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.

Orientador:	Fausto Véras Maranhão Ayres
Supervisor:	Francisco José Rodrigues Gomes
Coordenador do Curso:	Cândido José Ramos do Egypto
Empresa:	Tribunal Regional Eleitoral da Paraíba
Período:	10/04/2017 à 10/04/2019

João Pessoa - PB

APROVAÇÃO

Francisco José Rodrigues Gomes
Supervisor da Empresa

Cândido José Ramos do Egypto
Coordenador do CST de Sistemas para Internet

Prof. Dr. Fausto Véras 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

Agradeço a Deus e ao meu Anjo da Guarda por ter me acompanhado em todos os momentos da minha vida. À família por todo amparo e formação social, neste caso representada pelos meus Pais, meu Irmão, minha Esposa, Enteada e Filho. A todos os Docentes que dedicam a vida na formação de indivíduos conscientes e responsáveis, em especial aos Docentes do Instituto Federal da Paraíba, pois me surpreenderam desde o primeiro instante com a qualidade e dedicação em que realizam seu trabalho.

Antes mesmo de sair de Mato Grosso e chegar à Paraíba percebi que seria bem vindo nessa terra. Ainda analisando as possibilidades de vir, entrei em contato com a coordenação do curso de TSI solicitando informações sobre a inscrição do processo seletivo especial, que precisava ser feito pessoalmente. Então a inestimável, extraordinária e dedicada Prof.ª Valéria, coordenadora ná época, pediu que eu enviasse os documentos junto com uma procuração que ela realizaria a inscrição para mim. Desde esse momento vi as portas se abrindo para mim e minha família em uma nova vida que estava por vir. Fica aqui o agradecimento especial a Prof.ª Valéria e ao povo paraibano que nos acolheram.

Gostaria de dedicar um agradecimento especial a alguns professores que se destacaram pelo amor a profissão e a arte de ensinar. Dentre eles enfatizo, em ordem cronológica do curso, a Prof.ª Valéria, que sempre demonstrou muito empenho ao apresentar a disciplina de algorítimos para os novatos, ao Prof.º Fausto que sempre foi um grande amigo, rendendo boas discussões sobre tecnologias e mercado de trabalho, o Prof.º Jaildo pela sua ética impecável, o Prof.º Fred que faz jus ao apelido adquirido ao longo de sua profissão que é "O Mito"lembrado e enfatizado por gerações de turmas do curso de TSI, o Prof.º Dênio pelo domínio impecável de sua disciplina, a Prof.ª Damires por toda sua capacidade, classe e ética impecáveis ao conduzir suas aulas, pelas calorosas discussões sobre gestão e analise nas aulas da Prof.ª Heremita e Prof.ª Nadja e também o Prof.º Paulo que apresentou sua disciplina de maneira respeitosa e com muito amor.

Devemos mudar nossa atitude tradicional em relação à construção de programas. Em vez de imaginar que nossa principal tarefa é instruir o computador sobre o que ele deve fazer, vamos imaginar que nossa principal tarefa é explicar a seres humanos o que queremos que o computador faça. (Donald E. Knuth, Literate Programming).

RESUMO

PACOLA, José Ricardo Bettini. Veículos: Sistema de Gestão de Frotas do TRE-PB. 04/2019. 45 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 projeto Veículos: Sistema de Gestão de Frotas do TRE-PB. O projeto tinha previsão para ser concluído em até dois meses, porém uma mudança substancial do modelo teve que ser efetuada durante a fase de desenvolvimento da versão 1.0 e então o projeto se estendeu para cinco meses.Ele foi supervisionado pelo chefe da SEDES, Francisco Gomes, bem como acompanhado pelo supervisor técnico, que é o desenvolvedor responsável pelo projeto que também precisou ser substituído, todos efetivos no quadro do Tribunal. A equipe no projeto ficou 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. Equipe.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

PDS	Plano de Desenvolvimento de Software
MODUS	Modelo de Desenvolvimento de Software
SEDES	Seção de Análise e Desenvolvimento de Sistemas
SISBAN	Seção de Implantação de Sistemas e Bancos 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
MVC	Model View Control
POJO	Plain Old Java Objects
IDE	Integrated development environment
COSIS	Coordenadoria de Sistemas
COSEG	Coordenadoria de Serviços Gerais
SETRAN	Seção de Transportes
JEE	Java Enterprise Edition
LDAP	Lightweight Directory Access Protocol
PDF	Portable Document Format
DAO	Data Access Object

SUMÁRIO

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

[1] 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

[1] 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 - Atividades Realizadas: apresenta o projeto, no qual as atividades do estagiário foram realizadas, descrevendo os principais conceitos e funcionalidades. Relata as atividades realizadas ao longo do período de estágio descrevendo o fluxo e o processo de desenvolvimento e detalha a implementação de algumas funcionalidades.

Capítulo 4 – Considerações Finais: Apresenta um relato sobre as experiências adquiridas, metas e objetivos alcançados. O quão grande foi a contribuição do estágio na minha formação profissional.

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.

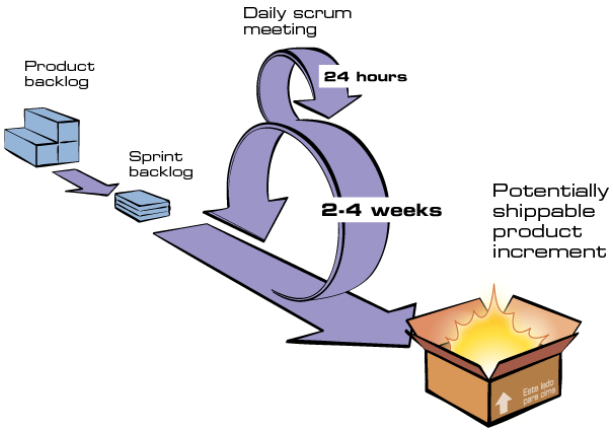
> O SCRUM assume-se como uma metodologia extremamente ágil e flexível, que tem por objetivo definir um processo de desenvolvimento interativo e incremental podendo ser aplicado a qualquer produto ou no gerenciamento de qualquer atividade complexa. (BISSI, 2007).

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 cliclo 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[1] 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 coorporativos. 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.

[1] http://www.oracle.com/technetwork/pt/java/index.html

2.3 JSF

De acordo com Cordeiro (2014), JavaServer Faces[2], 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[3] 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 e Hibernate

Java Persistense API[4] ou JPA é uma especificação criada em 2006 a partir do Hibernate[5], 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

[2] https://javaee.github.io/javaserverfaces-spec/
[3] https://primefaces.org/
[4] http://www.oracle.com/technetwork/java/javaee/tech/persistence-jsp-140049.html
[5] 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).

2.6 Spring Framework

O Spring surgiu em 2003 como uma resposta à complexidade das primeiras especificações do J2EE. Enquanto alguns consideram que Java EE e Spring estão competindo, o Spring é, de fato, complementar ao Java EE. O modelo de programação Spring não abrange a especificação da plataforma Java EE; em vez disso, integra-se com especificações individuais cuidadosamente selecionadas do JavaEE:

- Servlet API (JSR 340)
- WebSocket API (JSR 356)
- Concurrency Utilities (JSR 236)
- JSON Binding API (JSR 367)
- Bean Validation (JSR 303)
- JPA (JSR 338)
- JMS (JSR 914)

O framework também suporta injeção de dependências (JSR 330) e anotações comuns (JSR 250), que os desenvolvedores de aplicativos podem usar em vez dos mecanismos específicos do Spring.

2.6.1 Anotações de Estereótipo

São anotações usadas para declarar a função que o componente desempenha na aplicação. Por exemplo, a anotação @Repository no Spring Framework é uma marcação para qualquer classe que atenda a função de um repositório (também conhecido como Data Access Object ou DAO).

2.7 JasperReports

O JasperReports é um framework open source inteiramente escrito em Java. Ele é um dos mecanismos mais populares para a geração de relatórios na plataforma Java. O JasperReports nos fornece funcionalidades que permitem criar relatórios complexos de forma estruturada, a partir da elaboração de um template (LIMA, 2012).

O template é um arquivo XML com a extensão .jrxml. É neste arquivo que é especificada a estrutura do relatório, ou seja, é nele onde informamos os dados que irão compor o relatório, em que posição e de que forma serão exibidos, formando assim um layout. Utiliza-se o IReports para diagramação dos elementos de maneira gráfica.

A partir da definição do template e com o auxílio do framework JasperReports é possível gerar relatórios e exportá-los para diversos formatos, como: HTML, PDF e DOC.

2.8 Tomcat

O Apache Tomcat® é uma implementação open source das tecnologias Java Servlet, JavaServer Pages, Java Expression Language e Java WebSocket technologies.

Ele atende parte da especificação JEE com as tecnologias Servlet e JSP e tem a capacidade de atuar também como servidor web HTTP escrito puramente em Java. Ele inclui ferramentas para configuração e gerenciamento, o que também pode ser feito editando-se manualmente arquivos de configuração formatados em XML (WIKIPEDIA, 2018).

Atualmente o Tomcat está na versão 9.x. Para o projeto foi utilizado o Tomcat 8.0.x tanto nas máquinas de desenvolvimento quanto nos servidores. O TRE-PB faz o uso de múltiplos servidores de produção e homologação com o Tomcat, dividindo assim a carga das aplicações.

3 Atividades realizadas

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 até a implementação da aplicação em produção.

3.1 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.

O sistema permite a qualquer membro e servidor lotado no TRE-PB, através do seu login de usuário, solicitar um veículo descrevendo a data, local/trecho, hora de partida, hora de retorno, a atividade que deverá ser realizada e quais passageiros irão para o mesmo destino, como pode ser visto na Figura 3.

O gestor lotado na SETRAN, possui um perfil de administrador e em sua tela recebe as solicitações de todos os servidores e membros. Com a visão de todas as solicitações o gestor tem a possibilidade de analisar passageiros que irão para os mesmos destinos, ou até mesmo na rota, com horários similares e montar as viagens adequando a disponibilidade de motoristas e veículos. Através de um quadro, exibido na figura Figura 4, o gestor consegue visualizar a ocupação dos veículos e motoristas pelo horário e dia da semana. Com a viagem montada e deferida pelo gestor, o solicitante recebe uma confirmação em seu e-mail informando os detalhes da viagem. Os motoristas recebem uma guia contendo a autorização e detalhes da viagem como a rota e os passageiros. Nessa guia o motorista deve anotar a quilometragem do veículo e exibir para o vigia que deve conferir os valores e permitir a saída do veículo. No retorno o motorista novamente anota na guia a quilometragem de chegada que é conferida pelo vigia.

A modelagem inicial do sistema, na versão 1.0.0, não possuia uma separação dos modelos entre solicitação e viagem. Isso acabou ocasionando um gargalo no desenvolvimento, pois surgiram feedbacks ao longo das etapas, como pode ser visto na ata de reunião na Figura 5, em que há casos de uma solicitação necessitar de mais de uma viagem para ser concluída e uma viagem poder atender mais de uma solicitação. A aplicação precisou ser refatorada e um novo prazo foi apresentado.

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

Figura 2 – Catálogo técnico de sistemas - Veículos

Fonte: COSIS

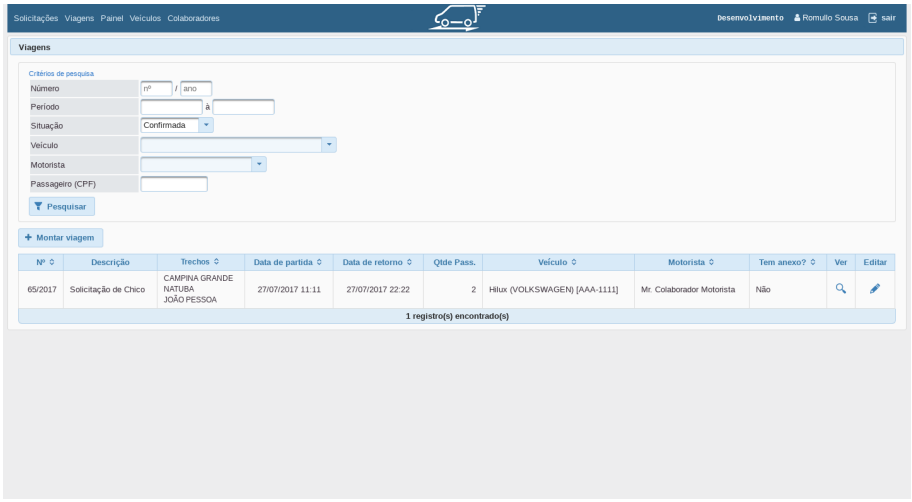
construído a partir de definições estabelecidas em portarias, conforme exemplo, no Anexo B da última publicada pela Diretoria Geral do TRE-PB no que se diz respeito aos padrões de governança em Tecnologia da Informação. O processo de desenvolvimento e manutenção de software se inicia com a autorização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

3.2 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.

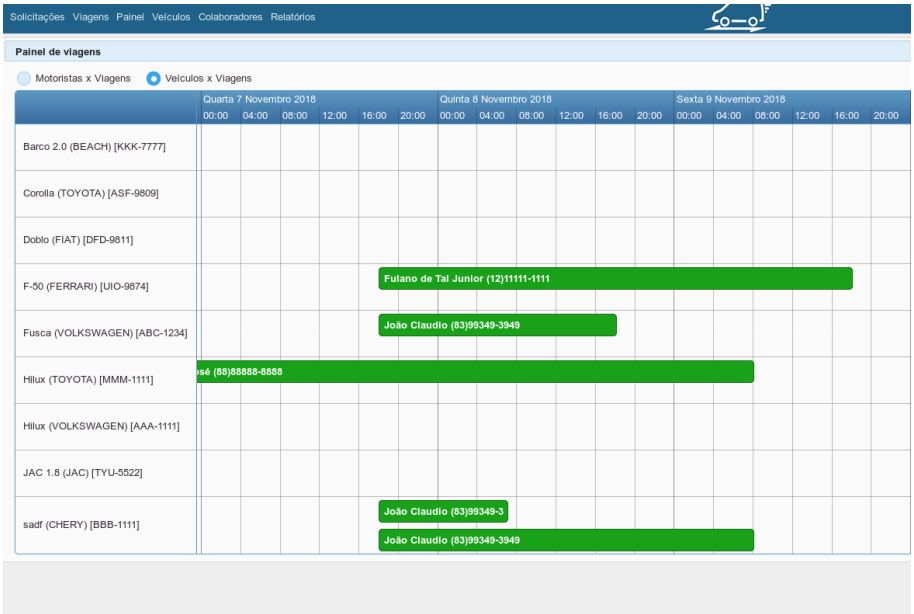
Para o time já se inicia o processo de imersão, onde após ouvir as necessidades descritas

Figura 3 – Veículos - Tela de solicitação



Fonte: SEDES

Figura 4 – Veículos - Painel de veículos e viagens

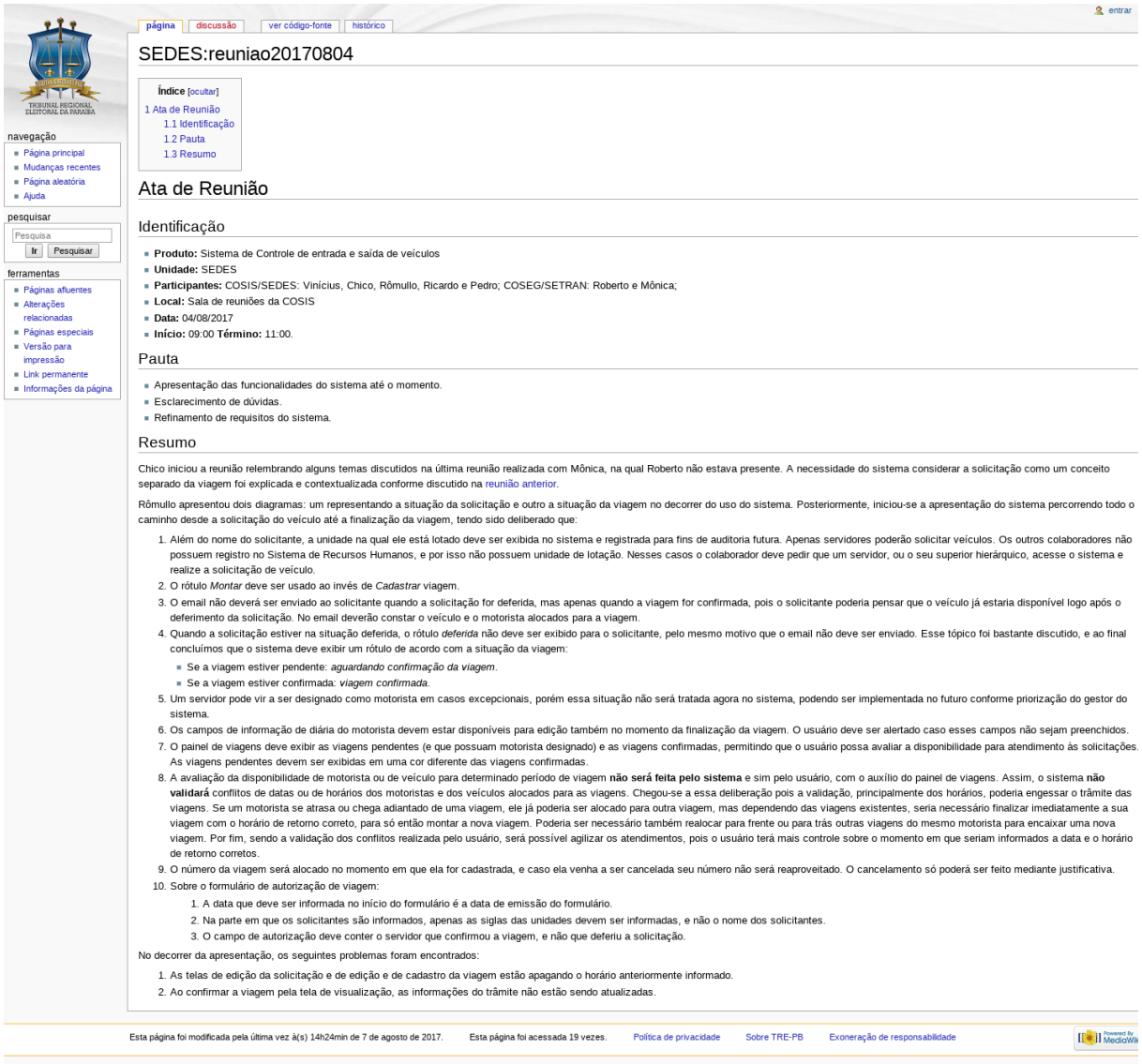


Fonte: SEDES

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 6. 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.

Figura 5 – Ata de reunião - feedback



página | discussão | ver código-fonte | histórico — entrar

TRIBUNAL REGIONAL ELEITORAL DA PARAÍBA

navegação
- Página principal
- Mudanças recentes
- Página aleatória
- Ajuda

pesquisar
Pesquisa
Ir | Pesquisar

ferramentas
- Páginas afluentes
- Alterações relacionadas
- Páginas especiais
- Versão para impressão
- Link permanente
- Informações da página

SEDES:reuniao20170804

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/SEDES: Vinícius, Chico, Rômullo, Ricardo e Pedro; COSEG/SETRAN: Roberto e Mônica;
- **Local:** Sala de reuniões da COSIS
- **Data:** 04/08/2017
- **Início:** 09:00 **Término:** 11:00.

Pauta

- Apresentação das funcionalidades do sistema até o momento.
- Esclarecimento de dúvidas.
- Refinamento de requisitos do sistema.

Resumo

Chico iniciou a reunião relembrando alguns temas discutidos na última reunião realizada com Mônica, na qual Roberto não estava presente. A necessidade do sistema considerar a solicitação como um conceito separado da viagem foi explicada e contextualizada conforme discutido na reunião anterior.

Rômullo apresentou dois diagramas: um representando a situação da solicitação e outro a situação da viagem no decorrer do uso do sistema. Posteriormente, iniciou-se a apresentação do sistema percorrendo todo o caminho desde a solicitação do veículo até a finalização da viagem, tendo sido deliberado que:

1. Além do nome do solicitante, a unidade na qual ele está lotado deve ser exibida no sistema e registrada para fins de auditoria futura. Apenas servidores poderão solicitar veículos. Os outros colaboradores não possuem registro no Sistema de Recursos Humanos, e por isso não possuem unidade de lotação. Nesses casos o colaborador deve pedir que um servidor, ou o seu superior hierárquico, acesse o sistema e realize a solicitação de veículo.
2. O rótulo *Montar* deve ser usado ao invés de *Cadastrar* viagem.
3. O email não deverá ser enviado ao solicitante quando a solicitação for deferida, mas apenas quando a viagem for confirmada, pois o solicitante poderia pensar que o veículo já estaria disponível logo após o deferimento da solicitação. No email deverão constar o veículo e o motorista alocados para a viagem.
4. Quando a solicitação estiver na situação deferida, o rótulo *deferida* não deve ser exibido para o solicitante, pelo mesmo motivo que o email não deve ser enviado. Esse tópico foi bastante discutido, e ao final concluímos que o sistema deve exibir um rótulo de acordo com a situação da viagem:
   - Se a viagem estiver pendente: *aguardando confirmação da viagem*.
   - Se a viagem estiver confirmada: *viagem confirmada*.
5. Um servidor pode vir a ser designado como motorista em casos excepcionais, porém essa situação não será tratada agora no sistema, podendo ser implementada no futuro conforme priorização do gestor do sistema.
6. Os campos de informação de diária do motorista devem estar disponíveis para edição também no momento da finalização da viagem. O usuário deve ser alertado caso esses campos não sejam preenchidos.
7. O painel de viagens deve exibir as viagens pendentes (e que possuam motorista designado) e as viagens confirmadas, permitindo que o usuário possa avaliar a disponibilidade para atendimento às solicitações. As viagens pendentes devem ser exibidas em uma cor diferente das viagens confirmadas.
8. A avaliação da disponibilidade de motorista ou de veículo para determinado período de viagem **não será feita pelo sistema** e sim pelo usuário, com o auxílio do painel de viagens. Assim, o sistema **não validará** conflitos de datas ou de horários dos motoristas e dos veículos alocados para as viagens. Chegou-se a essa deliberação pois a validação, principalmente dos horários, poderia engessar o trâmite das viagens. Se um motorista se atrasa ou chega adiantado de uma viagem, ele já poderia ser alocado para outra viagem, mas dependendo das viagens existentes, seria necessário finalizar imediatamente a sua viagem com o horário de retorno correto, para só então montar a nova viagem. Poderia ser necessário também realocar para frente ou para trás outras viagens do mesmo motorista para encaixar uma nova viagem. Por fim, sendo a validação dos conflitos realizada pelo usuário, será possível agilizar os atendimentos, pois o usuário terá mais controle sobre o momento em que seriam informados a data e o horário de retorno corretos.
9. O número da viagem será alocado no momento em que ela for cadastrada, e caso ela venha a ser cancelada seu número não será reaproveitado. O cancelamento só poderá ser feito mediante justificativa.
10. Sobre o formulário de autorização de viagem:
    1. A data que deve ser informada no início do formulário é a data de emissão do formulário.
    2. Na parte em que os solicitantes são informados, apenas as siglas das unidades devem ser informadas, e não o nome dos solicitantes.
    3. O campo de autorização deve conter o servidor que confirmou a viagem, e não que deferiu a solicitação.

No decorrer da apresentação, os seguintes problemas foram encontrados:

1. As telas de edição da solicitação e de edição e de cadastro da viagem estão apagando o horário anteriormente informado.
2. Ao confirmar a viagem pela tela de visualização, as informações do trâmite não estão sendo atualizadas.

Esta página foi modificada pela última vez à(s) 14h24min de 7 de agosto de 2017. Esta página foi acessada 19 vezes. Política de privacidade Sobre TRE-PB Exoneração de responsabilidade Powered By MediaWiki

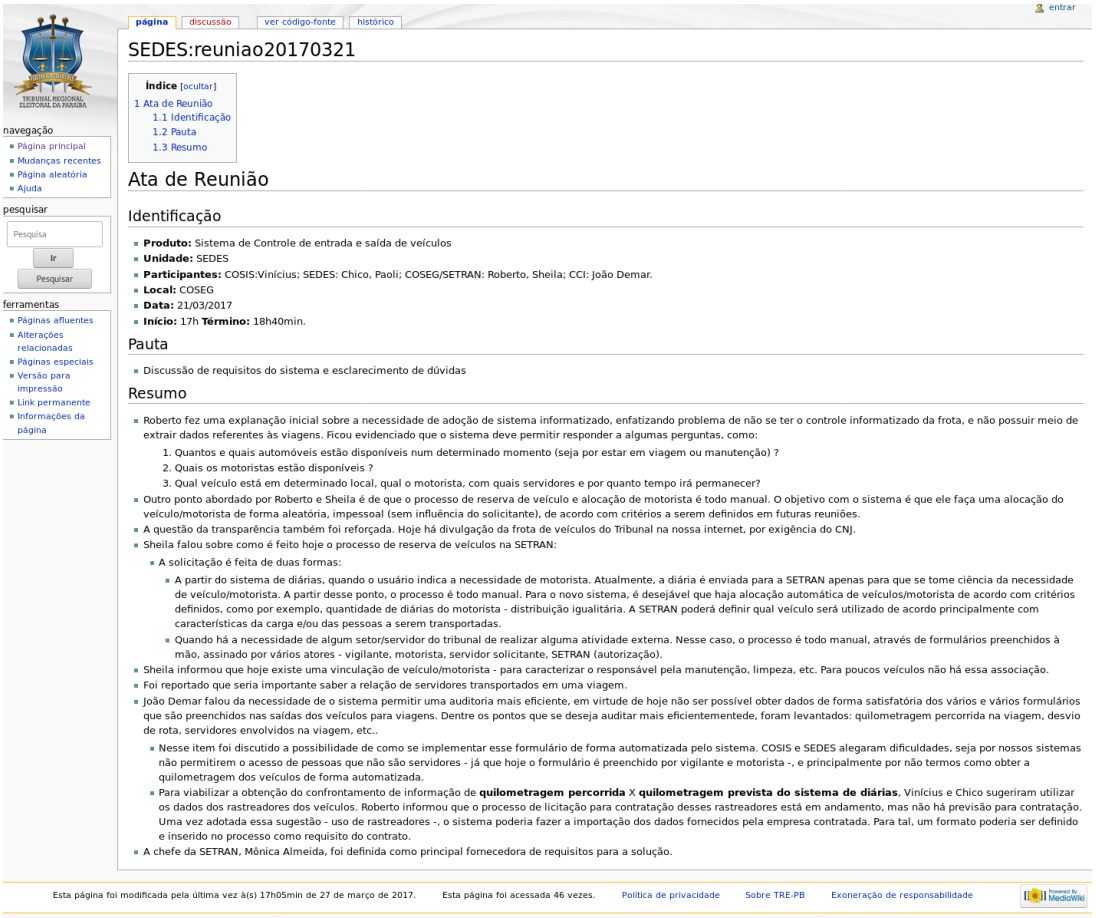
Fonte: SEDES

3.3 Planejamento

Todos os membros do time se reúnem para medir o tamanho de cada atividade. Uma a uma, as atividades são analisadas e discutidas por cada membro do time onde cada um expõe sua opinião e justifica o tamanho da atividade mensurada 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 e 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)

Figura 6 – Ata de Reunião - 21/03/2017



Fonte: SEDES

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 7.

3.4 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 e disponibilizado no repositório para todos os desenvolvedores. A SISBAN. que 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 8 a seguir é possível ver uma tarefa especifica 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"[1].

3.4.1 Criando as tabelas no banco

Com o domínio do problema já definido são então criadas as tabelas do banco de dados com seus atributos, relacionamentos e restrições. Scripts em SQL são gerados, conforme exemplo da Figura 9, para cada tabela, levando em consideração as particularidades da sintaxe SQL para o banco de dados Oracle.

3.4.2 Mapeando o modelo com o banco

Com as tabelas já criadas e relacionadas, começa-se a codificação propriamente dita na linguagem Java. As classes do modelo são codificadas com seus respectivos atributos e métodos e mapeadas através da interface JPA. Na Figura 10 é possível ver a entidade que define um veículo na aplicação já com as anotações que a mapeiam com o banco de dados.

3.4.3 MVC

As aplicações em Web desenvolvidas no TRE-PB que utilizam o JEE obedecem o padrão de projeto denominado MVC. O código tem uma divisão lógica. Na camada de modelo(Model) temos as entidades que definem o domínio da aplicação como pode ser visto na Subseção 3.4.2. Na camada de visão(View) da aplicação temos as páginas JSF que irão conter os componentes visuais da aplicação, como é o caso de formulários, botões, texto, entre outros, e foram desenvolvidos utilizando o framework JSF.

O JSF faz uso de beans[2] Java para possibilitar a separação do código de apresentação(View) do código de negócio(Control). Por sua vez, uma página JSF referencia um ou mais beans, que são classes Java que armazenam os código necessários para apresentação dos dados nas páginas da aplicação.

Estes beans têm seu ciclo de vida gerenciado pelo JSF, sendo assim chamados de Managed Beans. Os managed beans do JSF fazem o papel de controladores da nossa aplicação.

[1] 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

[2] São classes escritas de acordo com uma convenção em particular. São usados para encapsular muitos objetos em um único objeto (o bean). https://pt.wikipedia.org/wiki/JavaBeans

Eles recebem um estímulo da camada de visão para a execução de alguma operação e em seguida delegam esta execução à classe de negócio responsável, que são denominadas classes Manager. Após a execução da regra de negócio o controlador repassa o resultado da operação à camada de visão.

As classes Manager implementam as regras de negócio da aplicação. Essas camadas de serviço visam encapsular, isolar o código de negócio da aplicação, podendo serem reutilizadas em outras visões(beans) da aplicação. Essas classes tem acesso aos objetos da camada de persistência DAO através do Framework Spring que utiliza o conceito de injeção de dependências. Veja o exemplo na Figura 11. O Spring possui uma arquitetura baseada em interfaces e POJOs, oferecendo aos POJOs, no caso representados pelas classes do modelo, características como mecanismos de segurança e controle de transações (WIKIPEDIA, 2017).

Por último temos a camada de persistência DAO. Esses objetos são responsáveis pelas operações realizadas no banco de dados. A utilização do mesmo é um bom padrão de desenvolvimento, pois separa as regras de negócio da aplicação das operações de manipulação do banco de dados. Os objetos DAO fazem uso de um objeto do tipo EntityManager, que é criado pelo Hibernate e também utilizam o Spring para realizar a injeção de dependências no DAO, veja o exemplo na Figura 12.

3.4.4 Guia de autorização

Após uma viagem ser deferida, o operador do sistema na SETRAN emite a guia de autorização para entregar ao motorista. Essa guia é um arquivo em formato PDF e é fornecida pela aplicação na forma de um link para download contendo os dados da viagem. Para criação desse arquivo foram utilizadas as bibliotecas do framework JasperReports.

Um arquivo de template é criado, compilado e invocado. Esse template formata o posicionamento dos textos no papel conforme Figura 13, nesse caso foi utilizado o papel no formato A5. No trecho exibido pela Figura 14 é possível ver a invocação do template pelo código java passando dados da viagem como parâmetros.

3.5 Homologação

Ao final de cada Sprint é apresentada para o cliente uma prévia do que está sendo desenvolvido. A aplicação é então colocada em um servidor específico para sistemas em homologação rodando um Tomcat. Um schema no banco de dados também é criado e o código é compilado com as particularidades do ambiente de homologação que contém caminhos e usuários diferentes na rede para o banco de dados e servidores LDAP de autenticação.

O desenvolvedor responsável pela homologação, em seu próprio computador muda, na IDE, o perfil de desenvolvimento para homologação e realiza o comando de compilação do código. Como resultado ele tem um arquivo no formato WAR e está pronto para realizar o "deploy"enviando o arquivo para servidor.

Em alguns casos a funcionalidade apresentada já pode ser utilizada para realizar cadastros pelo cliente e caso não aconteçam mudanças significativas no modelo esses dados podem ser importados quando a aplicação entrar em produção.

3.6 Documentação

A documentação ocorre após a apresentação das releases e aplicação dos ajustes, caso ocorram. O objetivo dessa tarefa é fazer a inclusão ou atualização do produto no catálogo de produtos e a elaboração ou atualização do manual do usuário, que contempla as histórias implementadas e descreve passo a passo como realizar as tarefas. O manual é apresentado no formato PDF e é compilado junto com a aplicação que contém um link de ajuda apontando para o manual.

3.7 Produção

Após a entrega da última release prevista no escopo do projeto e a realização dos testes pelo cliente no servidor de homologação, já com as correções e ajustes sugeridas, ocorre a implantação da aplicação no servidor de produção. Um processo similar ao de implantação em homologação, porém com algumas particularidades.

Com o schema no banco de produção já criado e todas as tabelas e dados já implementados, o desenvolvedor responsável compila em sua IDE com o perfil de produção e realiza o deploy do aquivo WAR no servidor identificado pelo número da sua versão.

> ... processo de levar código do desenvolvimento e teste para produção. É comum chamar esse processo de deploy em ambiente de produção. Em português é até possível ouvir o neologismo deploiar, ou então o termo correto em português, que seria"implantar". (SATO, 2014, p. 19)

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[3] é então criada no sistema de controle de versões SVN e contém o número de versão e sua descrição e serve para permitir recuperar o código-fonte, caso seja necessário, conforme a versão foi construída. O cliente é então informado formalmente que a versão está disponível para uso em produção.

3.8 Encerramento

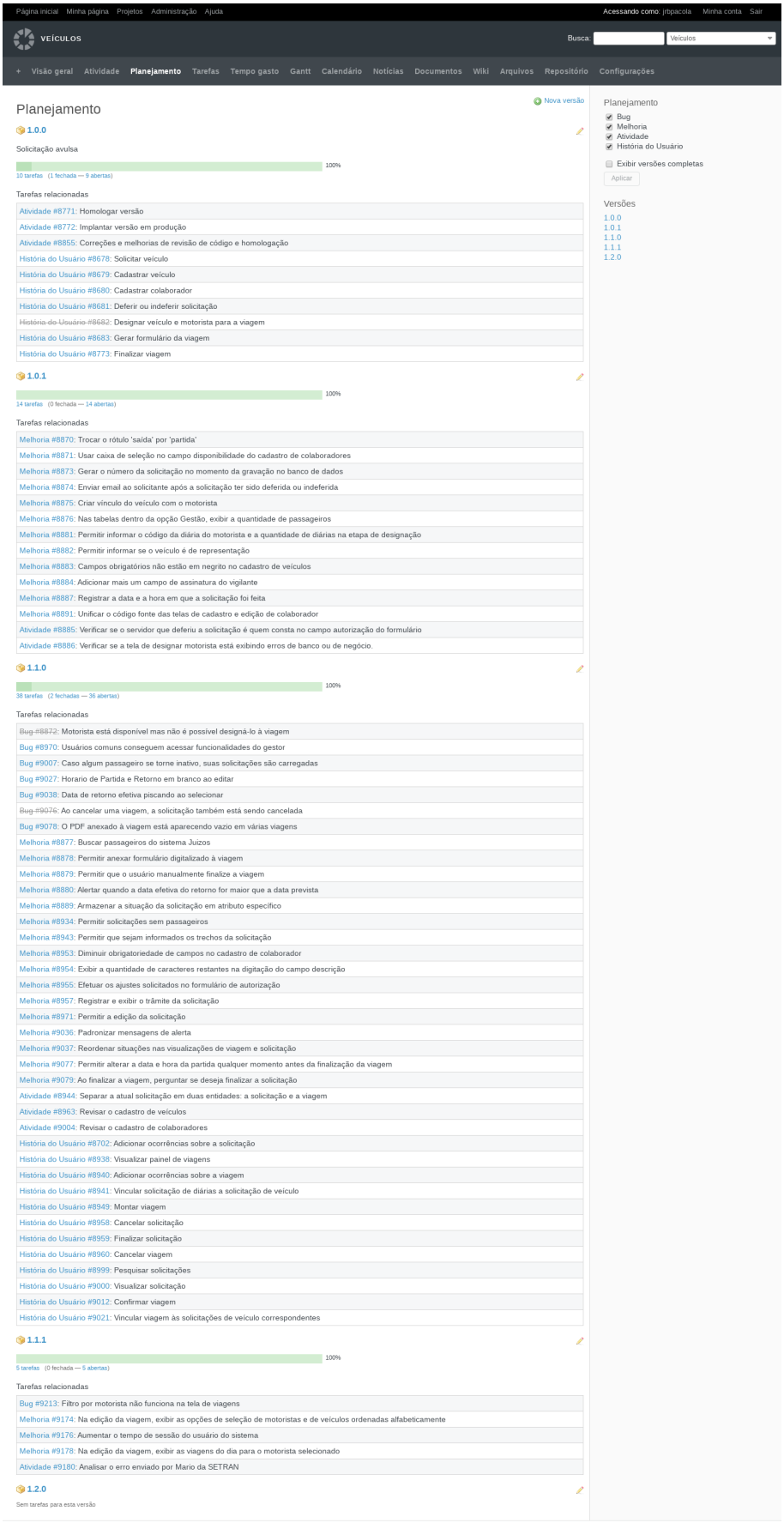
A execução das atividades de finalização do projeto se iniciam com a formalização, pelo responsável de suporte de negócio, sobre discussões de melhorias do projeto, do modelo/arquitetura utilizada, dos procedimentos, das técnicas e do método de desenvolvimento.

[3] É apenas um "snapshot" do projeto no tempo. http://svnbook.red-bean.com/en/1.7/svn.branchmerge.tags.html

O gerente do projeto realiza uma reunião final de entrega do produto com o time 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. Após a reunião comunica-se o encerramento do projeto aos interessados através de e-mail ou despacho/documento em processo administrativo.

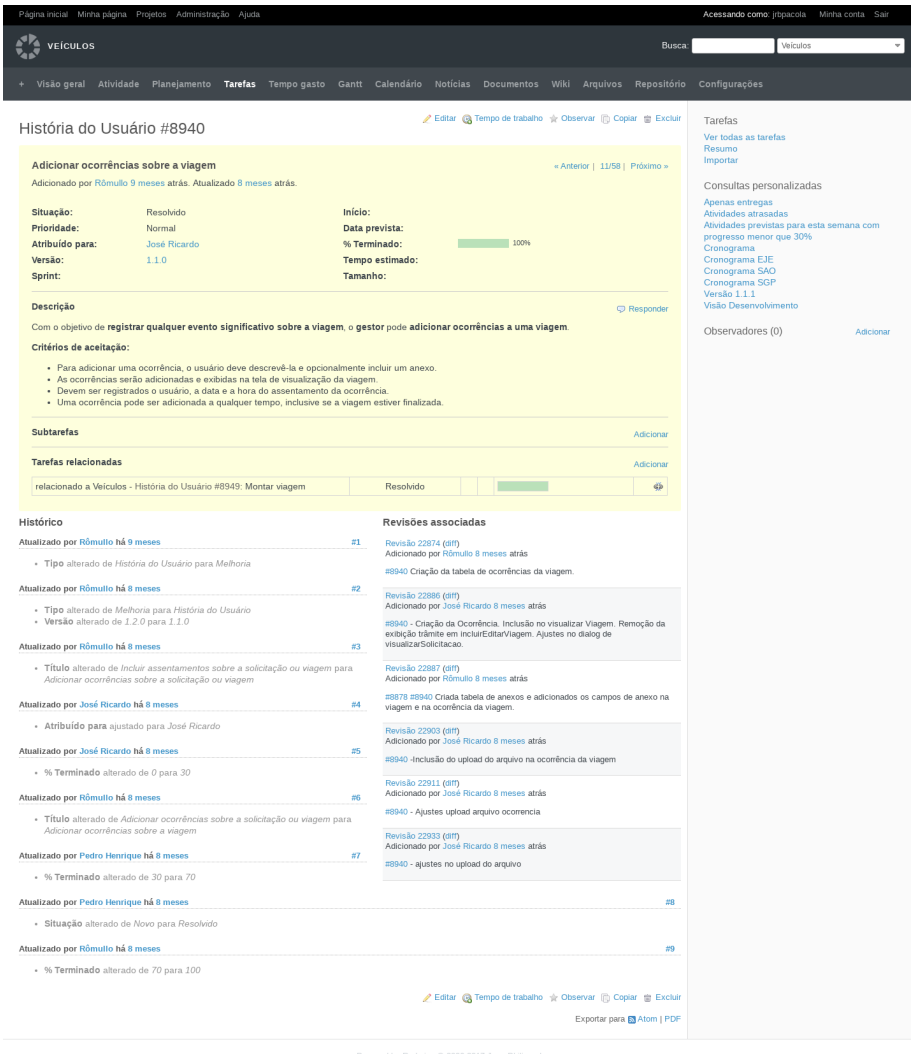
As lições aprendidas deverão gerar oportunidades de melhorias que deverão ser incluídas no backlog de demandas da SEDES para implementação em momento oportuno e adoção em futuros projetos.

Figura 7 – Planejamento das versões



Fonte: SEDES

Figura 8 – Atividade 8940



Fonte: SEDES

Figura 9 – Script SQL

```
CREATE TABLE "VEICULOS"."VEICULO"
 (    "ID" NUMBER(19,0) NOT NULL ENABLE,
  "CODIGO_RENAVAM" VARCHAR2(11 CHAR) NOT NULL ENABLE,
  "LOCAL" CHAR(1 CHAR) NOT NULL ENABLE,
  "MODELO" VARCHAR2(20 CHAR) NOT NULL ENABLE,
  "PLACA" VARCHAR2(7 CHAR) NOT NULL ENABLE,
  "SITUACAO" CHAR(1 CHAR) NOT NULL ENABLE,
  "CHASSI" VARCHAR2(17 CHAR) NOT NULL ENABLE,
  "TIPO_VEICULO" CHAR(1 CHAR) NOT NULL ENABLE,
  "MARCA_ID" NUMBER(19,0) NOT NULL ENABLE,
  "REPRESENTACAO" NUMBER(1,0) NOT NULL ENABLE,
  "MOTORISTA_RESPONSAVEL_ID" NUMBER(19,0),
  "ANO_FABRICACAO" NUMBER(4,0) NOT NULL ENABLE,
  "ANO_MODELO" NUMBER(4,0) NOT NULL ENABLE,
  "NUMERO_PASSAGEIROS" NUMBER(2,0) NOT NULL ENABLE,
  "ATIVO" NUMBER(1,0) NOT NULL ENABLE,
   CONSTRAINT "PK_VEICULO" PRIMARY KEY ("ID"),
   CONSTRAINT "UK_VEICULO_PLACA" UNIQUE ("PLACA"),
   CONSTRAINT "FK_VEICULO_MARCA" FOREIGN KEY ("MARCA_ID")
    REFERENCES "VEICULOS"."MARCA" ("ID") ENABLE
 )
```

Fonte: SEDES

Figura 10 – Entidade Java - Veículo

```
@Entity
@Table(name = "VEICULO")
@SequenceGenerator(name = "sequence", sequenceName = "SQ_VEICULO", allocationSize = 1)
public class Veiculo extends ObjetoIdentifiable {

    @ManyToOne
    @JoinColumn(name = "MOTORISTA_RESPONSAVEL_ID")
    private Motorista motoristaResponsavel;

    @Column(name = "CODIGO_RENAVAM")
    private String codigoRenavam;

    @Column(name = "PLACA")
    private String placa;

    @Column(name = "CHASSI")
    private String chassi;

    @ManyToOne
    @JoinColumn(name = "MARCA_ID")
    private Marca marca;

    @Column(name = "MODELO")
    private String modelo;

    @Column(name = "LOCAL")
    private String local;

    @Column(name = "ANO_MODELO")
    private Integer anoModelo;

    @Column(name = "ANO_FABRICACAO")
    private Integer anoFabricacao;

    @Column(name = "NUMERO_PASSAGEIROS")
    private Integer numeroPassageiros;
```

Fonte: SEDES

Figura 11 – Manager Java - Veículo

```java
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

@Service
@Transactional
public class VeiculoManager extends IdentifiableManager<Veiculo, Long, VeiculoDao> {

    public void salvar(Veiculo veiculo) {...77 linhas }

    public void inativar(Veiculo veiculo) {...6 linhas }

    public List<Veiculo> listar(VeiculoFilter filtro) {

        return getDao().listar(filtro);
    }

    public List<Veiculo> listarTodosVeiculos(){
        return getDao().listar();
    }

    public List<Veiculo> listarAtivosDisponiveis() {...8 linhas }

}
```

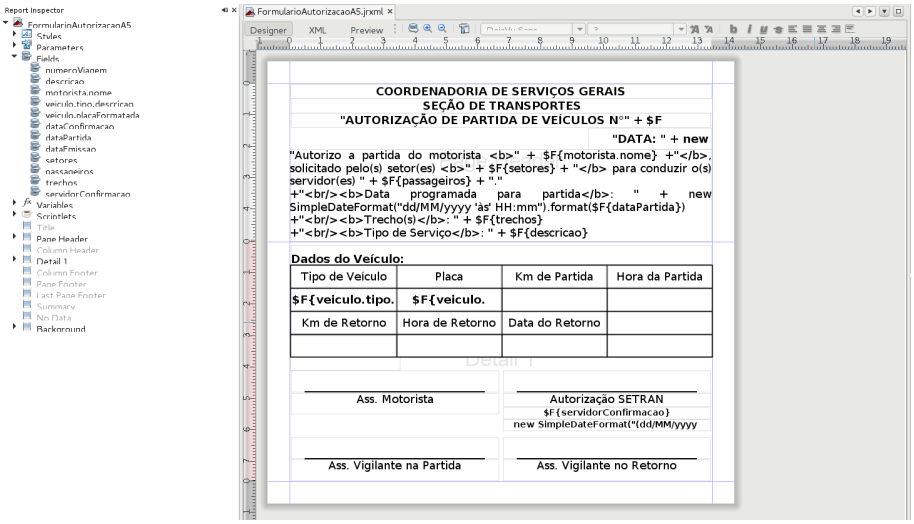
Fonte: SEDES

Figura 12 – DAO Java - Veículo

```java
import org.springframework.stereotype.Repository;

@Repository
public class VeiculoDao extends JpaDao<Veiculo, Long> {

    public List<Veiculo> listar(VeiculoFilter filtro) {

        SQLBuilder sql = new SQLBuilder("select v from Veiculo v where 1=1");

        if (filtro.getLocal() != null) {
            sql.add("and v.local = :local").param("local", filtro.getLocal().getCodigo());
        }
        if (filtro.getPlaca() != null && !filtro.getPlaca().isEmpty()) {
            sql.add("and v.placa like :placa").param("placa", "%" + filtro.getPlaca().toUpperCase() + "%");
        }
        if (filtro.getMarca() != null && !filtro.getMarca().getNome().isEmpty()) {
            sql.add("and v.marca = :marca").param("marca", filtro.getMarca());
        }
        if (filtro.getAtivo() != null) {
            sql.add("and v.ativo = :ativo").param("ativo", filtro.getAtivo());
        }
        if (filtro.getSituacao() != null) {
            sql.add("and v.situacao = :situacao").param("situacao", filtro.getSituacao().getCodigo());
        }
        if (filtro.getTipo() != null) {
            sql.add("and v.tipo = :tipo").param("tipo", filtro.getTipo().getCodigo());
        }
        sql.add("order by v.modelo");
        return criarQuery(sql).getResultList();
    }

    @Override
    public List<Veiculo> listar() {
        String sql = "select v from Veiculo v order by v.modelo";
        return criarQuery(sql).getResultList();
    }

}
```

Fonte: SEDES

Figura 13 – IReport- Veículo



Fonte: SEDES

Figura 14 – Bean do Formulário- Veículo

```
public void onImprimirFormulario() throws JRException, IOException {
    Map<String, Object> params = new HashMap<>();
    List<FormularioFilter> formList = new ArrayList<>();

    FormularioFilter formulario = new FormularioFilter(viagem);
    formList.add(formulario);

    CollectionReportBean report = new CollectionReportBean("FormularioAutorizacaoA5", params, formList);
    adicionarMensagemInformacao("Formulário da viagem " + viagem.getNumeroFormatado() + " gerado com sucesso");
    report.executePDF();
    FacesContext.getCurrentInstance().responseComplete();
    registrarLogAcessoFuncionalidade("imprimirFormulario", null, null);
}
```

Fonte: SEDES

4 Considerações Finais

A experiência de estágio no TRE-PB adicionou propriedade aos conhecimentos adquiridos ao longo da minha jornada como aluno e desenvolvedor de sistemas para internet. Durante os dois anos de estágio fui incluído em uma rotina de trabalho equivalente as teorias e práticas vistas e desenvolvidas em sala de aula. Fui amparado por excelentes profissionais, sempre dispostos a dar o seu melhor tanto para o ambiente de trabalho quanto para uma sociedade melhor e mais justa.

Assistir uma boa aula, realizar os exercícios em sala e ainda repetir todo esse processo durante o estágio proporcionou, sem dúvidas, um crescimento imensurável ao meu intelecto. Tive a oportunidade de participar de alguns projetos, dentre eles o Veículos: Sistema de Gestão de Frotas do TRE-PB , o qual participei já na primeira reunião até seu encerramento. Poder opinar, dar sugestões, incluir classes inteiras ao projeto, compilar o código e realizar o deploy da aplicação, até mesmo em produção me fez acreditar que eu estava no caminho correto e que seria possível me apresentar diante de qualquer empresa como um verdadeiro profissional.

Durante o projeto me deparei com situações de grande dificuldade, as vezes por não conseguir codificar a solução do problema, outras pelo tamanho da responsabilidade envolvida. Sempre fui encorajado a encontrar a solução, investigar, as vezes por dias. Fiz parte de uma equipe sempre pronta e disposta a resolver suas demandas de trabalho, esclarecer dúvidas, sempre discutindo possibilidades de melhorar. Sinto orgulho em dizer que, durante meu estágio, todos os desenvolvedores efetivos do TRE-PB foram alunos IFPB.

Saio satisfeito dessa jornada, sabendo que pude absorver o máximo do que foi exposto durante o curso e o estágio. Tratando-se de tecnologias, muito ainda é pouco. Tudo que aprendi e vivi serve apenas como base para minha carreira como desenvolvedor. Conquistei apenas a ponta do iceberg, tenho plena clareza que falta muito a aprender e evoluir ainda.

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 2 vezes nas páginas 10 e 12.

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/>. 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.

LIMA, M. A. V. de. **Devmedia: Gerando Relatórios com JasperReports**. 2012. Disponível em: <https://www.devmedia.com.br/gerando-relatorios-com-jasperreports/24798>. Acesso em: 12 de fevereiro de 2019. Citado na página 7.

SABBAGH, R. **Scrum**. 1. ed. Cidade: Casa do Código, 2014. Citado na página 5.

SATO, D. **DevOps**: Na prática: entrega de software confiável e automatizada. 1. ed. São Paulo: Casa do Código, 2014. Citado na página 16.

WIKIPEDIA. **Spring Framework**. 2017. Disponível em: <https://pt.wikipedia.org/wiki/Spring_Framework>. Acesso em: 8 de fevereiro de 2019. Citado na página 15.

WIKIPEDIA. **Apache Tomcat**. 2018. Disponível em: <https://pt.wikipedia.org/wiki/Apache_Tomcat>. Acesso em: 14 de fevereiro de 2019. Citado na página 8.

Anexos

ANEXO A – Modus 3.1

29/01/2019 13:12:08

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 *Scrum Master* do método ágil *Scrum*.
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

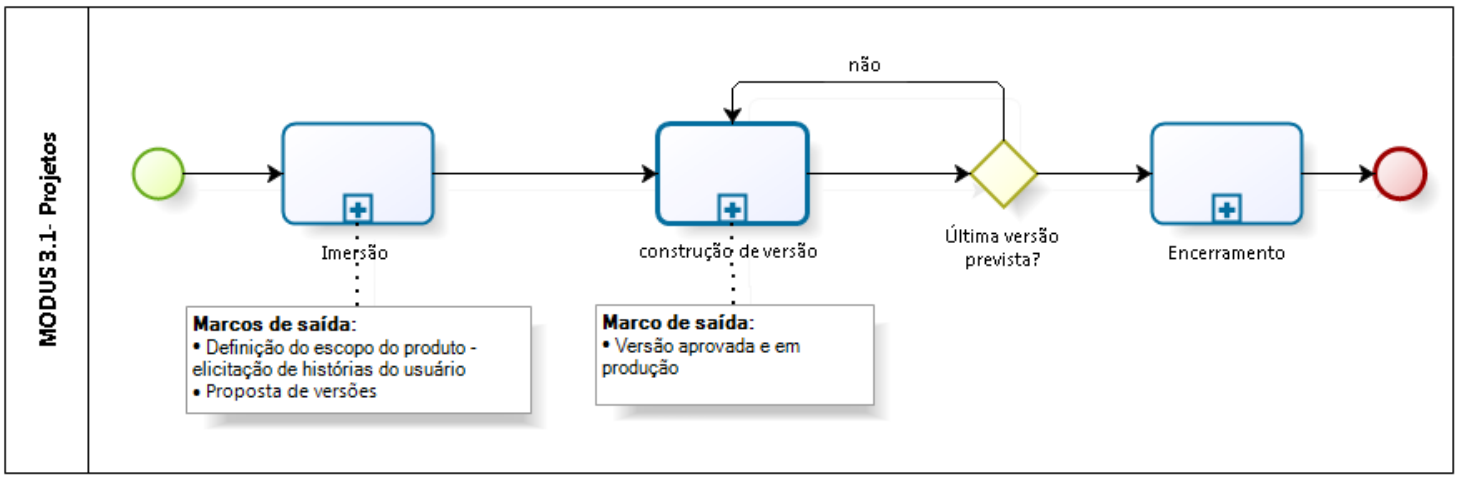


Ilustração 1: Fases do modelo de desenvolvimento para projetos

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.

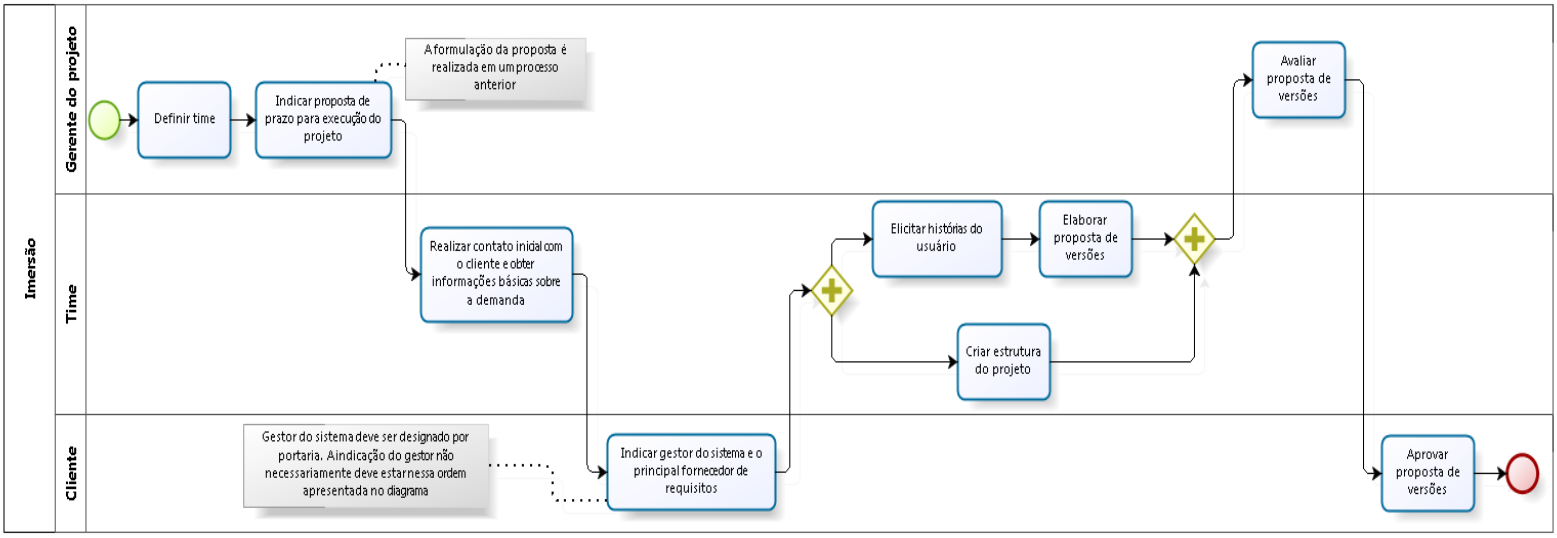


Ilustração 2: Fluxo da fase de imersão

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
• Projeto aprovado para execução	• Determinação do time (pessoas que executarão o projeto)

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
• Calendário de projetos da COSIS	• Indicação para o time do prazo sugerido

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	• Ata de reunião inicial

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	• Definição do gestor do sistema e do principal fornecedor de requisitos para o time

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	• Projeto no Redmine • Página do projeto • Inclusão do produto no catálogo • Arcabouço do sistema no SVN

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
• Projeto aprovado para execução • Ata de reunião inicial	• Histórias cadastradas no Redmine

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
• Histórias elicitadas no Redmine • Priorização do cliente • Sugestão de prazo de conclusão do projeto	• Proposta de versões do projeto no Redmine, indicando o escopo para cada uma delas.

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
• Histórias cadastradas no Redmine • Proposta de versões do projeto no Redmine	• Proposta de versões do projeto no Redmine acordada entre o time e o gerente

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
• Proposta de versões do projeto no Redmine acordada entre o time e o gerente	• Proposta de versões do projeto no Redmine acordada entre o time, o gerente e o cliente

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*.

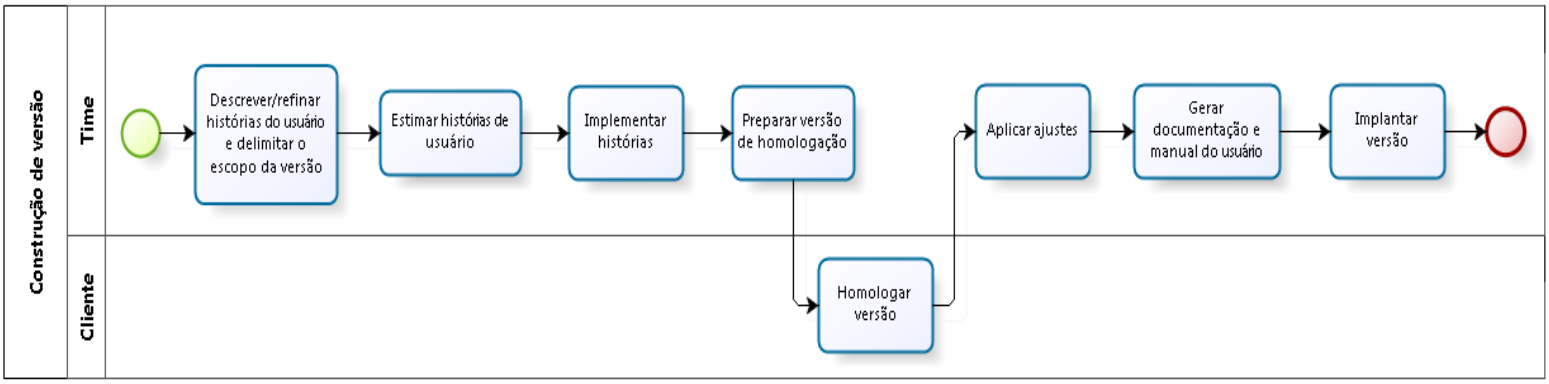


Ilustração 3: Fluxo de atividades da fase de construção de versão

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
• Histórias do usuário propostas para a versão • Feedback do cliente	• Histórias do usuário refinadas no Redmine • Escopo da versão ajustado

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
• Histórias escritas no Redmine	• Estimativas indicadas em cada história no Redmine

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

• Histórias do usuário detalhadas e refinadas selecionadas para a versão • Diagrama de modelagem de processos de negócio, se houver • Protótipos e outros artefatos produzidos na etapa de imersão	• Código-fonte da versão • Scripts de criação/atualização do banco de dados • Modelo entidade-relacionamento

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
• Código-fonte da versão • Scripts de banco de dados	• Versão implantada no ambiente de homologação • Base de dados para testes preparada • Roteiros para oficinas, se houver

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
• Versão implantada no ambiente de homologação	• Feedback do cliente para o time sobre aprovação da versão e

• Base de dados para testes preparada • Roteiro de oficinas, se houver	ajustes necessários • Ata da reunião de homologação

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
• Feedback do cliente para o time • Ata da reunião de homologação	• Código-fonte ajustado • Histórias do usuário ajustadas, se for o caso

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
• Versão implementada • Histórias do usuário detalhadas e refinadas selecionadas para a versão	• Catálogo de produtos atualizado • Manual do usuário elaborado ou atualizado

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
• Código-fonte da versão • Scripts de banco de dados	• Tag no sistema de controle de versões SVN • Versão implantada no ambiente de produção • Comunicação via e-mail da entrada em produção para o cliente e demais interessados

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.

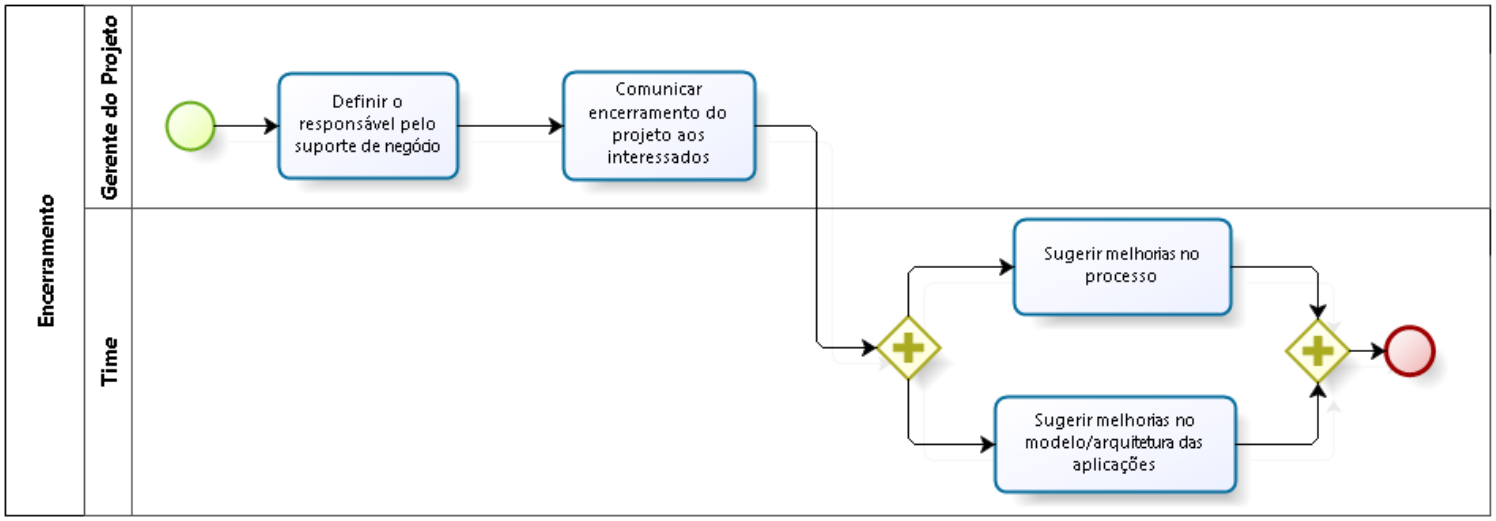


Ilustração 4: Fluxo de atividades da fase de encerramento do projeto

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
• Versões finalizadas	• Ata da reunião

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
• Versões finalizadas	• E-mail de encerramento do projeto ou documento/despacho de encerramento em processo administrativo dando publicidade aos interessados.

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
• Problemas enfrentados e soluções adotadas	• Melhorias cadastradas no Redmine

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
• Problemas enfrentados e soluções adotadas	• Melhorias cadastradas no Redmine

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.

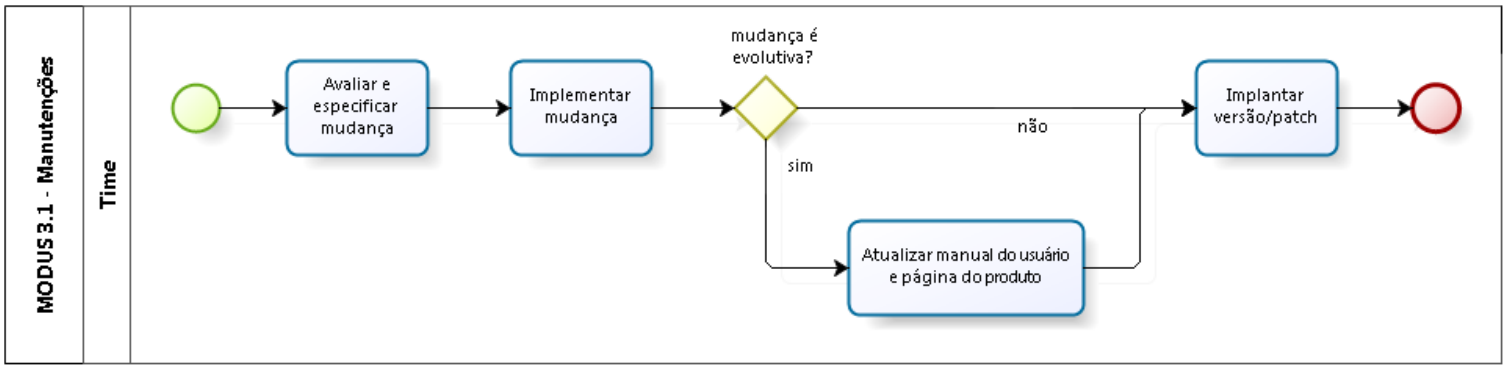


Ilustração 5: Fluxo de atividades do processo de manutenção de produtos

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
• Demanda selecionada	• Mudança especificada no Redmine como novas histórias do usuário, melhorias ou bugs • Prazo para entrega • Identificação do *patch* ou versão do produto a ser gerado

Implementar mudança

Ator: Time

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

Entradas	Saídas
• Mudança especificada no Redmine como novas histórias do usuário, melhorias ou bugs	• Código-fonte da versão • Scripts de criação/atualização do banco de dados • Modelo entidade-relacionamento

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
• Mudança especificada no Redmine como novas histórias do usuário, melhorias ou bugs • Mudança implementada	• Catálogo de produtos atualizado • Manual do usuário atualizado

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
• Código-fonte da versão • Scripts de banco de dados	• Tag no sistema de controle de versões SVN • Versão implantada no ambiente de produção • Comunicação via e-mail da entrada em produção para o cliente e demais interessados

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 informando o código verificador **0203840** e o código CRC **4B97C595**.

0002615-03.2017.6.15.8000 0203840v11