarefas

Tempo gasto

Gantt

Calendário

Notícias

Documentos

Wiki

Arquivos

Repositório

Configurações

Acessando como jltpacola

Minha conta

Sair

História do Usuário #8940

Editar

Tempo de trabalho

Observar

Copiar

Excluir

Adicionar ocorrências sobre a viagem

Adicionado por Rômulo 9 meses atrás. Atualizado 8 meses atrás.

« Anterior | 11/58 | Próximo »

Situação: Resolvido

Prioridade: Normal

Atribuído para: José Ricardo

Versão: 1.1.0

Sprint:

Início:

Data prevista:

% Terminado: 100%

Tempo estimado:

Tamanho:

Descrição

Com o objetivo de registrar qualquer evento significativo sobre a viagem, o gestor pode adicionar ocorrências a uma viagem.

Critérios de aceitação:

- Para adicionar uma ocorrência, o usuário deve descrevê-la e opcionalmente incluir um anexo.
- As ocorrências serão adicionadas e exibidas na tela de visualização da viagem.
- Devem ser registrados o usuário, a data e a hora do assentamento da ocorrência.
- Uma ocorrência pode ser adicionada a qualquer tempo, inclusive se a viagem estiver finalizada.

Subtarefas

Adicionar

Tarefas relacionadas

Adicionar

relacionado a Veículos - História do Usuário #8949: Montar viagem

Resolvido

Histórico

Atualizado por Rômulo há 9 meses

#1

- Tipo alterado de História do Usuário para Melhoria

Atualizado por Rômulo há 8 meses

#2

- Tipo alterado de Melhoria para História do Usuário
- Versão alterado de 1.2.0 para 1.1.0

Atualizado por Rômulo há 8 meses

#3

- Título alterado de Incluir assentamentos sobre a solicitação ou viagem para Adicionar ocorrências sobre a solicitação ou viagem

Atualizado por José Ricardo há 8 meses

#4

- Atribuído para ajustado para José Ricardo

Atualizado por José Ricardo há 8 meses

#5

- % Terminado alterado de 0 para 30

Atualizado por Rômulo há 8 meses

#6

- Título alterado de Adicionar ocorrências sobre a solicitação ou viagem para Adicionar ocorrências sobre a viagem

Atualizado por Pedro Henrique há 8 meses

#7

- % Terminado alterado de 30 para 70

Atualizado por Pedro Henrique há 8 meses

#8

- Situação alterado de Novo para Resolvido

Atualizado por Rômulo há 8 meses

#9

- % Terminado alterado de 70 para 100

Revisões associadas

Revisão 22874 (diff)

Adicionado por Rômulo 8 meses atrás

#8940 Criação da tabela de ocorrências da viagem.

Revisão 22886 (diff)

Adicionado por José Ricardo 8 meses atrás

#8940 - Criação da Ocorrência. Inclusão no visualizar Viagem. Remoção da exibição trâmite em incluirEditarViagem. Ajustes no dialog de visualizarSolicitacao.

Revisão 22887 (diff)

Adicionado por Rômulo 8 meses atrás

#8878 - #8940 Cria da tabela de anexos e adicionados os campos de anexo na viagem e na ocorrência da viagem.

Revisão 22903 (diff)

Adicionado por José Ricardo 8 meses atrás

#8940 -Inclusão do upload do arquivo na ocorrência da viagem

Revisão 22911 (diff)

Adicionado por José Ricardo 8 meses atrás

#8940 - Ajustes upload arquivo ocorrencia

Revisão 22933 (diff)

Adicionado por José Ricardo 8 meses atrás

#8940 - ajustes no upload do arquivo

Exportar para Atom | PDF

Powered by Redmine © 2006-2017 Jean-Philippe Lang

Fonte: SEDES

## 4 CONCLUSÃO

Parte final do texto, na qual se apresentam as conclusões do trabalho acadêmico. É importante fazer uma análise crítica do trabalho, destacando os principais resultados e as contribuições do trabalho para a área de pesquisa.

### 4.1 TRABALHOS FUTUROS

Também deve indicar, se possível e/ou conveniente, como o trabalho pode ser estendido ou aprimorado.

### 4.2 CONSIDERAÇÕES FINAIS

Encerramento do trabalho acadêmico.

## Referências

- BISSI, W. **Metodologia de desenvolvimento ágil**. 2. ed. Cidade: Campo Digital, 2007. Citado na página 4.
- CAVALCANTI, A. S. Portaria diretoria-geral no 37/2017 tre-pb/ptre/dg. **Tribunal Regional Eleitoral da Paraíba**, n. 37, 2017. Citado na página 9.
- CIVICI, C. gatay. **Primefaces User Guide 5.2**. [S.l.], 2015. 595 p. Disponível em: <[https://www.primefaces.org/docs/guide/primefaces\\_user\\_guide\\_5\\_2.pdf](https://www.primefaces.org/docs/guide/primefaces_user_guide_5_2.pdf)>. Acesso em: 28 de janeiro de 2019. Citado na página 6.
- CORDEIRO, G. **Aplicações Java para Web com JSF e JPA**. 1. ed. São Paulo: Casa do Código, 2014. Citado 2 vezes nas páginas 6 e 7.
- DEITEL, P. e. H. **Java Como Programar**. 8. ed. Cidade: Pearson, 2009. Citado na página 5.
- SABBAGH, R. **Scrum**. 1. ed. Cidade: Casa do Código, 2014. Citado na página 5.

## Apêndices

## **APÊNDICE A – Nome do apêndice**

Lembre-se que a diferença entre apêndice e anexo diz respeito à autoria do texto e/ou material ali colocado.

Caso o material ou texto suplementar ou complementar seja de sua autoria, então ele deverá ser colocado como um apêndice. Porém, caso a autoria seja de terceiros, então o material ou texto deverá ser colocado como anexo.

Caso seja conveniente, podem ser criados outros apêndices para o seu trabalho acadêmico. Basta recortar e colar este trecho neste mesmo documento. Lembre-se de alterar o "label" do apêndice.

Não é aconselhável colocar tudo que é complementar em um único apêndice. Organize os apêndices de modo que, em cada um deles, haja um único tipo de conteúdo. Isso facilita a leitura e compreensão para o leitor do trabalho.

**APÊNDICE B – Nome do outro apêndice**

conteúdo do novo apêndice

Anexos

**ANEXO A – Modus 3.1**





**Tribunal Regional Eleitoral da Paraíba**  
**Coordenadoria de Sistemas**  
**Seção de Análise e Desenvolvimento de Sistemas**

## **Modus 3.1 - Modelo de desenvolvimento de software**

### **Introdução**

O Modus é o processo de desenvolvimento de software adotado pela SEDES. Atualmente, o processo está na sua terceira versão. Optou-se pela elaboração de um processo mais enxuto, baseado em práticas que vem sendo adotadas ordinariamente pela unidade.

Não é objetivo deste documento explicar em detalhes as práticas adotadas pela SEDES oriundas de processos de desenvolvimento ágil, abordagem bastante utilizada atualmente em diversas instituições e empresas, com bastante eficiência. Apenas a título de informação, são utilizadas práticas e técnicas de algumas metodologias ágeis, tais como: Scrum, XP (extreme programming) e FDD (feature driving development).

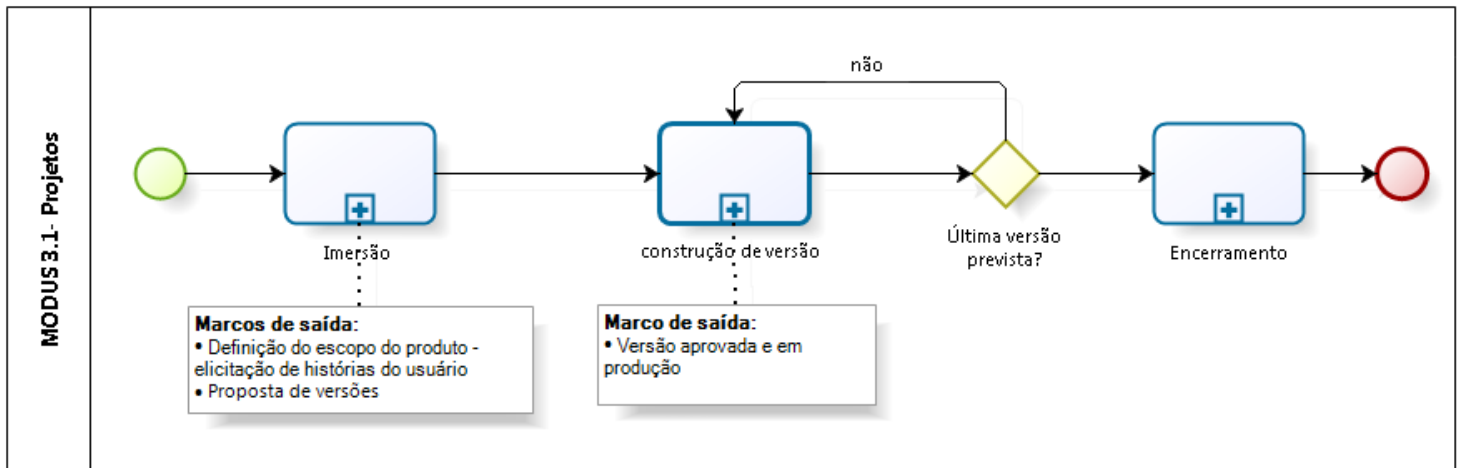
O modelo de processo utilizado é iterativo e incremental: as versões do produto são homologadas pelo cliente e melhorias e novas funcionalidades podem ser adicionadas durante o projeto, conforme feedback dos usuários.

Dentre as práticas e técnicas, destacam-se: reunião diária (stand-up meeting ou daily meeting), uso de taskboard, entregas frequentes (disponibilização de versões com funcionalidades já implementadas em curto espaço de tempo - tipicamente um mês), elicitação de requisitos através de histórias de usuários, planning poker (técnica para estimar tamanho - tempo - a ser dedicado a uma história de usuário) e melhoria contínua (lições aprendidas do projeto servem para a melhoria do processo e do modelo de desenvolvimento empregado).

## Papéis

Papel	Descrição
Gerente do projeto	É a pessoa responsável pela condução do projeto, planejando e coordenando o desenvolvimento, mantendo o time motivado e resolvendo impedimentos e conflitos de interesses que possam prejudicar o andamento do projeto. Dentre suas atribuições está o papel de <i>Scrum Master</i> do método ágil <i>Scrum</i> .
Time	É uma equipe formada por desenvolvedores que executarão as atividades de análise, construção e de manutenção dos produtos, sob a coordenação do gerente de projeto
Cliente	São pessoas interessadas no projeto que fornecem os requisitos para o time e homologam as entregas
Gestor do sistema	É uma pessoa destacada do grupo cliente com bastante interesse pelo projeto, disponibilidade e influência suficiente para advogar em favor dos propósitos do projeto. Junto ao time ele deve decidir sobre requisitos conflitantes e estabelecer prioridades. Deve ser designado por portaria.
Principal fornecedor de requisitos	É a pessoa do grupo cliente que detém um bom domínio do negócio e com maior disponibilidade para fornecer os requisitos ao time e esclarecer dúvidas. Em geral esse papel é assumido pelo gestor do sistema.

## Modus Projetos



*Ilustração 1: Fases do modelo de desenvolvimento para projetos*

### Imersão

O propósito desta etapa é realizar uma imersão no domínio do cliente, identificar claramente a demanda e os benefícios que ela deverá agregar. Com base nessas informações, o time deve elicitar as principais histórias do usuário e elaborar um plano macro e uma proposta de prazo para finalização do projeto.

Dentro da proposta de desenvolvimento ágil, obviamente, novas histórias e alterações no plano de versões podem ocorrer naturalmente no decorrer do projeto.

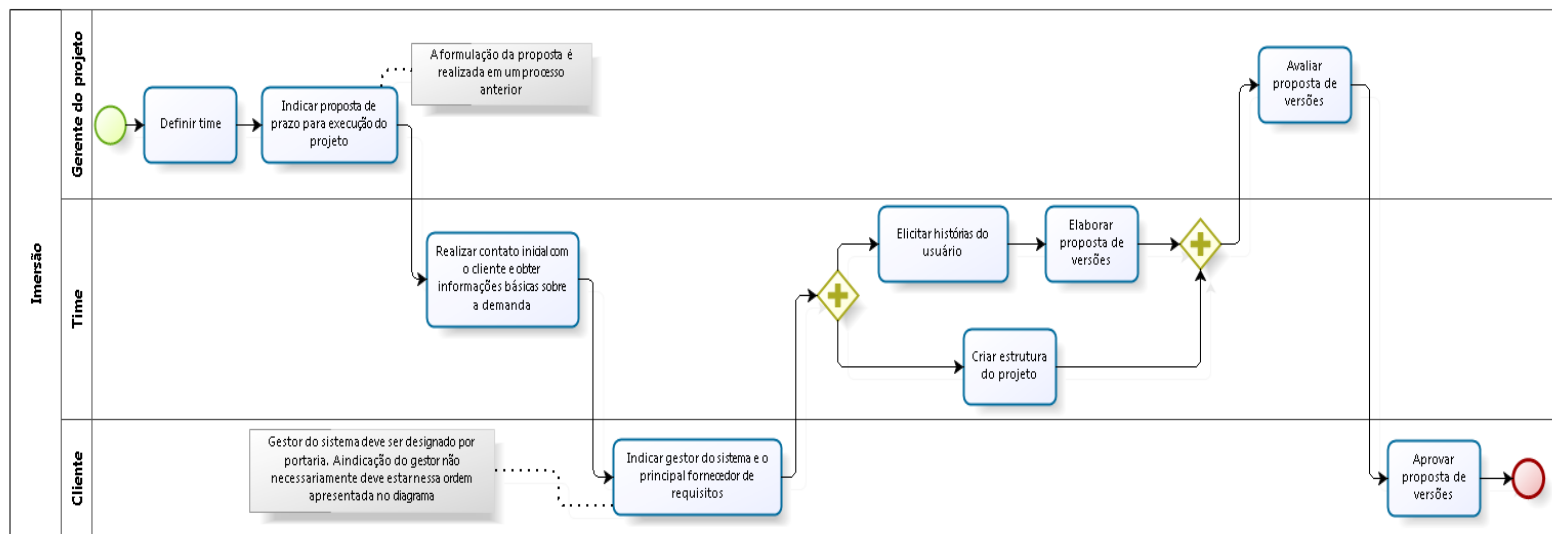


Ilustração 2: Fluxo da fase de imersão

## Definir time

### Ator: Gerente do projeto

Após a aprovação do desenvolvimento do projeto (realizada em um processo anterior), o gerente define o time do projeto, ou seja, quais pessoas atuarão na análise e construção do produto.

Entradas	Saídas
<ul style="list-style-type: none"> <li>Projeto aprovado para execução</li> </ul>	<ul style="list-style-type: none"> <li>Determinação do time (pessoas que executarão o projeto)</li> </ul>

## Indicar proposta de prazo para execução do projeto

### Ator: Gerente do projeto

O gerente deve informar ao time qual o prazo proposto pela STI e COSIS para realização do projeto. A definição dessa proposta é realizada em um processo anterior ao de desenvolvimento. Essa proposta de prazo poderá ser ajustada posteriormente com o time e também com o cliente, mas serve de base para a definição do escopo do projeto.

Entradas	Saídas
<ul style="list-style-type: none"> <li>Calendário de projetos da COSIS</li> </ul>	<ul style="list-style-type: none"> <li>Indicação para o time do prazo sugerido</li> </ul>

## Realizar contato inicial com o cliente e obter informações básicas sobre a demanda

### Ator: Time

Durante a execução do projeto são realizadas várias reuniões com o cliente, mas esse contato inicial destaca-se por ser um marco para o projeto. Nele, além de conhecer o cliente, o time e o gerente poderão extrair a ideia principal do produto e que benefícios ele visa alcançar.

Entradas	Saídas
Projeto aprovado para execução	<ul style="list-style-type: none"> <li>Ata de reunião inicial</li> </ul>

## Indicar gestor do sistema e o principal fornecedor de requisitos

### Ator: Cliente

Nas reuniões iniciais ou até antes delas, a pessoa que desempenhará o papel de gestor do sistema deve ser definida dentre os clientes. Portaria específica deve ser elaborada para designação do gestor do sistema e seu suplente.

Junto ao time, o gestor será responsável por determinar prioridades e atuar como referência no fornecimento de requisitos e no esclarecimento de dúvidas sobre o negócio.

Entradas	Saídas
Projeto aprovado para execução	<ul style="list-style-type: none"> <li>Definição do gestor do sistema e do principal fornecedor de requisitos para o time</li> </ul>

## Criar estrutura do projeto

### Ator: Time

Esta tarefa consiste em criar os artefatos estruturais do projeto:

- Projeto na ferramenta Redmine (<http://redmine.tre-pb.gov.br/>)
- Inclusão do produto no catálogo da COSIS (<http://redmine.tre-pb.gov.br/projects/cat-tec-sist>)
- Página do produto
- Arcabouço do sistema armazenado no repositório SVN (<http://svn.tre-pb.gov.br/svn/cosis>) e gerado a partir de arquétipo de arquitetura padrão das aplicações.

Entradas	Saídas
Projeto aprovado para execução	<ul style="list-style-type: none"> <li>Projeto no Redmine</li> <li>Página do projeto</li> <li>Inclusão do produto no catálogo</li> <li>Arcabouço do sistema no SVN</li> </ul>

## Elicitar histórias do usuário

### Ator: Time

Histórias de Usuário devem ser identificadas a partir dos registros em atas de reunião. São relatos de funcionalidades que agregam valor para o cliente. Assim, o time deve traduzir as expectativas e demandas do cliente em Histórias do Usuário.

Nesta fase, é necessária apenas uma descrição sucinta de cada história de usuário identificada durante os contatos com o cliente. Convém destacar que a ideia geral já deva estar descrita em ata de reunião – ou outro artefato – já realizada com o cliente.

As histórias são cadastradas como *tickets* dentro do projeto na ferramenta Redmine, ainda sem o detalhamento necessário a ser elaborado em outra etapa do processo. O conjunto dessas histórias determina o escopo base do projeto.

Entradas	Saídas
<ul style="list-style-type: none"> <li>Projeto aprovado para execução</li> <li>Ata de reunião inicial</li> </ul>	<ul style="list-style-type: none"> <li>Histórias cadastradas no Redmine</li> </ul>

## Elaborar proposta de versões

### Ator: Time

Com base nas histórias criadas, na priorização do cliente e ainda no prazo sugerido pelo gerente do projeto, o time elabora uma proposta de plano de versões do produto.

Cada versão é uma unidade funcional do sistema em produção contemplando um conjunto de histórias.

Entradas	Saídas
<ul style="list-style-type: none"> <li>Histórias elicítadas no Redmine</li> <li>Priorização do cliente</li> <li>Sugestão de prazo de conclusão do projeto</li> </ul>	<ul style="list-style-type: none"> <li>Proposta de versões do projeto no Redmine, indicando o escopo para cada uma delas.</li> </ul>

## Avaliar proposta de versões

### Ator: Gerente do projeto

Após a elaboração da proposta de versões, que corresponde ao plano de entregas do produto para o cliente, o time deve apresentar o resultado para o gerente do projeto.

A proposta pode apresentar um prazo final diferente da sugestão inicial do gerente e o time deve apontar as justificativas. O gerente então avalia o planejamento, aponta sugestões e todos definem a proposta final a ser entregue ao cliente.

Entradas	Saídas
<ul style="list-style-type: none"> <li>Histórias cadastradas no Redmine</li> <li>Proposta de versões do projeto no Redmine</li> </ul>	<ul style="list-style-type: none"> <li>Proposta de versões do projeto no Redmine acordada entre o time e o gerente</li> </ul>

### Aprovar proposta de versões

#### Ator: Cliente

A proposta de versões é apresentada e discutida com o cliente. Todos devem chegar a um consenso que é referendado com a aprovação da proposta pelo cliente.

Entradas	Saídas
<ul style="list-style-type: none"> <li>Proposta de versões do projeto no Redmine acordada entre o time e o gerente</li> </ul>	<ul style="list-style-type: none"> <li>Proposta de versões do projeto no Redmine acordada entre o time, o gerente e o cliente</li> </ul>

### Construção

Execução de atividades usando princípios de metodologias ágeis para construção e entrega de uma versão funcional para o cliente. O acompanhamento diário das atividades pode ser feito através do *taskboard*.

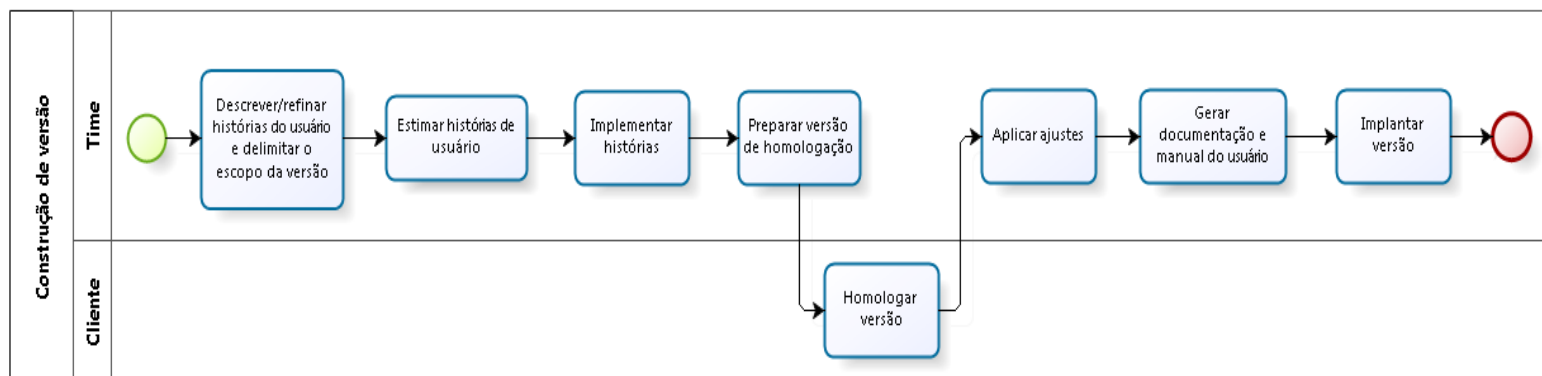


Ilustração 3: Fluxo de atividades da fase de construção de versão

### Descrever/refinar histórias do usuário e delimitar o escopo da versão

#### Ator: Time

As histórias de usuário já elicitadas deverão ser atualizadas de forma detalhada na ferramenta Redmine. Numa história deve ser indicado o propósito

que ela visa atender, que papel dentro do sistema executa a história e o que ela realiza.

São indicadas também as condições esperadas e as ações que deverão ser executadas. Com essas informações, são estabelecidos critérios de aceitação, sob os quais pode-se avaliar se o propósito foi atendido.

Com a evolução do time no domínio do negócio e com a experiência do cliente após as entregas das primeiras versões, pode ser necessário o refinamento das histórias para alinhar a versão a ser construída com as expectativas do cliente, bem como delimitar o escopo da versão em decorrência de mudanças no planejamento inicial. As histórias do usuário poderão ser ajustadas conforme entendimento entre o time e o cliente.

Entradas	Saídas
<ul style="list-style-type: none"> <li>Histórias do usuário propostas para a versão</li> <li>Feedback do cliente</li> </ul>	<ul style="list-style-type: none"> <li>Histórias do usuário refinadas no Redmine</li> <li>Escopo da versão ajustado</li> </ul>

## Estimar histórias

### Ator: Time

O time deve discutir as histórias e estimar o esforço de realização de cada uma delas individualmente. Todo o esforço despendido para analisar (se ainda existirem detalhes não especificados), implementar, revisar e documentar a história deve ser avaliado.

Esta tarefa é realizada com base na técnica conhecida como *Planning Poker* e a unidade de esforço é um dia de trabalho de um desenvolvedor.

Entradas	Saídas
<ul style="list-style-type: none"> <li>Histórias escritas no Redmine</li> </ul>	<ul style="list-style-type: none"> <li>Estimativas indicadas em cada história no Redmine</li> </ul>

## Implementar histórias

### Ator: Time

Implementação das histórias do usuário tendo como base os padrões de arquitetura, de banco de dados e de design estabelecidos, fazendo uso de técnicas de desenvolvimento ágil do Scrum, como por exemplo o uso de *taskboard* e a realização de reuniões diárias de acompanhamento. As histórias do usuário são divididas em atividades diárias que serão executadas e em seguida revisadas por outro membro do time. A revisão das atividades objetiva minimizar os defeitos e as não conformidades com a especificação da história.

Os artefatos produzidos (código fonte, modelos de dados, scripts, etc.) são armazenados no repositório SVN e associados ao número da história no Redmine para possibilitar a rastreabilidade.

Entradas	Saídas
----------	--------



<ul style="list-style-type: none"> <li>• Histórias do usuário detalhadas e refinadas selecionadas para a versão</li> <li>• Diagrama de modelagem de processos de negócio, se houver</li> <li>• Protótipos e outros artefatos produzidos na etapa de imersão</li> </ul>	<ul style="list-style-type: none"> <li>• Código-fonte da versão</li> <li>• Scripts de criação/atualização do banco de dados</li> <li>• Modelo entidade-relacionamento</li> </ul>
--	--

## Preparar versão de homologação

### Ator: Time

Construção da versão e implantação em ambiente de homologação para que o cliente valide as histórias implementadas. O banco de dados de homologação deve ser criado, ou atualizado, e as histórias implementadas devem ser testadas minimamente pelo time em ambiente de homologação para evitar falhas de configuração.

O time também deve preparar os dados básicos para que o cliente possa realizar os testes de aceitação. Dependendo da complexidade do sistema, diversidade de usuários e funcionalidades envolvidas, o time poderá elaborar também oficinas de treinamento no sistema. Nesse caso, roteiros das oficinas devem ser elaborados.

Entradas	Saídas
<ul style="list-style-type: none"> <li>• Código-fonte da versão</li> <li>• Scripts de banco de dados</li> </ul>	<ul style="list-style-type: none"> <li>• Versão implantada no ambiente de homologação</li> <li>• Base de dados para testes preparada</li> <li>• Roteiros para oficinas, se houver</li> </ul>

## Homologar versão

### Ator: Cliente

A equipe apresenta a versão ao cliente para que realize os testes de aceitação. O cliente avaliará principalmente a conformidade com o que foi solicitado. O cliente analisará também aspectos não-funcionais como a usabilidade do sistema, o tempo de resposta e a segurança.

Além da reunião com o time, a versão de homologação é disponibilizada para exploração dos usuários por alguns dias. Também podem ser realizadas oficinas de treinamento para clientes e usuários do sistema.

Ao final desse processo, o cliente deve dar sua aprovação à entrega ou indicar melhorias e correções necessárias.

Entradas	Saídas
<ul style="list-style-type: none"> <li>• Versão implantada no ambiente de homologação</li> </ul>	<ul style="list-style-type: none"> <li>• Feedback do cliente para o time sobre aprovação da versão e</li> </ul>

<ul style="list-style-type: none"> <li>• Base de dados para testes preparada</li> <li>• Roteiro de oficinas, se houver</li> </ul>	<ul style="list-style-type: none"> <li>• ajustes necessários</li> <li>• Ata da reunião de homologação</li> </ul>
---	--

## Aplicar ajustes

### Ator: Time

Essa tarefa consiste em implementar as melhorias e correções apontadas pelo cliente durante a homologação.

A estratégia de desenvolvimento de forma evolutiva e incremental associada a entregas frequentes, permite que não conformidades sejam identificadas e solucionadas de forma mais rápida, de modo que os ajustes apontados na homologação sejam geralmente pontuais e o time consiga fazer as adaptações sem comprometer a entrega planejada.

Entradas	Saídas
<ul style="list-style-type: none"> <li>• Feedback do cliente para o time</li> <li>• Ata da reunião de homologação</li> </ul>	<ul style="list-style-type: none"> <li>• Código-fonte ajustado</li> <li>• Histórias do usuário ajustadas, se for o caso</li> </ul>

## Gerar documentação e manual do usuário

### Ator: Time

O objetivo dessa tarefa é fazer a inclusão ou atualização do produto no catálogo de produtos, e elaboração ou atualização do manual do usuário contemplando as histórias implementadas na versão construída. Outros artefatos de documentação podem ser elaborados conforme a necessidade do sistema.

Entradas	Saídas
<ul style="list-style-type: none"> <li>• Versão implementada</li> <li>• Histórias do usuário detalhadas e refinadas selecionadas para a versão</li> </ul>	<ul style="list-style-type: none"> <li>• Catálogo de produtos atualizado</li> <li>• Manual do usuário elaborado ou atualizado</li> </ul>

## Implantar a versão

### Ator: Time

O objetivo desse passo é a implantação em ambiente de produção para uso do cliente. O banco de dados de produção deve ser criado ou atualizado e um pacote com a versão deve ser implantado no ambiente de produção.

A versão é numerada com dígitos na forma **X.Y.Z**. Os dois primeiros dígitos mais significativos (**X** e **Y**) são utilizados para incrementar o número de versão e o último (**Z**) para *patches* com correções de *bugs*.

Uma *tag* deve ser criada no sistema de controle de versões SVN para

permitir recuperar o código-fonte conforme a versão foi construída. O cliente deve ser informado que a versão está disponível para uso em produção.

Entradas	Saídas
<ul style="list-style-type: none"> <li>Código-fonte da versão</li> <li>Scripts de banco de dados</li> </ul>	<ul style="list-style-type: none"> <li>Tag no sistema de controle de versões SVN</li> <li>Versão implantada no ambiente de produção</li> <li>Comunicação via e-mail da entrada em produção para o cliente e demais interessados</li> </ul>

## Encerramento

Execução das atividades de finalização do projeto, como a formalização do responsável pelo suporte de negócio e a discussão sobre melhorias do projeto, de seu modelo/arquitetura, dos procedimentos, das técnicas e do método de desenvolvimento.

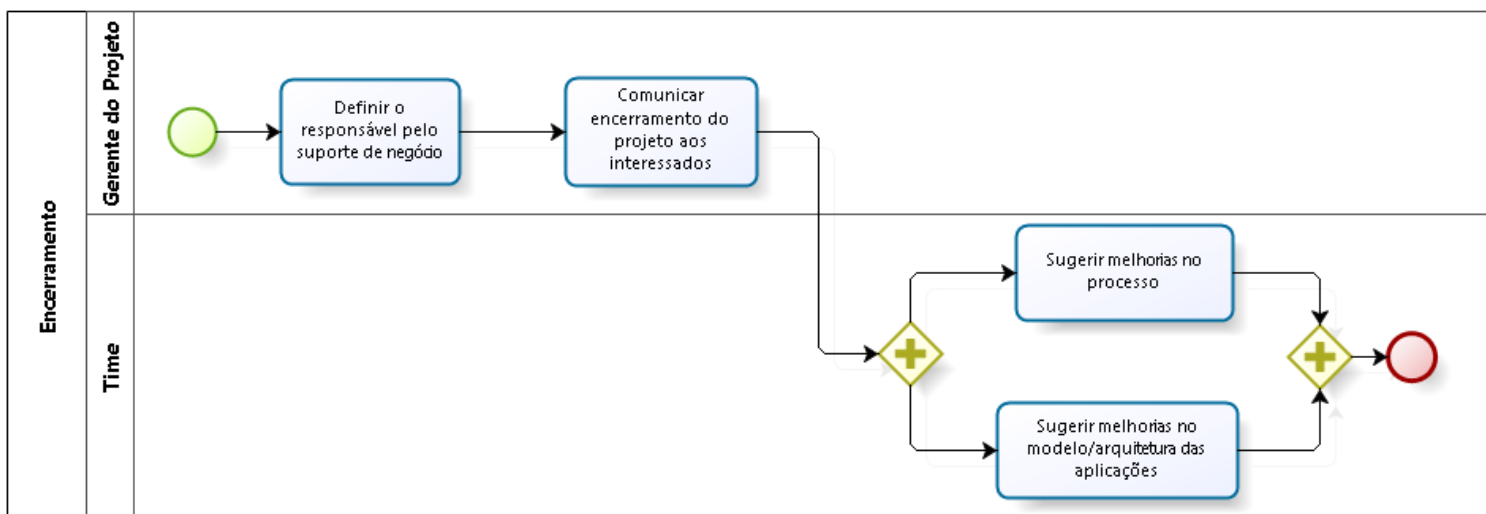


Ilustração 4: Fluxo de atividades da fase de encerramento do projeto

### Definir o responsável pelo suporte de negócio

#### Ator: Gerente do projeto

Realização de uma reunião final de entrega do produto com o time, o gerente e o cliente, na qual deverá ser definido o responsável pelo suporte de negócio do produto. Preferencialmente, o gestor do sistema, ou alguém por ele delegado, deve assumir este papel, que consiste em esclarecer as dúvidas

negociais reportadas pelos usuários do produto. À SEDES caberá o suporte técnico para correções e melhorias no sistema.

Entradas	Saídas
<ul style="list-style-type: none"> <li>Versões finalizadas</li> </ul>	<ul style="list-style-type: none"> <li>Ata da reunião</li> </ul>

### Comunicar encerramento do projeto aos interessados

#### Ator: Gerente do projeto

Comunicação aos interessados, através de e-mail ou despacho/documento em processo administrativo, informando o encerramento do projeto.

Entradas	Saídas
<ul style="list-style-type: none"> <li>Versões finalizadas</li> </ul>	<ul style="list-style-type: none"> <li>E-mail de encerramento do projeto ou documento/despacho de encerramento em processo administrativo dando publicidade aos interessados.</li> </ul>

### Sugerir melhorias no processo

#### Ator: Time

Realização de reunião entre o time e o gerente para discutir os problemas relativos ao processo de trabalho enfrentados e as soluções adotadas durante o projeto. As lições aprendidas deverão gerar oportunidades de melhorias que deverão ser incluídas no *backlog* de demandas da SEDES para implementação em momento oportuno e adoção em futuros projetos.

Entradas	Saídas
<ul style="list-style-type: none"> <li>Problemas enfrentados e soluções adotadas</li> </ul>	<ul style="list-style-type: none"> <li>Melhorias cadastradas no Redmine</li> </ul>

### Sugerir melhorias no modelo/arquitetura das aplicações

#### Ator: Time

Realização de reunião entre o time e o gerente para avaliação de sugestões de melhorias no modelo/arquitetura das aplicações. Serão discutidos os problemas enfrentados e as soluções adotadas durante o projeto. As melhorias serão incluídas no *backlog* de demandas da SEDES para implementação em momento oportuno.

Entradas	Saídas
<ul style="list-style-type: none"> <li>Problemas enfrentados e soluções adotadas</li> </ul>	<ul style="list-style-type: none"> <li>Melhorias cadastradas no Redmine</li> </ul>

## Modus Manutenções

Processo de manutenção de produtos para implementação de melhorias ou correções provenientes do *backlog* de demandas mantido no projeto SEDES Demandas do Redmine. Compreende as atividades realizadas após a seleção da demanda para ser executada, até a liberação do produto para produção. As atividades de cadastro de demandas, priorização e seleção para execução são realizadas em um processo anterior.

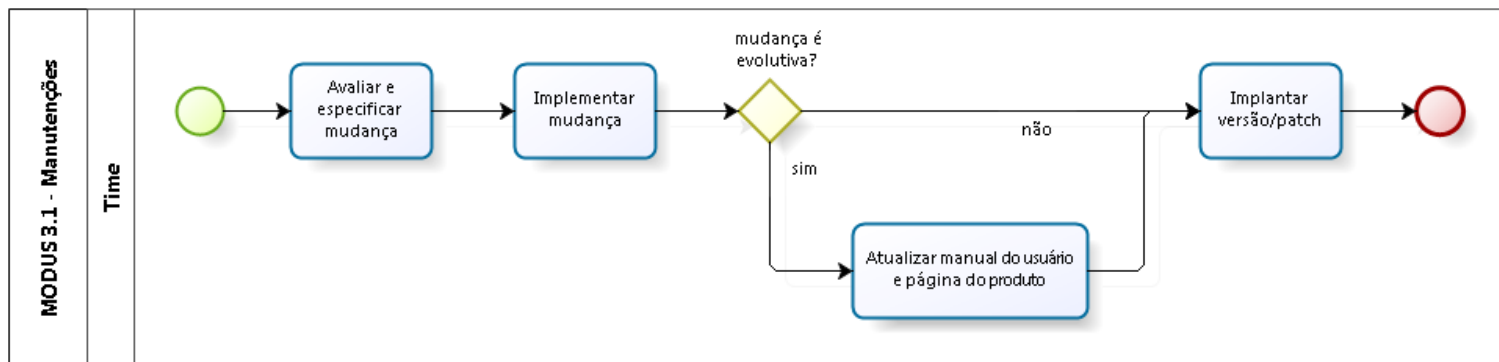


Ilustração 5: Fluxo de atividades do processo de manutenção de produtos

### Avaliar e especificar mudança

#### Ator: Time

Uma demanda acarretará em mudanças no produto com a incorporação de novas histórias do usuário, melhorias ou correções em histórias existentes. O time deve analisar mais detalhadamente o que foi solicitado na demanda e especificar a mudança, registrando novas histórias, melhorias ou correções no projeto correspondente ao produto no Redmine.

A tarefa de melhoria ou correção de bug deve obrigatoriamente ser associada a história do usuário correspondente no Redmine.

A mudança deverá gerar uma nova versão ou *patch* do produto. Se a mudança for evolutiva, uma nova versão deve ser criada incrementando o segundo dígito identificador em relação a versão anterior. Ou seja, se a versão anterior era a 1.2.4, agora a nova versão será 1.3.0. Se a mudança for apenas corretiva, deve ser gerado um novo *patch*, incrementando o terceiro dígito

identificador. Tomando o exemplo anterior, o *patch* seria nomeado como 1.2.5.

Por fim, com base na avaliação da mudança, o time poderá informar ao chefe da unidade e ao cliente qual o prazo de entrega da mudança.

Entradas	Saídas
<ul style="list-style-type: none"> <li>• Demanda selecionada</li> </ul>	<ul style="list-style-type: none"> <li>• Mudança especificada no Redmine como novas histórias do usuário, melhorias ou bugs</li> <li>• Prazo para entrega</li> <li>• Identificação do <i>patch</i> ou versão do produto a ser gerado</li> </ul>

### Implementar mudança

#### Ator: Time

Essa atividade é semelhante a atividade *Implementar histórias* descrita na fase *Construção da versão* do processo *Modus - Projetos*.

Entradas	Saídas
<ul style="list-style-type: none"> <li>• Mudança especificada no Redmine como novas histórias do usuário, melhorias ou bugs</li> </ul>	<ul style="list-style-type: none"> <li>• Código-fonte da versão</li> <li>• Scripts de criação/atualização do banco de dados</li> <li>• Modelo entidade-relacionamento</li> </ul>

**Atualizar manual do usuário e página do produto****Ator: Time**

Se a mudança é evolutiva, ou seja, novas características foram incorporadas, o manual do usuário e a página do produto precisam ser atualizados.

Entradas	Saídas
<ul style="list-style-type: none"> <li>Mudança especificada no Redmine como novas histórias do usuário, melhorias ou bugs</li> <li>Mudança implementada</li> </ul>	<ul style="list-style-type: none"> <li>Catálogo de produtos atualizado</li> <li>Manual do usuário atualizado</li> </ul>

**Implantar versão/patch****Ator: Time**

O objetivo dessa atividade é a implantação do *patch* ou versão em ambiente de produção para uso do cliente. Nela são realizadas as mesmas tarefas e procedimentos descritos na atividade *Implantar versão* da fase *Construção da versão* do processo *Modus - Projetos*.

Entradas	Saídas
<ul style="list-style-type: none"> <li>Código-fonte da versão</li> <li>Scripts de banco de dados</li> </ul>	<ul style="list-style-type: none"> <li>Tag no sistema de controle de versões SVN</li> <li>Versão implantada no ambiente de produção</li> <li>Comunicação via e-mail da entrada em produção para o cliente e demais interessados</li> </ul>

**ANEXO B – Portaria 37/2017**





TRIBUNAL REGIONAL ELEITORAL DA PARAÍBA

## **PORTARIA DIRETORIA-GERAL Nº 37/2017 TRE-PB/PTRE/DG**

O DIRETOR GERAL DO TRIBUNAL REGIONAL ELEITORAL DA PARAÍBA, no uso de suas atribuições legais e regimentais,

CONSIDERANDO a necessidade de aprimorar os padrões de governança em Tecnologia da Informação no Tribunal Regional Eleitoral da Paraíba;

CONSIDERANDO que o modelo MPS.BR é baseado nas melhores práticas de Engenharia de Software reconhecidas pela comunidade internacional, compatíveis com o modelo CMMI (padrão da indústria de software internacional) e em consonância com normas internacionais de qualidade (ISO/IEC 12207 - processos do ciclo de vida do software, ISO/IEC 15504 - avaliação de processos de software);

CONSIDERANDO a efetividade de adoção de metodologias ágeis de desenvolvimento e seu alinhamento ao modelo MPS.BR;

CONSIDERANDO a necessidade de desenvolver um processo padronizado de desenvolvimento de sistemas, em conformidade com as recomendações do TCU – Acórdãos 1603/2008 (item 9.1.14), 1233/2012 (itens 9.15.6, 9.15.7, 9.15.8, 9.15.9, 9.15.18, 9.15.18.5 e 9.15.18.6) e 2314/2013;

CONSIDERANDO recomendação de auditoria interna no Processo nº 23129/2013 - Governança, Riscos e Controles de TI, item 5.6,

### **RESOLVE:**

Art. 1º Instituir, baseado em práticas de metodologias ágeis e do modelo MPS.BR, o Processo de Desenvolvimento e Manutenção de Software no âmbito do Tribunal Regional Eleitoral da Paraíba.

Art. 2º Para os efeitos desta portaria, consideram-se as seguintes definições:

I. Ciclo de Desenvolvimento: unidade de planejamento com tempo predeterminado e escopo definido. O ciclo deve se concentrar em um único produto, ainda que o time de desenvolvimento tenha outros em seu portfólio.

II. Demanda: qualquer relato relacionado que requeira a criação ou manutenção de aplicações de software.

III. Gestor de sistema: servidor que exercerá as atribuições definidas pela Portaria nº 1115/2016 TRE-PB/PTRE/ASPRE.

IV. Implementação: codificação da solução, proposta em linguagem de programação.

V. Sistema de Controle de Versões: repositório que armazena todas as versões dos arquivos dos produtos. No desenvolvimento de software, sua importância reside na possibilidade de manter o histórico da evolução dos códigos-fonte, de modo que mais pessoas possam trabalhar de forma cooperativa e organizada.

VI. Sistema de Gerenciamento de Demandas: solução para registro e

acompanhamento das demandas destinadas às áreas de desenvolvimento, manutenção e implantação de sistemas de informação.

VII. Time de desenvolvimento: grupo de colaboradores com habilidades e conhecimentos para criação e manutenção de aplicações de software.

Art. 3º O processo de desenvolvimento e manutenção de software se inicia com a autorização de análise do problema, visando à elaboração de proposta de solução.

§ 1º O início do processo para demandas de manutenção de pequeno porte pode ser autorizado pela Coordenadoria de Sistemas (COSIS).

§ 2º O início do processo para demandas de novos sistemas ou manutenções de grande porte deve ser autorizado pelo Comitê Gestor (COGES).

Art. 4º Após a autorização, o gestor do sistema promoverá a reunião de partida entre o time de desenvolvimento e as partes interessadas, momento em que será esclarecido o problema de negócio, definidos papéis, explicado o processo de desenvolvimento e distribuídas responsabilidades.

Art. 5º No prazo de até 30 (trinta) dias a contar da data da reunião de partida, o gestor do sistema promoverá nova reunião para aprovação da proposta de solução escolhida e do planejamento inicial da execução.

Parágrafo único. Devem fazer parte do planejamento inicial:

I. O ciclo de vida para entrega da solução, refletindo necessidades identificadas quanto a atividades técnicas e não técnicas, tais como mapeamento de processos, desenvolvimento de software, implantação, testes, treinamento e normatização.

II. O cronograma de marcos, que priorizará a velocidade e a frequência de entregas, em detrimento de soluções completas, de longo prazo.

III. O escopo do primeiro ciclo de desenvolvimento.

Art. 6º Após reunião para aprovação da proposta e planejamento inicial, deve ser iniciado o primeiro ciclo de desenvolvimento.

§ 1º O escopo do ciclo deve ser aprovado pelo gestor do sistema e pelo time de desenvolvimento.

§ 2º Os requisitos que compõem o escopo aprovado devem ser registrados em Sistema de Gerenciamento de Demandas.

§ 3º Toda implementação feita pelo time de desenvolvimento deve ser apropriadamente gerenciada em Sistema de Controle de Versões.

§ 4º Deve haver rastreabilidade entre requisitos registrados e código-fonte.

Art. 7º Finda a fase de implementação no ciclo, deve ser realizada reunião de revisão a fim de que o time de desenvolvimento apresente ao gestor do sistema os resultados alcançados durante o ciclo.

Parágrafo único. O gestor deve promover a avaliação dos resultados e definir se a versão está apta a ser instalada em ambiente de produção.

Art. 8º Após conclusão do primeiro, novos ciclos de desenvolvimento devem ser promovidos enquanto os objetivos definidos não tiverem sido atingidos.

§ 1º O planejamento inicial será continuamente refinado e comunicado durante os ciclos de desenvolvimento.

§ 2º Mudanças de requisitos ocorridas ao longo dos ciclos devem ser apropriadamente registradas em Sistema de Gerenciamento de Demandas, após aprovação do gestor do sistema.

Art. 9º Após entrega da solução, deve ser realizada avaliação de lições

aprendidas para melhoria do processo de desenvolvimento e manutenção de software.

Parágrafo único. A incorporação das melhorias identificadas à arquitetura e aos processos de trabalho deve ser realizada, sempre que possível, antes do início da próxima iniciativa de desenvolvimento.

Art. 10 Compete ao gestor do sistema a condução de atividades não técnicas necessárias à solução, tais como treinamentos e normatizações.

Art. 11 Compete à Seção de Análise e Desenvolvimento de Sistemas a elaboração de guia específico para detalhamento das práticas descritas nesta norma.

**ANDRÉ SOARES CAVALCANTI**  
**Diretor Geral do TRE-PB**

João Pessoa, 27 de junho de 2017.



Documento assinado eletronicamente por **ANDRÉ SOARES CAVALCANTI, Diretor Geral**, em 30/06/2017, às 21:01, conforme art. 1º, III, "b", da Lei 11.419/2006.



A autenticidade do documento pode ser conferida no site [https://sei.tre-pb.jus.br/sei/controlador\\_externo.php?acao=documento\\_conferir&id\\_orgao\\_acesso\\_externo=0](https://sei.tre-pb.jus.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0) informando o código verificador **0203840** e o código CRC **4B97C595**.