METODOLOGIA DE DESENVOLVIMENTO DE SISTEMAS - MDS

Ministério do Meio Ambiente e

Mudança do Clima - MMA

1. INTRODUÇÃO

A Metodologia de Desenvolvimento de Sistemas – MDS descrita é uma coleção documentada de políticas e processos usados por times de desenvolvimento de software. Seu objetivo é melhorar o processo de criação em termos de produtividade da equipe de tecnologia e qualidade das soluções finais. No entanto, com a rápida evolução tecnológica e a necessidade de atender às novas demandas dos usuários, surgiu uma nova geração de MDS baseadas em metodologias ágeis.

As metodologias ágeis são projetadas para se adaptar ao ambiente de negócio dinâmico, proporcionando maior flexibilidade e capacidade de resposta a mudanças. Um exemplo é o Scrum, framework de gestão ágil que divide o processo de desenvolvimento de *software* em iterações curtas chamadas "*sprints*". Cada *sprint* tem uma duração fixa, geralmente de duas a quatro semanas, durante o qual um conjunto de funcionalidades é planejado, desenvolvido, testado e entregue ao cliente.

A MDS do MMA adota os princípios e práticas do *Scrum* como base para sua abordagem ágil, além de observar as diretrizes e normativos do órgão central de Tecnologia da Informação e Comunicação (TIC), os padrões de qualidade e normas técnicas, como a ABNT NBR ISO IEC IEEE 12.207:2020.

2. TIPOS DE SERVIÇOS

A área de desenvolvimento de software abrange um conjunto de serviços e atividades que busca entregar uma solução digital com o objetivo de atender às várias necessidades do órgão. De modo geral, esses serviços podem ser divididos nas seguintes categorias: projeto, manutenção e sustentação.

- **Projeto** – correspondem ao conjunto de atividades executadas com a finalidade de atender às necessidades do órgão ou entidade por meio da implementação de um novo software ou de um módulo de um software existente;
- **Manutenção** – contemplam um grupo de atividade que envolve mudanças para corrigir erros, uma melhoria negocial e/ou melhoria técnica para otimização de um sistema.

- **Sustentação** – corresponde ao conjunto de atividades necessárias para manter a disponibilidade, estabilidade e desempenho do software em produção, de modo que funcione dentro dos níveis de serviço esperados pelo órgão. Envolve a correção de defeitos e o monitoramento preventivo e proativo. Considera-se **serviços de sustentação** as pequenas evolutivas que apresentem tamanho funcional até 25SFP ou 25PF na sua contagem indicativa. Entretanto, os técnicos da CGTI poderão classificar estes serviços como manutenção de pequeno porte para os casos em que o tamanho funcional não consiga refletir adequadamente o esforço necessário para execução do serviço pretendido.

São previstas as seguintes atividades na execução dos serviços de projeto, manutenção e sustentação de software:

- aplicação de técnicas de Engenharia de Requisitos com vistas a identificar e especificar requisitos funcionais e não funcionais dos produtos a serem entregues;
- execução de procedimentos de *Design* / Arquitetura de software com vistas a estabelecer os padrões, tecnologias, formas de organização e de componentização dos recursos a serem utilizados na construção e manutenção dos sistemas;
- implementação dos códigos, componentes e recursos necessários à materialização do produto de software;
- realização de testes funcionais, unitários e de acessibilidade, desde a concepção dos produtos, com vistas a assegurar a qualidade do software;
- realização da homologação dos produtos junto aos clientes, com vistas a certificar-se que o software atende aos requisitos esperados;
- realização da implantação dos produtos junto às áreas de operação e suporte de rede, ou áreas equivalentes de sustentação de software, com o objetivo de assegurar a efetiva entrega do software em ambiente de produção.
- adoção das medidas necessárias para assegurar a disponibilidade, integridade, confidencialidade e autenticidade das informações a serem tratadas no âmbito da prestação dos serviços de desenvolvimento, manutenção, sustentação, testes e controle de qualidade de software.
- adoção das medidas para garantir a proteção dos dados, antecipando ameaças à privacidade, à segurança e à integridade, prevenindo acesso não autorizado às

informações disponibilizadas para prestação dos serviços de desenvolvimento, manutenção, sustentação, testes e controle de qualidade de software.

3. CLASSIFICAÇÃO DE PROJETOS

A Coordenação-Geral de Tecnologia da Informação – CGTI é responsável por prover e manter as soluções digitais do MMA, atuando para padronizar os processos inerentes ao tema. É primordial a participação da CGTI em todas as definições que envolvam produtos de software, sejam desenvolvidos internamente na sua gestão, ou externamente, na gestão da área de negócio.

Importante ressaltar que todo projeto de desenvolvimento de software precisa estar aderente a esta MDS, em especial, quanto ao desenvolvimento ágil e os artefatos aqui previstos.

Diante disso, os projetos são classificados como internos ou externos.

- **Projetos internos** – são executados com recursos da CGTI e acompanhados pela sua equipe, geralmente são apoiados por meio de contratos firmados pela CGTI ou exclusivamente pela equipe interna.
- **Projetos externos** – são executados fora do âmbito da CGTI por meio de instrumentos firmados pelas áreas de negócio com entidades externas ao MMA.

No caso dos Projetos externos, a atuação da CGTI se limita as seguintes atividades:

1. Comunicar e informar acerca das regras e normas internas aplicáveis – consiste em apresentar as diretrizes a serem seguidas no projeto, como tecnologias, guias, modelos e artefatos. Essa atividade deve ocorrer obrigatoriamente no início do projeto e tem como objetivo mitigar problemas futuros, como por exemplo, quando da internalização do software pelo MMA. É fundamental o cumprimento dessa etapa, a qual deve ser registrada em processo digital.
2. Orientações técnicas – realizadas sob demanda, visa esclarecer eventuais dúvidas relacionadas ao desenvolvimento e envolve as questões apresentadas na atividade anterior.

3. Disponibilizar infraestrutura – é uma atividade necessária para que o projeto aconteça, como acesso a VPN, criação dos ambientes de desenvolvimento, homologação e produção, criação de banco de dados, criação de repositório para o projeto e disponibilização da solução em produção.
4. Internalização da solução – consiste no recebimento do software a ser mantido e sustentado pela CGTI. Apenas poderá ser realizada quando a empresa entregar todos os artefatos apresentados no início do projeto, conforme previsto nesta metodologia, e realizar a transferência de conhecimento envolvendo a solução.

Para contratação envolvendo empresas públicas, é importante ressaltar que o processo de contratação é de responsabilidade da CGTI, conforme Instrução Normativa SGD/ME Nº 94, de 2022. Cabe à área de negócio encaminhar sua demanda para realização do estudo técnico que analisa as soluções possíveis de atendimento dos requisitos do negócio, além do custo-benefício, resultando na melhor solução para atendimento da necessidade.

4. ARQUITETURA TECNOLÓGICA

A arquitetura tecnológica refere-se à estrutura e organização dos componentes tecnológicos em um sistema ou ambiente de TI (Tecnologia da Informação) e, para o atendimento das necessidades da área de negócio, os projetos precisam ser desenvolvidos com base em um conjunto de tecnologias.

O desenvolvimento de produtos de software *ad hoc* e sem a observância do ambiente tecnológico em que ele estará inserido expõe a organização a riscos, além de outros problemas, como: incompatibilidade, dificuldade de manutenção, dificuldade integração e migração, falta de flexibilidade e escalabilidade.

Buscando mitigar esses e outros problemas que poderão surgir, ressalta-se a importância do desenvolvimento dos projetos em conformidade com as diretrizes e orientações da CGTI e em harmonia com o complexo tecnológico presente no órgão,

Diante disso, os serviços deverão ser executados observando-se as diretrizes de arquitetura tecnológica estabelecidas pela CGTI, conforme descrito no ANEXO I – ESPECIFICAÇÕES TÉCNICAS.

A adoção de tecnologia ou arquitetura diversa deverá ser autorizada previamente pela CGTI. Caso não seja autorizada, é vedado adotar nos projetos de software arquitetura, componentes ou tecnologias diferentes do padrão estabelecido.

Os projetos deverão adotar padrões de projeto (*Design Patterns*) ou padrões arquiteturais consolidados no mercado aderentes às necessidades da aplicação, além de métodos de codificação limpa (*Clean Code*).

Deverão ser observados na definição da arquitetura aspectos de desempenho, racionalização de recursos, sustentabilidade, clareza e segurança.

5. PAPÉIS E RESPONSABILIDADES

O processo de desenvolvimento de *software* é executado pelo Time ágil e deve ser observado e adaptado à realidade operacional do MMA.

Para desenvolvimento e manutenção, adota-se o Time Ágil, que é composto **no mínimo** pelos seguintes papéis:

- Scrum Master
- Dono do Produto
- Desenvolvedores
- Analista de Requisitos
- Líder do Projeto.

Esse time pode ser integrado também por profissionais especializados e outros stakeholders relevantes para o projeto.

Os membros do time ágil devem:

1. Participar ativamente dos eventos do Scrum (Sprint, Planejamento da Sprint, Reuniões diárias, Revisão da Sprint, Retrospectiva da Sprint);
2. Executar a sprint e entregar incrementos de software ao final de cada sprint;

3. Ser responsável por todas as atividades relacionadas com os produtos, desde a colaboração dos stakeholders, verificação, manutenção, operação, experimentação, investigação e desenvolvimento, assim como tudo o mais que possa ser necessário;
4. Adotar práticas de melhoria contínua;
5. Realizar testes funcionais e não funcionais concomitante ao desenvolvimento de software;
6. Apoiar na homologação das sprints e releases;
7. Elaborar manuais do usuário ou help de funcionalidades.

Participam do Time ágil, os seguintes **papéis**:

PAPEL		**ATIVIDADES QUE É RESPONSÁVEL OU QUE PARTICIPA COMO UM DOS RESPONSÁVEIS PELA EXECUÇÃO**
Scrum Master	Profissional com conhecimento aprofundado em técnicas ágeis.	• Garantir que o *Scrum* seja entendido e aplicado; • Assegurar que todos os eventos do *Scrum* têm lugar e são positivos, produtivos e mantidos dentro tempo previsto; • Apoiar o Dono do Produto e a organização na adoção de práticas ágeis; • Buscar melhoria contínua do time; • Facilitar a colaboração dos *stakeholders* conforme solicitado ou necessário; • Atualizar Gráfico de *Burndown*; • Remover impedimentos para a equipe de desenvolvimento durante a execução das *Sprints*.
Dono do Produto (Product Owner – PO)	Servidor representante da área demandante de soluções de software,	• Responsável por ordenar o trabalho a ser realizado pelo time, criando, mantendo e priorizando o(s) backlog(s) do(s) produto(s);

	designado por autoridade competente	• Criar e compartilhar a Visão do Produto; • Planejar o *Roadmap*; • Construir o *Backlog* do Produto; • Expressar claramente os itens do *Backlog* do Produto; • Ordenar e priorizar os itens do *Backlog* do Produto; • Apoiar na construção do *Sprint backlog* • Garantir que o time de desenvolvimento atenda os itens do *Backlog* do Produto no nível necessário; • Apoiar no planejamento do *Release*; • Validar Incremento de *Software*; • Validar *software* ao final de cada *sprint* e *release*; • Reportar a Avaliação de Satisfação do Dono do Produto.
Analista de Requisitos	Profissional contratado ou servidor do órgão responsável.	• Apoiar o Dono do Produto em suas atividades e construção dos artefatos sob responsabilidade do PO. • Realizar o levantamento das necessidades • Apoiar na construção do *Sprint backlog*
Desenvolvedores de software	Desenvolvedores que fazem parte do time ágil	• Criar o *Sprint Backlog*; • Construir o(s) produto(s) de *software*. • Realizar testes
Profissionais Especializados	Demais profissionais que podem ou não integrar o time com especialidades	• Contribuir para a construção ou aperfeiçoamento dos produtos de software.

	definidas, Arquiteto, Analista de UX/UI e Analista de BI.	
Partes interessadas (Stakeholders)	Profissionais impactados pela solução ou que possuam interesse na entrega da solução	• Opinar, influenciar, contribuir para o planejamento e tomadas de decisão do negócio ou projeto; • Esclarecer dúvidas; • Se necessário, apoiar o PO na validação da *sprint* ou *release*.
Analistas de Teste e Qualidade	Analistas de Teste e Qualidade que fazem parte do time ágil ou não.	• Garantir a qualidade dos sistemas durante todo o ciclo do processo de software até a sua implantação, minimizando a ocorrência de erros no ambiente de produção; • Realizar a revisão de código, realização de testes avançados e revisão da qualidade da documentação produzida; • Apoiar a fiscalização técnica dos contratos de desenvolvimento, manutenção e sustentação de software na revisão técnica dos critérios de aceitação e de qualidade dos produtos entregues.
Líder do Projeto	Servidor da área de tecnologia ou de outra área designado para acompanhar o projeto	• Planejar, acompanhar, controlar, apoiar a execução e o encerramento dos projetos • Identificar e envolver os diversos envolvidos; • Acompanhar e monitorar a execução o cronograma do projeto; • Elaborar e controlar o escopo do projeto junto ao Dono do Produto, que pode ser apoiado pelo analista e requisitos.

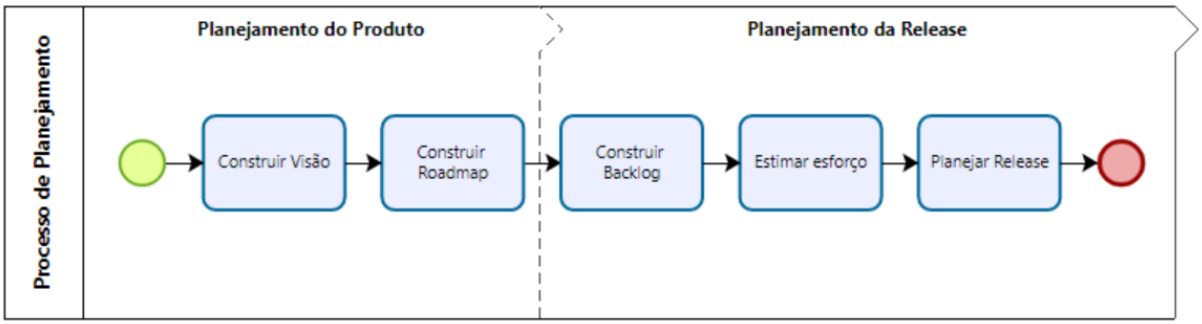
		• Gerenciar a análise de riscos – riscos de desenvolvimento, disponibilidade de recursos e tecnologia; • Gerenciar e controlar as mudanças do projeto. • Analisar e avaliar o desempenho dos membros do Time.

6. FASES DO PROCESSO DE DESENVOLVIMENTO

O processo de desenvolvimento pode ser dividido conforme as fases abaixo:

6.1. PLANEJAMENTO

O planejamento compreende atividades anteriores à execução da primeira *sprint* e está dividido em planejamento do produto e planejamento da release, composto pelas seguintes atividades:



6.1.1. FASE: PLANEJAMENTO DO PRODUTO

Atividades relacionadas:

a) Construir a Visão do Produto

b) Construir o *Roadmap* do Produto

A área requisitante do projeto de *software*, representada pelo dono do produto e *stakeholders*, fornece a compreensão do negócio, das necessidades, dos objetivos de negócio e dos requisitos do produto de *software*.

O time ágil deve oferecer o suporte necessário ao Dono do Produto nas atividades de planejamento do produto.

Atividade: Construir Visão	
Nessa atividade, entende-se que existe um problema a ser resolvido ou uma oportunidade a ser aproveitada.	
Responsáveis: PO e Analista de Requisitos	Entregáveis: Documento de Visão
Atividade: Construir Roadmap	
Dividir os objetivos de negócio e as características-chaves ou macro funções do produto em partes entregáveis, por ordem de prioridade. As partes são os *releases* que, por sua vez, são construídas a partir das características-chaves do produto priorizadas e ordenadas.	
Responsáveis: PO e Analista de Requisitos	Entregáveis: *Roadmap* do Produto

6.1.2. FASE: PLANEJAMENTO DA RELEASE

Neste grupo de atividades será construído o planejamento de um ou mais *releases***,** conforme o plano cronológico definido no *roadmap***.**

Cada release poderá ser construído em uma ou mais iterações (*Sprints*). O Time Ágil e eventuais stakeholders relevantes para o projeto o apoiam nas atividades de planejamento do *release*.

Planejamento do Release é composto pelas seguintes atividades, que podem ser adaptadas à realidade operacional do MMA:

a) Construir *Backlog*;

b) Estimar esforço;

c) Planejar *Release***.**

Atividade: Construir *Backlog*
• Construir e disponibilizar o *backlog* do produto, que é a lista priorizada dos itens necessários para o desenvolvimento e entrega do produto de software. • O *Backlog* do Produto representa tudo que é necessário para desenvolver e lançar um produto de valor agregado ao negócio. É uma lista de todos os requisitos (funcionais e não funcionais), funções, tecnologias, melhorias e correções de defeitos que constituem as mudanças que serão efetuadas no produto para versões futuras. • Os requisitos do software a serem desenvolvidos serão decompostos em estórias de usuários que, por sua vez, poderão ser subdivididas em tarefas. Este refinamento será feito no decorrer do projeto de acordo com a prioridade dos requisitos do software. • Os requisitos do software, as estórias de usuários e as tarefas compõem o backlog do produto. • As eventuais manutenções corretivas e adaptativas que venham a ser necessárias no software podem ser incluídas no *backlog* do produto. • O *backlog* do produto será priorizado pelo Product Owner (PO).

Nota 1: Nas atividades de planejamento do produto caso seja identificado um grande conhecimento da solução a ser produzida (objetivos de negócio, metas e características-chaves), com pequenas possibilidades de mudanças, a elaboração do backlog do produto poderá ser logo após a concepção do Documento de Visão. Essa estratégia possibilita que o planejamento do roadmap, com o plano cronológico de entrega dos releases seja estimado com maior precisão.

Nota 2: Os itens do backlog não devem ser detalhados nesta etapa, esse trabalho deve ser realizado para os itens priorizados e presentes no backlog da sprint.

Responsáveis: Product Owner apoiado pelo Analista de Requisitos	Entregáveis: *Backlog* do Produto

Atividade: Estimar esforço

- Atividade para estimar o tamanho do *backlog* do produto e esforço necessário para sua construção, utilizando técnicas de mensuração.
- Apoia no planejamento dos releases e quantidade de iterações.

Nota 1: a estimativa de tamanho e esforço pode ser feita por meio do método de ponto de função descrito no ROTEIRO DE MÉTRICAS do MMA.

Nota 2: O uso da métrica PF e seu registro por meio de software de gestão de projetos tanto em projetos internos e externos é relevante para estimativa de esforço, melhor gestão do projeto e histórico para estimativas futuras.

Responsáveis: Profissional designado	Entregáveis: Estimativa do tamanho e esforço do *Backlog* do produto

Atividade: Planejar Release

- Atividade para definir o plano do *release* com a meta a ser alcançada em função dos objetivos de negócio e características-chaves do produto. O planejamento do *release* parte do pressuposto que o *backlog do release* já foi definido a partir dos objetivos de negócio e características-chaves do release.
- A fragmentação do release em *sprints* ocorrerá conforme a priorização e complexidade dos itens do *backlog*, o tempo disponível para a construção do release ou do projeto, estimativa

de tamanho/esforço dos itens de backlog, histórico de produtividade da instituição e a estratégia de desenvolvimento estabelecida para o produto.

Nota 1: Podem ser inseridos no plano do release premissas, impedimentos e riscos envolvidos no release, além de prever atividades prévias ao início das iterações para que a equipe execute a criação/disponibilização dos ambientes de desenvolvimento e de testes necessários.

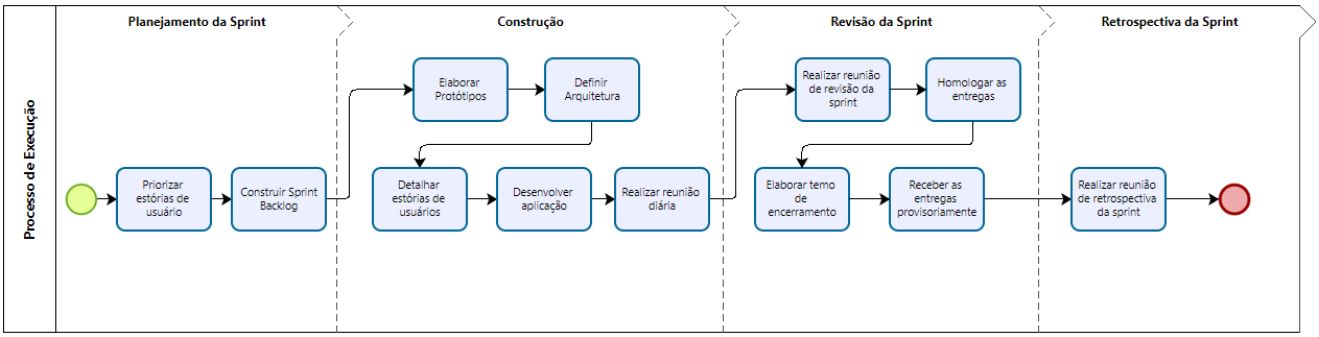
Devem ser definidos, para cada projeto, parâmetros para a execução das *sprints*, tais como:

- Configuração mínima do time que irá executar o conjunto de *sprints***,** indicando perfis profissionais mínimos e nível de compartilhamento aceitável para determinados perfis;
- Duração máxima da *sprint*;
- Meta de velocidade da *sprint*, como a quantidade de estórias de usuário e pontos de função;
- Meta de escopo planejado x realizado, que indica o percentual realizado a cada *sprint* em comparação ao escopo planejado; e

Meta de itens de *backlog* planejados x não planejados, que mapeia se o esforço, a cada *sprint*, está sendo gasto com novas funcionalidades planejadas ou com refatorações de código, dívidas técnicas e correções de falhas.

Responsáveis: Product Owner apoiado pelo Analista de Requisitos	Entregáveis: Plano da *Release*

6.2. FASE: EXECUÇÃO



Corresponde a um grupo de atividades para transformar os itens de backlog em um incremento de software: Sprint Planning, Daily Scrum, Implementação, Teste, Qualidade, Sprint Review, Sprint Retrospective.

- As Iterações/*Sprints* são ciclos de execução do projeto, com duração de 1 a 4 semanas, de acordo com as características de cada projeto.
- A duração das sprints e o dia e a duração das reuniões de planejamento, revisão e retrospectiva das sprints serão definidas pelo Líder do projeto juntamente com o Scrum Master no início do projeto. Estas definições poderão ser alteradas posteriormente, a critério da CGTI, mediante comunicação prévia à equipe de desenvolvimento do Contratado.
- O critério para aceitar cada sprint como "pronta" (finalizada, *done*) será definido para cada projeto e, quando necessário, de forma particular para cada tarefa.

6.2.1. ETAPA: PLANEJAMENTO DA SPRINT

Com base no *backlog* do produto priorizado, o time realiza o planejamento da *sprint*, selecionando os itens possíveis de serem desenvolvidos no tempo da *sprint***.**

O dono do produto estabelece os critérios de definição de pronto, ou seja, o que deve ser entregue ao final da *sprint*.

O time, então, se compromete a entregar o incremento de *software*, ao final da *sprint*.

Esse planejamento pode levar até 4 horas, dependendo do tamanho do *sprint***,** sendo necessário 1 hora para cada semana de duração da sprint. Por exemplo, um sprint com previsão de realização de 2 semanas, seu planejamento deve ter 2 horas de duração.

Os itens de *Product Backlog* selecionados para a *Sprint* são agrupados no *Backlog da Sprint***.**

Os itens do backlog da sprint podem ser vistos como o detalhamento do *Product Backlog* e geralmente quebrados em tarefas menores as quais podem incluir desenvolvimento de código, testes, design, documentação ou qualquer outra atividade necessária para entregar o item de trabalho planejado.

Na definição do *backlog* da *sprint*, deve-se monitorar a relação quantitativa entre itens planejados e itens não planejados, com vistas a assegurar que o maior esforço esteja sendo empreendido na entrega de valor.

Atividade: Priorizar estórias de usuário	
O Product Owner, com apoio dos membros da equipe ágil, seleciona quais são as estórias mais importantes para serem desenvolvidas na sprint.	
Responsáveis: Product Owner e Equipe ágil	Entregáveis: Lista de estórias de usuário
Atividade: Construir Sprint Backlog	
Com os itens selecionados, a equipe de desenvolvimento começa a detalhar o trabalho necessário para concluir cada item da Sprint Backlog e o atingimento dos objetivos da sprint.	
Responsáveis: Desenvolvedores e Scrum Master	Entregáveis: Sprint Backlog

6.2.2. ETAPA: CONSTRUÇÃO

O time de desenvolvimento inicia a implementação, construindo o incremento de software a partir do *backlog da sprint*.

A atividade de construção tem como tarefa principal a codificação, além disso deve:

- Elaborar arquitetura (quando necessário);
- Elaborar protótipos;
- Interação entre PO, analista de requisitos e outras partes interessadas com o objetivo de sanar dúvidas, esclarecer requisitos ou detalhá-los melhor e de transformar os requisitos do backlog da sprint em software;
- Atualização de demais artefatos.

Essas tarefas não são ordenadas nesta metodologia, pois tem como finalidade oferecer ao time ágil a faculdade de se auto-organizarem, conforme suas competências, habilidades e experiências para melhor entregarem o produto esperado pelo PO.

Durante esse período, o time também se reúne diariamente, por 15 minutos, para responder a três perguntas:

a) O que eu fiz desde a última reunião?

b) O que eu vou fazer até a próxima reunião?

c) Há algum impedimento para o trabalho ser realizado?

O *Sprint Backlog* é atualizado durante a execução da *Sprint*, à medida que se vai aprendendo mais e deve estar detalhado de forma que o seu progresso possa ser inspecionado nas reuniões diárias.

<table>
<tr><td colspan="2">Atividade: Elaborar protótipos</td></tr>
<tr><td colspan="2">Seleção dos itens possíveis de serem desenvolvidos no tempo da sprint.</td></tr>
<tr><td>Responsáveis: Analista de negócio/requisitos ou Analista de UX/UI.</td><td>Entregáveis: Protótipos</td></tr>
<tr><td colspan="2">Atividade: Definir arquitetura</td></tr>
<tr><td colspan="2">Descrição em alto nível da estrutura fundamental e da organização do sistema de software, incluindo componentes, módulos, interações e decisões de design.</td></tr>
<tr><td>Responsáveis: Arquiteto</td><td>Entregáveis: Documento de arquitetura.</td></tr>
<tr><td colspan="2">Atividade: Detalhar estórias de usuário</td></tr>
<tr><td colspan="2">Consiste em aprofundar e especificar de forma pormenorizada as estórias presentes no Sprint Backlog.</td></tr>
<tr><td>Responsáveis: Analista de negócio/requisitos</td><td>Entregáveis: Estórias detalhadas</td></tr>
<tr><td colspan="2">Atividade: Desenvolver aplicação</td></tr>
<tr><td colspan="2">Tem como a principal tarefa a codificação, mas também, contempla outras assessórias como inclusão do projeto no GIT e esteira, criação de scripts de banco, por exemplo.</td></tr>
<tr><td>Responsáveis: Desenvolvedores</td><td>Entregáveis:
Código fonte versionado no GIT
Aplicação rodando na esteira
Scripts de banco
MER do Banco de Dados
Evidência de teste unitário, componente e integração</td></tr>
</table>

	• Aplicação disponibilizada para o usuário
Atividade: Realizar reunião diária	
É um evento de 15 minutos para os membros do time ágil. Para reduzir a complexidade, é realizado à mesma hora e em todos os dias úteis do Sprint.	
Responsáveis: Time ágil	• Registro da reunião (gravação no Teams mp4)

6.2.3. ETAPA: REVISÃO DA SPRINT

Após o último dia da execução da *sprint*, a equipe se reúne para realizar a revisão do produto.

Nesse evento, o time de desenvolvimento apresenta o incremento de *software* construído para o Dono do Produto e os *stakeholders***.**

O time revisa o que foi realizado no *sprint* e avalia o que fazer a seguir.

O Dono do Produto analisará a resolução de cada estória apresentada e decidirá se está “pronta” ou não, de acordo com o critério previamente estabelecido.

Nesse momento, o *backlog* do produto pode ser atualizado para refletir as novas demandas e oportunidades.

Caso alguma estória seja reprovada, ela será reinserida no *Backlog***,** ficando disponível para uma próxima *sprint***.**

Essa revisão pode levar até 4 horas para *sprints* de 4 semanas, sendo 1 hora para cada semana.

Atividade: Realizar reunião de revisão da sprint	
É realizada uma reunião para apresentação dos resultados da sprint, o Time ágil e os stakeholders fazem uma análise da entrega.	
Responsáveis: Scrum Master	Entregáveis: • Registro da reunião (gravação no Teams mp4) • *Backlog* atualizado

Atividade: Homologar as entregas	
Os artefatos e o software serão verificados e validados pelo Product Owner e o Líder do projeto, estará apta a sua disponibilização em produção.	
Responsáveis: Product Owner e Líder do projeto	Entregáveis: Aceite do produto
Atividade: Elaborar termo de encerramento	
Consiste no documento que formaliza a entrega do incremento e contém todos os artefatos previstos para a sprint.	
Responsáveis: Scrum Master	Entregáveis: Termo de Encerramento
Atividade: Receber as entregas provisoriamente	
O Líder do projeto recebe provisoriamente a entrega do trabalho desenvolvido durante a Sprint para posterior verificação e validação junto ao Product Owner.	
Responsáveis: Scrum Master e Líder do projeto	Entregáveis: Termo de Recebimento Provisório

6.2.4. ETAPA: RETROSPECTIVA DA SPRINT

O último evento antes da conclusão da *sprint* é chamado de retrospectiva da *sprint*.

O propósito da retrospectiva da *sprint* é planejar maneiras de aumentar a qualidade e a eficácia do trabalho.

O time discute como foi a última *sprint* em relação a indivíduos, interações, processos, ferramentas e a Definição de Pronto.

Nesse evento responde-se basicamente a dois questionamentos:

a) O que deu certo?

b) O que pode ser melhorado?

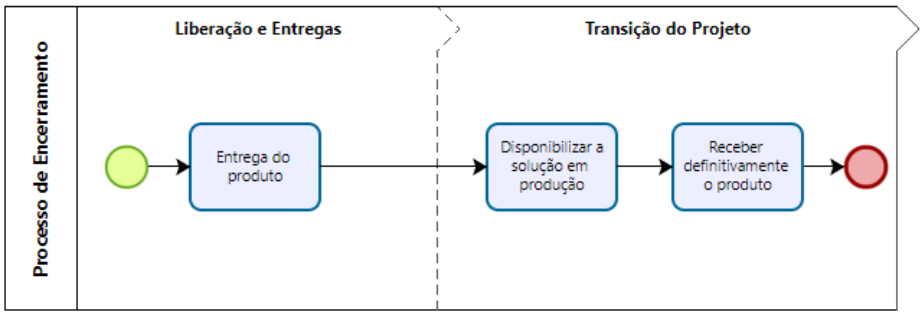
O time, então, identifica as mudanças mais úteis para melhorar sua eficácia e procura aplicar essas mudanças no *sprint* seguinte.

A retrospectiva pode levar até 4 horas para *sprints* de 4 semanas.

Finalmente, o time inicia o planejamento da próxima *sprint* e o ciclo se repete quantas vezes forem necessárias.

Atividade: Realizar reunião de retrospectiva da sprint	
Analisar os pontos fortes e fracos no desenvolvimento da *sprint* e iniciar a seleção dos itens possíveis de serem desenvolvidos no tempo da próxima *sprint*	
Responsáveis: Scrum Master	Entregáveis: Registro da reunião (gravação no Teams mp4)

6.3. FASE: ENCERRAMENTO



A fase de encerramento é composta por dois subprocessos: Liberação e Entregas e Transição do Projeto.

6.3.1. ETAPA: LIBERAÇÃO E ENTREGAS

A entrega dos produtos de *software* ocorre após a finalização da *sprint***,** quando o incremento de *software* é revisado e homologado. É um evento separado da *sprint*.

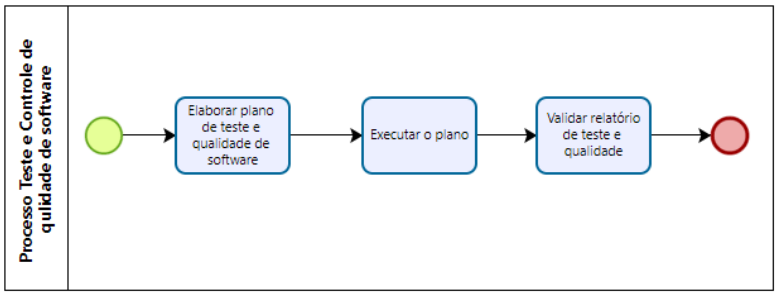
Atividade: Entrega do produto	
Os artefatos e o software serão entregues formalmente após homologação	
Responsáveis: Scrum master	Entregáveis: Incremento de software e documentação técnica

6.3.2. ETAPA: TRANSIÇÃO DO PROJETO

Este grupo de atividades tem a finalidade de realizar algumas verificações finais da *release* ou do projeto produzido e garantir uma versão útil do produto em ambiente de produção.

Atividade: Disponibilizar a solução em produção	
Conjunto de atividade necessária à disponibilização da aplicação no ambiente de produção.	
Responsáveis: Scrum master	Entregáveis: Aplicação executando em produção
Atividade: Receber definitivamente o produto	
Esta atividade consiste na formalização de que o produto está em conformidade ao demandado e atende os requisitos mínimos de sua aceitabilidade.	
Responsáveis: Equipe de gestão do contrato	Entregáveis: Termo de Recebimento Definitivo

7. TESTES E CONTROLE DE QUALIDADE DO SOFTWARE



Os testes e controle da qualidade do produto que serão entregues devem permear todas as fases e execução das sprints e serão incluídos conforme a necessidade e características do projeto.

Sempre que possível, deve-se promover as seguintes atividades, que podem ser adaptadas à realidade operacional da CGTI:

a) analisar riscos e a conformidade de processo, projetos, técnicas, práticas e ferramentas de desenvolvimento e testes das empresas que prestam serviços de desenvolvimento de sistemas em relação às normas e padrões da organização e às melhores práticas de mercado no que diz respeito a versionamento, arquitetura, padrão visual, segurança, testes e qualidade de software;

b) verificar a atualidade da documentação técnica dos sistemas de informação da organização em relação ao software que está operando;

c) promover o diagnóstico de situações de gargalos e problemas de desempenho nos sistemas;

d) verificar se os padrões da organização para desenvolvimento de aplicações sejam obedecidos;

e) detectar falhas e propor correções em processos de testes implantados na organização;

f) emitir pareceres técnicos relacionados ao ambiente de sistemas da organização;

g) elaborar Relatório Técnico de Análise de Qualidade.

Os testes devem ser planejados conforme práticas a seguir, que podem adaptadas à realidade operacional da CGTI:

a) realizar reuniões com os usuários e/ou times de desenvolvimento para modelar e elaborar estratégias de testes;

b) planejar testes funcionais e não funcionais de softwares;

c) elaborar artefatos como Roteiros/Casos de Teste, Listas de Verificação, Critérios de Aceite.

A execução dos testes deve abordar as seguintes atividades, que podem adaptadas à realidade operacional da CGTI:

a) elaborar artefatos de apoio a testes, como roteiros, **scripts** de testes, relatório de evidências de testes;

b) executar testes automáticos e/ou manuais em ambiente de testes e de homologação;

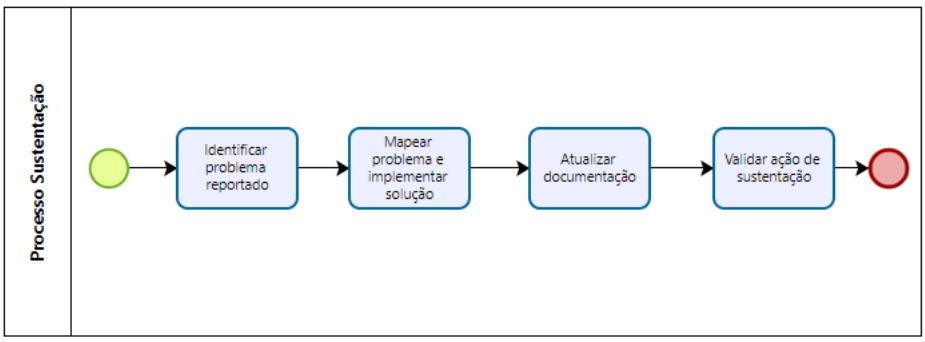
c) executar testes funcionais e não funcionais;

d) dar suporte aos testes realizados pelo usuário.

Atividade: Elaborar Plano de Teste e Qualidade do software
Consiste na elaboração das estratégias, atividades e serviços de teste e qualidade que serão realizadas no projeto.

Responsáveis: Analista de Teste/Qualidade	Entregáveis: Plano de Teste e Qualidade
Atividade: Executar o plano	
Nesta atividade serão realizadas as ações previstas no plano de teste e qualidade.	
Responsáveis: Analista de Teste/Qualidade	Entregáveis: Relatório de Teste e Qualidade
Atividade: Validar relatório de teste e qualidade	
Consiste na realização da verificação se o relatório apresentado atende os critérios mínimos de aceitação previamente definidos.	
Responsáveis: Equipe de gestão do contrato	Entregáveis: Termo de Recebimento Definitivo

8. SUSTENTAÇÃO



O serviço de sustentação de software corresponde ao conjunto de atividades necessárias para manter a disponibilidade, estabilidade e desempenho do software em produção, dentro dos níveis de serviço definidos pelo MMA.

Mensalmente, será aberta uma OS de sustentação com o backlog a ser atendido priorizado. No decorrer da execução desta OS, essa lista pode ser atualizada para acomodar demandas mais urgentes, sempre acordado com o time.

Atividade: Reportar problema
O problema identificado pela área responsável deverá ser formalizado para CGTI por meio da ferramenta de gestão de chamados de sistemas e deverá conter, no mínimo:

<table>
<tr><td colspan="2">

- A descrição do problema reportado;
- *Prints* de tela com as mensagens de erro;
- Descrição dos passos para reprodução do erro.

Todos os problemas identificados por outros usuários **<u>deverão</u>** ser enviados ao *Product Owner* que analisará e, se pertinente, formalizará junto à CGTI a demanda.
</td></tr>
<tr><td>Responsáveis: Product Owner</td><td>Entregáveis: Relato do problema</td></tr>
<tr><td colspan="2">Atividade: Identificar problema reportado</td></tr>
<tr><td colspan="2">Verificar se o relatado pelo Product Owner procede e atualizar o backlog da sustentação, revendo as prioridades, de acordo com a criticidade e urgência.

O reporte, acompanhamento e demais interações serão feitos no registro da ferramenta de gestão de chamados. A OS mensal de sustentação, a qual possui a lista prioritária de demandas de sustentação, deverá ser observada pela equipe de sustentação, pois lá estarão as demandas acordadas entre o time e que deverão ser executadas na OS mensal.</td></tr>
<tr><td>Responsáveis: Líder do Projeto</td><td>Entregáveis: Solicitação de ações de sustentação</td></tr>
<tr><td colspan="2">Atividade: Mapear problema e implementar solução</td></tr>
<tr><td colspan="2">Mapear problemas, cenários e soluções dos sistemas em produção e realizar ações corretivas</td></tr>
<tr><td>Responsáveis: Analista de Requisitos e Desenvolvedor</td><td>Entregáveis: Aplicação corrigida</td></tr>
<tr><td colspan="2">Atividade: Atualizar documentação</td></tr>
<tr><td colspan="2">Caso a realização das ações de sustentação tenha impacto na documentação, a mesma deverá ser atualizada.</td></tr>
<tr><td>Responsáveis: Analista de Requisitos</td><td>Entregáveis: Documentação atualizada</td></tr>
<tr><td colspan="2">Atividade: Validar ação de sustentação</td></tr>
<tr><td colspan="2">Consiste na verificação se o problema foi efetivamente solucionado.</td></tr>
<tr><td>Responsáveis: Product Owner e Líder do Projeto</td><td>Entregáveis: Aceite do produto</td></tr>
</table>

Para projetos externos sob responsabilidade das áreas fins, a sustentação deve no mínimo incluir as seguintes atividades:

- Identificar se o problema reportado efetivamente se tratar de uma demanda de sustentação;
- Mapear problemas, cenários e soluções dos sistemas em produção;
- Implementar a solução em conformidade com as diretrizes fixadas para o processo de desenvolvimento, por exemplo, design patterns, clean code, segurança da informação, qualidade de código, testes, etc.
- Atualizar a documentação de sistemas legados;
- Atuar de forma direta e ativa na interface com a área de infraestrutura, incluindo dirimir quaisquer dúvidas e apoio necessários ao bom funcionamento dos sistemas em produção;
- Apoiar tecnicamente na busca contínua pela melhoria de processos entre as áreas;
- Realizar absorção e configuração de sistemas no parque computacional do MMA.

9. DEFINIÇÃO DE PRONTO

A definição de pronto é uma descrição formal do estado do incremento, quando este cumpre as medidas de qualidade exigidas para o produto.

Quando um item do *Backlog* do produto satisfaz a Definição de Pronto, nasce um incremento.

A definição de pronto cria transparência; proporciona a todos uma compreensão do trabalho que foi concluído como parte do Incremento. Todo o time ágil deve estar em conformidade com a definição de pronto.

Se um item de *Product Backlog* não cumpre a definição de pronto, não pode ser lançado ou mesmo apresentado na Revisão da *Sprint*. Em vez disso, volta ao *Backlog* do produto para consideração futura.

A seguir, alguns critérios para aceitação dos produtos, que podem ser adaptados à realidade operacional do MMA:

Para admissibilidade do produto:

a) código-fonte submetido ao controle de versões da CGTI;

b) existência de testes unitários e do Relatório de Testes ou vídeo;

c) existência de *scripts* de banco de dados com dicionário de dados embutido nos metadados (ausência apenas quando não houver mudança no modelo de dados);

d) existência de arquivo para geração de *Build***;**

e) disponibilização de processos prontos para execução na ferramenta de CI/CD adotada, juntamente com a entrega e configuração de *containers* configurados pela ferramenta orquestração adotada;

f) existência de manual de implantação, conforme modelo disponibilizado pela CGTI;

g) existência documentação concluída, de acordo com os padrões de qualidade definidos pela CGTI e validadas pelo demandante.

h) resultado da execução de teste SAST indicando ausência de vulnerabilidades de nível HIGH ou CRITICAL, ou equivalente.

Para aceitação da demanda: após realizar a inspeção do produto quanto à sua admissibilidade, o Contratado deverá:

a) executar testes funcionais automatizados que tenham sido solicitados e, consequentemente, verificar se estão corretamente implementados ou mesmo se existem, além de observar os resultados da execução;

b) executar testes unitários ou verificar relatórios de execução destes que possam envolver porções críticas do produto;

c) realizar alguns testes funcionais, pelo menos nos principais fluxos do produto entregue.

Após a realização dos testes, a organização deve proceder a uma das ações a seguir:

a) **rejeição:** caso sejam percebidos defeitos de natureza impeditiva em alguma estória implementada ou não tenha coberto o escopo planejado de tal forma que a entrega não seja passível de aceitação;

b) **aceitação parcial**: caso a demanda possua alguns defeitos significativos de natureza não-impeditiva ou não tenha coberto o escopo planejado de tal forma que ainda seja passível de aceitação;

c) **aceitação integral:** caso a demanda esteja em nível de qualidade tal que não sejam percebidos defeitos significativos, bem como envolva cumprimento do escopo planejado.

O Contratado deve registrar todos os aspectos relevantes. Os defeitos percebidos nos casos de rejeição ou aceitação parcial da *sprint* devem fazer parte de um item de *backlog* da próxima *sprint*.

ANEXO I - ESPECIFICAÇÕES TÉCNICAS

As necessidades tecnológicas definem os padrões, metodologias, processos definidos, competências das equipes, cuidados com a segurança da informação, entre outros aspectos, que a solução deve atender para que atinja o desempenho e os resultados esperados. Para a realização dos serviços, deverá ser levado em consideração as tecnologias existentes no âmbito do MMA a serem adotadas para execução dos projetos. Deverão ser cumpridos os procedimentos, normas, padrões, modelos, guias e regulamentos do Órgão, podendo destacar:

1. Padrão de Objetos e Estrutura de Banco de Dados;
2. Metodologia Ágil de Desenvolvimento de Software do MMA;
3. Para o desenvolvimento de novos projetos de sistemas serão priorizados o uso da linguagem JAVA;
4. Os novos projetos mobile serão preferencialmente desenvolvido em JAVA (backend) e frameworks Angular e Ionic;
5. O SGBD prioritário para os sistemas e apps é PostgreSQL com sua extensão PostGIS para projetos que utilizar dados de georreferenciamento.

Os serviços de desenvolvimento e sustentação devem se basear na dinâmica definida no ambiente DevOps do MMA, bem como todo o processo de implantação no ambiente tecnológico do Órgão. Segue abaixo o conjunto de tecnologias e plataformas utilizados:

Tópico	**Recurso Tecnológico**
Linguagem de Programação, frameworks e tecnologias associadas:	- JAVA, PHP, .NET, Phyton, NodeJs -Wordpress e Joomla - Ionic, Cordova, Capacitor - JSF, Wicket, EJB, Hibernate, JPA

	- Lefleat, MapServer, I3GEO, Geoserver - React, JavaScript, Angular - REST, SOA, XML, View - JHipster, Springboot, Spring Framework, CAS Apereo
Plataforma:	- Web, Mobile (IOS - ANDROID - PWA)
Banco de dados:	- Oracle, PostgreSQL/PostGIS, MySQL, SQL Server
Esteira DEVOPS:	- OKD Kubernetes, *Rancher*, Git, Docker, GitOps, *Min.IO/Velero,LongHorn e Microsoft Teams.* - Nexus, SonarQube, Jhipster(Discovery), Helm Chart.
Gerência de Ordens de Serviço:	- Mantis BugTracker.
Ferramenta de Desenvolvimento:	- Eclipse, Netbeans ou ferramenta equivalente livre.
Ferramenta de Gestão de Equipes Ágeis	SGC
Ferramenta de Testes automatizados:	- Selenium ou ferramenta equivalente livre; - Jmeter ou ferramenta equivalente livre.
Servidores de Aplicações e Middlewares:	- Apache, JBOSS, Tomcat
Repositório de Autenticação e Autorização de Usuários:	- Servidor Active Directory - AD - Lightweight Directory Access Protocol - LDAP

Outras Tecnologias/Ferramentas:	Kibana, Apollo Server, Puppeteer, Elasticsearch, Vue.js, Mapbox GL JS, Rundeck, Pangea Server, Pentaho Data Integration, ogr2ogr e Logstash.

Além das necessidades de desenvolvimento, manutenção e sustentação de *software*, incluindo georreferenciamento, identificou-se também necessidades de desenvolvimento e manutenção de painéis e ambientes de *analytics* e a contratação de serviços de garantia de qualidade e teste de softwares.

A arquitetura empregada para o desenvolvimento deverá ser orientada a microsserviços, de forma desacoplada, em regra. Os microsserviços desenvolvidos deverão ser catalogados e reutilizados. Para que um sistema ou parte de um sistema seja desenvolvido com base em uma arquitetura monolítica, a CONTRATADA deverá justificar a inviabilidade de utilização da arquitetura orientada a microsserviços.

Deverão ser observados:

1. As diretrizes tecnológicas e Padrões Digitais do Governo Federal, principalmente, no que tange ao Design System (https://www.gov.br/ds/home).

2. Os aplicativos mobiles, além da necessidade de publicação nas lojas Apple Store (iOS) e Google Play (Android), a critério do Órgão, deverá também ser gerada e publicada a versão PWA (Progressive Web App) no servidor de aplicação do MMA.

3. Padrões Web em Governo Eletrônico (e-PWG), que contém cartilhas com recomendações sobre usabilidade, redação, codificação, manutenção e arquitetura de informação e desenho que orientam o desenvolvimento de páginas, sítios e portais do Governo Federal;

4. Modelo de Acessibilidade em Governo Eletrônico (e-MAG), que consiste em um conjunto de recomendações a ser considerado para que o processo de acessibilidade dos sítios e portais do governo brasileiro seja conduzido de forma padronizada e de fácil implementação;

5. Padrões de Interoperabilidade de Governo Eletrônico (e-PING), que define um conjunto mínimo de premissas, políticas e especificações técnicas que regulamentam a utilização da Tecnologia de Informação e Comunicação (TIC) no Governo Federal, estabelecendo as condições de interação com os demais Poderes e esferas de governo e com a sociedade em geral;

6. Modelo de Requisitos para Sistemas Informatizados de Gestão Arquivística de Documentos (e-ARQ Brasil);

7. Aderência às regulamentações da Infraestrutura de Chaves Públicas Brasileira (ICP-Brasil), conforme a Medida Provisória nº 2.200-2, de 24 de agosto de 2001, quando houver necessidade de utilização de certificação digital;

8. Requisitos e recomendações da Infraestrutura Nacional de Dados Espaciais (INDE). A INDE é um conjunto integrado de tecnologias, políticas, mecanismos e procedimentos de coordenação e monitoramento, padrões e acordos, necessário para facilitar e ordenar a geração, o armazenamento, o acesso, o compartilhamento, a disseminação e o uso dos dados geoespaciais de origem federal, estadual, distrital e municipal.

ANEXO II - MODELO DE VISÃO DO PRODUTO

1. Histórico de revisões

Versão (XX.YY)	***Data (DD/MMM/YYYY)***	***Autor***	***Descrição***

2. Visão Geral do Produto

Indicar problema, oportunidade, benefícios e necessidades que o produto/projeto resolve ou aproveita

3. Atores

Indicar quem são os clientes e usuários interessados na solução

4. Escopo do Produto

Lista de requisitos funcionais em alto nível que permitam o dimensionamento estimado do tamanho funcional da aplicação em Simple Function Point –SFP, permitido a realização de outras estimativas do projeto.

5. Requisitos Não Funcionais, premissas e restrições

Requisitos não-funcionais são os requisitos relacionados ao uso da aplicação em termos de desempenho, usabilidade, confiabilidade, segurança, disponibilidade, manutenibilidade e tecnologias envolvidas. Não é preciso o cliente dizer sobre eles, pois eles são características mínimas de um software de qualidade, ficando a cargo do desenvolvedor optar por atender esses requisitos ou não. Isto é:

• Demonstram qualidade acerca dos serviços ou funções disponibilizadas pelo sistema. Ex.: tempo, o processo de desenvolvimento, padrões, etc.
• Surgem conforme a necessidade dos usuários, em razão de orçamento e outros fatores.
• Podem estar relacionados à confiabilidade, tempo de resposta e espaço nas mídias de armazenamento disponíveis.
• Caso ocorra falha do não atendimento a um requisito não funcional, poderá tornar todo o sistema ineficaz. Ex.: requisito confiabilidade em um sistema de controle de voos.

Exemplos de requisitos não funcionais:

- *Requisitos de produtos: Requisitos que especificam o comportamento do produto.Ex. portabilidade; tempo na execução; confiabilidade, mobilidade, etc.*
- *Requisitos da organização: Requisitos decorrentes de políticas e procedimentos corporativos. Ex. padrões, infraestrutura, etc.*
- *Requisitos externos: Requisitos decorrentes de fatores externos ao sistema e ao processo de desenvolvimento. Ex. requisitos de interoperabilidade, legislação, localização geográfica etc.*
- *Requisitos de facilidade de uso. Ex.: usuários deverão operar o sistema após um determinado tempo de treinamento.*
- *Requisitos de eficiência. Ex.: o sistema deverá processar n requisições por um determinado tempo.*

- *Requisitos de confiabilidade. Ex.: o sistema deverá ter alta disponibilidade, por.exemplo, 99% do tempo.*
- *Requisitos de portabilidade. Ex.: o sistema deverá rodar em qualquer plataforma.*
- *Requisitos de entrega.Ex.: um relatório de acompanhamento deverá ser fornecido toda segunda-feira.*
- *Requisitos de implementação.: Ex.: o sistema deverá ser desenvolvido na linguagem Java.*
- *Requisitos de padrões.: Ex. uso de programação orientada a objeto sob a plataforma A.*
- *Requisitos de interoperabilidade.:Ex. o sistema deverá se comunicar com o SQL Server.*
- *Requisitos éticos. Ex.: o sistema não apresentará aos usuários quaisquer dados de cunho privativo.*
- *Requisitos legais. Ex.: o sistema deverá atender às normas legais, tais como padrões, leis, etc.*
- *Requisitos de Integração. Ex.: o sistema integra com outra aplicação*

Documento de Visão

1. Visão Geral do Projeto:

O projeto "Gestão Verde" visa desenvolver um software de gerenciamento de tarefas que permite às equipes acompanharem, priorizarem e colaborarem de forma eficiente. O objetivo é aumentar a produtividade e a transparência no fluxo de trabalho, melhorando a comunicação e a entrega de projetos dentro do prazo.

2. Atores:

- *Usuários internos da equipe - responsáveis por criar, atribuir e concluir tarefas, bem como gerenciar projetos.*
- *Gerentes de projeto - têm acesso a recursos avançados de relatórios e análises para monitorar o progresso do projeto.*

- *Administradores do sistema - responsáveis por configurar permissões, criar templates de tarefas e gerenciar a segurança do sistema.*

3. **Escopo (Requisitos Funcionais):**
   - *Os usuários devem poder criar, atribuir e acompanhar tarefas.*
   - *Deve ser possível definir prioridades e datas de vencimento para as tarefas.*
   - *O sistema deve permitir a criação e o gerenciamento de projetos, com a capacidade de agrupar tarefas relacionadas.*
   - *Os usuários devem poder comentar e colaborar em tarefas.*
   - *Deve haver recursos de notificação para manter os usuários informados sobre atualizações nas tarefas.*
4. **Requisitos Não Funcionais:**
   - *O sistema deve ser altamente disponível, com um tempo de resposta rápido para garantir uma experiência eficiente.*
   - *Deve ser seguro, com autenticação e autorização adequadas para proteger os dados dos usuários.*
   - *A interface do usuário deve ser intuitiva e de fácil utilização, com uma curva de aprendizado mínima.*
   - *O software deve ser escalável, capaz de lidar com um grande volume de tarefas e usuários simultaneamente.*
   - *O sistema deve ser compatível com diferentes navegadores e dispositivos para garantir a acessibilidade e a usabilidade em várias plataformas.*
5. **Premissas:**
   - *O sistema será desenvolvido utilizando a linguagem de programação Java.*
   - *A plataforma será hospedada em servidores Linux.*
   - *A equipe de desenvolvimento terá acesso aos recursos necessários para implementar as funcionalidades propostas.*
6. **Restrições:**
   - *O orçamento para o desenvolvimento do projeto é limitado.*
   - *O prazo para entrega do software é de seis meses.*
   - *O sistema deve ser compatível com as versões mais recentes dos navegadores Chrome, Firefox e Safari.*

- *A equipe de desenvolvimento será composta por cinco desenvolvedores em período integral.*

ANEXO III - MODELO DE ROADMAP DO PRODUTO

1. Histórico de revisões

Versão (XX.YY)	*Data (DD/MMM/YYYY)*	*Autor*	*Descrição*

2. Releases do Produto

Dividir os objetivos de negócio e as características-chaves ou macro funções do produto em partes entregáveis, por ordem de prioridade. As partes são os releases que, por sua vez, são construídas a partir das características-chaves do produto priorizadas e ordenadas.

3. Estimativa de Prazo

A previsão dos prazos terão como base o escopo definido no documento de visão.

4. Roadmap

Release	*Semana 1*	*Semana 2*	*Semana 3*	*Semana 4*	*.....*	*Semana N*

<table>
<tr><td rowspan="3">Produto 1</td><td>User Stories</td><td>Protótipos</td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td colspan="2">Teste e validação</td><td></td><td></td></tr>
<tr><td></td><td colspan="2">Desenvolvimento</td><td></td><td></td><td></td></tr>
<tr><td rowspan="3">Produto 2</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>

ANEXO XXX - MODELO DE MANUAL DE IMPLANTAÇÃO

1. Histórico de revisões

Versão (XX.YY)	***Data (DD/MMM/YYYY)***	***Autor***	***Descrição***

2. Checklist

Item	**Não**	**Sim**	**Qual? Em que situação? Descreva...**
Utiliza datasource?	X		
Faz integração com outros sistemas (interno e/ou externo)?	X		
Há envio de e-mail?	X		
Utiliza certificação digital?	X		
Permite acesso por usuário externo?	X		
Permite acesso por usuário estrangeiro?	X		
Possui páginas públicas?	X		

3. Arquivos no servidor de aplicação

[Caso necessário, especifique as permissões em diretórios no servidor]

[Quando não houver manipulação de arquivos pela aplicação, deve-se remover a tabela abaixo e colocar a informação de que este tópico não se aplica]

Questionário para dimensionamento de espaço em disco	**Resposta**
Quais os tipos de arquivos (extensão) serão manipulados pela aplicação?	
Qual o tamanho máximo (em MB) para cada tipo de arquivo?	
Qual o volume esperado (em GB) de crescimento ao longo de um mês?	
Qual o volume esperado (em GB) de crescimento ao longo de um ano?	
Qual o tempo (em anos) de retenção[1] dos arquivos?	
Necessita de *backup* dos arquivos após o tempo de retenção?	*[Sim/Não]*

3. Rotinas

[Quando não houver execução de rotinas ou jobs, deve-se remover as tabelas abaixo e colocar a informação de que este tópico não se aplica.]

Questionário sobre rotinas	**Resposta**
Qual o nome da rotina ou job?	*[Exemplo: app-job.jar]*

A rotina gera log no servidor de aplicação?	*[Sim/Não]*
A rotina necessita de agendamento? Se sim, especificar a periodicidade.	
Há necessidade de informar parâmetros para a execução da rotina? *[Caso afirmativo, deve-se preencher a tabela abaixo.]*	*[Sim/Não]*

Especificação dos parâmetros para execução da rotina		
Ordem de execução	**Parâmetros**	**Finalidade**
1	*[Exemplo: app-job.jar -h]*	*[Exibe o texto de ajuda para execução da rotina]*

4. Procedimentos adicionais/complementares

[Descreva neste tópico os procedimentos necessários e/ou complementares para implantação do projeto]

Ordem	**Procedimentos**
1	*[Exemplo: Criar o datasource especificado no item 3.1]*
2	*[Exemplo: Disponibilizar e/ou atualizar o arquivo appPlan.xml no servidor weblogic disponível no endereço www.mma.gov.br/svn/Projetos/Plan.xml, revisão SVN nº [especificar o número da revisão]]*

3	*[Exemplo: Configurar os parâmetros para upload de arquivos conforme detalhado no item 3.2]*
4	*[Exemplo: atualizar o arquivo app-ejb.jar (módulo ejb) para execução das rotinas]*
5	