PDS TCEPR - Processo de Desenvolvimento de Software do TCEPR

Versão 1.0 (28/07/2017)

APRESENTAÇÃO

O Processo de Desenvolvimento de Software do TCEPR (PDS TCEPR) aborda as atividades ligadas ao desenvolvimento de software e também as atividades ligadas ao planejamento dos recursos para que o software tenha o ambiente necessário para o seu funcionamento.

O PDS TCEPR foi concebido com o objetivo de padronizar as atividades de desenvolvimento de software do Tribunal.

O PDS TCEPR foi instituído na Instrução Normativa nº 123/2016 e o seu detalhamento pode ser revisado a qualquer momento, gerando uma nova versão, considerando a dinamicidade da área de Tecnologia da Informação, de forma a promover a melhoria contínua, conforme Art. 58 da referida Instrução Normativa.

ÍNDICE

1. VISÃO GERAL

2. CICLO DE VIDA

Análise de Demanda

Início do Projeto ou Sustentação

Desenvolvimento de Software (processo iterativo e incremental)

- Planejamento
- Construção
- Qualidade
- Segurança
- Arquitetura
- Homologação
- Implantação
- Encerramento

Término do Projeto ou Sustentação

Transferência de Tecnologia

3. DISCIPLINAS

Gestão de Demandas

- Tarefa: Elaborar Proposta de Projeto
- Tarefa: Elaborar Pré-Projeto

Gestão de Projetos

- Tarefa: Realizar Reunião de Abertura de Projeto
- Tarefa: Definir Escopo do Projeto (priorização e estimativa de esforço)
- Tarefa: Emitir Ordem de Serviço
- Tarefa: Iniciar Iteração (planejamento)
- Tarefa: Realizar Reunião Diária
- Tarefa: Elaborar Ata de Reunião
- Tarefa: Encerrar Iteração (entrega/revisão)
- Tarefa: Avaliar Iteração (retrospectiva)
- Tarefa: Apurar Indicadores de Qualidade e Gestão
- Tarefa: Avaliar Indicadores de Qualidade e Gestão
- Tarefa: Receber Serviços
- Tarefa: Realizar Reunião de Encerramento de Projeto

Requisitos

- Tarefa: Identificar e Especificar Requisitos
- Tarefa: Definir Critérios de Aceite
- Tarefa: Especificar Caso de Uso
- Tarefa: Especificar História de Usuário
- Tarefa: Especificar Change Request
- Tarefa: Desenhar Protótipo
- Tarefa: Elaborar Diagrama de Negócio

Desenvolvimento

- Tarefa: Implementar (codificar) software
- Tarefa: Entregar software e roteiro de publicação

Testes

- Tarefa: Elaborar Plano de Teste
- Tarefa: Implementar (codificar) Testes Automatizados
- Tarefa: Executar Testes Automatizados
- Tarefa: Executar Testes Manuais (Critérios de Aceite e Exploratórios)

Qualidade

- Tarefa: Validar qualidade das entregas e dos testes

Segurança

- Tarefa: Validar segurança do software

Arquitetura

- Tarefa: Especificar Arquitetura do Projeto
- Tarefa: Revisar arquitetura do software

Homologação

- Tarefa: Homologar software

Implantação

- Tarefa: Publicar software no Ambiente de Desenvolvimento ou Testes
- Tarefa: Publicar software no Ambiente de Homologação
- Tarefa: Publicar software no Ambiente de Produção

Transferência de Tecnologia

- Tarefa: Elaborar ou Atualizar Documento de Transferência de Tecnologia

- Tarefa: Aceitar Transferência de Tecnologia

4. PRODUTOS

Ata de Reunião

Backlog da Iteração

Backlog do Produto

Caso de Uso

Change Request

Checklist do Desenvolvedor

Código-fonte

Critérios de Aceite

Diagrama de Negócio

Documento de Arquitetura do Projeto

Documento de Retrospectiva

Documento de Revisão de Arquitetura

Documento de Transferência de Tecnologia

Documento de Validação de Interface

Documento de Validação de Qualidade

Documento de Validação de Segurança

História de Usuário

Incremento de Software

Indicadores de Qualidade e Gestão

Item de Trabalho do Repositório

Ordem de Serviço

Plano de Teste

Pré-Projeto

Proposta de Projeto

Protótipo

Regras de Negócio

Relatório de Execução dos Testes Automatizados

Relatório de Execução dos Testes Manuais (Critérios de Aceite e Exploratórios)

Roteiro de Publicação

Solicitação de Serviço

Termo de Abertura de Projeto

Termo de Encerramento de Projeto

Termo de Recebimento de Serviço

5. PAPÉIS

Administrador de Banco de Dados

Administrador de Infraestrutura

Analista de Demandas

Analista Programador

Arquiteto

Comitê de Tecnologia da Informação

Diretor de Tecnologia da Informação

Diretor Geral

Gerente de Infraestrutura

Gerente de Projeto

Gerente de Sustentação

Product Owner

Scrum Master

Time de Desenvolvimento

Usuário (Área de Negócio)

6. PADRÕES

- Padrões de Arquitetura
- Padrões de Banco de Dados
- Padrões de Código
- Padrões de Layout e Interface
- Padrões de Qualidade
- Padrões de Segurança

7. MANUAIS

Manual de Contagem de Pontos de Função do TCEPR

PDS TCE-PR → VISÃO GERAL

1. VISÃO GERAL DO PDS TCEPR

O **Processo de Desenvolvimento de Software do TCEPR (PDS TCEPR)** define **processos** e um conjunto mínimo de **tarefas** que devem ser executadas por **pessoas** a fim de produzir **artefatos** durante o desenvolvimento de software, seguindo **padrões** definidos.

Desta forma, o PDS TCEPR está organizado nos seguintes capítulos:

- CICLO DE VIDA: O QUE fazer? (PROCESSOS);
- DISCIPLINAS: COMO fazer? (TAREFAS);
- PRODUTOS: Entradas e Saídas dos Processos (ARTEFATOS);
- PAPÉIS: QUEM deve fazer? (PESSOAS);
- PADRÕES: Boas Práticas;
- MANUAIS: Guias de instruções.

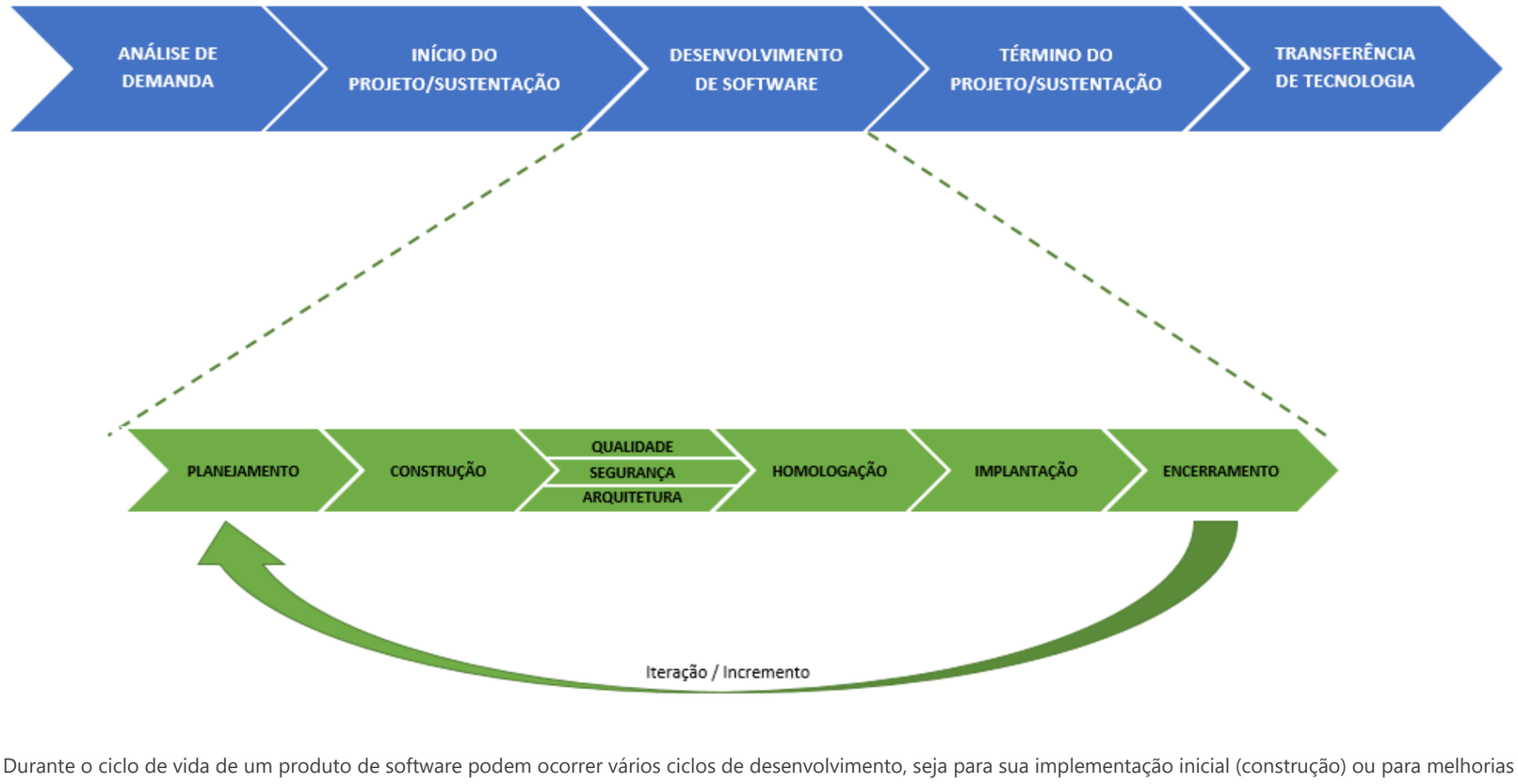
PDS TCE-PR → CICLO DE VIDA

2. CICLO DE VIDA DO PDS TCE-PR

O ciclo de vida do Processo de Desenvolvimento de Software do TCEPR (PDS TCEPR) é um conjunto de processos cujo esforço visa produzir um novo produto ou melhoria de um produto de software existente.

Este ciclo deve contemplar as seguintes fases:

- Análise de Demanda;
- Início do Projeto ou Sustentação;
- Desenvolvimento de Software (processo iterativo e incremental);
    - Planejamento;
    - Construção;
    - Qualidade;
    - Segurança;
    - Arquitetura;
    - Homologação;
    - Implantação;
    - Encerramento;
- Término do Projeto ou Sustentação;
- Transferência de Tecnologia.



Durante o ciclo de vida de um produto de software podem ocorrer vários ciclos de desenvolvimento, seja para sua implementação inicial (construção) ou para melhorias contínuas do produto (sustentação).

Os ciclos de melhoria podem ser desenvolvidos de duas maneiras: através de projetos ou por pequenas manutenções evolutivas realizadas pela sustentação.

ANÁLISE DE DEMANDA

Uma vez identificado um problema ou necessidade de negócio que requeira modificação ou criação de software para atender à requisição, ou aquisição de solução de TI, a área de negócio deverá registrar a demanda no sistema de solicitação de serviços em uso, que iniciará o processo da análise.

A Diretoria de Tecnologia da Informação (DTI) receberá a solicitação através de sua área de atendimento aos usuários (*Service Desk*), que irá avaliar a necessidade requerida. O pedido poderá ser atendido de imediato, no caso de incidente conhecido e de rápida atuação.

Nas hipóteses em que a solicitação não possa ser concluída pelo atendimento (*Service Desk*), será direcionada à área designada no portfólio de soluções de TI, assim como no acordo de nível de serviço vigente na DTI.

- Em caso de software próprio (não de terceiros), a área de sustentação dará início ao processo de desenvolvimento da correção necessária, considerando a fila de demandas, priorizando-se a correção de erros (*bugs*).
- Em caso de software de terceiros, a área responsável pelo software deverá dar encaminhamento de acordo com os termos do respectivo contrato.

A área de sustentação, sempre que a solicitação demandar esclarecimentos perante a unidade solicitante, encaminhará o pedido à área de demandas para as análises necessárias.

A área de demandas avaliará o esforço necessário para o desenvolvimento da nova funcionalidade, de forma a definir se a demanda se enquadra como sustentação (pequena manutenção evolutiva) ou se constitui necessidade a ser enfrentada mediante instituição de projeto.

- Em se tratando de pequena manutenção, a demanda será encaminhada à área de sustentação para o respectivo desenvolvimento a ser realizado no padrão fixado nesta normativa e respeitando-se a meta de estoque de demandas acordada com o Comitê de TI.
- Caso a área de sustentação não possa assumir o respectivo desenvolvimento, devolverá a solicitação, com a devida justificativa, à área de demandas, a fim de que esta encaminhe o pedido ao Comitê de TI para deliberação, acompanhado de informações acerca da situação atual da força de trabalho e da fila de demandas da sustentação, a fim de que o Comitê delibere acerca da priorização das demandas e promova eventual ajuste necessário da meta de estoque da sustentação.
- Novas funcionalidades que requeiram maior esforço, não se enquadrando nos critérios de sustentação, deverão ser objeto de avaliação pela área de demandas, documentando-se, conforme modelo padrão definido no detalhamento do PDS. A documentação será encaminhada ao Comitê de Tecnologia da Informação para as deliberações necessárias.
- A documentação prevista no tópico anterior será elaborada pela DTI em conjunto com a(s) área(s) de negócio solicitante(s) e deverá contemplar, no mínimo, as seguintes informações:
    - a) estimativa preliminar de custo, esforço, equipe e tempo necessários à implantação da solução;
    - b) principais riscos identificados, inclusive quanto à possível perda de oportunidade;
    - c) indicação das interações com outras soluções de TI que serão necessárias ao funcionamento da nova solução;
    - d) descrição da demanda, consignando os seus motivos e o escopo com as principais entregas previstas;
    - e) benefícios esperados;
    - f) documento de descrição do processo de trabalho a ser informatizado, o qual deverá ser elaborado pela(s) área(s) de negócio demandante(s);
    - g) o alinhamento da demanda com o Planejamento Estratégico do Tribunal e de TI;
    - h) a existência de soluções similares no TCEPR;
    - i) quando aplicável e possível, a comparação com soluções similares disponíveis para comercialização, softwares livres ou soluções passíveis de cessão por convênio com outras instituições;
    - j) o interesse de outras áreas do TCEPR (além da requisitante) na solução demandada.

A DTI deverá encaminhar, em conjunto com a documentação prevista acima, as informações da situação atual dos projetos em curso, as equipes alocadas em cada um, a previsão de conclusão, o escopo pendente de desenvolvimento, bem como informar os recursos disponíveis integral ou parcialmente.

Das deliberações do Comitê de TI constarão as propostas aprovadas preliminarmente, as que devam ser reapresentadas em época oportuna, os projetos que devam ser paralisados, cancelados ou objeto de redução de equipe, bem como a ordem de priorização para execução.

- As deliberações do Comitê de TI serão submetidas à aprovação da Presidência do Tribunal.

A avaliação da demanda pelo Comitê de TI e aprovação pela Presidência é indispensável para o início das atividades de provimento de novas soluções de TI.

Uma vez aprovada a continuidade da solicitação, a área de gestão de demandas, em conjunto com a(s) área(s) de negócio solicitante(s), deverá elaborar o pré-projeto, conforme modelo descrito no detalhamento do PDS, sendo submetido à apreciação do Comitê de TI para aprovação.

- Caso o pré-projeto não esteja aderente à solicitação inicialmente aprovada, será submetido à nova apreciação da Presidência do Tribunal.

A direção da DTI será comunicada da decisão da gestão da Casa, quando então indicará à Direção-Geral, para cada pré-projeto aprovado, um servidor para atuar como gerente de projeto junto à DTI. Formalizada a designação do gerente de projeto junto à DTI, dar-se-á início ao respectivo processo de desenvolvimento.

Disciplinas / Tarefas:	• Gestão de Demandas → Tarefa - Elaborar Proposta de Projeto • Gestão de Demandas → Tarefa - Elaborar Pré-Projeto • Requisitos → Tarefa - Identificar e Especificar Requisitos • Requisitos → Tarefa: Desenhar Protótipo

INÍCIO DO PROJETO OU SUSTENTAÇÃO

O início do projeto dar-se-á através da realização de uma reunião de abertura. Esta reunião deverá ser convocada pelo gerente do projeto junto à DTI, com a participação da(s) área(s) de negócio envolvida(s) no projeto. O produto desta reunião é o documento Termo de Abertura do Projeto, conforme modelo definido no detalhamento do PDS.

O início do processo de sustentação (pequena manutenção) dar-se-á por meio de registro no sistema de solicitação de serviços em uso.

Disciplinas / Tarefas:	• Gestão de Projetos → Tarefa: Realizar Reunião de Abertura de Projeto • Gestão de Projetos → Tarefa: Emitir Ordem de Serviço

DESENVOLVIMENTO DE SOFTWARE

Ambos os processos, de desenvolvimento de novo sistema/produto ou de sustentação (pequena manutenção), devem seguir os processos definidos no Processo de Desenvolvimento de Software do TCEPR (PDS TCEPR), de forma a garantir a entrega de um produto de qualidade e dentro do prazo acordado.

O gerente do projeto junto à DTI deverá atualizar periodicamente, no padrão e prazos definidos no detalhamento do PDS TCEPR, as informações acerca do cronograma, das entregas previstas, pendentes e concluídas, riscos detectados e ações para mitigação dos riscos. Deverá ainda informar à direção da DTI situações que possam afetar o cronograma do projeto e outras questões que demandem atuação da gestão.

Todos os projetos, na fase inicial, deverão ser submetidos às áreas de Arquitetura de TI, Qualidade e Segurança para as definições necessárias em relação aos padrões a serem utilizados, conforme fluxo previsto no PDS TCEPR.

A equipe de Infraestrutura deverá intervir no projeto, nos termos do fluxo fixado no PDS TCEPR, incluindo o detalhamento do PDS TCEPR, a fim de definir os itens afetos à Infraestrutura.

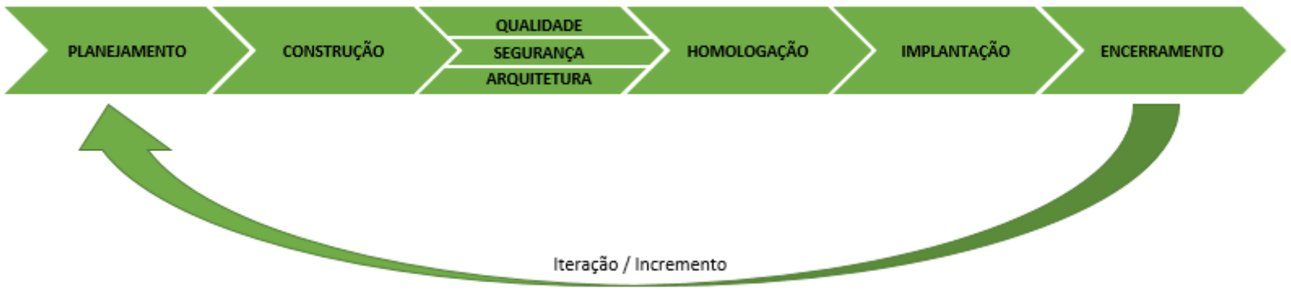
A avaliação do modelo de dados proposto pela equipe do projeto deverá ser realizada pela área de Administração de Dados, para especificar as informações que podem ser reaproveitadas de outros sistemas e/ou bases de dados, bem como garantir a atualização do Dicionário de Dados do Tribunal com informações do novo sistema.

O projeto deverá contemplar os critérios de aceite definidos pela área de negócio responsável pelo projeto, em conjunto com a DTI, conforme definido no PDS TCEPR.

Os projetos, além dos padrões de qualidade fixados no PDS TCEPR e em seu detalhamento, deverão ser objeto de aferição dos indicadores de qualidade e gestão, conforme definido pelo Núcleo de Qualidade. A definição dos indicadores vigentes deverá estar acessível aos servidores da unidade, em meio eletrônico.

O Desenvolvimento de Software é um processo iterativo e incremental, em que cada ciclo é constituído pelas fases:

- Planejamento
- Construção
- Qualidade
- Segurança
- Arquitetura
- Homologação
- Implantação
- Encerramento



DESENVOLVIMENTO DE SOFTWARE: PLANEJAMENTO

Um ciclo de desenvolvimento de software inicia-se com a fase de planejamento, em que serão definidas as entregas a serem realizadas pela equipe.

Antes do início do primeiro ciclo deve ser realizada a definição do Escopo do Projeto, confirmando o escopo definido no Pré-Projeto ou realizando as alterações necessárias em função da alteração de contexto desde a sua elaboração.

As definições de arquitetura do projeto, dos requisitos de qualidade e segurança, bem como demais características, devem ser realizadas antes do início do primeiro ciclo, de modo a comunicar à equipe as diretrizes a serem seguidas na construção da solução.

A cada ciclo de desenvolvimento deve ser realizada uma reunião de planejamento, que marcará o Início da Iteração.

Disciplinas / Tarefas:	• Gestão de Projetos → Tarefa: Definir Escopo do Projeto • Arquitetura → Tarefa: Especificar Arquitetura do Projeto • Gestão de Projetos → Tarefa: Iniciar Iteração (planejamento)

DESENVOLVIMENTO DE SOFTWARE: CONSTRUÇÃO

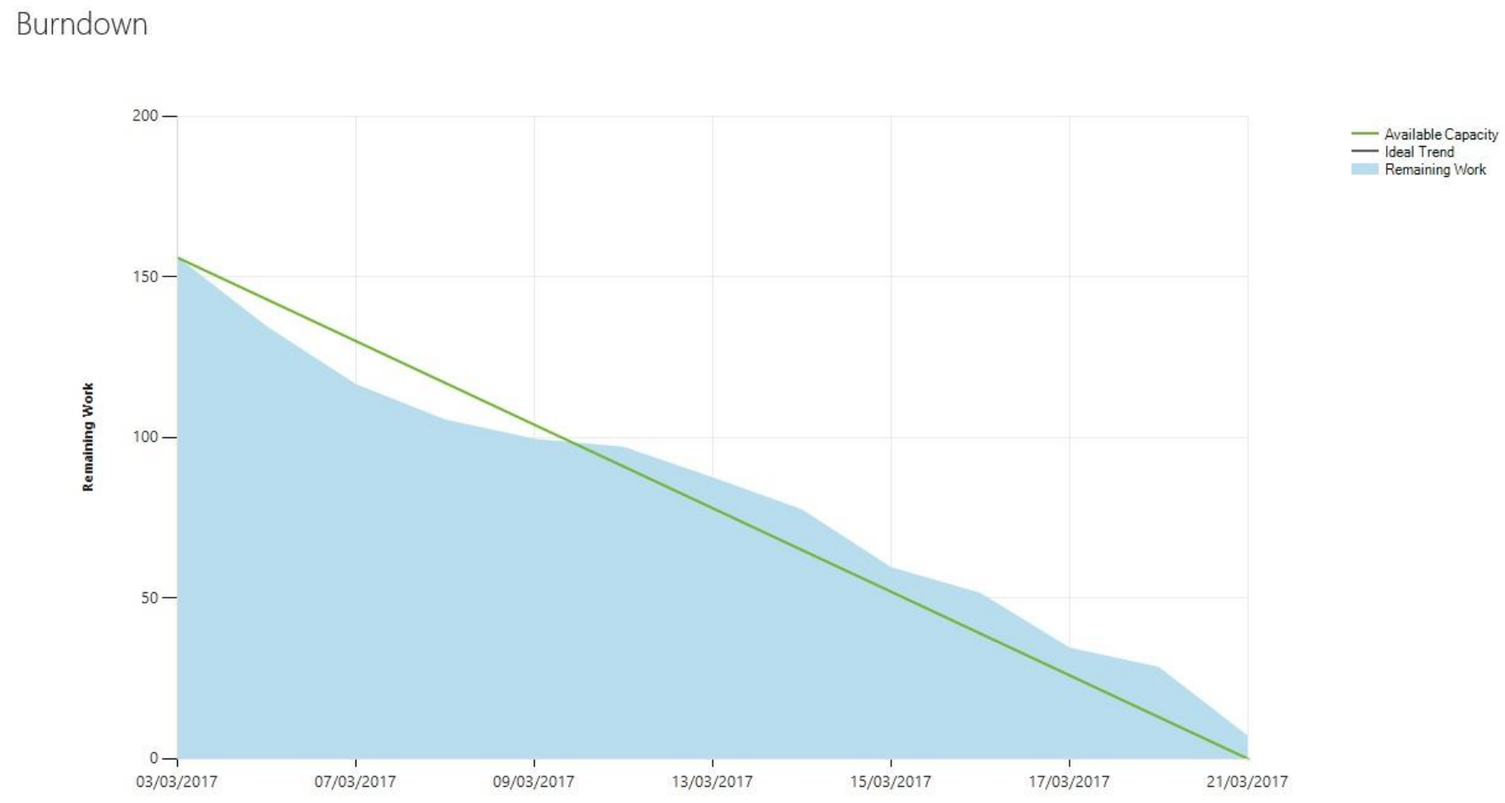
Durante a fase de construção do ciclo de desenvolvimento de software, as equipes devem executar as tarefas seguindo a ordem de prioridade definida pelo Product Owner.

Os membros da equipe devem sinalizar as tarefas que estão executando atribuindo-as a si e alterando o status da tarefa para "*In Progress*" (em andamento).

À medida que a tarefa for concluída deve-se alterar o seu status para "*Done*" (feito).

Cada Time de Desenvolvimento deverá definir o seu número máximo de itens de trabalho em andamento, ou "*Work In Progress*" (WIP).

Com base na atualização dos status e das estimativas das tarefas, ao longo da iteração, a equipe poderá utilizar o gráfico de *burndown* para acompanhar o andamento da iteração.



Diariamente deve ser realizada pela equipe uma breve reunião em que cada membro deve relatar de forma sucinta o que foi feito no dia anterior, o que será feito no dia atual e se há impedimentos ou riscos para o projeto. Caso hajam impedimentos relatados pela equipe, o Scrum Master deverá buscar removê-los o mais breve possível, enquanto a equipe dará prosseguimento às demais atividades planejadas na iteração, conforme ordem de prioridade definida. Os

riscos identificados devem ser gerenciados pela equipe de projeto.

As atividades devem ser realizadas pela equipe seguindo os padrões definidos no detalhamento do PDS TCEPR, principalmente quanto aos padrões de Qualidade, Segurança e Arquitetura.

Disciplinas / Tarefas:	• Requisitos → Tarefa: Identificar e Especificar Requisitos • Requisitos → Tarefa: Definir Critérios de Aceite • Requisitos → Tarefa: Especificar Caso de Uso • Requisitos → Tarefa: Especificar História de Usuário • Requisitos → Tarefa: Especificar Change Request • Requisitos → Tarefa: Desenhar Protótipo • Requisitos → Tarefa: Elaborar Diagrama de Negócio • Desenvolvimento → Tarefa: Implementar (codificar) software • Desenvolvimento → Tarefa: Entregar software e roteiro de publicação • Testes → Tarefa: Elaborar Plano de Teste • Testes → Tarefa: Implementar (codificar) Testes Automatizados • Testes → Tarefa: Executar Testes Automatizados • Testes → Tarefa: Executar Testes Manuais (Critérios de Aceite e Exploratórios) • Transferência de Tecnologia → Tarefa: Elaborar ou Atualizar Documento de Transferência de Tecnologia • Gestão de Projetos → Tarefa: Realizar Reunião Diária • Gestão de Projetos → Tarefa: Elaborar Ata de Reunião

DESENVOLVIMENTO DE SOFTWARE: QUALIDADE

Objetiva verificar se as boas práticas definidas em Padrões de Qualidade foram seguidas.

Disciplinas / Tarefas:	• Qualidade → Tarefa: Validar qualidade das entregas e dos testes

DESENVOLVIMENTO DE SOFTWARE: SEGURANÇA

Objetiva verificar se as boas práticas definidas em Padrões de Segurança foram seguidas.

Disciplinas / Tarefas:	• Segurança → Tarefa: Validar segurança do software

DESENVOLVIMENTO DE SOFTWARE: ARQUITETURA

Objetiva verificar se as boas práticas definidas em Padrões de Arquitetura foram seguidas.

Disciplinas / Tarefas:	• Arquitetura → Tarefa: Revisar arquitetura do software

DESENVOLVIMENTO DE SOFTWARE: HOMOLOGAÇÃO

Objetiva validar a conformidade do software desenvolvido com os requisitos de negócio e os critérios de aceite.

Disciplinas / Tarefas:	• Homologação → Tarefa: Homologar software

DESENVOLVIMENTO DE SOFTWARE: IMPLANTAÇÃO

Contempla as atividades necessárias para publicação do software construído nos ambientes de desenvolvimento, testes, homologação e produção.

Esta fase pode também envolver outras atividades, tais como treinamento, manualização, entre outras.

Disciplinas / Tarefas:	• Implantação → Tarefa: Publicar software no Ambiente de Desenvolvimento ou Testes • Implantação → Tarefa: Publicar software no Ambiente de Homologação • Implantação → Tarefa: Publicar software no Ambiente de Produção

DESENVOLVIMENTO DE SOFTWARE: ENCERRAMENTO

O encerramento de um ciclo de desenvolvimento deve ser marcado por uma reunião de entrega/revisão da iteração, onde a equipe apresentará ao cliente os produtos gerados durante o ciclo de desenvolvimento.

Após a reunião de entrega/revisão, a equipe deve realizar uma reunião de retrospectiva, quando devem ser avaliados pela equipe o que foi bom na iteração e o que pode ser melhorado na próxima iteração, registrando-se as lições aprendidas. Neste momento, a equipe também deve apurar e avaliar os Indicadores de Qualidade e Gestão.

Disciplinas / Tarefas:	• Gestão de Projetos → Tarefa: Encerrar Iteração (entrega/revisão) • Gestão de Projetos → Tarefa: Avaliar Iteração (retrospectiva) • Gestão de Projetos → Tarefa: Apurar Indicadores de Qualidade e Gestão • Gestão de Projetos → Tarefa: Avaliar Indicadores de Qualidade e Gestão

TÉRMINO DO PROJETO OU SUSTENTAÇÃO

O processo de encerramento do projeto dar-se-á através da realização de uma reunião de encerramento. Esta reunião deverá ser convocada pelo gerente do projeto junto à Diretoria de Tecnologia da Informação (DTI), com a participação da área de negócio envolvida no projeto. O produto desta reunião é o documento Termo de Encerramento do Projeto, conforme modelo apresentado no detalhamento do PDS TCEPR.

O gerente do projeto junto à DTI deverá encaminhar o documento preenchido e assinado para área de negócio, que deverá emitir o aceite do projeto, mediante assinatura no próprio documento Termo de Encerramento.

O Termo de Encerramento do Projeto será arquivado, em meio eletrônico, junto aos demais documentos que integraram o projeto.

O encerramento do processo de sustentação dar-se-á através da realização de registro no sistema próprio de controle de solicitações de serviços em uso (*Service Desk*).

Disciplinas / Tarefas:	• Gestão de Projetos → Tarefa: Realizar Reunião de Encerramento de Projeto

TRANSFERÊNCIA DE TECNOLOGIA

O processo de transferência de tecnologia, no que se refere aos sistemas produzidos pelo desenvolvimento de novos sistemas ou às alterações implementadas em sistemas já em produção e efetivo uso, tem como objetivo repassar o conhecimento necessário para áreas de sustentação e infraestrutura. Estas áreas irão sustentar a aplicação em produção, tanto para o suporte à aplicação, bem como para os recursos tecnológicos necessários para a funcionalidade plena do sistema.

No caso de desenvolvimento de novos sistemas, após o término do projeto, o gerente do projeto junto à Diretoria de Tecnologia da Informação (DTI) deverá realizar o processo de transferência de tecnologia através dos documentos de Transferência de Tecnologia, conforme definido no detalhamento do PDS TCEPR.

Após a transferência de tecnologia será acordado um período no qual uma parte da equipe do projeto será responsável pela solução de falhas do sistema (*bugs*), sendo acompanhada por pessoa(s) designada(s) da área de sustentação.

Disciplinas / Tarefas:	• Transferência de Tecnologia → Tarefa: Elaborar ou Atualizar Documento de Transferência de Tecnologia • Transferência de Tecnologia → Tarefa: Aceitar Transferência de Tecnologia

3. DISCIPLINAS

Uma disciplina é uma área de conhecimento da engenharia de software à qual se relaciona uma coleção de tarefas do projeto de desenvolvimento. Embora seja comum executar simultaneamente tarefas que pertençam a várias disciplinas, separar estas tarefas em disciplinas distintas é uma forma eficaz de organizar o conteúdo, tornando mais fácil a compreensão.

As disciplinas estabelecidas no PDS TCEPR são:

- Gestão de Demandas
- Gestão de Projetos
- Requisitos
- Desenvolvimento
- Testes
- Qualidade
- Segurança
- Arquitetura
- Homologação
- Implantação
- Transferência de Tecnologia

GESTÃO DE DEMANDAS

Esta disciplina agrupa as atividades relacionadas ao processo de Análise de Demanda.

As tarefas definidas nesta disciplina possuem os objetivos de:

- identificar e analisar problemas ou necessidades das áreas de negócio, propondo soluções que envolvam a criação, modificação ou aquisição (pesquisa, prospecção e orçamentação) de soluções de TI (software, sistemas ou serviços);
- analisar demandas das áreas de negócio, através da identificação de requisitos, objetivos e benefícios esperados;
- identificar o alinhamento da demanda ao Planejamento Estratégico do Tribunal;
- estimar prazos e custos de desenvolvimento de solução de TI para atendimento da demanda, além de riscos envolvidos, a fim de fornecer subsídios para avaliação do Comitê de Tecnologia da Informação.

Tarefas:	• Tarefa: Elaborar Proposta de Projeto • Tarefa: Elaborar Pré-Projeto

Tarefa: Elaborar Proposta de Projeto

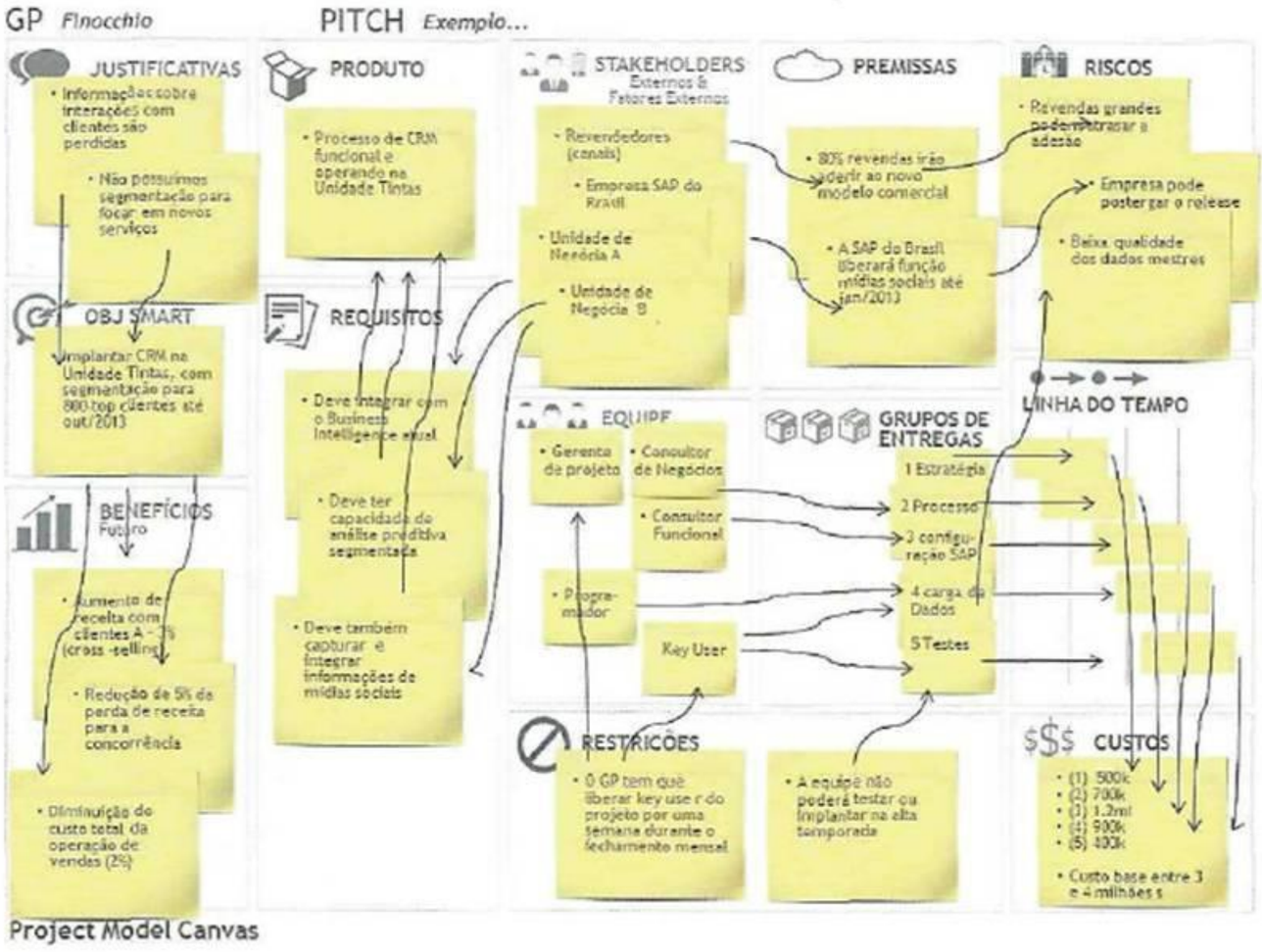
Entradas:	Solicitação de Serviço
Saídas:	Proposta de Projeto
Papéis:	Analista de Demandas
Disciplinas / Tarefas relacionadas:	Gestão de Demandas → Tarefa: Elaborar Pré-Projeto Requisitos → Tarefa: Identificar e Especificar Requisitos Requisitos → Tarefa: Desenhar Protótipo Requisitos → Tarefa: Elaborar Diagrama de Negócio

O documento de Proposta de Projeto deverá seguir o fluxo de aprovação definido na Instrução Normativa nº 123/2016, podendo resultar em um Pré-Projeto e/ou Projeto.

Uma proposta de projeto deve ser simples, rápida, objetiva e que apresente uma visão geral do produto a ser feito, com base nestas premissas foi selecionado um modelo de trabalho elaborado pelo professor José Finocchio chamado *Canvas Project* (http://www.pmcanvas.com.br/).

Neste modelo usa-se uma dinâmica junto ao cliente, onde são levantados os seguintes tópicos:

- Justificativas: Descreve o problema que existe atualmente na unidade que justifica a necessidade deste projeto.
- Objetivo *SMART*: Descreve o objetivo do projeto de forma Específica, Mensurável, Atingível, Realista e Temporizável.
- Benefícios: Descreve de forma objetiva quais os benefícios que o projeto trará para a unidade/organização.
- *Stakeholders* Externos: Identificação dos *stakeholders* e fatores externos que afetam, interagem e/ou causam influência com o projeto.
- *Stakeholders* Internos: Identificação dos papéis que estarão subordinados ao projeto, mesmos que estes estejam parcialmente ou temporariamente.
- Produto: Descreve o produto a ser construído.
- Requisitos: Descreve as características do produto de acordo com as necessidades e desejos do cliente.
- Premissas: Descreve suposições que precisam ser dadas como certas ou verdadeiras para que se possa concretizar o que está no planejamento.
- Restrições: Descreve restrições que limitam o trabalho ou as entregas produzidas pela equipe.
- Riscos: Identificação dos possíveis riscos que afetam ou podem afetar o projeto.
- Grupo de Entregas: Descreve produtos, serviços e resultados produzidos pelo projeto, agrupados em alto nível.
- Linha de Tempo ou Cronograma: Relaciona os grupos de entrega em um cronograma macro.
- Custos: Descreve a estimativa de custos para cada grupo de entrega.

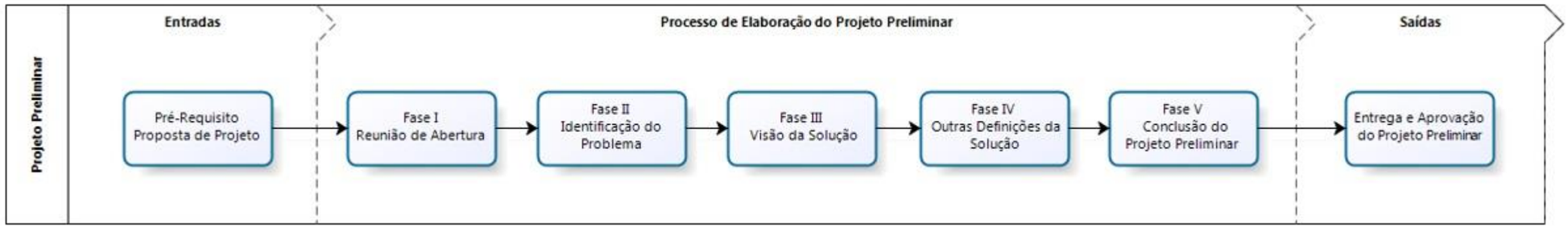


Uma vez finalizada esta dinâmica, as informações devem ser repassadas para o documento "Proposta de Projeto", que deve ser revisado pela equipe.

Tarefa: Elaborar Pré-Projeto

Entradas:	Proposta de Projeto
Saídas:	Pré-Projeto
Papéis:	Analista de Demandas
Disciplinas / Tarefas relacionadas:	Gestão de Demandas → Tarefa: Elaborar Proposta de Projeto Requisitos → Tarefa: Identificar e Especificar Requisitos Requisitos → Tarefa: Desenhar Protótipo Requisitos → Tarefa: Elaborar Diagrama de Negócio

A elaboração de um Pré-Projeto, ou Projeto Preliminar, deve seguir a metodologia apresentada abaixo.



FASE I - Reunião de Abertura do Pré-Projeto

Uma vez elaborado o Plano de Projeto e aprovada a continuidade dos trabalhos, o próximo passo é a formalização do início dos trabalhos para a elaboração do pré-projeto.

É de suma importância a participação dos diretores de TI e da(s) área(s) de negócio envolvida(s), uma vez que decisões estratégicas serão tomadas, acordadas e ratificadas.

Nesta reunião serão abordados, definidos e/ou ratificados os seguintes termos:

- O problema (justificativa);
- Os objetivos e metas (objetivo *SMART*);
- As partes interessadas (*stakeholders*);
- A metodologia de trabalho (número de reuniões semanais, duração das reuniões, local, etc.);
- A equipe responsável pela elaboração do pré-projeto;
- O líder da equipe responsável pela elaboração do pré-projeto;
- A duração do trabalho de desenvolvimento do pré-projeto (cronograma);

- Os artefatos esperados como resultado do pré-projeto (requisitos, protótipos de telas, restrições, previsão de custos e de tempo de desenvolvimento do projeto etc.).

Metodologia de Trabalho

O processo de elaboração do Projeto Preliminar deve ser realizado através de reuniões diárias entre os membros da equipe de projeto. No início dos trabalhos, recomenda-se a realização de reuniões diárias de 1h. Com o decorrer do tempo, segundo o entendimento da equipe, estas reuniões podem ser mais espaçadas e/ou ter uma duração um pouco mais longa.

IMPORTANTE: É necessário total comprometimento e disponibilidade dos membros da equipe durante o período de elaboração do Projeto Preliminar.

As boas práticas neste processo definem que:

- o período de elaboração do Projeto Preliminar não deve exceder a 30 dias úteis (*);
- as reuniões diárias devem ter duração máxima de 1 hora, iniciando-se e encerrando-se pontualmente nos horários marcados.

(*) Obs.: Este prazo pode ser reavaliado de acordo com o tamanho do projeto, porém caso o tempo de elaboração do Projeto Preliminar estenda-se demasiadamente, corre-se o risco do Projeto Preliminar transformar-se no Projeto propriamente dito. O Projeto Preliminar deve ater-se somente aos aspectos relevantes para o entendimento das necessidades, sem entrar nos seus detalhes.

Utilizar o portal colaborativo para divulgação dos artefatos, atas de reuniões, etc. Para o armazenamento e compartilhamento destes documentos deverá ser criada uma nova biblioteca para o projeto no Portal de Pré-Projetos.

DICA: Sugere-se que as reuniões de trabalho sejam registradas em ata.

Partes Interessadas (*Stakeholders*)

Antes de iniciar a elaboração do Projeto Preliminar é importante identificar as partes interessadas (*stakeholders*):

- Patrocinador (*Sponsor*): Indivíduo/grupo, interno ou externo, que provê os recursos financeiros para o projeto;
- Unidades (ou organizações) envolvidas no projeto: Diretoria de Tecnologia da Informação (DTI), outras unidades do Tribunal e/ou outras organizações.

NOTA: *Stakeholders* (ou partes interessadas) são indivíduos e organizações ativamente envolvidos no projeto, cujos interesses são afetados (positiva ou negativamente) por ele, ou que exercem influência sobre o mesmo. Incluem o próprio Tribunal e suas unidades, outras organizações, os membros da equipe de projeto e o patrocinador (*sponsor*) do projeto. Inclui também partes externas, como os jurisdicionados e a sociedade em geral.

IMPORTANTE: Em caso de dúvidas e/ou divergências relacionadas às diretrizes do projeto, o patrocinador (*sponsor*) deve ser consultado.

Equipe Responsável pela Elaboração do Projeto Preliminar

A elaboração do Projeto Preliminar é um trabalho que deve ser realizado em conjunto pela Diretoria de Tecnologia da Informação (DTI) e pelas áreas de negócio (unidades do Tribunal e/ou outras organizações) interessadas no projeto.

Considerando que o espaço de tempo para a elaboração do pré-projeto é pequeno e que a avaliação de prazo e preço será realizada a partir dos requisitos nele identificados, os técnicos designados para compor a equipe de trabalho devem ter conhecimento do processo e autonomia para definir os requisitos que representam o consenso da sua unidade de negócio.

Deve ser definido um líder da equipe responsável pela elaboração do projeto preliminar, que terá o papel de "facilitador" dos trabalhos, sendo responsável por:

- planejar e acompanhar as atividades de elaboração do Projeto Preliminar;
- agendar e conduzir as reuniões de trabalho;
- consolidar as informações elaboradas pelos integrantes da equipe.

NOTA: Os integrantes da equipe responsável pela elaboração do projeto preliminar não participarão, necessariamente, da execução do projeto, quando aprovado.

NOTA: Embora Objetivos e Metas possuam conceitos semelhantes, há diferenças em suas definições:

- Objetivo é a descrição daquilo que se pretende alcançar (alvo qualitativo).
- Meta é a definição do objetivo em termos quantitativos, e com um prazo determinado (alvo quantitativo).

Em outras palavras, a META é a quantificação de um OBJETIVO.

Ao final desta fase deverá ser criado o seguinte documento:

- Termo de Abertura de Pré-Projeto (TAPP).

DOWNLOAD: Documento Modelo de Termo de Abertura de Pré-Projeto: Termo de Abertura de Pré-Projeto - [NomeProjeto].docx

FASE II - Identificação do Problema

Nesta fase será iniciada a elaboração do Documento do Projeto Preliminar (DPP).

Pré-Requisitos

Antes do início desta fase recomenda-se:

Nivelamento da Equipe

Considerando que a equipe técnica, que irá trabalhar na elaboração do pré-projeto, seja composta de pessoas que podem não conhecer o que foi discutido e acordado nas fases anteriores, para nivelamento dos conceitos, é obrigatório a leitura dos documentos TADPP e TAPP por todos os componentes da equipe antes do início dos trabalhos de elaboração do pré-projeto. Isto busca introduzir as partes interessadas à definição clara e objetiva do problema, do cenário atual e das necessidades e oportunidades do projeto proposto.

DICA: Recomenda-se a participação da equipe responsável pela elaboração do pré-projeto na reunião de abertura, caso já haja uma pré-definição de seus integrantes.

Benchmarking

Caso necessário, todo o processo de benchmark (comparação com os pares - TCEs, TCU, Órgãos de Controle e Fiscalização -, pesquisa de mercado etc.) deverá ter sido realizado antes do início da elaboração do pré-projeto.

NOTA: Benchmarking é um processo de comparação de produtos, serviços e práticas empresariais, e é um importante instrumento de gestão das empresas.

Identificação das Necessidades do Cliente

Como primeira abordagem ao entendimento das necessidades do cliente, a situação ideal é a apresentação, por parte da área de negócio, de um descritivo com todas as necessidades que precisam ser atendidas.

Análise do Cenário Atual (As Is)

De posse do descritivo das necessidades, dos levantamentos durante as reuniões de trabalho e, até mesmo, de visitas à(s) unidade(s) para observação e acompanhamento dos processos, obter da equipe de negócios o conhecimento da situação atual como um todo (as is).

Identificação dos produtos existentes e desejados

Identificar os produtos (relatórios, consultas, portais, etc.) existentes e os desejados.

Identificação de controles paralelos

Identificar os controles paralelos (planilhas excel, doc, quadros, mural, agenda, etc.).

Matriz de Rastreabilidade: Necessidades vs Nível e Valor Agregado

Para cada necessidade do cliente identificada, classificar nos seguintes critérios:

- Nível: Estratégico, tático e operacional;
- Valor Agregado ao Processo: Inovação, Transformação ou Incorporação.

No final da matriz, calcular um resumo com o percentual de cada um dos critérios.

Exemplo: Matriz de Rastreabilidade.xlsx.

IMPORTANTE: É necessário atentar para o fato de que a mera implantação de um sistema informatizado não implica na solução dos problemas identificados.

Um projeto de implantação de um sistema deve ser sempre fortemente alicerçado no equilíbrio de três elementos:

PESSOAS x PROCESSOS x TECNOLOGIA.

Para que as reais expectativas do negócio sejam atendidas, é extremamente importante uma abordagem orientada a processos e serviços.

FASE III - Visão da Solução

A Visão da Solução fornece informações sobre a alternativa proposta para solucionar o problema identificado. Deve-se descrever a solução que o projeto irá fornecer e explicar qual o seu propósito.

Mapeamento da situação desejada (*To Be*)

Mapear os macroprocessos da situação desejada (*to be*). Esta prática permite identificar, de antemão, os processos críticos que demandarão mais tempo na identificação dos requisitos e, consequentemente, realizar um melhor planejamento do trabalho.

DICA: Pode-se dividir os macroprocessos em processos menores visando detalhar o processo.

Visão do Produto (Escopo)

A Visão do Produto, ou Escopo do Projeto, consiste na definição da abrangência do projeto. A Visão do Produto não tem o objetivo de ser uma apresentação detalhada dos requisitos e sim uma apresentação em alto nível de todos os módulos e funcionalidade que vão ser construídos. Pode incluir características de qualidade desejada, metas a serem atingidas com o projeto e o que mais for necessário para alinhar as expectativas das partes interessadas (*stakeholders*).

Visão Geral do Produto

Nesta seção deve ser definido e delimitado, em alto nível, o escopo do software e suas fronteiras. Também devem ser descritas as características principais do software proposto, suas principais funcionalidades e interações com usuários e outros sistemas, além de limitações do negócio.

Cenários de Negócio

Um cenário é uma narrativa breve que descreve uma situação de uso da aplicação, envolvendo usuários, processos e dados reais ou potenciais.

Esta técnica ajuda a identificar e compreender o funcionamento do negócio e do produto proposto. Sendo assim, um cenário visa:

- mostrar quem está utilizando o sistema e qual o resultado esperado;
- mostrar as restrições do usuário: quando e onde está trabalhando, porque está utilizando o sistema e o que necessita dele;
- descrever aspectos relevantes do contexto do usuário com o sistema; e
- indicar o que o usuário considera um sucesso no uso do sistema.

Fluxogramas dos Processos

O fluxograma é um diagrama utilizado para representar a sequência dos processos, através de símbolos gráficos. Os símbolos do fluxograma proporcionam uma melhor visualização do funcionamento do processo, ajudando no seu entendimento.

Características do Sistema

Nesta seção devem ser descritas as características essenciais do software proposto, ou seja, as características necessárias ao completo entendimento do seu conceito e finalidade, especificando-o de maneira inequívoca.

Módulos do Sistema

Módulo é a parte do sistema responsável por uma tarefa bem definida e que pode ser acoplado a um sistema para permitir ao mesmo executar a tarefa disponibilizada pelo módulo.

Um módulo é uma parte do sistema que utiliza a mesma arquitetura tecnológica do sistema, é responsável por atividades que satisfaz um assunto bem definido. As atividades do módulo utilizam tarefas e componentes comuns do sistema. Um módulo ou vários módulos compõem um sistema. Um módulo também é representado por um grupo de componentes de software que atende a um assunto bem definido.

Atores do Sistema

Atores são usuários (ou grupos de usuários) e/ou outros meios externos que desenvolvem algum papel em relação ao sistema. Os meios externos são hardwares e/ou softwares que, assim como os usuários, geram informações para o sistema ou necessitam de informações geradas a partir do sistema.

Existem atores que podem desempenhar mais de um papel no sistema, quando se pensar em atores é sempre bom pensar neles como papéis em vez de pensar como pessoas, cargos, máquinas. No sistema podem existir usuários com diferentes permissões, para isto é necessário criar um ator para cada diferente tipo de permissão. Cada ator deve possuir um nome que deverá ter relação direta com a sua função, possuirá uma descrição que definirá o que ele faz e quais funcionalidades do sistema ele executará.

Levantamento de Requisitos do Sistema

Um Requisito consiste da definição documentada de uma propriedade ou comportamento que um produto ou serviço particular deve atender. São os requisitos que definem os detalhes relativos ao escopo e, portanto, eles devem abranger todo o escopo definido para o software.

Requisitos Funcionais

Um requisito funcional define uma função de um sistema de software ou seu componente. Uma função é descrita como um conjunto de entradas, seu comportamento e as saídas.

Requisitos Não Funcionais

Requisitos não-funcionais são os requisitos relacionados ao uso da aplicação em termos de desempenho, usabilidade, confiabilidade, segurança, disponibilidade, manutenibilidade e tecnologias envolvidas. Em outras palavras, definem restrições e aspectos de qualidade.

Para cada requisito, mapear os critérios de aceite (conjunto das regras de negócio).

DICA: Identificar os diversos cenários e tudo o que tem de atender, não se esquecendo do comportamento dos fluxos alternativos (por exemplo: voltar ao estado anterior do requisito).

Para um maior detalhamento no processo de especificação de requisitos acessar, no Portal de Desenvolvimento:

- Obs.: Para mais informações sobre especificação de requisitos funcionais, acesse Padrões - Requisitos Funcionais.aspx.
- Obs.: Para mais informações sobre especificação de requisitos não funcionais, acesse Padrões - Requisitos Não Funcionais.aspx.

Prototipação de Telas do Sistema

Nesta etapa devem ser desenvolvidos os protótipos de telas das funcionalidades mais relevantes do sistema proposto.

O processo de criação dos protótipos deve seguir as orientações constantes em Padrões - Prototipação.aspx, porém nesta fase de elaboração do Projeto Preliminar os protótipos a serem contruídos devem ser de baixa fidelidade, somente ilustrativa (não sendo um compromisso de disposição de informações e layout), sendo apenas uma forma de comunicação do entendimento, de forma a priorizar o comportamento do sistema e não o layout.

DICA: Uma boa prática é utilizar protótipos do tipo ***wireframe***.

Wireframe é um desenho básico de uma interface, como um esqueleto, que demonstra de forma direta a arquitetura de como o objeto final será de acordo com as especificações relatadas. Este desenho deve ser feito da maneira mais simples possível, mostrando apenas o essencial, como uma espécie de rascunho, sem cores ou imagens. O objetivo do *Wireframe* é auxiliar o desenvolvedor no entendimento dos requisitos funcionais do sistema, assim como de sua usabilidade.

Matriz de Rastreabilidade: Requisitos vs Necessidades

Para cada requisito identificado, mapear quais necessidades são atendidas, de forma a facilitar o entendimento pelo cliente e a identificação de:

- quais necessidades foram atendidas;
- quais requisitos possuem maior relevância.

Exemplo: Matriz de Rastreabilidade.xlsxMatriz de Rastreabilidade.xlsx.

Matriz de Rastreabilidade: Requisitos vs Nível e Valor Agregado

Para cada requisito identificado, classificar nos seguintes critérios:

- Nível: Estratégico, tático e operacional;
- Valor Agregado ao Processo: Inovação, Transformação ou Incorporação.

No final da matriz, calcular um resumo com o percentual de cada um dos critérios.

Exemplo: Matriz de Rastreabilidade.xlsxMatriz de Rastreabilidade.xlsx.

Validação da solução com o patrocinador

Momento em que o patrocinador avaliará o trabalho, verificando se os objetivos, no nível gerencial, estão sendo atendidos.

FASE IV - Outras Definições da Solução

Premissas e Restrições do Projeto

Nesta etapa devem ser definidas as premissas e restrições do projeto.

Premissas são hipóteses, algo que se assume como verdadeiro no início do projeto por não termos informações suficientes. Basicamente, precisamos planejar o projeto e muitas vezes não há como ter todas as informações necessárias. Aqui, utilizamos suposições mais próximas da realidade para que nosso planejamento seja baseado em uma verdade.

Restrições são limitações impostas interna ou externamente ao projeto. Restrições podem ser a obrigatoriedade de se utilizar determinadas ferramentas e a forma de trabalho da equipe.

NOTA: Em geral, premissas geram riscos que devem ser considerados no projeto.

Exclusões do Projeto

Nesta etapa deve ser identificado o que não fará parte do escopo do projeto.

DICA: Declarar explicitamente o que está fora do escopo do projeto ajuda no gerenciamento das expectativas das partes interessadas.

Análise e Dimensionamento de Infraestrutura

Apresentar os requisitos para a equipe da infraestrutura visando à avaliação e definição conjunta das necessidades para atender o pré-projeto.

- Servidores/serviços utilizados (web, Agiles, intranet, serviços da receita, replicação, disponibilidade, certificação/assinatura digital, relógio do tempo)
- Armazenamento (quais documentos tem que ser armazenados (temporário/definitivo), por quanto tempo, quantidade, tamanho médio)
- Acesso: Formas de Acesso (web, móvel, relatório, consultas diretas ao banco), Quantidade de Acessos e Tráfego
- Backup: Tipos de arquivos, Bancos, Servidores, Frequência

A equipe de infra dimensionará o ambiente necessário, possíveis custos adicionais, possíveis riscos e restrições.

Integrações com outros sistemas ou serviços

Apresentar os requisitos para a gerência de desenvolvimento visando à avaliação e identificação das integrações com os demais sistemas ou serviços já existentes.

Mapeamento da existência de sistemas que possuam relação com o produto em questão:

- Sistemas que já tratem do problema (versão em produção);
- Sistemas que possam ser afetados pela solução;
- Identificar outros interessados (unidades de negócio) dentro do Tribunal.

Identificação de riscos

Nesta etapa devem ser identificados os riscos do projeto, ou seja, a ocorrência de eventos que possam comprometer os resultados esperados.

Os riscos de um projeto de software podem ser classificados basicamente em:

- Riscos de projeto - São riscos ligados diretamente ao projeto. Se os riscos de projeto se tornarem reais, o custo e o tempo de projeto podem aumentar drasticamente. Os fatores que estão intimamente ligados a estes riscos são: requisitos, pessoal, recursos, cliente e cronograma. São exemplos deste tipo de risco:
    - Cronograma;
    - Pessoal;
    - Orçamento.
- Riscos técnicos - São riscos relacionados à qualidade do software a ser desenvolvido. Se os riscos técnicos se tornarem reais, a implementação do sistema pode se tornar difícil ou impossível. São exemplos deste tipo de risco:
    - Análise, Design, Implementação, Interface, Testes e Manutenção;
    - Tecnologia - Ferramentas, hardware e software.
- Riscos de negócios - São riscos relacionados à viabilidade do projeto. Se os riscos de negócios se tornarem reais, o projeto pode ser até cancelado. São exemplos deste tipo de risco:
    - Mudança do problema/necessidades;
    - Mudanças no ambiente (interno ou externo);
    - Novas estratégias;
    - Requisitos e restrições organizacionais.

Para a identificação de riscos podem ser utilizadas diversas técnicas, tais como:

- Tabela de Identificação de Riscos - Brainstorm com a equipe técnica da área de negócio e DTI (análise + infra), gerência da área de negócio para identificar os possíveis riscos do projeto.
    - Exemplo:

Item	Tipo	Causa	Riscos	Efeito	Probabilidade	Impacto
1	Técnicos	Falta de habilidade técnica	Retrabalho em tarefas do projeto	Atraso na entrega de tarefas, impacto no cronograma e aumento de custos	Média	Alto
2	Técnicos	Alteração de membro da equipe	Demora na retomada do trabalho após troca do membro	Atraso na entrega de tarefas, impacto no cronograma e aumento de custos	Alta	Alto
3	Técnicos	Erro na Análise do Sistema	Erro na definição do projeto do projeto	Mudança do escopo	Média	Alto
4	Externo	Necessidade ou exigência do usuário	Alteração do escopo inicial em função de uma demanda do negócio	Mudança de escopo e replanejamento do projeto	Média	Alto
5	Externo	Necessidade ou exigência do usuário	Diminuição do tempo de entrega do projeto	Perda da qualidade	Média	Muito Alto
6	Externo	Falta de comprometimento da equipe do usuário	Atrasos nos processos durante as fases de definição do escopo e execução	Atraso na entrega de tarefas e aumento de custos	Alta	Alto
7	Organizacional	Falta de recursos humanos	Equipe não constituída	Atraso na entrega de tarefas e aumento de custos	Baixa	Alto
8	Organizacional	Recursos alocados em outros projetos	Indisponibilidade do colaborador requerido para a atividade	Perda de qualidade, quando a mesma for executada por pessoa menos experiente	Média	Alto
9	Organizacional	Normativa emitida antes do sistema	Prazos apertados	Perda da qualidade, alteração de escopo	Média	Alto

Tipos:
- Técnicos (Requisito, Tecnológico, Desempenho e Confiabilidade, Qualidade, etc...)
- Externos (Usuários, Fornecedores, ...)
- Organizacional (Recursos Humanos, Priorização, Dependência do Projeto, Normativas , ...)

- Matriz de Probabilidade.
    - Exemplo:

Probabilidade	Medição	Descrição
Muito Alto	Maior que 70%	Risco é eminente de ocorrer
Alto	Entre 50% e 70%	Risco é eminente de ocorrer
Médio	Entre 30% e 50%	Provável de ocorrer
Baixo	Entre 0% e 30%	Pequena

- Matriz de Impacto.
    - Exemplo:

Objetivos	Baixo	Médio	Alto	Muito Alto
Custo	Aumento de custo entre 5% e15%	Aumento de custo entre 15% e20%	Aumento de custo entre 20% e 30%	Aumento de custo acima de 30%
Tempo	Aumento do prazo entre 5% e15%	Aumento do prazo entre 15% e20%	Aumento do prazo entre 20% e 30%	Inaceitável pelo usuário
Escopo	Alguns entregáveis impactados, perceptíveis no aceite do projeto	Impacto muito significativo	Impacto muito significativo não aceito	Produto descaracterizado. Entrega do projeto sem Validade
Qualidade	Poucos entregáveis impactados, sem efeito no aceite do projeto	Poucos entregáveis impactados, perceptíveis no aceite do projeto	Impacto muito significativo para o usuário	Inaceitável pelo usuário

FASE V - Conclusão do Projeto Preliminar

Dimensionamento de Recursos

Nesta etapa deve-se estimar os recursos necessários para o desenvolvimento do sistema:

- Estimativa de esforço de desenvolvimento (conforme método de Análise de Pontos por Função);
- Estimativa de recursos de Infraestrutura;
- Outros recursos (por exemplo: acessos ao cadastro da Receita Federal, treinamentos para alguma nova plataforma etc.).

Estimativa de Esforço de Desenvolvimento (Tempo e Custo)

A mensuração do tamanho do software a ser desenvolvido deve ser realizada através do método de Análise de Pontos por Função ou FPA (*Function Point Analysis*) proposto pela NESMA (*Netherlands Software Metrics Users Association*).

Function Point Analysis (FPA)

http://www.nesma.nl/section/fpa/

FPA Counting Practices Manual

http://www.nesma.nl/section/books/NESMA_books.asp#bw_1

Para efeitos de avaliação de viabilidade do Projeto Preliminar, os aspectos essenciais a serem avaliados são prazo (tempo) e orçamento (custo). As estimativas de tempo e de custo de desenvolvimento do software proposto serão realizadas com base no tamanho do software em Pontos por Função.

Desta forma, deve-se utilizar a técnica de Contagem Antecipada de Pontos por Função (*Early FPA Counting*), de autoria da NESMA, para mensurar o tamanho do software proposto.

Contagem Antecipada de Pontos de Função

http://www.fattocs.com.br/traduzido/earlyfpa.asp

Early Function Point Counting

http://www.nesma.nl/section/fpa/earlyfpa.htm

A mensuração do tamanho do software deve ser realizada através da técnica de Contagem Indicativa da NESMA, também conhecida como "método holandês", que consiste basicamente em:

- determinar a quantidade das funções do tipo dado (ALIs e AIEs);
- calcular o total de pontos de função não ajustados da aplicação da seguinte forma:
    - tamanho indicativo (pf) = 35 x número de ALIs + 15 x número de AIEs.

Portanto, esta é uma técnica de estimativa simplificada, baseada somente na quantidade de arquivos lógicos existentes (ALIs e AIEs). A contagem indicativa é baseada na premissa de que existem aproximadamente três EEs (para adicionar, alterar, e excluir dados do ALI), duas SEs, e uma CE na média para cada ALI, e aproximadamente uma SE e uma CE para cada AIE.

NOTA: ALI = Arquivo Lógico Interno

AIE = Arquivo de Interface Externa

EE = Entrada Externa

SE = Saída Externa

CE = Consulta Externa

Recomendações

Ao final do Projeto Preliminar devem ser apresentadas as recomendações sobre o desenvolvimento do software proposto.

Anexos

Os documentos relacionados ao Projeto Preliminar, tais como fluxogramas, protótipos de tela etc., devem ser anexados ou referenciados.

Referências

Devem ser identificadas as origens de informações utilizadas na elaboração do Projeto Preliminar, preservando-se os direitos autorais e facilitando a consulta de informações complementares sobre o assunto.

Glossário

Devem ser descritos, de forma breve e objetiva, os significados dos termos, expressões e palavras utilizados no Projeto Preliminar e que não se encontram no domínio de conhecimento comum.

O Glossário também deve conter explicações de conceitos relevantes de um domínio de conhecimento específico, campo de estudo, ação ou termos técnicos.

Ao final desta fase deverá ser finalizado o seguinte documento:

- Documento do Projeto Preliminar (DPP).

ENTREGA E APROVAÇÃO DO PROJETO PRELIMINAR

Entrega do documento definitivo do pré-projeto ao(s) patrocinador(es) e obtenção da aprovação ou não do documento.

O documento deverá ter as assinaturas dos diretores da DTI e da(s) área(s) de negócio.

IMPORTANTE: O documento definitivo do pré-projeto deverá ter as assinaturas dos diretores da DTI e das áreas de negócio envolvidas (*sponsors*).

Ao final desta etapa deverá ser criado o seguinte documento:

- Termo de Entrega de Pré-Projeto (TEPP).

DOWNLOAD: Documento Modelo de Termo de Entrega de Pré-Projeto: Termo de Entrega de Pré-Projeto - [NomeProjeto].docxTermo de Entrega de Pré-Projeto - [NomeProjeto].docx

CHECKLIST DO PRÉ-PROJETO

Um Documento de Pré-Projeto é considerado concluído quando cumpre todos os seus objetivos propostos. Sendo assim, realize o Checklist do Pré-Projeto para avaliar a sua eficácia.

DOWNLOAD: Checklist do Pré-Projeto: Checklist do Pré-Projeto - [NomeProjeto].docxChecklist do Pré-Projeto - [NomeProjeto].docx

CONTROLE DE VERSÃO/REVISÃO DO PROJETO PRELIMINAR

As versões/revisões do Projeto Preliminar devem ser controladas durante sua elaboração, a fim de manter o histórico de alterações e identificar a versão mais recente do documento.

A identificação da versão/revisão deve constar na capa do documento, conforme exemplo abaixo:

EXEMPLO: VERSÃO/REVISÃO: 01.001 (01 DE AGOSTO DE 2014)

A numeração da versão e da revisão devem ser sequenciais e incrementadas seguindo a seguinte metodologia:

- Versão: As versões do Projeto Preliminar devem ser registradas no Histórico de Versões, a fim de manter um histórico de evolução do projeto, conforme modelo apresentado abaixo. A numeração da versão deve ser incrementada quando ocorrerem mudanças significativas no escopo do projeto a partir de uma versão já finalizada.
- Revisão: Durante a elaboração do Projeto Preliminar, diariamente ou a cada incremento significativo de funcionalidades.

Histórico de Versões

Versão	Data	Responsável	Descrição
01	01/08/2014	Responsável DTI	Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

ARQUIVAMENTO DO PROJETO PRELIMINAR

Os Projetos Preliminares devem ser arquivados no Portal de Pré-Projetos da DTI.

GESTÃO DE PROJETOS

Esta disciplina agrupa as atividades relacionadas ao Gerenciamento de Projetos.

As tarefas definidas nesta disciplina possuem os objetivos de:

- estabelecer e acompanhar a execução de projetos de desenvolvimento de soluções de TI;
- gerenciar o escopo e suas mudanças;
- definir e priorizar a execução dos trabalhos;
- gerenciar recursos e pessoas necessários para a execução do projeto;
- promover a comunicação contínua da equipe visando a maximização da produtividade;
- gerenciar os riscos do projeto;
- avaliar continuamente a execução do projeto, agindo imediatamente quando necessário.

Tarefas:	• Tarefa: Realizar Reunião de Abertura de Projeto • Tarefa: Definir Escopo do Projeto (priorização e estimativa de esforço) • Tarefa: Emitir Ordem de Serviço • Tarefa: Iniciar Iteração (planejamento) • Tarefa: Realizar Reunião Diária • Tarefa: Elaborar Ata de Reunião • Tarefa: Encerrar Iteração (entrega/revisão) • Tarefa: Avaliar Iteração (retrospectiva) • Tarefa: Apurar Indicadores de Qualidade e Gestão • Tarefa: Avaliar Indicadores de Qualidade e Gestão • Tarefa: Receber Serviços • Tarefa: Realizar Reunião de Encerramento de Projeto

Tarefa: Realizar Reunião de Abertura

Entradas:	Pré-Projeto
Saídas:	Termo de Abertura de Projeto
Papéis:	Gerente de Projeto Product Owner Scrum Master Usuário (Área de Negócio)
Disciplinas / Tarefas relacionadas:	Gestão de Projetos → Tarefa: Realizar Reunião de Encerramento de Projeto

A cada início de um projeto é importante reunir todos os participantes para definir vários assuntos referentes ao desenvolvimento e implantação do software. Esta reunião tem por finalidade notificar formalmente todas as partes que o projeto começou e para certificar-se que todos têm um entendimento comum sobre o Pré-Projeto, funções e responsabilidades. Os participantes podem ser todos que tenham interesse no projeto como a equipe de desenvolvimento, usuário, diretores, gestores, etc.

O responsável por conduzir a reunião deverá abordar geralmente os seguintes itens:

- Apresentação da equipe;
- Recapitulação da Proposta ou Pré-Projeto;
- Discussão e esclarecimento sobre as principais entregas, riscos, estimativas, prazos, etc.;
- Discussão das principais funções e responsabilidades envolvidas no projeto;
- Discussão da metodologia de desenvolvimento aplicada;
- Abertura para esclarecimento de dúvidas;
- Definição do prazo final para término do projeto.

Ao final da reunião, caso necessário, pode-se registrar uma Ata de Reunião.

Vale ressaltar que nem todas as reuniões de abertura de projeto são da mesma forma, mas basicamente o contexto é o mesmo.

Tarefa: Definir Escopo do Projeto (priorização e estimativa de esforço)

Entradas:	Pré-Projeto
Saídas:	Backlog do Produto Item de Trabalho do Repositório
Papéis:	Gerente de Projeto Product Owner Scrum Master Usuário (Área de Negócio)
Disciplinas / Tarefas relacionadas:	Gestão de Projetos → Tarefa: Emitir Ordem de Serviço

RESUMO

A definição do escopo do projeto é importante para que o Gerente de Projeto ou Product Owner possa planejar e acompanhar as entregas do projeto.

DEFINIÇÃO DO ESCOPO DO PROJETO

O escopo do projeto deve ser definido antes do início da fase de Desenvolvimento do Software. O ponto de partida para a definição do escopo deve ser o documento de Pré-Projeto, cujo escopo poderá ser ratificado ou alterado, em função das alterações surgidas desde a sua elaboração.

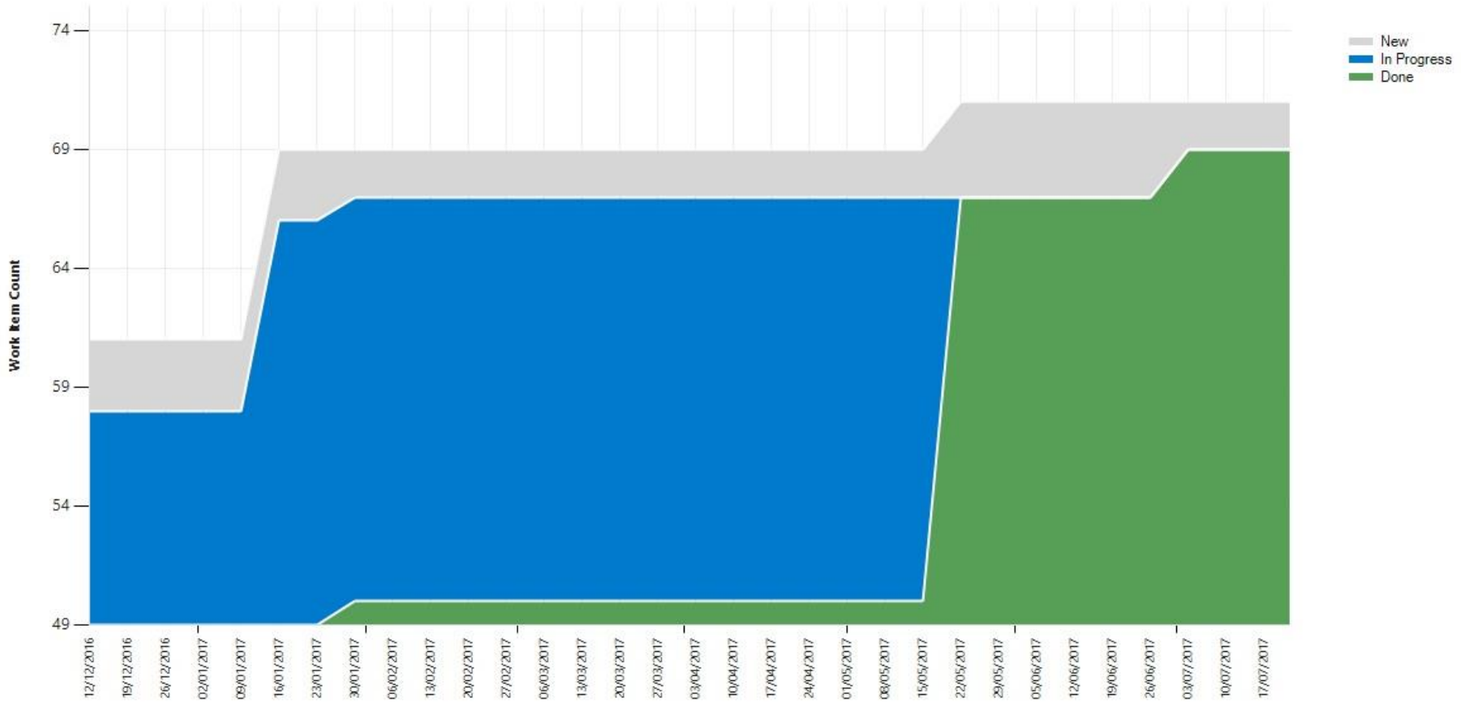
As funcionalidades definidas como escopo do projeto devem ser cadastradas no Repositório como funcionalidades, a fim de que se possa acompanhar o progresso das entregas, assim como eventuais alterações do escopo, através do gráfico do tipo *burnup* (*Product Cumulative Flow*).

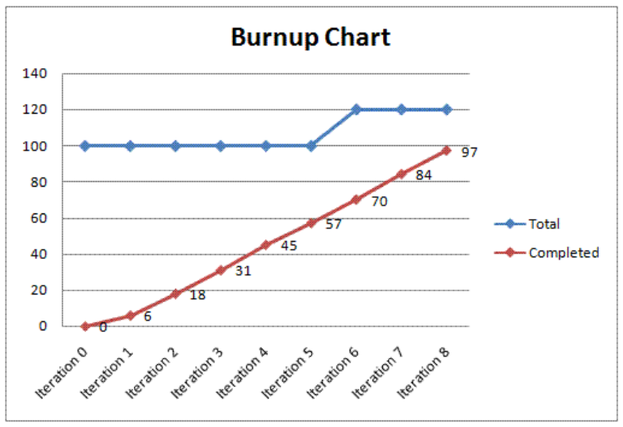
PRIORIZAÇÃO E ESTIMATIVAS DE ESFORÇO

Para um melhor gerenciamento do projeto, é importante que cada item do escopo esteja devidamente priorizado e estimado em seu esforço, auxiliando assim no planejamento das entregas.

Cabe ao Product Owner, com o apoio do Time de Desenvolvimento, estimar o esforço de construção para todos os itens do escopo (Backlog do Produto). Esta estimativa é extremamente importante para o acompanhamento do progresso do projeto (*Product & Release Burnup Charts*) e também para especificar a duração do projeto (em iterações/Sprints).

Cumulative flow





Referências:

- Scrum Guides: http://www.scrumguides.org/
- Scrum Process: https://www.visualstudio.com/en-us/docs/work/guidance/scrum-process
- Cumulative Flow: https://www.visualstudio.com/en-us/docs/report/guidance/cumulative-flow

Tarefa: Emitir Ordem de Serviço

Entradas:	Termo de Abertura de Projeto Solicitação de Serviço
Saídas:	Ordem de Serviço
Papéis:	Gerente de Projeto Gerente de Sustentação
Disciplinas / Tarefas relacionadas:	Gestão de Projetos → Tarefa: Receber Serviços

RESUMO

A Ordem de Serviço é o instrumento utilizado para solicitação de equipe técnica terceirizada para o desenvolvimento de um novo projeto de software ou para sustentação e manutenção em aplicação existentes, quando houver contrato de serviços de desenvolvimento, sustentação e manutenção de sistemas vigente.

ORDEM DE SERVIÇO

A Ordem de Serviço deve ser emitida sempre que houver necessidade de alocação de equipe técnica terceirizada, respeitando-se os limites do contrato vigente.

Neste documento, devem ser preenchidas as seguintes informações:

- tipo de serviço técnico: desenvolvimento de sistemas ou sustentação e manutenção de sistemas;
- projeto/serviço: identificação do projeto ou serviço a ser executado;
- descrição do projeto/serviço; descrição do projeto ou serviço a ser executado;
- data de início do projeto/serviço;
- data prevista de término do projeto/serviço;
- equipe técnica solicitada: quantidade de profissionais dos perfis Analista Programador nível pleno e Analista Programador nível sênior.

ADITIVO DE ORDEM DE SERVIÇO

O Aditivo de Ordem de Serviço deve ser emitido, mediante justificativa, sempre que houver a necessidade de:

- aumento de equipe;
- redução de equipe;
- substituição de membro da equipe;
- alteração de data prevista de término do projeto/serviço.

Tarefa: Iniciar Iteração (planejamento)

Entradas:	Pré-Projeto Backlog do Produto
Saídas:	Backlog da Iteração
Papéis:	Gerente de Projeto Product Owner Scrum Master Time de Desenvolvimento Usuário (Área de Negócio)
Disciplinas / Tarefas relacionadas:	Gestão de Projetos → Tarefa: Realizar Reunião Diária Gestão de Projetos → Tarefa: Encerrar Iteração (entrega/revisão) Gestão de Projetos → Tarefa: Avaliar Iteração (retrospectiva)

O objetivo desta tarefa é o planejamento das atividades da iteração ou Sprint. A cada inicio de iteração o Time de Desenvolvimento deve realizar uma reunião, juntamente com o Product Owner, e facilitada pelo Scrum Master para realizar o planejamento.

Nesta reunião de planejamento é definido o que vai ser entregue na Sprint e como a equipe realizará o trabalho para conseguir finalizar o que foi planejado. O Product Owner apresenta o Backlog do Produto ordenado e explica para a equipe o que deverá ser feito em cada item do Backlog. Após essa explicação, a equipe define quais os artefatos serão produzidos, cria e estima as tarefas necessárias para a construção de cada um, respeitando a todos os padrões estabelecidos no PDS TCEPR e a definição de pronto[1].

Sabendo o que é prioritário para o Product Owner e conhecendo o esforço para concluir cada um dos itens priorizados, a equipe e o Product Owner devem chegar a um acordo sobre o que vai ser feito na Sprint, surgindo assim o Backlog da Iteração.

[1] **Definição de Pronto**: é um acordo entre o Product Owner e Time de Desenvolvimento sobre o que é necessário para que um item ou o incremento do produto como um todo seja considerado pronto. A definição de pronto é a mesma para todos os itens do Product Backlog que representem funcionalidades a serem desenvolvidas.

Tarefa: Realizar Reunião Diária

Entradas:	Backlog da Iteração
Saídas:	-
Papéis:	Gerente de Projeto Product Owner Scrum Master Time de Desenvolvimento
Disciplinas / Tarefas relacionadas:	Gestão de Projetos → Tarefa: Iniciar Iteração (planejamento) Gestão de Projetos → Tarefa: Encerrar Iteração (entrega/revisão) Gestão de Projetos → Tarefa: Avaliar Iteração (retrospectiva)

Durante uma iteração, a equipe deverá fazer reuniões diárias. Estas reuniões devem ser rápidas, no máximo 15 mínutos, e têm o objetivo de discutir o que cada um da equipe fez no dia anterior, o que está fazendo e se existem impedimentos ou riscos identificados.

A reunião diária é importante para corrigir os rumos e mitigar os riscos, além de melhorar a comunicação e o engajamento da equipe. Ainda, proporciona a inspeção (do progresso), a adaptação (ajustes e impedimentos) diariamente e a transparência (todos sabem o que está acontecendo).

COMO FUNCIONA

- Todos os membros da equipe devem participar. O Scrum Master organiza, mas o time é quem conduz.
- Não pode passar de 15 minutos, preferencialmente.
- Mesmo local e horário todos os dias.
- Responder as 3 perguntas:
    - O que eu fiz ontem que ajudou o time a atender a meta da sprint?
    - O que eu farei hoje para ajudar o time a atender a meta da sprint?
    - Tem algum obstáculo ou impedimento que impeça a mim ou ao time de atender a meta da sprint?
- O Scrum Master termina a reunião e atualiza os registros de impedimentos que foram levantados afim de resolvê-los.

O QUE NÃO FAZER

- A reunião não deve ser usada como uma reunião para resolução de problemas. Caso seja necessário discutir algo mais detalhado, deve-se esperar terminar a reunião diária ou marcar uma reunião específica para o assunto.
- Deixar de fazer a reunião em razão da ausência de algum integrante.
- Não ultrapassar os 15 minutos.
- Alternância de horários.

Tarefa: Elaborar Ata de Reunião

Entradas:	-
Saídas:	Ata de Reunião
Papéis:	Todos os Papéis
Disciplinas / Tarefas relacionadas:	

RESUMO

Esta página tem por objetivo orientar o processo de registro de Relatórios de Reunião (ou Atas de Reunião) durante o Processo de Desenvolvimento de Software no Tribunal de Contas do Estado do Paraná (TCE-PR).

RELATÓRIO DE REUNIÃO

Todos os assuntos tratados em uma reunião, devem ser colocados em uma Ata de Reunião. Desta forma os assuntos abordados durante este evento, não serão esquecidos, e poderão ser retomados, se necessário.

No Processo de Desenvolvimento de Software do TCEPR deverá ser utilizado o modelo apresentado em PRODUTOS → Ata de Reunião.

Este modelo contém as seguintes informações:

- Número/Ano da Ata de Reunião
  - A numeração das atas deve ser sequencial por ano (reiniciam a cada ano).
- Informações
  - Projeto: Identificação do projeto a que se refere a reunião.
  - Data da Reunião.
  - Horário (Início/Término).
- Participantes da Reunião
  - Informar os participantes da reunião (internos ou externos):
    - caso seja um participante do Tribunal (interno), no último campo da lista digitar o nome e realizar a busca no diretório de usuários;
    - caso seja um participante externo, informar os campos nome, unidade (ou organização) e e-mail.
  - Informar se o participante estava presente ou não na reunião. (Obs.: Recomenda-se informar todos os participantes do grupo, mesmo que não estivessem presentes na reunião, para que recebam uma cópia da ata de reunião por e-mail e possam estar cientes das decisões.)
- Pauta
  - Descrição da pauta (assuntos) da reunião.
- Assuntos Pendentes da Reunião Anterior
  - Os assuntos pendentes da reunião anterior, caso existam, devem ser discutidos no início da reunião e registrados em ata, juntamente com a informação do responsável, prazo e situação atual.
- Tópicos Discutidos
  - Descrever os assuntos discutidos durante a reunião.
- Ações a serem tomadas

    - Relacionar as ações definidas na reunião (Plano de Ação).
- Observações
    - Campo para observações (opcional) da reunião.
- Responsável, data e hora do preenchimento da Ata de Reunião
    - Estes campos são preenchidos automaticamente pelo formulário Microsoft InfoPath.
- Nº de Revisão da Ata de Reunião
    - Este campo é preenchido automaticamente pelo formulário Microsoft InfoPath.

Após o preenchimento das informações, a Ata de Reunião deve ser salva no portal do projeto e uma cópia da Ata de Reunião deve ser enviada por e-mail a todos os participantes.

NOTA: A redação da ata de reunião deve ser realizada de forma objetiva e impessoal, descrevendo de forma clara e sucinta as decisões e as ações a serem realizadas.

DICAS PARA UMA REUNIÃO EFICAZ E PRODUTIVA:

- Identifique o objetivo da reunião (destaque no convite para a reunião e reforce no início da reunião);
- Prepare a reunião adequadamente (defina a pauta da reunião e encaminhe-a aos participantes com antecedência);
- Defina o tempo da reunião (não realize reuniões muito extensas, recomenda-se que a duração máxima seja de 1 hora);
- Convoque as pessoas certas;
- Controle o tempo da reunião (comece e termine a reunião pontualmente nos horários marcados);
- Mantenha a reunião sob controle (evite dispersão entre os participantes com conversas paralelas e assuntos);
- Seja objetivo e decisivo (direcione as discussões para uma definição);
- A ata de reunião deve ser um plano de ação;
- Registre e distribua a ata da reunião o mais rápido possível após a reunião (proporciona o reconhecimento das decisões pelos membros da equipe, fazendo com que as deliberações da reunião comecem a surtir efeito).

ARQUIVAMENTO DAS ATAS DE REUNIÃO

As Atas de Reunião devem ser arquivadas no portal do projeto na pasta Atas de Reunião da biblioteca Gerenciamento de Projetos.

Tarefa: Encerrar Iteração (entrega/revisão)

Entradas:	Backlog da Iteração Incremento de Software
Saídas:	-
Papéis:	Gerente de Projeto Product Owner Scrum Master Time de Desenvolvimento Usuário (Área de Negócio)
Disciplinas / Tarefas relacionadas:	Gestão de Projetos → Tarefa: Iniciar Iteração (planejamento) Gestão de Projetos → Tarefa: Realizar Reunião Diária Gestão de Projetos → Tarefa: Avaliar Iteração (retrospectiva) Gestão de Projetos → Tarefa: Apurar Indicadores de Qualidade e Gestão Gestão de Projetos → Tarefa: Avaliar Indicadores de Qualidade e Gestão

O objetivo desta reunião é apresentar ao Usuário (Área de Negócio) tudo o que foi produzido na sprint de acordo com o conceito de pronto[1] planejado na reunião de início de iteração. Então o Time de Desenvolvimento e o Product Owner apresentam e demonstram para os presentes o que foi produzido para então deles obter o feedback.

O feedback obtido pelos presentes na reunião é utilizado como matéria-prima para alterações no Product Backlog, ou seja, para modificar o produto que está sendo gerado de forma a melhor atender às necessidades do cliente.

[1] **Definição de Pronto**: é um acordo entre o Product Owner e Time de Desenvolvimento sobre o que é necessário para que um item ou o incremento do produto como um todo seja considerado pronto. A definição de pronto é a mesma para todos os itens do Product Backlog que representem funcionalidades a serem desenvolvidas.

Tarefa: Avaliar Iteração (retrospectiva)

Entradas:	Backlog da Iteração
Saídas:	Documento de Retrospectiva
Papéis:	Gerente de Projeto Product Owner Scrum Master Time de Desenvolvimento
Disciplinas / Tarefas relacionadas:	Gestão de Projetos → Tarefa: Iniciar Iteração (planejamento) Gestão de Projetos → Tarefa: Realizar Reunião Diária Gestão de Projetos → Tarefa: Encerrar Iteração (entrega/revisão) Gestão de Projetos → Tarefa: Apurar Indicadores de Qualidade e Gestão Gestão de Projetos → Tarefa: Avaliar Indicadores de Qualidade e Gestão

O objetivo desta reunião é identificar pontos de melhoria no processo de desenvolvimento, o que foi bem e o que pode melhorar, buscando formas práticas e traçando planos de ação para fazê-lo, tornando o time como um todo cada vez mais efetivo.

Tarefa: Apurar Indicadores de Qualidade e Gestão

Entradas:	Backlog da Iteração Código-fonte Incremento de Software Termo de Recebimento de Serviço Item de Trabalho do Repositório
Saídas:	Indicadores de Qualidade e Gestão
Papéis:	Gerente de Projeto Product Owner
Disciplinas / Tarefas relacionadas:	Gestão de Projetos → Tarefa: Avaliar Indicadores de Qualidade e Gestão

RESUMO

Após o encerramento de cada iteração deve ser realizada a apuração dos Indicadores de Qualidade e Gestão, a fim de acompanhar a execução do projeto.

APURAÇÃO DOS INDICADORES DE QUALIDADE E GESTÃO

A apuração e o acompanhamento dos Indicadores de Qualidade e Gestão devem ser realizados durante todo o projeto.

Os Indicadores de Qualidade e Gestão são apurados utilizando como métrica de referência a Análise por Pontos de Função (APF). Para efeitos de apuração dos indicadores, são utilizados 2 tipos de medição, conforme definido no Manual de Contagem de Pontos de Função do TCEPR:

- Contagem Indicativa, na fase de elaboração do Pré-Projeto ou no início do Projeto; e
- Contagem Estimativa, ao final de cada iteração e no final do Projeto.

Na elaboração do Pré-Projeto ou no início do Projeto deve-se:

- Estimar o tamanho do software (em PF).

Ao final de cada Iteração e no final do Projeto deve-se:

- Identificar as atividades em:
  - implementação de funcionalidade;
  - alteração de funcionalidade;
  - correção de defeitos (bugs).
- Estimar o esforço (em PF) das atividades de:

    - implementação de funcionalidade;
    - alteração de funcionalidade.
- Mensurar a quantidade (em Unidades) de defeitos (bugs) corrigidos.
- Mensurar o custo (em Horas) das atividades de:
    - implementação de funcionalidade;
    - alteração de funcionalidade;
    - correção de defeitos (bugs).
- Calcular o tamanho do software em produção (em PF).
- Calcular os Indicadores de Qualidade e Gestão:
    - IAEP - Índice de Aderência do Escopo do Projeto;
    - IQPDS - Índice de Qualidade do Processo de Desenvolvimento de Software;
    - IRPDS - Índice de Retrabalho do Processo de Desenvolvimento de Software;
    - CDSW - Custo de Desenvolvimento de Software;
    - ISC - Índice de Satisfação do Cliente;
    - IQESW - Índice de Qualidade da Estimativa de Software;
    - ICMI – Índice de Custo de Medição dos Indicadores.
- Elaborar gráfico de evolução dos indicadores.

As definições dos Indicadores de Qualidade e Gestão, e suas respectivas fórmulas de cálculo, são descritos no capítulo 4. PRODUTOS em Indicadores de Qualidade e Gestão.

Tarefa: Avaliar Indicadores de Qualidade e Gestão

Entradas:	Indicadores de Qualidade e Gestão
Saídas:	Indicadores de Qualidade e Gestão
Papéis:	Gerente de Projeto Product Owner Scrum Master
Disciplinas / Tarefas relacionadas:	Gestão de Projetos → Tarefa: Apurar Indicadores de Qualidade e Gestão

RESUMO

Após a apuração dos Indicadores de Qualidade e Gestão deve ser realizada a avaliação dos indicadores, a fim identificar e corrigir possíveis desvios e buscar o aperfeiçoamento contínuo dos processos.

AVALIAÇÃO DOS INDICADORES DE QUALIDADE E GESTÃO

Os indicadores devem ser apurados e avaliados periodicamente, e sempre que for identificado um indício de desvio quanto aos objetivos do projeto devem ser tomadas ações visando a correção dos rumos do projeto ou a alteração dos objetivos do projeto (principalmente quanto ao escopo, cronograma e custo do projeto), se necessário, provocando o Comitê de TI.

Os Indicadores de Qualidade e Gestão do projeto podem ser comparados com os demais projetos, buscando replicar as boas práticas e aperfeiçoar os processos.

Tarefa: Receber Serviços

Entradas:	Ordem de Serviço
Saídas:	Termo de Recebimento de Serviço
Papéis:	Gerente de Projeto Gerente de Sustentação
Disciplinas / Tarefas relacionadas:	Gestão de Projetos → Tarefa: Emitir Ordem de Serviço

RESUMO

O recebimento de serviços ocorre a cada entrega de produtos de software no âmbito de cada Ordem de Serviço. É através desta atividade que será realizado o cálculo da remuneração devida pelos serviços prestados, considerando o cumprimento dos níveis mínimos de serviço definidos em instrumento contratual.

TERMO DE RECEBIMENTO DE SERVIÇO

O Termo de Recebimento de Serviço deve ser emitido pela empresa terceirizada para o cálculo da remuneração devida pelos serviços prestados no âmbito de cada Ordem de Serviço.

Este documento deve ser preenchido conforme modelo definido em instrumento contratual.

O recebimento dos serviços deve ser realizado pelo TCEPR e é indispensável para a realização do pagamento dos serviços prestados.

Tarefa: Realizar Reunião de Encerramento de Projeto

Entradas:	-
Saídas:	Termo de Encerramento de Projeto
Papéis:	Gerente de Projeto Product Owner Scrum Master Usuário (Área de Negócio)
Disciplinas / Tarefas relacionadas:	Gestão de Projetos → Tarefa: Realizar Reunião de Abertura de Projeto

O objetivo da reunião de encerramento de projeto é formalizar o término do projeto.

O Gerente do Projeto deve convocar a reunião final com as partes interessadas necessárias para validar o escopo final do projeto, comparando o produto entregue com os requisitos e especificações do pré-projeto e assim determinar se todo o objetivo do negócio foram alcançados. Com isso devemos obter uma aprovação formal do usuário indicando que o projeto está concluído e que a manutenção será repassada para a equipe de sustentação.

REQUISITOS

Esta disciplina agrupa as atividades relacionadas a análise e especificação de Requisitos.

As tarefas definidas nesta disciplina possuem os objetivos de:

- entender os problemas ou as necessidades das áreas de negócio;
- identificar e especificar os requisitos das soluções de TI;
- definir os limites (escopo) das soluções de TI;
- identificar premissas e restrições técnicas das soluções de TI;
- gerar artefatos que auxiliem os clientes na compreensão da solução de TI a ser construída;
- traduzir as necessidades dos clientes em artefatos que facilitem o entendimento dos desenvolvedores da solução.

Tarefas:	• Tarefa: Identificar e Especificar Requisitos • Tarefa: Definir Critérios de Aceite • Tarefa: Especificar Caso de Uso • Tarefa: Especificar História de Usuário • Tarefa: Especificar Change Request • Tarefa: Desenhar Protótipo • Tarefa: Elaborar Diagrama de Negócio

Tarefa: Identificar e Especificar Requisitos

Entradas:	Pré-Projeto Protótipo Diagrama de Negócio
Saídas:	Caso de Uso Change Request História de Usuário
Papéis:	Analista de Demandas Gerente de Projeto Product Owner Time de Desenvolvimento
Disciplinas / Tarefas relacionadas:	Gestão de Demandas → Tarefa: Elaborar Proposta de Projeto Gestão de Demandas → Tarefa: Elaborar Pré-Projeto Requisitos → Tarefa: Definir Critérios de Aceite Requisitos → Tarefa: Especificar Caso de Uso Requisitos → Tarefa: Especificar História de Usuário Requisitos → Tarefa: Especificar Change Request Requisitos → Tarefa: Desenhar Protótipo Requisitos → Tarefa: Elaborar Diagrama de Negócio

RESUMO

Esta página tem por objetivo orientar o processo de Especificação de Requisitos no processo de Desenvolvimento de Software no Tribunal de Contas do Estado do Paraná (TCE-PR).

ENGENHARIA DE REQUISITOS

A Engenharia de Requisitos é um processo que engloba todas as atividades que contribuem para a produção de um documento de requisitos e sua manutenção ao longo do tempo.

O processo de Engenharia de Requisitos é composto por quatro atividades de alto nível:

- Identificação;
- Análise e Negociação;
- Especificação e Documentação;
- Validação.

Identificação de Requisitos

A identificação de requisitos envolve a compreensão do domínio, a identificação das partes interessadas, a captura dos requisitos pretendidos para o sistema pelo cliente e a identificação e análise de problemas.

Na etapa de identificação dos requisitos podem ser utilizadas diversas técnicas de levantamento de requisitos, conforme abordadas abaixo neste documento.

Análise e Negociação de Requisitos

Após a identificação dos requisitos do sistema, deve ser realizada a análise e negociação dos requisitos.

Os requisitos devem organizados, agrupados e avaliados. A análise dos requisitos deve levar em consideração a sua completude, complexidade, priorização e geração de valor.

Especificação e Documentação de Requisitos

É nesta fase que se dá a produção propriamente dita da Especificação de Requisitos. Este processo considera 2 tipos de requisitos:

- Requisitos funcionais: descrevem as funcionalidades que se espera que o sistema disponibilize, de uma forma completa e consistente. É aquilo que o utilizador espera que o sistema ofereça, atendendo aos propósitos para qual o sistema será desenvolvido.
- Requisitos não-funcionais: referem-se a aspectos não-funcionais do sistema, como restrições nas quais o sistema deve operar ou propriedades emergentes do sistema.

Validação de Requisitos

Nesta etapa deve-se verificar se o documento de requisitos produzido corresponde, de fato, ao sistema que o cliente pretende.

Os atributos a serem verificados são:

- Validade: a especificação resulta da análise dos requisitos identificados junto das diversas partes interessadas envolvidas. Como tal, requisitos identificados individualmente (isto é, junto de cada parte interessada) podem diferir da especificação final que se atinge após o cruzamento de informação e é necessário que cada cliente compreenda e aceite a especificação final obtida.
- Consistência: não devem existir conflitos entre os requisitos identificados.
- Compreensibilidade / Ambiguidade: os requisitos devem ser compreendidos de forma inequívoca pelas partes interessadas.
- Completude: todas as funcionalidades pretendidas devem fazer parte da especificação do sistema.
- Factível: dadas as restrições do projeto (tecnológicas, financeiras e temporais) o sistema especificado tem de ser implementável.
- Verificabilidade: de forma a evitar futuras discordâncias quanto à concretização dos requisitos especificados, estes devem ser descritos de modo que seja possível verificar se foram ou não concretizados, isto é, se o sistema final corresponde à especificação inicial.
- Rastreabilidade: a origem dos requisitos, em relação ao cliente, deve estar claramente identificada. Entre outros motivos, isto é importante para facilitar a gestão futura dos requisitos.
- Conformidade com normas: para além dos aspectos funcionais dos requisitos, a sua especificação deve obedecer às normas usadas ao longo de todo o documento.

TÉCNICAS DE LEVANTAMENTO DE REQUISITOS

Existem diversas técnicas de levantamento de requisitos, e que são adequadas a diferentes situações, entre as quais podemos citar:

- Entrevistas e Questionários;
- Brainstorming;
- Workshops de Requisitos;
- Cenários (Série de Eventos Hipotéticos);
- Prototipagem.

REQUISITOS FUNCIONAIS E NÃO FUNCIONAIS

Um Requisito consiste na definição documentada de uma propriedade ou comportamento que um produto ou serviço particular deve atender. São os requisitos que definem os detalhes relativos ao escopo e, portanto, eles devem abranger todo o escopo definido para o software.

Os requisitos podem ser do tipo:

- Funcionais;
- Não Funcionais.

REQUISITOS FUNCIONAIS

Definição

Um requisito funcional define uma função de um sistema de software ou seu componente, onde uma função deve ser descrita como um conjunto de entradas, seu comportamento e as saídas.

Modelo de Especificação de Requisito Funcional

A especificação de um requisito funcional deve utilizar o modelo apresentado abaixo, onde:

- Código:
- Nome:
- Atores:
- Descrição:
- Regras
- Observações:
- Requisitos correlacionados:

RF001 - Requisito Funcional 001

Código:	RF001
Nome:	Requisito Funcional 001
Atores:	Todos
Descrição: Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.	
Regras: R1.1) O sistema deve... R1.2) O sistema deve...	
Observações: 1) O sistema...	
Requisitos correlacionados: RF002, RNF001	

REQUISITOS NÃO FUNCIONAIS

Definição

Requisitos não funcionais são os requisitos relacionados ao uso da aplicação em termos de desempenho, usabilidade, confiabilidade, segurança, disponibilidade, manutenibilidade e tecnologias envolvidas. Em outras palavras, definem restrições e aspectos de qualidade.

Requisitos não funcionais:

- surgem conforme a necessidade dos usuários, em razão de orçamento, espaço de armazenamento disponível e outros fatores (restrições);
- podem estar relacionados à confiabilidade, tempo de resposta e outros aspectos (qualidade).
- demonstram qualidade acerca dos serviços ou funções a serem disponibilizadas pelo sistema;
- caso ocorra o não atendimento a um requisito não funcional, todo o sistema poderá tornar-se ineficaz.

Outros tipos de requisitos não funcionais:

- Requisitos de Negócio (opcional): Requisitos de Negócio descrevem em termos do negócio o que deve ser entregue ou conseguido para fornecer valor.
- Requisitos Legais, Estatutários e Regulamentares (opcional): São requisitos impostos por lei, estatutos, normas e regulamentos, de âmbito geral ou específico/interno.
- Requisitos Tecnológicos (opcional): Referem-se a arquitetura tecnológica, padrões, comunicação, ferramentas, linguagens, etc..
- Requisitos de Qualidade (opcional): Referem-se a requisitos de qualidade do software proposto.
- Outros Requisitos (opcional): Referem-se a outros tipos de requisitos, específicos e/ou especiais.

Modelo de Especificação de Requisito Não Funcional

A especificação de um requisito não funcional deve utilizar o modelo apresentado abaixo, onde:

- Código:
- Nome:
- Restrição:
- Categoria:
- Obrigatório:
- Permanente:

RNF001 - Requisito Não Funcional 001

Código:	RNF001	
Nome:	Requisito Não Funcional 001	
Restrição: Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.		
Categoria:	Categoria	
Obrigatório (X)		**Permanente (X)**

REFERÊNCIAS

MORAES, Janaína Bedani Dixon. Técnicas para levantamento de Requisitos. Disponível em http://www.devmedia.com.br/engenharia-de-software-2-tecnicas-para-levantamento-de-requisitos/9151#ixzz39Rmt1jac.

Tarefa: Especificar Caso de Uso

Entradas:	Pré-Projeto
Saídas:	Caso de Uso Regras de Negócio
Papéis:	Time de Desenvolvimento
Disciplinas / Tarefas relacionadas:	Requisitos → Tarefa: Identificar e Especificar Requisitos Requisitos → Tarefa: Definir Critérios de Aceite Requisitos → Tarefa: Especificar História de Usuário Requisitos → Tarefa: Especificar Change Request Requisitos → Tarefa: Desenhar Protótipo Requisitos → Tarefa: Elaborar Diagrama de Negócio

RESUMO

Um Caso de Uso é um documento narrativo que descreve a sequência de eventos de um ator que usa um sistema para completar um processo, fornecendo um **Resultado de Valor**. Conforme nossas experiências, resolvemos narrar nossos documentos baseados em tela (interface), facilitando assim o desenvolvimento e a manutenção.

O desenvolvimento de Caso de Uso pode requerer diversos passos dependendo do tipo de desenvolvimento (manutenção, funcionalidade batch, relatórios ou novo sistema).

PAPÉIS E RESPONSABILIDADES

O Analista Programador é responsável pela produção, qualidade e atualização do caso de uso. Todas as atividades necessárias para a implementação do caso de uso devem ser definidas em conjunto com o gerente de projeto e o seu registro feito pelo Analista Programador.

ORIENTAÇÕES PARA ELABORAÇÃO DO CASO DE USO

A seguir, estabelecemos um resumo do passo-a-passo sugerido para escrever um documento de Caso de Uso:

1. Identificação de Caso de Uso
    1. Identificar os atores (ou interessados) do sistema
    2. Identificar as funcionalidades para cada ator
    3. Identificar as telas e navegação entre as telas
    4. Identificar e nomear os casos de uso por tela, seguindo uma sequência cronológica
2. Prototipar a tela
3. Escrever o Caso de Uso orientado a tela
4. Revisar e Finalizar o Caso de Uso

5. Homologação com o Usuário

DOWNLOAD:

Manual - Criação de Caso de Uso v2.docx

Tarefa: Definir Critérios de Aceite

Entradas:	Caso de Uso Change Request Regras de Negócio História de Usuário Protótipo Diagrama de Negócio
Saídas:	Critérios de Aceite
Papéis:	Gerente de Projeto Product Owner Time de Desenvolvimento Usuário (Área de Negócio)
Disciplinas / Tarefas relacionadas:	Requisitos → Tarefa: Identificar e Especificar Requisitos Requisitos → Tarefa: Especificar Caso de Uso Requisitos → Tarefa: Especificar História de Usuário Requisitos → Tarefa: Especificar Change Request Requisitos → Tarefa: Desenhar Protótipo Requisitos → Tarefa: Elaborar Diagrama de Negócio

RESUMO

O Objetivo desta tarefa é agregar informações que servirão de base para a criação de casos de testes. Tanto para automatização quanto para testes manuais.

CRITÉRIOS DE ACEITE

Tarefa a ser realizada pelo Product Owner em conjunto com o Analista Programador e o Usuário (Área de Negócio). Nesta etapa, o Product Owner deverá promover reuniões (quantas forem necessárias) para esclarecer as regras de negócio aplicadas ao módulo a ser desenvolvido durante a iteração.

O foco dessas reuniões é esclarecer dúvidas e identificar casos de teste para automatização de testes, e testes de validação que serão realizados pelos usuários da área de negócio. Sempre deverão ser registrados os critérios de aceite do "caminho-feliz" de um processamento do sistema e os possíveis caminhos de falha. O seja, deve ficar claro para a time de desenvolvimento como o sistema a ser desenvolvido deve funcionar, nas situações de "caminho-feliz" e possíveis falhas, ou seja, devem ser imaginadas todas as respostas esperadas do sistema (mensagens, bloqueios, *rollbacks*, etc).

Sugere-se que os critérios de aceite sejam registrados em linguagem Gherkin para facilitar a automatização dos mesmos. Todos esses registros devem constar dentro do documento de "Critérios de Aceite".

Exemplo de Gherkin:

Scenario: <description of the test>

Given <a known state>
When <an event occurs>
Then <then this should happen>

Tarefa: Especificar História de Usuário

Entradas:	Pré-Projeto
Saídas:	História de Usuário
Papéis:	Time de Desenvolvimento
Disciplinas / Tarefas relacionadas:	Requisitos → Tarefa: Identificar e Especificar Requisitos Requisitos → Tarefa: Definir Critérios de Aceite Requisitos → Tarefa: Especificar Caso de Uso Requisitos → Tarefa: Especificar Change Request Requisitos → Tarefa: Desenhar Protótipo Requisitos → Tarefa: Elaborar Diagrama de Negócio

RESUMO

A História de Usuário é uma forma de descrever os requisitos funcionais e relacioná-los com seu principal ator.

HISTÓRIA DE USUÁRIO

Ao desenvolver uma história de usuário deve-se considerar a perspectiva de quem é o ator principal na história, qual ação ele pretende executar e qual o resultado pretendido ao executar tal ação. Em geral, as histórias de usuário devem ser sucintas e devem tratar uma única funcionalidade de forma clara.

Exemplo de história de usuário

Cenário Usuário que deseja listar as atividades em sua caixa de atividade

Como Usuário do painel de atividades
Quero Listar todas as atividades que estão sob minha responsabilidade
Para ter acesso ao link para a execução das atividades

Tarefa: Especificar Change Request

Entradas:	Caso de Uso Regras de Negócio
Saídas:	Change Request Regras de Negócio
Papéis:	Time de Desenvolvimento
Disciplinas / Tarefas relacionadas:	Requisitos → Tarefa: Identificar e Especificar Requisitos Requisitos → Tarefa: Definir Critérios de Aceite Requisitos → Tarefa: Especificar Caso de Uso Requisitos → Tarefa: Especificar História de Usuário Requisitos → Tarefa: Desenhar Protótipo Requisitos → Tarefa: Elaborar Diagrama de Negócio

Quando ocorrerem alterações de escopo após um sistema estar em produção, o analista deverá identificar o impacto desta alteração no sistema.

Quando houver alterações de grande impacto que invalidem a maior parte do que foi escrito no requisito de sua construção, este deverá ser reescrito.

Caso a alteração seja de pequeno impacto, o analista deverá especificar as alterações em um documento chamado Change Request. Para este caso, não há necessidade de homologação formal.

Tarefa: Desenhar Protótipo

Entradas:	Pré-Projeto Backlog do Produto Backlog da Iteração Caso de Uso Change Request História de Usuário
Saídas:	Protótipo
Papéis:	Analista de Demandas Gerente de Projeto Product Owner Time de Desenvolvimento
Disciplinas / Tarefas relacionadas:	Gestão de Demandas → Tarefa: Elaborar Proposta de Projeto Gestão de Demandas → Tarefa: Elaborar Pré-Projeto Requisitos → Tarefa: Identificar e Especificar Requisitos Requisitos → Tarefa: Definir Critérios de Aceite Requisitos → Tarefa: Especificar Caso de Uso Requisitos → Tarefa: Especificar História de Usuário Requisitos → Tarefa: Especificar Change Request Requisitos → Tarefa: Elaborar Diagrama de Negócio

RESUMO

Protótipos auxiliam o levantamento e a validação de requisitos. Um protótipo pode ser acompanhado de dois outros elementos: a descrição dos campos e o mapa de navegabilidade (opcional).

OBJETIVOS

A prototipação tem dois objetivos principais: levantar e validar requisitos.

Ao longo desta atividade, o analista tem a oportunidade de descobrir o que o usuário necessita e de experimentar opções de design de telas, usabilidade e implementação.

O produto final da atividade de prototipação apresenta elementos de interface, capazes de ilustrar requisitos e funcionalidades do sistema a ser desenvolvido.

ELEMENTOS

São elementos de um protótipo o desenho, a descrição dos campos e o mapa de navegabilidade.

Admite-se que o desenho possa apresentar alta ou baixa fidelidade. Vide os exemplos das figuras 1 e 1.2 (respectivamente).

Figura 1 - Elementos do protótipo - O desenho

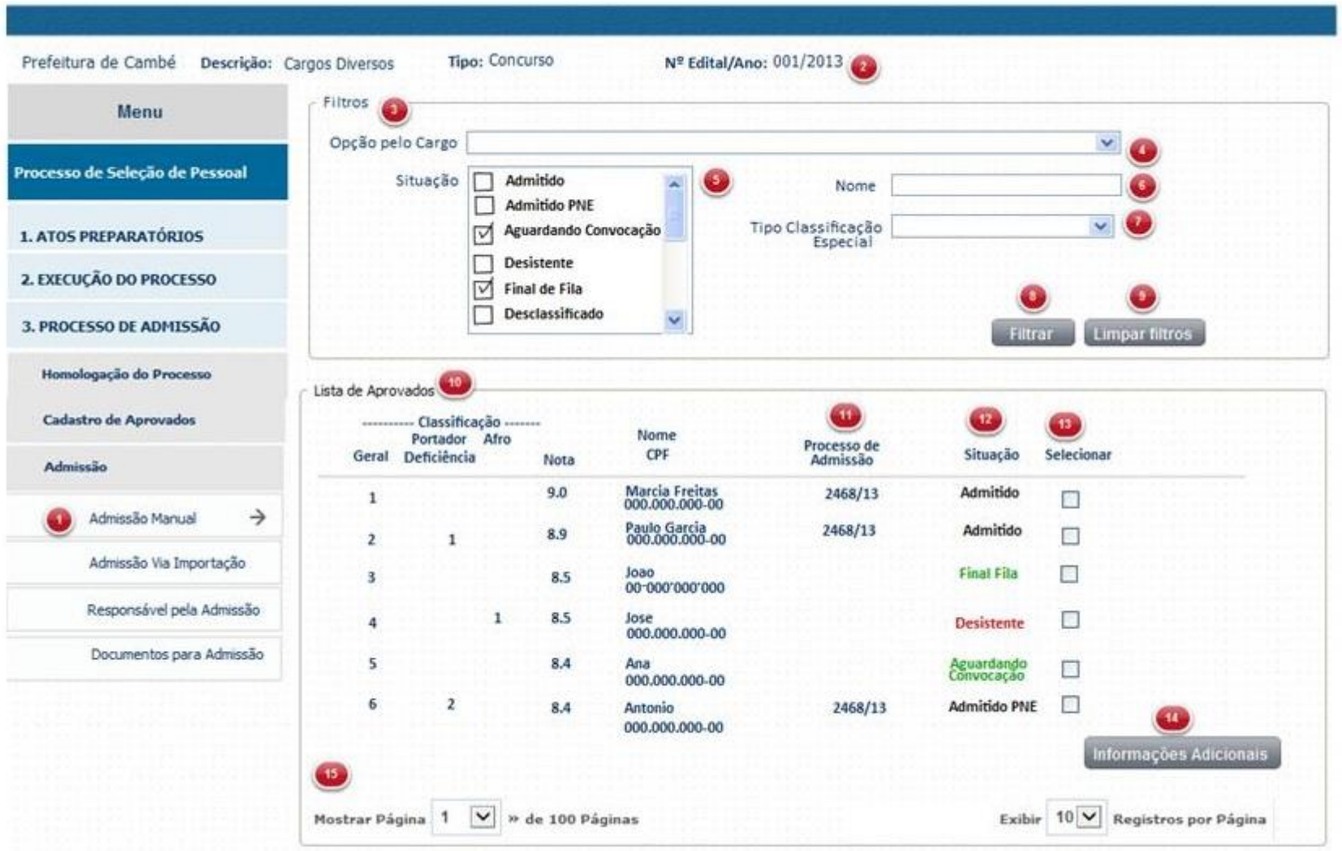


Figura 1.1 - Elementos do protótipo - Desenho com índices

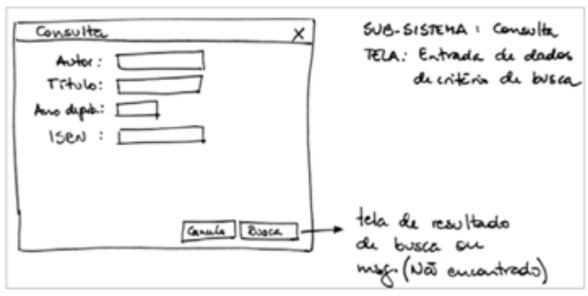


Figura 1.1 - Elementos do protótipo - Desenho de baixa fidelidade

Tab	Elemento	Tipo	Tamanho	Obrigatório	Comportamento
1	Ano	Texto	-	-	Ao abrir tela, traz o ano da remessa aberta.
2	Período	Texto	-	-	Ao abrir tela, traz o quadrimestre aberto.
*	Tree-view	Treeview			Ao abrir a tela, a treeview deve ser apresentada somente o primeiro nível espandido, ou seja, mostrando o Estado do Paraná e as entidades dependentes possuem registros em tabelas, para o ano/quadrmestre aberto. Se a entidade logada é não-dependente do Estado do Paraná (ou seja, envia seus próprios dados), o treeview inicia-se com o nome da Entidade (nivel 2)
3	Primeiro nível da tree-view (Estado do Paraná)	Texto	-	-	Primeiro nível da tree-view é a entidade Estado do Paraná. A sua seleção leva a seleção automática de todas as tabelas que a entidade pode enviar na remessa e que possuam count >0. A expansão deste nível leva à visualização do segundo nível da tree-view.

Figura 2 - Elementos do protótipo - Descrição dos campos

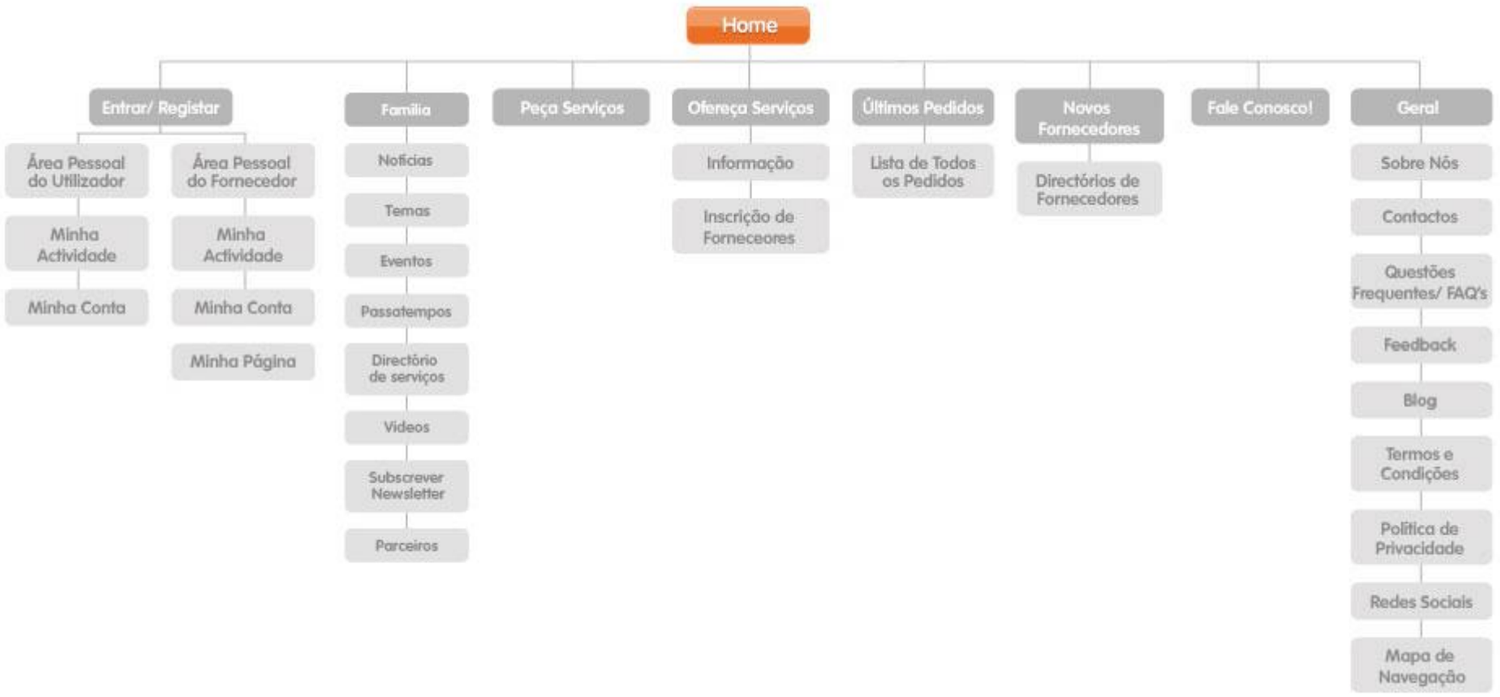


Figura 3 - Elementos do protótipo - Estrutura de menu

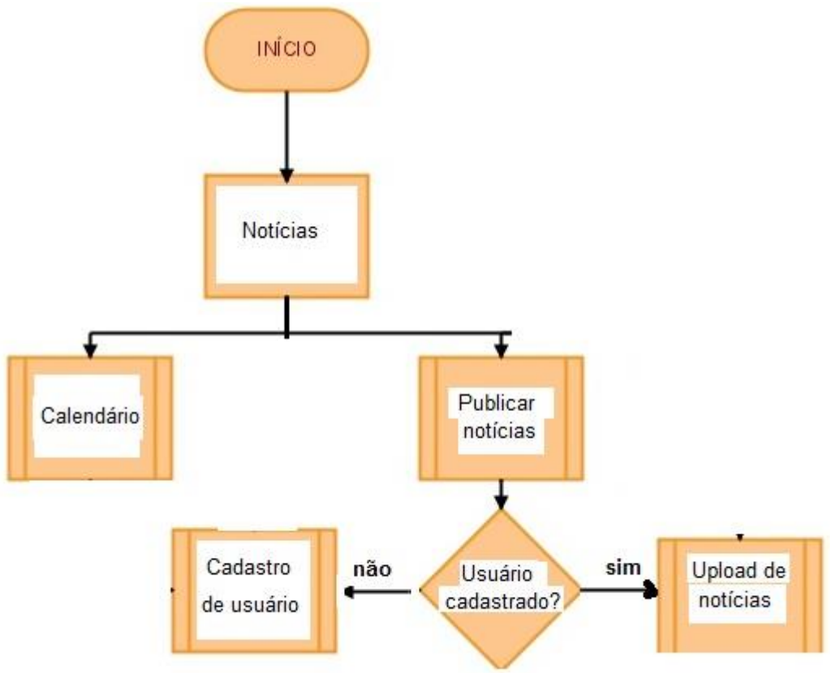


Figura 4 - Elementos do protótipo - Mapa de fluxo de navegabilidade

Tarefa: Elaborar Diagrama de Negócio

Entradas:	Pré-Projeto
Saídas:	Diagrama de Negócio
Papéis:	Analista de Demandas Gerente de Projeto Product Owner Time de Desenvolvimento
Disciplinas / Tarefas relacionadas:	Gestão de Demandas → Tarefa: Elaborar Proposta de Projeto Gestão de Demandas → Tarefa: Elaborar Pré-Projeto Requisitos → Tarefa: Identificar e Especificar Requisitos Requisitos → Tarefa: Definir Critérios de Aceite Requisitos → Tarefa: Especificar Caso de Uso Requisitos → Tarefa: Especificar História de Usuário Requisitos → Tarefa: Especificar Change Request Requisitos → Tarefa: Desenhar Protótipo

DIAGRAMA DE NEGÓCIO

O diagrama de negócio deve representar todo o caminho da informação, como é manuseada dentro da área de negócio. O propósito do diagrama do negócio é prover para toda a equipe de desenvolvimento o mecanismo de funcionamento, com os respectivos processos do cliente. Com isso é possível modelar o sistema para resolver os eventuais problemas do atual processo da área de negócio.

A maior vantagem de se utilizar o BPMN no início da especificação do pré-projeto é a facilidade de compreensão por todas as partes envolvidas: usuário (área de negócio), time de desenvolvimento.

IMPORTANTE: O diagrama de negócio deve ser um dos primeiros artefatos a serem gerados. E deve ser criado juntamente com um representante da área de negócio.

CHECKLIST DO DIAGRAMA DE NEGÓCIO

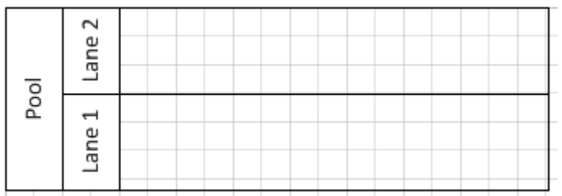
Um diagrama de negócio deve conter:

1. Todos os atores do processo representados em raias (*lanes*);
2. Todos os pontos de decisão do processo;
3. Todas as ações referentes à informação que é solicitada, ou entra no processo;
4. Todos os documentos criados na área de negócio;
5. Todos os acessos a sistemas legados do Tribunal.

COMO CRIAR O DIAGRAMA BPMN DO NEGÓCIO DO CLIENTE

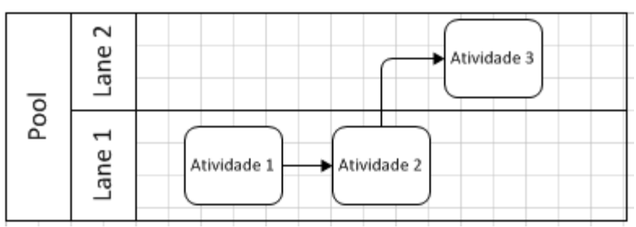
1. Identificar todos os atores (UML) participantes do negócio e suas funções no processo completo;

2. Desenhar cada ator como uma lane* no BPMN;

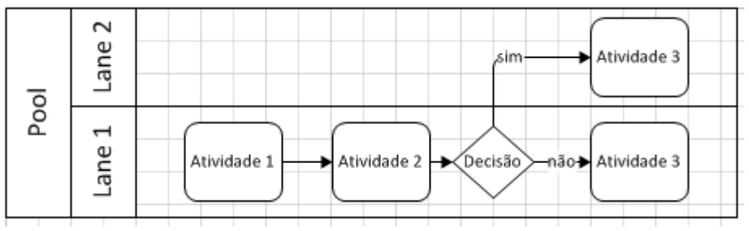


3. Especificar cada atividade** no processo; (Sugestão: incluir identificação do requisito ou caso de uso);

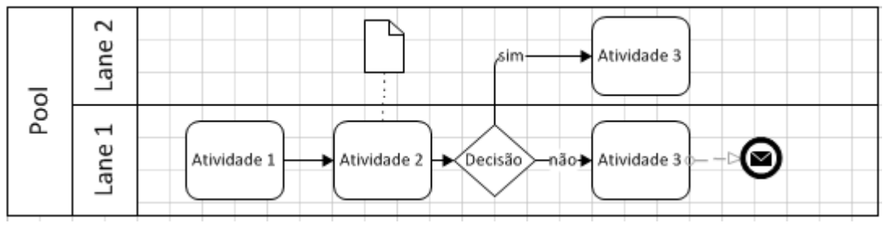
4. Posicionar cada atividade dentro da lane de seu respectivo executor (ator);



5. Identificar pontos de decisão*** que influenciem no caminho do processo;



6. Identificar envio de e-mails ou qualquer outro tipo de comunicação que devam acontecer no processo.



IMPORTANTE:

* Lanes devem representar os papéis dentro do negócio.

** Cada requisito funcional deve representar uma atividade no diagrama (caixa de processo).

*** Cada regra corresponde a um ponto de decisão no diagrama (losango).

DESENVOLVIMENTO

Esta disciplina agrupa as atividades relacionadas a construção das soluções de TI.

As tarefas definidas nesta disciplina possuem os objetivos de:

- produzir artefatos e produtos necessários ao atingimento do objetivo da iteração;
- construir o software de forma incremental, conforme padrões definidos;
- entregar incrementos de software de valor para o negócio.

Tarefas:	• Tarefa: Implementar (codificar) software • Tarefa: Entregar software e roteiro de publicação

Tarefa: Implementar (codificar) software

Entradas:	Backlog da Iteração Backlog do Produto Caso de Uso Change Request Critérios de Aceite Diagrama de Negócio Documento de Arquitetura do Projeto História de Usuário Protótipo Regras de Negócio
Saídas:	Código-fonte Incremento de Software Checklist do Desenvolvedor Item de Trabalho do Repositório Roteiro de Publicação
Papéis:	Time de Desenvolvimento
Disciplinas / Tarefas relacionadas:	Desenvolvimento → Tarefa: Entregar software e roteiro de publicação

RESUMO

Esta atividade visa produzir software funcional para o Usuário (Área de Negócio), com base na documentação elaborada pelo Time de Desenvolvimento.

IMPLEMENTAR (CODIFICAR) SOFTWARE

Selecionar a tarefa por ordem de prioridade dentro da Sprint corrente, assinalar para o seu nome e mudar o status para "*In Progress*".

A codificação é feita a partir dos artefatos de projeto: Casos de Uso, História de Usuário, Protótipos, Regras de Negócio, Modelo de Dados, Fluxo do Processo e Roteiro de Testes. A atividade de projeto deve considerar como será feito o tratamento de erros, já que uma parte significativa do código diz respeito ao tratamento de erros.

Desenvolver o código utilizando os padrões de arquitetura, segurança, qualidade, boas práticas de programação, padrões de regras de código e de versionamento: Padrões de Código.

Para alterações em Stored Procedures e Functions deve-se incluir comentários como autor, data, tarefa, descrição resumida da alteração e atualizar o .sql na Solution.

VERSIONAMENTO

Utilizar os comentários de Check-in ao versionar o código no TFS e vincular à tarefa (encerrar ou associar à tarefa) e incluir um comentário resumido do que foi feito.

ARTEFATOS PRODUZIDOS

- A codificação frequentemente leva a modificações em decisões anteriores. Portanto, depois da codificação, os artefatos já produzidos devem ser sincronizados com o código.
- Atualizar o documento do Roteiro de Publicação (alteração no Banco de dados, web.config, Procedures, function, Grants etc.).
- O próprio Código.

Tarefa: Entregar software e roteiro de publicação

Entradas:	Caso de Uso Change Request Código-fonte Documento de Arquitetura do Projeto História de Usuário Protótipo Regras de Negócio
Saídas:	Código-fonte Incremento de Software Roteiro de Publicação
Papéis:	Time de Desenvolvimento
Disciplinas / Tarefas relacionadas:	Desenvolvimento → Tarefa: Implementar (codificar) software
Local:	/$Projeto/Documents/Desenvolvimento/Roteiros de Publicação/Sprint XX – Roteiro de publicação.docx

ENTREGA DE SOFTWARE

A entrega de software é o incremento de software entregue ao término de cada iteração (Sprint).

ELABORAÇÃO DO DOCUMENTO ROTEIRO DE PUBLICAÇÃO

A elaboração do documento Roteiro de Publicação deverá considerar os seguintes aspectos:

- Listar todos objetos que sofreram alteração e qual alteração aconteceu (criação, alteração, remoção), especificando-os nas suas respectivas seções:
  - Tabelas;
  - Views;
  - Funções;
  - Sinônimos;
  - Procedures.
- Scripts: esta seção deverá conter os scripts necessários para implantação da iteração do sistema: inserts, updates e deletes de dados. Se julgar necessário, a equipe também poderá acrescentar aqui os scripts de alteração estrutural (DDL), além dos apontamentos já feitos na parte inicial do documento.
- Pastas e permissões: especificar pastas e permissões específicas dos servidores de produção que o sistema depende e que devem ter atenção durante a implantação.
- Aplicação – IIS: especificar detalhes de configuração do IIS/servidor que são necessários para a aplicação.
- Controle de Acesso – CIA: especificar novas funcionalidades que precisam ser inseridas no Controle de Acesso e/ou alterações em perfis/funcionalidades já existentes. Se preferir, a equipe pode colocar diretamente os scripts SQL necessários para ajustes no CIA.
- Script de *rollback*: caso a publicação seja mal-sucedida.

- Incremento de software: qual a *label* ou *branch* do versionamento de código deverá ser publicada.

TESTES

Esta disciplina agrupa as atividades relacionadas ao processo de Testes dos produtos de software.

A realização de testes é uma etapa muito importante do desenvolvimento de software. Esta etapa está fortemente vinculada à garantia de qualidade. Assim como outras etapas do desenvolvimento, devem ser padronizadas para uniformizar o entendimento de certos resultados, os testes também devem ter alguns parâmetros estabelecidos para fins de análise de resultados.

Testes durante o desenvolvimento

A forma mais eficiente de se trabalhar de forma iterativa e incremental no desenvolvimento de um sistema é se compartilhar a responsabilidade dos testes com os desenvolvedores, uma vez que, são eles que estão mais próximos do código e suas respectivas alterações. A cada comportamento entregue deve-se associar a automatização de testes, ou seja, a realização de código de teste que ponha a prova o comportamento esperado do código. O Comportamento em si deve ser de conhecimento do desenvolvedor antes mesmo do início da codificação e deve ser especificado pela área de negócio juntamente com a área de TI.

Automatização de testes

O propósito da automatização é reduzir o retrabalho de testes à medida que o sistema vai crescendo ou sendo alterado. Toma-se como constante o comportamento esperado e a partir daí, tudo pode mudar, o código, o algoritmo, a modularidade, menos o comportamento. O conceito relacionado a testes de comportamento é o conceito de BDD.

BDD (*Behaviour Driven Development*)

Behaviour Driven Development (BDD), ou em português, Desenvolvimento guiado por comportamento, propõe a automatização de testes com foco em comportamentos do sistema/módulo, podendo abranger ou um único método ou um conjunto de requisitos que se traduzem em uma interação com uma tela.

TDD (*Test Driven Development*)

Test Driven Development (TDD), ou em português Desenvolvimento guiado por testes, é uma técnica de desenvolvimento de software que se baseia em um ciclo curto de repetições. Primeiramente o desenvolvedor escreve um caso de teste automatizado que define uma melhoria desejada ou uma nova funcionalidade. Então, é produzido código que possa ser validado pelo teste, para posteriormente o código ser refatorado para um código sob padrões aceitáveis.

Testes após o desenvolvimento

Os testes exploratórios devem ser realizados principalmente pelo Analista do projeto para identificar erros não contemplados pelos critérios de aceite. Por exemplo: validação de dados em campos, verificação de dados inseridos, alterados ou excluídos no banco, falhas de exibição ou erros de português. Deve-se utilizar sempre um roteiro que pode ser traduzido em um checklist de itens que devem ser avaliados durante o teste exploratório. O checklist tem a finalidade de funcionar como um lembrete.

As tarefas definidas nesta disciplina possuem os objetivos de:

- garantir a qualidade dos softwares produzidos;
- garantir a satisfação das expectativas dos clientes, dentro daquilo que foi acordado (critérios de aceite);
- aumentar a eficiência do processo de testes através da automatização (testes de regressão).

Tarefas:	• Tarefa: Elaborar Plano de Teste • Tarefa: Implementar (codificar) Testes Automatizados • Tarefa: Executar Testes Automatizados • Tarefa: Executar Testes Manuais (Critérios de Aceite e Exploratórios)

Tarefa: Elaborar Plano de Teste

Entradas:	Critérios de Aceite
Saídas:	Plano de Teste
Papéis:	Time de Desenvolvimento
Disciplinas / Tarefas relacionadas:	Requisitos → Tarefa: Definir Critérios de Aceite Testes → Tarefa: Implementar (codificar) Testes Automatizados Testes → Tarefa: Executar Testes Automatizados Testes → Tarefa: Executar Testes Manuais (Critérios de Aceite e Exploratórios)

RESUMO

A elaboração do plano de teste deve ser conduzida pelo Analista Programador. Esse documento deve conter casos de testes para as regras identificadas nos critérios de aceite e mais as que o Analista Programador identificar.

PLANO DE TESTE

O plano de teste é decorrente dos critérios de aceite e deve ser elaborado pelo Analista Programador. O principal objetivo é identificar os comportamentos que o responsável pelo negócio espera encontrar no sistema quando este for entregue.

Para cada requisito deve haver um documento de Plano de testes formalizado no início do ciclo.

Este documento deve contemplar todos os itens de CRUD por requisito: inserção, alteração, pesquisa e exclusão e suas condições específicas de comportamento, possíveis vínculos e restrições com outros requisitos, dados necessários para automatização, scripts para inicialização/reinicialização do ambiente de testes e assim por diante.

Tarefa: Implementar (codificar) Testes Automatizados

Entradas:	Plano de Teste
Saídas:	Código-fonte
Papéis:	Time de Desenvolvimento
Disciplinas / Tarefas relacionadas:	Testes → Tarefa: Elaborar Plano de Teste Testes → Tarefa: Executar Testes Automatizados

RESUMO

Tarefa necessária para conseguir realizar mais facilmente os teste de regressão em sistemas em constante modificação.

AUTOMATIZAÇÃO DE TESTES

A automatização dos testes deve ser realizada pelo Analista Programador em paralelo ao desenvolvimento das funcionalidades.

a. Isole a funcionalidade a ser testada;
b. Identifique os dados que devem ser utilizados no teste;
c. Leia os cenários de teste;
d. Crie Script de inserção dos dados em ambiente de teste;
e. Crie o código que testa a funcionalidade (com *Specflow*);
f. Crie o Script de Exclusão dos dados utilizados para o teste da funcionalidade;
g. Execute o Script de Inclusão de dados;
h. Execute o teste automatizado;
i. Registre o relatório de teste (Se passou ou não o teste);
j. Execute o Script de exclusão de dados utilizados para o teste da funcionalidade.

Os testes devem ser produzidos de forma que possam ser sempre reexecutados.

O objetivo da automatização dos testes é prevenir o desgaste de re-testes complexos a cada pequena modificação de código que se faça necessária.

Tarefa: Executar Testes Automatizados

Entradas:	Plano de Teste Código-fonte
Saídas:	Relatório de Execução dos Testes Automatizados
Papéis:	Gerente de Projeto Product Owner Time de Desenvolvimento
Disciplinas / Tarefas relacionadas:	Testes → Tarefa: Elaborar Plano de Teste Testes → Tarefa: Implementar (codificar) Testes Automatizados

RESUMO

Tarefa que deve ser executada antes da liberação do sistema para homologação do Usuário.

EXECUÇÃO DOS TESTES AUTOMATIZADOS

A execução dos testes automatizados deve ser realizada sempre que houver acréscimo de funcionalidades no sistema ou mudanças de requisitos com o objetivo de verificar o impacto das novas funcionalidades ou alterações no que já está pronto. Seu principal atrativo é evitar o re-teste manual à medida que requisitos sejam alterados.

É de responsabilidade do Analista Programador e do *Product Owner* executar os testes automatizados em paralelo ao fim de cada iteração do projeto, com a finalidade de identificar o que foi impactado com o crescimento do sistema. É necessário também gerar o relatório dos testes que falharam e foram bem-sucedidos na execução.

Tarefa: Executar Testes Manuais (De acordo com Critérios de Aceite e Exploratórios)

Entradas:	Plano de Teste Critérios de Aceite
Saídas:	Relatório de Execução dos Testes Manuais (Critérios de Aceite e Exploratórios)
Papéis:	Gerente de Projeto Product Owner Time de Desenvolvimento
Disciplinas / Tarefas relacionadas:	Requisitos → Tarefa: Definir Critérios de Aceite Testes → Tarefa: Elaborar Plano de Teste

RESUMO

Os testes manuais são aqueles que não foram automatizados e podem seguir um roteiro (contido no plano de testes) ou não seguir um roteiro para descobrir falhas que não foram previstas.

EXECUÇÃO DE TESTES MANUAIS

Executar Testes Manuais (de acordo com os critérios de aceite)

Os testes manuais devem estar previstos no documento de critérios de aceite. Esses devem ser executados durante cada iteração do projeto para garantir que os produtos foram desenvolvidos conforme especificação e que não houve impacto no que já estava funcionando. É de responsabilidade do Analista Programador e do *Product Owner* executar os testes manuais.

Executar Testes Manuais (Exploratórios)

Nesse caso, os testes exploratórios não foram previstos no documento de critérios de aceite e devem ser realizados pelo *Product Owner* com o objetivo de identificar falhas. Cada falha encontrada deve ser documentada com um conjunto de informações de entrada que permitiram o teste (cenário). Essas informações devem então ser repassadas ao Analista Programador para que as correções sejam analisadas e realizadas.

QUALIDADE

Esta disciplina agrupa as atividades relacionadas ao processo de validação da qualidade do software.

As tarefas definidas nesta disciplina possuem os objetivos de:

- garantir a qualidade dos produtos de software entregues;
- garantir a qualidade da implementação e execução dos testes de software.

Tarefas:	• Tarefa: Validar qualidade das entregas e dos testes

Tarefa: Validar qualidade das entregas e dos testes

Entradas:	Padrões de Qualidade Plano de Teste Código-fonte Relatório de Execução dos Testes Automatizados Relatório de Execução dos Testes Manuais (Critérios de Aceite e Exploratórios)
Saídas:	Documento de Validação de Qualidade
Papéis:	Time de Desenvolvimento
Disciplinas / Tarefas relacionadas:	Testes → Tarefa: Elaborar Plano de Teste Testes → Tarefa: Implementar (codificar) Testes Automatizados Testes → Tarefa: Executar Testes Automatizados Testes → Tarefa: Executar Testes Manuais (Critérios de Aceite e Exploratórios)

RESUMO

Essa tarefa visa validar a qualidade de entrega realizada antes que o sistema seja disponibilizado para o usuário para homologação.

VALIDAÇÃO DA QUALIDADE DAS ENTREGAS E DOS TESTES

Para garantir que a entrega seja satisfatoriamente homologada pelo usuário com o mínimo de problemas, a validação da qualidade das entregas e dos testes deverá ser feita através de um checklist que garanta a:

- execução dos testes automatizados;
- realização dos testes manuais;
- validação do sistema com o checklist de qualidade;
- validação do código pelo *SonarQube*;
- validação das funcionalidades pelo Product Owner e pelo Analista Programador.

SEGURANÇA

Esta disciplina agrupa as atividades relacionadas ao processo de validação da segurança do software.

As tarefas definidas nesta disciplina possui o objetivo de:

- garantir a construção de softwares seguros, seguindo os padrões definidos.

Tarefas:	• Tarefa: Validar segurança do software

Tarefa: Validar segurança do software

Entradas:	Padrões de Segurança Código-fonte
Saídas:	Documento de Validação de Segurança
Papéis:	Time de Desenvolvimento
Disciplinas / Tarefas relacionadas:	-

RESUMO

Esta tarefa tem por objetivo validar se as boas práticas para a construção de produtos de software seguros, definido em Padrões de Segurança, foram seguidos.

VALIDAÇÃO DA SEGURANÇA DO SOFTWARE

Para validar é utilizado um checklist com as principais práticas e quem for executar esta tarefa deve identificar de forma objetiva se os padrões foram seguidos.

A segurança do software não será validada se alguma das perguntas do checklist for respondida com NÃO.

ARQUITETURA

Esta disciplina agrupa as atividades relacionadas ao processo de especificação e revisão da arquitetura do software.

As tarefas definidas nesta disciplina possuem os objetivos de:

- garantir que os softwares serão construídos conforme padrões de arquitetura definidos;
- garantir a integração dos sistemas;
- aumentar a performance e a segurança dos softwares desenvolvidos, através da implementação das melhores práticas de desenvolvimento;
- facilitar a manutenção dos sistemas, através da padronização da arquitetura.

Tarefas:	• Tarefa: Especificar Arquitetura do Projeto • Tarefa: Revisar arquitetura do software

Tarefa: Especificar Arquitetura do Projeto

Entradas:	Pré-Projeto Padrões de Arquitetura
Saídas:	Documento de Arquitetura
Papéis:	Arquiteto Time de Desenvolvimento
Disciplinas / Tarefas relacionadas:	Arquitetura → Tarefa: Revisar arquitetura do software
Local:	/$Projeto/Documents/Arquitetura/DA – Arquitetura do projeto.docx

RESUMO

A elaboração da Arquitetura do Projeto deve levar em conta os requisitos do projeto, integrações que serão realizadas com outros sistemas, modelo de dados, bem como demais características específicas (quantidade de usuários simultâneos, acessos externos/internos, etc).

REUNIÃO PARA ESPECIFICAÇÃO DA ARQUITETURA DO PROJETO

Deverá ser realizada uma reunião com a equipe do projeto, antes do início do desenvolvimento, para abordar os seguintes assuntos:

- Objetivo do projeto;
- Funcionalidades previstas;
- Requisitos;
- Modelo de Dados;
- Integrações com outros sistemas da casa;
- Integrações com sistemas de terceiros;
- Proposta de Arquitetura:
    - Padrões de design de aplicação que serão utilizados;
    - Padrão estrutural;
    - Tecnologia e versão de frameworks que serão utilizados;
- Justificativa da Arquitetura utilizada, caso enseje a utilização de uma arquitetura própria.

Nesta reunião, com base nos pontos discutidos acima, será elaborada a proposta de Arquitetura adequada para o projeto, bem como será justificada a utilização de uma arquitetura que seja diferente do padrão sugerido pelo Tribunal.

Dependendo dos requisitos do sistema, integrações, modelo de dados e outras características, podem ser necessárias alterações estruturais na Arquitetura, para que esta seja mais adequada e aderente ao sistema que será desenvolvido.

Tarefa: Revisar arquitetura do software

Entradas:	Padrões de Arquitetura Documento de Arquitetura Código-fonte Incremento de Software
Saídas:	Documento de Revisão de Arquitetura
Papéis:	Time de Desenvolvimento
Disciplinas / Tarefas relacionadas:	Arquitetura → Tarefa: Especificar Arquitetura do Projeto
Local:	/$Projeto/Documents/Arquitetura/Revisões/DA – Revisão de Arquitetura – Sprint XX.docx

RESUMO

A revisão deve ser feita a cada iteração (Sprint) do projeto, para acompanhar a execução do projeto e atestar se os padrões sugeridos e acordados no início do projeto (documento de Arquitetura do Projeto) estão sendo seguidos corretamente. E, para o caso de algo não estar sendo feito conforme combinado, evidenciar o problema e sugerir as correções necessárias.

ELABORAÇÃO DO DOCUMENTO DE REVISÃO DE ARQUITETURA

A elaboração do documento de Revisão de Arquitetura deverá considerar os seguintes aspectos:

- Conformidade do projeto/código entregue com a Arquitetura do Projeto;
- Conformidade com documentos de Especificação Técnica, caso existam;
- Conformidade de Estrutura do Projeto: verificar se existem violações de camadas, problemas de codificação, nomes de classes e objetos fora do padrão, etc;
- Conformidade de Padrões de Código: revisão automatizada que será realizada pela ferramenta *SonarQube*. Deve-se conferir o atendimento de todos os indicadores (*quality gates*) previstos no Termo de Referência;
- Considerações e sugestões de correção/melhoria: ao final do documento deverão ser colocadas as considerações com sugestões de correções e melhorias necessárias.

HOMOLOGAÇÃO

Esta disciplina agrupa as atividades relacionadas ao processo de homologação dos softwares construídos.

As tarefas definidas nesta disciplina possuem os objetivos de:

- garantir que o software foi construído conforme aquilo que foi especificado com os clientes;
- garantir que o software seja testado e validado pelo cliente antes de entrar em produção.

Tarefas:	• Tarefa: Homologar software

Tarefa: Homologar software

Entradas:	Critérios de Aceite Incremento de Software
Saídas:	-
Papéis:	Gerente de Projeto Product Owner Usuário (Área de Negócio)
Disciplinas / Tarefas relacionadas:	Implantação → Tarefa: Publicar software no Ambiente de Produção

RESUMO

Tarefa de responsabilidade do Usuário (Área de Negócio) que deve ser realizada a cada entrega e antes da publicação em produção.

HOMOLOGAÇÃO DE SOFTWARE

O usuário deverá utilizar o documento de Critérios de Aceite como referência para os testes que irá realizar. Todos os problemas devem ser reportados ao time de desenvolvimento para que sejam resolvidos antes da publicação do sistema em produção.

IMPLANTAÇÃO

Esta disciplina agrupa as atividades relacionadas ao processo de implantação dos softwares construídos nos ambientes de desenvolvimento, testes, homologação e produção.

As tarefas definidas nesta disciplina possuem os objetivos de:

- permitir a publicação das soluções de TI desenvolvidas nos ambientes de desenvolvimento, testes, homologação e produção;
- garantir que as publicações ocorram mediante fluxos de aprovação definidos, caso necessário, evitando transtornos em decorrência de publicações de softwares com problemas ou em momentos inadequados.

Tarefas:	• Tarefa: Publicar software no Ambiente de Desenvolvimento ou Testes • Tarefa: Publicar software no Ambiente de Homologação • Tarefa: Publicar software no Ambiente de Produção

Tarefa: Implantar o software no Ambiente de Desenvolvimento ou Testes

Entradas:	Código-fonte Incremento de Software Roteiro de Publicação
Saídas:	Incremento de Software
Papéis:	Time de Desenvolvimento
Disciplinas / Tarefas relacionadas:	Implantação → Tarefa: Publicar software no Ambiente de Homologação Implantação → Tarefa: Publicar software no Ambiente de Produção
Local:	Servidor de aplicação de desenvolvimento (IIS-DES)

RESUMO

A publicação de software em ambiente de desenvolvimento ou testes será feita a critério do time de desenvolvimento ou do Product Owner.

ORIENTAÇÕES PARA PUBLICAÇÃO DO SOFTWARE

1. Atualizar projeto de banco de dados do *branch* de desenvolvimento (banco de desenvolvimento -> projeto). Desprezar alterações de usuários e roles.
2. Efetuar check-in das alterações no projeto de banco de dados.
3. Alterar número de versão do código (arquivo AssemblyInfo.cs do projeto Web, utilizar a data invertida como número de versão).
4. Compilar nova versão do projeto e gerar os arquivos para publicação.
5. Retirar o sistema do ar (via app_offline.htm) no ambiente de desenvolvimento.
6. Publicar nova versão no servidor de desenvolvimento: copiar os arquivos manualmente ou via Wizard do Visual Studio. Deve-se ter atenção especial ao arquivo Web.Config, para que não sejam estragadas configurações de conexão com banco de dados, dependências do ReportingViewer, Elmah etc.
7. Remover o arquivo app_offline.htm para voltar a operação do sistema.
8. Conferir a versão da DLL publicada no servidor de produção (deve ser igual a data da publicação invertida).
9. Acessar o sistema e executar um teste rápido para verificar se está tudo funcionando conforme o esperado.

Tarefa: Implantar o software no Ambiente de Homologação

Entradas:	Código-fonte Incremento de Software Roteiro de Publicação
Saídas:	Incremento de Software
Papéis:	Administrador de Infraestrutura Time de Desenvolvimento
Disciplinas / Tarefas relacionadas:	Implantação → Tarefa: Publicar software no Ambiente de Desenvolvimento ou Testes Implantação → Tarefa: Publicar software no Ambiente de Produção
Local:	Servidor de aplicação de homologação (IIS-HML)

RESUMO

A publicação de software em ambiente de homologação será feita a critério do time de desenvolvimento ou do Product Owner, disponibilizando uma nova versão do sistema pronta para ser homologada e testada pelo Usuário (Área de Negócio).

ORIENTAÇÕES PARA PUBLICAÇÃO DO SOFTWARE

1. Merge de código (Desenvolvimento -> Base, Base -> Homologação). Neste merge também será atualizado o projeto de banco de dados de homologação. Efetuar check-in das alterações que subiram no merge.
2. Alterar número de versão do código (arquivo AssemblyInfo.cs do projeto Web, utilizar a data invertida como número de versão).
3. Atualizar banco de dados de homologação (se houver alterações significativas que podem gerar erros no sistema, antes deve-se tirar o sistema do ar com o arquivo app_offline.htm). Esta atualização pode ser feita comparando diretamente o banco de desenvolvimento e homologação, mas depois disso é importante comparar também o projeto de banco de homologação com o respectivo banco, para garantir que o projeto reflete o estado atual deste.
4. Compilar nova versão do projeto e gerar os arquivos para publicação.
5. Retirar o sistema do ar (via app_offline.htm) no ambiente de homologação.
6. Publicar nova versão no servidor de homologação: copiar os arquivos manualmente ou via Wizard do Visual Studio. Deve-se ter atenção especial ao arquivo Web.Config, para que não sejam estragadas configurações de conexão com banco de dados, dependências do ReportingViewer, Elmah etc.
7. Remover o arquivo app_offline.htm para voltar a operação do sistema.
8. Conferir a versão da DLL publicada no servidor de produção (deve ser igual a data da publicação invertida).
9. Acessar o sistema e executar um teste rápido para verificar se está tudo funcionando conforme o esperado.

Tarefa: Implantar o software no Ambiente de Produção

Entradas:	Código-fonte Incremento de Software Roteiro de Publicação
Saídas:	Incremento de Software
Papéis:	Administrador de Infraestrutura Time de Desenvolvimento
Disciplinas / Tarefas relacionadas:	Homologação → Tarefa: Homologar software Implantação → Tarefa: Publicar software no Ambiente de Desenvolvimento ou Testes Implantação → Tarefa: Publicar software no Ambiente de Homologação
Local:	Servidor de aplicação de produção

RESUMO

A publicação de software em ambiente de produção será feita a critério do Product Owner, disponibilizando uma nova versão do sistema pronta para utilização pelo Usuário (Área de Negócio).

ORIENTAÇÕES PARA PUBLICAÇÃO DO SOFTWARE

1. Merge de código (Homologação -> Base, Base -> Produção). Neste merge também será atualizado o projeto de banco de dados de produção; Efetuar check-in das alterações que subiram no merge.
2. Alterar número de versão do código (arquivo *AssemblyInfo.cs* do projeto Web, utilizar a data invertida como número de versão).
3. Atualizar banco de dados de produção (se houver alterações significativas que podem gerar erros no sistema, antes deve-se tirar o sistema do ar com o arquivo app_offline.htm). Esta atualização pode ser feita comparando diretamente o banco de homologação e produção, mas depois disso é importante comparar também o projeto de banco de produção com o respectivo banco, para garantir que o projeto reflete o estado atual deste.
    a. No caso de projetos que ainda não possuem base de dados própria e separada dos demais sistemas (utiliza a base de dados "TC"), as alterações de banco de dados apontadas no merge dos projetos deverão ser manualmente encaminhadas para a Equipe de Banco de Dados através de solicitação de serviço interna do TCE.
4. Gerar backup da versão atual do sistema no diretório do servidor de backup. Aqui deve ser respeitada a estrutura de pastas do servidor onde será publicada a nova versão. Dentro deste diretório, criar uma pasta cujo nome é a data da publicação invertida (ex: 20160211) e copiar o conteúdo completo do site.
5. Compilar nova versão do projeto e gerar os arquivos para publicação.
6. Retirar o sistema do ar (via app_offline.htm) no ambiente de produção.
7. Publicar nova versão no servidor de produção: copiar os arquivos manualmente ou via Wizard do Visual Studio. Deve-se ter atenção especial ao arquivo Web.Config, para que não sejam estragadas configurações de conexão com banco de dados, dependências do ReportingViewer, Elmah etc.
8. Remover o arquivo app_offline.htm para voltar a operação do sistema.
9. Conferir a versão da DLL publicada no servidor de produção (deve ser igual a data da publicação invertida).
10. Acessar o sistema e executar um teste rápido para verificar se está tudo funcionando conforme o esperado.

TRANSFERÊNCIA DE TECNOLOGIA

Esta disciplina agrupa as atividades relacionadas ao processo de transferência de tecnologia do software desenvolvido.

As tarefas definidas nesta disciplina possuem os objetivos de:

- registrar o conhecimento adquirido e utilizado na construção da solução de TI;
- permitir que as áreas de Sustentação e Infraestrutura possam manter o sistema operacional, construir novas funcionalidades e modificar o sistema transferido.

Tarefas:	• Tarefa: Elaborar ou Atualizar Documento de Transferência de Tecnologia • Tarefa: Aceitar Transferência de Tecnologia

Tarefa: Elaborar ou Atualizar Documento de Transferência de Tecnologia

Entradas:	Backlog do Produto Caso de Uso Change Request Código-fonte Diagrama de Negócio Documento de Arquitetura do Projeto História de Usuário Incremento de Software Plano de Teste Regras de Negócio Roteiro de Publicação
Saídas:	Documento de Transferência de Tecnologia
Papéis:	Gerente de Projeto Product Owner Time de Desenvolvimento
Disciplinas / Tarefas relacionadas:	Transferência de Tecnologia → Tarefa: Aceitar Transferência de Tecnologia

RESUMO

A elaboração e atualização do Documento de Transferência de Tecnologia é essencial para que o software desenvolvido possa ser transferido para as áreas de Sustentação e Infraestrutura após o término do projeto (ou da manutenção realizada pela equipe de Sustentação), permitindo que outros, que não o criador do projeto, possam manter o sistema operacional, construir novas funcionalidades e modificar o sistema transferido.

DOCUMENTO DE TRANSFERÊNCIA DE TECNOLOGIA

Este documento coleta informações sobre as tecnologias empregadas, os dados (ex.: modelo de dados), a arquitetura de software, o modelo de desenvolvimento de software, onde os códigos-fonte e executáveis são armazenados, as configurações de software e de hardware (ex.: configurações de servidores e de equipamentos de rede), os procedimentos de produção (ex.: forma de execução de backups), entre outros.

No documento devem ser descritas as características do sistema abaixo, conforme exemplificado no documento TT_Transferencia_Tecnologia_NomeProjeto.doc.

Dados do Projeto

- Informar dados do projeto.
- *Branches*: informar quais são os *branches* de desenvolvimento, homologação e produção.

Estrutura do Projeto

- Adicionar o print screen da Solution Explorer do Visual Studio.
- Diretórios: descrever o conteúdo do diretório, se houver.
- Projetos: descrever cada projeto da Solution.
- Diagrama de integração com outros sistemas: inserir o diagrama que mostre a integração desse sistema com os demais sistemas do TCE.

Bibliotecas

- Informar outras bibliotecas utilizadas no projeto.
- Para cada biblioteca informar o Nome, versão, site do Projeto e objetivo de utilização.

Diagrama de Rede

- Inserir o modelo para cada ambiente.

Estrutura do IIS

- Adicionar informações de cada ambiente de publicação.

Configuração de Banco de Dados

- Definir se existem componentes para serem instalados e configurados, triggers, índices e quais usuários autorizados.

Scripts

- Definir os scripts utilizados em cada projeto.

CSS

- Definir os temas utilizados no Projeto.

WebService de Integração

- Definir os WebServices oferecidos pelo projeto.

Serviço Windows

- Descrever os serviços Windows.

APIs

- Definir as APIs e as configurações utilizadas no projeto.

Integrações dos sistemas do TCE com esse projeto

- Definir os sistemas do TCE e WebServices consumidos pelo projeto.

Particularidades desse projeto

- Definir pontos específicos do projeto que não foram descritos acima. Ex.: registro de logs.

Tarefa: Aceitar Transferência de Tecnologia

Entradas:	Documento de Transferência de Tecnologia
Saídas:	Documento de Transferência de Tecnologia
Papéis:	Gerente de Sustentação Gerente de Infraestrutura
Disciplinas / Tarefas relacionadas:	Transferência de Tecnologia → Tarefa: Elaborar ou Atualizar Documento de Transferência de Tecnologia

RESUMO

A aceitação da transferência de tecnologia deve ser realizada pelas áreas de Sustentação e Infraestrutura.

PRÉ-REQUISITOS PARA TRANSFERÊNCIA DE TECNOLOGIA

Para que ocorra a transferência de tecnologia, a equipe de desenvolvimento deverá realizar:

- treinamento para capacitação das equipes técnicas das áreas de Sustentação e Infraestrutura;
- atualização do Portfólio de Sistemas.

DOCUMENTAÇÃO NECESSÁRIA PARA TRANSFERÊNCIA DE TECNOLOGIA

Além do Documento de Transferência de Tecnologia devidamente preenchido, a equipe de desenvolvimento deverá apresentar a seguinte documentação:

- Modelo de Dados;
- Casos de Uso ou Histórias de Usuário;
- Regras de Negócio;
- Diagrama de Negócio;
- Manual;
- Roteiro de Testes.

ACEITE DA TRANSFERÊNCIA DE TECNOLOGIA

O aceite da transferência de tecnologia deverá ser realizado pelo Gerente de Sustentação e pelo Gerente de Infraestrutura, mediante assinatura em campo próprio para esta finalidade no Documento de Transferência de Tecnologia.

MANUTENÇÕES REALIZADAS PELA EQUIPE DE SUSTENTAÇÃO

Quando as manutenções no software forem realizadas pela equipe de Sustentação, esta deverá atualizar o Documento de Transferência de Tecnologia e realizar a transferência de tecnologia para a área de Infraestrutura.

4. PRODUTOS

Produtos são gerados pela execução das atividades do processo de desenvolvimento de software. Além de saída dos processos e atividades, eles podem representar a entrada para outros processos e atividades.

Os produtos estabelecidos no PDS TCEPR são:

- Ata de Reunião
- Backlog da Iteração
- Backlog do Produto
- Caso de Uso
- Change Request
- Checklist do Desenvolvedor
- Código-fonte
- Critérios de Aceite
- Diagrama de Negócio
- Documento de Arquitetura do Projeto
- Documento de Retrospectiva
- Documento de Revisão de Arquitetura
- Documento de Transferência de Tecnologia
- Documento de Validação de Interface
- Documento de Validação de Qualidade
- Documento de Validação de Segurança
- História de Usuário
- Incremento de Software
- Indicadores de Qualidade e Gestão
- Item de Trabalho do Repositório
- Ordem de Serviço
- Plano de Teste
- Pré-Projeto
- Proposta de Projeto
- Protótipo
- Regras de Negócio
- Relatório de Execução dos Testes Automatizados
- Relatório de Execução dos Testes Manuais (Critérios de Aceite e Exploratórios)
- Roteiro de Publicação
- Solicitação de Serviço
- Termo de Abertura de Projeto
- Termo de Encerramento de Projeto
- Termo de Recebimento de Serviço

Ata de Reunião

Ciclo de Vida:	Todo o Ciclo de Vida
Disciplinas / Tarefas:	Gestão de Projetos → Tarefa: Elaborar Ata de Reunião
Papéis:	Todos os Papéis

O documento Ata de Reunião registra os assuntos tratados em uma reunião.

Documento Modelo:

[Projeto] - ATA Reunião [aaaammdd] - [Fase do Projeto].docx

Backlog da Iteração

Ciclo de Vida:	Desenvolvimento de Software
Disciplinas / Tarefas:	Gestão de Projetos → Tarefa: Iniciar Iteração (planejamento)
Papéis:	Gerente de Projeto Product Owner Scrum Master Time de Desenvolvimento Usuário (Área de Negócio)

É um conjunto de itens do Backlog do Produto selecionados para a iteração, juntamente com o plano para entregar o incremento do produto e atingir o objetivo da iteração/Sprint.

Somente o Time de Desenvolvimento pode alterar o Backlog da iteração/Sprint durante a sua execução.

Documento Modelo:

Backlog do Produto

Ciclo de Vida:	Desenvolvimento de Software
Disciplinas / Tarefas:	Gestão de Projetos → Tarefa: Definir Escopo do Projeto (priorização e estimativa de esforço)
Papéis:	Gerente de Projeto Product Owner Scrum Master Usuário (Área de Negócio)

É uma lista ordenada de tudo que deve ser necessário no produto, e é uma origem única dos requisitos para qualquer mudança a ser feita no produto. O Product Owner é responsável pelo Backlog do Produto, incluindo seu conteúdo, disponibilidade e ordenação.
O refinamento do Backlog do Produto é a ação de adicionar detalhes, estimativas e ordem aos itens no Backlog do Produto. Este é um processo contínuo em que o Product Owner e o Time de Desenvolvimento colaboram nos detalhes dos itens do Backlog do Produto.

Os itens do Backlog do Produto de ordem mais alta (topo da lista) devem ser mais claros e mais detalhados que os itens de ordem mais baixa. Os itens do Backlog do Produto que irão ocupar o desenvolvimento na próxima iteração são os mais refinados.

O Time de Desenvolvimento é responsável por todas as estimativas. O Product Owner deve influenciar o Time, ajudando no entendimento e nas decisões conflituosas, mas as pessoas que irão realizar o trabalham fazem a estimativa final.

Documento Modelo:

Caso de Uso

Ciclo de Vida:	Desenvolvimento de Software
Disciplinas / Tarefas:	Requisitos → Tarefa: Identificar e Especificar Requisitos Requisitos → Tarefa: Especificar Caso de Uso
Papéis:	Time de Desenvolvimento

O documento de Caso de Uso contém as especificações do software a ser desenvolvido.

Documento Modelo:

[Projeto] - UC[Numero] - [Nome Caso de uso] v3.docx

Change Request

Ciclo de Vida:	Desenvolvimento de Software
Disciplinas / Tarefas:	Requisitos → Tarefa: Identificar e Especificar Requisitos Requisitos → Tarefa: Especificar Change Request
Papéis:	Time de Desenvolvimento

O documento de Change Request contém as especificações de alteração a serem realizadas em um software em produção.

Documento Modelo:

[Projeto] - Change Request [Número] - [Titulo].docx

Checklist do Desenvolvedor

Ciclo de Vida:	Desenvolvimento de Software
Disciplinas / Tarefas:	Desenvolvimento → Tarefa: Implementar (codificar) software
Papéis:	Time de Desenvolvimento

O Checklist do Desenvolvedor deve ser utilizado a cada fim de iteração com a finalidade de lembrar o Analista Programador de pontos que devem ser testados manualmente a cada mudança no código da funcionalidade.

PROJETO: **- DATA:**
/ / - Sprint No: **DESENVOLVEDOR:**

CHECKLIST DO DESENVOLVEDOR	(OK/N.A/FALHA)
Para sistema web:	
Verificar comportamento do campo com os diversos tipos de dados inseridos (campos numéricos, data, caracteres, tamanho)	
Alinhamento dos itens da tela	
Correção ortográfica dos títulos, labels, mensagens e botões	
Comportamentos dos controles de tela	
Campos habilitados/desabilitados no momento correto (consulta, alteração, etc)	
Combos dependentes de outros campos/dados do banco	
Verificar alto contraste e outros itens de acessibilidade	
Coerência de padrões da interface (fidelidade ao protótipo) - checklist de protótipo/layout	
Verificar comportamento de TAB e ENTER e teclas de atalho	
Verificar se os dados são carregados na tela	
Verificar se os dados são limpos da tela	
Verificar se os dados são inseridos, alterados e excluídos no banco	
Verificar regras de código	
Verificar padrões de banco (se as consultas de grande volume estão sendo paginadas, etc)	
Testar os vários perfis de acesso	
Conferir campos obrigatórios e mensagens de alerta/erro	
Conferir campos com máscaras (cpf, datas, moedas, email, cnpj etc)	
Testar nos navegadores selecionados (IE, chrome, firefox)	
verificar log de auditoria	
Verificar Roteiro de Homologação e atualizá-lo caso necessário	
verificar se o nome das páginas está no padrão	
Anotar em forma de comentário, acima da declaração do método, todos os casos de uso que utilizam um método. Exemplo: -- Rastreabilidade: InserirNome(), ExcluirNome(),.....	
Executar os testes de rastreabilidade dos métodos que são utilizados por mais de um caso de uso (item 033)	
Verificar se a regra já está implementada no AGEN	

Documento Modelo:

Código-fonte

Ciclo de Vida:	Desenvolvimento de Software
Disciplinas / Tarefas:	Desenvolvimento → Tarefa: Implementar (codificar) software Testes → Tarefa: Implementar (codificar) Testes Automatizados
Papéis:	Time de Desenvolvimento

O código-fonte é o produto principal de todo o processo de desenvolvimento de software. Pode ser composto por:

- código-fonte e interface da aplicação (em liguagem de programação definida para o projeto);
- código-fonte de testes automatizados;
- definição da estrutura de banco de dados;
- scripts, funções, procedimentos armazenados ou gatilhos em linguagem SQL.

Ele deve ser desenvolvido conforme padrões estabelecidos e mantido completo e atualizado no Repositório, de forma a suportar manutenções e evoluções da solução.

Repositório

Critérios de Aceite

Ciclo de Vida:	Desenvolvimento de Software
Disciplinas / Tarefas:	Requisitos → Tarefa: Definir Critérios de Aceite
Papéis:	Gerente de Projeto Gerente de Sustentação Product Owner Time de Desenvolvimento

O documento de Critérios de Aceite serve para descrever o comportamento esperado do sistema, servindo para o planejamento dos testes. Abaixo, um exemplo do documento de critérios de aceite de um requisito:

OBJETO	CRITÉRIOS DE ACEITE		
	AÇÃO	RESULTADO ESPERADO	AVALIAÇÃO
RF 001 - Cadastro de Gestões do TCEPR	**cenário:** cadastrar gestão **dado que:** eu cadastrar dados de uma gestão **quando:** clicar em salvar	**Então:** Sistema deve permitir salvar e exibir mensagem de sucesso.	ok
	cenário: cadastrar gestão **dado que:** eu digito gestão com data sobreposta **quando:** clicar em salvar	**Então:** Sistema não deve permitir salvar e deve exibir mensagem de erro com a descrição do problema: As datas não podem ser sobrepostas.	OK
	cenário: cadastrar gestão **dado que:** eu não preencha um campo obrigatório **quando:** clicar em salvar	**Então:** Sistema não deve permitir salvar e deve exibir mensagem de erro com a descrição e realçar o campo obrigatório deixado em branco.	OK
	cenário: editar dados de uma gestão **dado que:** eu edito dados de uma gestão **quando:** clicar em salvar	**Então:** Sistema deve permitir salvar e exibir mensagem de sucesso.	OK
	cenário: exclusão de gestão **dado que:** não há vínculos à gestão **quando:** clicar em excluir	**Então:** Sistema deve permitir excluir e exibir mensagem de sucesso.	ok
	cenário: exclusão de gestões **dado que:** existem planejamentos cadastrados **quando:** clicar em excluir	**Então:** Sistema não pode excluir e deve exibir informação de que existem planejamentos cadastrados para a gestão.	avaliação futura
	cenário: exclusão de gestões **dado que:** existem ações cadastradas **quando:** clicar em excluir	**Então:** Sistema não pode excluir e deve exibir informação de que existem ações cadastradas para a gestão.	avaliação futura
	cenário: exclusão de gestões **dado que:** existem diretrizes cadastradas **quando:** clicar em excluir	**Então:** Sistema não pode excluir e deve exibir informação de que existem diretrizes cadastradas para a Gestão.	ok

Documento Modelo:

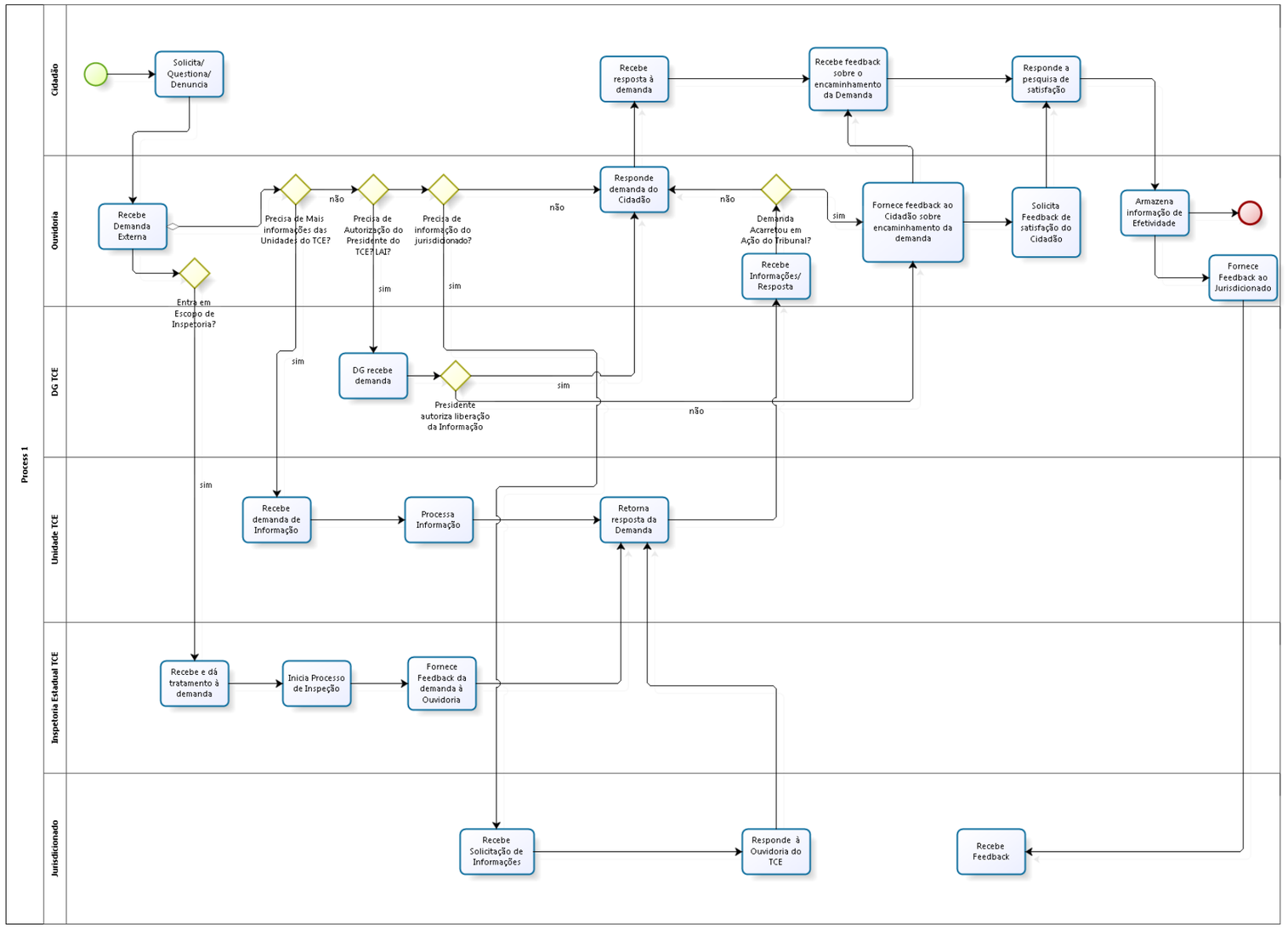
Diagrama de Negócio

Ciclo de Vida:	Análise de Demanda Desenvolvimento de Software
Disciplinas / Tarefas:	Requisitos → Tarefa: Elaborar Diagrama de Negócio
Papéis:	Analista de Demandas Gerente de Projeto Gerente de Sustentação Product Owner Time de Desenvolvimento

É um diagrama BPMN (*Business Process Model and Notation*) que representa o funcionamento do negócio, seus processos e atividades.

O diagrama serve para dar suporte ao projeto de software, uma vez que permite melhor visualização dos processos que serão automatizados no sistema a ser desenvolvido.

Exemplo de BPMN representando a Ouvidoria do TCE PR:



Documento Modelo:

Documento de Arquitetura

Ciclo de Vida:	Desenvolvimento de Software
Disciplinas / Tarefas:	Arquitetura → Tarefa: Especificar Arquitetura do Projeto
Papéis:	Arquiteto

Este documento tem por objetivo definir os padrões de arquitetura de software que serão adotados pela equipe de desenvolvimento do projeto, em função das suas características e requisitos próprios.

Documento Modelo:

DA - Arquitetura projeto - Artefato padrão.docx

Documento de Retrospectiva

Ciclo de Vida:	Desenvolvimento de Software
Disciplinas / Tarefas:	Gestão de Projetos → Tarefa: Avaliar Iteração
Papéis:	Gerente de Projeto Product Owner Scrum Master Time de Desenvolvimento Usuário (Área de Negócio)

O Documento de Retrospectiva deverá conter, no mínimo, as seguintes informações:

O QUE FOI BEM ?	O QUE PODE MELHORAR ?	AÇÕES
[Descreva o que foi bem na sprint, o que funcionou bem e deve ser mantido, seja ele um processo, documento, etc...]	[Descreva o que foi ruim e pode melhorar]	[Descreva as ações que serão feitas para melhorar]

Documento de Revisão de Arquitetura

Ciclo de Vida:	Desenvolvimento de Software
Disciplinas / Tarefas:	Arquitetura → Tarefa: Revisar arquitetura do software
Papéis:	Gerente de Projeto Product Owner Time de Desenvolvimento

O Documento de Revisão de Arquitetura irá balizar uma avaliação do projeto para verificar se o desenvolvimento está de acordo com o padrão de Arquitetura definido pelo TCEPR, ou com a Arquitetura específica proposta para o projeto (se houver), bem como validar se o desenvolvimento está de acordo com os documentos de Arquitetura escritos para cada requisito, quando existirem.

Documento Modelo:

DA - Revisão de arquitetura - Artefato padrão.docx

Documento de Transferência de Tecnologia

Ciclo de Vida:	Transferência de Tecnologia
Disciplinas / Tarefas:	Transferência de Tecnologia → Tarefa: Elaborar ou Atualizar Documento de Transferência de Tecnologia Transferência de Tecnologia → Tarefa: Aceitar Transferência de Tecnologia
Papéis:	Gerente de Projeto Product Owner Time de Desenvolvimento Gerente de Sustentação Gerente de Infraestrutura

O Documento de Transferência de Tecnologia possui o objetivo de permitir que outros, que não o criador do projeto, possam manter o sistema operacional, construir novas funcionalidades e modificar o sistema transferido.

Documento Modelo:

TT_Transferencia_Tecnologia_Modelo.doc

Documento de Validação de Interface

Ciclo de Vida:	Desenvolvimento de Software
Disciplinas / Tarefas:	
Papéis:	Gerente de Projeto Product Owner Time de Desenvolvimento

O documento de validação de interface deve ser criado para verificar os padrões e a usabilidade do software.

Documento Modelo:

(Projeto) - (Interface) - Checklist de Interface e Usabilidade.xlsx

Documento de Validação de Qualidade

Ciclo de Vida:	Desenvolvimento de Software
Disciplinas / Tarefas:	Qualidade → Tarefa: Validar qualidade das entregas e dos testes
Papéis:	Gerente de Projeto Product Owner Time de Desenvolvimento

O Documento de Validação de Qualidade serve para validar a qualidade de entrega realizada antes que o sistema seja disponibilizado para o usuário.

Incluir como anexo:

Este documento pode ser um E-mail contendo as seguintes informações:

- A entrega passou por todos os testes previstos no Plano de testes?
  ( ) sim ( ) não

- Os bugs encontrados nos testes foram corrigidos?
  ( ) sim ( ) não

- O código passou satisfatoriamente pelo padrão especificado no SonarQube?
  ( ) sim ( ) não

- O Sistema entregue está pronto para validação do Usuário?
  ( ) sim ( ) não

Documento Modelo:

Documento de Validação de Segurança

Ciclo de Vida:	Desenvolvimento de Software
Disciplinas / Tarefas:	Segurança → Tarefa: Validar segurança do software
Papéis:	Gerente de Projeto Product Owner Time de Desenvolvimento

O Documento de Validação de Segurança serve para validar se os Padrões de Segurança foram seguidos.

Documento Modelo:

Documento de Validação de Segurança.docx

História de Usuário

Ciclo de Vida:	Desenvolvimento de Software
Disciplinas / Tarefas:	Requisitos → Tarefa: Identificar e Especificar Requisitos Requisitos → Tarefa: Especificar História de Usuário
Papéis:	Time de Desenvolvimento

A História de Usuário é uma forma de descrever os requisitos funcionais e relacioná-los com seu principal ator.

Repositório

Incremento de Software

Ciclo de Vida:	Desenvolvimento de Software
Disciplinas / Tarefas:	Desenvolvimento → Tarefa: Entregar software e roteiro de publicação
Papéis:	Gerente de Projeto Product Owner Scrum Master Time de Desenvolvimento

O incremento de software é a soma de todos os itens do Backlog do Produto completados durante a iteração/Sprint.

Ao final da iteração um novo incremento deve estar "Pronto", o que significa que deve estar na condição utilizável e atender a definição de "Pronto" do Time Scrum, independente do Product Owner decidir por liberá-lo para uso.

Documento Modelo:

Indicadores de Qualidade e Gestão

Ciclo de Vida:	Desenvolvimento de Software
Disciplinas / Tarefas:	Gestão de Projetos → Tarefa: Apurar Indicadores de Qualidade e Gestão Gestão de Projetos → Tarefa: Avaliar Indicadores de Qualidade e Gestão
Papéis:	Gerente de Projeto Product Owner

Os Indicadores de Qualidade e Gestão são importantes no processo de desenvolvimento para o aperfeiçoamento contínuo dos processos.

Abordagem do Indicadores de Qualidade e Gestão

- Verificar a exatidão das estimativas
- Tamanho do software entregue
- Mudança de escopo do software durante o desenvolvimento
- Índice de Qualidade: Número de Defeitos / Tamanho do Software
- Nível de satisfação do cliente
- Qualidade do produto liberado
- Produtividade
- Retrabalho

Como Medir?

- Buscou-se uma prática bem sedimentada na gestão de software: Análise por Pontos de Função.
- Usada para estimar, mas pode ser uma forte auxiliar na gestão de qualidade.

Indicadores de Qualidade e Gestão

- IAEP - Índice de Aderência do Escopo do Projeto

Objetivo: Mensurar a variação de esforço utilizado para o desenvolvimento de funcionalidades previstas e não previstas no projeto.
Quanto se gastou com o previsto e quanto se gastou em itens não previstos?

Fórmula:
$IAEP=ET/EP$
onde:
ET = Esforço Total no Previsto no Escopo + Esforço do Não Previsto
EP = Esforço Total no Previsto no Escopo

Índice de Referência = 1 ou 100 %

- IQPDS - Índice de Qualidade do Processo de Desenvolvimento de Software

Objetivo: Mensurar a quantidade de defeitos (bugs) por ponto de função do software em produção.

Fórmula:
$IQPDS=DT/SP$
onde:
DT = Quantidade Total de Defeitos (em Unidade)
SP = Tamanho do Software em Produção (em Pontos de Função)

Índice de Referência = 0 defeitos / PF

- IRPDS - Índice de Retrabalho do Processo de Desenvolvimento de Software

Objetivo: Mensurar a variação do esforço total de desenvolvimento em relação ao tamanho do software entregue.
Quanto de retrabalho houve? Por analogia: quantos metros de piso tive que quebrar e refazer para ter um piso final?

Fórmula:
$IRPDS=ET/SP$
onde:
ET = Esforço Total de Desenvolvimento (em Pontos de Função - PF)
SP = Tamanho do Software em Produção (em PF)

Índice de Referência = 1 ou 100 %

- CDSW - Custo de Desenvolvimento de Software

Objetivo: Mensurar o custo de desenvolvimento (em homem-hora) por ponto de função (PF) do software entregue.

Fórmula:
$CDSW = CT/SP$
onde:
CT = Custo Total de Desenvolvimento (em Homem/Hora)
SP = Tamanho do Software em Produção (em Pontos de Função)

Índice de Referência = média histórica (HH / PF)

- ISC - Índice de Satisfação do Cliente

Objetivo: Mensurar a satisfação do cliente através de pesquisa.

Pesquisa de Satisfação do Cliente
Nome do Projeto: XXX
Período: 00/00/0000 a 00/00/0000
1. Avalie o processo de desenvolvimento de software da DTI, especificamente com relação ao projeto e ao período identificados acima, quanto:
   1.1. à CAPACIDADE DE ENTREGA (percepção quanto a demandas não atendidas, solicitações na fila de espera etc.).
   1.2. ao CUMPRIMENTO DE PRAZOS (percepção quanto a entrega nos prazos acordados).
   1.3. à QUALIDADE DO PRODUTO (percepção quanto a bugs e defeitos detectados após entrega do software para homologação/produção).
   1.4. à INTERFACE E USABILIDADE (percepção quanto a facilidade de uso dos sistemas).
   1.5. aos TREINAMENTOS E ORIENTAÇÕES (percepção quanto a qualidade dos treinamentos e orientações realizados).
   1.6. à EQUIPE (sua percepção quanto ao tamanho, capacidade técnica, proatividade etc.).
   1.7. ao ENTENDIMENTO DO NEGÓCIO (sua percepção quanto à capacidade de compreensão do negócio pela equipe da área de TI).
   1.8. à COMUNICAÇÃO (sua percepção quanto à comunicação entre as equipes).
2. Avalie, de uma forma geral, o seu grau de SATISFAÇÃO com relação ao processo de desenvolvimento de software referente ao projeto e período identificados acima.

Fórmula:
ISC = média da nota geral dos questionários

Índice de Referência >= 4,0

- IQESW - Índice de Qualidade da Estimativa de Software

Objetivo: Mensurar a variação entre a estimativa inicial de tamanho de software e o tamanho do software entregue.
Que tamanho de sistema foi previsto e que tamanho foi entregue?

Fórmula:
$IQESW = ES/SP$
onde:
ES = Estimativa Inicial de Tamanho do Software (em Pontos de Função PF)
SP = Tamanho do Software em Produção (em PF)

Índice de Referência = 1 ou 100 %

- ICMI – Índice de Custo de Medição dos Indicadores

Objetivo: Mensurar o índice de custo de medição dos indicadores.
Quanto custa o controle?

Fórmula:
$ICMI=CM/CT$
onde:
CM = Custo de Medição dos Indicadores de Qualidade e Gestão (em Homem/Hora)
CT = Custo Total de Desenvolvimento (em Homem/Hora)

Índice de Referência = média histórica (%)

Documento Modelo:

Indicadores de Qualidade e Gestão - Projeto MODELO.xlsx

Item de Trabalho do Repositório

Ciclo de Vida:	Desenvolvimento de Software
Disciplinas / Tarefas:	Gestão de Projetos → Tarefa: Definir Escopo do Projeto (priorização e estimativa de esforço) Gestão de Projetos → Tarefa: Iniciar Iteração (planejamento)
Papéis:	Gerente de Projeto Product Owner Scrum Master Time de Desenvolvimento

Um Item de Trabalho do Repositório (ou Work Item) é utilizado para descrever itens de escopo e atividades que serão executadas pelo time de desenvolvimento na execução do projeto.

O repositório utilizado atualmente pelo TCEPR é o Team Foundation Server (TFS). Os itens de trabalho devem ser registrados na Team Project Collection do projeto e podem ser do tipo:

- Funcionalidade (ou *Feature*);
- Backlog do Produto (ou *Product Backlog Item*);
- Defeito (ou *Bug*);
- Impedimento (ou *Impediment*);
- Tarefa (ou *Task*).

Referência:

- *Scrum process work item types and workflow*: https://www.visualstudio.com/en-us/docs/work/guidance/scrum-process-workflow

Documento Modelo:

Ordem de Serviço

Ciclo de Vida:	Início do Projeto ou Sustentação Desenvolvimento de Software Término do Projeto ou Sustentação
Disciplinas / Tarefas:	Gestão de Projetos → Tarefa: Emitir Ordem de Serviço
Papéis:	Gerente de Projeto Gerente de Sustentação

A Ordem de Serviço é o instrumento utilizado para solicitação de equipe técnica terceirizada para o desenvolvimento de um novo projeto de software ou para sustentação e manutenção em aplicação existentes, quando houver contrato de serviços de desenvolvimento, sustentação e manutenção de sistemas vigente.

Documento Modelo:

Ordem de Serviço.xlsx

Ordem de Serviço - Aditivo.xlsx

Plano de Teste

Ciclo de Vida:	Desenvolvimento de Software
Disciplinas / Tarefas:	Testes → Tarefa: Elaborar Plano de Teste
Papéis:	Time de Desenvolvimento

O Plano de Teste é um desdobramento dos Critérios de Aceite, sendo que este documento traz mais informações da análise de testes. O Analista Programador deverá complementar os dados de forma a decidir como o teste deverá ser projetado e executado no momento da codificação.

TCEPR TRIBUNAL DE CONTAS DO ESTADO DO PARANÁ

Plano de teste

UC SXX

PROJETO	Projeto XXX

OBJETO	Testes					
	AÇÃO	RESULTADO ESPERADO	TIPO DE TESTE	TRATAMENTO	SCRIPT DE INSERÇÃO/ EXCLUSÃO DE DADOS	EXECUÇÃO
RF 001 -regra de calculo de média	**cenário:** calculo de media aritmetica **dado que:** eu insira 10 e 2 **quando:** o agen acionar a regra	**Então:** o resultado gravado deve ser 6	COMPORTAMENTO DE REGRA	AUTOMATIZADO	SIM	
	cenário: cadastrar gestão **dado que:** eu cadastrar dados de uma gestão **quando:** clicar em salvar	**Então:** Sistema deve gravar dados na tabela xyz	COMPORTAMENTO DE BANCO	AUTOMATIZADO	SIM	

Documento Modelo:

Pré-Projeto

Ciclo de Vida:	Análise de Demanda
Disciplinas / Tarefas:	Gestão de Demandas → Tarefa: Elaborar Pré-Projeto
Papéis:	Analista de Demandas

O Pré-Projeto é um documento que identifica e define os requisitos do projeto proposto, com base na Proposta de Projeto, de forma a delimitar o escopo do projeto e a estimar a duração e o custo do projeto.

Documento Modelo:

Projeto Preliminar - [NomeProjeto].docx

Proposta de Projeto

Ciclo de Vida:	Análise de Demanda
Disciplinas / Tarefas:	Gestão de Demandas → Tarefa: Elaborar Proposta de Projeto
Papéis:	Analista de Demandas

A Proposta de Projeto é um documento que define de forma clara e objetiva as necessidades e objetivos do projeto proposto.

Documento Modelo:

Proposta de Projeto - [NomeProjeto].docx

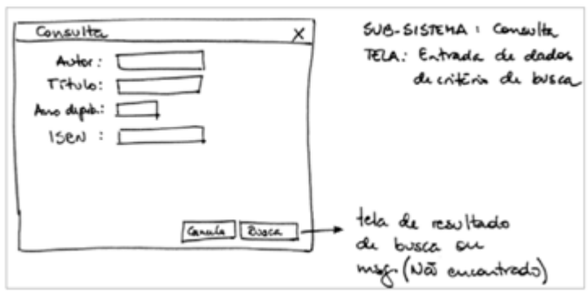
Protótipo

Ciclo de Vida:	Análise de Demanda Desenvolvimento de Software
Disciplinas / Tarefas:	Requisitos → Tarefa: Desenhar Protótipo
Papéis:	Analista de Demandas Gerente de Projeto Gerente de Sustentação Product Owner Time de Desenvolvimento

O Protótipo consiste em um desenho que permite aos futuros usuários do sistema visualizar as funcionalidades e a interface do software.

Dependendo do objetivo, os protótipos podem ser de baixa ou de alta fidelidade.

PROTÓTIPO DE BAIXA FIDELIDADE (WIREFRAME)



PROTÓTIPO DE ALTA FIDELIDADE

Documento Modelo:

Não há um modelo específico. Utilize as ferramentas de prototipação disponíveis.

Regras de Negócio

Ciclo de Vida:	Desenvolvimento de Software
Disciplinas / Tarefas:	Requisitos → Tarefa: Especificar Caso de Uso Requisitos → Tarefa: Especificar Change Request
Papéis:	Time de Desenvolvimento

O documento de Regras de Negócio contém as especificações das regras aplicadas ao software. As regras contidas neste documento são referenciadas pelos documentos de Caso de Uso e de Change Request.

Documento Modelo:

[Projeto] - Regras de Negócio.docx

Relatório de Execução dos Testes Automatizados

Ciclo de Vida:	Desenvolvimento de Software
Disciplinas / Tarefas:	Testes → Tarefa: Executar Testes Automatizados
Papéis:	Gerente de Projeto Gerente de Sustentação Product Owner Time de Desenvolvimento

O Relatório de execução dos testes é gerado automaticamente pela plataforma de desenvolvimento, quando um grupo de testes é executado.

O relatório deve ser registrado pelo Analista Programador e armazenado no repositório do projeto.

Documento Modelo:

Relatório de Execução dos Testes Manuais (Critérios de Aceite e Exploratórios)

Ciclo de Vida:	Desenvolvimento de Software
Disciplinas / Tarefas:	Testes → Tarefa: Executar Testes Manuais (Critérios de Aceite e Exploratórios)
Papéis:	Gerente de Projeto Gerente de Sustentação Product Owner Time de Desenvolvimento Usuário (Área de Negócio)

O Relatório de testes exploratórios é registrado manualmente e deve conter os cenários executados e as evidências dos problemas encontrados.

BUG / MELHORIA	**DATA**	DD/MM/AAAA
	Quem Testou	ANA PAULA
	Classificação	BUG
	Descrição detalhada do Problema	O campo código de controle está permitindo digitar qualquer caracter. Deveria permitir somente números e limitar a 9 caracteres.
	Sistema/Módulo	SIAP Pensão
CENÁRIO	**Página**	http://www..... siapPensaoCadastrarTipoDependente.aspx
	Ambiente	Homologação
	Usuário Logado	cpf/cnpj: 82038198934
	Nome da Entidade	Paranaprevidência
	Passos realizados	Na tela de Cadastrar Novo Tipo de Dependente: - preenchi o campo Código de Controle com o valor "@123ABCDEFGadfsdfdfadfadfadfad"; - Ao tentar salvar o sistema exibe a mensagem "Código de controle inválido"; - O correto seria ter uma máscara no campo permitindo apenas números e máximo 9 dígitos.

Documento Modelo:

Roteiro de Publicação

Ciclo de Vida:	Desenvolvimento de Software
Disciplinas / Tarefas:	Desenvolvimento → Tarefa: Entregar software e roteiro de publicação
Papéis:	Time de Desenvolvimento

O Roteiro de Publicação deverá conter todas as informações necessárias para orientar na correta publicação do sistema ao término de cada iteração de desenvolvimento.

Documento Modelo:

DA - Sprint XX - Roteiro Publicação.docx

Solicitação de Serviço

Ciclo de Vida:	Análise de Demanda
Disciplinas / Tarefas:	-
Papéis:	Usuário (Área de Negócio)

A Solicitação de Serviço é o principal instrumento de comunicação do usuário com a área de Tecnologia da Informação.

Sempre que for identificado um problema ou necessidade de negócio que requeira modificação ou criação de software, o usuário deve registrar a demanda no Sistema de Solicitação de Serviços.

Sistema de Solicitação de Serviços

Termo de Abertura de Projeto

Ciclo de Vida:	Início do Projeto ou Sustentação
Disciplinas / Tarefas:	Gestão de Projetos → Tarefa: Realizar Reunião de Abertura de Projeto
Papéis:	Gerente de Projeto Product Owner Scrum Master Usuário (Área de Negócio)

Este documento tem como objetivo autorizar formalmente o início de um projeto e contém informações necessárias para o entendimento do projeto, fornecendo uma visão macro do produto a ser desenvolvido.

Documento Modelo:

[Projeto] - Termo de Abertura de Projeto.docx

Termo de Encerramento de Projeto

Ciclo de Vida:	Término do Projeto ou Sustentação
Disciplinas / Tarefas:	Gestão de Projetos → Tarefa: Realizar Reunião de Encerramento de Projeto
Papéis:	Gerente de Projeto Product Owner Scrum Master Usuário (Área de Negócio)

O Termo de Encerramento de Projeto tem por finalidade atestar a conclusão do projeto.

Documento Modelo:

[Projeto] - Termo de Encerramento de Projeto.docx

Termo de Recebimento de Serviço

Ciclo de Vida:	Desenvolvimento de Software Término do Projeto ou Sustentação
Disciplinas / Tarefas:	Gestão de Projetos → Tarefa: Receber Serviços
Papéis:	Gerente de Projeto Gerente de Sustentação

O Termo de Recebimento de Serviço é o instrumento utilizado para o cálculo da remuneração devida pelos serviços prestados no âmbito de cada Ordem de Serviço, considerando o cumprimento dos níveis mínimos de serviço definidos em instrumento contratual.

Existem 2 modelos de Termo de Recebimento de Serviço, sendo um para projetos de desenvolvimento de sistemas e outro para sustentação e manutenção de sistemas.

Documento Modelo:

Termo de Recebimento Definitivo - Projeto de Desenvolvimento de Sistemas.docx

Termo de Recebimento Definitivo - Sustentação e Manutenção de Sistemas.docx

5. PAPÉIS

Papéis identificam as pessoas envolvidas no processo de desenvolvimento de software. Os papéis não representam responsabilidades individuais sobre as atividades ou artefatos, mas sim atividades que as pessoas podem realizar quando trabalham em conjunto, em uma perspectiva de colaboração. Executar um ou mais papéis pode ajudar as equipes a exprimir diferentes pontos de vista para criar uma solução. Esta perspectiva sobre os papéis fortalece a nova geração de soluções, mais focada na interação entre pessoas.

Os papéis estabelecidos no PDS TCEPR são:

- Administrador de Banco de Dados
- Administrador de Infraestrutura
- Analista de Demandas
- Analista Programador
- Arquiteto
- Comitê de Tecnologia da Informação
- Diretor de Tecnologia da Informação
- Diretor Geral
- Gerente de Infraestrutura
- Gerente de Projeto
- Gerente de Sustentação
- Product Owner
- Scrum Master
- Time de Desenvolvimento
- Usuário (Área de Negócio)

Administrador de Banco de Dados

O Administrador de Banco de Dados, ou *Database Administrator* (DBA), é o responsável pelo gerenciamento do banco de dados.

Atribuições:	• Manter o banco de dados corporativo operacional. • Monitorar o desempenho da resposta do banco de dados para as aplicações dentro dos parâmetros acordados pela organização. • Manter a integridade das informações disponíveis no banco de dados. • Apoiar as equipes de desenvolvimento e de arquitetura no processo da modelagem de dados. • Zelar pela adoção e uso de melhores práticas e padrões de banco de dados pelas equipes de desenvolvimento e arquitetura. • Propor melhorias nos processos. • Propor e executar processos de atualização tecnológica de banco de dados. • Acompanhar os contratos e licitações de projetos da área. • Elaborar os termos de referências para aquisições da área. • Elaborar e manter relatórios de informações dos processos relativos a sua área. • Propor melhorias na infraestrutura de banco de dados. • Gerenciar o processo de cópias de segurança das bases de dados. • Manter a infraestrutura de servidores de banco de dados.
Tarefas:	•

Administrador de Infraestrutura

O Administrador de Infraestrutura é o responsável pelo gerenciamento e a manutenção da infraestrutura computacional da organização.

Atribuições:	• Manter a infraestrutura de Tecnologia de Informação e Comunicação (TIC) disponível dentro dos parâmetros acordados pela organização. • Prospectar novas tecnologias disponíveis. • Gerenciar o processo de cópias de segurança da organização. • Manter os serviços de infraestrutura como e-mail, comunicação unificada, entre outros. • Manter a infraestrutura de rede de dados da organização. • Elaborar os termos de referências para aquisições de infraestrutura. • Apoiar as equipes de desenvolvimento e arquitetura no uso dos recursos de infraestrutura disponíveis. • Zelar pela adoção e uso de melhores práticas e padrões de TIC. • Acompanhar os contratos e licitações de projetos da área. • Gerenciar o processo de cópias de segurança das informações da organização. • Monitorar a infraestrutura de TIC. • Avaliar a capacidade dos recursos computacionais disponível e propor aquisições e melhorias. • Propor melhorias nos processos.
Tarefas:	• Implantação → Tarefa: Publicar software no Ambiente de Homologação • Implantação → Tarefa: Publicar software no Ambiente de Produção

Analista de Demandas

O Analista de Demandas é o responsável por identificar e analisar problemas ou necessidades das áreas de negócio e avaliar a necessidade de criação ou modificação de software, sistemas ou serviços, assim como aquisição de solução de TI.

Atribuições:	• Identificar e analisar problemas ou necessidades das áreas de negócio, propondo soluções que envolvam a criação, modificação ou aquisição (pesquisa, prospecção e orçamentação) de soluções de TI (software, sistemas ou serviços). • Avaliar a complexidade e o esforço necessário para atender as demandas das áreas de negócio, definindo se a demanda se enquadra como sustentação (pequena manutenção evolutiva) ou se constitui necessidade a ser enfrentada mediante instituição de projeto. • Elaborar, em conjunto com a(s) área(s) de negócio, Proposta de Projeto para atendimento das demandas que não se enquadram nos critérios de sustentação. • Elaborar, em conjunto com a(s) área(s) de negócio, Pré-Projeto da solução de TI após a aprovação da Proposta de Projeto pelo Comitê de TI.
Tarefas:	• Gestão de Demandas → Tarefa: Elaborar Proposta de Projeto • Gestão de Demandas → Tarefa: Elaborar Pré-Projeto • Gestão de Projetos → Tarefa: Elaborar Ata de Reunião • Requisitos → Tarefa: Identificar e Especificar Requisitos • Requisitos → Tarefa: Desenhar Protótipo • Requisitos → Tarefa: Elaborar Diagrama de Negócio

Analista Programador

O Analista Programador é o profissional que compõe o Time de Desenvolvimento. Ele é o principal responsável, junto com os outros integrantes do Time de Desenvolvimento, por desenvolver os artefatos e produtos planejados, executando todas as tarefas que forem necessárias para alcançar os objetivos.

Atribuições:	• Identificar e especificar requisitos funcionais, não funcionais e requisitos de acessibilidade, inclusive participando de reuniões com o usuário final, se necessário. • Apoiar o papel de Product Owner (PO) na definição e especificação de requisitos (refinamento dos itens do backlog do produto). • Especificar Casos de Uso e Histórias de Usuário. • Especificar solicitação de mudanças (Change Request). • Elaborar matriz de rastreabilidade. • Desenhar protótipos e elaborar diagramas de negócio. • Especificar Arquitetura. • Elaborar análise e projeto de software orientado a objetos. • Elaborar modelagem de dados (modelo lógico e físico) e modelo de domínio. • Aplicar os processos de desenvolvimento seguro, de acordo com os padrões definidos. • Programar a solução através de codificação de software. • Gerar pacotes de software, scripts de banco de dados e roteiro de publicação. • Controle de versões de código-fonte de software e geração de *builds*. • Realizar a transferência de tecnologia através do repasse do conhecimento acerca do projeto para a equipe do TCEPR, através da elaboração de artefatos e reuniões técnicas. • Implantar, configurar e fazer publicação (deploy) do software no ambiente de desenvolvimento, testes, homologação e produção dos sistemas desenvolvidos nestes ambientes, utilizando entregas e integração contínuas. • Elaborar casos de teste, especificar Casos e Planos de Testes em BDD (Behavior Driven Development – Desenvolvimento Orientado a Comportamento) e TDD (Test-Driven Development – Desenvolvimento Orientado a Testes) baseado em Histórias de Usuários, elaborar plano de testes, automatizar testes, executar testes automatizados, testes manuais, testes exploratórios, testes unitários, de integração, funcional, aceitação/estória, de carga, de desempenho, de vulnerabilidade, de usabilidade e de acessibilidade. • Participar ativamente nas reuniões de planejamento, diárias, de entrega e de retrospectiva, ou em quaisquer práticas inerentes ao desenvolvimento. • Aplicar os requisitos de qualidade e segurança, conforme padrões definidos, nos produtos entregues. • Apurar indicadores de qualidade e gestão, aplicando métricas de medição no software produzido, conforme Manual de Contagem de Pontos de Função do TCEPR.
Tarefas:	• Gestão de Projetos → Tarefa: Iniciar Iteração (planejamento) • Gestão de Projetos → Tarefa: Realizar Reunião Diária • Gestão de Projetos → Tarefa: Elaborar Ata de Reunião • Gestão de Projetos → Tarefa: Encerrar Iteração (entrega/revisão) • Gestão de Projetos → Tarefa: Avaliar Iteração (retrospectiva) • Requisitos → Tarefa: Identificar e Especificar Requisitos

	• Requisitos → Tarefa: Definir Critérios de Aceite • Requisitos → Tarefa: Especificar Caso de Uso • Requisitos → Tarefa: Especificar História de Usuário • Requisitos → Tarefa: Especificar Change Request • Requisitos → Tarefa: Desenhar Protótipo • Requisitos → Tarefa: Elaborar Diagrama de Negócio • Desenvolvimento → Tarefa: Implementar (codificar) software • Desenvolvimento → Tarefa: Entregar software e roteiro de publicação • Testes → Tarefa: Elaborar Plano de Teste • Testes → Tarefa: Implementar (codificar) Testes Automatizados • Testes → Tarefa: Executar Testes Automatizados • Testes → Tarefa: Executar Testes Manuais (Critérios de Aceite e Exploratórios) • Qualidade → Tarefa: Validar qualidade das entregas e dos testes • Segurança → Tarefa: Validar segurança do software • Arquitetura → Tarefa: Especificar Arquitetura do Projeto • Arquitetura → Tarefa: Revisar arquitetura do software • Implantação → Tarefa: Publicar software no Ambiente de Desenvolvimento ou Testes • Implantação → Tarefa: Publicar software no Ambiente de Homologação • Implantação → Tarefa: Publicar software no Ambiente de Produção • Transferência de Tecnologia → Tarefa: Elaborar ou Atualizar Documento de Transferência de Tecnologia

Arquiteto

O Arquiteto deve manter as coerências de integração, de codificação e de implementação, além de delimitar os domínios dos sistemas.

Atribuições:	• Orientar as escolhas durante o desenvolvimento em:
	  ○ padrões para desenvolvimento de aplicações;
	  ○ definição/criação de um framework para a aplicação.
	• Indicar pontos potenciais de reutilização de blocos de construção, na organização ou dentro da aplicação, para:
	  ○ abordagem mais abrangente;
	  ○ adoção de um design de componentização;
	  ○ disseminar conhecimento entre aplicações da organização.

Tarefas:	• Tarefa: Especificar Arquitetura do Projeto

Comitê de Tecnologia da Informação

O Comitê de Tecnologia da Informação é o responsável por garantir adequada governança corporativa na área de Tecnologia da Informação, estabelecer políticas e diretrizes estratégicas de segurança, prioridades, demandas e investimentos da área, alinhado ao Planejamento Estratégico do TCEPR.

Atribuições:	• Propor o Planejamento Estratégico de Tecnologia de Informação para promover o alinhamento das ações da área às diretrizes estratégicas do Tribunal. • Propor prioridades de execução de projetos, considerando as demandas consolidadas e apresentadas pela Diretoria de Tecnologia da Informação. • Propor o Plano de Ações e Investimentos, acompanhar o desenvolvimento e a implantação dos respectivos projetos. • Propor a Política de Segurança da Informação e Comunicações, bem como demais normas correlatas e encaminhar à Presidência do Tribunal para sua formalização. • Dirimir dúvidas e deliberar sobre questões não contempladas na Política de Segurança da Informação e Comunicações e demais normas correlatas. • Avaliar pedidos de novas aquisições ou contratações relacionadas à área de Tecnologia da Informação. • Realizar, a cada dois anos, ou em prazo menor, quando grandes mudanças na área tecnológica, organizacional e legal forem constatadas, a revisão do Planejamento Estratégico de Tecnologia da Informação para promover o alinhamento das ações da área às diretrizes estratégicas do Tribunal.

Diretor de Tecnologia da Informação

O Diretor de Tecnologia da Informação é o responsável por gerir a área de TI do TCEPR.

Atribuições:	• Zelar pelo bom andamento dos serviços, assim como pelo desenvolvimento e desempenho do seu pessoal.

Diretor Geral

O Diretor Geral é o responsável por planejar e coordenar as ações conjuntas de todas as Diretorias do TCEPR.

Atribuições:	• Zelar pelo bom andamento dos serviços do TCEPR.

Gerente de Infraestrutura

O Gerente de Infraestrutura é o responsável pela organização e gestão das áreas de atendimento, banco de dados e suporte à infraestrutura computacional da organização.

Atribuições:	• Acompanhar e designar a fiscalização dos contratos de sua área. • Apoio à direção da unidade em demandas que envolvam sua área. • Liderança das equipes com foco no melhor desempenho e crescimento pessoal. • Gerenciar os processos de atendimento, suporte, backup, continuidade, segurança e banco de dados. • Monitorar e dar visibilidade aos processos da área. • Gerenciar projetos da área. • Avaliar a aderência as melhores práticas de mercado. • Gerenciar e motivar o processo de inovação constante na área. • Gerenciar programa de aperfeiçoamento constante das equipes. • Valorizar o trabalho das equipes. • Gerenciar os contratos com terceiros. • Apoiar e gerenciar os processos de aquisição da área.
Tarefas:	• Transferência de Tecnologia → Tarefa: Aceitar Transferência de Tecnologia

Gerente de Projeto

O Gerente de Projeto é o responsável por garantir que o projeto será concluído e os objetivos sejam alcançados. Ele também é o responsável por acompanhar o cronograma do projeto, mediante o planejamento e a alocação adequada de recursos, devendo prestar contas periodicamente do andamento do projeto.

Sua principal atribuição é evitar que as falhas inerentes aos processos aconteçam. O gerente de projeto deve ser capaz de prever as dificuldades e agir preventivamente assegurando o bom andamento dos trabalhos.

Atribuições:	• Garantir o registro das atividades no padrão e meio eletrônico vigente, constando no mínimo a descrição de cada atividade, o responsável pela execução, a quantidade de tempo estimada, assim como o tempo efetivamente empregado. • Registrar as falhas identificadas pelo usuário (bugs) no documento de critérios de aceite ou ainda no sistema de solicitação de serviços vigente a época. • Realizar periodicamente o planejamento de atividades e a respectiva revisão após o decurso de cada período planejado, o qual não deverá ser superior ao período máximo de planejamento definido para cada projeto, observando o limite de 30 dias. • Ao final de cada período planejado, deverá apurar os indicadores de qualidade e gestão do projeto, definidos no processo de trabalho da área de Desenvolvimento, e realizar a análise crítica dos resultados, propondo e executando ações preventivas, corretivas e/ou de melhoria para o projeto e/ou para o processo de trabalho. • Deverá reportar ao seu gestor eventuais necessidades de pessoal ou de outros recursos, bem como disponibilizar pessoal ou recursos com previsão de não serem integralmente aplicados ao projeto, ainda que por um dado período.
Tarefas:	• Gestão de Projetos → Tarefa: Realizar Reunião de Abertura de Projeto • Gestão de Projetos → Tarefa: Definir Escopo do Projeto (priorização e estimativa de esforço) • Gestão de Projetos → Tarefa: Emitir Ordem de Serviço • Gestão de Projetos → Tarefa: Iniciar Iteração (planejamento) • Gestão de Projetos → Tarefa: Realizar Reunião Diária • Gestão de Projetos → Tarefa: Elaborar Ata de Reunião • Gestão de Projetos → Tarefa: Encerrar Iteração (entrega/revisão) • Gestão de Projetos → Tarefa: Avaliar Iteração (retrospectiva) • Gestão de Projetos → Tarefa: Apurar Indicadores de Qualidade e Gestão • Gestão de Projetos → Tarefa: Avaliar Indicadores de Qualidade e Gestão • Gestão de Projetos → Tarefa: Receber Serviços • Gestão de Projetos → Tarefa: Realizar Reunião de Encerramento de Projeto • Requisitos → Tarefa: Identificar e Especificar Requisitos • Requisitos → Tarefa: Definir Critérios de Aceite • Requisitos → Tarefa: Desenhar Protótipo • Requisitos → Tarefa: Elaborar Diagrama de Negócio • Testes → Tarefa: Executar Testes Automatizados • Testes → Tarefa: Executar Testes Manuais (Critérios de Aceite e Exploratórios) • Qualidade → Tarefa: Validar qualidade das entregas e dos testes • Segurança → Tarefa: Validar segurança do software • Arquitetura → Tarefa: Revisar arquitetura do software • Homologação → Tarefa: Homologar software

	• Transferência de Tecnologia → Tarefa: Elaborar ou Atualizar Documento de Transferência de Tecnologia

Gerente de Sustentação

O Gerente de Sustentação é o responsável por gerir a Área de Sustentação de Sistemas.

A sustentação de sistemas consiste na gestão corretiva, adaptativa e evolutiva dos sistemas de TI utilizados no dia a dia de trabalho do TCEPR, além de garantir a disponibilidade dos serviços e sistemas legados.

Atribuições:	• Controlar o prazo de atendimento e a qualidade na execução das demandas, visando manter o acordo de nível de serviço. • Avaliar continuamente sua equipe, com feedbacks construtivos, e foco no desempenho individual. • Auxiliar o Diretor da DTI nas demandas específicas à sua área e naquelas da unidade, quando couber. • Gerenciar a equipe de sustentação discriminando todas as aplicações a serem suportadas. • Promover a distribuição equitativa de atividades. • Propor melhorias nas aplicações, de modo a mitigar o risco de potenciais falhas. • Gerar indicadores dos serviços de sustentação e das aplicações. • Apresentar à Direção os sistemas e serviços que possam/devam ser objeto de evolução ou descontinuidade.
Tarefas:	• Gestão de Projetos → Tarefa: Emitir Ordem de Serviço • Gestão de Projetos → Tarefa: Receber Serviços • Transferência de Tecnologia → Tarefa: Aceitar Transferência de Tecnologia

Product Owner

O Product Owner (PO) representa os interesses de todos os envolvidos (stakeholders), define as funcionalidades do produto e prioriza os itens de Product Backlog. Ele é uma pessoa e não um comitê.

É a única pessoa responsável por gerenciar o Backlog do Produto.

Atribuições:	• Definir as funcionalidades do produto (Backlog do Produto). • Priorizar as funcionalidades de acordo com o valor de negócio. • Ajustar funcionalidades e prioridades a cada Sprint, conforme necessário. • Garantir que o Backlog do Produto seja visível, transparente e claro para todos. • Garantir que o Time de Desenvolvimento entenda os itens do Backlog do Produto no nível necessário. • Decidir a data de liberação e conteúdo do Release. • Aceitar ou Rejeitar os resultados de trabalho.
Tarefas:	• Gestão de Projetos → Tarefa: Realizar Reunião de Abertura de Projeto • Gestão de Projetos → Tarefa: Definir Escopo do Projeto (priorização e estimativa de esforço) • Gestão de Projetos → Tarefa: Iniciar Iteração (planejamento) • Gestão de Projetos → Tarefa: Realizar Reunião Diária • Gestão de Projetos → Tarefa: Elaborar Ata de Reunião • Gestão de Projetos → Tarefa: Encerrar Iteração (entrega/revisão) • Gestão de Projetos → Tarefa: Avaliar Iteração (retrospectiva) • Gestão de Projetos → Tarefa: Apurar Indicadores de Qualidade e Gestão • Gestão de Projetos → Tarefa: Avaliar Indicadores de Qualidade e Gestão • Gestão de Projetos → Tarefa: Realizar Reunião de Encerramento de Projeto • Requisitos → Tarefa: Identificar e Especificar Requisitos • Requisitos → Tarefa: Definir Critérios de Aceite • Requisitos → Tarefa: Desenhar Protótipo • Requisitos → Tarefa: Elaborar Diagrama de Negócio • Testes → Tarefa: Executar Testes Automatizados • Testes → Tarefa: Executar Testes Manuais (Critérios de Aceite e Exploratórios) • Qualidade → Tarefa: Validar qualidade das entregas e dos testes • Segurança → Tarefa: Validar segurança do software • Arquitetura → Tarefa: Revisar arquitetura do software • Homologação → Tarefa: Homologar software • Transferência de Tecnologia → Tarefa: Elaborar ou Atualizar Documento de Transferência de Tecnologia

Scrum Master

O Scrum Master é o responsável por garantir que o Time Scrum se oriente pelos valores e práticas do Scrum. O Scrum Master protege o time, certificando-se de que os membros não se comprometam com atividades além daquelas que eles possam cumprir dentro de uma iteração. O Scrum Master facilita a Reunião Diária e se torna o responsável pela remoção de quaisquer impedimentos observados pelo time durante estas reuniões.

O Scrum Master é responsável por facilitar o trabalho do Time Scrum em seu dia a dia e nos eventos do Scrum, de forma a aumentar a autonomia de seus membros para que juntos desenvolvam o produto, comuniquem-se efetivamente e busquem continuamente melhorar seus processos de trabalho, realizando-o com qualidade e produtividade. Ele representa a gestão do projeto (ligação), mas não é o gerente de projetos, e sim um líder facilitador que gerencia processos Scrum.

Atribuições:	• Difundir os valores do Scrum e suas práticas. • Ser parte do time: dividir responsabilidades com outros membros. • Reunir-se, regular e fisicamente, com os demais membros do time (Reuniões Diárias). • Garantir a remoção dos impedimentos que atrapalham o trabalho do Time de Desenvolvimento e ajuda a prevenir que os impedimentos aconteçam, quando possível. • Proteger o time contra interferências externas. • Monitorar as tarefas da Sprint para assegurar o sucesso. • Assegurar que o time esteja totalmente funcional e produtivo. • Possibilitar uma cooperação estreita entre todos os papéis e funções.
Tarefas:	• Gestão de Projetos → Tarefa: Realizar Reunião de Abertura de Projeto • Gestão de Projetos → Tarefa: Definir Escopo do Projeto (priorização e estimativa de esforço) • Gestão de Projetos → Tarefa: Iniciar Iteração (planejamento) • Gestão de Projetos → Tarefa: Realizar Reunião Diária • Gestão de Projetos → Tarefa: Elaborar Ata de Reunião • Gestão de Projetos → Tarefa: Encerrar Iteração (entrega/revisão) • Gestão de Projetos → Tarefa: Avaliar Iteração (retrospectiva) • Gestão de Projetos → Tarefa: Avaliar Indicadores de Qualidade e Gestão • Gestão de Projetos → Tarefa: Realizar Reunião de Encerramento de Projeto

Time de Desenvolvimento

O Time de Desenvolvimento é o responsável por entregar uma versão usável que potencialmente incrementa o produto ao final de cada iteração. É auto organizado, com um alto grau de autonomia, responsabilidade e colaboração.

Times de Desenvolvimento são multifuncionais, possuindo todas as habilidades necessárias, enquanto equipe, para criar o incremento do produto.

A partir das prioridades definidas pelo Product Owner, o Time de Desenvolvimento gera, em cada iteração, um Incremento do produto "Pronto" e que significa valor visível para os clientes do projeto.

Um membro do Time de Desenvolvimento é chamado de Desenvolvedor, independentemente do trabalho que está sendo realizado pela pessoa. Individualmente os integrantes do Time de Desenvolvimento podem ter habilidades específicas e área de especialização, mas a responsabilidade pertence ao Time de Desenvolvimento como um todo.

Times de Desenvolvimento não contém sub-times dedicados a domínios específicos de conhecimento, tais como teste ou análise de negócios.

Atribuições:	• Planejar seu trabalho, definindo com o Product Owner a meta da iteração e o que será realizado no decorrer da iteração, para então detalhar, de forma autônoma, como esse trabalho será realizado. • Realizar as tarefas de desenvolvimento do produto para atingir a meta da iteração, garantindo a qualidade do que é produzido, além de acompanhar seu progresso na iteração em direção a essa meta. • Colaborar com o Product Owner durante a iteração, sempre que necessário, para ter dúvidas esclarecidas ou solicitar decisões quanto ao produto, e para refinar e aprimorar o Backlog do Produto, preparando-o para a próxima iteração. • Identificar e informar ao Scrum Master sobre impedimentos que obstruam seu trabalho e previnir-se deles, quando possível. • Obter feedback dos clientes do projeto e demais partes interessadas sobre o trabalho realizado durante a iteração, ao apresentar e demonstrar os resultados desse trabalho ao final da iteração. • Entregar valor com frequência para os clientes do projeto.
Tarefas:	• Gestão de Projetos → Tarefa: Iniciar Iteração (planejamento) • Gestão de Projetos → Tarefa: Realizar Reunião Diária • Gestão de Projetos → Tarefa: Elaborar Ata de Reunião • Gestão de Projetos → Tarefa: Encerrar Iteração (entrega/revisão) • Gestão de Projetos → Tarefa: Avaliar Iteração (retrospectiva) • Requisitos → Tarefa: Identificar e Especificar Requisitos • Requisitos → Tarefa: Definir Critérios de Aceite • Requisitos → Tarefa: Especificar Caso de Uso • Requisitos → Tarefa: Especificar História de Usuário • Requisitos → Tarefa: Especificar Change Request • Requisitos → Tarefa: Desenhar Protótipo • Requisitos → Tarefa: Elaborar Diagrama de Negócio • Desenvolvimento → Tarefa: Implementar (codificar) software • Desenvolvimento → Tarefa: Entregar software e roteiro de publicação • Testes → Tarefa: Elaborar Plano de Teste • Testes → Tarefa: Implementar (codificar) Testes Automatizados • Testes → Tarefa: Executar Testes Automatizados • Testes → Tarefa: Executar Testes Manuais (Critérios de Aceite e Exploratórios)

	• Qualidade → Tarefa: Validar qualidade das entregas e dos testes • Segurança → Tarefa: Validar segurança do software • Arquitetura → Tarefa: Especificar Arquitetura do Projeto • Arquitetura → Tarefa: Revisar arquitetura do software • Implantação → Tarefa: Publicar software no Ambiente de Desenvolvimento ou Testes • Implantação → Tarefa: Publicar software no Ambiente de Homologação • Implantação → Tarefa: Publicar software no Ambiente de Produção • Transferência de Tecnologia → Tarefa: Elaborar ou Atualizar Documento de Transferência de Tecnologia

PDS TCE-PR → PAPÉIS → Usuário (Área de Negócio)

Usuário (Área de Negócio)

O Usuário (Área de Negócio) é um indivíduo, uma unidade, ou ainda um conjunto de representantes de várias unidades, responsável pela origem da demanda e/ou destinatária direta do uso da solução a ser desenvolvida ou adquirida.

A área de negócio demandante indicará equipe, preferencialmente multidisciplinar, que conheça e realize o processo de trabalho envolvido, a qual será encarregada das atribuições do projeto em relação à(s) área(s) de negócio, destacando-se um dos membros para figurar como gerente do projeto da área de negócio, o qual deverá, além de possuir experiência nos temas relacionados ao projeto, preferencialmente, deter conhecimento das técnicas de gerenciamento de projeto.

Esta equipe deverá estar disponível e acessível à equipe da Diretoria de Tecnologia da Informação (DTI) durante todo o período do projeto, devendo comparecer a todas as reuniões a que forem solicitadas sua presença, participar de todas as etapas de desenvolvimento do sistema, desde aprovações de artefatos e documentos até a operação do sistema em produção.

Atribuições:	• Detalhar e validar as especificações de cada uma das funcionalidades a serem desenvolvidas, conforme os padrões definidos pela Diretoria de Tecnologia da Informação. • Definir os critérios de aceite para as funcionalidades a serem desenvolvidas, conforme os padrões definidos pela Diretoria de Tecnologia da Informação. • Homologar as funcionalidades desenvolvidas, de acordo com os critérios de aceite definidos. • Respeitar os prazos estabelecidos, de forma a garantir o cumprimento do cronograma do projeto. • Comunicar à DTI todas as mudanças de escopo e funcionalidades, conforme padrão por ela definido, especificando-se a motivação e os impactos no projeto.
Tarefas:	• Gestão de Demandas → Tarefa: Elaborar Proposta de Projeto • Gestão de Demandas → Tarefa: Elaborar Pré-Projeto • Gestão de Projetos → Tarefa: Realizar Reunião de Abertura de Projeto • Gestão de Projetos → Tarefa: Definir Escopo do Projeto (priorização e estimativa de esforço) • Gestão de Projetos → Tarefa: Iniciar Iteração (planejamento) • Gestão de Projetos → Tarefa: Elaborar Ata de Reunião • Gestão de Projetos → Tarefa: Encerrar Iteração (entrega/revisão) • Gestão de Projetos → Tarefa: Realizar Reunião de Encerramento de Projeto • Requisitos → Tarefa: Definir Critérios de Aceite • Homologação → Tarefa: Homologar software

6. PADRÕES

Padrões estabelecem normas e regras a serem seguidas no desenvolvimento de software, além de apresentar boas práticas a serem observadas na execução das atividades, com vistas à obtenção de produtos de qualidade e de fácil manutenibilidade.

Os padrões estabelecidos no PDS TCEPR são:

- Padrões de Arquitetura
- Padrões de Banco de Dados
- Padrões de Código
- Padrões de Layout e Interface
- Padrões de Qualidade
- Padrões de Segurança

Padrões de Arquitetura

RESUMO

O objetivo deste documento é definir os padrões de arquitetura de software que serão adotados pela equipe de desenvolvimento do Tribunal, abrangendo assuntos como a política de merge e atualização de sistemas, implementação de padrões de layout, políticas de aceite e validação para check-ins de código, padrões para verificação de complexidade de código etc.

Os assuntos serão divididos em três grandes categorias, a saber: **Ambiente**, para tratar de configurações, políticas e catálogos dos ambientes de software; **Código**, para tratar das diretrizes e boas práticas de programação que visam uma melhor manutenção, qualidade e desempenho do software; **Processo de desenvolvimento**, para definir padrões de documentação, o papel da arquitetura e procedimentos que possam garantir uma melhor qualidade e aderência aos padrões de arquitetura do software produzido.

AMBIENTE

Catálogo de ambientes

Construir um catálogo de ambientes disponíveis no Tribunal (servidores e bancos de dados) contendo, no mínimo, o seu nome, sua finalidade, endereço de acesso, como utilizar e suas integrações com outros sistemas.

Catálogo de serviços

Construir um catálogo de serviços disponíveis para integração e composição de aplicações, como por exemplo: serviços do CIA, SICAD, Atoteca etc. Este catálogo deve abranger WebServices, bibliotecas para integração (dll's ou qualquer outro formato), serviços do schema "comum" no banco de dados e serviços Windows que estão executando nos diversos ambientes. Deve também especificar o que é necessário de configurações básicas para utilizar cada serviço, o objetivo de cada funcionalidade, quando utilizá-la e quem pode consumi-la.

É importante incluir também as bibliotecas e componentes de terceiros (inclusive comprados) que são utilizados no desenvolvimento das aplicações do Tribunal.

Sugestão de ferramenta para gerenciar os catálogos de serviço e de ambientes, inclusive com a capacidade de exibir e representar a interdependência e integrações entre os serviços de forma gráfica: "**iTop**".

Mais informações podem ser encontradas no site do fabricante: http://www.combodo.com/itop.

Há também uma versão já publicada do iTop para testes no ambiente do TCE: http://apache/itop/pages/UI.php. Usuário: admin, Senha: admin.

Arquitetura de serviços

Os serviços utilizados para integração entre sistemas poderão ser feitos de duas maneiras: diretamente pelo banco de dados, criando uma camada de abstração dos diversos serviços disponíveis através do schema "COMUM" que deve trabalhar como uma espécie de interface dos serviços disponibilizados, ou disponibilizando WebServices na camada de aplicação Web.

Não existe diretriz absoluta para realizar a integração de uma forma ou de outra, cada caso deve ser analisado a parte. A recomendação é que as integrações entre sistemas do próprio Tribunal e que usam a mesma base de dados, quando possível, sejam feitas utilizando o schema COMUM, via banco de dados, para otimizar o desempenho dos sistemas.

Em linhas gerais:

- **WebServices**: utilizar para integração com sistemas que não são do Tribunal; disponibilizar funcionalidades que dependem de validações de negócio que devem ser feitas na camada de aplicação;
- **Banco de dados**: utilizar para integração entre sistemas que são do Tribunal e exigem alto desempenho; utilizar entre sistemas que são do Tribunal em cenários onde a integração é bastante simples e não exige validações complexas.

Importante: Sempre que possível a integração deve evitar o acesso direto as tabelas de outros sistemas via banco de dados, pois isso gera um acoplamento muito forte e dificultará manutenções futuras. É sempre importante passar pelo schema COMUM, criando uma interface de abstração que utilize views ou procedures.

CÓDIGO

Padronização de arquivos de layout (.css)

Utilizar os *bundles* (pacotes) de arquivos do ASP.NET, obtendo diretamente os seguintes benefícios:

- Melhoria na organização do projeto, pois os diversos arquivos css e Javascript correlatos podem ser declarados juntos em um único caminho virtual, sendo que somente este arquivo virtual será referenciado na página. Por exemplo, uma biblioteca "foo" que contém 15 arquivos, pode ter todos os seus arquivos declarados no caminho "~/bundles/foo", bastando declarar este último na página que irá utilizá-los.
- Menor número de requisições para o servidor: como diversos arquivos são referenciados através de um único caminho virtual, ganha-se desempenho diminuindo o número de requisições. No exemplo acima, dos 15 arquivos que inicialmente seriam carregados (através de 15 requisições), somente uma requisição será feita para o caminho virtual do bundle que será carregado como um único arquivo.
- Compactação de arquivos: utilizando as ferramentas de otimização dos pacotes, o framework automaticamente "minifica" os arquivos Javascript e css, removendo todas as linhas e espaços em branco desnecessários do meio do código, bem como comentários. Além disso, a "minificação" também irá substituir automaticamente nomes longos de variáveis por apenas uma letra, economizando bastante no tamanho dos arquivos.

Segundo o site ASP.NET, a grande maioria dos browsers mais utilizados hoje limita o número máximo de conexões para um mesmo host em 6 simultâneas. Isto é, se a sua página possuir mais de seis arquivos, serão baixados os primeiros seis, e só depois que algum arquivo terminar o browser irá fazer download do próximo. Por isso, quanto mais pudermos compactar as dependências do site em poucos arquivos, melhor e mais rápido o site irá abrir.

O empacotamento (bundling) e "minificação" são ativados automaticamente quando o site não está em debug, isto é, quando a tag "compilation" estiver no formato abaixo, dentro do Web.config:

```
<system.web>
  <compilation debug="false" />
  <!-- Lines removed for clarity. -->
</system.web>
```

Existe uma forma de sempre forçar o empacotamento, através da linha de código abaixo que pode ser acrescentada no método RegisterBundles da classe BundleConfig:

```
BundleTable.EnableOptimizations
=
true;
```

A utilização dos bundles do ASP.NET garante que os arquivos css e Javascript sempre serão atualizados no cliente, quando necessário, pois sempre que houver uma alteração em algum arquivo, a chave de referência criada pelo ASP.NET será alterada. Desta forma, numa nova requisição da página, o browser irá interpretar as dependências da página como se fossem novos arquivos, de nomes diferentes, e irá atualizá-los conforme necessário.

Passo-a-passo para utilização de bundles nos projetos antigos

Na criação de novos projetos utilizando o VS2013, o bundle será utilizado automaticamente pelo projeto inicial em MVC ou WebForms. Para projetos antigos utilizando WebForms, a configuração poderá ser feita nos seguintes passos:

1. Migrar a versão do Framework para a 4.5;
2. Acrescentar, utilizando o NuGet, o pacote "Microsoft ASP.NET Web Optimization Framework";
3. Acrescentar a classe BundleConfig na pasta "App_Start", para construir os bundles de arquivos Javascript, conforme exemplo abaixo:
    - ```
      public class BundleConfig
      {
          public static void RegisterBundles(BundleCollection bundles)
          {
              bundles.Add(new ScriptBundle("~/bundles/jquery").Include(
                          "~/Scripts/jquery-{version}.js"));

              bundles.Add(new ScriptBundle("~/bundles/jqueryui").Include(
                          "~/Scripts/jquery-ui-{version}.js"));

              bundles.Add(new ScriptBundle("~/bundles/jqueryval").Include(
                          "~/Scripts/jquery.unobtrusive*",
                          "~/Scripts/jquery.validate*"));
          }
      }
      ```
4. Registrar a chamada ao RegisterBundles no método Application_Start do Global.asax do site:
    - ```
      void Application_Start(object sender, EventArgs e)
      {
          BundleConfig.RegisterBundles(BundleTable.Bundles);
      ```

```
}
```

5. Criar um arquivo xml para configuração de bundles css, com o nome "Bundle.config" na pasta raíz do projeto, conforme exemplo abaixo:
    -
```
<?xml version="1.0" encoding="utf-8" ?>

<bundles version="1.0">
  <styleBundle path="~/Content/css">
    <include path="~/Content/bootstrap.css" />
    <include path="~/Content/Site.css" />
  </styleBundle>

</bundles>
```

6. Referenciar os bundles na página, utilizando as classes Styles ou Scripts, conforme exemplos abaixo:
    -
```
<asp:PlaceHolder runat="server">

    <%: Styles.Render("~/Content/themes/base/css" %>
    <%: Scripts.Render("~/bundles/jquery") %>
    <%: Scripts.Render("~/bundles/jqueryui") %>

</asp:PlaceHolder>
```

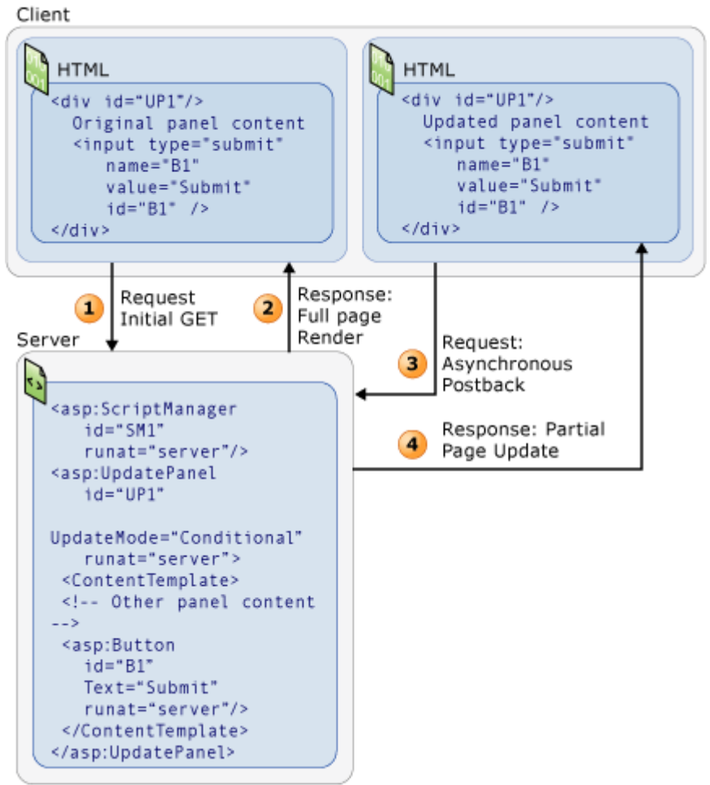
Referências:

http://www.asp.net/mvc/overview/performance/bundling-and-minification
http://blogs.msdn.com/b/rickandy/archive/2012/08/14/adding-bundling-and-minification-to-web-forms.aspx
http://boniestdeveloper.net/post/2012/10/30/Using-the-Bundling-and-Minification-Web-Optimisations-with-AspNet-WebForms-App_Themes.aspx

Utilização de requisições parciais - AJAX

Como proposta para requisições parciais, em WebForms, utilizar o UpdatePanel, componente nativo do ASP.NET. Algumas recomendações:

- Minimizar a quantidade de controles dentro de um painel de atualização, que terá de ser transferido para o cliente a cada postback.
- Se um controle deve ser atualizado baseado em um evento de outro controle não precisa colocar tudo no mesmo UpdatePanel. Só o controle que é afetado pode ser atualizado por trigger (aps:AsyncPostBackTrigger ou aps:PostBackTrigger).
- Evitar armazenar dados no ViewState para minimizar a quantidade de informação a ser transmitida a partir do servidor em cada postback (de maneira geral, minimizar as manutenções de estado entre requests, como ViewState, Session, Cookies, banco).
- Na maioria dos casos a propriedade do UpdatePanel (UpdateMode) pode ser configurado como "Condicional". Caso outros updatePanels precisarem ser atualizados, podem ser feitos por trigger.



- Uso de triggers do UpdatePanel:

<asp:AsyncPostBackTrigger>	Especifica um controle e evento para atualizar o UpdatePanel que contém esta referência gatilho.
<asp:PostBackTrigger>	Especifica um controle e evento para uma atualização de página inteira. Pode ser usado para forçar uma atualização completa quando um controle acionar o processamento parcial.

Mensagem de confirmação

Como sugestão podem ser feitas utilizando jQuery e manipulação da mensagem no html da div. Dessa forma podemos utilizar a mesma div para todas as mensagens de confirmação no sistema. Exemplo:

```
$(document).ready(function () {

    $('#deleteConfirmationDialog').dialog({
        autoOpen: false,
        resizable: false,
        height: 140,
        modal: true,
        buttons: {
            "Delete": function () {
                $(this).dialog("close");
            },
            Cancel: function () {
                $(this).dialog("close");
            }
        }
    });

});


function deleteItem(uniqueID, itemID) {
    var dialogTitle = 'Excluir Item ' + itemID + '?';
    $("#deleteConfirmationDialog").html('<p>Deseja excluir o item?</p>');

    $("#deleteConfirmationDialog").dialog({
        title: dialogTitle,
        buttons: {
            "Delete": function () {
                __doPostBack(uniqueID, '');
                $(this).dialog("close");
            },
            "Cancel": function () { $(this).dialog("close"); }
        }
    });
    $('#deleteConfirmationDialog').dialog('open');
    return false;
}
```

Janelas modais

Utilizar apenas uma div com as ferramentas para criação de janela modal do jQuery, pois ele mesmo já resolve automaticamente a sobreposição da janela sobre os demais elementos, bem como o bloqueio da tela que fica por trás do modal. Para implementar, basta simplesmente chamar o método ".dialog()" do jQuery, conforme o exemplo abaixo:

```
$( "#dialog" ).dialog({
  dialogClass: "no-close",
  buttons: [
    {
      text: "OK",
      click: function() {
        $( this ).dialog( "close" );
      }
    }
  ]
});
```

Referências:

https://msdn.microsoft.com/en-us/library/bb386454(v=vs.140).aspx
https://msdn.microsoft.com/en-us/library/bb398867(v=vs.140).aspx
https://jqueryui.com/dialog/#modal-form

Parametrização de perfis de usuário

O tratamento das permissões de acesso às páginas (telas) dos sistemas depende dos diferentes perfis de usuário especificados no CIA e pode ainda ser especificada para cada componente da tela, utilizando o motor de configuração de telas disponível no projeto Modelo. Este motor configura automaticamente os campos da tela (exibir/esconder, habilitar/desabilitar) de acordo com um arquivo xml de configuração e o controle de acesso.

Sugerimos que para uma futura versão do CIA este controle poderia ser acoplado ao controle de acesso e utilizar a configuração de perfis/campos a partir de cadastro no banco de dados ao invés de um arquivo XML de configuração presente em cada sistema.

O cadastro de configurações de campos deverá ser carregado em memória pela aplicação que está utilizando, de modo a realizar um cache destas configurações e evitar chamadas de I/O ou idas ao banco de dados desnecessárias. Porém deve haver uma política de renovação deste cache com um intervalo de alguns minutos, para que alterações feitas nos perfis sejam rapidamente refletidas na aplicação.

Validação mínima de padrões de código

Sugestões de configurações para serem aplicadas nos Team Projects:

- Impedir check-in se o projeto não compila: configurar no Team Project\Settings\Source Control, aba Check-in policy, acrescentar a policy "Builds";
- Sempre exigir no mínimo um work-item vinculado check-in ao enviar alterações.
- Sempre exigir comentários ao fazer check-in do código: configurar no Team Project\Settings\Source Control, aba Check-in policy, acrescentar a policy "Changeset Comments";
- Se possível, exigir uma análise mínima de código antes de fazer check-in: configurar no Team Project\Settings\Source Control, aba Check-on policy, acrescentar a policy "Code Analysis".

PROCESSO DE DESENVOLVIMENTO

Documento de Especificação Técnica

Este é o formato padrão para o documento de especificação técnica que irá compor o PDS TCEPR. O objetivo principal é descrever como desenvolver um caso de uso com um maior detalhamento técnico, aproximando-o mais da visão do desenvolvedor (codificação) do que das definições de negócio.

Com base nesta premissa, este documento deverá especificar para os programadores, no mínimo:

- As tabelas do banco de dados que serão acessadas (no formato de diagrama ER);
- A estrutura dos métodos e classes das diversas camadas que deverão ser implementados e onde ficarão as regras de negócio (no formato de diagrama de sequência);
- Um diagrama de pacotes que representa uma visão lógica da interação dos diversos componentes envolvidos no caso de uso (servidores de aplicação, banco de dados, integração com outros sistemas);

A princípio, a recomendação é que exista pelo menos uma especificação técnica para cada requisito ou caso de uso do sistema que está sendo desenvolvido.

Gestão de mudança/configuração

Padrões para versionamento de código

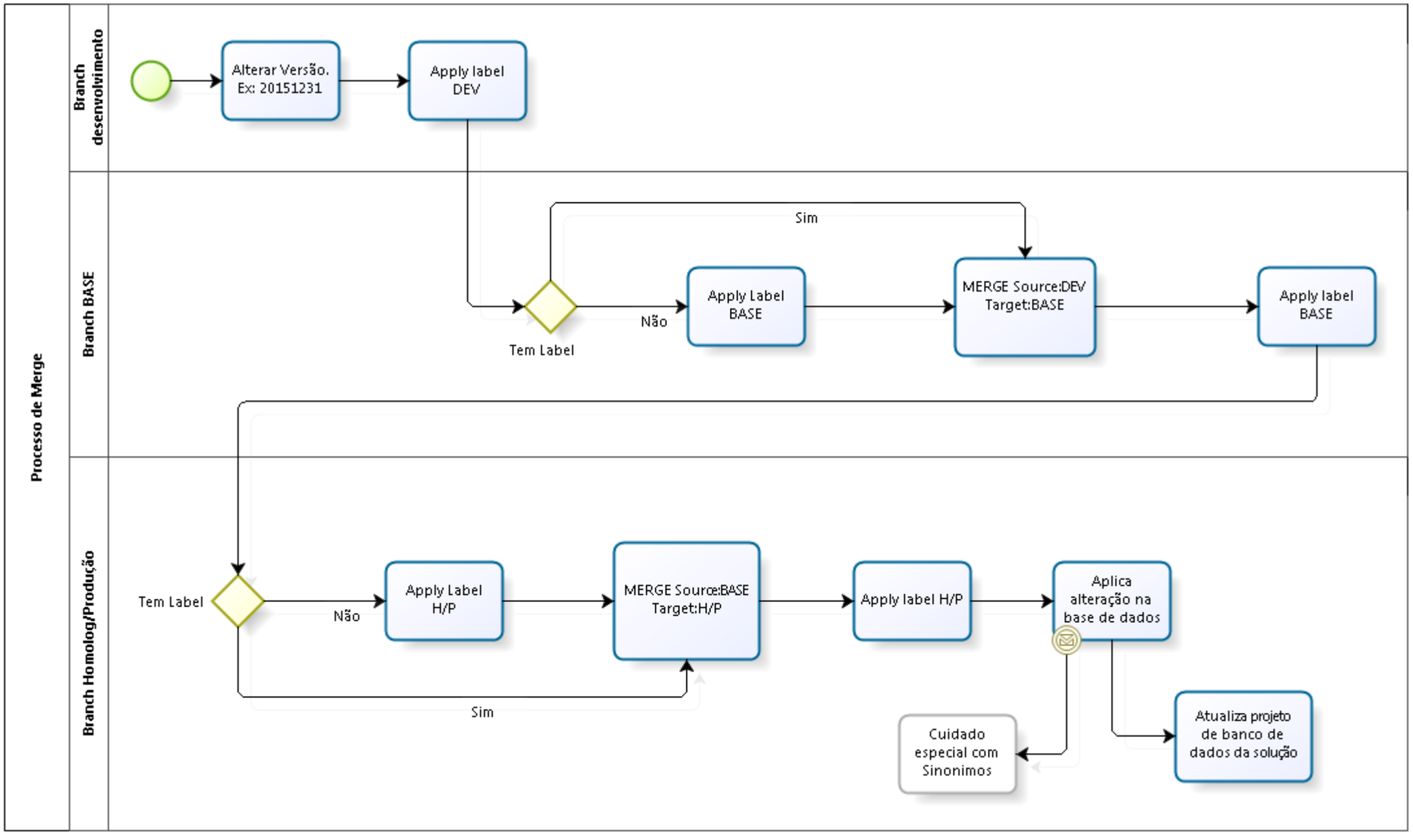
Os branchs de projeto seguirão a seguinte estrutura:

Base |---- Desenvolvimento

Base |---- Homologação

Base |---- Produção

- **Desenvolvimento**: onde os desenvolvedores fazem todo desenvolvimento, melhorias e evoluções do produto.
- **Homologação**: resultado do merge do Desenvolvimento para a Base e da Base para a homologação. Alterações somente para correção de bugs emergenciais e que devem ter uma segunda atividade programada para corrigir concomitantemente o branch de Desenvolvimento. Não recomendamos fazer o merge "para trás", da Homologação para o Desenvolvimento. Neste caso é preferível efetuar a correção duas vezes.
- **Produção**: resultado do merge da Homologação para a Base e da Base para a Produção. Alterações somente para correção de bugs emergenciais e que devem ter outras atividades programadas para corrigir concomitantemente os branches de Desenvolvimento e Homologação. Não recomendamos fazer o merge "para trás", da Produção para Homologação ou Desenvolvimento. Neste caso é preferível efetuar a correção novamente nos outros branches.
- **Base**: Somente recebe alterações resultantes das operações de merge e serve como auxiliar para repassar aos outros branches.
- **Merge**: sempre do Desenvolvimento para o Base e do Base para Homologação ou Produção (forward only, sempre "para frente").



Labels e versionamento

Para toda publicação no IIS, seja em homologação (IIS-DES) ou em produção (luana2), deve ser criado um label no TFS. O nome da label deve seguir o padrão especificado abaixo. Nos comentários, é recomendável fazer a indicação dos workitens (bugs, task, backlogs, etc) disponibilizados naquele momento, ou da Sprint que está sendo publicada.

Padrão de nome do label: data invertida (yyyyMMdd) HHmm - operação - nome de quem criou o label

Exemplo:

- 20140318 – Publicação em homologação – Alessandro
- 20140319 – Publicação em produção – Valdair
- 20140319 1400 – Republicação – Robson

Recomendações para publicação em produção

Visando garantir que o código presente nos respectivos branches do source control do TFS realmente reflete a situação dos ambientes reais das aplicações publicadas, seguem algumas recomendações da equipe de arquitetura.

O acesso aos servidores de produção deve ser restringido. Somente o pessoal autorizado (arquitetos ou gestores de configuração) terá acesso a esse servidor.

Com a restrição de acesso, deve ser definido um papel efetivo de gestão de configuração, de modo que somente o pessoal deste papel será o responsável pela alteração e manutenção destes ambientes, conforme orientação dos roteiros de publicação.

Partindo do princípio dos papeis definidos, surge o momento onde poderão ser aplicadas as revisões do conteúdo a ser publicado, com um check-list do roteiro de publicação contendo os seguintes itens:

1. Os testes necessários foram realizados?
2. A versão do assembly foi atualizada?
3. A label foi criada no branch refletindo a versão do assembly?
4. As definições de banco no source control refletem o banco real do ambiente?
5. Foram relacionados os Workitems/Sprint/CACO/SS que está sendo publicada na label?
6. No arquivo de configuração do site (Web.config), existe alguma chave que está apontando para algum ambiente diferente da produção?

Ao publicar uma nova versão da aplicação, deve ser feito backup da atual versão na pasta \\beta\backup_Aplicacoes. Esta pasta está separada pelos nomes dos servidores (Luana, iis-des etc), sendo que dentro de cada pasta de servidor existirá uma pasta para cada sistema publicado. Dentro da pasta do sistema, a versão deve ser armazenada em um arquivo zipado no formato "YYYYMMDD.zip".

Antes de publicar efetivamente os arquivos atualizados no servidor de aplicações, **sempre criar o arquivo "app_offline.htm" na pasta do servidor para "desligar" o aplicativo e avisar os usuários de que uma manutenção está ocorrendo**.

Para a publicação de uma nova versão do sistema, ele deverá sempre estar acompanhando do artefato "Ambiente e dependências do sistema".

Conceitos Gerais

1. Único conceito por teste, ou seja, cada método deve ter uma afirmação do que se propõe a testar;
2. Não escreva testes grandes demais, quando ficar muito grande deixe-o e escreva testes menores que representem partes dele. Cada teste deve ser o mais simples possível.
3. Todo método de teste deve ter pelo menos uma assertiva no final (Assert);
4. Não faça check-in de testes quebrados, para que não seja perdida a rastreabilidade de quanto o problema surgiu;
5. Isole as dependências de objetos dentro de MOQs ou Fakes (Objetos simulados ou falsos);
6. Cada classe de teste deve corresponder a uma classe de negócio;
7. Se necessário, separe os códigos de "Arrange" (preparação do teste) em uma classe [Partial], assim a classe de teste principal terá somente os métodos de teste;
8. Nos testes, se possível, evite:
    1. Conversar com o banco de dados;
    2. Comunicar com a rede;
    3. Ler arquivos externos;
    4. Concorrer com outros métodos de teste;
    5. Configurar ou preparar o ambiente (Externo) para executa-los;
9. Estruture o seu método de teste em três partes:
    1. Arrange (Preparar os dados para fazer o teste);
    2. Action (Chamada do método a ser testado);
    3. Assert (Testar o resultado obtido);
10. Dados com significado evidente:
    1. O teste é escrito para alguém ler, serve como documentação do software.
    2. Escreva assertivas que representem o objetivo do teste;
11. Aplique o modelo F.I.R.S.T.
    1. Fast - Rápidos: devem ser rápidos, pois testam apenas uma unidade;
    2. Isolated - Testes unitários são isolados, testando individualmente as unidades e não sua integração;
    3. Repeatable - Repetição nos testes, com resultados de comportamento constante;
    4. Self-verifying - A auto-verificação deve verificar se passou ou se deu como falha o teste. Ainda que o objetivo seja o teste de uma falha, o resultado deve ser positivo;
    5. Timely - O teste deve ser oportuno, sendo um teste por unidade.
12. Se possível, aplique os princípios básicos do **TDD**:
    1. Não escreva nenhum código de produção até que tenha escrito antes um teste de unidade que detecte uma falha;
    2. Não escreva mais que um teste de unidade do que o suficiente para fazê-la falhar;
    3. Você não pode escrever mais código de produção do que o que seja suficiente para passar pelo atual teste de unidade;

Recomendações para a implementação dos testes unitários

- Utilizar a ferramenta de testes unitários própria do Visual Studio 2010/2013.
- Deve existir uma classe de teste para cada classe de negócio ou controller do sistema, padronizando o nome com o formato: ClasseNegocio/ClasseController + sufixo "Test";
- Deverão existir quantos métodos de testes quanto forem necessários para cobrir todos os cenários de testes previstos. Utilizar como padrão para o nome dos métodos: nome do método original + condição testada + resultado esperado. Exemplo: ao inserir com sucesso uma pessoa utilizando a classe PessoaController, o método de teste deverá ficar com o nome: Inserir_ComNomeCorreto_Sucesso().
- Utilizar a classe básica de testes do projeto modelo **BaseTeste.cs** para realizar etapas básicas de "Arrange".
- Para cada classe de teste, deve haver um método separado para a carga inicial dos dados utilizados nos testes (separação da etapa de "Arrange").
- Utilizar uma transação dentro de cada método de teste sem efetivar as alterações, de modo que o teste possa ser executado até o fim (chegue no banco de dados) e, no final, as alterações sejam revertidas e os dados não sejam estragados. **Somente no caso do teste precisar usar o banco de dados.**
- A opção mais recomendada é utilizar ou MOQS ou fakes das camadas de acesso a dados.
- Para cada classe de teste declarar apenas uma vez uma instância privada da classe de negócio que será testada.

Torna-se um pré-requisito necessário para a construção dos testes unitários/automatizados a documentação com os cenários de teste que deverão existir. Como exemplo de planilha de cenários de teste, pode-se utilizar o arquivo PlanilhaExemplo.xlsx que pode ser baixado nesta página, ou na pasta "Documents" do projeto modelo.

Sugestões de ferramentas para testes funcionais:

- Utilizar o Test Manager da Microsoft.
- Selenium.
- SoapUI.

DOWNLOAD:

Checklist_EspecTecnica.xlsx

Padrões de Banco de Dados

RESUMO

O objetivo deste documento é definir padrões para criação e definição de elementos do banco de dados.

PADRÕES DE BANCO DE DADOS

A padronização descrita abaixo deve ser observada pela equipe de desenvolvimento e contempla tanto padronização de criação quanto definições de elementos do banco de dados.

Servidores de Banco de Dados

PRODUÇÃO: ALPHA

DESENVOLVIMENTO: SQL-DES

IMPORTANTE: As alterações e/ou atualizações em banco de dados de PRODUÇÃO devem sempre ser feitas através de SOLICITAÇÃO DE SERVIÇO.

Criação de Novo Database

Para criação de databases específicos para as aplicações em desenvolvimento, deve-se usar exclusivamente o servidor SQL-DES.

Banco de Dados de Desenvolvimento

Para criar um novo banco de dados para desenvolvimento de nova uma aplicação utilize o seguinte padrão: sigladaaplicacaoDES.

EXEMPLO:

sgaDES
agenDES
apDES

Banco de Dados de Homologação

Para criar um novo banco de dados para desenvolvimento de nova uma aplicação utilize o seguinte padrão: sigladaaplicacaoHOMOLOG.

EXEMPLO:

sgaHOMOLOG
agenHOMOLOG
apHOMOLOG

Banco de Dados de Produção

Os bancos de dados de desenvolvimento se transformam em schemas no banco de dados de produção TC no servidor de produção ALPHA.

EXEMPLO: Os nomes dos schemas seguem o nome do banco de dados de desenvolvimento sem o sufixo.

sga
agen

NOMENCLATURAS

Schemas

Os esquemas devem ser nominados da mesma forma que o sistema no qual eles pertencem.

EXEMPLO:

A tabela cpProcesso (dbo.) que pertence ao sistema de Trâmite de Processos passaria a ser chamada de Tramite.Processo onde Tramite é o nome do schema referente ao sistema de Trâmite de Processos e Processo é a descrição da tabela.

Tabelas

- Regra para criação de tabelas:
  - Nome do Schema + '.' + Descrição

- Regras para a descrição resumida do conteúdo:
  - Nome Significativo da tabela ou visão, iniciando as palavras com maiúsculas (Yyyyyy)
  - Usar substantivos no singular;
  - Não devem ser usados caracteres especiais. Exemplo: "_" (underscore).

EXEMPLO:

Tramite.Processo, SimAM.Empenho

- Tabelas temporárias:
  - Nome do Schema + '.' + Nome Tabela + 'Temp'

EXEMPLO:

SimAM.EmpenhoTemp

Todas as tabelas deverão ter um comentário devidamente preenchido na ferramenta de modelagem para facilitar a dicionarização

Nome de Atributos

- O nome de uma coluna deverá ser composto da seguinte forma:
  - Descritor + Descrição do Atributo (1ª letra maiúscula, procurando ser o mais explícito possível).

- Não se qualifica o atributo com a sigla do sistema: ex.: nrProcesso e não nrProcessoFF

- Os descritores são identificadores do tipo de informação que o campo armazena. Devem seguir a tabela:.

DESCRITOR	NOME	DEFINIÇÃO	EXEMPLO	TIPO DE DADOS
nr	Número	Número de existência própria, não gerado internamente.	nrCPF	tinyint (inteiros até 255) smallint (inteiros até 32767) int bigint numeric (decimais)
id	Identificador	Número gerado internamente pelo sistema e que identifica uma ocorrência unívoca no BD	idPessoa	semelhante a nr, exceto que não se utiliza o formato numeric

vl	Valor	Número utilizado para denotar valor monetário	vlSalario	numeric
cd	Código	Número utilizado como substituição de uma ocorrência evitando o uso de campo caractere	cdRelatório	semelhante a id
dt	Data	Data onde não seja necessária a utilização de hora	dtNascimento	smalldatetime
dh	Data/Hora	Data onde seja necessária a utilização de hora	dhEncaminha	datetime
ds	Descrição	Descrição de uma ocorrência	dsBem	char
				varchar
nm	Nome	Identificador de uma ocorrência	nmPessoa	char
				varchar
fl	Flag	Variável booleana	flSituacao	Utilizar preferencialmente 'S' e 'N'
ano	ano	Ano	anoLRF	smallint, utilizar sempre com 4 posições
mes	mês	Mês	mesLRF	tinyint, utilizar sempre com 2 posições
dia	dia	Dia	diaLRF	tinyint, utilizar sempre com 2 posições
qt	Quantidade		qtSalario	semelhante a nr
st	Situação		stEncaminhamento	tinyint
pc	Percentual		pcPrevidencia	numeric
rs	Resultado		rsAnalise	char
				varchar
				text
hr	Hora		hrEntrada	datetime
hs	Histórico		hsCadastroEntidade	char
				varchar
				text
nv	Nível		nvHierárquico	tinyint
				smallint
sld	Saldo		sldHoras	numeric
				int
str	**String**	**Não deve ser utilizado**		
md	Modelo		mdManual	char
				varchar
av	Avaliação		avInstrutor	
sg	Sigla		sgSistema	char
				varchar
sq	Sequencia		sqAssunto	numeric
				int
tp	Tipo		tpModelo	char
				varchar

EXEMPLO:

Nome do cliente da empresa: nmCliente
Data de Nascimento: dtNascimento

Situações de dúvida:

1. nm e ds. Bem exemplificado no seguinte caso: nmMarcaProduto e dsProduto. No primeiro atributo estabelecemos a marca (Brastemp); no segundo, descrevemos o produto (Lava Louça 220V com Aquecimento). O campo nm, assim como o campo nr, tem existência própria fora do BD.
2. id e cd. Pode-se considerar id como um "número de documento interno" do sistema, que identifica uma única ocorrência. Ex. idPessoa O código (que deve ser numérico) é utilizado como SUBSTITUTO de uma tabela fixa de ocorrências. Ex: cdTipoAto.
3. id e nr. O identificador nr deve ser usado quando: a) o número em questão possui existência própria fora do BD (Ex: nrCPF); b) quando o número identifica um seqüencial de numeração gerado pelo sistema e que terá existência própria (nrAto), inclusive em meio físico (papel, p.ex.). O id é preferível em casos onde o número é gerado internamente, mas não terá – ou não deveria ter - uma existência própria (idPessoa).
4. fl, tp e st. fl deve ser usado unicamente para variáveis com 2 valores discretos (0 e 1, S e N, F e M, etc...). Se houver mais situações, deve-se usar ou cd ou tp. st deve ser usado unicamente em situações de diferenciação de estado de um atributo, isto é, enquanto cd e tp são atributos estáticos, st é atributo dinâmico, constantemente modificado pelos sistemas.

Critério para criação de descritores que não constam na tabela acima:

- Caso o tipo da informação possua até três letras, utilizar a mesma palavra para o descritor, sem acentuação. Ex: dia, mes, ano.
- Caso o tipo da informação possua mais de três letras, utilizar as três primeiras letras da palavra ou uma abreviação comumente utilizada.
- Usar termos em PORTUGUÊS.

Observações Gerais:

- Evitar o máximo possível abreviaturas, para qualquer caso (tabelas, sps, atributos).
- Atributos que são usados em várias tabelas, procurar usar a descrição de acordo com a função na tabela. Ex: idPessoa, idJuridica, idFisica, idFuncionario. Nomes diferentes para o mesmo atributo.
- Usar termos em PORTUGUÊS.
- Todas os campos das tabelas deverão ter um comentário devidamente preenchido na ferramenta de modelagem para facilitar a dicionarização.

Chave Primária

- A nomenclatura de constraints de chave primária deverá seguir à regra, sendo o indexador facultativo:
  - "PK" + "nome da tabela"

EXEMPLO:

Tabela cpProcesso - PKcpProcesso
Tabela Empenho - PKEmpenho

- Em geral, a chave primária deverá ser CLUSTERED a não ser em casos específicos a critério do desenvolvedor, desde que corretamente justificada.

- Primary key única como id + nome tabela sem prefixo e sendo tinyint, smallint, int, bigint ou qualquer outro tipo desde que numérico inteiro.

- As colunas de chave primária deve ter seus valores controlados por uma tabela de geração de números, evitando o uso do IDENTITY e do SEQUENCE.
- Tendo em vista a obrigatoriedade de unicidade do nome de constraints, caso necessário, usar um indexador ao final do nome. Isto se deve ao fato de duas tabelas poderem ter o mesmo nome, porém em schemas diferentes.

EXEMPLO:

Tabela seiced.Pessoa - PKPessoa1
Tabela simam.Pessoa - PKPessoa2

Estratégia para geração dos números da chave primária:

- Criar uma procedure no formato <schema>.PegaProximoNumero. Esta procedure deverá receber por parâmetro o nome da tabela cuja chave será gerada e devolver o próximo número. Seguir o exemplo de código abaixo:

```
CREATE PROCEDURE [<schema>].[PegaProximoNumero]
 @nmTabela char(50)
AS
DECLARE @idProximo int
IF NOT EXISTS(SELECT idProximo FROM ap.Numero WHERE nmTabela = @nmTabela)
BEGIN
 INSERT INTO <schema>.Numero (nmTabela, idProximo) VALUES (@nmTabela, 2)
 SET @idProximo = 1
 GOTO FIM
END
UPDATE< schema>.Numero
 SET @idProximo = idProximo,
     idProximo = idProximo+1
 WHERE nmTabela = @nmTabela
FIM:
SELECT @idProximo AS idProximo
```

- A procedure utiliza a tabela <schema>.Numero para controlar a geração de números. A estrutura desta tabela deve ser a descrita abaixo:

```
CREATE TABLE [<schema>].[Numero](
     [nmTabela] [char](50) NOT NULL PRIMARY KEY,
     [idProximo] [int] NOT NULL
)
```

Chave Estrangeira

- Nomear as chaves e evitar que o SQLServer as nomeie.
- A nomenclatura de constraints de chave estrangeira deverá seguir à regra:
  - "FK" + "tabela filho" + "tabela pai"

EXEMPLO:

FKcpEncaminhamentocpProcesso

- Se houver mais de uma referenciando a mesma tabela, usar indexador ao final do nome.

EXEMPLO:

FKVinculacaoPessoa1, FKVinculacaoPessoa2

Chave Alternativa

- Recomenda-se que exista uma chave alternativa que represente a chave primária real do negócio implementada como um índice único.
- A nomenclatura de constraints de chave alternativa deverá seguir à regra:
  - "AK" + "Nome da Tabela" + "indexador"

EXEMPLO:

AKPeJuridica1

Índices de Campo

- De preferência, nomear os índices. Evitar que o SQLServer os nomeie.
- A nomenclatura de índices em campos deverá seguir as regras:
  - Índices cluster: IXTabelaNomeIndexador

EXEMPLO:

IXcpProcesso1

  - Índices não cluster: INTabelaNomeNomeColuna do Índice

EXEMPLO:

INcpProcessoDataCancelamento

- **E se tiver várias colunas? Por que não seguir a mesma regra de chave alternativa?**

Procedimentos Armazenados, Funções e Pacotes

Nomenclatura

O nome deve ser composto pelo schema mais o nome do método a que se refere a execução, conforme o padrão de codificação (link)

EXEMPLO:

Tramite.Peticionar
Licitacao.RetornarTotalEmpenhado
Atos.PublicarPauta

Cabeçalho

Em cada procedimento armazenado, trigger, funções e views deverá existir um cabeçalho de controle de alterações em produção contendo algumas informações, conforme o exemplo abaixo.

EXEMPLO:

-- Descrição: Cálculo de impostos para pessoa física.

Grid

Data da alteração - Quem - Solicitação de Serviço - O que foi alterado

Regras Gerais

- É obrigatória a utilização de identação e comentário entre blocos de linhas para facilitar a manutenção dos procedimentos armazenados, funções e triggers.
- Manter as transações curtas.
- Utilizar codificação simples e de fácil entendimento. Onde não for possível, explicar os comandos utilizados e sua justificativa em breves comentários. Se for código obtido externamente identificar a origem (URL, livro etc).

- Deve-se evitar a construção dinâmica de comandos SQL dentro de stored procedures.
- Fazer o tratamento de erro após emitir um comando de atualização (INSERT, UPDATE ou DELETE) ou após a chamada de um outro procedimento armazenado. Caso o comando esteja dentro de uma transação, efetuar o ROLLBACK após capturar o erro.
- No tratamento de erros, diferenciar os erros de negócio dos erros de banco de dados, utilizando blocos TRY CATCH no SQL. Os erros de negócio devem ser lançados usando o código 50000. Este código será tratado como uma mensagem de erro de negócio a ser exibida ao usuário, por padrão, pelo modelo de aplicação .Net.
- Evitar o uso de cursor nos casos em que existe uma cláusula SQL capaz de obter o mesmo resultado, por exemplo, contagens e somas.
- Fechar e desalocar todos cursores que foram abertos mesmo após a ocorrência de um erro.
- Não utilizar o comando "SELECT * FROM..". Especificar todos os campos a serem selecionados.
- Especificar todos os nomes dos campos no comando INSERT e não somente os valores a serem inseridos.

EXEMPLO:

"INSERT INTO TABELA1 (CODIGO, NOME, TIPO) VALUES (1,'jose', NULL);"

Gatilhos (Triggers)

- A nomenclatura de gatilhos deverá seguir à regra:
  - "Nome do schema" + "." + "tg" + "Nome da Tabela" + "Abreviação do Tipo"
    - tg: fixo, identificador de trigger
    - Tabela: nome da tabela onde atua o trigger
    - Função: função do trigger. No caso das funções básicas (Insert, Update, Delete), utilizar INS, UPD, DEL.

Tipo	Abreviação do Tipo
Insert	Ins
Update	Upd
Delete	Del

EXEMPLO:

tgcpEncaminhamentoINS

- Todo trigger deverá prever tratamento para múltiplas linhas.

Visões (Views)

A nomenclatura de visões deverá seguir à regra:

- "Nome do schema" + "nome significativo de visão" iniciando as palavras com maiúsculas

Regras de Validação (Rules)

A nomenclatura de Rules deverá seguir à regra:

- "RU" + "ObjetivoDaRule"

EXEMPLO:

RUValidaCPF

Sinônimos

No desenvolvimento utilizar sinônimos que refereciem objetos externos do schema mantendo o schema e nome originais dos objetos para permitir colocar o sistema em produção sem alterações.

EXEMPLO:

Ao referenciar o objeto dbo.cpProcesso do TCDes, utilizar um sinônimo com o nome dbo.cpProcesso no banco de dados/schema da aplicação sendo desenvolvida.

Integração - Schema Comum

Definição:

Padrão de nome para objetos de integração deverá respeitar o schema comum e a indicação do nome será feita com a junção do schema original com o nome dentro do schema.

EXEMPLO:

comum.cadListarX deverá executar diretamente cad.ListarX passando os mesmos parâmetros, incluindo eventualmente testes de acesso/permissãocomum.cadListarX deverá executar diretamente cad.ListarX passando os mesmos parâmetros, incluindo eventualmente testes de acesso/permissão (token)

As stored procedures do schema Comum deverão utilizar o EXECUTE AS Comum.

Padrões de Código

RESUMO

O objetivo deste documento é definir padrões para codificação (programação) de sistemas.

BOAS PRÁTICAS DE PROGRAMAÇÃO

Orientações Gerais para Nomear Componentes

- Nomes tem que deixar clara a intenção do componente
  - int d - não tem significado ou depende do contexto
  - int diasPassados - tem significado claro sem depender do contexto
- Nomes devem ser pronunciáveis
  - string semPass - tem significado, mas é complicado de ler;
  - string semanaPassada- tem significado claro e facilita a leitura
- Nomes devem ser pesquisáveis
  - public void retornaObjeto() - tem significado, é pronunciável, mas é difícil de pesquisar, pois é muito genérico;
  - public void retornaFuncionario() - é passível de pesquisa, pois é específico.

Padrões de Funções

- Forma
  - Tamanho Pequeno
  - Quanto menor a função maior a legibilidade
  - Devem possuir no máximo 20 linhas.
- Blocos e Indentação
  - Blocos de instruções de controle (if, else, while...) devem possuir no máximo uma linha: chamada a uma função tratadora.
  - Além de manter o código limpo, mantém o código documentado pelo nome altamente descritivo da função.
  - Funções não devem ser grandes o suficiente para conter blocos aninhados (aceita-se no máximo dois).
    - Ps. Tamanho da função e profundidade dos blocos aninhados compõe métrica de complexidade da função.
- Uso de Switch
  - O objetivo do Switch é dar tratamento especializado a diferentes casos de uma mesma regra de negócio.
  - Para aumentar a legibilidade do código sugere-se que a complexidade esteja oculta por uma interface que representa a regra enquanto o tratamento especializado esteja encapsulado em uma fábrica que seleciona uma implementação da regra adequada a cada caso.
- Verbos e Palavras chave
  - Utilize verbos para definir o que a função faz, e palavras chaves para especializar seus argumentos.

        - AssertEquals(expected,actual) //genérico
        - AssertExpectedEqualsActual(expected,Actual) //especialista
- Número de Argumentos da Função
    - O número de argumentos tem muita influência na legibilidade do código. Idealmente funções devem possuir apenas um argumento, seguido de perto por dois argumentos. Três argumentos devem ser evitados quando possível, enquanto o uso de quatro argumentos ou mais exigem justificativa especial.
    - Argumentos e funções estão em diferentes níveis de abstração. Um número grande de argumentos obscura o foco do objetivo (PARA) representado pelo nome da função.
    - A testabilidade do código cresce exponencialmente de acordo com o número de argumentos.
    - Casos Comuns:
        - Eventos são funções que usualmente aceitam um único argumento e tem objetivo de alterar estado do sistema ao invés de produzir um resultado desacoplado e específico. Ao utilizar tal forma é necessário deixar claro através do nome que a função representa um evento.
        - Há conjuntos de argumentos que, devido à sua coesão e ordenação natural são fáceis de entender (como pontos cartesianos).
        - O processo de leitura de uma lista argumentos sem coesão exige uma pausa entre os argumentos que pode diminuir a atenção necessária ao entendimento.
    - Quando possível, deve-se convertê-los para a um único argumento através de delegação ou herança.
    - Para o caso de muitos argumentos, agrupá-los em objetos não diminui a complexidade do teste, porém facilita a leitura e entendimento do código.
- Retorno e Argumentos
    - A leitura de uma função parte do princípio de que um argumento alimenta um processo de transformação que gera um retorno.
    - Argumentos de saída (out args[]) são mais difíceis de interpretar que argumentos de entrada. A análise de informações externalizadas através de argumentos requer uma checagem dupla.
    - Retorno de método condicionado à sua execução com sucesso garante sua transacionalidade, enquanto a externalização de resultados através de argumentos pode provocar um estado inconsistente quando parte do resultado foi externalizado e uma exceção ocorre antes do término da função.
- Faça Uma Coisa
    - Funções devem fazer uma coisa. Devem fazer bem, e fazer somente isso. Se a função executa apenas os passos indicados pelo nível de abstração representado pelo seu nome, ela faz apenas uma coisa.
    - Controle:
        - Uma função faz mais de uma coisa quando for possível extrair dela outra função cujo nome não seja apenas uma releitura do nome de sua função de origem.
        - Funções divididas em sessões são um exemplo claro de funções fazendo mais de uma coisa.
- Um Nível de Abstração por Função
    - Garanta que as sentenças que compõe sua função estejam no mesmo nível de abstração.
    - Detalhes de implementação não devem estar misturados com conceitos essenciais.
    - Controle:
        - Quando uma função apresenta diferentes níveis de abstração é indício de que ela está fazendo mais de uma coisa.
- Leitura de Cima para Baixo
    - A leitura do código em forma de narrativa é um critério de legibilidade.
    - Deve ser possível ler um parágrafo como uma série de parágrafos divididos em objetivos (PARA) e condições para atingir esses objetivos (DEVO).
    - Controle:
        - É importante atentar que a legibilidade depende de conceitos anteriores, tornando-se um importante critério para avaliação destes conceitos.
        - Nomes significativos, boa estrutura de blocos e indentação adequada melhoram a legibilidade do código.
        - A legibilidade do texto necessita que as sentenças de uma função estejam no mesmo nível de abstração.
        - O nome da função define seu objetivo, suas sentenças e condições.
        - Se as sentenças não estão no mesmo nível de abstração a legibilidade fica prejudicada.
- Funções Devem Fazer Algo ou Responder a Algo, Nunca Ambos:
    - Funções que retornam um resultado em retorno a uma chamada não devem alterar o estado do sistema, sob pena de ocultar efeitos colaterais indesejados.
    - Por exemplo:
    - O método 'boolean validarSenha()' é invocado sempre que o usuário efetua o login. Ao efetuar o login, é necessário 'inicializarVariáveisAmbiente()' do usuário, por isso, esse método é invocado ao 'validarSenha'. Tal decisão faz que o método 'validarSenha' quebre a regra 'faça uma coisa' e causando um efeito colateral que não será identificado na leitura do método chamador.
    - Preferencialmente, quebre em métodos diferentes.
    - Quando inevitável, mude o nome da função para tornar seu efeito explícito ('validarSenhaEPrepararAmbiente').

Tratamento de Exceções

- Use exceções ao invés de códigos de retorno
  - Códigos de retorno para erros são legado de linguagens que não dão suporte ao tratamento de exceções
  - Escreva seus blocos de tratamento primeiro
- Blocos de tratamento de exceção definem um escopo.
  - Use catch para que seu sistema retorne a um estado consistente independentemente do que acontece no bloco try.
- Sempre trate System.Exception (DISCUTIR SOLUÇÃO TÉCNICA/DE ARQUITETURA - projeto modelo)
  - Uma mudança na assinatura de um método em camadas mais baixas do software pode forçar em cascata uma mudança no tratamento de exceções nas camadas mais altas.
  - O uso de exceções não checadas força que as camadas superiores implementem tratamentos genéricos ou deleguem explicitamente o tratamento a camadas superiores. Dessa forma, é possível tornar o código resiliente a mudanças em suas dependências, como componentes e módulos.
- Proveja contexto com às exceções
  - Toda exceção deve prover contexto o suficiente para determinar a fonte e localização de um erro.
  - Ao encapsular uma exceção, sempre anexar a exceção de origem como inner exception da nova exceção.
- Defina Classes de Exceção Conforme Necessidades do Chamador
  - Detalhes técnicos da exceção são importantes para a rastreabilidade do problema, mas a relevância dessas exceções depende do contexto da classe chamadora.
    - Detalhes de banco não tem relevância quando tratadas regras de negócio.
    - O uso de bibliotecas de terceiros fornecem uma vasta gama de exceções que podem ou não ser relevantes segundo contexto da classe chamadora.
  - Aconselha-se criar Wrappers para o uso de bibliotecas específicas para que convertam as exceções da API terceira em exceções significativas para o contexto da classe chamadora.
    - Criar Wrappers para APIs de terceiros é uma boa prática que permite o isolamento da dependência e a modelagem de APIS adaptadas às necessidades do sistema.
- Não use Try Catch para controlar fluxo.
  - Saiba diferenciar fluxo normal de fluxo de exceção
  - Há casos em que uma exceção de negócio determina uma condição de negócio, e não uma interrupção. Tal comportamento fere o Princípio do Menos Atônito - o resultado de uma operação deve ser óbvio, consistente e previsível - tornando o código ilegível.
  - Quando não puder ser evitado que exceções controlem o fluxo regular, isole a decisão em funções específicas.

PADRÕES DE CÓDIGO

Padrões Aplicados ao Projeto Modelo

No projeto modelo de aplicação .Net, além dos padrões listados acima, usa-se como boa prática a seguinte forma de nomenclatura dos métodos das classes de negócio e repositório:

- Para métodos que retornam listas de objetos, utilizar o nome “Listar”;
- Para métodos que retornam um único objeto, utilizar o nome “Obter”;
- Se houver necessidade de que os objetos sejam retornados fora do contexto do Entity Framework (opção NoTracking na consulta), utilizar o sufixo “Detached” no nome;
- Os demais métodos CRUD deverão seguir o padrão: “Incluir”, “Alterar” e “Excluir”.

Para a nomenclatura das classes, utilizar o padrão abaixo para identificar de qual projeto a classe faz parte:

- Nomes de classes do projeto repositório devem utilizar o sufixo “Repositorio”;
- Nomes de classes do projeto negócio devem utilizar o sufixo “Negocio”.

Padrões de Nomenclatura .NET

Fonte: http://msdn.microsoft.com/en-us/library/ms229002(v=vs.110).aspx

Uso de letras maiúsculas e minúsculas

Para todos os casos, quando um identificador é formado por palavras compostas, estas são diferenciadas tendo sua primeira letra maiúscula. Demais, diferenciam-se pelo uso de maiúscula e minúscula na primeira letra quando:

- Camel case: utiliza primeira letra do identificador em caixa baixa.
    - Ex: camelCase.
- Pascal case: utliza primeira letra do identificador em caixa alta.
    - Ex: PascalCase.

Identificador	**Case**	**Exemplo**
Namespace	Pascal	`namespace System.Security { ... }`
Type	Pascal	`public class StreamReader { ... }`
Interface	Pascal	`public interface IEnumerable { ... }`

Method	Pascal	`public class Object {`
		`public virtual string ToString();`
		`}`

Property	Pascal	`public class String {`
		`public int Length { get; }`
		`}`

Event	Pascal	`public class Process {`
		`public event EventHandler Exited;`
		`}`

Field	Pascal	`public class MessageQueue {`
		`public static readonly TimeSpan`
		`InfiniteTimeout;`
		`}`
		`public struct UInt32 {`
		`public const Min = 0;`
		`}`

Enum value	Pascal	`public enum FileMode {` `Append,`

		... }

Parameter	Camel	public class Convert {
		public static int ToInt32(string value);
		}

Nomenclatura de Assemblies e DLLs

Os nomes de Assemblies devem corresponder ao namespace.

- `[assembly: AssemblyTitle("TCEPR.PDI.Negocio")]`

Considere nomear DLLs segundo o padrão <companhia>.<componente>.dll.

O <componente> pode ser composto por clausulas divididas por ponto.

- Ex: TCEPR.PDI.Negocio.dll

Nomenclatura de Namespaces

A nomenclatura do namespace deve ser suficientemente clara para que o programador imediatamente entenda seu conteúdo.

O template especifica a regra geral de nomenclatura para namespaces.

<Company>.(<Product>|<Technology>)[.<Feature>][.<Subnamespace>]

Nomenclatura de Classes, Estruturas e Interfaces

Interfaces

- Utilize como prefixo a letra 'I' para indicar que o identificador representa uma interface.
- Garanta que, no caso de pares classe-interface, a diferença entre os identificadores seja apenas o prefixo 'I'.

Tipos Genéricos

- Utilize nomes descritivos para parâmetros genéricos, exceto quando uma única letra for autoexplicativa.
- Ex:
    - Repositorio<T>
    - Repositorio<T, USession>

Tipos Comuns

Tipo Base	Derivação/ Implementação
`System.Attribute`	Utilize o sufixo "Attribute".
`System.Delegate`	Utilize o sufixo "EventHandler" quando utilizado em eventos. Utilize o sufixo "Callback" quando utilizado em outros casos que não eventos.
`System.EventArgs`	Utilize o sufixo "EventArgs."
`System.Enum`	**Não** utilize o suffixo "Enum" or "Flag."
`System.Exception`	Utilize o sufixo "Exception."
`IDictionary` `IDictionary<TKey,TValue>`	Utilize o sufixo "Dictionary." `IDictionary` é um tipo específico de collection, porém utiliza este sufixo como diferenciação.

`IEnumerable`
`ICollection`
`IList`
`IEnumerable<T>`
`ICollection<T>`
`IList<T>`
	Utilize o sufixo "Collection", exceto quando nomeando Propriedades.

`System.IO.Stream`	Utilize o sufixo "Stream."
`CodeAccessPermission` `IPermission`	Utilize o sufixo "Permission

Nomenclatura de Membros de uma Classe

Métodos

- Utilize verbos ou frases verbais

Propriedades

- Utilize substantivos ou frases nominais

- Não utilize nomes de propriedades que conflitam com métodos Get:
    - public string TextWriter { get {...} set {...} }
      public string GetTextWriter(int value) { ... }
- Utilize nomes ou frases nominais no plural ao invés de utilizar o sufixo ‘Collection’.
- Nomeie booleanos com frases afirmativas.
- Considere dar à propriedade o mesmo nome de seu tipo:
    - public enum Color {...} public class Control {
        public Color Color { get {...} set {...} }
      }

Eventos

- Utilize verbos ou frases verbais
- Utilize os prefixos ‘Pre’ e ‘Pos’ quando cabível.
- Utilize os dois parâmetros chamados ‘sender’ e ‘e’.
- Utilize o sufixo ‘EventHandler’.
- Utilize o sufixo ‘EventArgs’ para sufixo de classes que encapsulam parâmetros do evento.

Atributos (Fields)

- Utilize Pascal Case.
- Utilize substantivos e frases nominais.
- Não utilize prefixos para identificar atributos estáticos ou dinâmicos.

Nomenclatura de Parâmetros

- Utilize CamelCase;
- Utilize nomes descritivos baseados no seu significado, e não no seu tipo;

PADRÕES PARA VERSIONAMENTO DO CÓDIGO

Os branchs de projeto seguirão a seguinte estrutura

Base |---- Desenvolvimento
Base |---- Homologação
Base |---- Produção

- Desenvolvimento: onde os desenvolvedores fazem todo desenvolvimento
- Homologação: resultado do merge de Desenvolvimento à base à homologação. Alterações somente para correção de bugs
- Produção: resultado do merge de homologação à base à produção. Alterações somente para correção de bugs.
- Base: Somente recebe alterações resultantes das operações de merge.
- Merge: sempre do desenvolvimento para o base – homologação ou base – produção (forward only)

Labels

Para toda publicação no IIS, seja em desenvolvimento/homologação (IIS-DES) ou em produção (luana2), deve ser criado um label no TFS. O nome da label deve seguir o padrão abaixo. Nos comentários, fazer a indicação dos workitens (bugs, task, backlogs, etc) disponibilizados naquele momento.

Padrão de nome do label: (data invertida HHmm - operação - nome de quem criou o label)

Exemplos:

- 20140318 – Merge – Alessandro
- 20140319 – Publicação – Valdair
- 20140319 1400 – Republicação – Robson

Numeração de versões

Para identificar a versão do sistema que está publicada nos servidores de desenvolvimento, homologação e produção, utilizar como padrão de número de versão a data do build do sistema invertida. Por exemplo, se o sistema foi complicado no dia 14/07/2014, o número de versão do assembly deverá ficar como 2014.07.14. Esta informação pode ser configurada no arquivo AssemblyInfo.cs do projeto Web, dentro da pasta "Properties".

Basta alterar as duas últimas linhas de código deste arquivo:

```
// You can specify all the values or you can default the Revision and Build Numbers
// by using the '*' as shown below:
[assembly: AssemblyVersion("2014.07.03")]
[assembly: AssemblyFileVersion("2014.07.03")]
```

Desta forma, é possível saber ao certo qual versão do sistema está publicada, comparando a versão do sistema com a label que foi gerada para a publicação no controle de versões.

A criação de bugs e tarefas deverão seguir as seguintes diretrizes:

- A criação de bugs deve ser realizada/aprovada pelo analista responsável pelo sistema. Bugs não devem ser direcionados diretamente aos programadores. Para isso, criar as tarefas específicas, com estimativas de horas para resolução do bug.
- Bug de produção: criar as tarefas para corrigir o bug no branch de produção (investigação, correção, testes, publicação em produção, e correção no desenvolvimento)
- Bug de homologação: criar as tarefas para corrigir o bug no branch de homologação (investigação, correção, testes, publicação em homologação, e correção no desenvolvimento)
- Criação de tarefa para execução dos Merges. Comunicar a equipe que alterações em arquivos (webconfigs, templates, ou arquivos que precisam ser copiados manualmente, anotar na tarefa de merge da semana).
- Dia da semana para efetuar Merge: todas as quintas-feiras.

Padrões de Layout e Interface

RESUMO

O objetivo deste documento é orientar os desenvolvedores sobre os padrões de layout e interface a serem adotados na criação e alteração de sistemas.

Os padrões foram elaborados visando a compatibilidade com as últimas tendências de tecnologia (mobile e web).

PADRÕES DE LAYOUT E INTERFACE

O desenvolvimento de novos softwares deve seguir o padrão de layout e interface do Projeto Modelo do TCEPR.

FRAMEWORK FRONT-END

O Projeto Modelo utiliza o Bootstrap, que consiste em um framework front-end que facilita a vida dos desenvolvedores web a criar sites com tecnologia mobile (responsivo).

Maiores informações podem ser encontradas em: http://getbootstrap.com/.

Componentes (plugins)

Além disto, o Bootstrap possui vários componentes (plugins) que auxiliam os desenvolvedores na implementação de software.

O TCEPR utiliza o plugin ACE - Responsive Admin Template (http://www.bootstraptemplates.net/ace-responsive-admin-template).

No link a seguir é possível visualizar exemplos do tema em funcionamento: http://ace.jeka.by/.

RECOMENDAÇÕES DE LEITURA

Padrões Web em Governo Eletrônico - Cartilha de Usabilidade: e-pwg-usabilidade.pdf

Padrões de Qualidade

RESUMO

O objetivo deste documento é definir os padrões de qualidade a serem seguidos no desenvolvimento de software.

Os padrões de qualidade são diretrizes a serem seguidas por toda a equipe de desenvolvimento, com o objetivo de entregar o melhor produto possível em cada entrega.

PADRÕES DE QUALIDADE

Os padrões de qualidade que cada entrega deve atender são:

1) Para toda entrega deve haver um Plano de Testes;

2) Todos os testes previstos no Plano de Teste devem ser executados (tanto os automatizados quanto os manuais);

3) Todos os testes automatizados devem ser executados e bem-sucedidos;

4) O código deve ser submetido ao *SonarQube* e atender aos parâmetros mínimos definidos.

Padrões de Segurança

RESUMO

O objetivo deste documento é definir os padrões mínimos de segurança a serem seguidos no desenvolvimento de aplicações.

A responsabilidade pela segurança de um software não deve ficar na responsabilidade de uma pessoa ou um grupo, mas sim diluída em todo o desenvolvimento. Deve fazer parte do dia a dia do Time de Desenvolvimento, pois não adianta um trabalhar para diminuir as falhas de segurança enquanto outros programam sem segurança e criam novas falhas.

PADRÕES DE SEGURANÇA

Desenvolver em Camadas

Ao desenvolver, sempre obedecer às camadas do framework utilizado. Por exemplo, nunca fazer validação na camada de interface, nem executar comando de banco em uma camada de controle.

Validar todo campo de entrada

Além das entradas do usuário, algumas entradas podem ser de outros webservices, banco, outros sites e também devem ser validadas na entrada. Outros campos podem não serem imediatamente usadas pelo sistema, mas mesmo assim devem ser validados pois podem ser usados por outros sistemas ou outras interfaces do mesmo sistema.

Validar no lado do Servidor

Várias bibliotecas ajudam a deixar a interface muito bonita e amigável, mas devem servir apenas para isso, todas as entradas devem ser validadas do lado do servidor, pois a camada de interface roda no lado do cliente e pode ser facilmente modificada.

Campos de formulário como CPF, datas, dinheiro normalmente são apenas validados utilizando algum componente na camada de interface, mas devem sempre ser validados no lado do servidor também, valide por exemplo o tamanho, formato, tipo e intervalo.

Banco de dados

Acesso ao banco deve ser sempre feito respeitando as camadas e utilizando bibliotecas especificas para acesso de dados (ex.: biblioteca ADO) evitando assim ataques de banco como SQL *injection*.

Não concatene chamadas do banco de dados pois é uma falha que também pode ser atacada com SQL *injection*.

Informações Sigilosas

Tomar bastante cautela ao mostrar informações de outros usuários/entidades. Por exemplo, em busca por CPF, se não bem definida, um usuário mal intencionado pode mostrar dados de todo o cadastro do TC.

Ocultar mensagens de erro do cliente

Quando ocorrer alguma exceção no sistema não expor a mensagem de erro para o cliente, gravar em log e mostrar uma mensagem amigável para o cliente.

Padrões

Seguir outros Padrões do desenvolvimento. Esta medida garante bastante segurança no processo de desenvolvimento porque todos os outros padrões também ajudam a manter um nível elevado de segurança.

7. MANUAIS

Neste capítulo são disponibilizados manuais, ou guias de instruções, para execução de atividades relacionadas ao processo de desenvolvimento de software.

Os manuais disponíveis são:

- Manual de Contagem de Pontos de Função do TCEPR

Manual de Contagem de Pontos de Função do TCEPR

RESUMO

O objetivo deste manual é orientar sobre a metodologia de contagem de pontos de função adotada pelo TCEPR.

MANUAL

Manual de Contagem de Pontos de Função do TCEPR.docx