2ª versão

FNDE ÁGIL

METODOLOGIA DE DESENVOLVIMENTO DE SISTEMAS – MDS

Fundo Nacional de Desenvolvimento da Educação – FNDE

fnde
MINISTÉRIO DA EDUCAÇÃO
GOVERNO DO BRASIL
DO LADO DO POVO BRASILEIRO

Ministério da Educação
Fundo Nacional de Desenvolvimento da Educação

Presidência
Fernanda Macedo Pacobahyba – Presidente
Sylvia Cristina Toledo Gouveia– Chefe de Gabinete

Diretoria de Tecnologia e Inovação – DIRTI
Delson Pereira da Silva – Diretor de Tecnologia e Inovação

Coordenação-Geral de Soluções Digitais - CGSD
Alessandra Maria Costa e Lima - Coordenadora-Geral de Soluções Digitais
Edinilson Sousa - Coordenador de Automação, Arquitetura e Experiência do Usuário
Felipe Velter - Coordenador de Soluções Administrativas e de Suporte ao Negócio

---

Equipe de Elaboração
Eletiele Rocha da Silva
Erica Alessandra de Almeida Silva
Luiz Felipe da Paixão Carvalho
Vitor Vinícius Silva Oliveira

Brasília, abril de 2026

Sumário

01 INTRODUÇÃO

Introdução

1.1 A Metodologia de Desenvolvimento de Sistemas (MDS) do FNDE

A Metodologia de Desenvolvimento de Sistemas do Fundo Nacional de Desenvolvimento da Educação, denominada FNDE Ágil, é um conjunto de práticas adotadas pelas equipes de desenvolvimento e manutenção de sistemas do FNDE.

Ela foi desenvolvida em conformidade com a Portaria nº 647, de 05 de outubro de 2023, que instituiu a Política de Governança Digital do FNDE (disponível em **PORTARIA Nº 647, DE 5 DE OUTUBRO DE 2023 - PORTARIA Nº 647, DE 5 DE OUTUBRO DE 2023 - DOU - Imprensa Nacional**), a qual estabelece no seu artigo 24:

> *Art. 24. O FNDE deve possuir e executar um processo de software, assim entendido como o processo de trabalho usado pela organização na produção e na gestão do ciclo de vida de sistemas e aplicações - abrangendo as atividades realizadas desde a demanda, o provimento (desenvolvimento ou aquisição), a operação, a sustentação até a eventual desativação.*

A FNDE Ágil utiliza conceitos de Metodologias de Projetos Ágeis, Design Thinking e Lean Inception, além de seguir o **Modelo para Contratação de Serviços de Desenvolvimento, Manutenção e Sustentação de Software**, conforme a **Portaria SGD/MGI nº 750, de 20 de março de 2023** do Governo Digital.

Portanto, a utilização de metodologia ágil permite elaborar atividades complexas de forma simples, entregando primeiro o que é de maior valor para o cliente. Com isto, obtém-se a melhoria no desempenho, o que reflete em aumento da eficácia, eficiência, efetividade, produtividade e satisfação das áreas demandantes. É um processo empírico, no qual nem tudo está desenhado, e o time, em conjunto com a área negocial, vai continuamente tomando decisões, com um ritmo de desenvolvimento sustentável, em ciclos curtos de priorização, desenvolvimento e progresso real de produto.

Esta versão atualizada da Metodologia FNDE Ágil representa uma evolução significativa em relação à sua versão inicial, atualizando e detalhando os papeis das squads, de acordo com a contratação atualmente em vigor, e incorporando anexos com templates utilizados pelas equipes. Destacam-se também como melhorias: a integração mais

profunda entre Scrum, Design Thinking e UX Design ao longo de todo o ciclo de vida das soluções, o fortalecimento das práticas de qualidade, testes e acessibilidade desde a concepção até a entrega, bem como a ampliação dos artefatos e templates de apoio à gestão ágil. A metodologia também passou a contemplar de forma mais estruturada temas como métricas, indicadores de desempenho, planejamento de releases, integração com práticas de ITIL e governança, promovendo maior previsibilidade, transparência e foco na entrega contínua de valor para as áreas de negócio e para a sociedade.

1.2 Desing Thinking e Metodologia Ágil

O processo de desenvolvimento de Soluções Digitais envolve uma abordagem holística englobando a Análise de Necessidades, o Design de Experiência do Usuário (UX), a Integração de Tecnologias, Segurança e Conformidade, Manutenção e Suporte e Treinamento e Capacitação.

A Análise de Necessidades envolve compreender os problemas e as necessidades do negócio para criar uma solução que realmente agregue valor. Já o design de experiência do usuário garante que a solução seja intuitiva e fácil de usar. A integração de tecnologias conecta diferentes sistemas e tecnologias para trabalhar de forma harmoniosa. Segurança e conformidade visam proteger os dados e garantir que a solução esteja em conformidade com as regulamentações, enquanto Manutenção visa oferecer suporte contínuo e atualizações para manter a solução eficiente e segura.

Para atender todas essas etapas, a metodologia FNDE Ágil combina a estrutura do Scrum com os princípios do Design Thinking, reunindo o melhor da agilidade com o foco no usuário.

O Scrum oferece uma abordagem iterativa e adaptável para gerenciamento de projetos, promovendo entregas contínuas de valor de forma ágil. Já o Design Thinking contribui com uma visão empática e criativa, voltada para a compreensão profunda dos usuários e a solução de problemas reais.

Juntas, essas metodologias se complementam: enquanto o Design Thinking orienta a equipe na identificação dos problemas certos e na geração de soluções relevantes, o Scrum fornece a disciplina e a cadência necessárias para transformar essas soluções em produtos concretos e evolutivos. Essa integração torna o processo mais eficiente, centrado nas pessoas e capaz de gerar inovação no âmbito dos sistemas e serviços do FNDE.

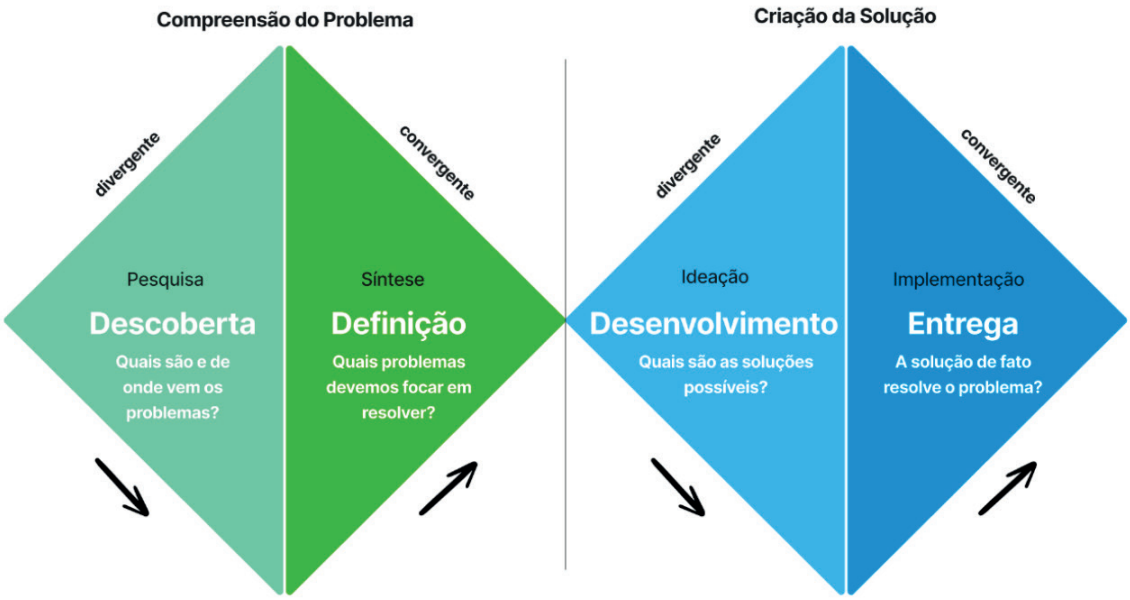
Design Thinking

O uso do Design Thinking no desenvolvimento de sistemas do FNDE tem como objetivo garantir que as soluções geradas atendam às reais necessidades das pessoas. Para isso, **coloca-se o ser humano no centro do processo, promovendo empatia, colaboração e foco na experiência do usuário.**

A metodologia equilibra três pilares essenciais: **o que é desejável para as pessoas, o que é tecnicamente viável** e o que **é sustentável do ponto de vista institucional e econômico.**

Ao contrário do método tradicional, que tende a buscar a solução mais óbvia, essa abordagem incentiva a exploração de diversas possibilidades antes de decidir qual caminho seguir, evitando escolhas precipitadas seguidas de reavaliações.

Estrutura em Duplo Diamante



Esse processo pode ser representado visualmente pelo modelo do Duplo Diamante *(Double Diamond)*, que organiza a jornada em dois grandes momentos (diamantes) – cada um com duas fases principais que alternam momentos de divergência e convergência.

1º diamante: Descoberta e Definição

- Descoberta: o primeiro diamante começa com a ampliação do olhar para

compreender o problema. Iniciamos com a fase de imersão ou empatia, investigando como o sistema atual funciona, ouvindo os envolvidos e pesquisando com os usuários para conhecer suas experiências e dificuldades. Também coletamos e analisamos dados relevantes para aprofundar o entendimento do contexto.

- Definição ou desenho: após a exploração inicial, convergimos e sintetizamos os aprendizados para identificar os principais desafios e definir com clareza qual problema deve ser resolvido. Esta definição direciona todo o restante do processo.

2º diamante: Desenvolvimento e Entrega

- Desenvolvimento ou construção: no segundo diamante, voltamos a ampliar o pensamento na fase de ideação, promovendo sessões criativas para gerar múltiplas possibilidades de solução. Em seguida, passamos para a prototipação, elaborando jornadas dos usuários e modelos que permitam simular as soluções propostas.
- Teste e Implementação: por fim, testamos os protótipos com usuários reais, aplicando roteiros para avaliar a experiência e identificar pontos de melhoria. Após os ajustes, avançamos para a implementação: desenvolvemos manuais, criamos o design system, estruturamos o sistema e realizamos a codificação final, preparando a solução para entrega e adoção.

Essa abordagem atende a Política de Governança Digital do FNDE, que preconiza:
I - o processo de software deve considerar, no mínimo, as seguintes fases essenciais:

a. descoberta: identificação, compreensão e análise do problema, necessidade e/ou oportunidade de geração de valor;
b. desenho: definição dos objetivos do projeto, elaboração do escopo, priorização, definição e planejamento do Mínimo Produto Viável (MVP);
c. construção: desenvolvimento iterativo e incremental do produto, de acordo com os requisitos definidos e priorizados - incluindo a validação técnica e negocial (testes) das entregas para garantir que atendem aos requisitos e não contenham erros;
d. implantação: implementação da entrega em ambiente de produção;
e. entrega: entrega do produto e/ou do incremento ao cliente, considerando a validação do valor gerado; e
f. manutenção: manter o software atualizado e resolver quaisquer problemas que possam surgir.

II - o processo ágil de software deverá ser caracterizado por:

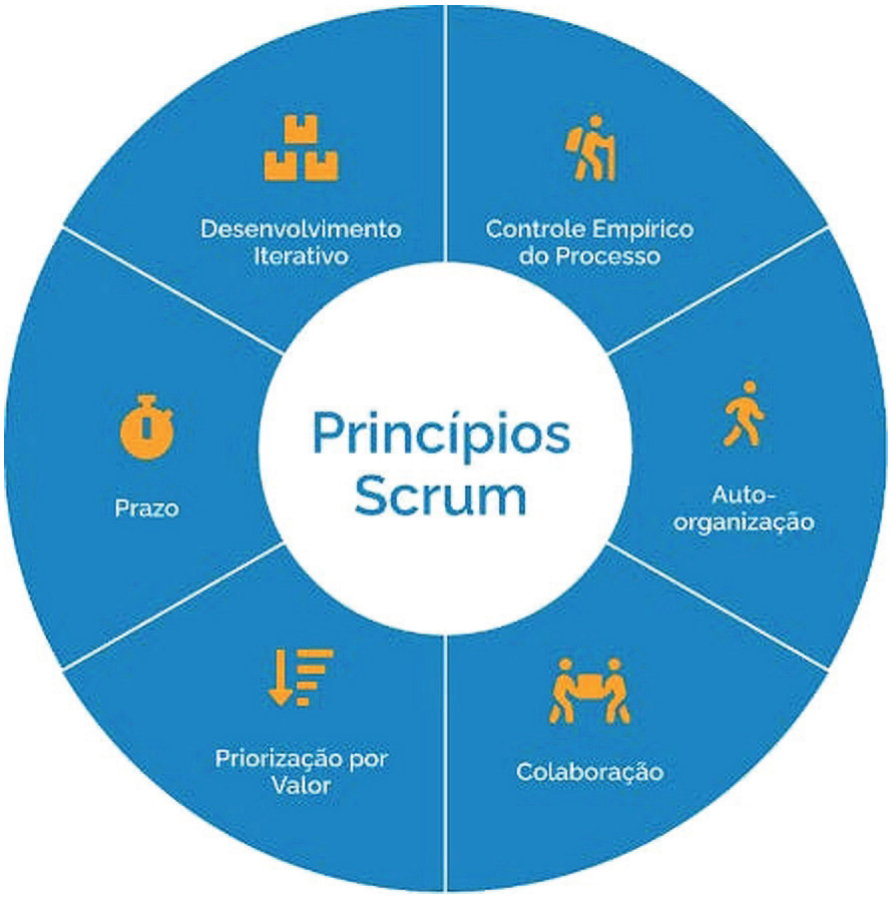
a. iteratividade: o processo é dividido em iterações curtas, geralmente de duas a

quatro semanas. Isso permite que a equipe entregue software em funcionamento de forma rápida e frequente.

b. incrementalidade: o software é desenvolvido de forma incremental, ou seja, a cada iteração, uma nova funcionalidade é adicionada ao software. Isso permite que a equipe entregue software de alta qualidade de forma gradual.
c. participação do cliente: o cliente é envolvido no processo de desenvolvimento desde o início. Isso permite que o cliente dê feedback sobre o software em desenvolvimento e garante que o software atenda às suas necessidades.

Como dito, a integração do Scrum e Design Thinking fortalece a metodologia de desenvolvimento de soluções digitais adotada pelo FNDE ao unir agilidade e entrega incremental de valor com criatividade e foco no usuário. Essa abordagem híbrida torna o processo de desenvolvimento mais eficiente, inovador e alinhado às reais necessidades da sociedade e dos gestores da educação pública.

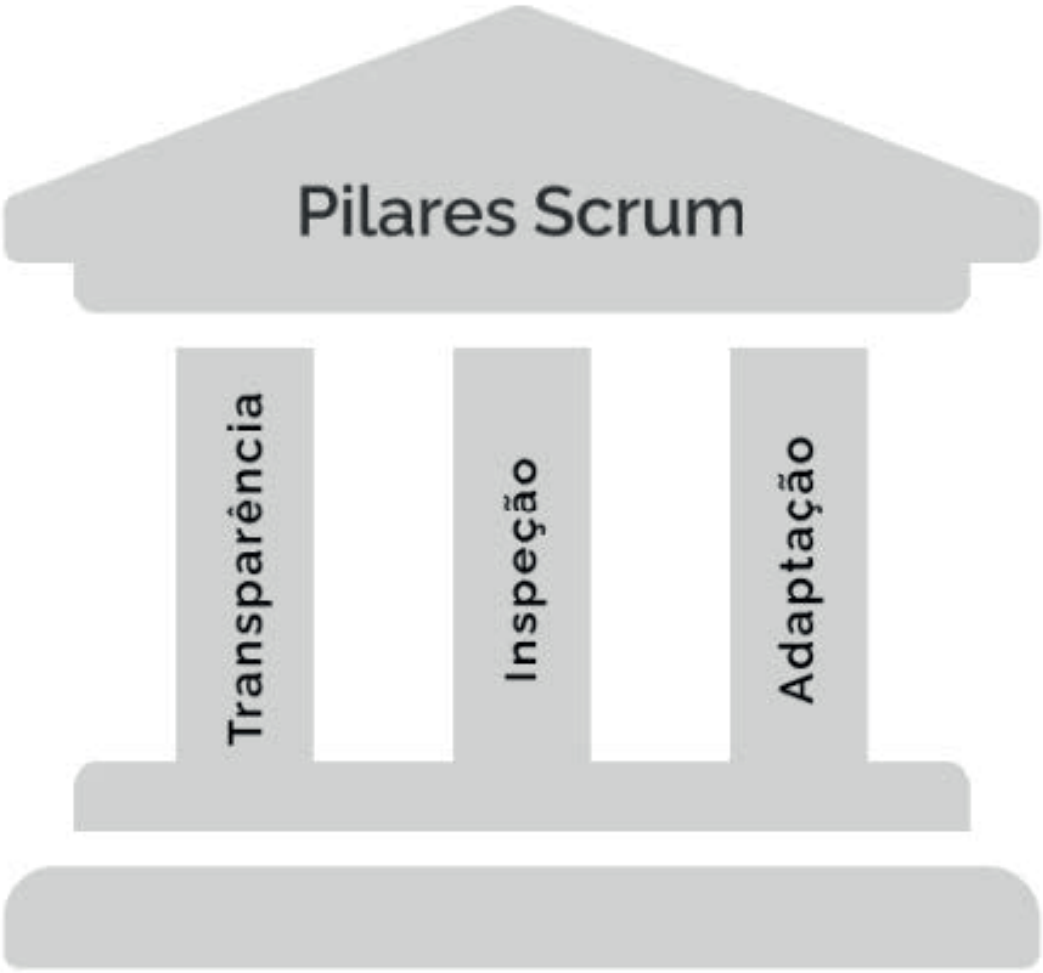
Scrum: Princípios, Pilares e Valores



Scrum é uma forma de organizar o trabalho em equipe, usada especialmente em desenvolvimento de software. O termo vem do rugby e envolve oito jogadores de linha em cada equipe (conhecidos como forwards) se unindo de forma compacta para disputar a posse da bola. Assim como no esporte, as equipes de desenvolvimento devem trabalhar colaborativamente para solucionar problemas e avançar com a solução.

A proposta do Scrum é ser diferente das metodologias rígidas e hierárquicas, incentivando a equipe a aprender com as experiências, se organizar para resolver problemas e melhorar continuamente. Ele abraça a incerteza e promove a criatividade, em vez de seguir um processo fixo e prescritivo.

Pilares Scrum



Os três pilares fundamentais sustentam todo o processo Scrum. Eles formam a base para que as equipes possam aprender com a experiencia e se adaptar continuamente.

- Transparência: garantir que todos os envolvidos tenham clareza sobre o processo, com informações importantes visíveis para quem é responsável pelos resultados.
- Inspeção: monitorar constantemente o que está sendo feito, por meio de reuniões diárias e revisões ao final de cada sprint. O objetivo é detectar qualquer desvio ou problema que possa impedir o alcance das metas.
- Adaptação: ajustar o processo sempre que houver algum problema ou desvio indesejado.

Valores Scrum

A metodologia se baseia em 5 valores fundamentais que guiam o comportamento e as interações dos membros da equipe. Esses valores são a "cola" que une os pilares do Scrum.

- **Compromisso:** a equipe Scrum se dedica a alcançar seus objetivos.
- **Foco:** o principal foco é o trabalho da Sprint.
- **Abertura:** a equipe e as partes interessadas são abertas ao trabalho e aos desafios.
- **Respeito:** os membros da equipe se respeitam e são respeitados.
- **Coragem:** a equipe tem coragem para fazer o que é certo e enfrentar problemas difíceis.

02 PAPÉIS NA MDS

2. Squad e Papéis na MDS-FNDE

2.1 Squads

Squads são equipes pequenas e multidisciplinares, com membros que têm habilidades e competências diferentes. Elas são autônomas, tomando decisões de forma descentralizada e transparente.

Essas equipes são formadas conforme a necessidade e disponibilidade de profissionais, aumentando a produtividade e incentivando a inovação. Neste modelo, as estruturas organizacionais hierárquicas tradicionais são rompidas e as equipes colaboram de forma ágil e transversal.

Os squads buscam ter alto alinhamento e autonomia, com todos participando da solução. O tamanho da equipe depende da complexidade do projeto, prazos e orçamento. A formação dos squads considera os conhecimentos, habilidades e atitudes necessárias para a sprint, mas alguns papéis são sempre presentes, como por exemplo os desenvolvedores. Já outros participam paralelamente de mais de um squad, como os arquitetos de software, os analistas de dados e os UX/UI designers.

Composição das Squads

As squads no FNDE serão formadas no mínimo por:

- 1 Dono do Produto (Product Owner, PO);
- 1 Scrum Master;

Time de Desenvolvimento, composto, no mínimo por:

- 1 Arquiteto de Software;
- 1 Administrador de Dados;
- 1 UX Designer.
- 2 Analistas Desenvolvedores Full Stack (pleno ou sênior).

2.2 Papéis na MDS

Para o bom funcionamento da metodologia de desenvolvimento FNDE Ágil, é fundamental compreender os papéis exercidos por cada integrante da equipe, uma vez que a responsabilidade pela conclusão das tarefas é compartilhada por todos. O cliente, mais que um demandante, é considerado um parceiro que participa ativamente da execução das atividades, sendo representado pelo Dono do Produto (PO).

2.3 Dono do Produto

O Dono do Produto (PO) é responsável por **representar os interesses dos stakeholders em um projeto.**

O PO desempenha um papel crucial como ponte entre as necessidades de negócio e a equipe de desenvolvimento, sendo responsável pela visão do produto, gerenciamento do backlog e comunicação com as diversas partes interessadas. Ele atua como o principal responsável por maximizar o valor do produto resultante do trabalho da equipe de desenvolvimento. A sua capacidade de equilibrar as expectativas de diversos atores é essencial para o sucesso do projeto, pois garante que as decisões sobre o produto sejam baseadas em necessidades reais e dados.

Responsabilidades e Atividades:

1. Representar Stakeholders:

    - Atuar como a principal voz dos Stakeholders, representando suas necessidades e interesses.

2. Conhecimento do Negócio:

    - Possuir profundo entendimento do negócio para responder às dúvidas da equipe de desenvolvimento;
    - Comunicar claramente a meta do produto e a visão do que deve ser produzido.

3. Gerenciamento de Demandas:

    - Abrir, refinar e homologar as demandas;
    - Priorizar as tarefas com base no impacto para o negócio e no risco envolvido;
    - Organizar as tarefas em uma lista conhecida como Backlog do Produto.

4. Comunicação e Transparência:

    - Garantir que os itens do Backlog do Produto sejam comunicados de forma clara e transparente;
    - Assegurar que a lista seja visível e compreensível para todos os membros da equipe.

5. Criação de Histórias de Usuários e Critérios de Aceitação:

    - Criar Histórias de Usuários (HU) e Critérios de Aceitação.

6. Definição de “Pronto”:

    - Ter uma ideia clara dos objetivos finais do projeto;
    - Definir, juntamente com a equipe de desenvolvimento, o que significa que uma tarefa está completa.

Qualificações Necessárias:

- Profundo conhecimento das necessidades do negócio.
- Habilidade de comunicação clara e eficaz.

2.3 Scrum Master

Scrum Master é responsável por treinar a equipe na estrutura do Scrum e ajudar a eliminar qualquer obstáculo que possa diminuir o ritmo de trabalho. Ele foca na facilitação e na aplicação de processos adequados para maximizar a produtividade da equipe.

Responsabilidades e Atividades:

1. Treinamento e Facilitação:

- Garantir que o framework Scrum seja compreendido e aplicado corretamente pela equipe e facilitar reuniões ágeis, como daily scrum, sprint planning, sprint review e retrospectivas;
- Facilitar a comunicação entre todos os envolvidos no projeto;
- Treinar a equipe e os stakeholders sobre os valores, princípios e práticas ágeis;
- Orientar novos membros sobre os processos ágeis e o funcionamento da equipe;
- Incentivar o uso de padrões ágeis e boas práticas de trabalho colaborativo;
- Promover de um Ambiente Saudável e incentivar um ambiente de trabalho que valorize a segurança psicológica e a criatividade;
- Promover uma cultura de aprendizado contínuo e inovação dentro do time;
- Promover o alinhamento entre as expectativas dos stakeholders e a capacidade da equipe.

2. Remoção de Obstáculos:

- Proteger a equipe contra interferências externas, garantindo o foco nas metas da sprint, identificar e remover impedimentos que dificultem o progresso do time;
- Resolver conflitos internos na equipe, promovendo a colaboração e o engajamento, estimular a autonomia e a autogestão do time, fortalecendo o espírito de equipe.
- Implementar ações corretivas para maximizar a eficiência da equipe;
- Monitorar o bem-estar da equipe e intervir quando necessário para evitar sobrecarga de trabalho e apoiar equipes multidisciplinares com mindset ágil;

- Comunicar ao líder técnico e/ou ao coordenador responsável os impedimentos encontrados.

3. Controle e Registro de Ritos:

- Acompanhar o progresso das sprints, assegurando a conclusão de itens do backlog e auxiliar a equipe no gerenciamento do backlog de produto e priorização de tarefas;
- Monitorar métricas ágeis (como burndown charts e velocity) para identificar áreas de melhoria;
- Realizar retrospectivas para identificar pontos de melhoria nos processos e propor ajustes;
- Garantir que as sprints sejam revisadas e ajustadas para otimizar entregas futuras;
- Registrar e manter o registro de comparecimento aos ritos;
- Garantir que o time registre as tarefas na ferramenta ALM (Jira);
- Respeitar o “time box” dos ritos, garantindo que os problemas sejam resolvidos fora da reunião diária (Daily).

4. Suporte aos Times e ao FNDE:

- Apoiar o Product Owner na gestão do backlog, garantindo que as histórias de usuário sejam bem definidas e priorizadas;
- Estimular a transparência no processo de desenvolvimento e a comunicação aberta;
- Promover de um Ambiente Saudável e incentivar um ambiente de trabalho que valorize a segurança psicológica e a criatividade;
- Acompanhar o progresso das sprints, assegurando a conclusão de itens do backlog e auxiliar a equipe no gerenciamento do backlog de produto e priorização de tarefas.

Qualificações Necessárias:

- Conhecimento profundo das práticas e estrutura do Scrum;
- Habilidade em facilitar reuniões e eventos Scrum;
- Capacidade de identificar e resolver obstáculos de forma eficaz;
- Experiência em registro e controle dos ritos do Scrum.

2.4 Time de Desenvolvimento

Um time autogerenciável e multifuncional, composto por 3 a 9 membros, com todas as habilidades necessárias para concretizar a visão do Dono do Produto. Esse time é responsável por transformar os itens do backlog em software funcional, sem a necessidade de um líder ou controle hierárquico.

Responsabilidades e Atividades:

1. **Analisar a Demanda:**

    - Avaliar tecnicamente cada solicitação para entender os requisitos e as implicações;
    - Garantir que todos os aspectos técnicos da demanda sejam compreendidos.

2. **Estimar a Demanda:**

    - Determinar o esforço e o tempo necessário para completar cada tarefa;
    - Utilizar técnicas de estimativa adequadas para prever a complexidade e o trabalho envolvido.

3. **Executar a Demanda:**

    - Desenvolver e implementar as soluções conforme as especificações;
    - Garantir que o software funcione conforme o esperado e atenda aos requisitos do cliente.

4. **Definir o Pronto:**

    - Colaborar com o Product Owner (PO) para definir os critérios de aceitação e o conceito de “pronto”;
    - Assegurar que todos os itens de backlog atendam aos critérios de “pronto” antes da entrega.Desenvolvedores Full-Stack

São os profissionais com conhecimento abrangente em tecnologias de front-end e back-end, atuando de forma colaborativa na construção de soluções completas. No contexto ágil, integram o time de desenvolvimento, contribuindo ativamente para a entrega de incrementos de produto com qualidade, valor e alinhamento aos objetivos do time e da

organização.

Principais responsabilidades:

Desenvolver soluções completas

- Implementar funcionalidades tanto no lado do servidor (back-end) quanto na interface com o usuário (front-end), garantindo integração fluida entre as camadas da aplicação;
- Colaborar com o time Scrum: Participar dos eventos Scrum (Daily, Planejamento, Review, Retrospectiva), contribuindo o alinhamento entre equipe técnica, Product Owner e Scrum Master;
- Revisar e melhorar o código: Aplicar boas práticas de versionamento, revisão de código (code review), testes automatizados e padrões de arquitetura;
- Seguir alinhamento com os objetivos do produto: Trabalhar com foco no valor de negócio, compreendendo o impacto de cada funcionalidade para o usuário final;
- Produzir documentação técnica e suporte ao time: Documentar soluções, apoiar colegas em dúvidas técnicas e contribuir para o reuso e manutenção sustentável do código.

Desenvolvedores Full-Stack

São os profissionais com conhecimento abrangente em tecnologias de front-end e back-end, atuando de forma colaborativa na construção de soluções completas. No contexto ágil, integram o time de desenvolvimento, contribuindo ativamente para a entrega de incrementos de produto com qualidade, valor e alinhamento aos objetivos do time e da organização.

Principais responsabilidades:

Desenvolver soluções completas

- Desenvolver recursos e capacidades para usuários finais por meio de plataformas, ferramentas de desenvolvimento ou aprendizado de máquina, escrevendo código de qualidade, claro e testável, em conformidade com padrões e boas práticas de arquitetura, design, implementação e segurança;

- Configurar e personalizar software;
- Desenvolver blocos de construção de software reutilizáveis para permitir uma entrega mais rápida;
- Colaborar no desenvolvimento de modelos de aprendizado de máquina e pipelines de dados;
- Implementar funcionalidades tanto no lado do servidor (back-end) quanto na interface com o usuário (front-end), garantindo integração fluida entre as camadas da aplicação.

Colaborar com o time Scrum:

- Participar do trabalho de estimativa e previsão de trabalho;
- Apoiar equipes multidisciplinares com mindset ágil;
- Participar dos eventos Scrum (Daily, Planejamento, Review, Retrospectiva), contribuindo o alinhamento entre equipe técnica, Product Owner e Scrum Master.

Revisar e melhorar o código:

- Investigar e propor soluções para problemas de desenvolvimento e design;
- Melhorar o desempenho do software existente, diagnosticando e resolvendo problemas críticos;
- Aplicar boas práticas de versionamento, revisão de código (code review), testes automatizados e padrões de arquitetura.

Seguir alinhamento com os objetivos do produto:

- Conduzir análises para determinar necessidades de integração, bem como projetar e planejar integrações;
- Apoiar gerencialmente a execução e fiscalização de contratos de TIC na sua área de especialidade;
- Trabalhar com foco no valor de negócio, compreendendo o impacto de cada funcionalidade para o usuário final.

Produzir documentação técnica e suporte ao time:

- Preparar documentação técnica;
- Documentar soluções, apoiar colegas em dúvidas técnicas e contribuir para o

reuso e manutenção sustentável do código.

Arquiteto de Software

Atua como referência técnica estratégica, apoiando decisões que garantam escalabilidade, manutenibilidade, segurança e integração sistêmica das soluções desenvolvidas.

No contexto ágil, é parte integrante do Time de Desenvolvimento, colaborando com os demais membros para entregar incrementos de valor com qualidade e alinhamento técnico à visão do produto.

Principais responsabilidades:

- Definir e evoluir a arquitetura de soluções: propor e manter uma arquitetura técnica coerente com os objetivos do produto, considerando padrões, integrações, desempenho e segurança;
- Atuar de forma colaborativa com o time: trabalhar lado a lado com desenvolvedores(as), QA, DevOps e analistas, promovendo decisões técnicas alinhadas ao time e à realidade do negócio;
- Facilitar a adoção de boas práticas: incentivar a aplicação de padrões arquiteturais, testes, versionamento, integração contínua e documentação;
- Apoiar o Product Owner e o Scrum Master: esclarecer restrições técnicas, avaliar viabilidade de soluções e colaborar com a priorização de backlog em alinhamento com capacidades técnicas e riscos;
- Promover a visão sistêmica: avaliar impactos de novas funcionalidades em sistemas legados, integrações e infraestrutura existente.

Administrador de Dados

Esse perfil atua na garantia da qualidade das estruturas dos metadados das soluções alinhadas aos padrões de arquitetura de dados da organização, apoia na organização da informação corporativa objeto das aplicações em desenvolvimento, na garantia da integração e na aplicação das melhores práticas de administração de dados corporativos. São atividades comumente executadas por esse perfil (sem se limitar a essas):

Principais responsabilidades:

- Acompanha projetos e manutenção de bancos de dados;
- Desenvolve projetos de banco de dados e propõe melhorias;
- Analisa e soluciona problemas de dados;
- Elabora modelos e dicionários de dados;
- Executa scripts para manutenção de dados;
- Cria ambientes de banco de dados;
- Gera relatórios de administração de dados;
- Concede permissões e apoia times no modelo ágil;
- Projeta modelos de dados (conceitual, lógico e físico);
- Acompanha e orienta as equipes durante a modelagem de dados;
- Avalia modelos de dados produzidos pelas equipes de desenvolvimento;
- Apoia na busca e utilização de informações corporativas e compartilhadas;
- Dissemina os conceitos das entidades representadas nos modelos de dados;
- Mantém atualizados os repositórios de modelos de dados e metadados;
- Propõe mudanças na arquitetura corporativa de dados;
- Realiza estudos sobre a análise de impacto das alterações propostas nos modelos de dados corporativos e compartilhados;
- Emite relatórios técnicos e pareceres sobre o uso dos metadados nos âmbitos conceitual e lógico;
- Apoia os demais profissionais nas atividades referentes à qualidade de dados e gestão de dados mestres e de referência;
- Apoia na elaboração de Vocabulário e Glossário de dados, metodologia de gestão e governança de dados e demais documentos relativos à gestão de dados;
- Apoiar equipes multidisciplinares com mindset ágil.

Analista de Teste Qualidade

Esse perfil atua na garantia da entrega de software com alta qualidade, planejando, implementando e automatizando os testes de software e de garantia de qualidade de software. O analista de Teste e Qualidade busca desenvolver planos de teste, criar casos de teste, escrever código de automação de teste e relatar resultados, avaliar a qualidade técnica e funcional dos produtos, identificar riscos e possíveis falhas relacionadas aos códigos e funcionalidades entregues. São atividades comumente executadas por esse perfil

(sem se limitar a essas):

- Desenvolver planos de teste detalhados com base nos requisitos do projeto e identificar cenários de teste, criar casos de teste e definir estratégias para cobertura de testes;
- Planejar testes manuais e automatizados, considerando os diferentes níveis (unitários, integrados, sistema e aceitação);
- Realizar testes manuais em diferentes ambientes, simulando cenários reais de uso;
- Escrever, manter e executar scripts de automação utilizando ferramentas específicas e executar scripts de automação para testes funcionais, de desempenho, carga e regressão;
- Documentar e relatar os resultados dos testes, evidenciando falhas e inconsistências encontradas;
- Configurar e gerenciar ambientes e pipelines de teste automatizados;
- Colaborar na integração de testes automatizados nos processos de CI/CD;
- Avaliar a qualidade técnica e funcional do software entregue;
- Identificar riscos e propor soluções para mitigar falhas potenciais;
- Analisar métricas de teste e sugerir melhorias no processo de desenvolvimento;
- Trabalhar em estreita colaboração com desenvolvedores, Product Owners e Scrum Masters para alinhar expectativas de qualidade;
- Revisar e entender histórias de usuário e critérios de aceitação para garantir a cobertura de teste;
- Participar de reuniões de planejamento e refinamento para antecipar cenários de teste;
- Registrar planos, casos e resultados de teste em ferramentas apropriadas;
- Gerar relatórios claros e objetivos sobre a qualidade do produto, destacando problemas encontrados e ações corretivas;
- Explorar novas ferramentas, técnicas e metodologias de teste para otimizar processos;
- Estar atualizado sobre tendências e melhores práticas em testes e automação;
- Reportar, acompanhar e priorizar defeitos em sistemas de controle de bugs;
- Validar a correção de erros e realizar testes de regressão para garantir a estabilidade;
- Apoiar equipes multidisciplinares com mindset ágil.

Analista de UX/UI

Esse perfil atua na criação de soluções tecnológicas para melhorar a experiência do usuário de um produto ou serviço de software. Atua também na definição das características de interface com o usuário (design), de modo a garantir usabilidade e disposição da informação no meio de comunicação. São atividades comumente executadas por esse perfil (sem se limitar a essas):

- Realizar entrevistas, pesquisas e testes com usuários para identificar necessidades, comportamentos e problemas, conduzir análises de usabilidade e revisar feedbacks de usuários para aprimorar a experiência;
- Mapear jornadas do usuário (user journey) e construir personas representativas;
- Implementar Design de Experiência do Usuário (UX) e criar fluxos de navegação (user flows) claros e eficientes;
- Definir e documentar a arquitetura da informação de sistemas e produtos digitais, propor soluções para melhorar a experiência em funcionalidades complexas;
- Design de Interface do Usuário (UI): criar wireframes, protótipos de baixa e alta fidelidade e mockups; Desenvolver interfaces visuais atraentes, respeitando os princípios de design e identidade visual do produto;
- Garantir a consistência visual e funcional utilizando guias de estilo e bibliotecas de componentes;
- Planejar e executar testes de usabilidade com usuários reais ou simulados e coletar métricas e feedbacks para validar soluções propostas e fazer ajustes, realizar testes A/B para comparar diferentes abordagens de design;
- Trabalhar em conjunto com desenvolvedores, Product Owners, Scrum Masters e stakeholders;
- Garantir que as soluções projetadas sejam tecnicamente viáveis e alinhadas com os objetivos do negócio;
- Participar de reuniões de planejamento e refinamento para entender requisitos e limitações técnicas;
- Estabelecer e manter guias de estilo, padrões de design e bibliotecas de componentes reutilizáveis;
- Incorporar tendências de design e boas práticas de acessibilidade (WCAG, por exemplo), garantir que os designs sejam responsivos e adaptáveis a diferentes dispositivos e plataformas;
- Liderar iniciativas de design em projetos complexos ou estratégicos e atuar como

referência técnica e ponto de contato para decisões de UX/UI na organização.
- Acompanhar métricas de engajamento e desempenho das interfaces criadas e identificar oportunidades de melhoria contínua e propor evoluções no design;
- Garantir que as soluções entregues acompanhem mudanças nas necessidades do usuário e do mercado;
- Garantir que os textos sejam claros, concisos e úteis, criar/editar textos para botões, menus, campos de formulário, mensagens de erro e notificações;
- Criar microcopy, ou seja, pequenos trechos de texto que guiam o usuário em manuais, tutoriais e passo a passo nas aplicações/produtos;
- Criar um guia de escrita com estilos e diretrizes para manter a consistência na comunicação.

Analista de Negócios/ Requisitos

Esse perfil atua na identificação, definição e documentação de processos de negócios e de requisitos de software a serem implementados. O analista de negócio busca assegurar uma ligação consistente entre as equipes de negócios e a equipe de desenvolvedores, facilitando a comunicação e auxiliando no aprofundamento do domínio do negócio objeto da implementação. Atua, também, na propositura de funcionalidades e na organização das informações, no comportamento e fluxo do processo da aplicação satisfazendo as necessidades de negócio declaradas e não declaradas. São atividades comumente executadas por esse perfil (sem se limitar a essas):

- Levantar necessidades de negócio;
- Levantar e analisar histórias de usuário;
- Atuar em conjunto com os times de tecnologia e negócio;
- Gerenciar, fatiar, descartar e priorizar o backlog do produto – em apoio ao dono do produto;
- Definir objetivos de sprints;
- Definir critérios de aceitação de histórias de usuários codificadas;
- Conduzir a homologação das entregas junto ao gestor de negócio;
- Apoiar os usuários quanto a utilização e criação/atualização de manuais utilização de sistemas.

03 FLUXOS DE DESENVOLVIMENTO NO FNDE

3. Fluxos de Desenvolvimento no FNDE

O FNDE usa o Scrum em todas as demandas, tanto para novos projetos quanto para manutenção. Abaixo estão os passos gerais para novos projetos e evoluções.

3. 1 Novos Projetos

Os novos projetos começam com uma solicitação formal da área de negócio feita pelo SEI. Essa solicitação é encaminhada para o Escritório de Projetos, que verifica se a solicitação está incluída no PDTIC. Se for uma nova demanda de desenvolvimento, será necessário preencher o Termo de Abertura de Projeto (TAP), e a demanda será enviada para a CGSD.

No caso de projetos de novas soluções, independente do seu tamanho, a CGSD pode realizar uma Lean Inception, na qual serão definidos a visão do produto, seus objetivos e o Produto Mínimo Viável (MVP). O MVP e os incrementos formam o Backlog do Produto. Após a realização da Lean, as sprints de desenvolvimento são então iniciadas.

Transformar uma ideia em uma nova solução de software é um processo que consiste em várias etapas de desenvolvimento, cada uma com seus desafios significantes.

Estruturação de Dados e Arquitetura de Software

Um dos primeiros e mais cruciais desafios é decidir como estruturar os dados e projetar a arquitetura do software. É como construir a planta de uma casa antes de assentar o primeiro tijolo. Uma arquitetura bem definida é a base para um sistema robusto, escalável e fácil de manter. Perguntas como “Quais dados precisamos armazenar?”, “Como esses dados se relacionam?”, “Que tipo de banco de dados é o mais adequado?” e “Como as diferentes partes do sistema se comunicarão?” são fundamentais aqui. Uma decisão errada nesta fase pode levar a retrabalho custoso e limitações futuras, impactando a performance e a capacidade de expansão da solução.

Implementação dos Detalhes do Projeto

Depois de definir a arquitetura, o próximo passo é implementar os detalhes do projeto. Isso envolve traduzir os conceitos abstratos em componentes concretos. Aqui, o desafio reside em garantir que cada parte do sistema, desde as interfaces de usuário até os

módulos de processamento de dados e as integrações com outros sistemas, seja desenvolvida de forma coesa e eficiente. É preciso lidar com a complexidade de cada funcionalidade, garantindo que o código seja limpo, modular e siga as melhores práticas de desenvolvimento, o que muitas vezes exige um alto nível de detalhe e precisão.

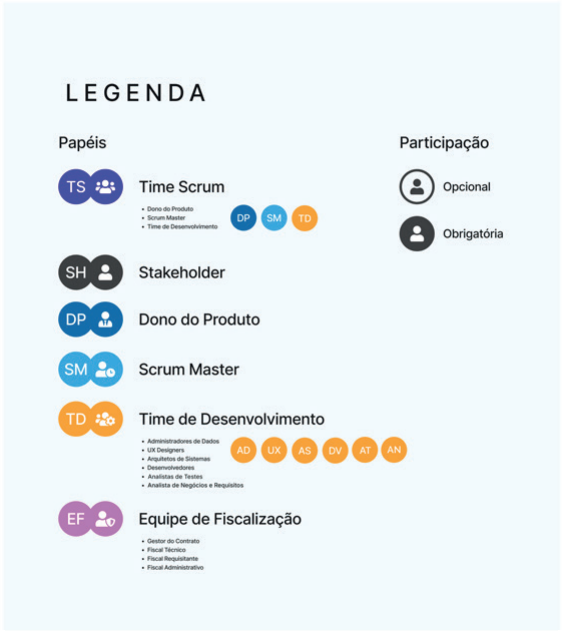
Desenvolvimento - Tradução para a Linguagem de Programação

Em seguida, vem o desafio de como traduzir o projeto para a linguagem de programação escolhida. Cada linguagem tem suas particularidades e portanto a equipe precisa ter um conhecimento aprofundado da linguagem e das ferramentas selecionadas para implementar o design de forma eficaz. Isso não se trata apenas de escrever código, mas de otimizá-lo para performance, segurança e manutenibilidade. Além disso, a escolha da linguagem e das bibliotecas e frameworks pode ter um impacto significativo na velocidade do desenvolvimento e na qualidade do produto final.

Realização dos Testes

Por fim, e não menos importante, está o desafio de como realizar os testes. Testar uma nova solução não é apenas verificar se ela funciona, mas garantir que ela atenda a todos os requisitos, seja robusta contra erros e falhas, e ofereça uma experiência de usuário satisfatória. Isso envolve diferentes tipos de testes: unitários, de integração, de sistema, de aceitação e de performance, entre outros. O desafio é criar um plano de testes abrangente, automatizar o máximo possível e interpretar os resultados para identificar e corrigir falhas de forma eficiente. Um processo de teste inadequado pode levar ao lançamento de um produto com bugs, impactando negativamente a reputação e a satisfação do cliente.

Fluxo de Novos Projetos



3.2 Demandas Evolutivas

As demandas evolutivas são melhorias ou novas funcionalidades que surgem ao longo do desenvolvimento. A fase de manutenção, portanto, começa depois que o software já foi desenvolvido e está em uso. Manutenções são necessárias para corrigir erros, fazer adaptações conforme o software evolui e atender a novas demandas dos clientes. Isso permite que a equipe adapte o produto continuamente, tornando-o mais útil e alinhado à realidade.

Essa fase reaplica os passos de definição e desenvolvimento, mas dentro do contexto do software já existente.

Tipos de Manutenção

- Correção: a manutenção corretiva ajusta o software para corrigir defeitos.
- Adaptação: a manutenção adaptativa faz mudanças no software para que ele se adapte a novas situações, como uma nova regra de negócio, lei ou norma.
- Evolução: a manutenção perfectiva melhora o software, adicionando novas funcionalidades além das originais, tornando-o mais útil ao longo do tempo.

3.3 Débito técnico (ou dívida técnica)

O débito técnico é uma metáfora criada por Ward Cunningham (um dos autores do Manifesto Ágil). Ela descreve o custo futuro de se escolher uma solução fácil ou rápida agora, em vez de uma abordagem melhor e mais robusta que levaria mais tempo.

Ela é utilizada para classificar algum trabalho técnico que é realizado sem as condições ideais, mas que, em algum momento próximo, terá que ser melhorado.

Assim como uma dívida financeira, o débito técnico acumula juros: quanto mais tempo ele permanece no código, mais difícil e caro fica implementar novas funcionalidades, pois a equipe precisa “pagar” a complexidade extra toda vez que mexe no sistema.

As causas para a existência do débito técnico são várias, como pressão de negócio (lançar rápido), falta de conhecimento do time, falta de padrões de arquitetura, evolução tecnológica ou falta de testes.

Independente da causa, essa dívida deverá ser identificada, visualizada, gerenciada e paga o quanto antes.

Para isso, deverá ser criado na ferramenta de ALM tickets específicos para refatoração e melhoria técnica.

Se possível, deve ser reservado um percentual da sprint (ex: 10%) para pagamento de débito técnico e melhorias de infraestrutura.3.2 Demandas Evolutivas

LEGENDA

Papéis

- TS Time Scrum
  - Dono do Produto
  - Scrum Master
  - Time de Desenvolvimento
  - DP SM TD
- SH Stakeholder
- DP Dono do Produto
- SM Scrum Master
- TD Time de Desenvolvimento
  - Administradores de Dados
  - UX Designers
  - Arquitetos de Sistemas
  - Desenvolvedores
  - Analistas de Testes
  - Analista de Negócios e Requisitos
  - AD UX AS DV AT AN
- EF Equipe de Fiscalização
  - Gestor do Contrato
  - Fiscal Técnico
  - Fiscal Requisitante
  - Fiscal Administrativo

Participação

- Opcional
- Obrigatória

Fluxo Evolutivo

3.3 Demandas de Sustentação

A sustentação de sistemas envolve todas as atividades necessárias para garantir que os sistemas e aplicativos continuem funcionando bem em um ambiente de produção. É essencial para manter os sistemas operando de forma eficiente e confiável.

A sustentação de sistemas é diferente da sustentação de infraestrutura. Enquanto a sustentação de sistemas foca nos aspectos funcionais e lógicos dos aplicativos e software, a sustentação de infraestrutura lida com os componentes físicos e de rede que suportam a TI.

A sustentação não inclui mudanças nas funcionalidades ou nas regras de negócio, mas assegura que tudo funcione conforme o esperado.

Essas atividades incluem:

- **Manutenção corretiva:** reparar falhas ou problemas nos sistemas que estão impedindo seu funcionamento.
- **Backup e recuperação de dados:** garantir que os dados sejam armazenados de forma segura e possam ser recuperados em caso de perda ou dano.

A sustentação deve ser ajustada conforme a importância e a disponibilidade dos sistemas. Sistemas críticos, essenciais para a organização, precisam de alta disponibilidade. Sistemas menos críticos podem ter menor disponibilidade e exigir menos recursos. A disponibilidade se refere ao tempo em que um sistema está operando sem interrupções, como 99,9% do tempo em um ano.

3.4 Modernização de Aplicações

Há no FNDE há muitas aplicações legadas, desktops, com processos e regras de negócios antigas, linguagens defasadas, muitos débitos técnicos, incluindo restrições de segurança. Em sua maior parte, esses sistemas perduram por possuir grande valor de negócio.

Entretanto, modernizar essas aplicações, construindo novas soluções, não é tarefa simples, nem rápida.

Em primeiro lugar é necessário fazer uma análise a fim de priorizar quais sistemas serão migrados. Para isto, recomenda-se utilizar a metodologia TIME, do Gartner.

Não é escopo deste documento aprofundar nessa metodologia. Basta saber que o nome TIME vem do acrônimo de Tolerate, Invest, Migrate e Eliminate (tolerar, investir, migrar ou eliminar).

A migração é recomendada nos casos em que o software tem alto valor de negócio, mas a qualidade técnica é baixa (legado, difícil de manter, inseguro).

Caso a análise demonstra a necessidade de migração, é importante ter em mente questões como "Onde estamos", "Quais são os aspectos mais relevantes que precisamos corrigir em curto e longo prazo".

Greenfield

Quando a análise TIME resultar na necessidade de construção de uma nova aplicação, mais moderna, com arquitetura sustentável, e com plataforma integrada, recomenda-se a utilização da técnica Greenfield.

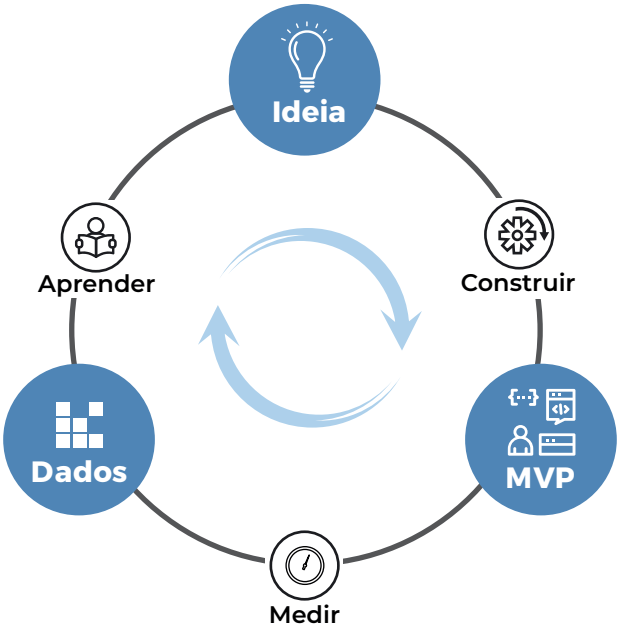
No caso de legados, a aplicação da Lean Inception não é tão recomendada, a não ser que seja necessária uma nova visão sobre o produto. Outro ponto a considerar aqui é a dificuldade de estabelecer um MVP, uma vez que a solução tem que ser substituída.

3.5 MVP - Mínimo Produto Viável

O conceito de Mínimo Produto Viável (MVP), do inglês Minimum viable Product é central no desenvolvimento de software moderno, unindo a fundamentação teórica da engenharia de software e gestão com a aplicação prática disseminada por comunidades ágeis.

Em Engenharia de Software o MVP é uma ferramenta para mitigar incertezas técnicas e de mercado, servindo para testar a "hipótese mais arriscada" do projeto antes de investir recursos massivos em arquiteturas complexas.

O produto é construído de forma gradual, com funcionalidades validadas, sendo adicionadas ao produto consolidado já existente. A entrega contínua e incremental proporciona o aumento do valor do produto ao longo do tempo."



"O MVP está na interseção entre valioso, usável e factível, representando, respectivamente, o interesse do negócio, a aceitação (e admiração) dos usuários e o que é possível construir"

Para saber mais: https://caroli.org/tipos-de-mvp/

04 LEAN INCEPTION

4. Lean Inception

Lean Inception é a denominação de um método desenvolvido por Paulo Caroli para auxiliar a definição de um Mínimo Produto Viável (MVP). Ele mescla atividades de Inception e de Design Thinking a fim de auxiliar equipes a entender os objetivos do produto, quais são os principais usuários, qual é o escopo funcional de alto nível e a definir as funcionalidades de um MVP.

De acordo com o autor, em seu livro "Lean Inception: como alinhar pessoas e construir o produto certo", publicado pela Editora Caroli:

> *"Projetos ágeis enfatizam a entrega contínua e antecipada de um software valioso cujo valor vem dos objetivos de negócios e das necessidades dos clientes. LeanStartup corrobora com isso através da liberação incremental de um MVP (minimum viable product ou produto mínimo viável) - uma versão simples do produto que é entregue aos usuários para validar as principais premissas do negócio.*
> *(...)*
> *Mesmo num projeto ágil, antes de sair fazendo, é preciso alinhar e definir os objetivos, as estratégias e o escopo do produto.*
> *(...)*
> *A Lean Inception é útil quando o time precisa desenvolver um MVP e criar um produto de forma iterativa e incremental. Apesar de o termo ser frequentemente mal entendido, a característica principal de um MVP é que fazemos algo para aprender se vale a pena continuar construindo o produto. Então escolhemos funcionalidades que nos ajudem a validar o que é valioso para nossos usuários. Para isso, precisamos entender quem são os usuários, para qual atividade que eles fazem que o produto suporta, se consegue se adequar e como medir se eles acham o produto útil ou não.*

Na FNDE Ágil, a Lean Inception deverá ser a primeira etapa realizada no início do desenvolvimento de um novo projeto de software. São utilizadas dinâmicas colaborativas e lúdicas, podendo ser realizado de forma presencial ou remota. O workshop ajuda a escolher e validar as funcionalidades que são realmente valiosas para seus usuários.

Evento
Lean Inception

Participantes:
- Stakeholder
- Dono do Produto
- Scrum Master
- Equipe de Desenvolvimento

Até 40 horas

São objetivos da Lean Inception:

1. Alinhar e definir objetivos, estratégias e escopo do produto;
2. Criar uma lista ordenada de Histórias de Usuário (HUs)
3. Compreender o Mínimo Produto Viável (MVP)

Quando fazer?

O autor recomenda realizar o workshop em dois casos:

- no começo de novos projetos, quando há apenas um esboço de produto em mente;
- no caso de projetos inovadores e de alto impacto para o negócio

É muito importante realizar a Lean Inception antes de iniciar seu desenvolvimento, a fim de definir:

- Qual produto será construído;
- Quais são as funcionalidades do produto.

De acordo com o autor, ao final do workshop, a Lean Inception permite que a equipe:

- descreva a visão do produto;
- priorize os objetivos do produto;
- descreva os principais usuários, seus perfis e suas necessidades;
- explore as principais funcionalidades;
- compreenda os níveis de incerteza, esforço, valor para o usuário e valor de negócio por funcionalidade;
- descreva as jornadas mais importantes dos usuários;
- crie um plano de entrega incremental do produto, impulsionado pelo conceito de MVP.

Participantes

- **Stakeholder -** qualquer pessoa impactada pelo projeto. São pessoas altamente interessadas no direcionamento e no resultado da Lean Inception, mas que não têm tempo para participar de todas as sessões. É importante que esses participem das sessões de kick-off e de showcase.
- **Membro Ativo -** qualquer pessoa diretamente envolvida na compreensão e implementação do produto. São os Donos de Produto, o coordenador da CGSA responsável pelo projeto, os Analistas de Negócio, e o Time de Desenvolvimento
- **Facilitador -** é o mediador, a pessoa que explicará as atividades, esclarecerá dúvidas e propiciará o fluxo de ideias e conversações. Ele deve ser neutro, sem intervir na tomada de decisões. No FNDE o facilitador geralmente será o Scrum Master do Projeto. Ele também é responsável por planejar e organizar o workshop.

É importante que haja ao menos dois participantes com conhecimento nas principais perspectivas: Visão de Negócio, Experiência do Usuário e Tecnologia.

Preparação

A Lean Inception é conduzida pelo facilitador, que no caso do FNDE é o Scrum Master.

Ela é composta por dinâmicas com tempo determinado, utilizando técnicas de brainstorm e construção conjunta. Para melhor estimular ideias e criatividade, deve ser criado um ambiente agradável e lúdico, no qual todas as pessoas são convidadas a participar.

Antes da agenda é importante os facilitadores realizarem as seguintes ações de preparo:

1. Definir: quem será o facilitador?
2. Definir: presencial ou remota?
3. Definir: quais serão os participantes?
4. Criar agendas com os participantes
5. Se for remota, deve ser criar o quadro da apresentação (ver modelo abaixo)
6. Dar permissão para os participantes no quadro
7. Ensinar como trabalhar na ferramenta do quadro

Agenda

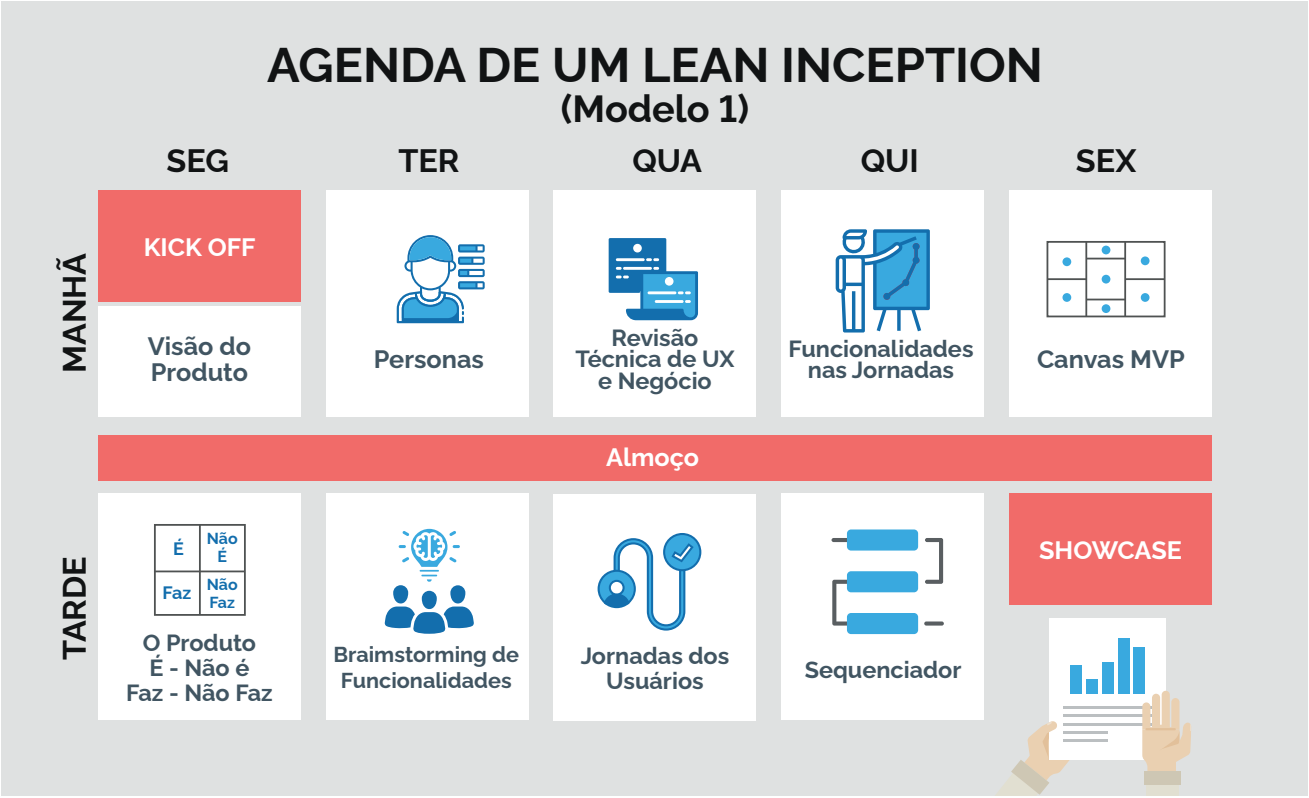
O workshop é dividido nas seguintes etapas:

- kick-off
- visão de produto;
- o produto é-não-é-faz-não-faz;
- personas;
- brainstorming de funcionalidades;
- revisão técnica, de UX e de negócio;
- jornadas do usuário;
- funcionalidades nas jornadas;
- sequenciador;
- canvas MVP;
- apresentação dos resultados

O workshop inicia com a reunião de kick-off. Este evento começa com as apresentações dos participantes, de preferência utilizando uma atividade quebra-gelo. Depois o facilitador apresenta uma visão geral das atividades.

Em seu livro, assim como no blog, Caroli sugere duas agendas para a Lean Inception. No primeiro modelo, o "Brainstorming de Funcionalidades" ocorre antes das "Jornadas dos Usuários". Essa abordagem é recomendada quando a criação das funcionalidades é fortemente influenciada pelos objetivos de negócio e pelas necessidades das personas. As "Jornadas dos Usuários" e as "Funcionalidades nas Jornadas" são então utilizadas para validar ou aprimorar a experiência do usuário. No segundo modelo, as "Jornadas dos Usuários" são realizadas antes do "Brainstorming de Funcionalidades", seguido pela "Revisão Técnica de UX e Negócio". Só então as funcionalidades são inseridas nas jornadas. Essa sequência é recomendada quando a ideação das funcionalidades deve ser fortemente guiada pelas jornadas dos usuários.

O FNDE utiliza como padrão o segundo modelo:



Visão de Produto

Para...	Que...	O...	É um...	Que...	Diferentemente de...	Este...
- Público-alvo - Cliente final - Usuário - Persona	- Situação a ser melhorada - Problema a ser resolvido	Nome do produto	Tipo / Categoria de produto	O que faz / principal benefício	Alternativa	Principal diferencial competitivo

A visão do produto ajuda a trilhar o caminho inicial. Nesta etapa a equipe deverá explicar o que é o projeto ou produto a ser realizado, de forma simples e direta. Devem ser listadas informações importantes para comunicar a ideia do produto. O objetivo é definir a essência do valor de negócio do produto.

É / Não É / Faz / Não Faz

> *"Muitas vezes é mais fácil descrever o que alguma coisa não é ou não faz. A atividade É-Não é-Faz-Não faz (ENFN) busca classificações sobre o produto, seguindo as quatro diretrizes, indagando, especificamente, cada aspecto positivo e negativo sobre o produto ser ou fazer algo."*
>
> *(Do livro 'Lean Inception: como alinhar pessoas e construir o produto certo', de Paulo Caroli)*

Essa atividade auxilia a esclarecer o produto e a construir uma visão alinhada sobre o que o produto faz e o que ele não faz.

Nesta dinâmica é solicitado que cada participante descreva o produto, escrevendo as características em post-its e inserindo-as nas devidas áreas.

Dica

Ao preencher é “para descrever o produto como substantivo ou adjetivo, coloque o post-it no ”É”. Mas se for um verbo, indicando uma ação, coloque no “Faz”.

Exemplos:

- O produto é um site responsivo, gratuito, app mobile
- O produto não é complicado de usar.

Outra proposta é dividir os participantes em dois grupos. O primeiro fará uma reflexão sobre o que o produto é e o que ele faz. O segundo reflete sobre o que o produto não é e o que ele não faz. Após o tempo determinado pelo facilitador, é feito um brainstorming colaborativo para alinhar e definir as características e funcionalidades principais do produto.

Ao final é preenchido um quadro semelhante ao apresentado a seguir:

Objetivos

Essa atividade auxilia a esclarecer os objetivos do produto. Para tal, os participantes devem escrever em post-it, de forma conjunta, três objetivos do produto.

Personas

Uma persona representa um usuário (ou grupo de usuários) do produto ou do serviço.

Essa atividade objetiva, portanto, identificar os usuários do produto ou serviço, com a descrição do seu papel e das suas necessidades específicas.

A ideia é despertar uma representação realista de usuários, a fim de auxiliar o time a descrever funcionalidades do ponto de vista de quem realmente irá interagir com o produto final.

Jornadas de Usuários

Nesta etapa, para cada uma das personas anteriores, são listados os passos e atividades realizadas por eles, relacionados ao produto que está sendo construído. Estabelecer as jornadas dos usuários é importante para olhar o produto sob a perspectiva do usuário, entender quais são os objetivos dela ao usar o produto.

Duas perguntas orientadoras são:

- Como esta persona começa o seu dia?
- Qual objetivo esta persona quer alcançar?

Brainstorming de Funcionalidades

Esta atividade objetiva listar as funcionalidades (features) necessárias para que o produto atenda às necessidades dos usuários:

- O que deverá ter no produto para atender às necessidades das personas?
- Quais funcionalidades devemos construir para atingir o objetivo do produto?Para auxiliar esse processo pode ser feita uma tabela com os objetivos no eixo horizontal e as personas no eixo vertical.

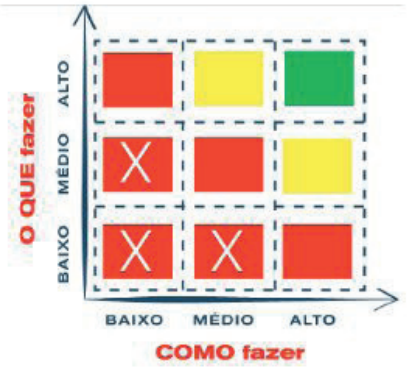
Revisão Técnica, de Negócio e de Experiência

Esforço	E	EE	EEE	Esforço para o Time de Desenvolvimento.
Negócio	$	$$	$$$	Valor para o Negócio.
UX	♥	♥♥	♥♥♥	Valor para o Usuário.

Nesta etapa cada funcionalidade é avaliada de acordo com 4 critérios: esforço, valor de negócio, valor de UX e o nível de confiança sobre o que é a funcionalidade e como construí-la.

Para o nível de confiança, considera-se “o que fazer” e “como fazer”. São atribuídas cores, de acordo com o “gráfico do semáforo”:

Verde: pode ir tranquilo
Amarelo: prestar atenção
Vermelho: para e espere antes de prosseguir



O eixo Y trata sobre a clareza do que é funcionalidade, o que o negócio ou o usuário querem do item. Já o eixo X trata do entendimento da equipe quanto aos desafios técnicos, às dependências e aos requisitos de infraestrutura.

Perguntas norteadoras:

Para o gráfico

- Eixo X: Quão confiante você está sobre COMO fazer a funcionalidade?
- Eixo Y: Quão confiante você está sobre O QUE o negócio e/ou os usuários querem dessa funcionalidade? Você já fez isso antes?

Como exemplo, vamos classificar uma funcionalidade chamada "cadastrar endereço":

- Esforço: médio (EE);
- Valor de UX: alto (♥♥♥);
- Valor de negócio: baixo ($);
- Confiança: alta (verde).

Ao final da atividade as funcionalidades em cartões vermelhos com X representam altíssimos riscos para o projeto.

Sequenciador

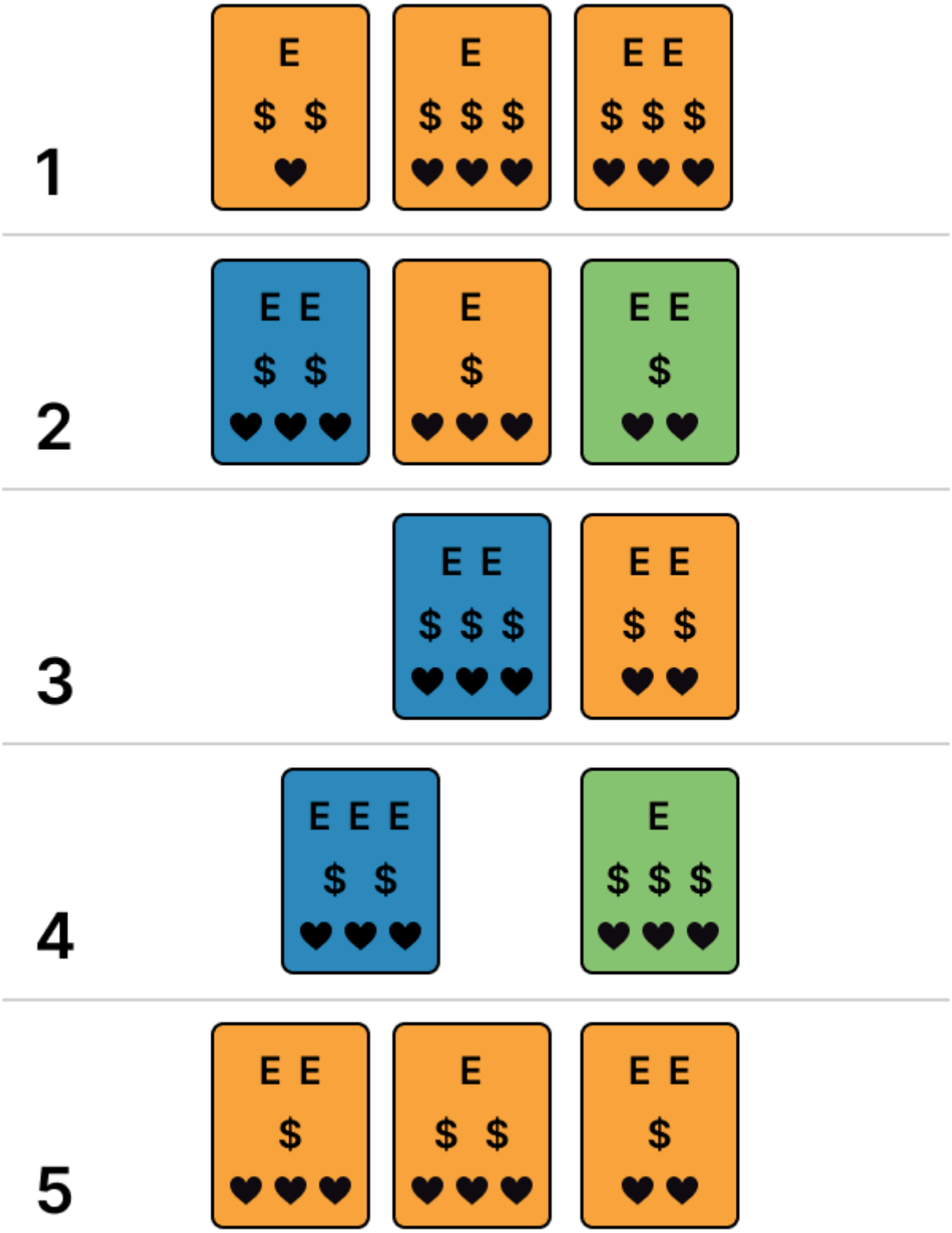
Dadas as funcionalidades e sua classificação, deve-se agora priorizá-las. O objetivo é executar primeiro a funcionalidade mais impactante, para que atinja objetivos o mais cedo possível.

Uma pergunta norteadora é: "qual é a combinação mínima de funcionalidades que devem ser disponibilizadas para validar um pequeno conjunto de hipóteses sobre o negócio?"

Essa priorização é feita por meio da atividade denominada "Sequenciador", na qual devem ser obedecidas as seguintes regras:

- Uma onda pode conter, no máximo, 3 cartões;
- Uma onda NÃO pode conter mais de 1 cartão vermelho;
- Uma onda NÃO pode conter SOMENTE amarelos ou vermelhos;
- A soma de esforço dos cartões não pode ultrapassar 5E;
- A soma de valor dos cartões não pode ser menos de 4$ e 4♥;
- Se um cartão depende de outro, esse outro deve estar em alguma onda anterior.

Exemplo de ondas no Sequenciador.



05 EVENTOS SCRUM

5. EVENTOS SCRUM

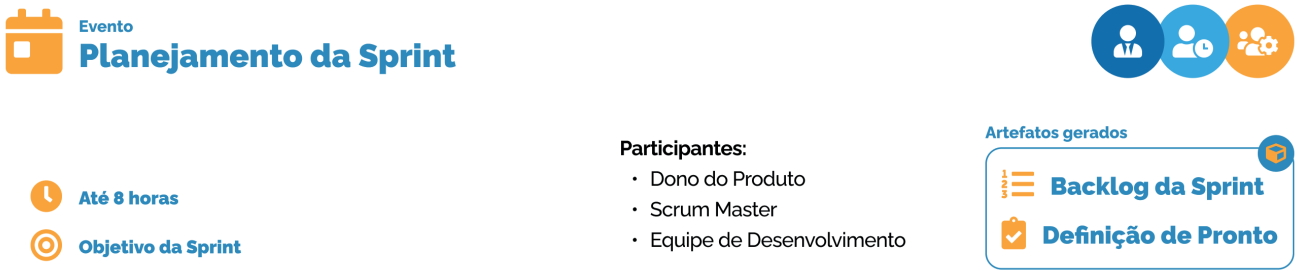
Eventos Scrum

No Scrum, o trabalho é dividido em ciclos curtos chamados Sprints. Cada Sprint dura até quatro semanas e deve entregar pelo menos uma melhoria no produto. Quanto mais curtos forem os ciclos, mais rápido o progresso e a validação do trabalho.

Cada Sprint inclui as seguintes reuniões:

- Planejamento da Sprint;
- Reunião Diária durante a Execução;
- Revisão;
- Retrospectiva.

Essas reuniões serão explicadas nos próximos itens.



A reunião começa pela leitura do item mais prioritário do Backlog do Produto. Caso seja necessário, o PO pode explicar melhor o item, detalhá-lo, a fim de garantir que todos entendam o que precisa ser feito. Esclarecidas as dúvidas, o time calcula o esforço necessário para transformar o item em uma entrega pronta. Esse esforço é calculado em Pontos de História, ou *Story points*, como será explicado mais adiante.
A equipe repete o processo acima com os próximos itens do Backlog do Produto, até que não caibam mais tarefas na *sprint*.

Um ponto importante no planejamento da *sprint* é que o foco está no **conteúdo a ser entregue durante o período, e não em quando cada tarefa será finalizada.** A prioridade está em atingir o objetivo da sprint como um todo, e não em prazos individuais.

A duração da *sprint* é definida, sendo sempre inferior a 1 mês.

Uma vez que o Objetivo da *Sprint* foi definido e aceito, ele não pode ser alterado. Durante a *Sprint*, a equipe trabalha de forma autônoma para concluir o que foi planejado. Nenhum item pode ser adicionado à *Sprint*, mesmo que seja urgente. Se surgir uma urgência, o Product Owner (PO) deve avaliar se a demanda pode ser incluída na próxima *Sprint*.

Etapas da reunião de planejamento – "o que" e "como"

Na primeira etapa (o "o quê"), a equipe analisa e avalia os itens mais importantes do backlog e define a meta da *Sprint*.

Na etapa de planejamento (o "como"), a equipe cria o backlog da *Sprint* e faz as estimativas usando o Planning Poker. Os itens do backlog são divididos em tarefas menores, focando na parte técnica.

O backlog da *Sprint* inclui os itens escolhidos e as tarefas necessárias para desenvolvê-los.

Meta ou Objetivo da Sprint

A Meta da *Sprint* é um objetivo claro que a equipe se compromete a alcançar durante a *Sprint*. Ela define o que será entregue e o valor que será adicionado ao produto ao final do período. A meta orienta e foca a equipe, ajudando todos a trabalharem juntos para alcançar um resultado concreto dentro do prazo.

Todos os participantes da reunião de Planejamento devem concordar com a Meta (ou Objetivo) da *Sprint*.

Sprint Backlog ou Backlog da Sprint

Durante a reunião de planejamento, é criada o Backlog da *Sprint*, que é uma lista dos itens selecionados e as tarefas planejadas para transformá-los em um incremento do produto. Esse backlog é organizado como um Quadro Kanban, mostrando claramente as tarefas a serem feitas, as que estão em andamento e as que foram concluídas.

O backlog da *sprint* é um subconjunto do backlog do produto, escolhido para ser desenvolvido na iteração atual.

A qualidade de um item de backlog deve ser avaliada tanto na entrada (Definição de Preparado) quanto na saída (Definição de Pronto).

Definição de Preparado (definition of ready)

A Definição de Preparado é um conjunto de critérios formais que um item do Backlog do Produto deve atender para ser considerado pronto para ser selecionado e trabalhado em uma *sprint*. Pode incluir critérios como:

- É claro? – todo o Time de Desenvolvimento compreende o item, sem ambiguidades?
- Foi estimado? – o esforço necessário para completar o item foi estimado?
- É pequeno? – é pequeno o suficiente para ser completado dentro de uma sprint?
- É testável?
- Possui valor?
- Os Critérios de Aceitação estão definidos?
- As dependências foram identificadas?

Definição de Pronto (Definition of done)

No Scrum, o progresso é medido pelos itens totalmente concluídos. Mesmo que 80% de uma tarefa esteja feita, ela não é considerada pronta até ser finalizada completamente. A "Definição de Pronto" (Definition of Done) deixa claro quando um incremento do produto pode ser considerado completo. Ela é fundamental para garantir a qualidade técnica.

Duração da Sprint

A duração da Sprint deve ser definida considerando a frequência de entregas ou feedbacks necessários. Sprints mais curtas aumentam a agilidade do projeto e reduzem o risco de desenvolver algo que não seja útil.

O quadro a seguir mostra uma recomendação de duração para as tarefas durante o planejamento da Sprint:


Sprint de 30 dias
• Planejamento: 8 horas
• O quê (objetivos e backlog): 4 horas
• Como (estratégias e tarefas): 4 horas
30
Sprint de 15 dias
• Planejamento: 4 horas
• O quê (objetivos e backlog): 2 horas
• Como (estratégias e tarefas): 2 horas
15
Sprint de 07 dias
• Planejamento: 2 horas
• O quê (objetivos e backlog): 1 hora
• Como (estratégias e tarefas): 1 hora
07

5.2 Execução

Durante a execução, a equipe trabalha para transformar cada item em uma entrega real do produto. O mais importante deve ser finalizado primeiro, e só depois se começa a trabalhar no próximo item. O quadro Kanban deve ser atualizado constantemente para acompanhar o progresso e ajustar o trabalho, se necessário. Porém, é importante não abrir mais de uma tarefa ao mesmo tempo.

> *"Um mantra para a agilidade de projetos deveria ser: pare de iniciar e comece a finalizar!"*

(Do livro 'Tire seu projeto do papel com Scrum', de Alexandre Magno)

Reunião Diária (Daily Meeting)

A Daily tem como objetivo acompanhar o progresso da equipe para alcançar a meta da Sprint e verificar o andamento das tarefas no backlog.

Essa reunião diária melhora a comunicação e a criatividade do time. Apenas o Time de Desenvolvimento é obrigado a participar, no papel de Scrum Master, garante que a reunião aconteça, e o Product Owner pode participar se quiser.

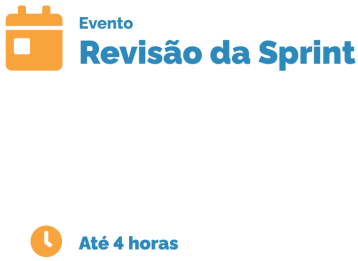
A Daily acontece durante a execução da Sprint e deve durar no máximo 15 minutos. É ideal que seja realizada sempre no mesmo horário, para atualizar o planejamento e avaliar o progresso.

Durante a reunião, a equipe responde a três perguntas:

- O que foi feito no dia anterior?
- O que será feito hoje?
- Quais são os impedimentos?

O foco é discutir os problemas, não as soluções.

Revisão da Sprint



Após o período de execução, no último dia da Sprint é realizada a Revisão, na qual são apresentados os incrementos de produto que estiverem totalmente prontos. O estado atual do produto é avaliado e são colhidos feedbacks. O Backlog do Produto e o progresso do projeto são atualizados.

Participantes: Dono do Produto, Analista de Negócios, Scrum Master e Time de Desenvolvimento.

Tempo de duração: até 4 horas

Objetivo: obter feedback do cliente sobre o produto que foi gerado durante a Sprint.

Atenção: o propósito da reunião não é a aprovação formal do Dono do Produto. Não é uma reunião de testes de aceitação

Dessa forma, os objetivos da Revisão da Sprint são:

- Conferir o resultado da Sprint;
- Verificar o desenvolvimento do produto;
- Transparência

A reunião inicia com o Scrum Master apresentando a meta da sprint e o quadro Kanban. Depois o Time de Desenvolvimento apresenta os itens desenvolvidos.

O Dono do Produto pode fazer perguntas e obter respostas.

O time também pode convidar os participantes a experimentarem o produto, a fim de estimular a fornecer feedback. Mas é importante salientar que a reunião não é para testes.

Após essa apresentação o Dono do Produto e o Time de Desenvolvimento verificam se o realizado atingiu a meta da Sprint.

Por fim, o Scrum Master deve mostrar o que foi realizado em comparação com o MVP ou com o roadmap do produto.

Retrospectiva da Sprint

A Retrospectiva é a última etapa de uma Sprint, onde a equipe reflete sobre como o trabalho foi realizado, identificando áreas para melhorar o processo.

Diferente da Revisão, o foco aqui é no processo, não no produto. Durante a retrospectiva, pode-se usar ferramentas online como Miro ou Mural, onde os participantes colocam post-its para destacar o que funcionou bem, o que não funcionou e ideias para a próxima Sprint.

É recomendado usar as técnicas do site Fun Retrospectives.

Participam da Retrospectiva: o Product Owner, o Analista, o Scrum Master e o Time de Desenvolvimento.

06 BACKLOG DO PRODUTO, ÉPICOS E HISTÓRIAS DE USUÁRIOS

6. Backlog do Produto, Épicos e Histórias de Usuários

Conforme citado anteriormente, durante o Planejamento da Sprint a equipe visualiza o backlog do produto e prevê quantas tarefas serão realizadas.

Mas o que é o Backlog do Produto? Qual é a sua composição?

> *"A qualquer momento, o backlog é a visão única e definitiva de tudo que a equipe poderia um dia vir a realizar, em ordem de prioridade."*
>
> *(Do livro 'Scrum: A Arte de Fazer o Dobro do Trabalho na Metade do Tempo', de Jeff Sutherland)*

Backlog do Produto

O **backlog do produto** lista todas as características, funções, requisitos, melhorias e correções que formam as mudanças que devem ser feitas no produto nas futuras versões. São todos os requisitos de qualquer mudança a ser feita no produto. Assim, o backlog de produto é uma lista priorizada, onde cada item é um potencial incremento do seu produto, ou seja, uma parte ou pedaço do seu produto. No topo da lista, estão os itens de ordem mais alta, que devem estar mais claros e mais detalhados do que os demais. Os itens mais prioritários são os únicos que precisam estar mais detalhados, granulares e estáveis, já que serão os primeiros a serem executados.

O backlog do produto é dinâmico, nunca está completo, e evolui ao longo de toda a vida do produto. Há contínuas alterações de detalhamento, entradas, saída e prioridades.

De acordo com o livro "Tire seu projeto do papel com Scrum", de Alexandre Magno:

> "No Scrum, (...) um artefato chamado Product Backlog, (...) além de nos ajudar a ter sucesso em ambientes de incerteza, ainda nos apoia no combate à paralisia analítica e na validação rápida do que é mais crítico no nosso projeto."

No FNDE, o backlog do produto é feito pelo Dono do Produto (PO).

Artefato
Backlog do Produto

É recomendado que o Backlog de Produto nunca seja representado por uma lista de atividades ou tarefas a serem completadas, mas sim como uma lista de pedaços do produto que devem ser construídos ou completados. O PO deve decidir como priorizar essa lista ao longo de todo o projeto devendo consultar todos os envolvidos e o time, para garantir que ele reflete tanto o que é desejado, quanto o que é possível de ser feito.

No caso de novos projetos, o primeiro Backlog do Produto é obtido após a realização da Lean Inception, na qual são definidos o Mínimo Produto Viável (MVP) e os incrementos posteriores.

6.1 Refinamento Contínuo do Backlog do Produto

Como serão os primeiros a serem executados, os itens no topo da lista devem estar mais claros tanto em tamanho quanto em detalhamento. Eles devem conter um conjunto de detalhes que ajudará o Time de Desenvolvimento a entender o que precisa ser feito.

Dessa forma, o Dono do Produto deve estar sempre trabalhando nos itens candidatos para a próxima sprint. Sugere-se que os itens que serão trabalhados em um futuro distante não sejam refinados antecipadamente.

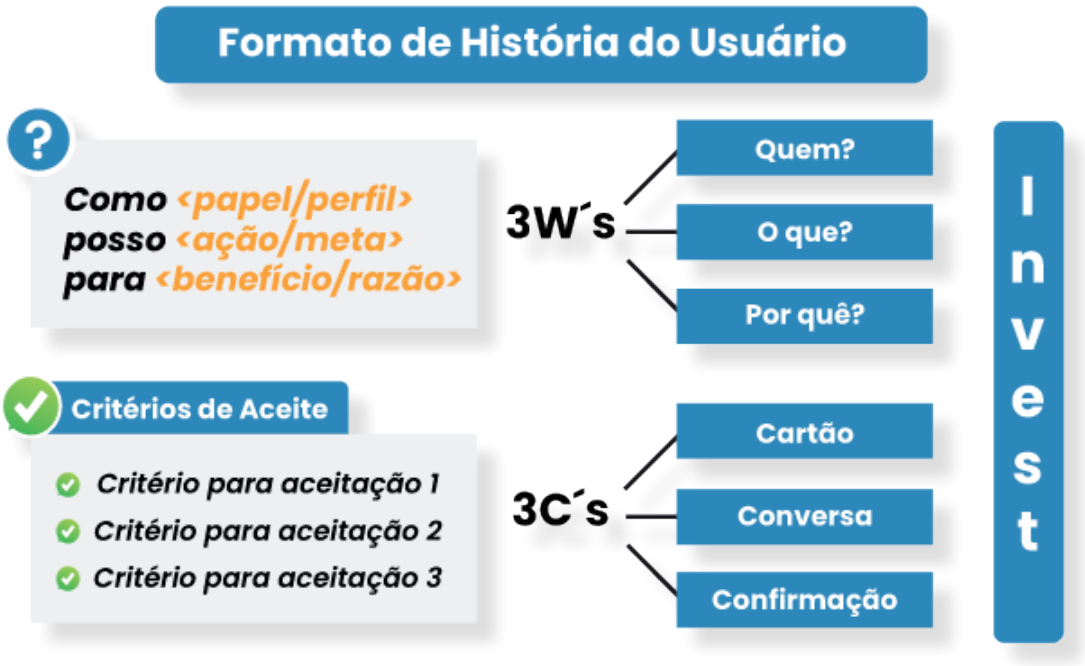
> *"Enquanto a equipe trabalha nos itens do backlog da sprint, em algum momento ela precisa se reunir com o Product Owner no que é chamado de refinamento do backlog. Esse é um momento crucial para o sucesso do Scrum. É nessa reunião que o Dono do Produto mostra todas as suas ótimas ideias para sprints futuros e trabalha com a equipe para deixar essas ideias prontas para execução." Eles decidem precisamente o que é o item e quais são os critérios para julgar se esse item está concluído (ou não)."*
>
> J.J. Sutherland

6.2 História de Usuário

> "a melhor maneira de construir um produto em conformidade com as necessidades do usuário é utilizar Histórias do Usuário."
>
> De acordo com Mike Cohn, em seu livro "User Stories Applied: For Agile Software Development",

A História do Usuário (HU ou User Story) é uma técnica que busca capturar os requisitos funcionais de um sistema sob a perspectiva do usuário final.

Ela é escrita pelo Dono do Produto (PO), em colaboração com os stakeholders e é usada para comunicar e documentar os requisitos em um formato fácil de entender. É uma descrição curta e concisa de uma funcionalidade do sistema, escrita na linguagem do usuário, e geralmente segue a estrutura "como um..." e "eu quero para...". Dessa forma, cada HU é um convite para uma conversa entre os interessados e os desenvolvedores, na forma de uma descrição de funcionalidades descrevendo funções, metas, benefícios e motivação. Ela diz 3 coisas sobre o requisito: quem, o que e por quê.



- Quem, no negócio, é a pessoa mais beneficiada com a funcionalidade?
- O que é exatamente essa funcionalidade ou característica?
- Qual o principal benefício que ela entrega para o negócio?

As Histórias do Usuário são utilizadas para definir os itens do Backlog do Produto e são priorizadas com base em seu valor e impacto para os usuários. Durante o planejamento da Sprint, elas são detalhadas em tarefas menores que são estimadas e atribuídas aos membros da equipe de desenvolvimento.

Acesse o templete no Confluence: www.fnde.confluence.com

Modelo INVEST

O ideal é que as Histórias de Usuário sejam independentes, negociáveis, valiosas, estimáveis, sucintas e testáveis (as iniciais das palavras formam, em inglês, o termo "invest"): .

- Independente – cada história não deve depender de outra para que possa ser projetada, desenvolvida, testada e aceita;
- Negociável – não é um "contrato fechado" - é primordial a colaboração entre as partes interessadas e a evolução da história à medida que a implementação se aproxima;
- Valiosa – a história deve entregar valor aos stakeholders;
- Estimável – a história deve fornecer informações suficientes para que o time possa elaborar uma estimativa de alto nível;
- Sob medida (pequena) – uma boa história deve ser relativamente pequena em tamanho, para que seja concluída no menor tempo possível, e para caber em uma iteração;
- Testável – uma história deve estar clara o suficiente para que possam ser definidos testes.

Modelo 3Cs

Além do modelo "Invest", mostrado anteriormente, uma boa história de usuário consiste em três elementos, chamados de 3 Cs: cartão, conversa e confirmação.

Cartão

A descrição feita para a HU deve caber em um cartão índice, com informações breves e suficientes para identifica-la.

Segue abaixo um exemplo de cartão índice:

HU001 - SIOPE MAVS - CADASTRAMENTO BANCO AGÊNCIA NO CADASTRO CORPORATIVO MAVS E CONVENIADAS

REQ6 (REQ000000263340

REQ PAI (REQ000000263254)

COMO...

- Um usuário do SIOPE_MAVS

Quero...

- Incluir opção para cadastramento de banco e agência a serem inseridos no cadastro corporativo do FNDE. Aproveitando-o tanto para o MAVS quanto para o CONVENIADAS.

Para...

- Permitir aos usuários inserir dados bancários (nome do banco e número da agência) no cadastro corporativo do FNDE.

User Story

Siopes Mavs

PHP. React

Conversa

As conversas devem ser contínuas, não somente quando o requisito é definido.

Confirmação

Na etapa de confirmação é determinado se o objetivo da HU foi alcançado. Os critérios de aceitação devem ser utilizados para isso.

6.3 Critérios de Aceitação

O critério de Aceitação ou de Aceite é uma parte essencial da definição de uma História do Usuário em um projeto ágil. Enquanto Histórias de Usuário são subjetivas, os Critérios de Aceite provêm a objetividade necessária para que uma HU seja considerada completa ou não, estabelecendo os requisitos específicos que devem ser cumpridos para que a História do Usuário seja considerada concluída e aceita pelo cliente ou usuário final.

Os critérios de aceitação são escritos na forma de afirmações claras e mensuráveis

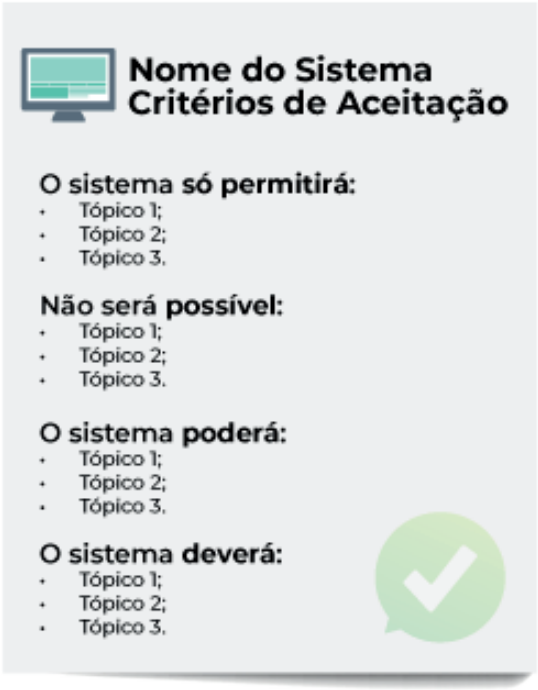
que descrevem as funcionalidades ou comportamentos esperados de cada História do Usuário. Eles refinam os requisitos, ajudam a garantir que todos tenham uma compreensão comum do que será entregue, removem ambiguidades, ajudam a alinhar expectativas e definem uma base para a verificação e validação do trabalho realizado.

Alguns exemplos de critérios de aceitação podem incluir:

Quando um usuário autenticado selecionar a opção 'Cadastrar Escola', ele deve ser redirecionado para um formulário de cadastro com os campos obrigatórios devidamente preenchidos;

Quando um usuário pesquisar por uma escola usando o número do Cadastro Nacional de Pessoa Jurídica (CNPJ), o sistema deve exibir todas as informações relevantes da escola, incluindo endereço, contato e situação cadastral;

O s critérios de aceitação ajudam a orientar o trabalho da equipe de desenvolvimento, fornecendo uma visão clara dos resultados esperados. Eles também são úteis para evitar ambiguidades e garantir a satisfação do cliente ou usuário final ao finalizar uma História do Usuário.



6.4 Épicos

Um Épico é uma unidade de trabalho de grande escala que representa uma iniciativa ou funcionalidade ampla em um projeto. Ele geralmente é dividido em histórias de usuários menores e mais gerenciáveis para facilitar a implementação. Pode-se dizer que Épicos são grandes Histórias de Usuários (ou HUs) não refinadas, constantes do Backlog do Produto.

Exemplo de Épico: “Desenvolvimento de um aplicativo de gerenciamento de tarefas e pedidos”.

Esse Épico envolve a criação de um aplicativo que permite aos usuários organizar, priorizar e acompanhar suas tarefas diárias. Ele abrange várias funcionalidades, como criação de tarefas, definição de prazos, atribuição de responsáveis, acompanhamento de progresso e notificações de lembrete.

Para implementar este épico, a equipe o divide em histórias de usuários menores, por exemplo: “como usuário, quero criar uma tarefa com título e descrição” ou “como usuário, quero poder definir um prazo para cada tarefa” e assim por diante. Cada História de Usuário representa um passo incremental para a conclusão do Épico como um todo. Ao dividir o Épico em HUs menores, a equipe pode priorizá-las e implementá-las em iterações ou ciclos de desenvolvimento. Isso permite que o aplicativo seja entregue em partes, com funcionalidades básicas disponíveis primeiro e incrementos subsequentes adicionando recursos e aprimoramentos adicionais.

Quando Épicos são priorizados no Backlog do Produto para serem implementados na próxima Sprint, eles são segmentados em Histórias de Usuários menores, mais granuladas. Essas Histórias de Usuários devem ser simples, curtas e de fácil implementação.

6.5 Métodos de Priorização de Histórias de Usuários

MoSCoW

Must have (tem que fazer)

Tarefas essenciais que devem ser cumpridas logo para que um projeto ou solução seja um sucesso. Logo, são consideradas uma obrigação para as equipes. Você não pode

simplesmente fechar os olhos e ignorá-las. Se está no escopo de trabalho, faça.

Should have (deve fazer)

São as tarefas importantes, mas não vitais para determinado projeto. Isso porque você pode priorizá-las em segundo momento, após cumprir as obrigatoriedades. Ao serem feitas, tais atividades irão gerar um valor significativo ao projeto.

Could have (pode fazer)

Funcionado como um acessório, tais tarefas dessa seção não são necessárias para o funcionamento do projeto. Mas se houver tempo, após realização das acima pode ser incluída nas demandas do grupo. E se, ainda houver mais itens a serem adicionados como prioritários, as tarefas daqui poderão ser retiradas sem prejudicar o resultado.

Won't have (não vou fazer – por agora)

As atividades dessa categoria são aquelas que você pode fazer ou não, logo não precisa se cobrar ou gerar expectativa sobre a realização. Certo?

Comparação em Pares

A comparação em pares é uma técnica utilizada para priorizar Histórias de Usuário (HUs) em um backlog. Ela envolve a comparação de duas HUs por vez e a escolha daquela que possui maior prioridade ou valor para o projeto.

- A comparação em pares pode ser feita por meio de perguntas como:
- Qual das duas HUs é mais importante para os usuários?
- Qual das duas HUs trará mais benefícios para o negócio?
- Qual das duas HUs é mais viável de ser implementada?

Ao realizarvárias comparações em pares, é possível estabelecer uma ordem de prioridade para todas as HUs do backlog. Essa técnica é útil para tomar decisões objetivas e evitar a sobrecarga de trabalho, garantindo que as HUs mais relevantes sejam implementadas primeiro.

100 pontos

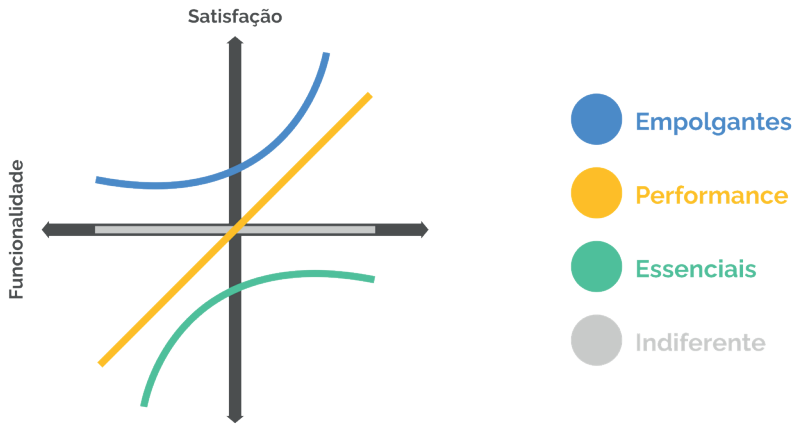
Imagine um grupo de pessoas tentando priorizar cinco itens:

- 100 pontos por pessoa;
- Uma pessoa pode decidir que cada projeto é igualmente importante e atribuir 20 itens a cada projeto;
- Ou considere o projeto 1 mais importante que o projeto 2, o projeto 2 é mais importante que o projeto 3 e assim por diante. Como resultado, seus votos foram atribuídos de forma ponderada, obtendo-se 40 votos no caso 1, 30 votos no caso 2, 15 votos no caso 3, e assim sucessivamente, até que todos os votos sejam alocados.;
- Todos os boletins de voto são então somados para determinar a votação final para cada item, tidos em uma lista de prioridades.

Análise Kano

Features podem ser divididas em quatro categorias, dependendo de como os clientes reagem ao nível de funcionalidade fornecido:

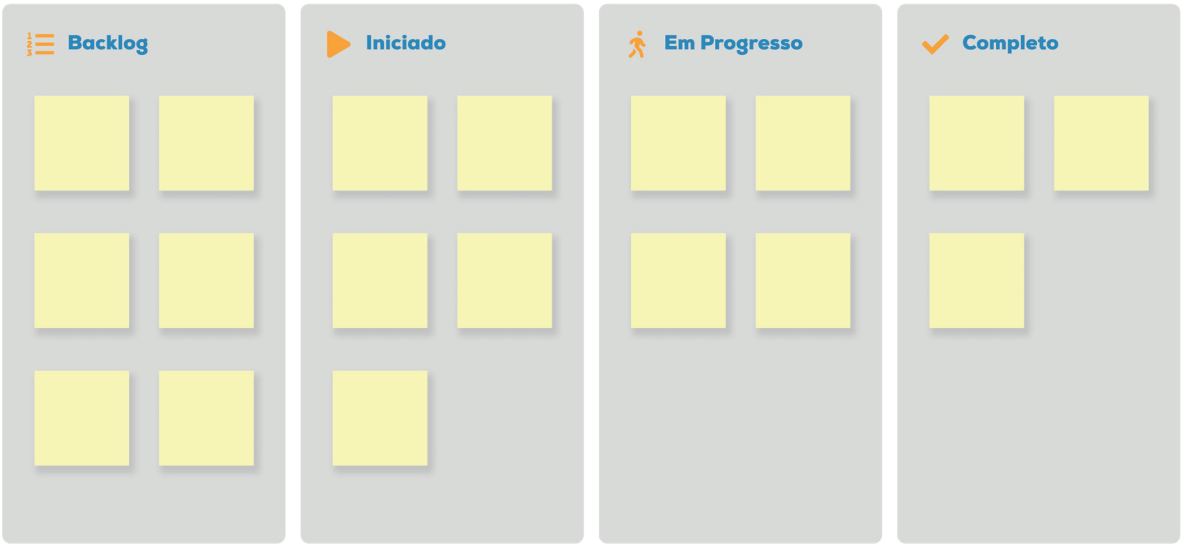
- Features de performance: features do produto se comportam como o que pensamos intuitivamente que a satisfação funciona: quanto mais fornecemos, mais satisfeitos nossos clientes ficam;
- Features essenciais (básicas): São features do produto que simplesmente são esperados pelos clientes. Se o produto não as possuir, será considerado incompleto ou simplesmente ruim;
- Features empolgantes (excitantes): São features inesperados que, quando apresentados, causam uma reação positiva. Estes são geralmente chamados de atraentes, excitantes (Delighters);



Features indiferentes: São features pelas quais nos sentimos indiferentes. Aqueles cuja presença (ou ausência) não faz uma diferença real em nossa reação ao produto.

6.6 Backlog da Sprint

-



É o conjunto de itens do Backlog do Produto selecionados para a Sprint. O Time de Desenvolvimento modifica o Backlog da Sprint ao longo de toda a Sprint, adicionando um novo trabalho sempre que necessário para atingir o objetivo da Sprint e removendo os elementos desnecessários.

6.7 Definição de Pronto

A definição de pronto (DdP, Definition of Done ou DoD) é um critério pré-definido que estabelece as condições para que um item de trabalho seja considerado completo e pronto para ser entregue. É uma lista de requisitos ou critérios que devem ser atendidos para que uma tarefa, história de usuário ou incremento sejam considerados finalizados. É uma ferramenta importante no gerenciamento ágil de projetos, pois garante que o trabalho seja concluído de acordo com os padrões de qualidade estabelecidos pela equipe.Ao realizar várias comparações em pares, é possível estabelecer uma ordem de prioridade para todas as HU's do backlog. Essa técnica é útil para tomar decisões objetivas e evitar a sobrecarga de trabalho, garantindo que as HUs mais relevantes sejam implementadas primeiro.

Ela promove transparência, eficiência e colaboração, ajudando a equipe a entregar um produto de alto nível.

Exemplo de Definição de Pronto (DdP) de Projeto de Desenvolvimento de um Sistema de Gestão Escolar:

1. Todos os requisitos do sistema foram implementados corretamente;
2. Todas as funcionalidades foram testadas e não apresentam erros críticos;
3. As telas e interações do sistema estão alinhadas com o design definido;
4. O código do sistema foi revisado e aprovado pela equipe de desenvolvimento.

07 MÉTRICAS

7. Métricas

7.1 Pontos de História (Story points)

Pontos de História, ou *Story points*, é uma técnica utilizada para planejamento e estimativa. Ela representa uma estimativa de esforço para desenvolvimento de uma atividade. Esta estimativa é feita na reunião de Planejamento da *Sprint* pela Equipe de Desenvolvimento.

A estimativa não é milimetricamente certeira, é uma ideia sobre tempo e esforço necessários para a realização da tarefa. Desta forma, a certeza sobre prazo, tamanho, esforço e complexidade ficarão mais precisos à medida que for feito o refinamento contínuo.

O principal objetivo de usar *story* points é facilitar a estimativa relativa, permitindo que a equipe se concentre no valor que está sendo entregue em vez de prever prazos exatos. Ao longo do tempo, a equipe desenvolve uma velocidade (a média de *story points* concluídos por *sprint*), o que ajuda a prever quanto trabalho pode ser realizado em *sprints* futuros.

Para uma estimativa ser bem-sucedida, é importante seguir os seguintes princípios:

- Consenso da equipe: a estimativa considera a opinião de cada membro do time técnico, mas no final é um acordo do time sobre a complexidade;
- Rapidez e objetividade: estimativas demoradas não são interessantes;
- Participação do Dono do Produto: o Planning Poker deve ser realizado na presença do Dono do Produto

Geralmente, a atribuição de story points leva em consideração três fatores principais :

- Complexidade da tarefa: Quão difícil é a tarefa? Os objetivos estão claros?
- Risco envolvido: Quantas equipes precisarão colaborar? Existem incertezas?
- Esforço necessário: Quantas tarefas individuais estão envolvidas na história e quanto trabalho é necessário para cada uma.

A métrica de Pontos de História utiliza números abstratos que dão a ideia de proporcionalidade entre os requisitos. Para realizar essa estimativa, no FNDE Ágil é utilizada a técnica de Planning Poker.

7.2 Pontos da Sprint

Os pontos da sprint têm como objetivo oferecer uma maneira de estimar o esforço necessário para completar uma determinada atividade, levando em consideração fatores como complexidade, esforço, riscos e incertezas envolvidos.

Os pontos da Sprint serão a soma dos pontos de cada uma das Histórias que a compõem.

7.3 "Planning Poker"

O Planning Poker é uma espécie de jogo, usado para estimar a complexidade do item do backlog ou da História do Usuário. Essas estimativas são feitas com base no domínio e na opinião de especialistas técnicos do time. Todos os membros da equipe contribuem durante o jogo, mas somente aqueles que fazem o trabalho fornecem uma estimativa.

Participantes

Dono do Produto (PO);
Scrum Master;
Time de Desenvolvimento.

Cartas

São utilizadas cartas numeradas a partir da sequência de Fibonacci (1, 2, 3, 5, 8, 13, 20, 40, 100), que indicam o esforço/tempo de desenvolvimento, conforme mostrado a seguir:

Fácil

As tarefas mais fáceis são estimadas com pontuação 1, 2 ou 3. São pequenas alterações, sem riscos e sem dependências.

Pode-se utilizar o seguinte critério para estimar a pontuação:
1 – é a mais fácil (fácil fácil) - ficará pronta em 1 hora
2 – médio fácil - ficará pronta em poucas horas
3 – fácil, mas não muito rápida (fácil "difícil") - ficará pronta ainda hoje, ou amanhã

Médio

As tarefas medianas são estimadas com pontuação 5, 8 ou 13. São desenvolvimentos que requerem uma lógica nova, mas sem grandes incertezas.

Pode-se utilizar o seguinte critério para estimar a pontuação:
1 – é a mais fácil das médias (média fácil) – pode demorar 1 semana
2 – média média – parece não ser difícil, podem aparecer algumas surpresas
3 – a mais difícil dentre as de complexidade média – será necessário lidar com muitos trechos de código diferentes

Difícil

As tarefas difíceis são estimadas com pontuação 20, 40 ou 100. São histórias que envolvem múltiplos módulos, alta incerteza e grande impacto.

Pode-se utilizar o seguinte critério para estimar a pontuação:
1 – é a mais fácil das difíceis – pode ser necessária 1 sprint inteira
2 – média – tarefa grande, complexa, melhor quebrá-la
3 – a mais difícil, que demandaria um longo tempo. Melhor discuti-la e talvez dividi-la em partes menores.

Se a reunião é presencial, são utilizadas cartas físicas.

Se a reunião é remota, é utilizado um aplicativo próprio para o Planning Poker, como por exemplo ‹Planning poker online | Sisfisk sprint2› ou https://planningpokeronline.com/

Planning Poker

Do Backlog à Votação

1 Apresentação e Discussão

O Dono do Produto explica o item e a equipe tira todas as dúvidas.

Estimativa Privada 2

Cada desenvolvedor seleciona sua carta secretamente, baseando-se no esforço necessário.

Revelação Simultânea 3

O Scrum Master solicita que todos mostrem suas cartas ao mesmo tempo.

O Caminho para o Consenso

5 Ciclo de Debate

Havendo divergência, os membros discutem suas razões e repetem a votação.

4 Verificação de Consenso

Se todos os números forem iguais, a estimativa é definida e aceita.

Finalização da Sprint 6

O processo se repete até que todas as histórias da Sprint sejam estimadas.

O Jogo

Passo 1: O Planning Poker começa com a leitura da descrição um item de backlog pelo Dono do Produto, explicando-a para toda a equipe, da forma mais clara possível.

Passo 2: os membros da Equipe de Desenvolvimento discutem o item. O Dono do Produto deve esclarecer todas as dúvidas que possam surgir.

Passo 3: cada membro do Time de Desenvolvimento seleciona uma carta, de maneira privada (sem mostrar aos demais), para representar uma estimativa de duração. Esta escolha é baseada na sua opinião em relação ao período necessário para o desenvolvimento da HU.

Passo 4: realizada a escolha, o Scrum Master solicita que todos mostrem as cartas. Se todos os números forem iguais, então há um consenso e, portanto, uma estimativa.

Passo 5: se as cartas forem diferentes, os jogadores devem expor suas opiniões e ideias, explicando suas razões para a escolha. Após essa discussão, deve-se retornar ao passo 3.

Passo 6: o Time de Desenvolvimento e o Dono do Produto escolhem outro item de backlog (outra HU) para análise

O jogo termina quando forem discutidas todas as HUs que serão desenvolvidas na Sprint.

7.4 Como avaliar a complexidade de uma História de Usuário

As perguntas abaixo podem auxiliar o Time de Desenvolvimento para analisar a complexidade de uma HU

1 – Alguém já fez algo parecido antes?

2 – Qual o número de subtarefas ou etapas necessárias para completar a história?

3 – Afeta várias funcionalidades ou módulos?

5 - Requer mudanças em várias partes do sistema (backend, frontend ou banco de dados)?

6 – Implementa funcionalidade para múltiplos dispositivos (desktop, mobile, tablet)?

7 – Envolve novas tecnologias ou ferramentas para a equipe?

8 – Requer muito esforço de planejamento, codificação e testes?

9 – Trabalha com código legado, mal documentado, ou mal estruturado?

10 – Há um grande volume de campos ou atributos a serem apresentados para os usuários?

11 – Há muitas informações de controle e/ou regras de negócio que influenciam a construção da funcionalidade?

12 – Há muitos dados a serem validados?

13 – Envolve cálculos matemáticos complexos?

14 – É necessária a recuperação ou a consulta de dados de outras aplicações?

15 – Há integração com uma ou mais aplicações?

16 – Há restrições de qualidade, como confiabilidade, eficiência e portabilidade?

17 – Há restrições de interoperabilidade, segurança ou privacidade?

18 – Há restrições de implementação, como por exemplo linguagem de desenvolvimento e prazo de entrega?

19 – Quais são as chances de surgirem imprevistos?

08 ACESSIBILIDADE

8. Acessibilidade

Dentre as diversas abordagens para o termo, podemos defini-la como:

> a flexibilidade necessária para **garantir o acesso à informação e a interação** com sistemas digitais, permitindo que usuários com diferentes necessidades possam utilizá-los de forma plena e eficaz. Melo, A. M.; Baranauskas, M. C. C. (2005). "Design e Avaliação de Tecnologia Web-Acessível".

Portanto, uma interface acessível não pode impor barreiras para interação e para o acesso à informação, nem no hardware e nem no software.

Segundo o Censo Demográfico de 2022, realizado pelo IBGE, cerca de 14,4 milhões de pessoas no Brasil declararam possuir algum tipo de deficiência, o que corresponde a **7,3% da população com dois anos ou mais.** Esses números reforçam a necessidade de cuidado com a obediência às diretrizes e normas de acessibilidade pelo Governo Federal.

No Brasil, vários normativos tratam dessa temática. O principal exemplo é a lei Nº 13.146, de 6 de julho de 2015, que ficou conhecida como **Estatuto da Pessoa com Deficiência**. Ela trouxe a seguinte definição de acessibilidade:

> **a possibilidade e condição de alcance para utilização**, com segurança e autonomia, de espaços, mobiliários, equipamentos urbanos, edificações, transportes, informação e comunicação, **inclusive seus sistemas e tecnologias**, bem como de outros serviços e instalações abertos ao público (...). Neste sentido, a acessibilidade atribui **igual importância a pessoas com e sem limitações** na capacidade de movimento, de percepção, de cognição e de aprendizado.

Esta mesma lei dispõe sobre princípios, regras e instrumentos para o aumento da eficiência da administração pública, e estabelece a acessibilidade como um **princípio e diretriz do Governo Digital e da eficiência pública**, o que a torna **obrigatória para o âmbito de serviços digitais a inclusão de dispositivos móveis** a obediência das boas práticas de acessibilidade nos sítios da internet mantidos por empresas com sede ou representação comercial no País ou por órgãos de governo, para uso da pessoa com deficiência, garantindo-lhe acesso às informações disponíveis, conforme as melhores práticas e diretrizes de acessibilidade adotadas internacionalmente.

Portanto, os sites, portais e aplicações de Governo devem ser projetados de modo que todas as pessoas possam perceber, entender, navegar e interagir de maneira efetiva com as páginas e serviços, conforme estabelecido nas já citadas legislações.

Acessibilidade nos serviços do Governo

Soluções digitais devem promover a eliminação de barreiras na Web, seja no ambiente Desktop ou no Mobile. A ideia pressupõe que os sites, portais e aplicações de Governo sejam projetados de modo que todas as pessoas possam perceber, entender, navegar e interagir de maneira efetiva com as páginas e serviços.

A Portaria Nº 7.508, de 22 de Novembro de 2022 disciplina a implantação e a gestão do Padrão Digital de Governo nos órgãos e entidades do Poder Executivo federal. O normativo orienta que, para garantir a acessibilidade das soluções, devem ser adotados:

- as recomendações do **W3C**; e o
- Modelo de Acessibilidade de Governo Eletrônico **(eMAG)**.

8.1 Modelo de Acessibilidade de Governo Eletrônico (eMAG).

O **Modelo de Acessibilidade em Governo Eletrônico (eMAG)** é um conjunto de recomendações criado pelo governo federal para orientar a construção e adaptação de sites, sistemas e conteúdos digitais do setor público, garantindo que sejam acessíveis ao maior número possível de pessoas.

Ele estabelece diretrizes técnicas e de usabilidade que padronizam a acessibilidade nos portais governamentais, facilitando a eliminação de barreiras e promovendo inclusão digital.

O eMAG está alinhado a padrões internacionais como as **Diretrizes de Acessibilidade para Conteúdo Web (WCAG)** e foi institucionalizado como **obrigatório** para órgãos do governo brasileiro, garantindo conformidade, padronização e atendimento às necessidades de cidadãos com diferentes tipos de deficiência.

Saiba mais

8.2 O W3C e as Diretrizes WCAG: Fundamentos de Acessibilidade Web

O **World Wide Web Consortium (W3C)** é uma comunidade internacional dedicada ao desenvolvimento de padrões que asseguram o funcionamento e a evolução da Web. Reconhecendo a necessidade de promover inclusão e eliminar barreiras digitais, criou a Web Accessibility Initiative (WAI), responsável por desenvolver diretrizes e recursos voltados à acessibilidade. Entre suas principais contribuições estão as **Diretrizes de Acessibilidade para Conteúdo Web (WCAG)**, um conjunto estruturado de recomendações destinadas a tornar o conteúdo da Web efetivamente acessível para todas as pessoas, incluindo pessoas com deficiência.

O WCAG 2.1 possui **4 princípios**, cada um deles com suas respectivas diretrizes. Ao todo são **13 diretrizes** que estão inseridas dentro dos princípios. Elas fornecem os objetivos básicos que devem ser observados para tornar seu conteúdo acessível às diferentes formas de deficiência.

Seguir estas diretrizes irá tornar o conteúdo acessível a um maior número de pessoas com deficiência, incluindo acomodações para cegueira e baixa visão, surdez e baixa audição, limitações de movimentos, incapacidade de fala, fotossensibilidade e combinações destas características, e alguma acomodação para dificuldades de aprendizagem e limitações cognitivas, mas não abordará todas as necessidades de usuários com essas deficiências.

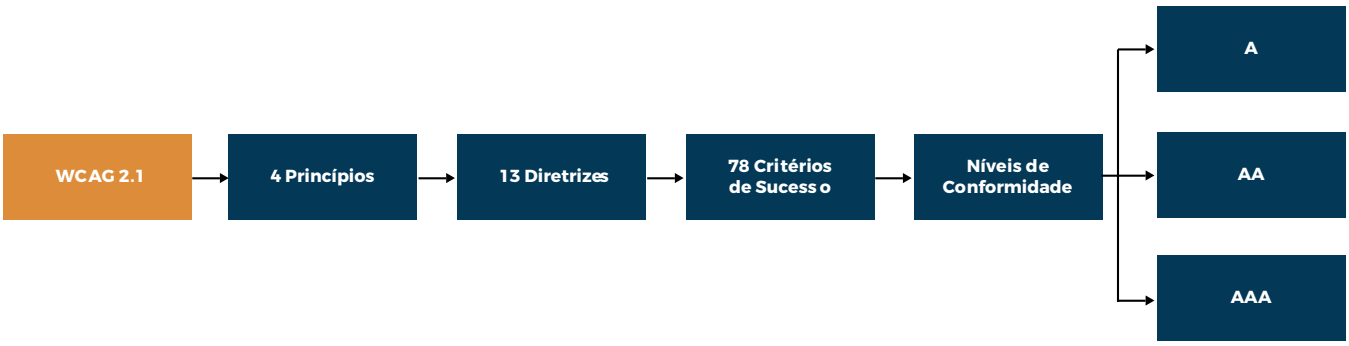


Figura 1 - descrição gráfica do WCAG 2.1

Veja a descrição simplificada dos Princípios de Acessibilidade – WCAG 2.1

Princípio 1. Perceptível

As informações e os componentes da interface do usuário devem ser apresentados em formas que possam ser percebidas pelo usuário.

- **Diretriz 1.1 – Alternativas em Texto:** Fornecer alternativas textuais para qualquer conteúdo não textual, permitindo sua transformação em outras formas (fonte ampliada, braille, fala, símbolos ou linguagem simples).
- **Diretriz 1.2 – Mídias com base em tempo:** Fornecer alternativas para mídias baseadas em tempo.
- **Diretriz 1.3 – Adaptável:** Criar conteúdo que possa ser apresentado de diferentes maneiras (ex.: layout simplificado), sem perder informação ou estrutura.
- **Diretriz 1.4 – Discernível:** Facilitar a audição e visualização do conteúdo, incluindo a separação entre primeiro plano e plano de fundo.

Princípio 2. Operável

Os componentes da interface de usuário e a navegação devem ser operáveis.

- **Diretriz 2.1 – Acessível por Teclado:** Garantir que toda funcionalidade esteja disponível a partir de um teclado.
- **Diretriz 2.2 – Tempo Suficiente:** Fornecer aos usuários tempo suficiente para ler e utilizar o conteúdo.
- **Diretriz 2.3 – Convulsões e Reações Físicas:** Evitar conteúdo que possa causar convulsões ou reações físicas.
- **Diretriz 2.4 – Navegável:** Oferecer maneiras de ajudar os usuários a navegar, localizar conteúdos e identificar sua posição.
- **Diretriz 2.5 – Modalidades de Entrada:** Facilitar a operação por meio de várias entradas além do teclado.

Princípio 3. Compreensível

A informação e a operação da interface de usuário devem ser compreensíveis.

- **Diretriz 3.1 – Legível:** Tornar o conteúdo textual legível e compreensível.
- **Diretriz 3.2 – Previsível:** As páginas devem aparecer e funcionar de modo previsível.

- **Diretriz 3.3 – Assistência de Entrada:** Ajudar os usuários a evitar e corrigir erros.

Princípio 4. Robusto

O conteúdo deve ser robusto o suficiente para ser interpretado de forma confiável por uma ampla variedade de agentes de usuário, incluindo tecnologias assistivas.

- **Diretriz 4.1 – Compatível:** Maximizar a compatibilidade entre agentes de usuário atuais e futuros, incluindo tecnologias assistivas.

Níveis de Conformidade – WCAG 2.1

As diretrizes são atendidas por meio de critérios de sucesso testáveis, organizados em três níveis de conformidade:

Nível A:

- Critérios essenciais que eliminam barreiras significativas de acessibilidade.
- Se não atendidos, tecnologias assistivas podem não conseguir ler, entender ou operar páginas.

Nível AA (Duplo A):

- Inclui todos os critérios do nível A e novos critérios adicionais.
- Garante acessibilidade para a maioria dos usuários em diversas circunstâncias.

Nível AAA (Triplo A):

- Inclui todos os critérios dos níveis anteriores e critérios mais detalhados.
- Representa um nível mais sofisticado de acessibilidade.
- Pode ser obrigatório em determinados contextos para garantir acessibilidade ao público-alvo.

Acesse os critérios de sucesso aqui:

https://www.gov.br/governodigital/pt-br/acessibilidade-e-usuario/acessibilidade-digital/AnexoAMapeamentoeMAGxWCAG.ods

8.3 Acessibilidade e o Padrão Digital do Governo Federal

O Padrão Digital de Governo é o *Design System* do Governo Federal do Brasil.

Ele se propõe a ser um documento vivo com diversas diretrizes e componentes que servem como referência para garantir a eficiência na produção de produtos digitais e a garantir a padronização da experiência única para todos os usuários.

O Padrão Digital de Governo está alinhado com os princípios de acessibilidade e usabilidade?

Acessibilidade é parte integrante da inclusão digital e é frequentemente regulamentada por leis e diretrizes, como as Web Content Accessibility Guidelines (WCAG) e o Modelo de Acessibilidade em Governo Eletrônico (e-MAG). Não beneficia apenas as pessoas com deficiências, mas também melhora a experiência do usuário para todos.

Além disso, existem outras referências na página do Padrão Digital de Governo e na wiki onde o assunto acessibilidade é tratado e detalhado:

- https://www.gov.br/ds/acessibilidade
- https://govbr-ds.gitlab.io/tools/govbr-ds-wiki/design/guias/acessibilidade/

Princípios do Design System

- https://www.gov.br/ds/introducao/principios

Acessibilidade no Código HTML(GUIA)

- https://www.gov.br/ds/guias/acessibilidade-html

Uso do WAI-ARIA

- https://www.gov.br/ds/guias/wai-aria

Acessibilidade na WIKI GOV.BR-DS –Guias e Boas Práticas
https://gov.br/ds/wiki/acessibilidade/

8.4 Critérios de aceitação segundo a FNDE Ágil

A equipe Scrum deverá assegurar que todas as soluções digitais desenvolvidas atendam, no mínimo, ao **nível AA das diretrizes WCAG 2.1**, conforme estabelecido pelo **eMAG** e

pelas recomendações da **Lei Brasileira de Inclusão (LBI, Art. 63).**

Software avaliadores

Os softwares avaliadores de acessibilidade realizam uma análise automática do código da página web, procurando detectar a existência de erros e omissões que possam se constituir em barreiras para a acessibilidade. De acordo com o resultado dessa análise, quando uma página não apresenta erros em sua validação, uma classificação é feita, conforme os três níveis de acessibilidade estabelecidos pelo WCAG 2.1.

A WAI disponibiliza em seu site uma lista com diversas ferramentas para validação de acessibilidade. Apesar de se constituírem em um grande auxílio para os desenvolvedores, esses avaliadores automáticos não são suficientes para garantir a acessibilidade de uma página web, em nenhum nível. Por isso, são necessários outros softwares e, principalmente, a avaliação humana por usuários e especialistas.

Validadores de código

- Validador (X)HTML
- Validador (X)HTML
- Validador CSS
- Validador de links

Validadores automáticos de acessibilidade

- AMAWeb - Avaliação e Monitoramento de Acessibilidade na Web
- Cynthia Says (WCAG 1.0)
- eExaminator (WCAG 1.0)
- Functional Accessibility Evaluator 1.1
- Wave (WCAG 1.0 e Section 508)

Checklist para validação Humana

- Checklist Manual de Acessibilidade - Desenvolvedores
- Checklist Manual de Acessibilidade - Deficientes Visuais

Extensões para navegadores (para avaliação de acessibilidade)

Extensões para navegadores (para avaliação de acessibilidade)

Chrome:

- eScanner
- Web Developer

Firefox:

- WAVE Web Accessibility Evaluation Toolbar
- WCAG Contrast checker
- Web Developer

Internet Explorer:

- Web Accessibility Toolbar para IE

Ferramentas para análise de relação de contraste

- Luminosity Colour Contrast Ratio Analyser (online)
- Color Contrast Analyser (online)
- Check my colours (online)
- Contrast Analyser 2.2

Leitores de Tela

- Jaws for Windows - Leitor de tela americano produzido pela Freedom Scientific
- NVDA - Leitor de tela gratuito e de código aberto para Windows
- Orca - Leitor de tela gratuito e de código aberto para Linux
- VoiceOver - Leitor de tela para MAC OS
- Leitor de tela canadense fabricado pela GW Micro
- DOSVOX: Interface especializada desenvolvida pela UFRJ
- Descrição dos Leitores de Tela (documento pdf - 2.71 MB)
- Avaliação dos Pontos de Fragilidades em Leitores de Tela (documento pdf - 1.47 MB)

Navegadores Textuais

- Lynx
- Lynx Viewer (simulador)

Simulador para Cegueira Cromática (Daltonismo)

- Vischeck

Tradutores automáticos para Libras

- VLibras
- Handtalk
- Rybená
- Prodeaf

Acesse mais recursos de acessibilidade no portal Governo Digital:

https://www.gov.br/governodigital/pt-br/acessibilidade-e-usuario/acessibilidade-digital/recursos-de-acessibilidade

8.5 Componentes e Acessibilidade no Design System Gov.br

O Design System GOV.BR disponibiliza para grande parte dos componentes, além das abas de "Desenvolvedor" e "Design", uma nova aba de "Acessibilidade" com informações, orientações e recomendações sobre acessibilidade aplicada ao cada componente específico.

Exemplo para o componente "Button"

Criação e Desenvolvimento de novo componentes

O checklist abaixo deve ser utilizado para construção de qualquer elemento ou interfaces (componente, padrões, fluxos, etc.) conforme o DS.GOV.

O documento é voltado para designers, portanto, podem existir pontos importantes não explorados por não fazer sentido para este tipo de usuário (como por exemplo, codificação semântica do HTML, etc.)

O detalhamento de cada item no checklist pode ser encontrado nos documentos de Recomendações de Acessibilidade.

Acessibilidade

O DS Gov.br já vem:

- WCAG 2.1
- Navegação por teclado
- ARIA configurado

Tema	UX Designer	Desenvolvedor	Trabalho Conjunto
Foco Central	Experiência do usuário	Implementação técnica	Produto útil e funcional
Protótipos	Cria	Consulta/valida	Ajustes para viabilidade
Acessibilidade	Orienta e desenha	Implementa	Conformidade eMAG/WCAG
Teste	Usabilidade, fluxo	Unitários, integração	Validação final

Esse trabalho colaborativo assegura que cada incremento entregue seja tecnicamente consistente, acessível e centrado na experiência do usuário.

Ainda o TR prevê indicadores como:

- **IAS – Indicador de Aceitação de Sprints,**
- **IQC – Indicador de Qualidade de Código,**
- **IPP – Indicador Individual dos Perfis,**
- **ISP – Satisfação do Dono do Produto.**

Todos esses indicadores **dependem diretamente do trabalho conjunto**, pois:

- se o design não estiver claro → o código será inadequado;
- se o código não refletir o design → a entrega será rejeitada;
- se a comunicação for falha → a sprint não será aceita.

Para saber mais sobre os fluxos de criação de desenvolvimento acesse:

- https://www.gov.br/ds/como-comecar/fluxo-criacao
- http://www.gov.br/ds/como-comecar/fluxo-dev

09 UX DESIGN: DESIGN DE EXPERIÊNCIA DO USUÁRIO

9. UX Design – Design de Experiência do Usuário

O UX Design é parte fundamental da metodologia FNDE Ágil. Enquanto o Design Thinking ajuda a entender as necessidades reais dos usuários, o Scrum organiza a entrega de soluções de forma ágil e incremental, ele c**onecta essas etapas**, garantindo que a **experiência do usuário esteja no centro do processo** - do entendimento do problema à entrega final.

De maneira geral, entre várias atividades desempenhadas pelo UX Designer podemos citar de maneira bastante simplificada:

Pesquisa e análise

Realizar entrevistas, pesquisas e testes com usuários para identificar necessidades, comportamentos e problemas.
Conduzir análises de usabilidade e revisar feedbacks para aprimorar a experiência.
Mapear jornadas do usuário (user journey) e construir personas representativas.

Design da Experiência do Usuário (UX)

- Criar fluxos de navegação (user flows) claros e eficientes.
- Definir e documentar a arquitetura da informação de sistemas e produtos digitais.
- Propor soluções para funcionalidades complexas, garantindo melhor experiência.

Desenvolvimento de Interface do Usuário (UI)

- Criar wireframes, protótipos de baixa e alta fidelidade e mockups.
- Desenvolver interfaces visuais atraentes, alinhadas à identidade do produto.
- Garantir consistência visual e funcional com guias de estilo e bibliotecas de componentes.

Validação e testes

- Planejar e executar testes de usabilidade com usuários reais ou simulados.
- Coletar métricas e feedbacks para validar soluções e realizar ajustes.

- Conduzir testes A/B para comparar diferentes abordagens de design.

Colaboração e alinhamento com equipe Scrum e stakeholders

- Trabalhar em conjunto com desenvolvedores, Product Owners, Scrum Masters, stakeholders e, principalmente, atuar como parceiro dos Analistas de Requisitos em suas atividades.
- Garantir viabilidade técnica das soluções e alinhamento com objetivos do negócio.
- Participar de reuniões de planejamento e refinamento para entender requisitos e limitações.

Padrões e acessibilidade

- Estabelecer e manter guias de estilo, padrões de design e bibliotecas reutilizáveis.
- Incorporar tendências de design e boas práticas de acessibilidade (ex.: WCAG).
- Garantir designs responsivos e adaptáveis a diferentes dispositivos e plataformas.

Liderança e evolução contínua

- Liderar iniciativas de design em projetos complexos ou estratégicos.
- Atuar como referência técnica e ponto de contato para decisões de UX/UI.
- Acompanhar métricas de engajamento e desempenho das interfaces.
- Identificar oportunidades de melhoria contínua e propor evoluções no design.
- Assegurar que soluções acompanhem mudanças nas necessidades do usuário e do mercado.

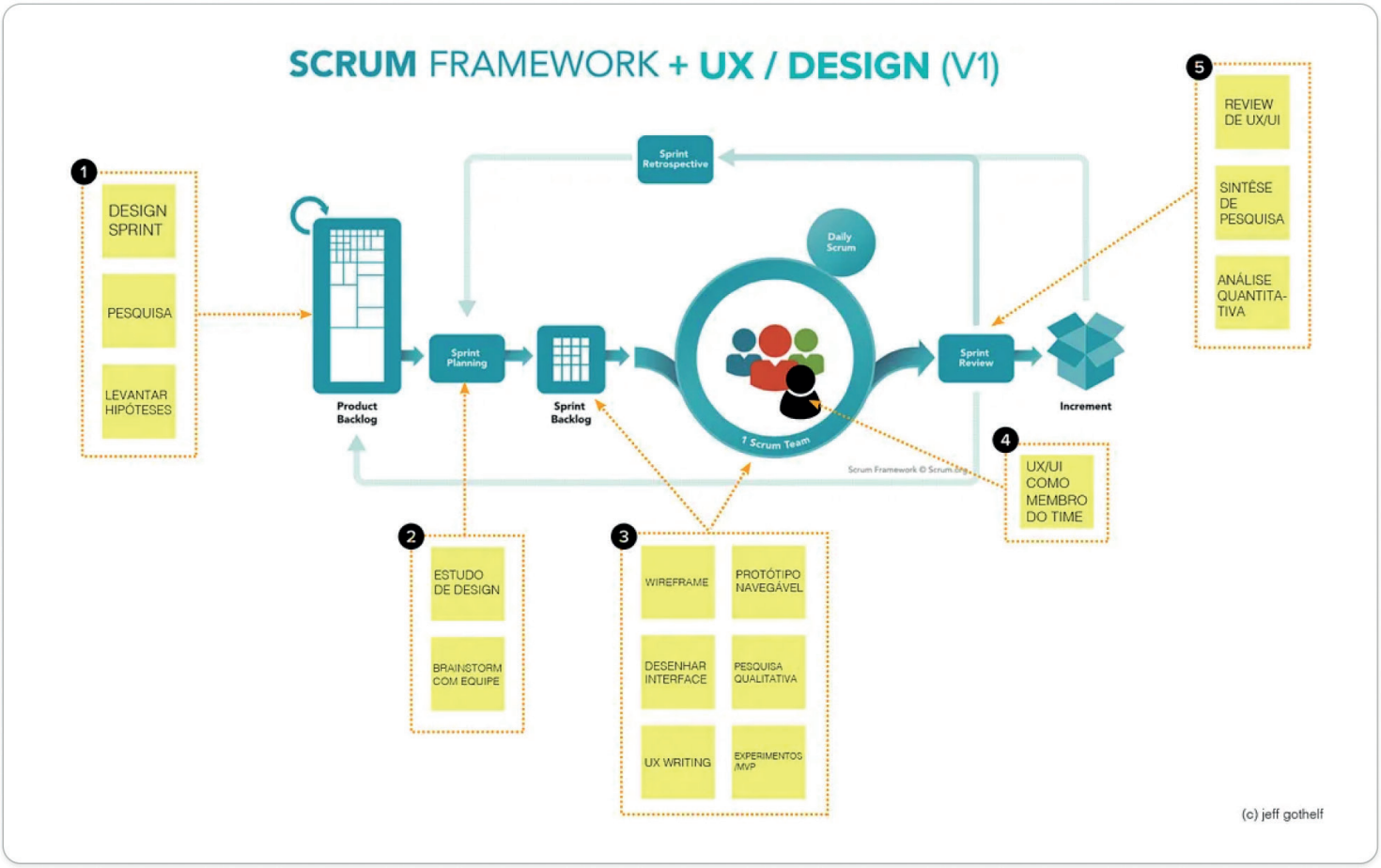
UX Writing

- Produzir textos claros, concisos e úteis para botões, menus, formulários, mensagens de erro e notificações.
- Criar microcopy para guiar usuários em manuais, tutoriais e passo a passo.
- Seguir guia de escrita com estilos e diretrizes para manter consistência na comunicação.

9.1 UX Design na Metodologia FNDE Ágil

Na metodologia FNDE Ágil, o UX Designer **não atua somente na concepção de telas e protótipos** para soluções. Ele atua de forma colaborativa com os principais papéis da equipe para garantir que o produto seja funcional e centrado no usuário.

9.1 Integração com o Time Scrum



O UX Designer atua no time de desenvolvimento, colaborando desde a concepção do produto até sua entrega e validação. Tem atuação fundamental na ideação das soluções, levantamento de requisitos, construção de personas e Mapas de Jornadas do Usuário.

Deve participar das cerimônias do Scrum: Daily, Review e Retrospectiva, trazendo o olhar do usuário para o processo.

Sua interação é mais intensa com o Product Owner (PO), com os Desenvolvedores, com o Analista de Requisitos. Conta com o suporte do Scrum Master, que facilita a comunicação e remove impedimentos que possam afetar o processo de design. Dessa forma, o UX

Designer contribui para que o fluxo de desenvolvimento seja integrado.

UX Designer e o PO

Ele ajuda o PO na elaboração e refinamento do Backlog do Produto, contribuindo com Histórias de Usuário, critérios de aceitação e priorização com base em validações com usuários e testes de usabilidade.

UX Designer e o Analista de Requisitos

A integração entre o papel do UX Designer e do Analista de Requisitos é fundamental para assegurar que o produto de software atenda simultaneamente às necessidades de negócio e às expectativas dos usuários finais, conciliando viabilidade técnica, clareza funcional e excelência na experiência de uso.

Responsabilidades	
UX Designer	**Analista de Requisitos**
• Traduzir requisitos em soluções visuais e interativas centradas no usuário. • Conduzir pesquisas, testes de usabilidade e validações de interface. • Assegurar que o produto seja intuitivo, acessível e eficiente em sua utilização	• Realizar o levantamento, análise e documentação das necessidades de negócio. • Definir requisitos funcionais e não funcionais, bem como critérios de aceitação. • Atuar como elo entre stakeholders e equipe técnica, garantindo alinhamento estratégico.

Formas de Integração	
Levantamento de Necessidades	• Participação conjunta em entrevistas, workshops e sessões de descoberta; • O analista foca em regras de negócio; o UX Designer identifica expectativas e comportamentos dos usuários.
Definição de Histórias de Usuário	• O analista estrutura histórias com critérios objetivos; • O UX Designer contribui para que as histórias reflitam jornadas reais de interação.
Prototipação e Validação	• O UX Designer desenvolve protótipos e wireframes; • O analista verifica conformidade com requisitos e restrições técnicas.
Revisões e Entregas	• Ambos participam das revisões de sprint, avaliando se o incremento atende às regras de negócio e à experiência planejada.

Dentre as várias ferramentas existentes para levantamento de requisitos, e que também podem ser utilizadas para o mapeamento de Histórias de Usuários, a metodologia FNDE Ágil **propõe fortemente** a utilização contínua de prototipação como ferramenta de apoio. Ela é usada para materializar rapidamente ideias, facilitar a comunicação entre equipe técnica e usuários e coletar feedback.

A colaboração entre as especialidades favorece o alinhamento entre requisitos técnicos e experiência do usuário, assegurando coerência entre demandas de negócio e usabilidade. Essa integração reduz riscos de inconsistência entre expectativas dos stakeholders e a aplicação prática do produto, além de aumentar a eficiência do processo de desenvolvimento, resultando em entregas mais assertivas e de maior valor. Consolida-se um fluxo integrado que conecta de forma clara o que o sistema deve realizar com a maneira como o usuário interage com ele.

UX Designer e os Desenvolvedores

UX Designer e Desenvolvedores trabalham de forma integrada para garantir que as soluções planejadas sejam viáveis e centradas no usuário. O UX Designer define fluxos, protótipos e interações que orientam a experiência desejada, enquanto os Desenvolvedores avaliam a viabilidade técnica e implementam essas propostas no código.

Integrando UX e Desenvolvimento

Na metodologia FNDE Ágil, o trabalho de ambos os profissionais ocorre de forma integrada e contínua ao longo de todo o ciclo de desenvolvimento. Essa colaboração se materializa desde o refinamento do backlog, quando os protótipos e critérios de aceitação apresentados pelo UX são avaliados pelos desenvolvedores quanto à viabilidade técnica.

Durante a sprint, UX acompanha a implementação, esclarece interações, revisa componentes e valida aderência ao design, enquanto os desenvolvedores ajustam comportamentos, implementam funcionalidades e garantem conformidade com requisitos de qualidade, segurança e acessibilidade.

Ao final, ambos participam da Review, apresentando conjuntamente as entregas, e da

Retrospectiva, alinhando melhorias de processo e comunicação para os ciclos seguintes. apresentados pelo UX são avaliados pelos desenvolvedores quanto à viabilidade técnica.

Esse trabalho colaborativo assegura que cada incremento entregue seja tecnicamente consistente, acessível e centrado na experiência do usuário.

Resumo Comparativo
Ainda o TR prevê indicadores como:

- **IAS – Indicador de Aceitação de Sprints,**
- **IQC – Indicador de Qualidade de Código,**
- **IPP – Indicador Individual dos Perfis,**
- **ISP – Satisfação do Dono do Produto.**

Todos esses indicadores **dependem diretamente do trabalho conjunto**, pois:

Tema	UX Designer	Desenvolvedor	Trabalho Conjunto
Foco Central	Experiência do usuário	Implementação técnica	Produto útil e funcional
Protótipos	Cria	Consulta/valida	Ajustes para viabilidade
Acessibilidade	Orienta e desenha	Implementa	Conformidade eMAG/WCAG
Teste	Usabilidade, fluxo	Unitários, integração	Validação final

- se o design não estiver claro → o código será inadequado;
- se o código não refletir o design → a entrega será rejeitada;
- se a comunicação for falha → a sprint não será aceita.

Para saber mais sobre os fluxos de criação de desenvolvimento acesse:

- https://www.gov.br/ds/como-comecar/fluxo-criacao
- http://www.gov.br/ds/como-comecar/fluxo-dev

9.2 Prototipação e desenvolvimento de interfaces

Como citado, uma das tarefas desenvolvidas pelo UX Designer é a criação de protótipos navegáveis e interfaces para validar hipóteses antes da codificação. Estes objetos são

criados com base nas histórias de usuário priorizadas, seguindo o Padrão Digital do Governo Federal.

Dessa forma, o produto deve não apenas manter a qualidade aprovada pelo PO, mas também atender às proposições fundamentadas pelo Padrão Digital do Governo Federal - fundamentos visuais, ilustrações, fonts, textos, usabilidade e acessibilidade.

A usabilidade é um dos pilares desse processo, garantindo que o usuário consiga interagir com o sistema de forma clara, eficiente e sem esforço desnecessário. Ela orienta decisões como hierarquia visual, fluxo de navegação, clareza dos elementos da interface e previsibilidade das interações, assegurando que cada componente da solução cumpra seu papel de maneira intuitiva. Ao aplicar princípios de usabilidade desde a prototipação, as squads reduzem retrabalhos, fortalecem a consistência entre as etapas de design e desenvolvimento e elevam a qualidade das entregas. Dessa forma, assegura-se que o produto final seja não apenas funcional, mas também fácil de usar, coerente com as diretrizes do Design System Gov.br e acessível a usuários com diferentes níveis de familiaridade tecnológica.

A usabilidade é um dos pilares desse processo, garantindo que o usuário consiga interagir com o sistema de forma clara, eficiente e sem esforço desnecessário. Ela orienta decisões como hierarquia visual, fluxo de navegação, clareza dos elementos da interface e previsibilidade das interações, assegurando que cada componente da solução cumpra seu papel de maneira intuitiva. Ao aplicar princípios de usabilidade desde a prototipação, as squads reduzem retrabalhos, fortalecem a consistência entre as etapas de design e desenvolvimento e elevam a qualidade das entregas. Dessa forma, assegura-se que o produto final seja não apenas funcional, mas também fácil de usar, coerente com as diretrizes do Design System Gov.br e acessível a usuários com diferentes níveis de familiaridade tecnológica.

9.3 Validação

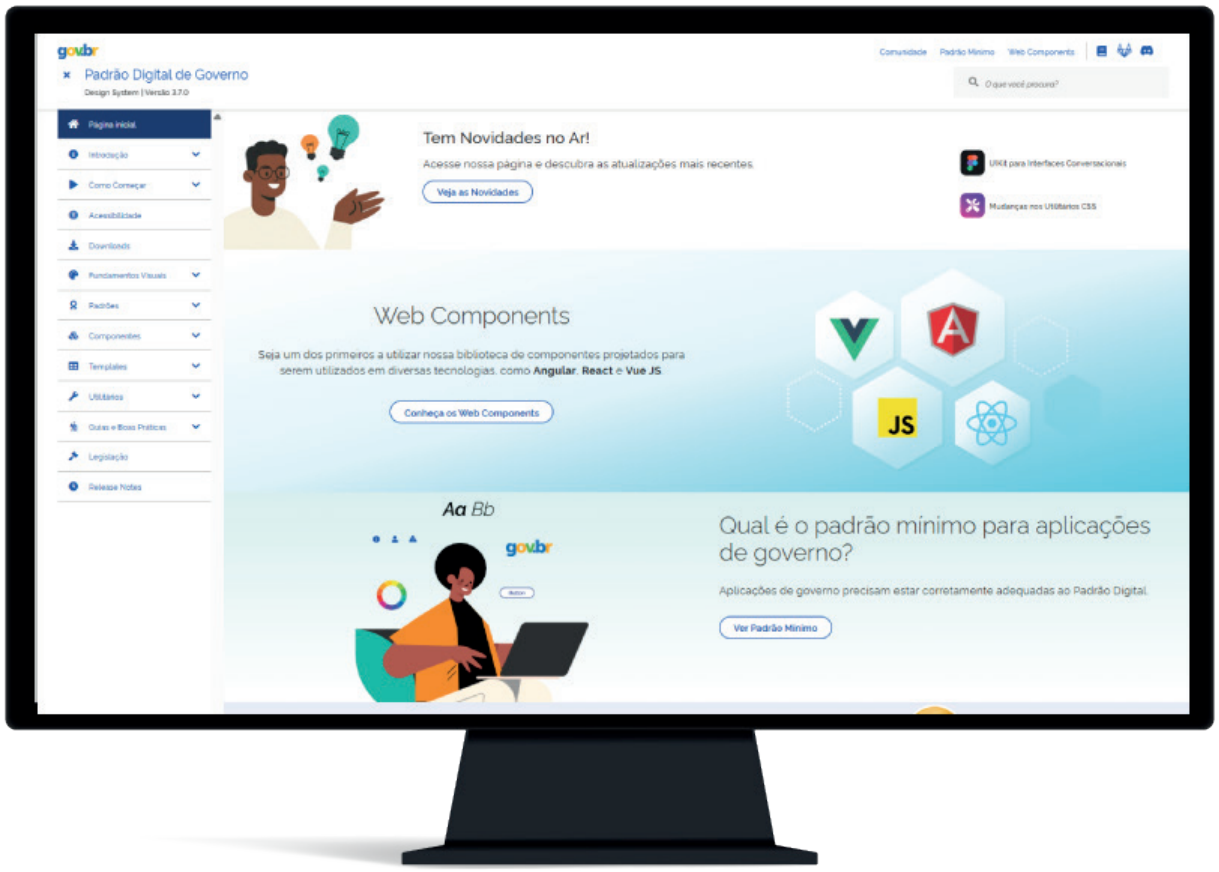
A validação de protótipo faz parte da lista de requisitos na Definição de Pronto da metodologia, tal como a revisão ou aprovação de código e a ausência de erros críticos. O que for implementado deve seguir as orientações do que foi prototipado.

A realização de testes de usabilidade rápidos, quando possível com usuários reais, dentro do tempo da sprint. Antes da entrega, verifica-se a conformidade com a Definição de

Pronto (Definition of Done - DoD), que inclui:

- Validação do protótipo implementado;
- Ausência de erros críticos;
- Conformidade com o design system e critérios de acessibilidade.

9.4 O Padrão Digital de Governo: Design System Gov.br



O Design System Gov.br é um conjunto de padrões, componentes e diretrizes criado para unificar a experiência digital dos cidadãos nos serviços públicos federais, garantindo consistência visual, acessibilidade e eficiência em todas as interfaces.

Ele apresenta os fundamentos visuais, os padrões de design, as boas práticas e as diretrizes de acessibilidade que orientam o trabalho das equipes de desenvolvimento. Além disso, oferece bibliotecas de componentes reutilizáveis, modelos de tela e os artefatos de design, isto é, estruturas pré-definidas que podem ser reutilizadas para criar páginas web consistentes e estruturadas.

Lembre-se! O DS Gov.Br será o guia essencial na “estante” de todo UX Designer. Acesse

em: https://www.gov.br/ds/

Artefatos: https://www.gov.br/ds/downloads/assets

As ferramentas Figma e Adobe XD são indicadas no FNDE como referência para as atividades, pois promovem colaboração e a visualização antecipada da solução pelos integrantes das squads.

9.5 Processos e Fases de UX Design

O processo de UX Design tradicional é dividido em diferentes fases, a depender das diferentes abordagens. Contudo, para melhor adaptação à metodologia FNDE Ágil, vamos resumir em quatro fases principais, cada uma com ferramentas específicas para compreender os usuários, gerar soluções e testar resultados:

Dentro de uma squad ágil, essas fases não acontecem de forma linear ou isolada: elas são distribuídas em ciclos curtos (sprints), podendo se sobrepor e se repetir conforme surgem novas necessidades.

O processo de UX torna-se dinâmico e iterativo, garantindo que cada sprint seja uma oportunidade de aprender com os usuários, ajustar soluções e evoluir a experiência de forma contínua e colaborativa.

Portanto, a regra essencial é: bom senso e adaptabilidade.

Vamos analisar brevemente cada uma das fases:

1 – Descoberta (Entendimento) e Pesquisa

Objetivo: compreender o problema e as necessidades dos usuários.

- Ferramentas:
- Entrevistas com Product Owners (POs) e usuários
- Pesquisas quantitativas (surveys)
- Desk research (análise de dados existentes)
- Mapa de stakeholders
- Shadowing (observação em contexto real)

2. Análise e Síntese

Objetivo: transformar dados em insights e definir necessidades.

Ferramentas:

- Personas (representações de perfis de usuários)
- Mapa de empatia (sentimentos, pensamentos e comportamentos)
- Jornada do usuário (user journey)
- Diagrama de afinidades (agrupamento de insights

Exemplo de mapa de empatia:

3. Ideação
Objetivo: gerar soluções criativas e variadas.

Ferramentas:

- Brainstorming
- Crazy 8's (esboço rápido em 8 quadros)
- Matriz CSD (certezas, suposições e dúvidas) Dentro de uma squad ágil, essas fases não acontecem de forma linear ou isolada: elas são distribuídas em ciclos curtos (sprints), podendo se sobrepor e se repetir conforme surgem novas necessidades.

4. Validação e Testes
Objetivo: verificar eficácia e usabilidade das soluções.

Ferramentas:

- Prototipagem (baixa e alta fidelidade)
- Testes de usabilidade com usuários reais
- Teste A/B (comparação de versões)
- Card Sorting (organização de conteúdo)
- Avaliação heurística (Heurísticas de Nielsen)

UX Design e os Fundamentos Visuais

Dominar os Fundamentos Visuais é passo importante garantir uma linguagem visual unificada. Incluem elementos essenciais padronizados, como cores, tipografias, espaçamentos, ícones, entre outros, para garantir consistência e clareza.

Acesse em: https://www.gov.br/ds/fundamentos-visuais/visao-geral

UX Design e os Padrões de Design

Padrões de design oferecem soluções comprovadas para problemas comuns em projetos de desenvolvimento web. São fundamentais para garantir que um produto digital seja facilmente utilizável pelos usuários. O Padrão Digital de Governo aborda alguns desses aspectos como: ajuda e comunicação, boas práticas para a criação de formulários, padrões de navegação, uso de gráficos, densidade, entre outros.

Acesse em: https://www.gov.br/ds/padroes/visao-geral

Checklist de Qualidade em UX

Seguir os princípios do Design System

- **Considerar as leis de UX:**
  - Lei de Hick (minimizar escolhas)
  - Lei de Fitts (alvos de toque acessíveis)
  - Efeito Von Restorff (destaque visual)
  - Efeito Zeigarnik (clareza de progresso)
  - Efeito Serial Position (ordem de memorização)
  - Lei da Proximidade e Lei da Região Comum (organização visual)

- Garantir acessibilidade digital: contraste adequado, navegação por teclado, compatibilidade com leitores de tela, legendas em vídeos.
- Validar fundamentos visuais e padrões do Gov.br DS
- Envolver stakeholders em grandes impactos (SGD e SERPRO)
- Realizar análise de viabilidade com desenvolvedores

UX Design e Acessibilidade

No ciclo de vida do desenvolvimento de soluções, a acessibilidade deve ser considerada desde as etapas iniciais dos projetos, orientando decisões de concepção, design, desenvolvimento e avaliação. Essa abordagem contribui para:

- redução de barreiras de uso;
- maior eficiência e clareza dos sistemas;
- conformidade normativa;
- melhoria da experiência do cidadão.

Como dito, trabalhar, desenvolver e aplicar a acessibilidade tem a ver com um número maior de pessoas conseguindo fazer uso dos serviços digitais.

O profissional de UX desempenha papel estratégico nesse processo. Entre suas principais responsabilidades estão:

- incorporar acessibilidade desde a concepção das soluções (accessibility by design);
- transformar requisitos legais e técnicos em decisões práticas de interface, conteúdo e interação;
- identificar e antecipar barreiras de uso por meio de pesquisas, testes e avaliações;
- apoiar equipes multidisciplinares na adoção da acessibilidade como critério de qualidade.

Além das diretrizes citadas no capítulo Acessibilidade desta metodologia, o site Governo Digital possui grande acervo de informações sobre o tema. Acesse em: www.gov.br/governodigital/pt-br/acessibilidade-e-usuario/acessibilidade-digital

O Padrão Digital de Governo, como uma solução pública que é referência para o desenvolvimento de soluções digitais, fornecem os recursos construtivos que possibilitam a existência dessa acessibilidade digital nos produtos e serviços nele baseados.

Acessibilidade x Usabilidade

Embora caminhem juntas, acessibilidade e usabilidade têm papéis distintos na construção de interfaces. A usabilidade busca garantir que qualquer pessoa consiga utilizar o sistema de forma simples, eficiente e intuitiva. Já a acessibilidade amplia esse alcance, assegurando que pessoas com diferentes tipos de deficiência também possam perceber, compreender, navegar e interagir com o produto digital sem barreiras. Enquanto a usabilidade otimiza a experiência geral, a acessibilidade garante inclusão e conformidade legal, especialmente alinhada ao eMAG e às diretrizes WCAG.

Em conjunto, ambas fortalecem a entrega de soluções mais completas, equitativas e alinhadas às obrigações do Governo Federal no atendimento ao cidadão.

Checklist de Acessibilidade e UX

Interface e Design Visual

- **Contraste de cores:** Garantir legibilidade entre texto e fundo.
- **Hierarquia:** Organizar elementos por importância visual e lógica.
- **Espaçamento e Legibilidade:** Fontes adequadas e respiro entre elementos.
- **Layout Responsivo:** Adaptação perfeita em diferentes tamanhos de tela.

Interação e Navegação

- **Navegação:** Fluxo intuitivo e caminhos claros para o usuário.
- **Área mínima de Interação:** Alvos de clique e toque fáceis de acionar.
- **Teclas de Atalho:** Facilitação do uso via teclado.
- **Tecnologia assistiva:** Compatibilidade com leitores de tela e outros dispositivos.

Conteúdo e Feedback

- **Informações, Conteúdos e Feedback:** Mensagens claras sobre ações do sistema.
- **Formulários:** Campos bem identificados e validações acessíveis.
- **Som e Movimento:** Opções para pausar animações ou ajustar áudio.
- **Suporte a Idiomas**: Preparação para tradução e caracteres especiais.

Processos e Validação

- **Documentação:** Registro de padrões e guias de uso.
- **Teste e Pesquisa:** Validação real com usuários e ferramentas de auditoria.

1. Integração com o Time Scrum

- O UX Designer atua no **time de desenvolvimento**, colaborando desde a **concepção do produto** até sua **entrega e validação**. Tem atuação fundamental na ideação das soluções, levantamento de requisitos, construção de personas e Mapas de Jornadas do Usuário.
- Participa das cerimônias do Scrum - Daily, Review e Retrospectiva, trazendo o olhar do usuário para o processo.

2. Apoio ao Dono do Produto (PO)

- O UX Designer ajuda o PO na elaboração e refinamento do Backlog do Produto, contribuindo com Histórias de Usuário, critérios de aceitação e priorização com base em validações com usuários e testes de usabilidade.

3. Prototipação e Validação

- Criação de **protótipos navegáveis** para validar hipóteses antes da codificação. Criam protótipos navegáveis com base nas histórias de usuário priorizadas, seguindo o Padrão Digital do Governo Federal. Utilizam ferramentas como o Figma e Adobe XD, que promovem colaboração e a visualização antecipada da solução.

- Realização de **testes de usabilidade** rápidos, quando possível com usuários reais, dentro do tempo da sprint. Antes da entrega, verifica-se a conformidade com a Definição de Pronto (Definition of Done - DoD), que inclui:
    - Validação do protótipo implementado;
    - Ausência de erros críticos;
    - Conformidade com o design system e critérios de acessibilidade.

4. Feedback Contínuo

- O UX participa da **Sprint Review** e observa o uso real dos incrementos, sugerindo melhorias com base em dados e interações dos usuários.

Levantamento de Requisitos e HU´s

Dentre as várias técnicas existentes para levantamento de requisitos, e que também podem ser utilizadas para o mapeamento de Histórias de Usuários, a metodologia FNDE Ágil propõe fortemente a utilização contínua de **prototipação como ferramenta de apoio. Ela é usada para materializar rapidamente ideias, facilitar a comunicação entre equipe técnica e usuários e coletar feedback.**

Neste contexto, os protótipos são feitos pelos UX Designers, seguindo as recomendações do Padrão Digital do Governo - Design System Gov.br. O objetivo de uso do design system Gov.br é garantir uma experiência unificada na interação entre os usuários e os sistemas e serviços, sempre seguindo os princípios e boas práticas recomendadas.

Dessa forma, o produto final deve não apenas manter a qualidade aprovada pelo demandante mas também atender às proposições fundamentadas de usabilidade: o **que for implementado deve seguir as orientações do que foi prototipado.**

Essa validação de protótipo faz parte da lista de requisitos na Definição de Pronto da metodologia, tal como a revisão ou aprovação de código e a ausência de erros críticos.

Ainda, está em construção uma biblioteca de componentes React alinhados com o Padrão Digital do Governo para padronizar e otimizar o desenvolvimento de novos produtos no futuro.

Ferramentas/Técnicas de UX Design

Além das já exemplificadas dentro da metodologia FNDE Ágil, algumas técnicas ajudam a entender melhor os usuários para criar soluções mais eficientes. Devido a grande demanda de trabalho, um grande objetivo a ser alcançado é a utilização dessas ferramentas no dia a dia dos integrantes da equipe.

1. Descoberta e Pesquisa (Entendimento do problema e do usuário)

- **Entrevistas com usuários:** coleta de informações qualitativas sobre hábitos, dores e expectativas.
- **Pesquisas quantitativas (surveys):** coleta de dados estruturados em maior escala.
- **Desk research:** análise de dados e materiais existentes.
- **Mapa de stakeholders:** identificação de todos os envolvidos na solução.
- **Shadowing:** observação do usuário em seu contexto real.

2. Análise e Síntese (Compreensão e definição de necessidades)

- **Personas:** representação fictícia de tipos de usuários.
- **Mapa de empatia:** entendimento profundo de sentimentos, pensamentos e comportamentos.
- **Jornada do usuário (user journey):** mapeamento do caminho do usuário ao interagir com o serviço ou produto.
- **Diagrama de afinidades (affinity diagram):** agrupamento de insights e dados qualitativos por temas.

3. Ideação (Geração de soluções)

- **Brainstorming:** geração livre de ideias.
- **Crazy 8´s:** técnica visual para gerar múltiplas ideias rapidamente.
- **Matriz CSD (certezas, suposições e dúvidas):** organiza o nível de clareza sobre o problema.

4. Validação e Testes

- **Testes de usabilidade:** aplicação prática do protótipo por usuários reais para identificar dificuldades.
- **Teste A/B:** comparação entre duas versões de uma solução.

- **Card sorting:** organização de conteúdos para melhorar a arquitetura da informação.
- **Heurísticas de Nielsen:** avaliação da interface com base em critérios estabelecidos.

10 TESTE E QUALIDADE

10. Teste e Qualidade

Qualidade de Software no Desenvolvimento Ágil

A qualidade de software é um dos pilares do desenvolvimento ágil. Ela influencia diretamente a satisfação do usuário, a manutenção do sistema e a viabilidade comercial do produto.

No contexto ágil, qualidade não é responsabilidade apenas do QA ou testador – é responsabilidade de todo o time.

Colaboração e Responsabilidade Compartilhada

No formato ágil, toda a equipe é responsável pela qualidade. QAs e testadores colaboram de forma contínua com POs, desenvolvedores e stakeholders, garantindo que as entregas estejam alinhadas ao valor de negócio.

O time de desenvolvimento, autogerenciável e multifuncional, apoia o Product Owner com as habilidades necessárias para definir objetivos e estratégias, criar e priorizar o backlog e entender o Mínimo Produto Viável (MVP).

A comunicação constante entre os membros da equipe permite detectar e corrigir defeitos antecipadamente, elevando a qualidade do Produto Final.

Participação do QA no Ciclo Ágil

O QA atua de forma ativa e contínua nas cerimônias ágeis:

- Refinamento: valida critérios de aceitação e identifica riscos de qualidade.
- Planejamento: colabora na estimativa de esforço e estratégias de teste.
- Dailys: reporta bloqueios e acompanha o progresso da equipe.
- Reviews: valida entregas e coleta feedbacks dos usuários e stakeholders.

Qualidade, no ágil, significa entregar valor real. Mais do que atender requisitos técnicos, é garantir que o produto seja útil, usável e confiável.

O backlog é refinado com critérios de aceitação claros, que orientam o desenvolvimento

e os testes.

Práticas Ágeis que Promovem Qualidade

- Testes contínuos durante o sprint.
- Automação de testes para regressão e integração.
- Definição de Pronto (DoD) com critérios objetivos.
- Feedback rápido e eficaz entre desenvolvedores e testadores.
- Prevenção de defeitos ao invés de apenas detecção.

Fluxo de Teste e Qualidade no Ágil

1. Planejamento e Preparação

- Definir objetivos e escopo de testes no sprint.
- Identificar histórias com critérios de aceitação claros.
- Escolher ferramentas de automação e gestão.
- Estimar esforço junto ao time (story points).

2. Design de Testes

- Criar cenários baseados nos critérios de aceitação.
- Priorizar testes (funcionais, regressão, performance, segurança).
- Preparar dados e ambientes de teste.

3. Execução de Testes

- Manuais: exploratórios e de validação de requisitos.
- Automatizados: regressão, smoke e integração contínua.
- Registrar evidências e bugs diretamente na ferramenta.

4. Monitoramento e Feedback

- Discutir bloqueios de QA nas dailies.
- Acompanhar métricas como % de histórias testadas, defeitos por fase e tempo médio para correção.

5. Encerramento da Sprint

-
- Validar entregas com base na Definição de Pronto.
- Realizar retrospectiva e identificar melhorias.
- Atualizar a suíte de regressão.

Fluxo Simplificado

Após a História de Usuário (HU) ser definida e homologada pelo Gestor/PO, o QA elabora os casos de teste e revisa os critérios de aceitação da HU em questão.

Com a sprint em andamento, o QA cria os cenários de teste e prepara a automação, quando aplicável.

Quando a feature está pronta – ou seja, com o desenvolvimento concluído e liberado para testes – a equipe de QA executa os testes manuais, testes automatizados (quando aplicável) e testes de integração contínua.

Durante a execução dos testes, o QA registra as evidências de sucesso e as evidências de falha, notificando a equipe de desenvolvimento sobre a necessidade de correção das inconsistências identificadas.

Após a correção dos defeitos pelo desenvolvedor e a liberação para reteste, a equipe de QA realiza o teste de fumaça (smoke test), que consiste em uma bateria rápida de testes preliminares para validar as funcionalidades básicas e a integração entre a versão anterior e a versão atual, conforme a HU.

Em seguida, são realizados os testes de fluxo ponta a ponta (end-to-end), validando interações entre telas, módulos ou sistemas, bem como comportamentos encadeados. Esses testes têm como objetivo garantir que o sistema esteja estável e com as inconsistências corrigidas, estando apto para a homologação pelos Gestores/POs.

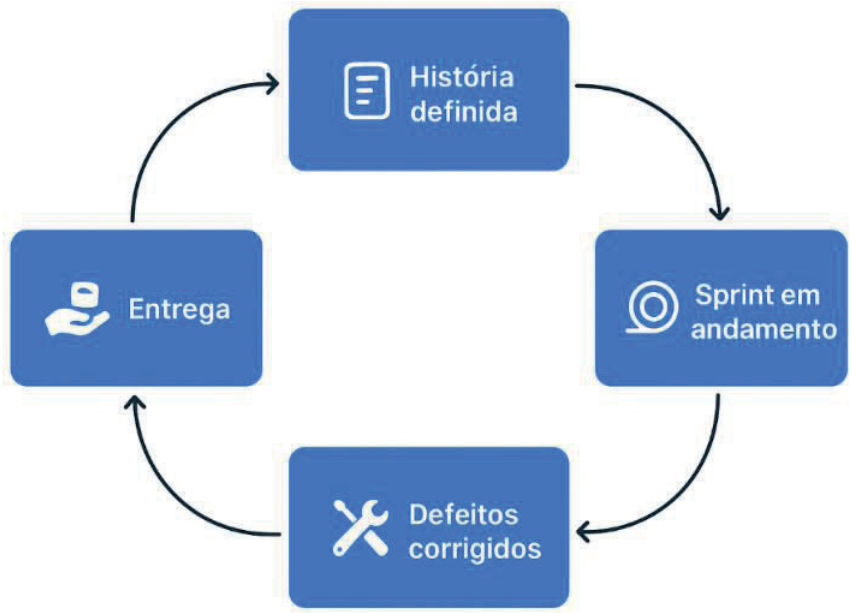
Sempre que uma funcionalidade for finalizada dentro da sprint, o status da atividade deve ser atualizado e disponibilizado para homologação pelo Gestor/PO, sendo atribuída ao responsável.

Após a homologação pelo PO, devem ser incluídas as evidências de sucesso e o status da

atividade deve ser alterado para “Concluído”, formalizando a homologação.

Em caso de falha na homologação, devem ser incluídas as evidências da inconsistência na ferramenta utilizada, com a descrição detalhada da falha identificada.

Todas as atribuições, registros de evidências e notificações devem ser realizadas no JIRA, nas atividades correspondentes à sprint em questão.



Entregáveis do QA em Projetos Ágeis

- Testes automatizados
- Planos de teste (leves e objetivos)
- Testes manuais e relatórios de defeitos
- Logs e métricas de resultados

A documentação deve ser enxuta e útil, priorizando o que agrega valor ao time e ao cliente.

Em contextos regulatórios ou críticos, pode ser necessário formalizar artefatos adicionais.

Qualidade Técnica Contínua

O teste realizado pela equipe de QA tem como objetivo validar se o sistema atende à Definição de Pronto (DoD). Esse processo garante que cada funcionalidade se comporte conforme os requisitos e critérios de aceitação definidos, sendo executado antes da homologação.

Funcionalidades não previstas na História de Usuário (HU) não devem ser consideradas erro. Caso não tenham sido solicitadas ou acordadas durante a sprint, devem ser tratadas como melhorias. Da mesma forma, tudo o que não estiver documentado na HU, no layout ou nos critérios de aceitação deve ser considerado escopo futuro, sendo incluído na fila de demandas para as próximas sprints, conforme a prioridade definida.

Alterações em fluxos de negócio ou regras após a HU estar pronta e homologada devem ser tratadas como ajustes de requisitos e planejadas como demanda para uma sprint futura.

A validação realizada pelo cliente, usuário final ou Product Owner (PO) tem como objetivo confirmar se o sistema atende às necessidades do negócio, assegurando que o produto entrega valor ao usuário e está apto para ser disponibilizado em produção.

Outros aspectos essenciais para a qualidade do produto incluem manutenibilidade, performance, segurança, confiabilidade, integração e entrega contínuas (CI/CD) e refatoração contínua.

Métricas de cobertura de testes auxiliam na avaliação do desempenho e da eficácia das atividades de teste, garantindo níveis mínimos de cobertura definidos pelo time.

No desenvolvimento ágil, a qualidade não é uma etapa final, mas um processo contínuo e colaborativo. Quando todos os membros da equipe compartilham a responsabilidade pela qualidade, o resultado é um produto mais estável, relevante e valioso para o usuário.

11 RELEASE

11. Release

Prática complementar ao Scrum, a Release é uma versão do produto que é disponibilizada aos usuários ou clientes em um determinado momento. Ela representa um conjunto de funcionalidades e melhorias que foram desenvolvidas ao longo de várias sprints e que estão prontas para serem entregues. Tem como objetivo fornecer valor aos usuários e permite que a equipe de desenvolvimento e os stakeholders acompanhem o progresso do projeto.

Apesar do conceito de Release estar vinculado ao ambiente de Produção, ou seja, a entrega de um Produto em ambiente produtivo, ela necessariamente deve passar pelo ambiente de homologação para ser homologada pelo Dono do Produto, e se possível pela esteira de qualidade (QA – quality assurance) sendo acompanhada pelo Arquiteto alocado no Time Scrum.

Na esteira de QA devem ser executadas as duas principais atividades:

1. Planejamento e execução de testes: os testes são realizados para verificar se o sistema atende aos requisitos de qualidade estabelecidos. Eles podem ser realizados manualmente ou usando ferramentas de automação.
2. Gestão de defeitos: os defeitos são identificados e corrigidos durante os testes. A esteira de qualidade é responsável por gerenciar o processo de identificação e correção de defeitos juntamente com a Fábrica de Software

Somente após a homologação e teste da esteira de qualidade, o Produto construído pode ser disponibilizado em ambiente de Produção.

O time pode elaborar um documento de Planejamento de Release, na qual é feita uma previsão de quando um conjunto de funcionalidades será disponibilizado, contendo o escopo provável (a lista de itens a ser incluída na release) e a data prevista de entrega. Pode-se incluir também uma estimativa de capacidade do time por sprint e uma estimativa de alto nível de cada um dos itens da lista. Esse plano é adaptável e deve ser revisto durante a Revisão da Sprint e deve ser transparente a todos os envolvidos.

Plano de Release

- Objetivo da Release
- Itens de maior prioridade
- Principais riscos
- Características gerais
- Quais funcionalidades constarão da release
- Data prevista de entrega

12 MONITORAMENTO E INSPEÇÃO

12. Monitoramento e Inspeção

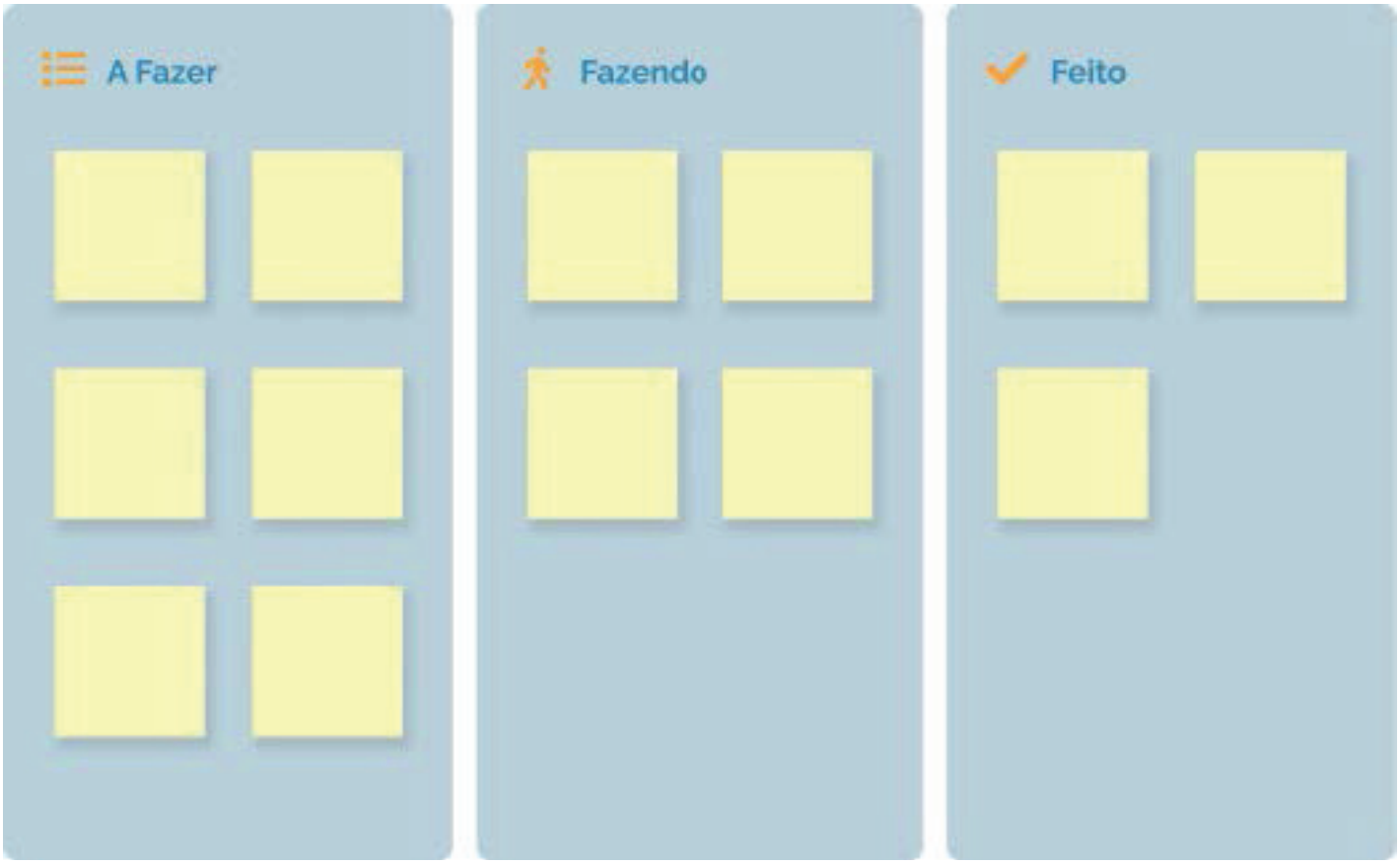
A inspeção promove a transparência das atividades realizadas pelos times. Ela deve ser contínua e faz parte de todas as etapas do Scrum.

12.1 Kanban – quadro Kanban

"Deixe o trabalho visível e não inicie mais trabalho do que você pode lidar."

O Kanban é uma metodologia visual de gestão de fluxo de trabalho que ajuda as equipes a visualizar, organizar e acompanhar suas tarefas de maneira eficiente. Ele utiliza quadros ou painéis para representar as etapas de um processo e cartões para representar as tarefas individuais.

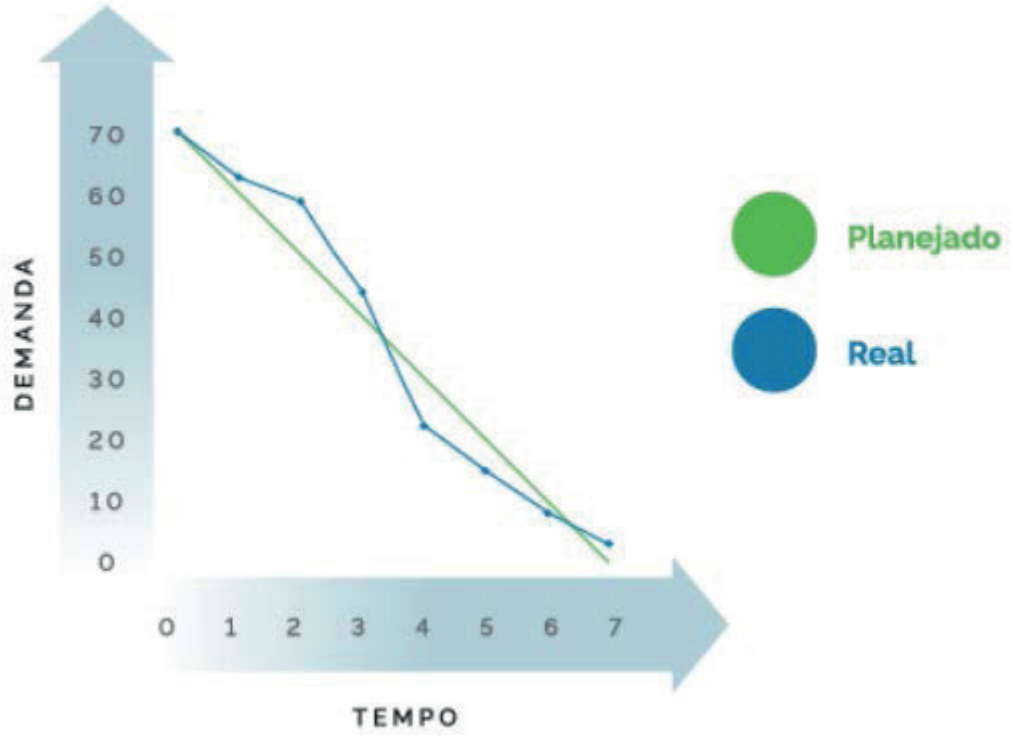
O uso do Kanban ajuda as equipes a se tornarem mais eficientes, focadas e produtivas, ao mesmo tempo em que proporciona maior transparência e controle sobre o trabalho realizado.



12.2 Gráfico Burndown

É uma ferramenta visual amplamente utilizada em metodologias ágeis, como o Scrum, para acompanhar o progresso de um projeto ao longo do tempo.

Ele fornece uma representação clara e concisa do trabalho restante versus o tempo disponível, ou os objetivos que precisam ser alcançados versus tempo.



Ao final de cada Sprint, a equipe deve revisar sua velocidade (número de pontos de história concluídos) com o objetivo de manter uma cadência sustentável e previsível. O Gráfico Burndown auxilia na visualização do ritmo de entrega e na identificação de eventuais impedimentos ou desvios.

13 REFERÊNCIAS

13. Referências

CAROLI, Paulo. Lean Inception. Como alinhar pessoas e construir o produto certo. Editora Caroli, 2018

LEWRICK, Michel. A Jornada do Design Thinking: transformação digital prática de equipes, produtos, serviços, negócios e ecossistemas. Alta Brooks, 2019

MAGNO, Alexandre. Tire seu projeto do papel com Scrum. Ed LeYa, 2019

SUTHERLAND, Jeff e J.J. – SCRUM, A arte de fazer o dobro do trabalho na metade do tempo. Ed Sextante, 2014

BRASIL. Lei n. 13.146, de 6 de julho de 2015. Lei do Governo Digital. Disponível em: https://www.planalto.gov.br/ccivil_03/_ato2015-2018/2015/lei/l13146.htm. Acesso em: 15 mar. 2023.

CENSO, I. B. G. E. Disponível em: https://www.ibge.gov.br/estatisticas/sociais/rendimento-despesa-e-consumo/9662-censo-demografico-2010.html. Acesso em: 15 mar. 2023.

Fenner, Priscila. “Entenda O WCAG 2.0 de Forma Simples E Rápida - Hand Talk.”, 4 Apr. 2019, ‹www.handtalk.me/br/blog/wcag-2-0/›.

MELO, Amanda Meincke; BARANAUSKAS, M. Cecília C. Design para a inclusão: desafios e proposta. In: Proceedings of VII Brazilian symposium on Human factors in computing systems. 2006. p. 11-20.

SCHIMIGUEL, Juliano et al. Accessibility as a quality requirement: geographic information systems on the web. In: Proceedings of the 2005 Latin American conference on Human-computer interaction. 2005. p. 8-19.

BRASIL. Lei n. 14.129, de 29 de março de 2021. Lei do Governo Digital. Disponível em: https://www.planalto.gov.br/ccivil_03/_ato2019-2022/2021/lei/l14129.htm. Acesso em: 15 mar. 2023.

SUTHERLAND, J. J. Scrum: guia prático. Rio de Janeiro: Editora Sextante, 2020.

Ecoprod - Página inicial (google.com)

14 ANEXOS

Template para História do Usuário (HU)

História do Usuário é uma descrição simples, breve e informal de uma funcionalidade ou necessidade do usuário, escrita a partir do ponto de vista desse usuário. Ela representa o que o usuário deseja realizar com o produto e por qual motivo, focando no valor entregue e nas suas reais demandas.

Indicador correspondente no Termo de Referência (TR):

- NMS-05: INDICADOR DE ACEITAÇÃO DE SPRINTS/ENTREGAS (IAS)
- NMS-09: INDICADOR DE PRODUTIVIDADE ÁGIL (IPA)

Template Sugerido:
‹ Deverá ser incluído e transcrito na ferramenta Atlassian/ Jira no "espaço" do Projeto›

TÍTULO: Sigla do Sistema–HU‹nº sequencial›: Nome da Funcionalidade
‹seguir padrão de nomenclatura›

1. DESCRIÇÃO

COMO xxxx
‹Descreva quem é o usuário ou persona que terá a necessidade.›

QUERO xxxx
‹ Especifique o que o usuário quer fazer ou a funcionalidade desejada.›

PARA QUE xxxxx
‹Explique o motivo ou benefício que essa ação/funcionalidade trará ao usuário.›

DESCRIÇÃO RESUMIDA do que vai ser construído / entregue (Objetivo):

- xx

FLUXO:
‹Representa a sequência de passos ou etapas que um usuário realiza para alcançar um objetivo ou executar uma funcionalidade no sistema. É como um roteiro da interação do usuário, explicando o caminho esperado na experiência.›

- xx

DADO QUE ... xxxxxxx,
‹Usado para descrever as condições iniciais ou o estado do sistema antes que a ação principal ocorra. Define o cenário prévio que deve estar estabelecido para que o usuário possa iniciar o fluxo. Essa etapa garante que todos saibam qual é a situação de contexto ou pré-requisito para o início daquele passo.›

E xxxxxxx,
E xxxxxxx.

QUANDO xxxxxx;
‹Indica o evento ou ação realizada pelo usuário que desencadeia uma mudança ou o próximo passo no fluxo. Representa o gatilho ou condição que faz o sistema responder ou avançar no processo. Junto com "Dado" e "Então" forma a estrutura condicional típica do comportamento esperado em histórias e casos de uso.›

E xxxxxxx,
E xxxxxxx.

ENTÃO:
‹Tem a finalidade de descrever o resultado ou o comportamento esperado do sistema após a ocorrência do evento especificado no "QUANDO". É a resposta do sistema às ações do usuário, indicando o que deve acontecer de forma clara e objetiva para que a história seja considerada cumprida.›

- xxxxxxxx
- xxxxxxxx

2. CRITÉRIOS DE ACEITE:

‹Definir condições específicas que deve cumprir para ser considerada pronta.›

‹Servem como testes e validações da história.›

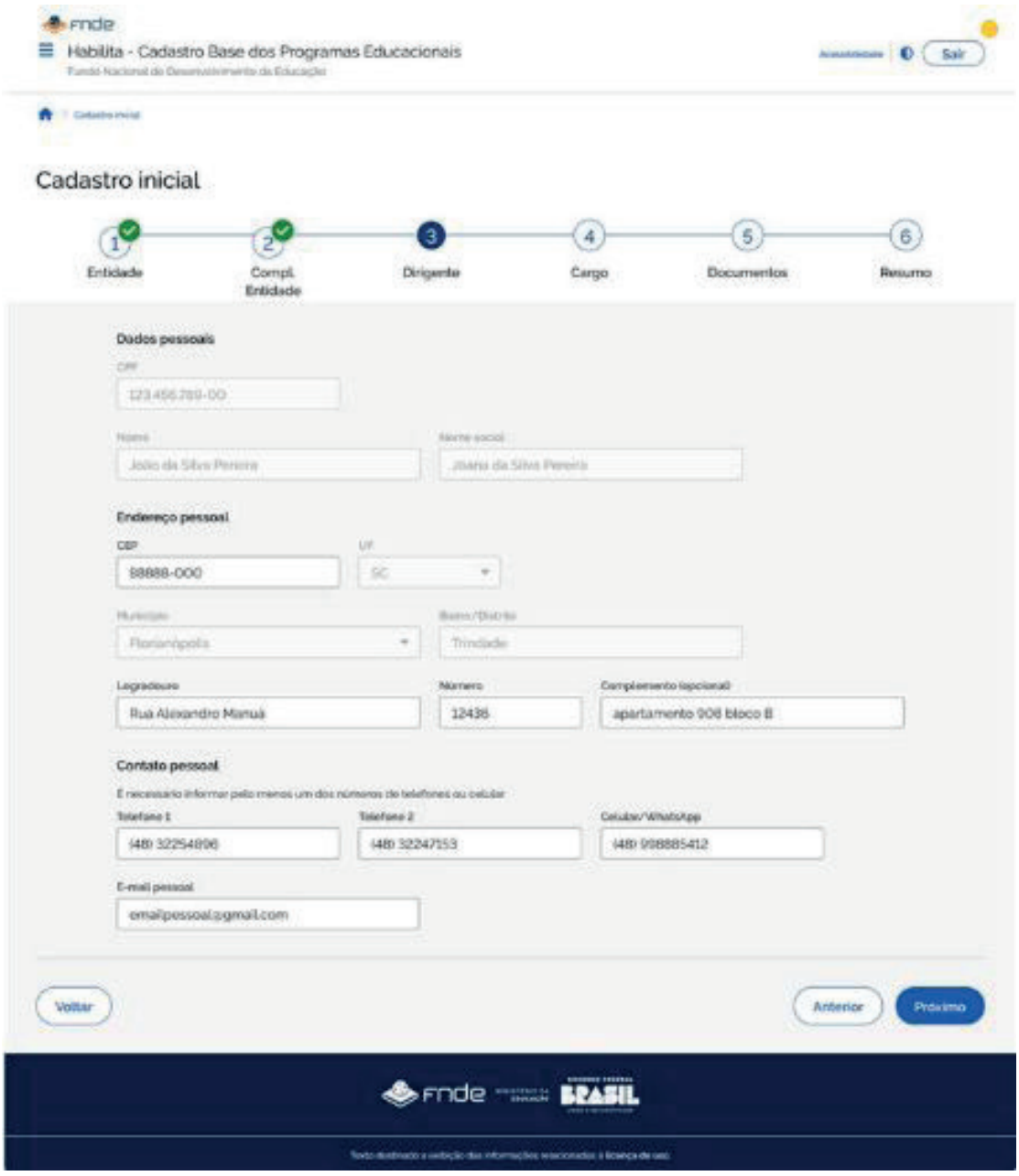
1. xxxxxxxxx
2. xxxxxxxxx

3. PROTÓTIPO:

‹refere-se à representação visual ou ao modelo inicial da interface ou funcionalidade planejada. Ajuda a conectar a história do usuário com aspectos visuais e de usabilidade concretos, garantindo que o desenvolvimento atenda ao design e à experiência planejada.›

Exemplo:

Cadastro Base › Habilita Cadastro Base dos Programas › Cadastro Inicial



(*) Informar o nome do Sistema > Nome da Funcionalidade Principal > Tela Específica
xxxxxxxxx

Tela(s):
‹incluir protótipo HU›

4. PRIORIDADE
‹Alta, Média, Baixa - para ajudar na ordenação do backlog.›

5. ESTIMATIVA
‹Tamanho ou esforço esperado para implementação, em “story points” (pontos de história).›

6. NOTAS / OBSERVAÇÕES
‹Refere-se a informações adicionais, restrições, dependências e detalhes técnicos relevantes ›

- **Regras de Negócio**

‹Descrita (quando aplicável apenas à HU específica) ou Referência Regra Geral (que deverá ser documentada de forma reutilizável e centralizada para evitar repetição e facilitar manutenção – criar link referenciando Repositório de Regras de Negócio na ferramenta Jira – que poderá ser por sistema ou regra estruturante do FNDE.›

- ‹‹ Link do Jira com as Regras de Negócio compilada – Documento Regras de Negócio e Mensagens do Sistema (RNG - ‹Nome Sistema› ››

7. DICIONÁRIO DE DADOS:
‹Documento estruturado que descreve detalhadamente os dados envolvidos na funcionalidade a ser desenvolvida.›

Nome do campo	Descrição	Tipo de campo	Tipo do Dado	Tamanho	Máscara	Editável	Obrigatório	Regras
CPF	Preenchimento automático pelo sistema	texto	numérico	N/A	000.000.000-00	N	S	O sistema deve preencher automaticamente com base no serviço **API WS RECEITA.**
								RNG001
								RNG002

Caso de Teste

Caso de Teste é um **conjunto estruturado de condições, dados e passos** criados para **verificar se uma funcionalidade ou requisito do sistema funciona corretamente**. É um **cenário controlado de verificação** que define **o que será testado, como será testado e qual o resultado é esperado.**

Indicador correspondente no TR:

- NMS-11: INDICADOR DE COBERTURA DE TESTES (ICT).
- NMS-07: INDICADOR DE QUALIDADE DE CÓDIGO (IQC)

Template Sugerido:

<< deve estar descrito no sistema Jira no diretório do respectivo sistema>>

- **Título:** Sigla do Sistema–HU<nº sequencial>:CT - Nome da HU.

<seguir padrão de nomenclatura.>

- Pré-condição de acesso a funcionalidade

<Para cada condição de execução, descrever o estado obrigatório do sistema antes do início do teste.>

- Caso de Teste: Sigla do Sistema–CT<nº sequencial>: Nome do Caso de Teste

<Para cada caso descrever o caminho de determinada funcionalidade ou verificação para atendimento de um requisito específico e/ou validação de regra ou mensagem.>

- Pré-Condição de Execução do Caso de Teste

<Para cada condição de execução (Caso de Teste), descrever o estado obrigatório de acesso a funcionalidade, regra, mensagem ou requisito específico, antes do início do teste.>

Passos:

‹Descrever a sequências de passos a serem executados durante o teste, os valores de entrada de dados e os pontos de verificação onde é necessária maior atenção do Testador.›

- Resultado Esperado

‹É o estado resultante ou as condições observáveis esperadas como resultado da execução do teste. Observar que isso pode incluir respostas positivas e negativas (como condições de erro e falhas).›

- Resultado do Teste

‹É o estado resultante ou as condições observáveis da execução do teste. Observar que isso pode incluir respostas positivas e negativas (como condições de erro e falhas).›

- Pós-Condição

‹Para cada condição de execução (Caso de Teste), descrever o estado ao qual o sistema deverá retornar para permitir a execução de testes subsequentes.›

Exemplo no JIRA:

CB–HU CB367:CT - [Prefs. e Secr] Visualizar solicitação de cadastro 'Em análise'

Descrição

- **Título:** *CB–HU CB-367:CT - [Prefs. e Secr] Visualiza solicitação de cadastro 'Em análise'*
- **Pré-condição de acesso a funcionalidade**

Autenticar nos sistema com um dos seguintes perfis: Prefeito(a) ou Dirigente no sistema Cadastro Base

- **Caso de Teste:** CB–CT01: Visualizar solicitação com situação Em Análise
- **Pré-Condição de Execução do Caso de Teste**

Cadastro aprovado de usuário no sistema Cadastro Base como perfeito ou dirigente. Ter realizasdo o cadastro da entidade e possuir número ativo da solicitação. Possuir situação igual Em Análise.

- **Passos:**

1. Acessar Cadastro inicial opção de Prefeituras e Secretarias - Consultar Entidades - Prefeituras e Secretarias;
2. Clicar no Ícone Visualizar Solicitação;

- **Resultado Esperado**

O sistema apresenta exibe os dados que correspondem exatamente ao cadastro enviado pelo dirigente. Apresenta o número da solicitação no canto superior direito. Os dados cadastrados são organizados em seções em modo somente leitura (não editáveis).

Dirigente (Dados pessoais, Endereço pessoal, Contato pessoal), **Entidade** (Dados Identificadores e contatos, Tipo e Esfera, Endereço da entidade, Informações do cargo e Dados bancários da entidade), **Documentos** (Documentos anexados e Informações complementares) e **Consulta SIOPE.**

- **Resultado do Teste**

O sistema apresenta exibe os dados que correspondem exatamente ao cadastro enviado pelo dirigente. Apresenta o número da solicitação no canto superior direito. Os dados cadastrados são organizados em seções em modo somente leitura (não editáveis).

Evidência de Teste

Evidência de Teste é **o registro comprovável do resultado da execução de um caso de teste** – ou seja, **a prova de que o teste foi realmente executado e qual foi o comportamento do sistema**. É **a documentação** (visual ou textual) que mostra **o que aconteceu de fato** quando o testador executou um caso de teste.

- Título: Sigla do Sistema–HU‹nº sequencial›: ET - Nome da HU.

‹Seguir padrão de nomenclatura. Descrever o nome do roteiro de testes para no qual os defeitos foram identificados.›

Execuções de Testes

- *Caso de Teste: Sigla do Sistema‹nº sequencial›: ET - Nome do Caso de Teste*

‹Seguir padrão de nomenclatura› ‹ Identificação de determinada funcionalidade, pode ser a verificação para atendimento de um requisito específico. Engloba validação de regra de negócio, validação de uma mensagem, contempla respostas positivas e negativas (como condições de erro e falhas).›

- Descrição da(s) Evidência(s)

‹Descrever de uma forma breve e clara a evidência do teste realizado, basicamente a diferença do resultado esperado para o resultado encontrado.›

- Evidência ‹X› - Nome para a evidência

‹Inserir a evidência ocorrida durante o teste, como: Imagem de captura da tela, Mensagens, Vídeos, Conteúdo de arquivo de log entre outras.›

- Evidência ‹XY› - Nome para a evidência

‹Relatar quantas evidências forem encontradas para o caso de teste, quando for necessária mais de uma para apresentar a inconsistência.›

Exemplo no JIRA:

Evidência de Teste - CB-HU CB-367:CT - [Prefs. e Secr] Visualiza solicitação de cadastro 'Em análise'

Descrição

- **Título:** *CB-HU CB-367:CT - [Prefs. e Secr] Visualiza solicitação de cadastro 'Em análise'*

Execuções de Testes

- **Caso de Teste:** CB-CT02: Botão Voltar - Visualizar solicitação com situação Em Análise
- **Descrição da(s) Evidência(s)**

O sistema não apresenta o botão Voltar.

- **Evidência 1** - Falta o botão voltar na tela.

- **Pós Condição**: *Não se aplica a esse caso.*

Documento de Visão

O documento de visão define o escopo de alto nível e o propósito de um projeto. Uma definição clara do problema, solução proposta e os recursos de alto nível do produto ajudam a estabelecer expectativas e a reduzir riscos. Este documento deve conter as informações identificadas durante o entendimento da demanda, tais como objetivos de negócio, características-chaves do produto, aspectos tecnológicos e riscos, utilizados para orientar o desenvolvimento do produto de software.

Indicador correspondente no TR:

- NMS-08: INDICADOR DE SATISFAÇÃO DO DONO DO PRODUTO (ISP)

Template Sugerido:

Documento de Visão

INTRODUÇÃO

Este documento tem por objetivo trazer clareza quanto à necessidade de desenvolver/implantar o *‹nome do sistema/nome do novo módulo/evolução›*, disponibilizando detalhes sobre as características-chave necessárias para o produto, partes interessadas, restrições e riscos, dentre outras informações relevantes para entendimento da necessidade e do que se pretende alcançar com o projeto.

Escopo e Alinhamento Estratégico

‹Descreve brevemente o escopo deste documento de visão, incluindo a quais *programas, projetos, aplicativos e/ou processos de negócios o documento está associado.* Listar as metas e/ou necessidades do *Plano Diretor de Tecnologia da Informação e Comunicação,* as quais o projeto/manutenção evolutiva está relacionado.›

Termos e Definições

‹Define todos os termos, acrônimos e abreviações necessários para interpretar a visão do negócio corretamente.›

TERMOS E DEFINIÇÕES		
Termo	**Definição**	**Fonte**

ANÁLISE DE CONTEXTO

Oportunidade de Negócios:

‹Exemplo:›

O xxxxxxxxxxx são workshops em que mentores se voluntariam para ajudar meninas a criarem um website. Muitas meninas participam desse evento sem nenhuma ideia do que é desenvolver um sistema, e se apaixonam pela programação. O xxxx acontece sempre presencialmente e não há uma plataforma específica para dar suporte a este tipo de evento, ou até mesmo torná-lo inteiramente remoto.

Relato de Problema:

Xxxxxxxxxxxxxxx

xxxxxxxxxxxxxxx

Resumo do problema:

- O problema xxxxxxxxxxxxxx
- afeta xxxxxxxx
- cujo impacto é xxxxxxxxxxxxxx .
- Uma boa solução seria xxxxxxxxxxxxxx.

‹Exemplo:›

2.2.4 A pandemia da COVID-19 inviabiliza que tais workshops ocorram da forma como são planejados tradicionalmente. Mesmo em períodos normais, muitas meninas que vivem em localidades mais distantes não têm a oportunidade de participar dos workshops

presenciais. Uma plataforma de mentoria online permite não só a inclusão de pessoas de outras regiões geográficas, mas também flexibiliza o tempo de realização do workshop, pois poderá ocorrer assincronamente. Assim, a ferramenta tem como objetivo sanar esse problema e manter todo o acolhimento que os workshops do xxxxxx promovem.

xxxxxxxxxxxxxxx

Resumo do problema:

- O problema da inviabilidade de realização de workshops presenciais, agravada pela pandemia da COVID-19 e pelas barreiras geográficas,
- Afeta meninas que vivem em localidades distantes ou que não dispõem de flexibilidade de tempo para participar das atividades,
- Cujo impacto é a exclusão dessas meninas das ações de mentoria, acolhimento e formação promovidas pelo xxxxxx, reduzindo o alcance e a efetividade da iniciativa.
- Uma boa solução seria o desenvolvimento de uma plataforma de mentoria online, que permita a participação remota e assíncrona, ampliando o acesso, mantendo o acolhimento e garantindo a continuidade dos workshops.

Detalhamento da Necessidade

Parte(s) afetada(s)

‹Mencionar as áreas internas e externas que são afetadas direta ou indiretamente pelo problema e/ou necessidade, indicar o “como” e “por que” (sempre que possível).›

Impacto

‹Descreve o impacto causado pelo problema ou pela ausência da solução para a(s) área(s) de negócio›

Solução de Sucesso

‹Lista as principais características, recursos e benefícios de uma solução bem-sucedida para o problema e/ou necessidade.›

DESCRIÇÃO DOS ENVOLVIDOS E DOS USUÁRIOS

‹Para fornecer produtos e serviços que atendam às necessidades das partes interessadas e dos usuários do sistema, deve-se identificar e envolver todas as partes interessadas como parte do processo de definição dos requisitos. Deve-se também identificar os usuários do sistema e assegurar que as partes interessadas os representem adequadamente. ›

Detalhar a partir do documento "Lean Inception".

Partes Interessadas

‹Lista todas as partes interessadas identificadas.›

Unidade Interessada	***Descrição***	***Responsabilidades***
<nome da unidade de negócio interessada>	*<Nome(s) do(s) contato(s) que representa(m) a parte interessada>*	*<detalhar o envolvimento da parte interessada com o projeto, quais os interesses em relação ao projeto e em que ela influência direta e indiretamente>*
<nome da unidade de negócio interessada>	*<Nome(s) do(s) contato(s) que representa(m) a parte interessada>*	*<detalhar o envolvimento da parte interessada com o projeto, quais os interesses em relação ao projeto e em que ela influência direta e indiretamente>*

Exemplo:

Unidade Interessada	***Descrição***	***Responsabilidades***
Equipe de Gestão de Projeto.	*Grupo de alunos da xxx matriculados na disciplina de Engenharia de Produto de Software.*	*Gerenciar a equipe, executando o planejamento, sustentando a comunicação entre os membros da equipe e acompanhando o desenvolvimento do produto.*
Clientes	*Grupo de mentores e aprendizes da plataforma "xxxxx".*	*Informar os requisitos necessários, expor os erros, além de acompanhar o progresso da aplicação e validá-la.*

Tipos de Usuários do Sistema

Resumo dos Tipos de Usuários

Nome	Descrição	Responsabilidades
[Especifique o nome do tipo de usuário.]	*[Descreva brevemente o tipo de usuário.]*	*[Resuma as principais responsabilidades do usuário no que diz respeito ao sistema que está sendo desenvolvido; ou seja, seu interesse como envolvido.]*
[Solicitante Interno]	*[Todo funcionário CAIXA. Prestadores não são solicitantes.]*	*[Abrir uma ocorrência no sistema e acompanhar seu atendimento]*
***Ex:** Professores das escolas primárias.*	*Irão utilizar a aplicação com o objetivo de dar suporte de forma remota aos alunos, de acordo com a demanda*	*Publicar conteúdo e material didático para cada uma das turmas e aulas; Realizar atendimento remoto por meio da plataforma.*

VISÃO GERAL DO PRODUTO

‹Descrever também a perspectiva do produto com relação a outros produtos relacionados e ao ambiente do usuário. Se o produto for independente e totalmente autocontido, indique-o aqui. Se o produto for um componente de um sistema maior, relacione como esses sistemas/componentes interagem. Uma maneira de exibir os principais componentes, interconexões e interfaces externas é por meio do mapeamento do processo de negócios ou através da criação de diagramas de casos de uso.›

Esta seção fornece uma visualização de alto nível das capacidades do produto, interfaces para outros aplicativos e configurações dos sistemas.

Características-Chave do Produto

‹Listar e descreve brevemente as características-chave do produto (serviço ou macro funcionalidade que agrega valor ao negócio do cliente, dispensando uma descrição mais detalhada). As características-chaves são atributos de valor ou serviços providos pela solução, derivados dos objetivos do negócio, para satisfazer necessidades de negócio dos clientes, em outros termos, são os aspectos mais relevantes do produto de software que o cliente valorizará com mais facilidade. Podem ser funcionalidades de negócio, performance, segurança, escalabilidade, etc.. As necessidades e expectativas das partes interessadas, relacionadas aos objetivos de negócio, deverão ser utilizadas para

definir o conjunto de características-chaves do produto final e deverão estar fortemente alinhadas.›

‹As características-chave devem ser listadas com o auxílio de uma tabela, como no exemplo a seguir.›

Detalhar a partir do documento "Lean Inception".

Sequenciador	Funcionalidades MVP	Descrição Macro da Funcionalidade*
1	*Cadastro de xxxxxxxxxxx*	
	Registro xxxxxxxxxx	
	Consultar xxxxxxxxxx	
2	*Cadastro de xxxxxxxxxxx*	
	Registro xxxxxxxxxx	
	Consultar xxxxxxxxxx	
...		

(*) Será detalhada no documento "História do Usuário (HU)"

Requisitos não funcionais

‹Definir os requisitos não funcionais para a solução. Requisitos não funcionais são condições que não se relacionam diretamente ao comportamento ou funcionalidade da solução, mas descrevem condições ambientais sob as quais a solução deve permanecer efetiva, ou qualidades que os softwares precisam possuir. Também são conhecidos como requisitos de qualidade ou suplementares. Podem incluir requisitos relacionados à capacidade, velocidade, segurança, disponibilidade, arquitetura da informação e apresentação da interface com o usuário.›

Tipo	Descrição	Prioridade
< Usabilidade, Acessibilidade, Desempenho, Segurança, Interface, etc.>	<Breve descrição do requisito>	<Prioridade de atendimento em relação aos demais requisitos>

Suposições e Dependências

‹Listar cada um dos fatores que afetam os recursos que o documento de visão inclui. Lista as suposições que, se modificadas, alterarão o documento de visão. Por exemplo, uma suposição pode indicar que um sistema operacional específico fique disponível para o hardware designado para o produto de software. Se o sistema operacional não estiver disponível, será necessário alterar o documento de visão.›

Custos

‹Registrar os impactos e restrições relevantes de custo e precificação. Por exemplo, os custos de distribuição (número de licenças de software para visualização de dados) ou outras restrições de custo de disponibilização, que podem ser fundamentais para o sucesso do projeto.›

OUTROS REQUISITOS DO PRODUTO

Em um alto nível, lista os padrões aplicáveis, os requisitos de hardware ou plataforma, os requisitos de desempenho e os requisitos ambientais.

Padrões e Normativos Aplicáveis

‹Lista todos os padrões e normativos que o produto deve estar em conformidade, tais como:

- Padrões e normativos jurídicos e regulamentares;
- Padrões e normativos de qualidade e segurança;
- Padrões de interoperabilidade (Ex. ePING);
- Padrões de acessibilidade (Ex. eMAG, WCAG);
- Padrões de usabilidade. ›

Padrão ou normativo

Padrão ou normativo	Descrição	Acesso
<Nome do padrão ou normativo>	*<Breve descrição do padrão>*	*<Indicar como ter acesso ao documento>*

Requisitos Técnicos da Solução

‹Define os requisitos necessários para a solução. Incluem os sistemas operacionais, SGBDs, memória, espaço em disco, processamento, dispositivos periféricos, softwares de terceiros, etc. ›

Tipo	Descrição	Quantidade
<Memória, Processamento, Espaço de armazenamento, etc.>	*<Breve descrição sobre o requisito e, se necessário, as justificativas>*	*<Quantidade numérica necessária do recurso>*

Requisitos de Documentação

‹Esta seção descreve a documentação que deve ser desenvolvida para que o projeto seja bem-sucedido. A documentação pode ser relacionada a guias de utilização, guias de instalação, etc..

Tipo	Descrição	Formato
<Documentação técnica, documentação de usuário>	*<Breve descrição sobre os detalhes necessários na documentação>*	*<Formato de disponibilização da documentação (Web, PDF, dentro do sistema, etc.)>*

NÃO ESCOPO

‹Neste campo devem ser descritos os impactos, custos e restrições que não serão tratados pelo projeto. O objetivo é deixar claro o que não será abordado ou resolvido pelo projeto, evitando expectativas incorretas, assim, delimitando claramente suas responsabilidades de entrega do projeto.›

Detalhar a partir do documento “Lean Inception”.

NÃO FAZ PARTE DO ESCOPO DO PROJETO	
Id	***Descrição***

PRINCIPAIS RISCOS IDENTIFICADOS

‹Neste item devem ser descritos os principais riscos que podem comprometer o desenvolvimento, a implantação ou o sucesso do projeto. Devem ser considerados riscos de natureza técnica, operacional, financeira, organizacional e externa, como atrasos no cronograma, limitações de recursos, falhas tecnológicas, dependência de terceiros ou mudanças no contexto externo. Para cada risco identificado, recomenda-se apresentar, sempre que possível, uma breve descrição de sua causa e impacto potencial (Alto / Médio / Baixo), de modo a apoiar a tomada de decisões e o planejamento de ações de mitigação.›

PRINCIPAIS RISCOS			
Id	**Descrição**	**Causa**	**Impacto**

RESTRIÇÕES:

‹Listar todas as restrições de design, restrições externas, como requisitos operacionais ou regulamentares, ou outras dependências.›

Tipo	Nome	Descrição
<Financeira, Recursos Humanos, Tempo, Infraestrutura Tecnológica, etc.>	<nome da restrição>	<detalhamento da restrição>

Estudo para Implementação para o "Processo de Liberação" no FNDE

Devem ser consideras as melhores práticas de mercado para a gestão de serviços de TI, em especial a ITIL (Information Technology Infrastructure Library, tradução, "Biblioteca de infraestrutura de tecnologia da informação").

1. Organizações e pessoas;
2. Informação e tecnologia;
3. Parceiros e fornecedores;
4. Fluxos de valor e processos.

Essas dimensões são aplicáveis ao **sistema de valor do serviço** em geral e a serviços específicos.

Sistema de valor de serviço

O Sistema de Valor de Serviço (SVS) representa "como todos os componentes e atividades de uma organização trabalham juntos para facilitar a criação de valor". O SVS do ITIL 4 inclui diversos elementos:

Os principais pontos do ITIL v4 incluem:

- **Princípios orientadores:** como foco no valor, colaboração e otimização.
- **Sistema de valor de serviço (SVS)**: integra governança, práticas, melhoria contínua e cadeia de valor.
- **Práticas de gerenciamento:** substitui o termo "processos" por **práticas**, divididas em:
    - **Gerenciamento geral** (ex.: melhoria contínua, gestão de riscos)
    - **Gerenciamento de serviços** (ex.: gerenciamento de incidentes, problemas, mudanças)
    - **Gerenciamento técnico** (ex.: desenvolvimento de software, gestão de infraestrutura)

Temos no total 34 práticas na versão 4 da ITIL como "conjuntos de recursos organizacionais concebidos para executar um trabalho ou atingir um objetivo".

Para cada prática, a ITIL v4 fornece vários tipos de orientação, como termos e conceitos-chave, fatores de sucesso, atividades-chave, objetos de informação, etc.

As **práticas gerais de gestão** da ITIL v4 incluem:

Melhoria Contínua – Garantir que os serviços, processos e práticas sejam constantemente aprimorados. A parte operacional do dia a dia da melhoria contínua, juntamente com a visão estratégica e a melhoria da cadeia de valor do serviço.
Gestão de Mudanças Organizacionais – Apoiar mudanças culturais e estruturais na organização.

1. **Melhoria Contínua** – Garantir que os serviços, processos e práticas sejam constantemente aprimorados. A parte operacional do dia a dia da melhoria contínua, juntamente com a visão estratégica e a melhoria da cadeia de valor do serviço.
2. **Gestão de Mudanças Organizacionais** – Apoiar mudanças culturais e estruturais na organização.
3. **Gestão de Portfólio** – Gerenciar todos os serviços e produtos oferecidos pela organização.
4. **Gestão Financeira** – Controlar custos e investimentos relacionados aos serviços.
5. **Gestão de Estratégia** – Definir objetivos e direção para os serviços de TI.
6. **Gestão de Riscos (Governança, Risco e Compliance)** – Identificar, avaliar e mitigar riscos.
7. **Gestão de Segurança da Informação** – Proteger dados e informações contra ameaças.
8. **Gestão de Conhecimento** – Garantir que informações úteis estejam disponíveis para quem precisa.
9. **Gestão de Medição e Relatórios** – Monitorar e reportar métricas de desempenho.
10. **Gestão de Fornecedores** – Controlar contratos e relacionamentos com fornecedores.
11. **Gestão de Arquitetura** – Planejar e manter a arquitetura de TI alinhada ao negócio.
12. **Gestão de Projetos** – Planejar e executar projetos de forma eficiente.
13. **Gestão de Recursos Humanos** – Garantir que as pessoas certas estejam nas funções certas.
14. **Gestão de Relacionamento** – Manter boas relações com stakeholders internos e externos.

As **práticas de gerenciamento de serviços** da ITIL v4 incluem:

15. **Gestão de Incidentes** – Restaurar serviços rapidamente após interrupções.
16. **Gestão de Problemas** – Identificar causas raiz e prevenir recorrências.
17. **Gestão de Mudanças** – Controlar alterações para reduzir riscos.
18. **Gestão de Liberação** – Planejar e implementar novas versões de serviços.
19. **Gestão de Configuração** – Manter informações sobre ativos e seus relacionamentos.
20. **Gestão de Disponibilidade** – Garantir que serviços estejam disponíveis conforme acordado.
21. **Gestão de Capacidade** – Assegurar que recursos atendam às demandas.
22. **Gestão de Continuidade de Serviço** – Garantir recuperação após desastres.
23. **Gestão de Nível de Serviço** – Monitorar e garantir cumprimento dos SLAs.
24. **Gestão de Catálogo de Serviços** – Manter lista clara dos serviços oferecidos.
25. **Gestão de Requisição de Serviço** – Atender solicitações dos usuários.
26. **Gestão de Acesso** – Controlar quem pode acessar quais serviços.
27. **Gestão de Disponibilidade de Serviços** – Monitorar uptime e confiabilidade.
28. **Gestão de Monitoramento e Eventos** – Detectar e responder a alertas e eventos.
29. **Gestão de Implantação** – Planejar e executar a entrega de novos serviços.
30. **Gestão de Validação e Testes** – Garantir que serviços atendam aos requisitos.
31. **Gestão de Incidentes Maiores** – Tratar incidentes críticos com prioridade.

As **práticas de gerenciamento técnico** da ITIL 4 incluem:

32. **Gestão de Desenvolvimento de Software** – Criar e manter aplicações.
33. **Gestão de Infraestrutura e Plataforma** – Administrar servidores, redes e ambientes.
34. **Gestão de Implantação de TI** – Garantir que sistemas sejam implementados corretamente.

Existem soluções de mercado que possuem de forma nativa os processos da ITIL implantados, sendo algumas delas certificadas pela entidade certificadora **PinkElephant**. As soluções, atualmente, são certificadas em até 28 processos da ITIL, sendo eles:

IA – Capacidade de IA	FM – Gestão Financeira	PIM – Gestão de Desempenho e Melhoria	SD – Central de Serviços
AM – Gestão de Ativos de TI	GRC – Governança, Risco e Compliance	PJM – Gerenciamento de Projetos	SLM – Gerenciamento de Nível de Serviço
AVM – Gerenciamento de Disponibilidade	IM – Gerenciamento de Incidentes	PM – Gestão de Problemas	SPM – Gestão de Portfólio de Serviços
BRM – Gestão de Relacionamento Comercial	ITO – Gestão de Operações de TI	RDM – Gerenciamento de Lançamento e Implantação	ST – Garantia de Serviço e Testes
CAP – Gestão de Capacidade	KM – Gestão do Conhecimento	RM – Gestão de Pedidos	VM – Gerenciamento de fornecedores
CHG – Gestão de Mudanças	MA – Monitoramento e Alertas	SCA – Gestão de Catálogo de Serviços	WRM – Gestão de Recursos da Força de Trabalho
CON – Gerenciamento de Configuração	OCM – Gestão de Mudanças Organizacionais	SCO – Gerenciamento de Continuidade de Serviço	XLM – Gestão da Experiência

E podemos conferir as soluções certificadas a partir do link oficial da PinkElephant:

- https://www.pinkelephant.com/en-US/pinkverify/pinkverify-certification

A **Metodologia de Desenvolvimento de Soluções Digitais** utilizada no **FNDE** (**FNDE Ágil**) e o novo **Termo de Referência** (itens 9.32.1.9) possuem a previsão de execução do "Planejamento de Releases" / "Planejamento de entregas" (Release Plan), que tem como objetivo principal a disponibilização da versão do produto, formalmente aceita pelo **Product Owner**, em **ambiente de Produção**.

O time deverá elaborar um documento de **Planejamento de Release**, no qual é feita uma previsão de quando um conjunto de funcionalidades será disponibilizado, contendo o escopo provável (a lista de itens a ser incluída na release) e a data prevista de entrega.

Pode-se incluir também uma estimativa de capacidade do time por sprint e uma estimativa de alto nível de cada um dos itens da lista. Esse plano é adaptável e deve ser revisto durante a Revisão da Sprint e deve ser transparente a todos os envolvidos.

1 - Release Request

O processo de **Gestão de Liberação** começa com solicitações de recursos ou modificações nos recursos atuais. Não é garantido que cada solicitação resultará em uma versão futura. O Guia de Gerenciamento de Aplicativos ITIL descreve que cada solicitação é avaliada quanto à sua explicação, sustentabilidade e potencial de atendimento por meio da reconfiguração da versão de produção do aplicativo.

2 - Release Plan ("Planejamento de Releases" / "Planejamento de entregas")

Este é o **estágio mais essencial no desenvolvimento de um release**. A estrutura do lançamento é especificada aqui. Para **planejar e priorizar lançamentos de acordo com as necessidades de negócios e objetivos estratégicos**, o Gerente de Liberação trabalha com outras partes interessadas. Isso inclui criar e reutilizar o escopo e estabelecer metas, marcos e responsabilidades. Também inclui a definição de prazos e a atribuição de recursos necessários para uma implementação bem-sucedida do lançamento.

3 - Release Development (Software Build)

O estágio de desenvolvimento envolve o processo real de projeto e construção dos componentes de lançamento, incluindo aplicativos de software, configurações e documentação. Esta etapa visa emitir ordens de serviço e solicitações de compra para que os componentes de liberação possam ser desenvolvidos internamente ou adquiridos de fornecedores externos. Quando o desenvolvimento estiver concluído, os componentes de lançamento estarão prontos para entrar na fase de testes.

4 - Release Testing

Nesta fase, os componentes de lançamento são exaustivamente testados em vários ambientes, incluindo desenvolvimento, testes e preparação, para identificar e corrigir quaisquer defeitos ou problemas. Isso inclui, entre outras coisas, testes funcionais, testes de integração, testes de desempenho, testes de segurança e testes de aceitação do usuário (UAT – User Acceptance Testing). Este estágio trata de testes extensivos, garantia de qualidade (QA – Quality Assurance) e validação dos componentes de lançamento para garantir que eles atendam às necessidades específicas e estejam prontos para implantação.

5 - Release Deployment

Os componentes de lançamento são implantados no ambiente ativo após passarem em todos os testes necessários e atenderem aos critérios de qualidade definidos. O Release Manager garante que a versão seja implantada de acordo com o cronograma e planos definidos e trabalha com diversas equipes, como operações, infraestrutura e suporte. Este processo visa garantir uma implantação tranquila com impacto mínimo no ambiente ativo e nos usuários.

6 - Release Review

Após a implantação da versão, é realizada uma revisão para medir o sucesso e identificar quaisquer problemas ou lições aprendidas. Esta etapa envolve a revisão dos principais Key Performance Indicators (KPIs), feedback do cliente e relatórios de incidentes para avaliar a eficácia do lançamento e identificar áreas para melhorias futuras.

O Gerente de Liberação e outras partes interessadas avaliam o desempenho e o impacto da versão no ambiente ativo, nas operações de negócios e na satisfação do cliente, e quaisquer lições aprendidas são registradas para projetos futuros.

Esse feedback é usado para melhorar continuamente o processo de gerenciamento de versões e garantir que as versões sejam entregues com alta qualidade e com menor risco.

7 - Release Closure

Nesta fase, a versão é encerrada e a documentação relevante é atualizada assim que for implantada e revisada com sucesso. Isso inclui fazer alterações nas notas de versão, registros de configuração e artigos de conhecimento para refletir as alterações. Também envolve a realização de uma revisão final de todo o processo para capturar e documentar qualquer aprendizado, melhores práticas e recomendações para projetos futuros. O Gerente de Liberação garante que todas as atividades relacionadas, como comunicação, documentação e relatórios, sejam concluídas e que a liberação seja formalmente encerrada. Seguindo a ITIL Framework, as organizações podem simplificar seus processos de Gerenciamento de Liberação.

Indicador equivalente no Termo de Referência (TR):

- NMS-05: INDICADOR DE ACEITAÇÃO DE SPRINTS/ENTREGAS (IAS)
- NMS-08: INDICADOR DE SATISFAÇÃO DO DONO DO PRODUTO (ISP)
- NMS-10: INDICADOR DE CONFORMIDADE EM HOMOLOGAÇÃO (ICH)
- NMS-11: INDICADOR DE COBERTURA DE TESTES (ICT)

Planejamento de Release (Release Plan)

‹ Deverá ser incluído e transcrito na ferramenta Atlassian/Jira no "espaço" do Projeto, existirá mais de um plano de liberação por sistema.›

Planejamento de Release (Release Plan)

1. Identificação da Liberação (Release)

‹seguir padrão de nomenclatura›

TÍTULO: Sigla do Sistema–REL‹versão sistema›: ‹ Nome do novo projeto de software ›

Prazos	
Data prevista de entrada em produção Release:	
Data real de entrada em produção Release:	

2. Objetivo da Liberação

‹Descrever o problema ou necessidade a liberação atende, e o seu valor para o negócio e usuários›

2.1. xxxxxxxx

2.2. Papel e Responsabilidades

Papel e Responsabilidades	
Nome	**Responsabilidades/Função para a release**
	SM/Release Manager (Gerente de Liberação)
	Product Owner/Validar proposta da liberação
	Desenvolvedor xxxxxxx
	Infraestrutura

3. Escopo da Liberação

Conjunto de funcionalidades	
Data Prevista de Entrega: dd/mm/yyyy	
Sprint-<núm controle>	*Sigla do Sistema–HU<nº sequencial>: <Nome da Funcionalidade>*
Data Prevista de Entrega: dd/mm/yyyy	
Sprint-<núm controle>	*Sigla do Sistema–HU<nº sequencial>: <Nome da Funcionalidade>*

3.1. Exclusões (o que não será entregue)

O que não será entregue	Motivo

4. Critérios de Aceite

‹Condições para considerar a liberação concluída. ›

Critérios de Aceite	
Id	***Descrição***

5. Riscos e Dependências

5.1. Principais riscos identificados.

Riscos Identificados	
Id	***Descrição***

5.2. Dependências externas

Dependências Externas (outros times, fornecedores...)	
Id	***Descrição***

5.3. Recursos Necessários

Recursos Necessários (Licenças, Infra, BD...)	
Id	***Recurso***

6. Plano de Comunicação

‹ Indicar como e quando os stakeholders serão informados e os respectivos canais (e-mail, reuniões, dashboards)›

6.1. xxxxxxxxxxxxxxx

7. Plano de Rollback

‹ Descrever a estratégia caso a liberação falhe e os procedimentos para restaurar versão anterior›

7.1. xxxxxxxxxxxxxxx

8. Métricas e Indicadores de Sucesso (previstos no Termo de Referência)

‹ Serão considerados os indicadores abaixo para medir o sucesso ›

- NMS-05: INDICADOR DE ACEITAÇÃO DE SPRINTS/ENTREGAS (IAS)
- NMS-08: INDICADOR DE SATISFAÇÃO DO DONO DO PRODUTO (ISP)
- NMS-10: INDICADOR DE CONFORMIDADE EM HOMOLOGAÇÃO (ICH)
- NMS-11: INDICADOR DE COBERTURA DE TESTES (ICT)

Documento de Arquitetura de Software (DAS)

Este documento provê uma visão de alto nível das integrações do sistema, usando uma visão arquitetural e fornecendo um alinhamento para as decisões de arquitetura e projeto, desde a fase conceitual até a implementação.

Bons diagramas de arquitetura de software auxiliam na comunicação dentro e fora das equipes de desenvolvimento/produto de software, integração eficiente de novos funcionários, revisões/avaliações de arquitetura, identificação de riscos, modelagem de ameaças, etc.

O FNDE definiu como padrão de arquitetura o modelo C4, que busca aumentar o nível de maturidade associado aos diagramas de arquitetura de software. De acordo com esse modelo, existem 5 níveis para representar os modelos de arquitetura de software:

Level 1	Level 2	Level 3	Level 4	Level 5
Initial No software architecture diagrams.	**Ad hoc** Software architecture diagrams with ad hoc abstractions and notation, in a general purpose diagramming tool.	**Defined** Software architecture diagrams with defined abstractions and notation, in a general purpose diagramming tool.	**Modelled** Software architecture diagrams with defined abstractions and notation, in a modelling tool, authored manually.	**Optimising** - Model elements are shared between teams across the organisation. - Models are used as queryable datasets. - Automatic generation of model elements from source code, deployment environment, logs, etc. - etc

Para criar alguns “mapas do seu código”, primeiro precisamos de um conjunto comum de abstrações que possamos usar para descrever a estrutura estática de um sistema de software. No modelo C4:

> “Um sistema de software é composto por um ou mais contêineres (aplicações e repositórios de dados), cada um dos quais contém um ou mais componentes, que por sua vez são implementados por um ou mais elementos de código (classes, interfaces, objetos, funções, etc). E as pessoas (atores, papéis, personas, indivíduos nomeados, etc.) usam os sistemas de software que construímos.”

O modelo C4 é uma abordagem “abstração em primeiro lugar” para diagramar arquitetura de software, baseada em abstrações que refletem como arquitetos e desenvolvedores de software pensam e constroem software. O pequeno conjunto de abstrações e tipos de diagramas torna o modelo C4 fácil de aprender e usar.

Template Sugerido:

Documento de Arquitetura

‹ Deverá ser incluído e transcrito na ferramenta Atlassian/ Jira no “espaço” do Projeto›

TÍTULO: Sigla do Sistema–ARQ: ‹ Nome do novo projeto de software ›

‹seguir padrão de nomenclatura›

1. DESCRIÇÃO

‹‹ descrever arquitetura escolhida ››

2. SISTEMA

EXEMPLO:

2.1 BPS

O sistema **BPS** é uma **aplicação** monolítica desenvolvido em **Java JSP**. Chamada BOLETIM DE PESSOAL E SERVIÇO (BPS), desenvolvido com foco na geração, gerenciamento e publicação de boletins de pessoal e documentos relacionados

Integrações Internas:

Banco de Dados:

- Utiliza o Hibernate para conexão com o banco de dados Oracle.
- Conexão via JNDI: jdbc/BoletimDS

Integrações Externas:

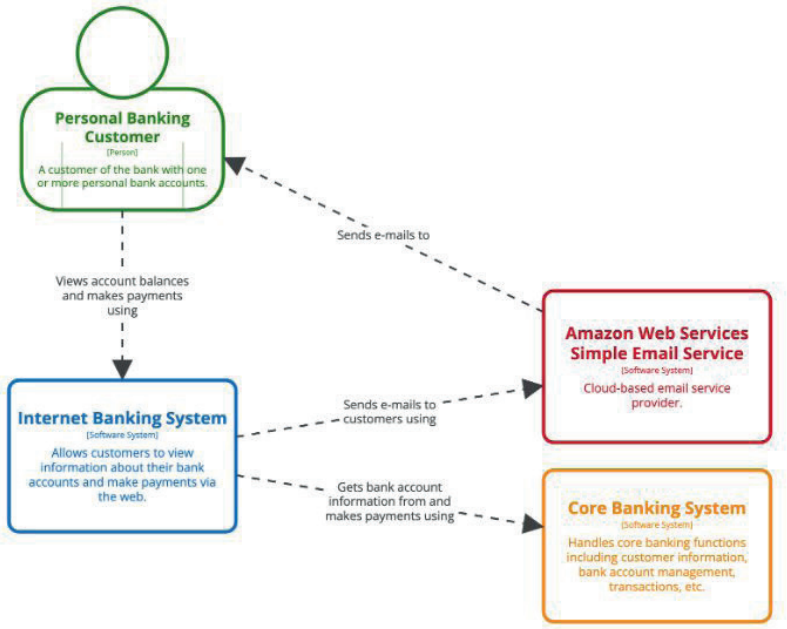
SSO e Autenticação: Integração com o SSO do GOV.BR, usando endpoints como https://sso.acesso.gov.br.

Integrações Externas:

SSO e Autenticação: Integração com o SSO do GOV.BR, usando endpoints como https://sso.acesso.gov.br.

3. DIAGRAMA DE CONTEXTO DO SISTEMA

‹ Um **diagrama de contexto do sistema** é um bom ponto de partida para diagramar e documentar um sistema de software, permitindo a visualização de seu panorama geral. Detalhes não são importantes aqui, pois esta é sua visão afastada mostrando uma visão geral do cenário do sistema. O foco deve estar nas pessoas (atores, papéis, personas, etc.) e nos sistemas de software, em vez de tecnologias, protocolos e outros detalhes de baixo nível. É o tipo de diagrama que você poderia mostrar para **pessoas não técnicas.** ›



4. DIAGRAMA DE CONTÊINER

‹‹ Explode a visualização na fronteira do sistema com um diagrama de contêineres. No C4, um container é uma aplicação ou um armazenamento de dados.

O **diagrama do contêiner** mostra a **estrutura de alto nível da arquitetura de software** e como as responsabilidades são distribuídas entre ela. Também mostra as principais escolhas tecnológicas e como os contêineres se comunicam entre si. É um diagrama simples, focado em tecnologia de alto nível, útil tanto para desenvolvedores de software

quanto para equipe de suporte/operações.››

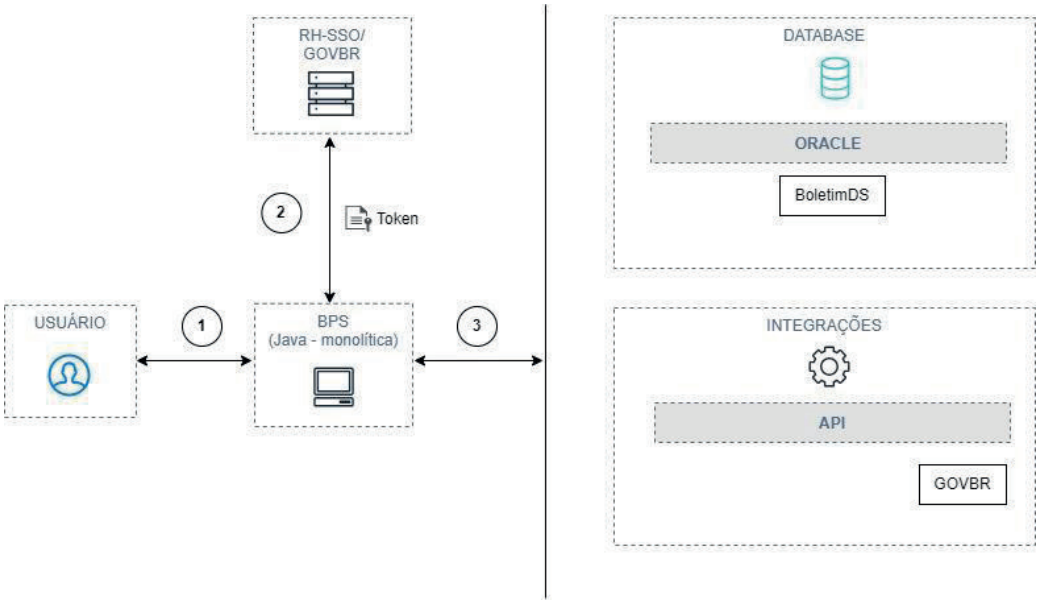
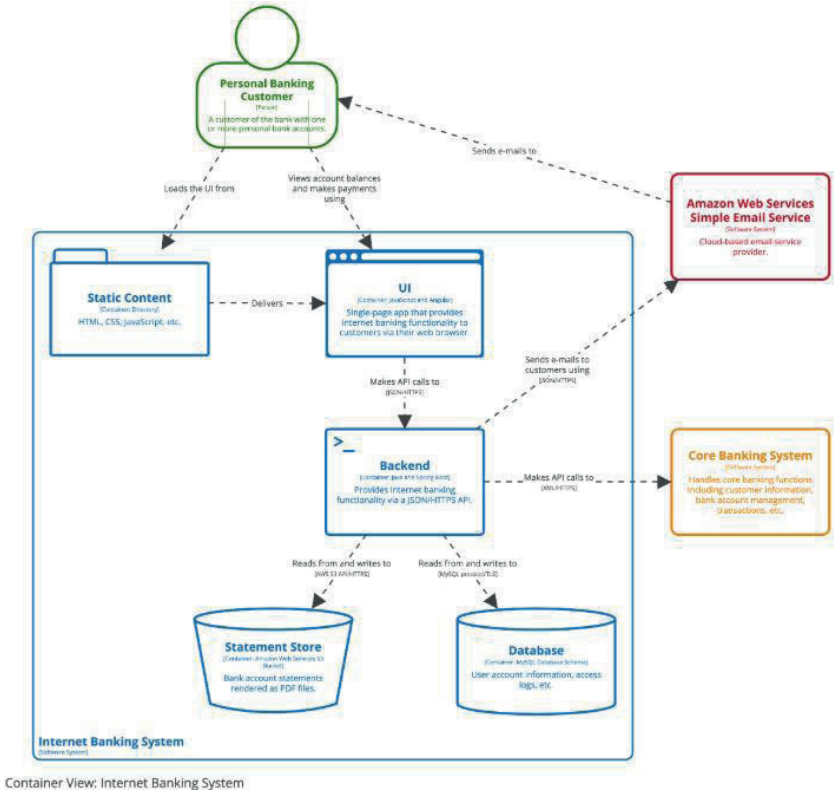


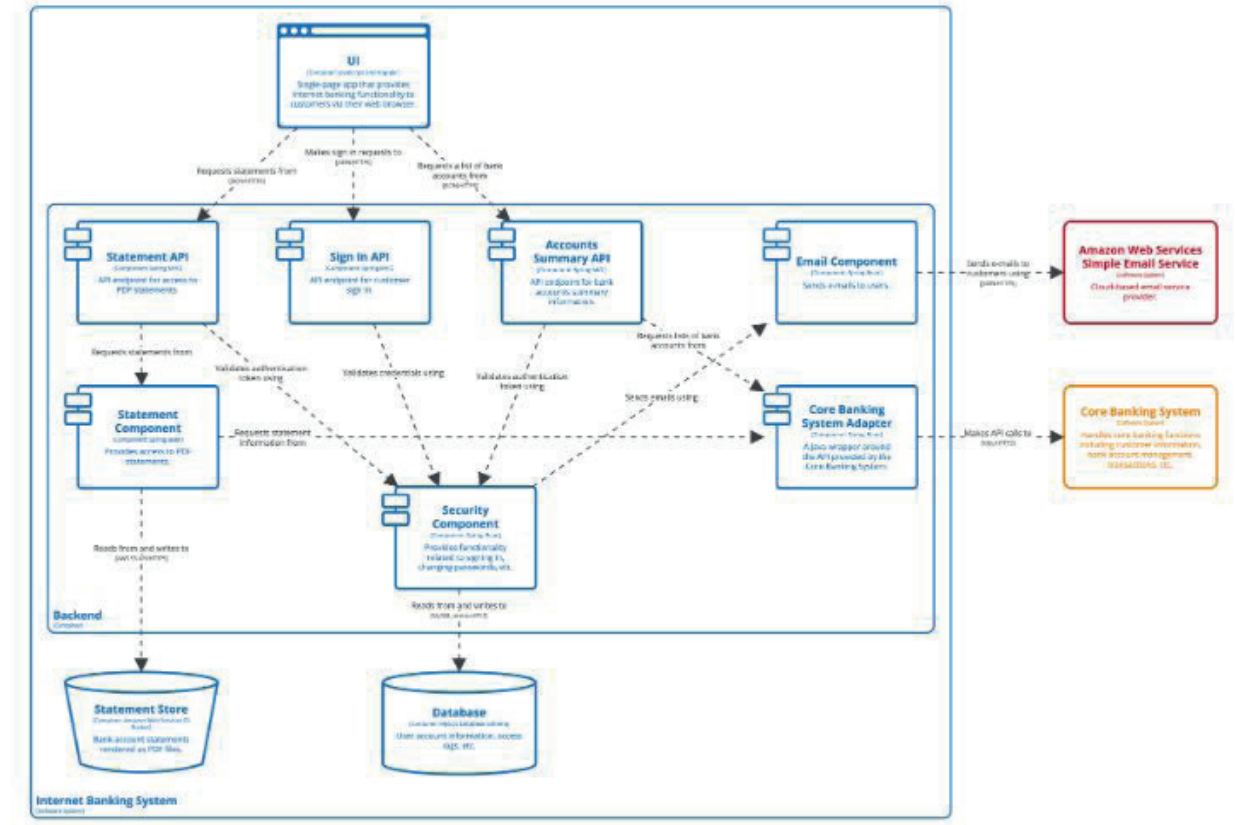
Imagem 1 - Diagrama mostrando a arquitetura do BPS



Container View: Internet Banking System

5. DIAGRAMA DOS COMPONENTES

‹‹ Detalha a composição do container para descrever os componentes que residem dentro dele; incluindo suas responsabilidades e detalhes de tecnologia/implementação. ››



Component View: Internet Banking System - Backend

Documento de Regras de Negócio Implementadas e Mensagens Apresentadas no Sistema

Este documento reúne as regras de negócio específicas a serem implementadas ou já implementadas dentro de um sistema de software, incluindo, por exemplo: condições, restrições e decisões que governam o funcionamento operacional da aplicação, bem como as mensagens e notificações que são exibidas ao usuário para comunicar estados, erros ou confirmações relacionados a essas regras. Esse documento serve como referência para garantir que o sistema opere conforme os objetivos do negócio, respeitando as normas internas, fluxos de trabalho definidos e padrões de comunicação com o usuário.

Template Sugerido:

‹ Deverá ser incluído e transcrito na ferramenta Atlassian/Jira no “espaço” do Projeto. Será apenas um documento por sistema, que deverá ser incrementado conforme sua evolução›

PADRÃO NOMENCLATURA (CADASTRO JIRA):

- Sigla do Sistema–RNG: ‹ Nome do novo projeto de software ›

‹seguir padrão de nomenclatura›

1. REGRA DE NEGÓCIO

‹ São regras lógicas e condicionais que determinam o comportamento do sistema de acordo com as necessidades do negócio. ›

Exemplo:

RNG001 - PERFIS DE ACESSO

O sistema deve apresentar os seguintes perfis de acesso:

Perfil	Cargo	Descrição
Dirigente	Diretor(a), Prefeito(a), Presidente, Reitor(a), Secretário Estadual de Ciência, Tecnologia e Inovação, Secretário(a) Estadual de Educação, Secretário Municipal de Educação	Responsáveis por solicitar habilitações para as respectivas entidades à participarem dos programas da educação
TÉCNICO FNDE	Técnico FNDE	Responsável por analisar e gerenciar as solicitações realizadas.

RNG002 - VALIDAR CNPJ

O sistema deve permitir ao usuário informar o CNPJ. Ao ser preenchido e o usuário tirar o foco do campo, o sistema deve verificar se o CNPJ é válido e se existe na base da Receita Federal, caso o CNPJ atenda aos critérios definidos, o sistema deverá preencher de forma automática e não permitir edição nos seguintes campos: Nome empresarial, Nome fantasia e Natureza jurídica.

PADRÃO NOMENCLATURA (cadastro Jira):

- RNG‹ Número Sequencial›

‹seguir padrão de nomenclatura›

xxxxxxxxxxxx

2. MENSAGENS DO SISTEMA

‹ São regras lógicas e condicionais que determinam o comportamento do sistema de acordo com as necessidades do negócio. Quando forem regras comuns aos sistemas, deverá ser observada sua padronização.›

Exemplos:

Código	Descrição	Tipo*
MSG001	Campo de preenchimento obrigatório.	Exclamativa
MSG002	Deve ser selecionada pelo menos uma escola mantida!	Alerta
MSG003	Confirma a(s) escola(s) mantida(s) selecionada(s)? (Opções de seleção: Confirmar / Cancelar)	Confirmação

(*) **Tipo**: Sucesso / Exclamativa / Alerta / Confirmação / Informativa

PADRÃO NOMENCLATURA (cadastro Jira):

- MSG‹ Número Sequencial› - xxxx

‹seguir padrão de nomenclatura›

Padranização Nomenclatura de Documentação

1. LEAN INCEPTION (LI)

- Sigla do Sistema–LI (MVP): < Nome do novo projeto de software >

2. HISTÓRIA DE USUÁRIO (HU)

- Sigla do Sistema–HU‹nº sequencial›: ‹Nome da Funcionalidade›

3. REGRA DE NEGÓCIO E MENSAGENS (RNG)

- Sigla do Sistema–RNG: < Nome do novo projeto de software >

4. SPRINT (SPT)

- Sigla Sistema-Sprint-‹nº sequencial›

5. CASO DE TESTE (CT)

- Sigla do Sistema–HU‹nº sequencial›: CT-‹ Nome do Caso de Teste›

6. EVIDÊNCIA DE TESTE (ET)

- Sigla do Sistema–HU‹nº sequencial›: ET-‹ Nome do Caso de Teste›

7. PLANEJAMENTO DE RELEASE (ENTREGAS PRODUÇÃO) (REL)

- Sigla do Sistema–REL‹versão sistema›: < Nome do novo projeto de software >

8. ARQUITETURA (ARQ)

- Sigla do Sistema–ARQ: < Nome do novo projeto de software >

Lean Inception

1. INTRODUÇÃO

1.1 O presente documento registra as atividades realizadas com a equipe da Coordenação-xxxxxxx, entre os dias dd/mm/yyyy e dd/mm/yyyy, em períodos alternados. Essas atividades foram feitas de forma ‹presencial e/ou remota› utilizando a ferramenta ‹indicar o nome da ferramenta›.

1.2 A oficina "Lean Inception", referente ao projeto "xxxxxx", foi conduzida pelos profissionais:

- Scrum Master: xxxxxxxxxx
- xxxxxxxxxxx

1.3 Foram convidados para as reuniões os senhores(as):

- Xxxxxxxxxxx – Coordenação xxxxxxxxxx
- Xxxxxxxxxxx – CGxx/xxxxx
- Xxxxxxxxxxx - ‹Função›

1.4 Abaixo estão os participantes das reuniões ocorridas:

Antônio (Gerente de Negócios)	Derick (UX Designer)	Diogo (Scrum Master)
Pedro (Desenvolvedor)	Arthur (Desenvolvedor)	Igor (Scrum Master)
Carlos Augusto (Scrum Master)	Rosana (AD)	Diogo (Scrum Master)

2. AGENDA

Lean Inception – Projeto: xxxxxxxxx			
Data: dd/mm/yyyy	**Data: dd/mm/yyyy**	**Data: dd/mm/yyyy**	**Data: dd/mm/yyyy**
Reunião de Kickoff	Personas	Brainstorming de funcionalidades	Revisão técnica (UX/UI e Negócio)
Visão do Produto	Jornada do Usuário	Funcionalidades da Jornada	Sequenciador
É-não-é / faz-não-faz			Identificação dos incrementos do negócio
Objetivo do Produto			

<< ajustar a tabela acima, conforme datas das oficinas realizadas>>

3. VISÃO DO PRODUTO

3.1 O objetivo desta etapa é definir a essência do valor de negócio do produto. Para isto, a equipe, de forma conjunta, conforme "Visão do Produto", utilizou post-its para completar a frase:

- Para [cliente final], que [problema], o [produto] é um [categoria] que [benefício-chave] diferente de [concorrência ou alternativa], nosso produto [diferença chave / diferencial].

‹ Exemplo:›

3.2 Assim, o resultado da dinâmica é apresentado conforme abaixo:

- Para Entidades públicas municipais, estaduais e federais. Fornecedores e Equipe técnica do FNDE. Órgão de Controle e sociedade,
- que ...utilizam, aderem, gerenciam e consultam as atas de registro de preços dos pregões de Registo de Preços Nacional do FNDE,
- o ...SIGARP
- é um ... Sistema WEB
- que ...Atende os normativos e legislações vigentes para gerenciar as atas de registros de preços do RPN, de forma unificada com os produtos digitais de governo,
- diferente de ... do atual SIGARP,
- nosso produto .....

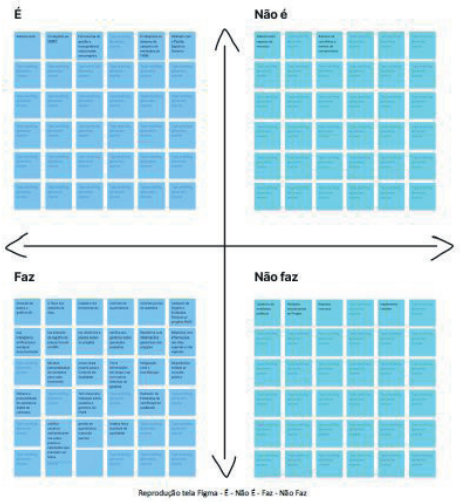
4. É, NÃO-É, FAZ, NÃO-FAZ

‹descreve o propósito, objetivos e valor do produto a ser desenvolvido.›
‹define claramente o que o produto é, o que não é, o que faz e o que não faz.›

4.1 De acordo com o autor, Paulo Caroli:

"Muitas vezes é mais fácil descrever o que alguma coisa não é ou não faz. A atividade É-Não é-Faz-Não faz (ENFN, abreviado) busca classificações sobre o produto seguindo as quatro diretrizes, indagando, especificamente, cada aspecto positivo e negativo sobre o produto ser ou fazer algo."

‹‹ demonstrar a evidência da atividade – fotos e/ou print imagem (ex: Figma ou Miro)



Reprodução tela Figma - É - Não É - Faz - Não Faz

É	NÃO É

FAZ	NÃO FAZ

5. OBJETIVOS DO PRODUTO

‹Refere-se ao processo de identificar, ordenar e focar nos objetivos estratégicos do produto que são mais importantes para o sucesso do MVP e do negócio. Esses objetivos são compartilhados e discutidos entre todos os participantes, garantindo alinhamento quanto às metas fundamentais que o produto deve alcançar.›

Identificador	Objetivos Estratégicos (MVP)
Ex: 001	*Armazenar, tratar e disponibilizar informações estratégicas sobre o RPN;*

6. PRINCIPAIS USUÁRIOS (PERSONAS):

6.1 Uma persona representa um usuário do produto ou serviço. Essa atividade objetiva, portanto, identificar os usuários do produto ou serviço, com a descrição do seu papel e das suas necessidades específicas. A ideia é despertar uma representação realista de usuários, a fim de auxiliar o time a descrever funcionalidades do ponto de vista de quem realmente irá interagir com o produto final.

6.2 As personas obtidas durante a oficina foram:

‹Principais personas e perfil associado que utilizarão e/ou utilizam o produto, indicar as necessidades e características gerais de cada perfil.›

Persona 1	
Nome ou Apelido	**Perfil**
Comportamento	**Necessidades e Características Gerais**

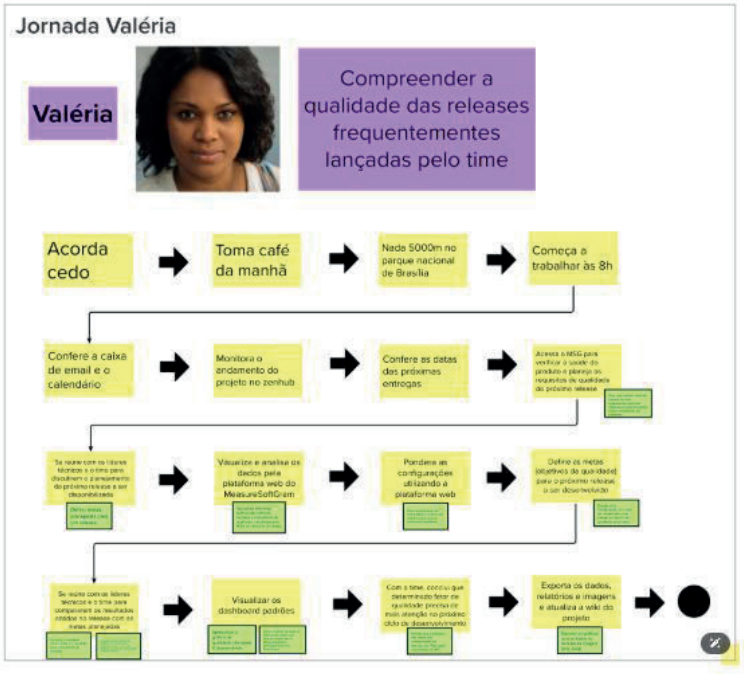
Persona 2	
Nome ou Apelido	**Perfil**
Comportamento	**Necessidades e Características Gerais**

7. JORNADAS DO USUÁRIO

‹ A jornada do usuário descreve uma sequência de atividades e passos que o usuário realiza para completar determinada ação e alcançar um objetivo específico. Nesta atividade, esta jornada deve ser detalhada de acordo com o contexto de uso do produto idealizado baseado na trajetória das personas criadas. Deve ser criado uma jornada para cada persona.›

‹‹ ferramenta de apoio a ser definida pelo FNDE – ex: Miro, Canva, Bizagi››

Exemplo:



8. BRAINSTORMING DE FUNCIONALIDADES:

‹Lista inicial das funcionalidades propostas para o produto nas jornadas›

8.1 Esta atividade objetiva listar as funcionalidades (“features”) necessárias, para que o produto atenda às necessidades dos usuários.

8.2 Busca responder: O que deverá ter no produto para atender às necessidades das personas? Quais funcionalidades devemos construir para atingir o objetivo do produto?

Identificador	*Funcionalidades Iniciais da Jornada*
001	

9. REVISÃO TÉCNICA, DE UX E DE NEGÓCIO

9.1 Análise das funcionalidades propostas quanto à viabilidade, experiência do usuário e impacto no negócio.

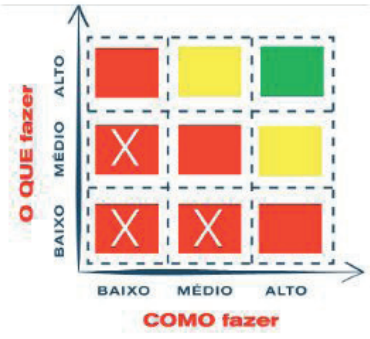
9.2 Nesta etapa cada funcionalidade é avaliada de acordo com 4 critérios: esforço, valor de negócio, valor de UX e o nível de confiança sobre o que é a funcionalidade e como construí-la.

Negócio	$	$$	$$$	**Valor para o Negócio**
UX	♥	♥ ♥	♥ ♥ ♥	**Valor para o Usuário**
Esforço	E	EE	EEE	**Esforço**

9.3 Para o nível de confiança, considera-se “o que fazer” e “como fazer”. São atribuídas cores, de acordo com o “gráfico Semáforo”.

9.4 As cores representam:

Verde: pode ir tranquilo
Amarelo: prestar atenção
Vermelho: para e espere antes de prosseguir



9.5 Dentre os requisitos identificados e elencados anteriormente, alguns são "não funcionais". Eles são referentes a desempenho, usabilidade, segurança e não são considerados na oficina de "Lean Inception". No entanto, é importante o registro de quais são:

‹ exemplo:›

- Intuitivo
- Responsivo

9.6 O resultado da revisão técnica, de negócio e de UX é demonstrado abaixo:

Funcionalidade	Nível de Confiança*	Esforço	Valor de Negócio

< exemplo:>

Funcionalidade	Nível de Confiança*	Esforço	Valor de Negócio
Cadastro dos tipos de demandas (elogio, sugestão, esclarecimento e reclamação).	Verde	E	$$$
Registro de dados de usuários – perfil de acesso, nomes, ativação.	Vermelho	EEE	$$$
Registro dos contratos firmados (número contrato, empresa, ano, vigência, objeto, representante)	Amarelo	EE	$$

(*) resultado da análise do "gráfico do semáforo".

10. PLANO DE ENTREGA INCREMENTAL DO PRODUTO – SEQUENCIADOR DO MVP

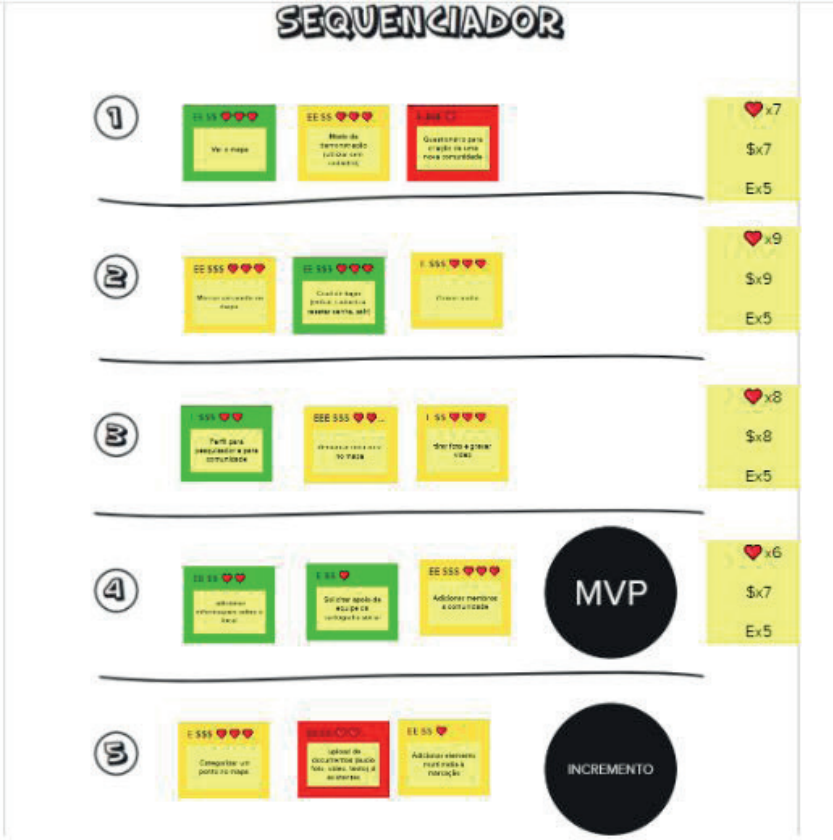
‹‹ Sequenciador de MVP: indica a priorização e ordenação das funcionalidades para construir o Produto Mínimo Viável.››

10.1 Dadas as funcionalidades e sua classificação, deve-se agora priorizá-las. O objetivo é executar primeiro a funcionalidade mais impactante, para que atinja objetivos o mais cedo

possível. Uma pergunta norteadora é “Qual é a combinação mínima de funcionalidades que deve ser disponibilizada para validar um pequeno conjunto de hipóteses sobre o negócio?”.

10.2 Essa priorização é feita por meio da atividade denominada “Sequenciador”, na qual deve ser obedecidas as seguintes regras:

1. Uma onda pode conter, no máximo, 3 cartões;
2. Uma onda não pode conter mais de 1 cartão vermelho;
3. Uma onda não pode conter 3 cartões somente amarelos ou vermelhos;
4. A soma de “esforço” dos cartões deve ser menor que 5 “E”;
5. A soma de “valor de negócio” dos cartões não pode ser menos de 4 “$” e a soma de “UX” não pode ser menos de 4 ;
6. Se um cartão depende de outro, esse outro deve estar em alguma onda anterior.



10.3 Sequenciador de Funcionalidades:

Sequenciador	Funcionalidades MVP
1	Cadastro de xxxxxxxxxxx
	Registro xxxxxxxxxx
	Consultar xxxxxxxxxx
2	Cadastro de xxxxxxxxxxx
	Registro xxxxxxxxxx
	Consultar xxxxxxxxxx
...	

11. CANVAS

11.1 Canvas do MVP é uma síntese visual que resume os elementos essenciais do MVP, seus critérios de sucesso e métricas.

Proposta do MVP – Qual é a proposta deste MVP?

1. Personas segmentadas – Para quem é esse MVP? Podemos segmentar e testar este MVP em um grupo menor?
2. Jornadas – Quais jornadas são atendidas ou melhoradas com este MVP?
3. Funcionalidades – O que vamos construir neste MVP? Que ações serão simplificadas ou melhoradas neste MVP?
4. Resultado esperado – Que aprendizado ou resultado estamos buscando neste MVP?
5. Métricas para validar as hipóteses do negócio – Como podemos medir os resultados deste MVP?
6. Custo & Cronograma – Qual é o custo e a data prevista para a entrega deste MVP? Depois de entregue, quanto tempo precisamos coletar os dados para decidir se pivotamos ou prosseguimos?

11.2 Canvas do MVP é uma síntese visual que resume os elementos essenciais do MVP, seus critérios de sucesso e métricas.

‹‹ ferramenta de apoio a ser definida pelo FNDE – ex: Miro››

Modelo Canvas Lean Inception do MVP:

Matriz RACI

MATRIZ RACI									
N/A - não se aplica Metodologia de Desenvolvimento de Sistema (MDS) FNDE Ágil Novos projetos / Demandas Evolutivas	CGSD	Dono do Produto (PO)	Dono do Produto (PO)	Scrum Master	Analista de Negócio / Requisito	Analista UX / UI	Arquitetura de Software	Desenvolvedor	Analista de Teste / Qualidade
Levantamento e Detalhamento da Solução									
Lean Inception (LI)	P	R / A	R	P	P	I	I	I	I
Backlog do Produto	I	R / A	P	P	P	I	I	I	I
Documento de Visão (DV)	I	P / A	P	R	P	I	I	I	I
Documento de Arquitetura	C	I	I	C	I	R	P	P	I
História do Usuário (HU)	I	C / A	I	R	P	I	I	C	I
Regra de Negócio e Mensagens (RNG)	I	C / A	I	R	P	C	C	C	I
Protótipo Funcional (PTF)	I	A	I	P	R	I	I	C	I
Código-Fonte	I	I	I	C	C	C	R	P	I
Teste Interno (TI - não envolve usuário)									
Caso de teste (CT)	I	N/A	I	C	C	C	C	I	R
Evidência de Teste (ET)	I	N/A	I	C	C	C	C	I	R
Realease (Liberação)									
Planejamento de Realease (REL)	I	A	R	I	I	I	P	I	P