**ANEXO 01**

**REQUISITOS DA PRESTAÇÃO DE SERVIÇOS**

**1. Requisitos de negócio da área requisitante**

1.1 Atendimento às atividades inerentes à manutenção corretiva, perfectiva, adaptativa e evolutiva dos sistemas de informação do TSE e realização de testes de software.

1.2 Evolução das bases e consultas de dados gerenciais e estratégicos por meio de plataforma de *Business Intelligen*ce (BI) e administração corporativa de dados.

1.3 Desenvolvimento de sistemas de informação para o TSE.

1.4 Prestação de suporte aos usuários dos sistemas de informação quanto a funcionalidades e usabilidade do software, sistematizando o atendimento.

1.5 Planejamento, preparação e condução de eventos de testes e condução de testes de integração de sistemas críticos.

**2. Requisitos de arquitetura tecnológica**

2.1. A Contratada deverá atender necessariamente ao arcabouço arquitetural utilizado para processos de desenvolvimento e manutenção de sistemas do TSE:

1. Arquitetura corporativa de sistemas Web Java e Angular, com utilização obrigatória das tecnologias formalmente suportas pela infraestrutura computacional do TSE e do pipeline de entrega contínua, salvo quando a utilização do pipeline for tecnicamente inviável;
2. Arquitetura especializada em sistemas embarcados utilizando a linguagem de programação C++ para o processo eleitoral;
3. Arquitetura de sistemas desenvolvidos em *Apex*, majoritariamente relacionados a relatório, e sistemas transacionais utilizando o motor de fluxo JBPM;
4. Arquiteturas de gerenciamento de conteúdo, por exemplo, *Zope* e *Plone*; e
5. Manutenção de sistemas legados, por exemplo, *Delphi* e *Cobol*.

**3. Requisitos de testes**

3.1 Todos os produtos de software gerados no apoio ao desenvolvimento feito pela Contratada deverão ser entregues devidamente testados, não havendo lançamento apartado das atividades de testes nas ordens de serviços de evolução e sustentação de software;

3.1.1 Os serviços de planejamento, preparação e condução dos eventos de testes e os serviços de testes de integração de sistemas críticos serão objeto de ordem de serviço específica.

3.1.2 Testes de outros sistemas adotados ou desenvolvidos diretamente por profissionais do quadro da Justiça Eleitoral ou sob sua supervisão e sem o apoio da Contratada poderão ser demandados por ordem de serviço específica.

3.2 Poderão ser demandados testes em diversas técnicas e níveis, como, por exemplo, testes de unidade, de integração, de caixa-preta, de caixa-branca, regressão, estresse, aceitação, exploratório.

3.3 Caso seja do interesse do TSE, os processos de testes poderão ser automatizados a partir da comparação dos resultados esperados com os resultados reais, da configuração das pré-condições de teste e de outras funções de controle e relatório de testes.

**4. Requisitos para as atividades de testes de integração e de eventos de testes**

4.1 Para efeitos dos serviços contratados, entende-se por evento de teste o esforço coordenado, realizado em um prazo determinado, envolvendo as áreas de TI e de negócio do TSE e tribunais regionais, com o objetivo de executar testes nos sistemas com a simulação de diversas situações de uso (cenários de teste) e validando a estabilidade dos produtos de software.

4.2 Os serviços necessários para viabilização dos eventos de testes são a identificação e especificação das necessidades de testes, dos requisitos de testes, dos participantes, dos recursos materiais e dos cenários e dados de testes. Ainda, a escrita dos roteiros de teste, geração da massa de dados, suporte e configuração do ambiente e dos sistemas de apoio ao evento. Todas essas atividades culminam na produção do conjunto de cenários de testes validados e aptos à utilização durante o evento de teste.

4.3 Os testes de integração de sistemas críticos a partir da liberação dos sistemas pelos times de desenvolvimento, com a verificação das funcionalidades que perpassam as fronteiras de cada sistema.

**5. Requisitos quanto à evolução dos painéis de dados gerenciais e estratégicos:**

5.1 Evoluir os projetos de Data Mart e Business Intelligence (BI) atualmente implantados no TSE, atendendo às suas novas necessidades.

**6. Quanto aos serviços de suporte:**

6.1 Prestar suporte aos usuários dos sistemas de informação quanto às suas funcionalidades e à usabilidade do software, até que essa atividade seja repassada ao *Service Desk*.

6.2 Prestar suporte às demais equipes de TI do TSE no tocante às características e necessidades de infraestrutura do sistema mantido.

6.3 Realizar tarefas de apoio na configuração, ajustes, instalação e implantação de sistemas em harmonia com a área de infraestrutura de TI, permitindo o bom funcionamento dos sistemas mantidos.

**7. Quanto aos requisitos não funcionais:**

7.1 Em virtude da ampla abrangência da contratação, ou seja, atendimento a múltiplos sistemas legados, com arquiteturas distintas e de diversos tribunais eleitorais, não se faz possível esgotar neste instrumento todos os requisitos não funcionais. Assim, a Contratada poderá, durante o processo de abertura da Ordem de Serviço (OS), receber os requisitos solicitados.

7.2 Todos os produtos e serviços de apoio à manutenção e aos testes a serem realizados deverão estar em conformidade com os padrões, procedimentos e metodologias definidas pelo TSE.

7.3 No que diz respeito ao processo de desenvolvimento, o framework de desenvolvimento do TSE se denomina AgiTSE. O AgiTSE não possui um guia de referência, mas define diretrizes práticas de condução do ciclo de desenvolvimento a partir modelos ágeis de mercado. O detalhamento operacional é feito pela abordagem especializada da equipe responsável pelo AgiTSE junto com o time de desenvolvimento.

7.3.1 As abordagens, técnicas e práticas ágeis constantes no AgiTSE estão em constante exercitação, podendo ser alteradas ou incrementadas ao longo desta contratação, com base em modelos de mercado. O AgiTSE, desta forma, não é uma metodologia monolítica, mas um framework dinâmico de melhores práticas de desenvolvimento ágil. Um resumo deste framework está especificado no item 11 deste anexo – AgiTSE - Framework de Desenvolvimento de Software do TSE.

7.3.2 As alterações mais significativas no AgiTSE deverão ser formalizadas à Contratada com antecedência mínima de 30 dias corridos.

7.4 A critério do TSE, pode-se estabelecer procedimento operacional obrigatório, desde que formalmente comunicado à Contratada com, no mínimo, 30 dias corridos de antecedência.

7.5 É obrigação da equipe técnica da Contratada manter atualizado o andamento das atividades a ela alocadas nas ferramentas do acompanhamento da execução dos serviços.

7.6 Sempre que possível, os serviços de apoio ao desenvolvimento e à sustentação deverão observar as políticas, premissas e especificações técnicas definidas no Modelo Nacional de Interoperabilidade (MNI) do Poder Judiciário; quando houver a necessidade de utilização de certificação digital, devem aderir às regulamentações da Infraestrutura de Chaves Públicas Brasileiras (ICP-Brasil) e observar, quando aplicáveis, as orientações, premissas e especificações técnicas e funcionais definidas no Modelo de Requisitos para Sistemas Informatizados de Gestão de Processos e Documentos do Poder Judiciário (Moreq-Jus).

7.6.1 A necessidade de atendimento a qualquer um desses modelos será definida pelo TSE no momento da abertura da OS.

**8. Requisitos de manutenção e sustentação**

8.1 O apoio à manutenção e sustentação contempla a manutenção corretiva, perfectiva, adaptativa e evolutiva além do serviço atendimento a chamados (incidentes, requisições, problemas e mudanças) de usuários e outras áreas técnicas dentro dos níveis de serviço acordados.

8.2 A manutenção visa identificação e correção de pontos falhos, melhoria no desempenho, segurança e qualidade do sistema, adequação a novas regras de negócio e adequação ao parque computacional e soluções de infraestrutura do TSE.

8.3 O apoio à sustentação compreende essas manutenções e a prestação de suporte aos usuários e unidades técnicas, em atividades como, por exemplo, esclarecimento de dúvidas, extração de informações, orientações, etc.

**9. Macro Requisitos Tecnológicos da Solução de TI**

9.1 Ferramentas de apoio à execução contratual e de gestão de serviços

9.1.1 A Contratada deverá adequar-se aos sistemas de gestão de desenvolvimento e de suporte de TI, além dos sistemas administrativos disponíveis no ambiente do Contratante para o processamento de informações, produtos e serviços.

9.1.2 As soluções de controle de prestação de serviços, eventualmente fornecidas pela Contratada e utilizadas nas instalações do Contratante, deverão estar aderentes aos padrões tecnológicos de execução do Contrato e, preferencialmente, integradas aos sistemas de controle do Contratante, a seguir relacionados:

a) GDS — Sistema de Gestão de Demandas de Serviço;

b) SIGA — Sistema de Gestão de Ordens de Serviço;

c) MAPTI — Sistema de Mapeamento de Perfis dos Profissionais de TI;

d) GSTI — Sistema de Abertura e Gerenciamento de Chamados;

e) REDMINE — Sistema de Gerenciamento de Projetos;

f) Hórus — Sistema de Informações Corporativas de Softwares;

g) Git — Sistema de Controle de Versões Distribuído de Softwares;

h) TortoiseSVN — Ferramenta de Subversionamento para Microsoft Windows;

i) Bugzilla – Ferramenta de Subversionamento para Microsoft Windows;

j) Testlink – Ferramenta para Gestão de Casos de Teste; e

k) JIRA – Ferramenta integrada que substituirá as ferramentas apontadas nos itens (e), (i) e (j).

9.1.1.2 Além dos sistemas citados, outros poderão ser incorporados pela Contratante ao longo da execução contratual.

**10. Transferência do conhecimento**

10.1 A Contratada deverá propiciar o registro e a transferência de conhecimento aos servidores do Contratante durante toda a execução contratual e no processo de finalização contratual, garantindo uma eficiente gestão de transferência dos conhecimentos.

**11. AgiTSE - Framework de Desenvolvimento de Software do TSE**

11. A seguir é apresentado um breve resumo da framework de desenvolvimento de software – AgiTSE – utilizada pelo Contratante.

**Perspectiva: Papéis**

* **Product Owner**

Essa pessoa é quem tem a visão do que sua equipe fará, produzirá ou realizará. Ela leva em consideração os riscos e as recompensas, o que é possível e o que pode ser feito. É o indivíduo que exerce o papel de cliente em um processo de desenvolvimento de software. É uma pessoa e não um comitê. O product owner (PO) é o responsável por gerenciar e priorizar o backlog do produto, além de determinar quais as características de maior valor e que devem estar contidas no produto final.

Segundo o cocriador do Scrum, Jeff Sutherland, as características essenciais de um product owner são:

**Um**, ele precisa ter conhecimento sobre o campo, ou seja, deve entender o processo que a equipe está executando bem o suficiente para saber o que pode ser realizado e, tão importante quanto isso, o que não pode ser feito.

**Dois**, ter o poder de tomar decisões. Assim como a gerência não deve interferir na equipe, ele deve receber carta branca para tomar decisões sobre qual será a visão do produto e o que precisa ser feito para chegar lá.

**Três**, estar disponível para a equipe a fim de explicar o que precisa ser feito e por quê. Em última instância, ele é o responsável pelo backlog, por essa razão é necessário que haja um diálogo constante com o time.

**Quatro**, ele precisa ser responsável pelo valor que será agregado com o esforço do time.

* **Time de Desenvolvimento**

Profissionais comprometidos que realizam o trabalho de entregar um incremento do produto ao final de cada sprint atendendo ao critério de pronto definido pelo product owner. Desejável que o time seja composto de 3 a 9 indivíduos, ou seja, pequeno o suficiente para se manter ágil e grande o suficiente para completar um trabalho significativo dentro da sprint.

Cabe destacar algumas características importantes de times de desenvolvimento:

**Transcendentes**: Eles têm noção de propósito que vai além do comum. Esse objetivo lhes permite ultrapassar o trivial e alcançar o extraordinário. A decisão de não se contentar com a média, mas de ser grande, muda por si só a forma como o time se vê e o que é capaz de realizar.

**Autônomas**: Os times são auto-organizados e se autogerenciam. Eles podem decidir como executar o trabalho e têm o poder de fazer com que suas decisões sejam cumpridas.

**Multifuncionais**: Possuem todas as habilidades necessárias para completar um projeto. E essas habilidades alimentam e reforçam umas às outras.

**Perspectiva: Qualidade**

* **Cobertura** **de testes**

Cobertura de testes é uma medida utilizada para indicar a proporção devidamente testada do código-fonte de um software. Ela pode ser medida por tipo de teste (unitário, manual, integração, etc.) ou por agregação da proporção dos tipos de testes utilizados no projeto.

* **Funcionalidade**

Funcionalidade descreve a capacidade de um software em atender as necessidades explícitas e implícitas. Ela é mensurada por meio de suas subcaracterísticas: adequação, acurácia, interoperabilidade, segurança de acesso e conformidade relacionada à funcionalidade.

* **TDD**

*Test Driven Development* ou Desenvolvimento Orientado por Testes trata o *software* baseado em testes escritos antes mesmo da implementação do código-fonte para o incremento de *software* resultante na *sprint*. Basicamente, o TDD consiste em pequenos ciclos de repetições, nos quais um teste é criado para cada funcionalidade do sistema, este teste, por sua vez, falhará em razão da não existência de código-fonte correspondente. Após isso, a implementação da funcionalidade ocorre para fazer o teste unitário ser executado com sucesso.

Entretanto, a regra acima é apenas o início. Considere as três leis abaixo:

* Primeira Lei: não se deve escrever o código de produção até criar um teste de unidade de falhas.
* Segunda Lei: não se deve escrever mais de um teste de unidade do que o necessário para falhar, e não compilar é falhar.
* Terceira Lei: não se deve escrever mais códigos de produção do que o necessário para aplicar o teste de falha atual.

Segundo Robert Martin, essas três leis colocam o desenvolvedor numa rotina que talvez dure trinta segundos. Os testes e o código de produção são escritos juntos, com os testes apenas alguns segundos adiantados. Se trabalhado dessa forma, se criariam dezenas de testes a cada dia, centenas a cada mês e milhares a cada ano; os testes de unidade cobririam praticamente todo o código de produção.

Referência: Martin, R. (2007). Professionalism and TDD, IEEE Software (Vol.24, Nº 3) pp. 32-36

* **BDD**

*Behavior Driven Development* – Desenvolvimento Guiado por Design ou Desenvolvimento Orientado a Domínio – visa integrar regras de negócios com linguagem de programação, focando o comportamento do *software*. Além disso, pode-se dizer também, que BDD é a evolução do TDD, isso porque é uma técnica voltada para o comportamento da aplicação, por meio da qual usuários do produto também podem compreender e contribuir com a escrita dos testes.

* **Métodos Single-source**

Single-source trata-se da inserção de textos explicativos, em forma de um comentário especial, para apresentar o escopo de uma classe ou um método, fundamentais para facilitar o entendimento de outras pessoas e orientar sobre sua utilização adequada.

* **Boas práticas de codificação**

As boas práticas de codificação são técnicas que ajudam o código a ser mais legível, de fácil compreensão e manutenção, colaborando assim para que o ciclo de desenvolvimento de sistemas ocorra de maneira mais ágil.

**Aprender a criar códigos limpos é uma tarefa árdua e requer mais do que o simples conhecimento dos princípios e padrões.** O desenvolvedor deve ter a "sensibilidade", ao revisar um código já existente ou escrever uma nova funcionalidade, para ver alternativas de implementação que melhor se adequem à arquitetura do projeto.

* **Teste de segurança**

Testes de segurança são aplicados para **fortalecer a confiabilidade** e determinar se a segurança do produto de software está satisfatória e de acordo com os requisitos do cliente.

Eles visam garantir que o software se comporta adequadamente diante de tentativas ilegais de acesso, buscando **identificar possíveis vulnerabilidades e falhas**.

A aplicação é realizada mediante testes dos mecanismos de proteção embutidos na aplicação.

* **Modelagem**

**O modelo é a representação de determinada realidade**, por exemplo, a planta baixa de uma casa.

A modelagem de sistemas consiste na **criação abstrações acerca de um produto**, **serviço ou processo**, de maneira que cada visão ou perspectiva diferente do sistema seja observada.

Emprega-se a notação gráfica para facilitar o entendimento das funcionalidades do sistema e promover uma **eficiente comunicação com os clientes**.

* **Markdown no Gitlab**

Segundo os próprios criadores da linguagem, John Gruber e Aaron Swartz, o Markdown é uma ferramenta de conversão de textos para HTML, voltada a escritores da web. Permite escrever usando um formato de texto simples, de fácil leitura e escrita, e com possibilidade de conversão em arquivos XHTML (ou HTML) estruturalmente válidos. Sendo suportado por cada vez mais ferramentas, com ele é possível, de maneira limpa, precisa e com codificação mínima, marcar títulos, textos, imagens, códigos-fontes, fórmulas matemáticas, *links*, tabelas, listas e diagramas, muito melhor que escrever diretamente em HTML.

* **Apoio à segurança**

O Apoio à segurança consiste em processos de cálculos e análise de métrica de segurança cuja utilização é capaz de fornecer dados para que os gestores de TI possam avaliar e **melhorar o nível de segurança da informação** em suas organizações.

Devido aos crescentes números de incidentes de segurança que preocupam as corporações e os governos, esta é uma medida imprescindível nos dias atuais, uma vez que a interrupção de serviços importantes e o roubo de dados sigilosos causam **grande prejuízo a toda a sociedade**.

* **Qualidade de código-fonte**

Em uma visão ampla, trata-se do processo de busca da conformidade a requisitos funcionais e de desempenho declarados explicitamente, padrões de desenvolvimento claramente documentados e critérios de qualidade.

Quando as equipes de desenvolvimento enfatizam em entregar código-fonte com qualidade, possivelmente, ela **reduzirá a quantidade de retrabalho**, resultando em custos menores e **menor tempo de disponibilização do produto ao cliente**, que, no caso da Justiça Eleitoral, é a própria sociedade brasileira.

Obter um código de alta qualidade é uma atividade complexa e deve ser apoiada pelo uso de ferramentas e técnicas apropriadas, **tais como avaliação dos indicadores sobre cobertura de testes, análise estática de código, integração contínua, produtividade e quantitativo de defeitos reportados e corrigidos**. O TSE dispõe ainda do SonarQube, que se mostra como alternativa para a gestão da qualidade de código-fonte e pode dar visibilidade à atual situação do *software* monitorado.

* **Casos de abuso**

Os casos de abuso são especificações que descrevem o mau uso, intencional ou não, do *software*. O emprego desse recurso consiste na adaptação da técnica de modelagem orientada a objetos, casos de uso (*use* *cases*), para a captura e análise dos requisitos de segurança de maneira simplificada. Uma vez elaborados, os casos de abuso **promovem o melhor entendimento dos problemas de segurança** e ajudam a encontrar soluções para eles.

* **Revisão de Código**

Processo de revisão colaborativo objetivando a melhoria do código-fonte.

A revisão de código pode ocorrer com a participação de outros membros do time de desenvolvimento e pode ser assistida por ferramenta de integração e análise do código.

* **Teste unitário**

Teste referente a menor parte de um componente de *software* (rotinas, módulos ou fragmentos de código).

Esse tipo de teste tem como objetivo verificar, em um cenário conhecido, se existem divergências no funcionamento esperado da unidade de *software*.

* **Processo de Teste Ágil**

Teste ágil são as práticas de teste exercidas no modelo de desenvolvimento ágil, executadas desde o inicio do projeto, de forma preventiva, contínua e integradas às demais atividades necessárias para entregar um produto de *software* de alta qualidade. O teste ágil promove a transparência, colaboração e melhoria contínua durante o ciclo de desenvolvimento.

* **Release Notes**

*Release* *Notes* são documentos descritivos que relatam o conteúdo de uma entrega de um conjunto de *software*.

Um documento de *release* *notes* pode conter informações de mudanças, novas funcionalidades, correções de erros ou até mesmo notificação de erros conhecidos pelo mantenedor em uma versão específica do *software*.

* **Manutenibilidade**

Manutenibilidade é a capacidade de um *software* de ser modificado.

A manutenibilidade refere-se à analisabilidade, modificabilidade, estabilidade/ testabilidade do *software* e à consequente facilidade, segurança, precisão e custo em realizar manutenções no *software*, sejam elas em razão de correções, melhorias ou adaptações.

Esta característica é de interesse especialmente de desenvolvedores e não deve ser confundida com a possibilidade de configurar o *software*.

* **Teste de integração de componentes**

O Teste de integração de componentes é utilizado para avaliar a conformidade quanto ao funcionamento integrado de diferentes partes de um *software*, incluindo suas interfaces de comunicação e dependências.

* **Teste de integração de sistemas**

O Teste de integração de componentes é utilizado para avaliar a conformidade quanto ao funcionamento integrado de diferentes partes de um *software*, incluindo suas interfaces de comunicação e dependências.

* **ATDD**

*Acceptance Test-Driven Development* (ATTD) ou Desenvolvimento Orientado a Testes de Aceitação é uma prática que envolve membros da equipe com diferentes perspectivas (cliente, desenvolvimento, teste), os quais, por sua vez, colaboram para escrever testes de aceitação antes de implementar a funcionalidade correspondente.

Os testes de aceitação representam o ponto de vista do usuário e funcionam como uma forma de especificar os requisitos que descrevem como o *software* deve se comportar.

* **Programação pareada**

Programação pareada é uma técnica de desenvolvimento de *software* que consiste em dois programadores trabalhando em uma mesma parte do código-fonte. Na programação pareada os programadores desempenham dois papéis: o controlador – responsável por escrever o código-fonte; e o navegador – responsável por revisar e planejar as ações.

* **Refatoração de Código**

Refatoração de código é uma técnica para reestruturar um *software* existente de forma controlada, potencialmente melhorando o seu *design*, funcionamento e manutenibilidade. O processo de refatoração geralmente consiste em executar múltiplas pequenas alterações, sem criar novas funcionalidades ou alterar o comportamento esperado do *software*, até que o conjunto dessas modificações representem uma melhoria relevante. O teste contínuo das alterações é fundamental para evitar mudanças inesperadas durante o processo de refatoração de determinado *software*.

* **Especificação Por Exemplo – EPE**

A Especificação Por Exemplo (EPE) é uma documentação elaborada conjuntamente com diferentes perfis de usuário (ao menos o time de desenvolvimento e o PO) com o objetivo de descrever os requisitos do sistema baseando-se nos comportamentos que o sistema deve apresentar e na visão do usuário sobre a aplicação.

Deve-se atentar para que o foco esteja sempre na razão da criação do código e não nos detalhes técnicos de desenvolvimento – ou seja, o time deve ter ciência de que problema resolverá. Sua elaboração ocorre paralelamente ao desenvolvimento e ao levantamento de requisitos.

É um documento vivo, que utiliza uma linguagem simples e padronizada que pode ser entendida por todos os envolvidos e, principalmente, **servirá de insumo para a implementação dos testes automatizados do sistema – seguindo os conceitos apresentados pelo BDD**. É composto basicamente por duas partes:

* Funcionalidade: contendo a definição da funcionalidade a ser implementada ou história de usuário, e
* Cenários: validações do requisito que farão o usuário aceitar a funcionalidade.

O padrão de palavras-chave da estrutura da EPE facilita a conversão da especificação para o ambiente de testes e outras automatizações.

|  |  |  |
| --- | --- | --- |
| **Palavra-chave** | **Função** | **Obrigatório** |
| Funcionalidade | Descreve a história de usuário (para quem, por que e a finalidade) | Sim |
| Contexto | Apresenta as condições iniciais da funcionalidade. Pré condições, dados, situações | Não |
| Cenário | Descreve um fluxo que será executado apenas uma vez | Sim, ao menos um ou um esquema |
| Esquema do Cenário | Descreve um fluxo que será executado inúmeras vezes | Não, se houver um cenário |
| Exemplos | Define os diferentes valores que serão utilizados na repetição do esquema do cenário | Sim, quando tiver esquema |

Conforme o quadro acima, toda funcionalidade deve ter ao menos um cenário (ou esquema de cenário).

Cada cenário apresenta um título e uma sequencia de passos que representam as interações do usuário com o *software*. Os passos são representados pelas palavras-chave abaixo:

|  |  |  |
| --- | --- | --- |
| **Palavra-chave** | **Função** | **Obrigatório** |
| Dado/Dada | Especifica as pré-condições do cenário | Não |
| Quando | Descreve os eventos que devem ocorrer para a execução do cenário. Geralmente, representa a ação do usuário para iniciar o Cenário | Sim |
| Então | Especifica as pós condições da execução do cenário. Pode ter mais de um resultado e deve-se utilizar o conectivo "E" | Sim |
| E | Utilizado para dar continuidade aos passos (Dado, Dada, Quando, Então). Tem sentido aditivo | Não |
| Mas | Utilizado para dar continuidade aos passos (Dado, Dada, Quando, Então). Tem caráter de exceção | Não |

Cabe ressaltar que há diferenciação entre as letras maiúsculas e minúsculas.

* **Processo Elementar**

O Processo Elementar (PE) é uma métrica de mensuração de *software* customizada para o TSE e adaptada da metodologia de Análise de Pontos de Função (APF), com o objetivo de definir o tamanho funcional do escopo de uma aplicação e com isso remunerar seu desenvolvimento.

O PE pode ser considerado como uma simplificação da APF, visto que não considera questões de complexidade, funções de dados, classificação e outros conceitos ou cálculos da técnica original.

Ademais, alguns cenários de medição de funcionalidades definidas na APF foram customizados nesta métrica interna de forma a atender a realidade do TSE.

Como o próprio nome diz, esta técnica considera o Processo Elementar (PE) como forma de calcular o tamanho funcional do *software*. Mas o que é o processo elementar em si?

Segundo o CPM (*Counting Practices Manual*) do IFPUG, é a menor unidade de atividade reconhecida pelo usuário (pode ser uma tela, um relatório, um cadastro, desde que possua os requisitos para ser um PE).

**Perspectiva: Ferramentas e Técnicas**

* **Integração contínua (GitLab, Sonar, Artifactory, Jenkins)**

É uma prática que automatiza a compilação (build) dos códigos-fontes adicionados ao repositório central em momentos distintos e por desenvolvedores diferentes.

Trata-se de uma boa prática que visa maior qualidade do produto entregue, pois permite integração periódica do software e extração de informações importantes que podem indicar falhas de codificação, vulnerabilidades e incompatibilidades com a arquitetura corporativa.

* **Selenium**

O Selenium é uma suíte de ferramentas utilizadas para automatizar ações em navegadores web. Amplamente utilizado na construção e execução de testes de software automatizados, ele é composto de ferramentas para prototipação rápida de scripts de testes (Selenium IDE), codificação de testes automatizados em diversas linguagens (Selenium WebDriver) e plataforma de execução paralela e multiambiente de testes (Selenium-Grid).

* **Angular**

Angular é uma plataforma para aplicações web que integra templates declarativos, injeção de dependência e ferramentas de apoio. No TSE, a Arquitetura de Referência Derivada AngularJS estabelece as diretrizes para a utilização do Angular no desenvolvimento de software.

* **Redmine**

O Redmine é um software gratuito de código aberto utilizado no apoio ao gerenciamento de projetos. Conhecido por sua flexibilidade, o Redmine possibilita a parametrização de processos e funcionalidades se adaptando a realidade dos times.

Além disso, permite a integração com diversas outras ferramentas de apoio ao desenvolvimento, como Git, Svn, Jenkins, entre outros.

* **Gitlab**

O GitLab é uma plataforma de apoio às práticas de DevOps, baseada no sistema de controle de versões Git, que inclui recursos para o gerenciamento do projeto, a comunicação do time de desenvolvimento, a elaboração e publicação de documentação e a integração contínua.

* **Prototipação**

A prototipação é uma técnica que permite a construção de uma versão preliminar do software para facilitar o entendimento dos requisitos e avaliar a proposta de software.

* **Sonar**

O Sonar é uma ferramenta para inspeção contínua da qualidade do código-fonte, com suporte a múltiplas linguagens e plugins que facilitam sua customização e integração com outras ferramentas do ciclo de desenvolvimento. Ele apoia o time de desenvolvimento indicando inconformidades e verificando uma variedade de critérios de qualidade de maneira automática e contínua.

* **É, não É; Faz, não Faz**

A atividade "É, não É; Faz, não Faz" ajuda a definir um tópico, seja um produto, um time ou uma estratégia.

Por vezes, é mais fácil descrever algo pelo que tal coisa não é ou deixa de fazer.

Esta técnica ajudará de forma lúdica capturar ideias e clarear expectativas.

* **MVP**

O mínimo produto viável, Minimum Viable Product - MVP, é a versão mais simples do produto que pode ser lançada com a quantidade mínima de esforço e desenvolvimento. Trata-se de uma prática da Administração de Empresas que consiste basicamente em lançar um novo produto ou serviço com o menor investimento possível, para testar o negócio antes de aportar grandes investimentos.

A aplicação do MVP no desenvolvimento de software acaba sendo de extrema importância, visto que podemos encarar um software como um produto lançando-o no mercado em versão que conste das suas funcionalidades prioritárias. Posteriormente, é possível realizar maiores investimentos ou até mesmo saber, já de início, se este será um produto aceito pelo mercado ou não.

* **Monopoly**

O Monopoly faz uso da mesma ideia trazida pelo jogo de tabuleiro chamado Banco Imobiliário, e a principal ideia aqui é trabalhar com notas de dinheiro para que o cliente possa priorizar o que realmente agrega valor ao negócio. Tal ferramenta pode ser usada para priorizar entregáveis de uma forma geral, como: Histórias, Temas, Épicos, Casos de Uso e até mesmo realizar a priorização entre projetos diferentes

* **Relacionamento com o PO**

Comunicação é a principal característica do product owner, pois ele deve ser capaz de deixar todos os interessados no produto sempre alinhados e também colaborando com novas funcionalidades e tendências de mercado. Fora tal ponto, teremos também a necessidade da comunicação intensa entre ele e o time scrum.

No TSE há diversos tipos de relacionamentos com o product owner devido ao elevado número de perfis distintos. Existem pessoas que acumulam esse papel com outros, por exemplo: são chefes de seção do próprio time de desenvolvimento. Alguns possuem um perfil mais técnico, outros mais gerencial; alguns são mais solícitos e disponíveis, outros possuem uma agenda apertada e se tornam indisponíveis. Para cada um, deve ser estabelecido um tipo de comunicação que seja mais eficiente.

* **Relacionamento com o scrum master**

O scrum master é a ponte de ligação entre o product owner e o time de desenvolvimento. Colocamos ele como o nosso principal facilitador, quem descobre mecanismos para elevar o potencial de produtividade do time. É importante ter em mente que o scrum master não é um gerente, um chefe, ele age como um líder servidor, um facilitador.

Nesse sentido, o scrum master deve estar sempre à disposição, tanto do time de desenvolvimento quanto do product owner, apoiando o andamento de todo o projeto. Veremos então que ele tem como papel garantir a colaboração e comunicação do time scrum.

* **Relacionamento com o gestor**

Um dos principais fatores críticos de sucesso em projetos que trabalham com abordagens ágeis é a comunicação e, mais importante ainda, a comunicação de quem está desenvolvendo o produto final com quem está precisando deste produto, ou seja, a comunicação entre o time de desenvolvimento e o gestor/cliente.

Um papel que se torna essencial, mas que não exclui a necessidade de integração entre gestor/cliente e o time de desenvolvimento é o do product owner.

* **Integração contínua (Controle de acesso)**

Apresenta as políticas da Secretaria de Tecnologia da Informação do TSE para a definição de controle de acesso às ferramentas de integração contínua. Trazemos para vocês também um conteúdo adicional sobre as principais ferramentas que poderão ser utilizadas e que já estão implantadas no TSE para garantir a execução da integração contínua como: GitLab, SonarQube e Jenkins.

* **Versionamento de artefatos**

Artefatos de software podem ser de diversos tipos - imagens de containers docker, executáveis, bibliotecas, entre outros.

A política para o versionamento dos artefatos de software define uma estrutura adequada para o registro das versões de um artefato de software.

No TSE, o Artifactory e Nexus são os gerenciadores de repositórios utilizados para armazenar, versionar e distribuir esses artefatos de software.

* **Planning Poker**

Técnica "gamificada", baseada em consenso para nivelar o entendimento do time scrum sobre histórias de usuários (HUs) que serão realizadas. Usada para estimar o esforço ou tamanho relativo das metas de desenvolvimento de software. Entretanto, o principal benefício gerado pelo uso da técnica é o esclarecimento de características das HUs antes de sua implementação para todos os envolvidos do time.

Normalmente, utiliza-se a sequência de Fibonacci (1, 2, 3, 5, 8, 13, 21) atribuindo uma nota para cada HU e comparando-as entre si de acordo com as perspectivas de dificuldade, complexidade e esforço. Além disso, podem ser utilizados também outros tamanhos relativos, por exemplo, medida de roupas (BabyLook, PP, P, M, G, GG).

Recomenda-se uma explicação inicial e três rodadas por HU, sendo que as avaliações polarizadas, menor e maior notas, são explicadas pelos participantes que pontuaram dessa maneira, detalhando o seu ponto de vista e agregando mais informação para próxima rodada.

Quando a nota de uma HU é avaliada pelo time como muito alta, este é um indicativo de que esta deve ser quebrada, pois pode ser que ela seja um tema ou um épico.

* **Gráfico de Burndown**

Gráfico utilizado pelas equipes scrum para representar diariamente o progresso do trabalho em desenvolvimento dentro da sprint.

* **Apoio à integração contínua**

A integração contínua é uma prática de desenvolvimento de software que consiste na integração frequente (contínua) das alterações do código-fonte, geralmente acompanhadas da execução de testes, procedimento de compilação, empacotamento e publicação do software.

A Seprop pode apoiar no uso da integração contínua no seu projeto, avaliando o impacto de levar o projeto para o Jenkins Corporativo e auxiliando nos trabalhos decorrentes.

**Perspectiva: Ciclo de Desenvolvimento**

* **Equipe de Segurança da Informação**

Equipe responsável por apoiar a melhoria da segurança dos sistemas do TSE, suas ações permeiam todo o ciclo de desenvolvimento: criação de casos de abuso e identificação de normas de segurança; reutilização de soluções corporativas; análise estática de código, análise dinâmica e teste de invasão; *hardening* de infraestrutura.

São responsáveis por auditar, criar e medir indicadores de segurança, além de definir a estratégia de capacitação da força de trabalho em segurança da informação no âmbito do Tribunal.

* **Analista de Negócio**

Indivíduo que atua como **facilitador da comunicação entre as unidades de negócio**, obtendo as reais necessidades das partes interessadas, não apenas os seus desejos expressos, definindo e validando as soluções que satisfaçam as necessidades de negócio, metas ou objetivos.

“*Um Analista Negócio é qualquer pessoa que exerça atividades de Análise de Negócio, não importando qual seja seu cargo, função ou papel*”, segundo o Guia BABok®. Sendo importante ressaltar que **esse profissional não é um Analista de Requisitos**.

* **Equipe de Arquitetura Corporativa**

Responsável pela definição e implantação de padrões arquiteturais, realização de POCs e prospecção de soluções tecnológicas mais adequadas e eficientes. Realiza auditorias e análise de desempenho e segurança de aplicações, configuração e otimização de projetos e servidores, além de prestar mentorias e suporte ao desenvolvimento para qualidade de código bem como avalia conformidade.

Responsável também pela administração de ferramentas para Integrações e Entregas Contínuas (Git/Jenkins/Sonar/Artifactory/WSO2/FlywayDB), desenvolvimento e manutenção de componentes/arquiteturas-base/arquétipos para aplicações back-end e front-end.

* ***Scrum Master***

Responsável por assegurar que a equipe esteja **utilizando corretamente as práticas do *framework* *Scrum***, motivada e focada na meta da *sprint* (iteração).

O *scrum master* é um **facilitador para o time *scrum*** cuja função primária é remover impedimentos à capacidade da equipe de entregar o objetivo da *sprint*.

* **Definição de Itens do *backlog***

O *backlog* é uma lista ordenada de tudo que é conhecido ser necessário no produto. É a única origem dos requisitos para qualquer mudança a ser feita no desenvolvimento do produto. A definição dos seus itens deverá ser realizada por todos os membros do time *scrum*, sendo de responsabilidade do *product* *owner* a sua priorização e decisão referente aos itens que deverão ou não entrar nas próximas *sprints*.

Essa lista existe e evolui ao longo de toda a vida do produto, ela é o mapa a ser seguido. Só existe um *backlog* do produto. Isso significa que o *product* *owner* precisa tomar decisões de como priorizar as tarefas ao longo de todo o projeto e também consultar todos os *stakeholders* e o time de desenvolvimento para certificar de que está representando tanto o que as pessoas querem quanto o que é possível ser feito.

* **Kanban como *framework***

O Kanban é um sistema de controle que ajuda a organizar as tarefas que compõem um processo para que elas sejam realizadas e entregues como em uma linha de produção.

* **Refinamento**

O refinamento das histórias de usuários, também conhecido como Grooming, tem como propósito amadurecer o entendimento dos itens por parte do time com o auxílio do *product owner* no esclarecimento de dúvidas e do real valor agregado com a implementação da funcionalidade ou serviço.

* ***Sprint***

É o coração do *Scrum*, possui um tempo definido (*time-box*), que varia de acordo com time de desenvolvimento, normalmente **períodos menores do que um mês**, preferencialmente entre uma e três semanas.

Nesse tempo é onde o trabalho do time é efetivamente realizado. Ao final de cada *sprint* tenha algo pronto - algo que possa ser usado e vai trazer valor reconhecido pelo cliente.

* **Reunião Diária**

Evento diário com tempo definido (*time-box*) de 15 minutos para o time de desenvolvimento. A reunião diária é realizada em todos os dias da *sprint*.

Neste evento o time de desenvolvimento deve **manifestar os impedimentos existentes**, apresentar, de forma sucinta, o trabalho que estará atuando até a próxima reunião diária e a evolução do trabalho da *sprint* nas últimas 24 horas.

Esse encontro ajuda a equipe inteira a saber exatamente em que ponto as coisas estão na *sprint*. Todas as tarefas serão completadas a tempo? **Há oportunidades para auxiliar colegas com dificuldades?**

Não existe delegação de tarefas. O time é autônomo, são os integrantes que definem isso. Ninguém faz um relatório detalhado para a gerência.

* ***Roadmap***

É um panorama visual dos lançamentos (*releases*) do produto e suas funcionalidades principais na linha do tempo. Visão de alto nível do produto criada pelo cliente e que apresenta os principais marcos do projeto no decorrer do tempo.

* ***Feedback* do Produto**

A comunicação entre *product owner* e time de desenvolvimento para **avaliar o produto final ou parcial resultante de uma *sprint***.

O *feedback* do produto é de extrema importância para que o **time conheça a percepção do cliente quanto ao produto** que está sendo entregue ou que já está em produção.

Caso seja necessário, **mudanças são apontadas para que melhorias possam ser implementadas** a fim de atender os requisitos pertinentes.

* ***Release***

*Release* é um conjunto de funcionalidades implementadas que será disponibilizado para utilização e que **visa atender uma ou várias necessidades do cliente**.

As funcionalidades que compõem um release são escolhidas de acordo com a priorização do *backlog* do produto na ordem de maior valor para o negócio.

* **Planejamento da *Sprint***

O trabalho a ser realizado na *sprint* é planejado durante a Cerimônia de Planejamento da *Sprint*. Este plano é criado com o trabalho colaborativo de todo o time *scrum* e o dono do produto. A comunicação com o dono do produto.

* **Revisão da *Sprint***

Trata-se de uma cerimônia realizada no final da *sprint* para inspecionar o incremento e adaptar o *backlog* do produto, se necessário. Durante a revisão da *sprint* o time *scrum* e as partes interessadas colaboram sobre o que foi feito na *sprint*.

* ***Scrum***

*Scrum* não é um processo, técnica ou um método definitivo. Em vez disso, é um *framework* estrutural usado para organizar o trabalho no desenvolvimento de produtos complexos, dentro do qual pode ser empregado vários processos ou técnicas.

Os valores do *scrum* são: comprometimento, coragem, foco e respeito, que são incorporados e vividos pelo time *scrum*. E os seus pilares são: transparência, inspeção e adaptação, que se tornam vivos e constroem a confiança para todos.

* **Retrospectiva da *Sprint***

A retrospectiva da *sprint* é uma oportunidade para o time *scrum* inspecionar a si próprio e criar um plano para melhorias a serem aplicadas na próxima *sprint*. Ela ocorre depois da entrega de resultados ao *product* *owner* na revisão da *sprint* e antes do planejamento da próxima *sprint*.

Para que seja eficaz, essa reunião requer certa maturidade emocional e um clima de confiança.

O essencial é lembrar que não estarão procurando um culpado, mas apenas identificando pontos de melhoria.

**12. Requisitos de normas legais e regulamentares**

12.1 As normas legais, regulamentares e convencionais com as quais a contratação e solução apresentada deverão estar em conformidade:

* Lei nº 8.666/1993, que dispõe sobre licitações e contratos;
* Lei 10.520/2002, que institui a modalidade de licitação denominada pregão;
* Lei nº 13.709/2018, que dispõe sobre a proteção de dados pessoais (LGPD);
* Lei nº 14.442/2022, que dispõe sobre o pagamento de auxílio-alimentação ao empregado e altera a Lei nº 6.321, de 14 de abril de 1976, e a Consolidação das Leis do Trabalho;
* Decreto-Lei nº 5.452/1943 que aprova a Consolidação das leis Trabalhistas;
* Decreto nº 7.174/2010, que regulamenta a contratação de bens e serviços de informática e automação;
* Decreto nº 8.538/2015, que regulamenta o tratamento favorecido, diferenciado e simplificado para microempresas, empresas de pequeno porte, agricultores familiares, produtores rurais pessoa física, microempreendedores individuais e sociedades cooperativas nas contratações públicas de bens, serviços e obras;
* Decreto nº 9.637/2018, que institui a Política Nacional de Segurança da Informação;
* Decreto n º 9.507/2018, que dispõe sobre a execução indireta, mediante contratação, de serviços da administração pública federal direta, autárquica e fundacional e das empresas públicas e das sociedades de economia mista controladas pela União;
* Decreto nº 10.024/2019, que regulamenta a licitação na modalidade pregão, na forma eletrônica, para a aquisição de bens e contratação de serviços comuns;
* Decreto nº 10.520/2002, que institui a modalidade de licitação denominada pregão, para aquisição de bens e serviços comuns;
* Resolução CNJ nº 468/2022 que dispõe sobre diretrizes para as contratações de Solução de Tecnologia da Informação e Comunicação pelos órgãos submetidos ao controle administrativo e financeiro do Conselho Nacional de Justiça;
* Resolução TSE nº 23.702/2022, que dispõe sobre a Política de Governança das contratações na Justiça Eleitoral e dá outras providências.;
* Resolução TSE nº 23.644/2021, que dispõe sobre a Política de Segurança da Informação (PSI) no âmbito da Justiça Eleitoral;
* Instrução Normativa (IN) nº 11/TSE/2021, que regulamenta as fases das contratações no âmbito do Tribunal Superior Eleitoral;
* Instrução Normativa (IN) nº 12/TSE/2017, que estabelece critérios na adoção do regime de serviço suplementar prestado por postos de trabalho de empresas contratadas pelo Tribunal Superior Eleitoral;
* Instrução Normativa (IN) nº 01/SEGES/ME, de 04 de abril de 2019, que dispõe sobre o processo de contratação de soluções de Tecnologia da Informação e Comunicação - TIC pelos órgãos e entidades integrantes do Sistema de Administração dos Recursos de Tecnologia da Informação - SISP do Poder Executivo Federal;
* Instrução Normativa (IN) nº 05/SEGES/ME, de 26 de maio de 2017, que dispõe sobre as regras e diretrizes do procedimento de contratação de serviços sob o regime de execução indireta no âmbito da Administração Pública federal direta, autárquica e fundacional;
* Modelo Nacional de Interoperabilidade (MNI) do Poder Judiciário; e
* Padrão de Interoperabilidade de Governo Eletrônico – e-Ping.

12.2. Deverão ser observadas, no que se aplicar, as boas práticas de mercado conforme estabelecido nos padrões e metodologias descritas a seguir:

* NBR ISO/IEC n° 14.764 (Processo de manutenção de software);
* NBR ISO/IEC n° 12.207 (Processo de ciclo de vida de software);
* NBR ISO/IEC n° 25.010 (Qualidade de software);
* NBR ISO/IEC n° 27001 (Sistemas de gestão da segurança da informação — Requisitos);
* NBR ISO/IEC n° 27002 (Código de prática para controles de segurança da informação);
* NBR ISO/IEC n° 22301 (Segurança da sociedade — Sistema de gestão de continuidade de negócios — Requisitos);
* NBR ISO/IEC n° 27005 (Gestão de riscos de segurança da informação);
* NBR ISO/IEC n° 31000 (Gestão de riscos – Diretrizes);
* NBR ISO/IEC 38500 (Governança da TI para a organização); e
* OWASP Testing Guide, OWASP TOP 10 ou equivalentes.