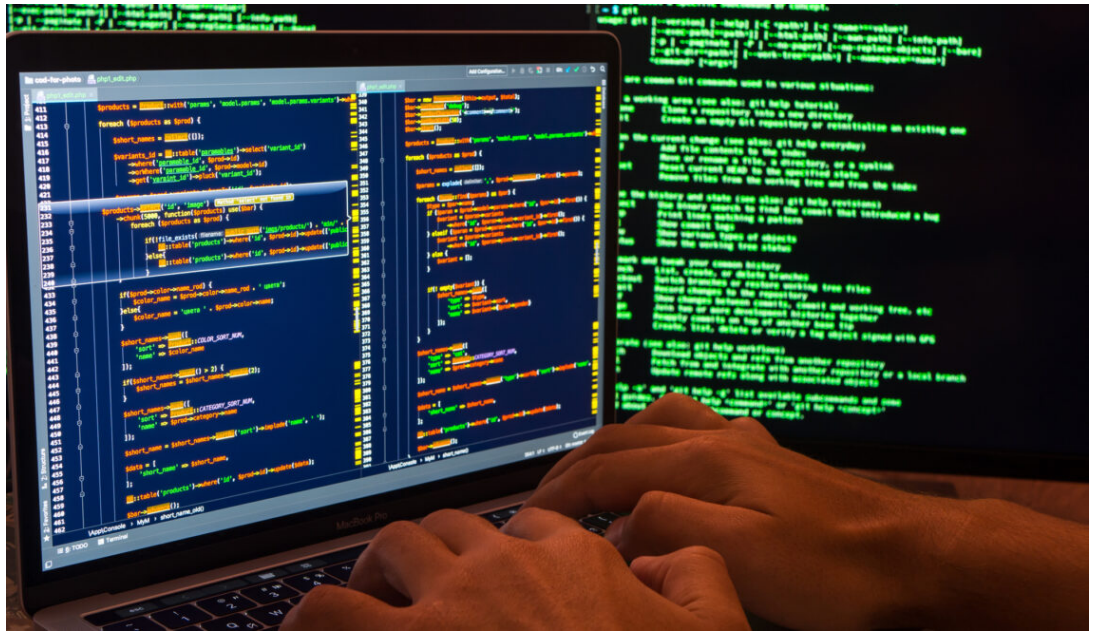


Semana 3 – uc01 - TDS

DANIEL GARCIA

DÚVIDAS E CRONOGRAMA DA AULA

- GRUPO WHATS
- ATIVIDADES 1 E 2
- PROJETO INTEGRADOR UC05
- METODOLOGIAS DE DESENVOLVIMENTO DE SOFTWARE
- INTERACÕES NA BLACKBOARD (cronograma, envio de atividades, fale

Restrições de software

As restrições de um projeto de software são limitações que podem afetar a execução, o planejamento e o resultado final do projeto

Resumindo: se adequar a realidade do cliente

Exemplos de restrições

- Restrições de tempo
- Restrições de orçamento
- Restrições de hardware
- Restrições de tecnologias/software
- Restrições de segurança
- Restrições legais

Exemplos de restrições - tempo

- **Projeto de Software para Evento Específico**: Uma empresa precisa desenvolver um aplicativo móvel para um evento esportivo que acontecerá em 6 meses. O prazo é imutável, pois o software precisa estar operacional antes do evento começar.

Exemplos de restrições - recursos

- **Compatibilidade com Sistemas Operacionais Antigos**: Um novo software de edição de vídeo quer atingir o maior número possível de usuários, mas tem recursos limitados para testar e garantir a compatibilidade com versões mais antigas de sistemas operacionais.

Exemplos de restrições - requisitos

- **Aplicativo de Mensagem Instantânea**: Um novo aplicativo de mensagens precisa funcionar tanto em conexões de internet de alta velocidade quanto em áreas com conexão lenta e instável, o que restringe o tamanho e a complexidade das mensagens e mídias que podem ser enviadas.

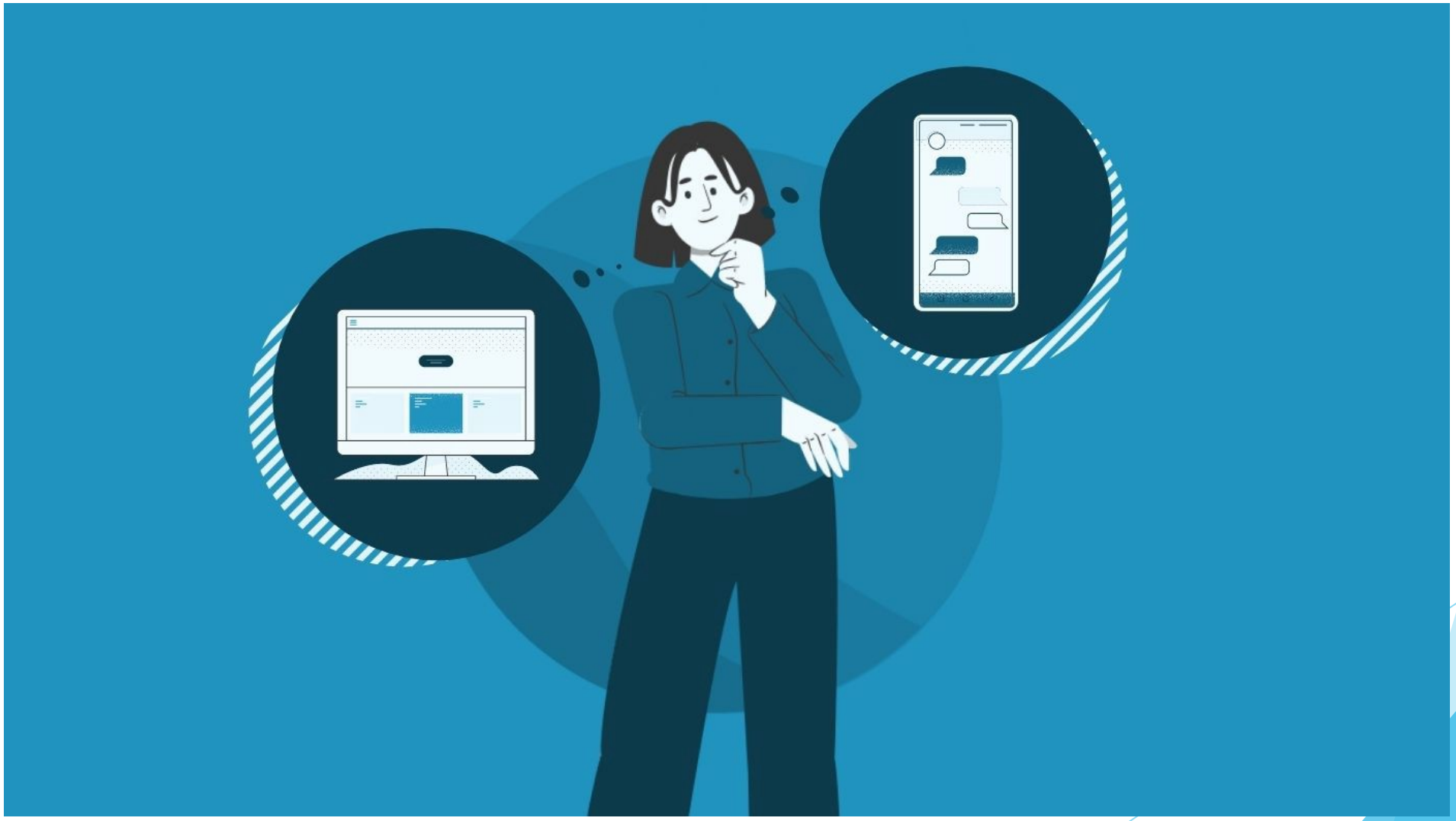
Exemplos de restrições - tecnologia

- Sistema deve funcionar no navegador Chrome.
- Sistema deve funcionar no navegador Mozilla.
- Sistema desktop deve funcionar no EDGE.
- Sistema deve funcionar nos 3.

Exemplos de restrições - tecnologia

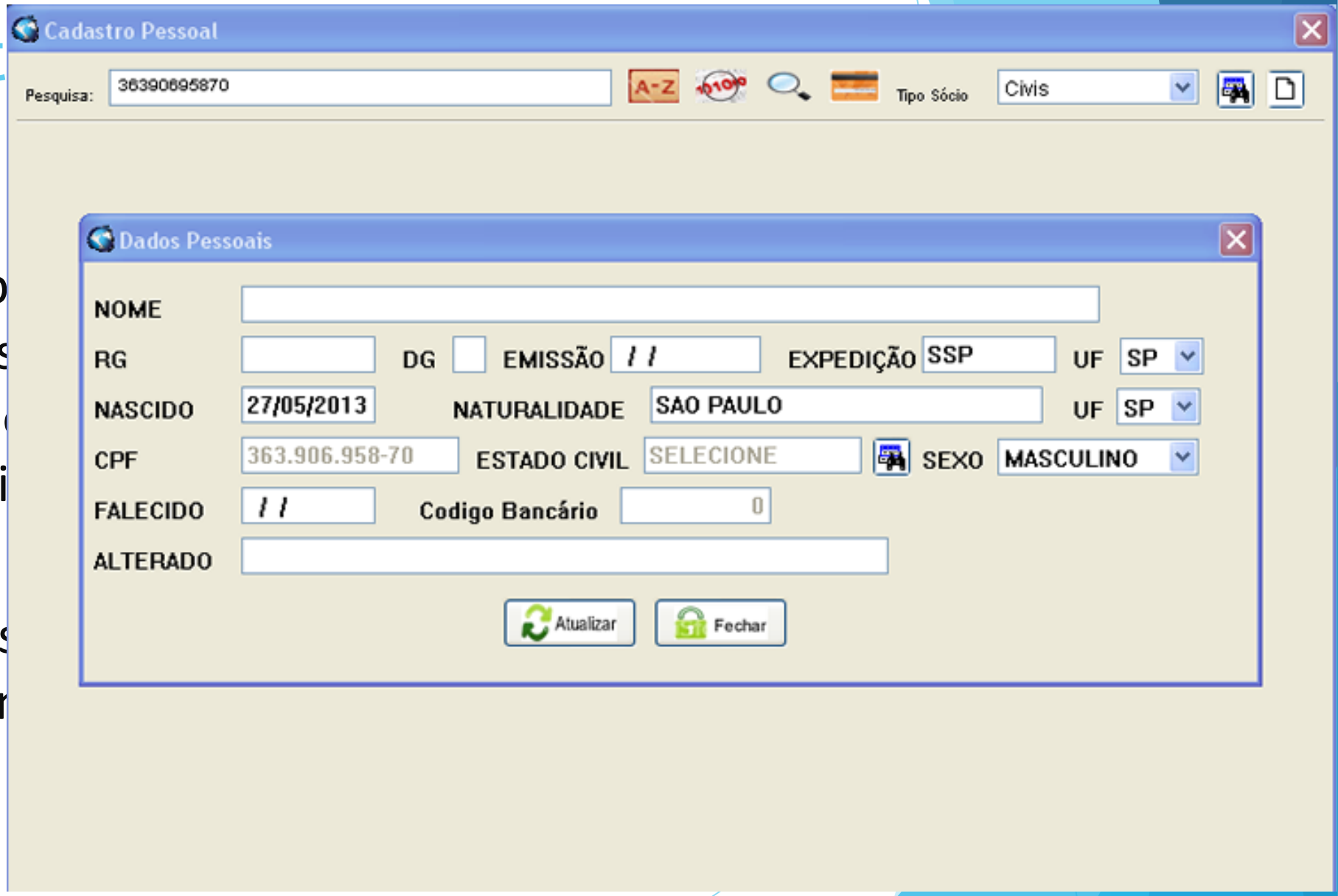
- a linguagem utilizada deve ser Java

Tipos de sistemas



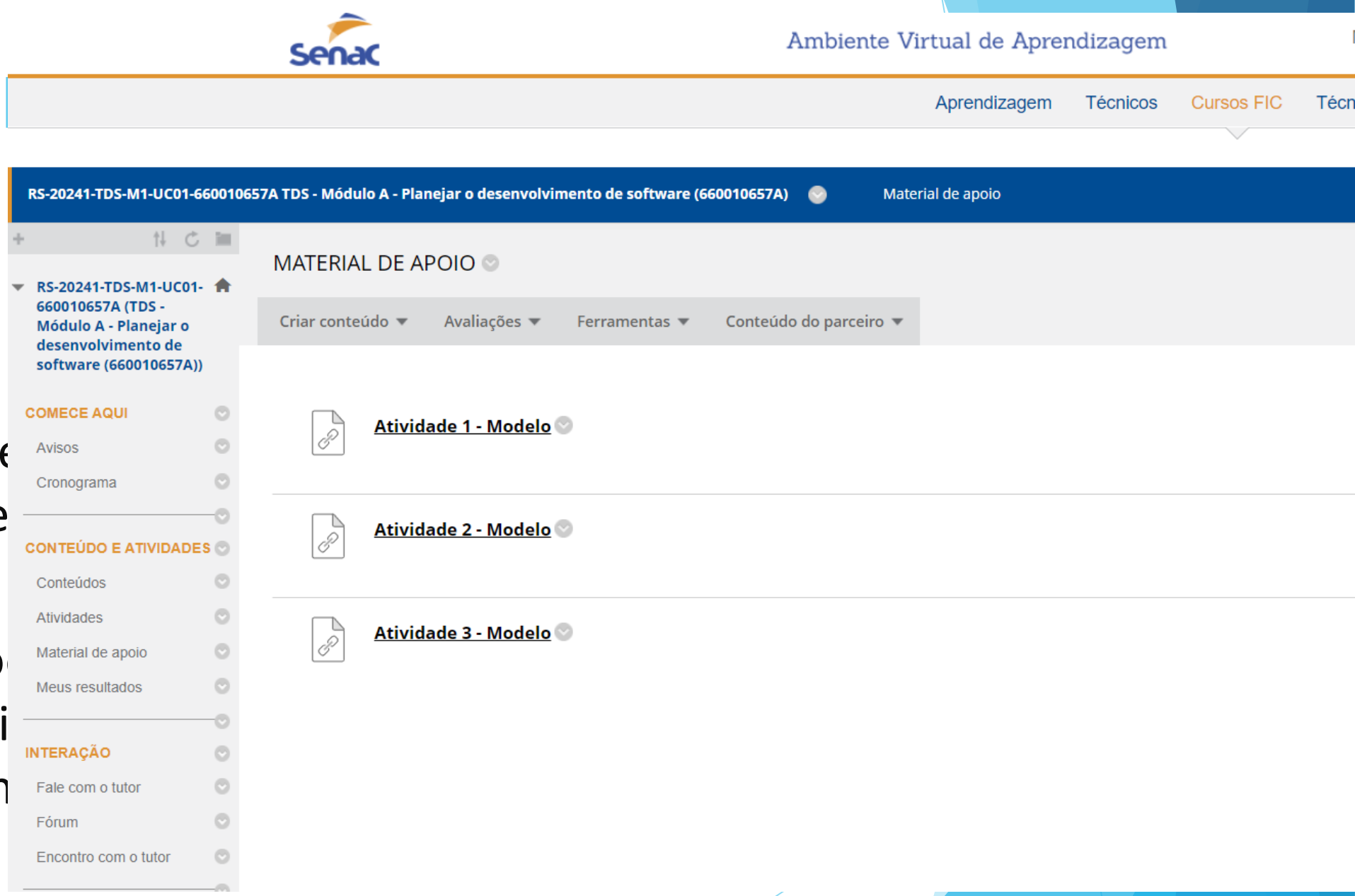
Sistema deskt

- Sistemas desktop
  diretamente em s
  Eles são projetad
  operacional do di
  Linux.
- **Exemplos**: Micros
  Photoshop. Siste



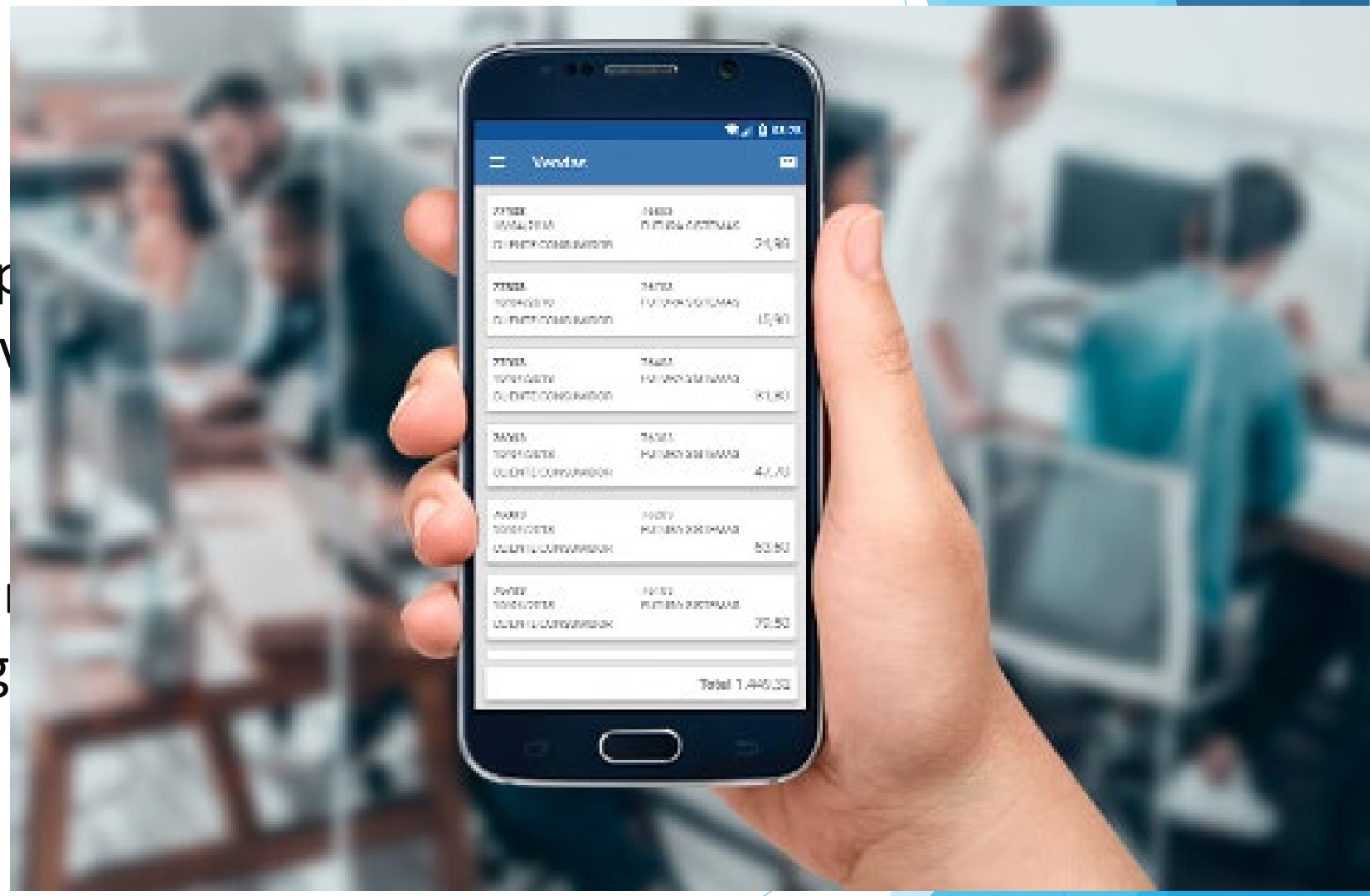
Sistema we

- Sistemas web
  navegador de
  Eles não requ
  e são acessíve
  internet
- **Exemplos**: Go
  internet banki
  como Amazon

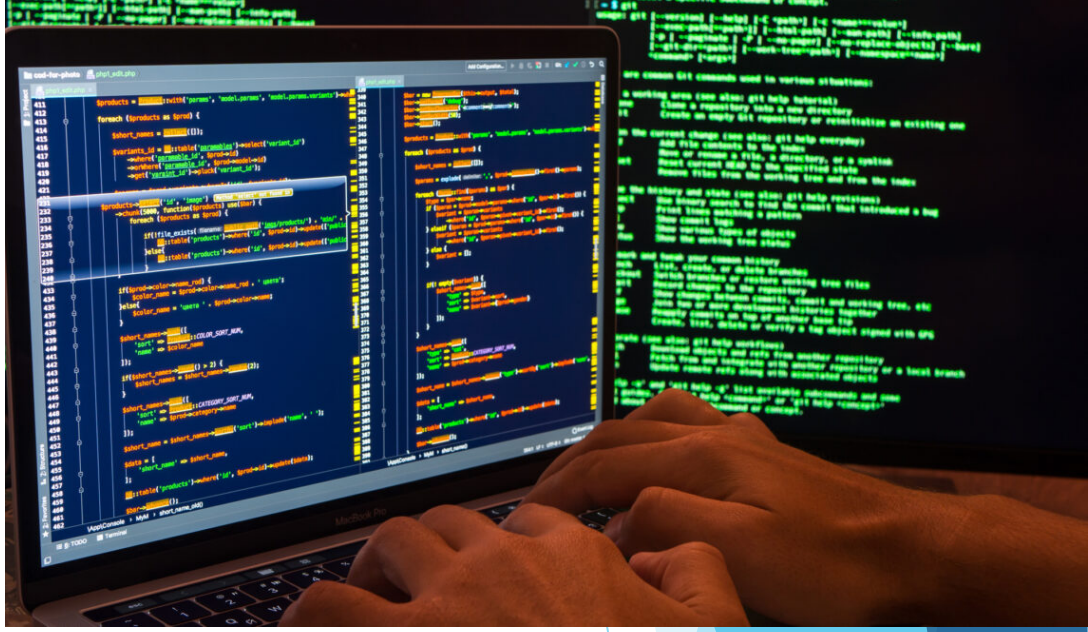


Sistema mobile

- Sistemas mobile são ap
  funcionar em dispositiv
  tablets.
- **Exemplos**: WhatsApp,
  como o do Banco do Br
  como Candy Crush Sag

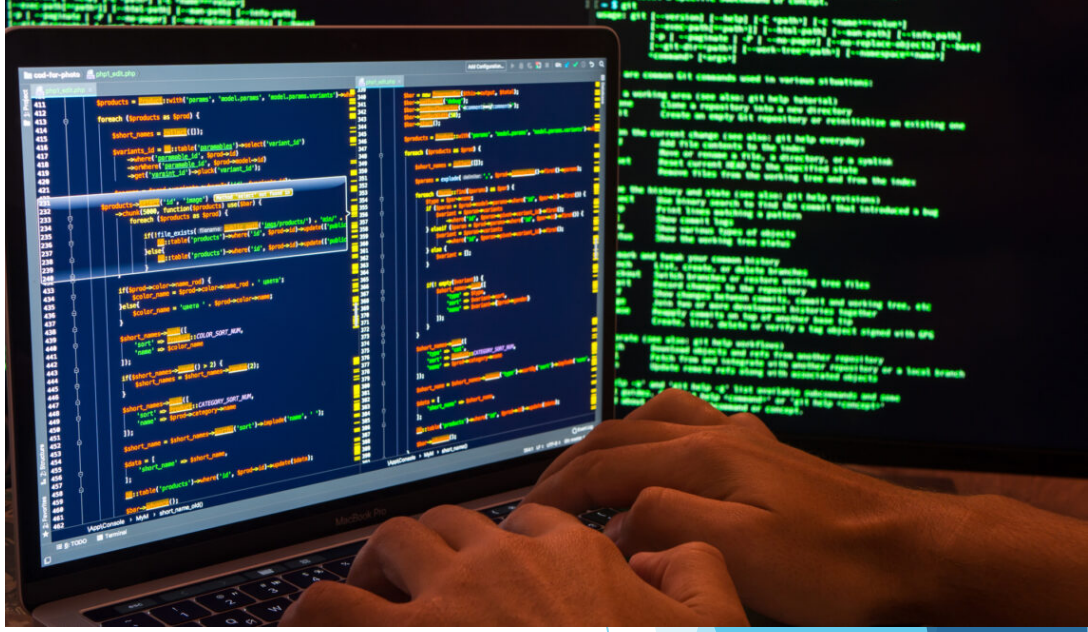


Linguagens de programação



- Desktop= qual usar
- Web = qual usar
- Mobile = qual usar

IDE



- Escolhida a linguagem qual IDE utilizar

A sigla IDE significa (Integrated Development Environment). Um ambiente de desenvolvimento integrado (IDE) é um software para criar aplicações

LINGUAGEM/BIBLIOTECAS/TESTES E ETC

Hardware

Parte física

do computador

Software

Parte lógica

do computador

Algoritmo para Cadastrar um Produto em um Site

1. Faça login no painel de administração do site com seu nome de usuário e senha.
2. No painel de administração, localize e clique na opção "Produtos" ou "Gerenciar Produtos". Em seguida, clique no botão "Adicionar Produto" ou "Novo Produto".
3. Digite as informações necessárias sobre o produto em cada campo correspondente. Isso geralmente inclui:
4. Após verificar se todas as informações estão corretas e completas, clique no botão "Salvar", "Publicar" ou similar para adicionar o produto ao site.

METODOLOGIAS DE DESENVOLVIMENTO DE SOFTWARE

- Uma metodologia de desenvolvimento de software é um conjunto de processos, práticas e regras usadas para planejar, criar, testar e entregar um software. Imagine que você está construindo uma casa. Antes de começar, você precisa de um plano sobre como vai construí-la, quais materiais vai usar, como os diferentes profissionais vão trabalhar juntos e em que ordem as coisas vão ser feitas. Da mesma forma, uma metodologia de desenvolvimento de software ajuda as equipes de desenvolvedores a organizar como eles vão trabalhar no projeto de software, desde a ideia inicial até o produto final ser entregue aos usuários.

Metodologia tradicional

Modelo cascata

Levantamento de requisitos

- Nessa etapa, a equipe de desenvolvimento definirá, junto ao cliente, as expectativas para o projeto, montando o escopo de requisitos. É fundamental a excelência nesse momento, para que o projeto cumpra com as expectativas do cliente, sendo um projeto de sucesso.

Projeto

- Com a lista de requisitos em mãos, você entrará na segunda etapa: o projeto para desenvolvimento do *software*. É nesse momento que você planejará como serão as próximas etapas, criando o cronograma do desenvolvimento do projeto, a definição escrita do escopo, a estimativa para cada etapa, a montagem do time de desenvolvimento e, em alguns casos, a modelagem da arquitetura, dos protótipos e das estruturas para o *software*.

Implementação

- A etapa de implementação é a etapa de desenvolvimento do *software*. Baseado nos requisitos e nas atividades montadas na etapa de projeto, a equipe desenvolverá todo o projeto. A duração dessa fase está diretamente ligada à quantidade de pessoas na equipe e o conhecimento técnico desta.

Testes

- Com o código totalmente desenvolvido, a equipe testará o *software*, validando o resultado da etapa de implementação e garantindo que o desenvolvimento tenha cumprido o objetivo do projeto.
- Quaisquer falhas encontradas nesse momento deverão ser corrigidas pela equipe de desenvolvimento antes de partir para a próxima etapa.

Manutenção

- A etapa de manutenção do sistema envolve a implantação do *software* para o cliente, sendo definida como a entrega do projeto e, em alguns casos, a manutenção contínua. Entretanto, no modelo cascata não há espaço para novos requisitos no escopo, sendo necessário o reinício desse modelo. Nesse modelo, a manutenção reflete apenas suportar aquilo que já foi desenvolvido e entregue.

Vantagens:

Simplicidade e Facilidade de Entendimento: A natureza sequencial da Cascata torna-a fácil de entender e gerenciar, com fases claramente definidas e um processo linear.

Planejamento Detalhado: Permite um planejamento e design abrangentes no início do projeto, o que pode ajudar a identificar problemas antecipadamente.

Mudanças Minimizadas: Como as mudanças são desencorajadas uma vez que o projeto avança para a próxima fase, isso pode levar a uma maior estabilidade do projeto e dos requisitos.

Desvantagens:

Flexibilidade Limitada: A dificuldade em acomodar mudanças nos requisitos após o início do projeto pode resultar em um produto final que não atende às necessidades atuais do usuário.

Entrega Tardia: O produto ou sistema é entregue apenas ao final do projeto, o que pode atrasar a detecção de problemas ou a realização de ajustes necessários.

Risco Maior: Se um erro é descoberto nas fases finais, pode ser muito custoso e demorado para corrigir, aumentando o risco do projeto.

Metodologia Ágil

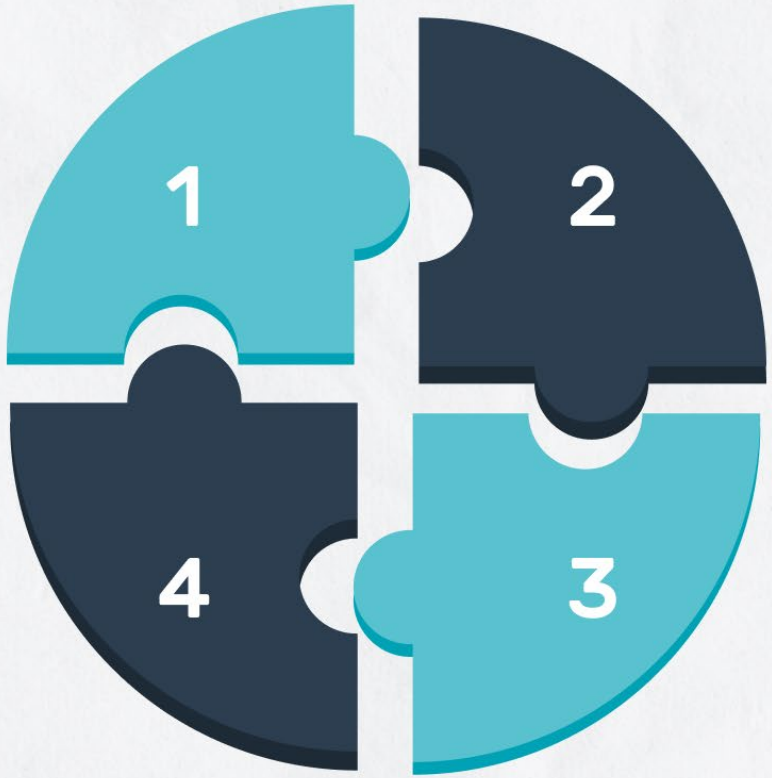
Manifesto ágil

- O Manifesto Ágil é uma declaração de valores e princípios essenciais para o desenvolvimento de software.

- Um documento escrito inicialmente em 2001, entre 17 pessoas que já praticavam métodos ágeis, um guia sobre metodologias ágeis

MANIFESTO ÁGIL

Indivíduos e interação
mais que processos e ferramentas

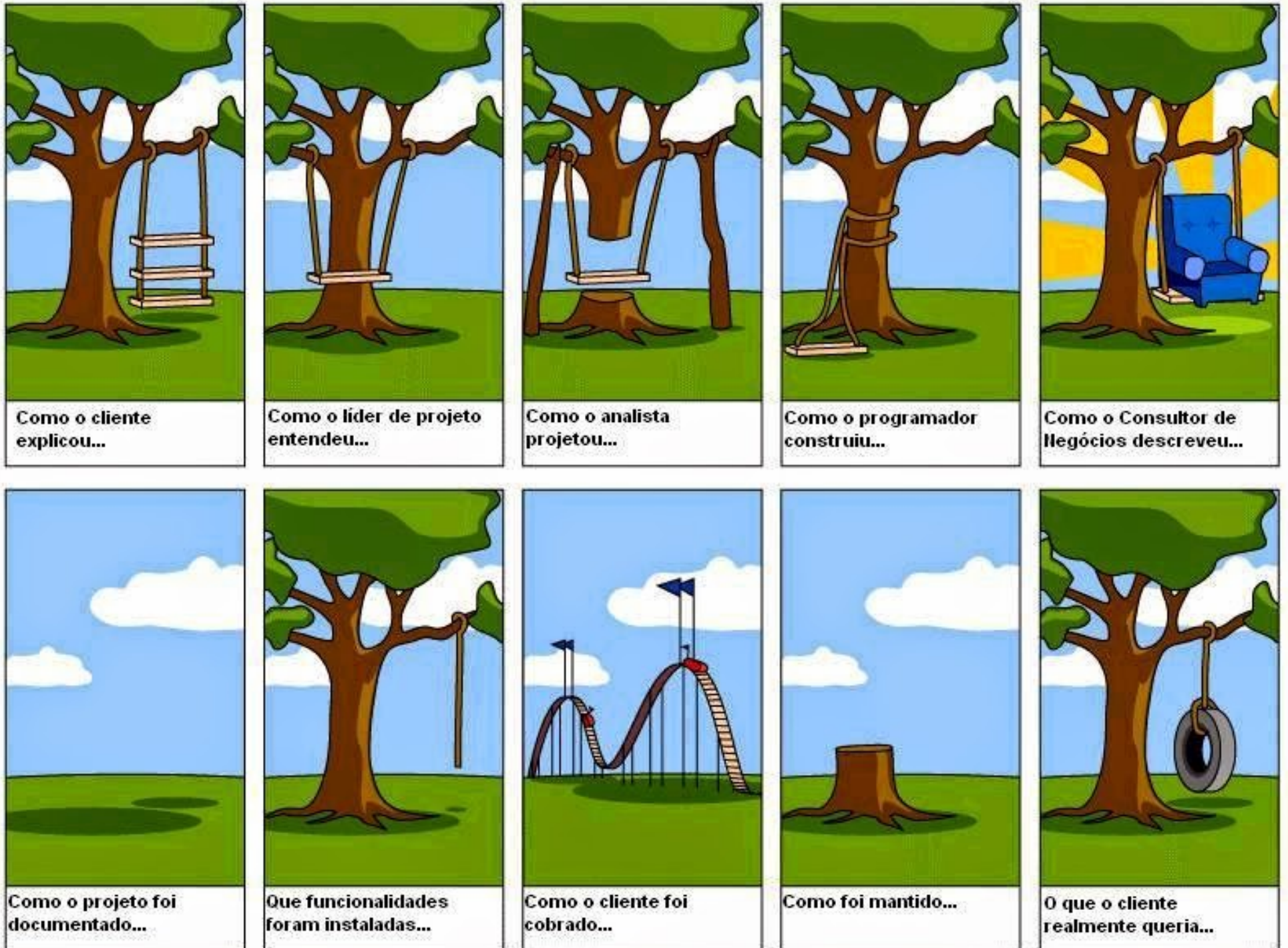


Software em funcionamento
mais que documentação abrangente

Responder a mudanças
mais que seguir um plano

Colaboração do cliente
mais que negociação de contratos

Quais são os 12 princípios do Manifesto Ágil?


Como o cliente explicou...
Como o líder de projeto entendeu...
Como o analista projetou...
Como o programador construiu...
Como o Consultor de Negócios descreveu...
Como o projeto foi documentado...
Que funcionalidades foram instaladas...
Como o cliente foi cobrado...
Como foi mantido...
O que o cliente realmente queria...

MÉTODOS ÁGEIS

- É uma coleção de metodologias baseada na prática para modelagem efetiva de sistemas baseados em software. É uma filosofia onde muitas metodologias se encaixam.

- As metodologias ágeis aplicam uma coleção de práticas, guiadas por princípios e valores que podem ser aplicadas por profissionais de software no dia a dia.

O QUE SÃO OS MODELOS ÁGEIS

Um modelo ágil é um modelo bom o suficiente, nada mais, o que implica que ele exibe as seguintes características

1. **Ele** atende seu propósito
2. **Ele** é inteligível
3. **Ele** é suficientemente preciso
4. **Ele** é suficientemente consistente
5. **Ele** é suficientemente detalhado
6. **Ele** provê um valor positivo
7. **Ele** é tão simples quanto possível

O QUE É (E NÃO É) MÉTODOS ÁGEIS

1 É uma atitude, não um processo prescritivo

2 É um suplemento aos métodos existentes, ele não é uma metodologia completa

3 É uma forma efetiva de se trabalhar um conjunto para atingir as necessidades das partes interessadas no projeto

4 É uma coisa que funciona na prática, não é uma teoria acadêmica

O QUE É (E NÃO É) MÉTODOS ÁGEIS (CONT)

5 Não é um ataque à documentação, pelo contrário aconselha a criação de documentos que tem valor

METODOLOGIAS ÁGEIS

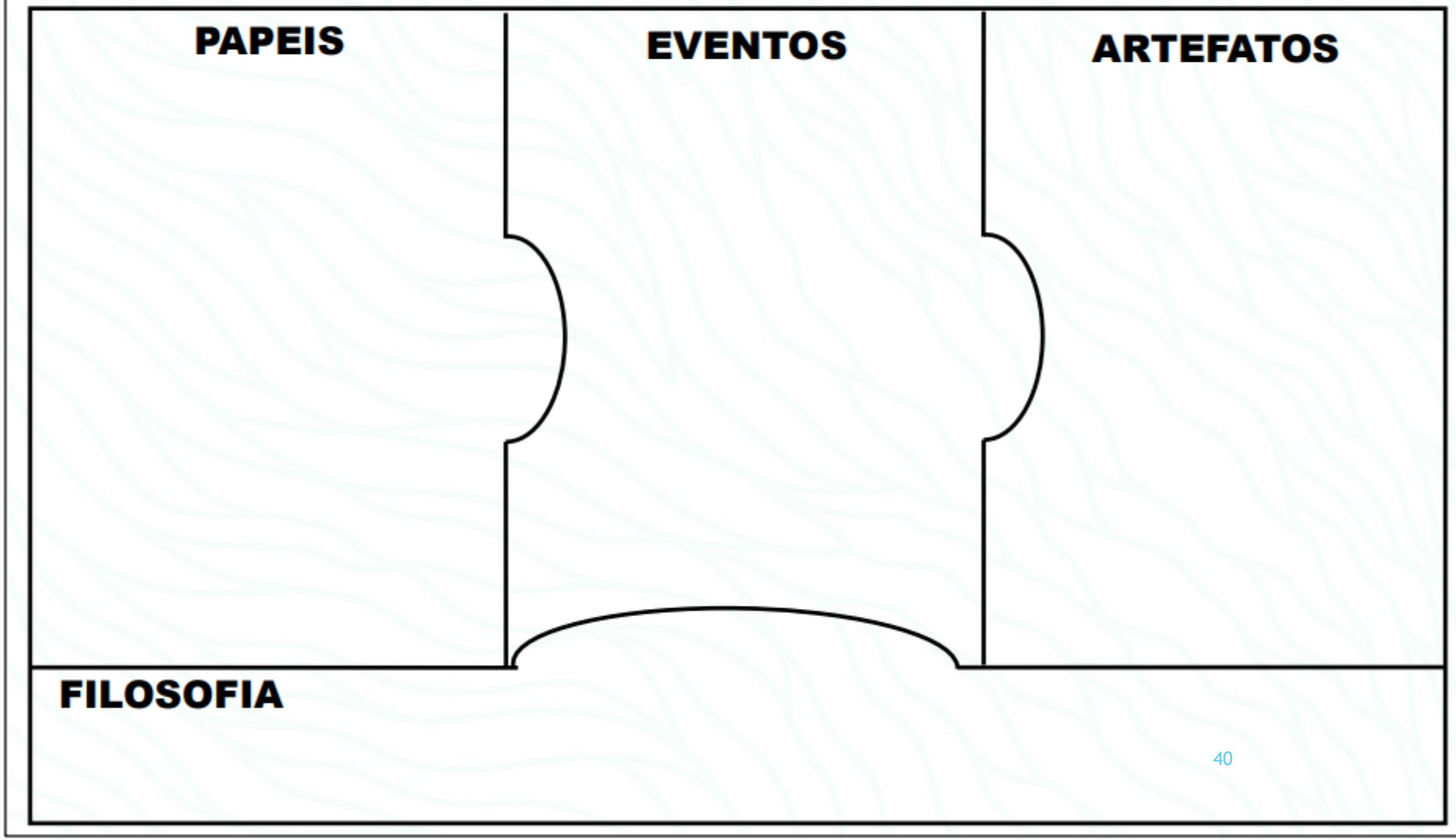
- SCRUM
- eXtreme Programming (XP)
- LEAN
- KanBan

SCRUM

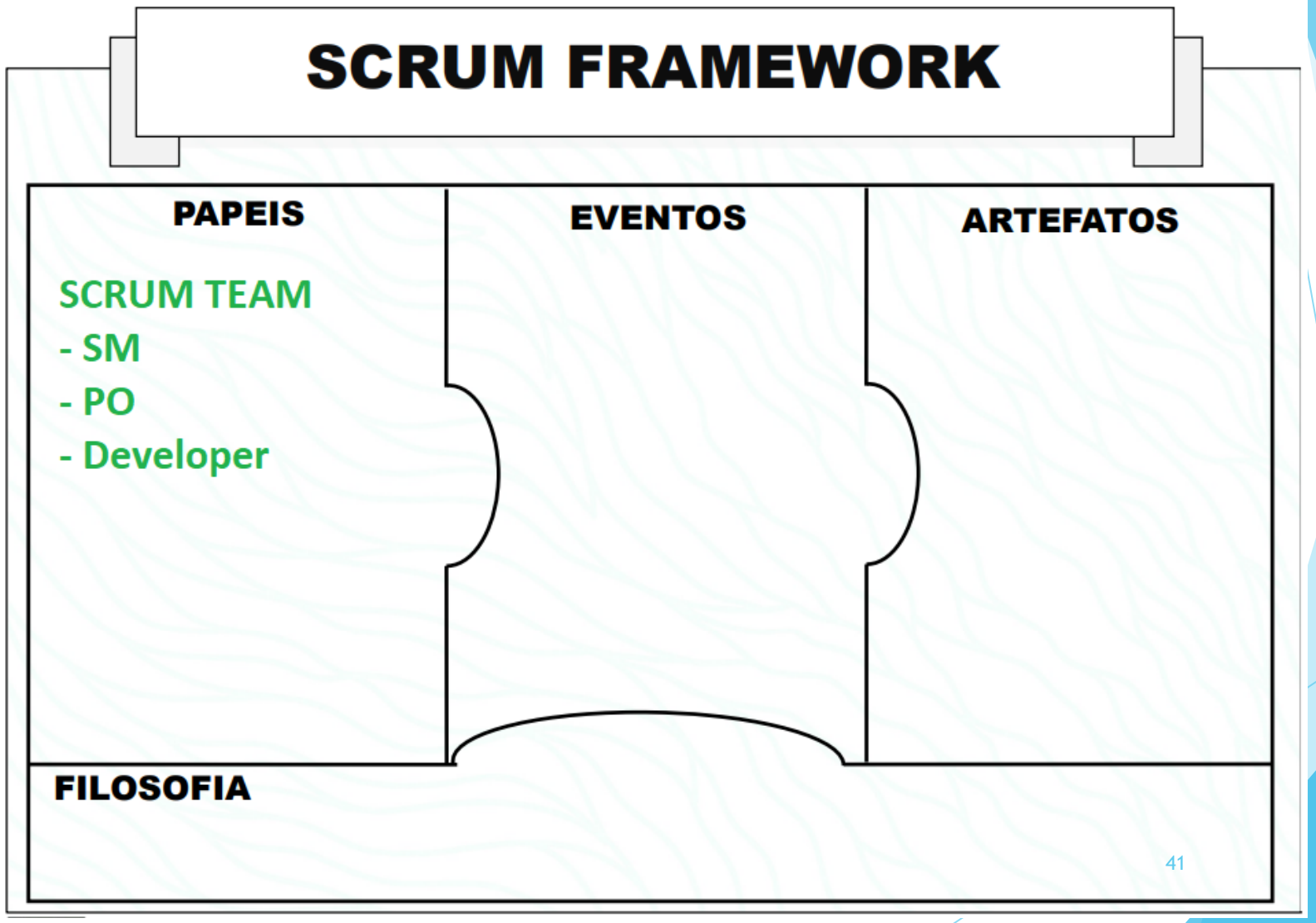
- É um processo para construir software incrementalmente em ambientes complexos, onde os requisitos não são claros ou mudam com muita frequência
- Scrum é um método ágil
- Equipes pequenas, requisitos pouco estáveis ou desconhecidos e iterações curtas para promover visibilidade para o desenvolvimento

SCRUM é um framework leve que ajuda pessoas, times e organizações a gerar valor por meio de soluções adaptativas para problemas complexos

SCRUM FRAMEWORK



PAPEIS


SCRUM FRAMEWORK
PAPEIS
EVENTOS
ARTEFATOS
SCRUM TEAM
- SM
- PO
- Developer
FILOSOFIA

SCRUM TEAM

- **Equipe com no máximo 10 pessoas**
  - Objetivo: a meta do produto dentro de um prazo estipulado
  - Membros
    - **Scrum Master** (1 pessoa – gerenciar as pessoas e processos)
    - **Product Owner** (1 pessoa – regras de negócio)
    - **Developers** (até 8 pessoas - controí)
  - Não há sub-times ou hierarquia
  - Todo o Scrum Team é responsável por criar um incremento valioso e útil a cada sprint.

Developers

- São as pessoas do Scrum Team que estão comprometidas em criar qualquer aspecto de incremento utilizável a cada sprint.
  - Criar o plano para a Sprint, o Sprint Backlog
  - Constrói o produto com qualidade, de forma correta, fazem estimativas de tamanho e tempo
  - Precisam ter skill necessárias para a construção com qualidade

- Quem faz parte?
  - Desenvolvedores
  - Designers
  - Testadores
  - Qualquer outro profissional que desenvolverá o projeto no dia a dia

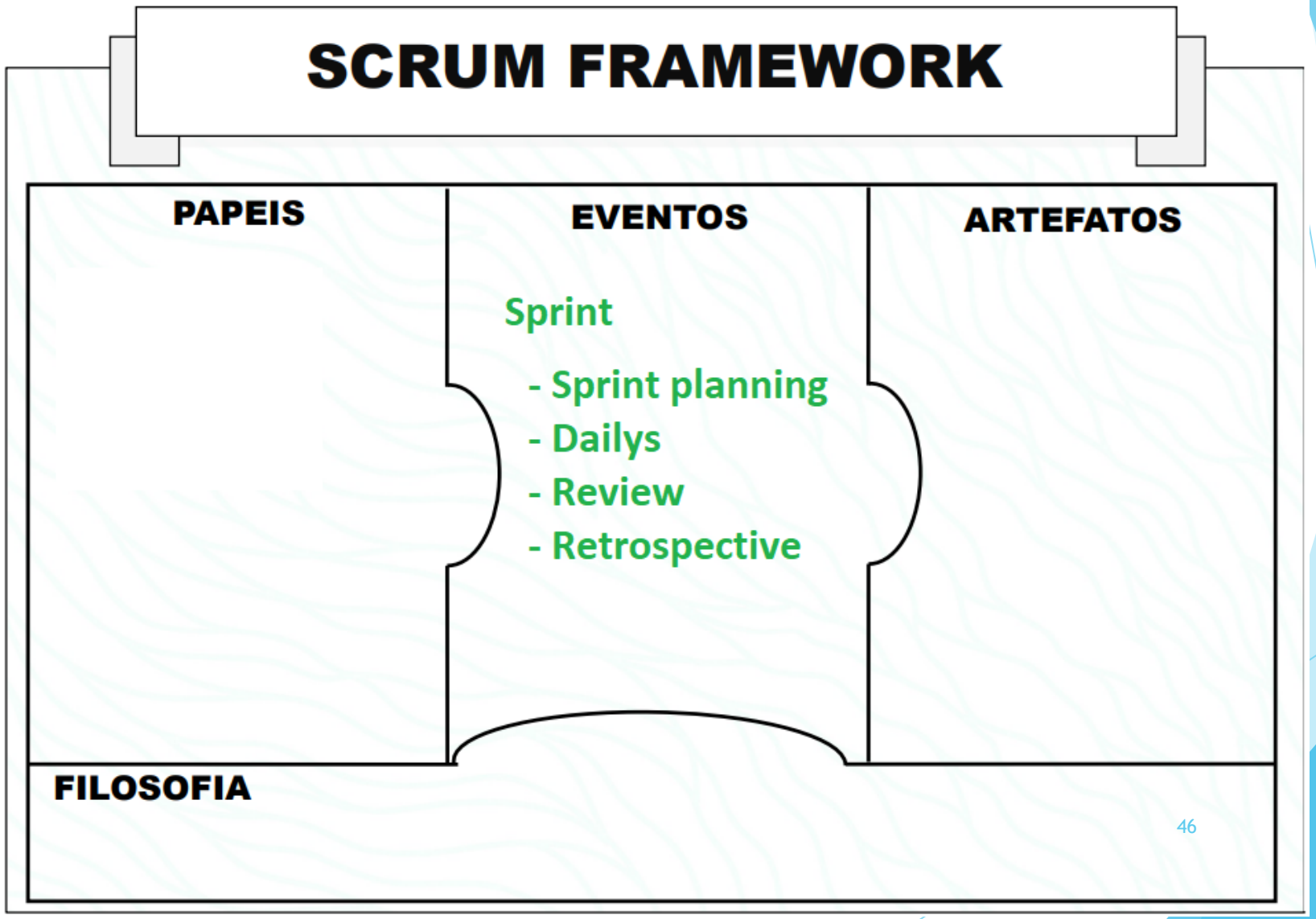
Product Owner - PO

- Responsável por maximizar o valor do produto resultante do trabalho do Scrum Team
  - Desenvolver e comunicar explicitamente a meta do produto
  - Criar e comunicar claramente os itens do Product Backlog
  - Ordenar os itens do Produt Backlog
  - Garantir que o Produtct Backlog seja transparente, visível e compreensível
  - “dono do produto” – conhece a regra de negócio de fato!

Scrum Master

- Responsável pela eficiência ao Scrum Team. Verdadeiros líderes
  - Treina os membros do time
  - Ajuda o Scrum Team a se concentrar na criação de incrementos de alto valor
  - Provocar a remoção de impedimentos
  - Garantir que todos os eventos SCRUM sejam positivos, produtivos e mantidos dentro da timebox (caixa de tempo)

Eventos ou cerimônias



Sprint (1/2)

- Coração do Scrum é a Sprint
- 2 a 4 semanas duração
- Todo o trabalho necessário para alcançar o Product Goal (o que precisamos entregar), incluindo:
  - Sprint Planning (inicio)
  - Daily Scrums (diário)
  - Sprint Review (entrega)
  - Sprint Retrospective (entrega)

Sprint (2/2)

- **Durante a Sprint**
  - Não são feitas alterações que possam por em perigo a Sprint Goal
  - Product Backlog é refinado conforme o necessário
  - Pode ser clarificado e renegociado com o Product Owner à medica que mais se for apreendendo
- Toda e qualquer ajuste precisa ser conversado do PO
- Sprint só pode ser cancelada se o Sprint Goal se tornar obsoleto.
- Apenas o PO tem a autoridade para cancelar a Sprint

Sprint Planning

- **Sprint Planning** inicia a Sprint, determinando o trabalho a ser realizado. Este plano resultante é criado pelo trabalho colaborativo de toda a Scrum Team

- Sprint Planning é **limitada a um máximo de oito horas para um Sprint de um mês.**

- Sprint mais reduzidas, o evento Sprint Planning é mais curto

Daily Scrum

- Objetivo é inspecionar o progresso rumo ao Sprint Goal e adaptar o Sprint Backlog conforme o necessário, ajustando o trabalho planejado que se aproxima
- Daily Scrum é um evento diário de 15 minutos para todos os Developers da Scrum Team
  - O que respondido?
    - O que está trabalhando?
    - O que foi feito?
    - O que vai fazer?
    - E se tem algum impedimento para fazer a tarefa diária!
- Só participa os desenvolvedores

Sprint Review “demonstração”

- Objeto é inspecionar o resultado do Sprint e determinar adaptações futuras
- Scrum Team apresenta os resultados do seu trabalho durante a Sprint, aos principais stakeholders e são discutidos os progressos rumo ao Product Goal

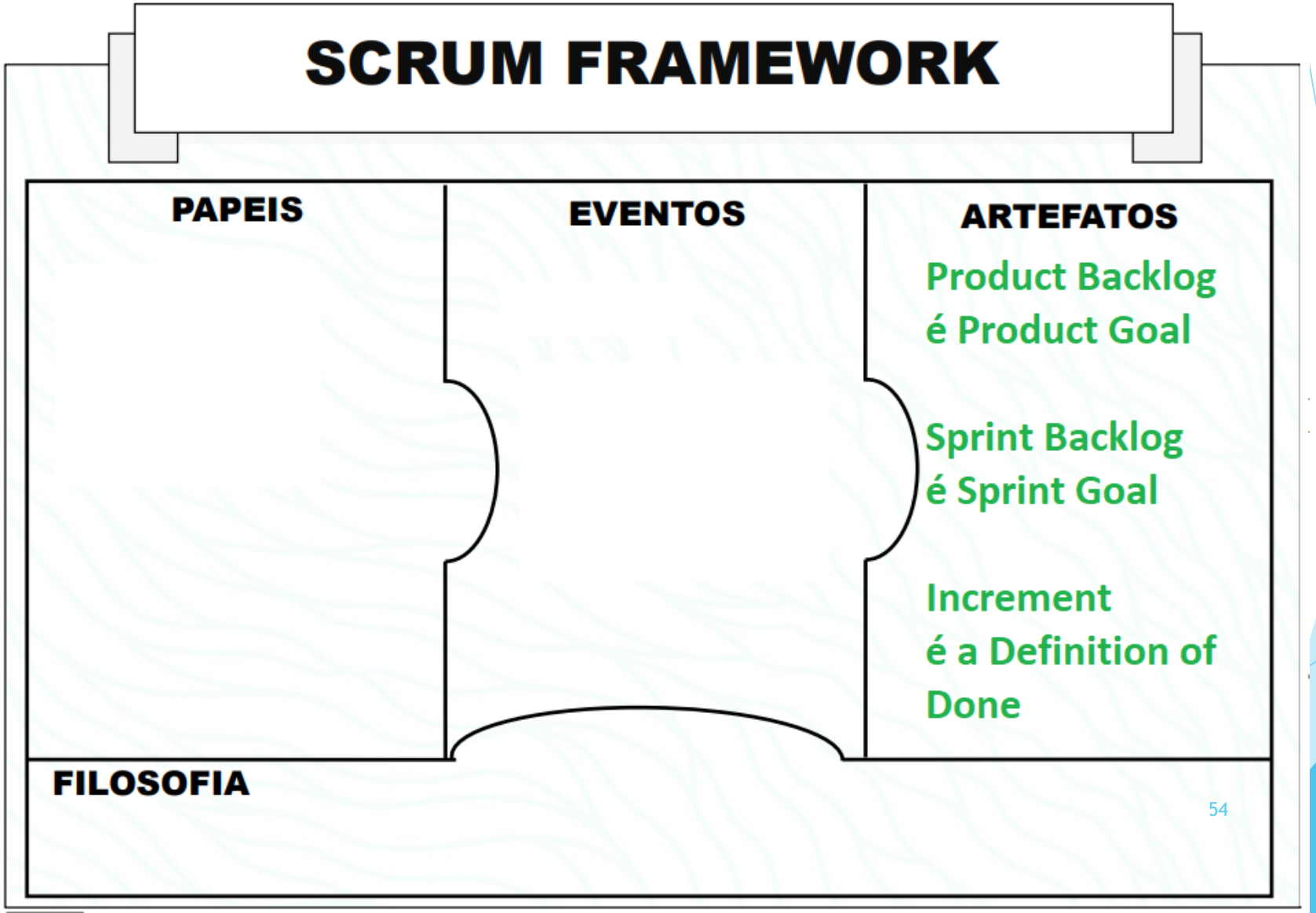
Sprint Restropective “retro”

- Objetivo é planejar formas de aumentar a qualidade e eficácia
- Reunião aberta para o time de desenvolvedores, junto ao Scrum Master, para analisar a última sprint realizada e debater sobre potenciais, pontos de melhorias, buscando otimizar os processos da equipe.

CALENDÁRIO

SEGUNDA	TERÇA	QUARTA	QUINTA	SEXTA
Sprint Planning 3	Daily Scrum 4	Daily Scrum 5	Daily Scrum 6	Daily Scrum 7
Daily Scrum 10	Daily Scrum 11	Daily Scrum 12	Daily Scrum 13	Daily Scrum 14
Daily Scrum 17	Daily Scrum 18	Daily Scrum 19	Daily Scrum 20	Daily Scrum 21
Daily Scrum 24	Daily Scrum 25	Daily Scrum 26	Daily Scrum 27	Sprint Review Sprint Retrospective 28

Artefatos


SCRUM FRAMEWORK
PAPEIS
EVENTOS
ARTEFATOS
Product Backlog
é Product Goal
Sprint Backlog
é Sprint Goal
Increment
é a Definition of
Done
FILOSOFIA

Artefatos

- Representam trabalho ou valor
- Cada artefato contém um compromisso, para assegurar que fornece informação que aumenta a transparência e o foco, contra o qual o progresso pode ser medido:
  - **Product Backlog** é Product Goal
  - **Sprint Backlog** é a Sprint Goal
  - **Increment** é a Definition of Done

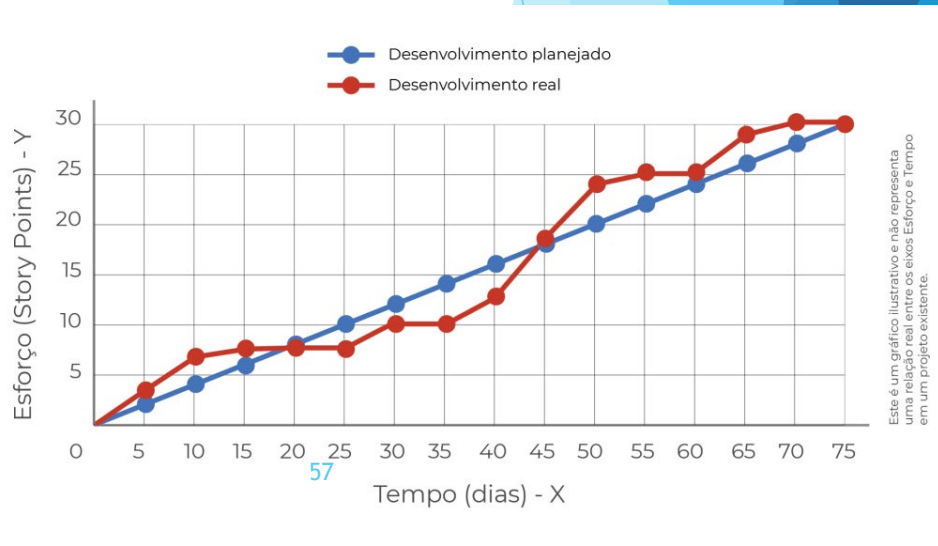
Product Backlog

- É a nossa lista de tarefas
- A partir do Product Backlog, as atividades serão relacionadas para desenvolvimento e entregues por meio de incrementos construídos

Compromisso: Product Goal (o que precisamos entregar)

Sprint Backlog

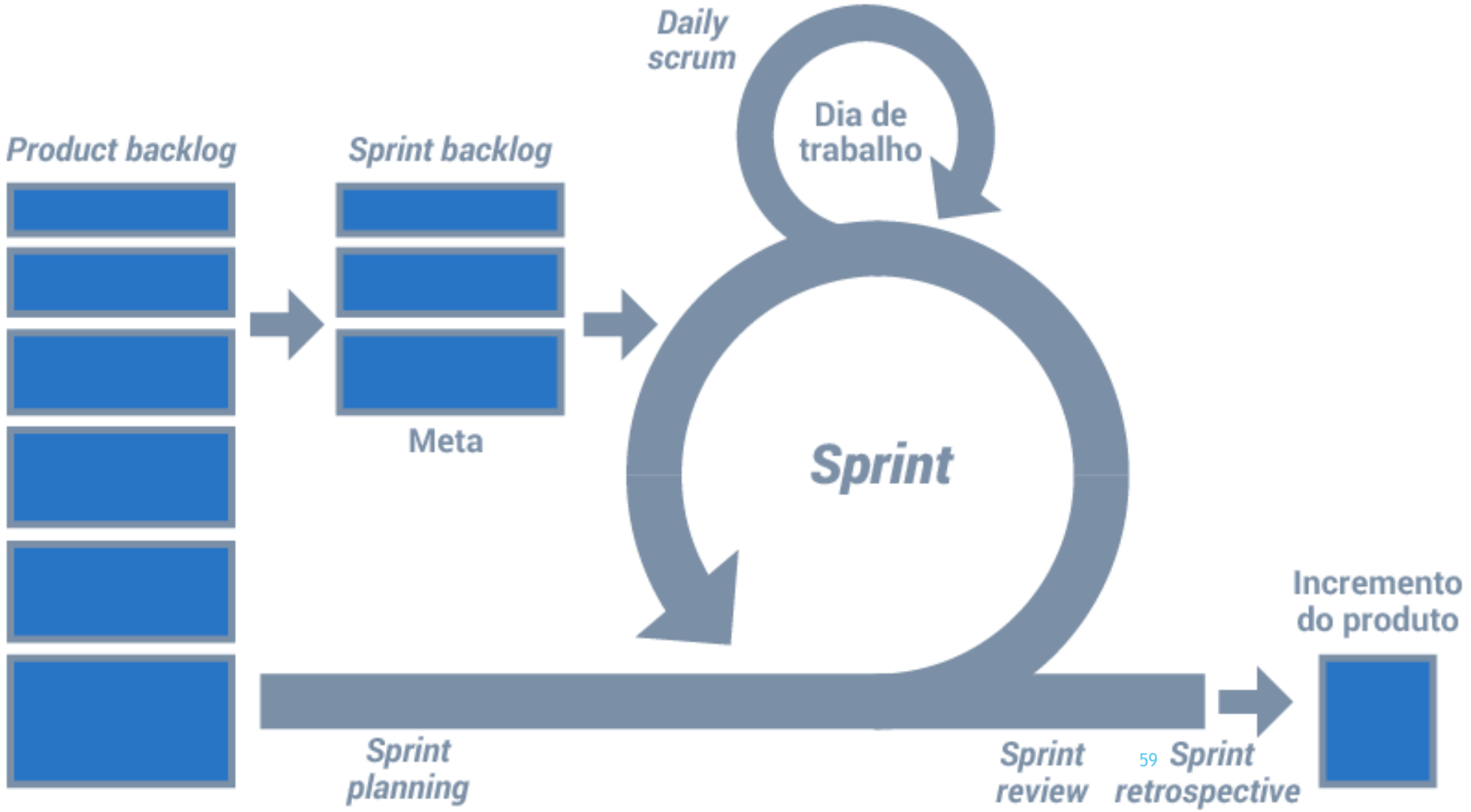
- É uma lista de atividades que precisam ser feitas durante uma Sprint
- O Sprint Backlog é composto pelo Sprint Goal (porque), o conjunto de itens do Produto Backlog selecionados para o Sprint (o quê), bem como um plano acionável para a entrega do Increment (como)
- Para monitoramento pode usar o gráfico Burndow



Increment

- Um incremente é um degrau concreto em direção ao Product Goal.
- Cada increment é acrescentado a todos os increments anteriores e cuidadosamente verificado, asseguramos que todos os increments funcionam em conjunto

SCRUM


Daily scrum
Dia de trabalho
Product backlog
Sprint backlog
Meta
Sprint
Incremento do produto
Sprint planning
Sprint review
Sprint retrospective

Exemplo fictício de um SCRUM loja de carros

- **Scrum Master**: João, responsável por garantir que a equipe siga as práticas do Scrum, ajudar a resolver impedimentos e facilitar as reuniões.
- **Product Owner (PO)**: Ana, que representa os interesses do cliente, mantém e prioriza o Product Backlog, e define os requisitos do software.
- **Desenvolvedores**: Carlos, Lívia e Rafael, que trabalham nas tarefas de desenvolvimento, testes, design e outras atividades para criar o produto.

Exemplo fictício de um scrum loja de carros

Product Backlog (Lista de Requisitos)

1. **Cadastro de Carros**: Permitir o registro de novos carros no sistema.
2. **Listagem de Carros**: Exibir uma lista de todos os carros cadastrados.
3. **Pesquisa de Carros**: Permitir a pesquisa de carros por marca, modelo ou ano.
4. **Edição de Carros**: Possibilitar a edição dos detalhes de carros cadastrados.
5. **Remoção de Carros**: Permitir a exclusão de carros do sistema.

Exemplo fictício de um scrum loja de carros

Sprint 1 (Duração: 2 semanas)

Objetivo da Sprint: Implementar as funcionalidades de cadastro e listagem de carros.

Sprint Planning

Tarefas Escolhidas:

- Desenvolver a interface de usuário para o cadastro de carros.
- Implementar a lógica de backend para armazenar informações de carros.
- Criar a interface de usuário para listagem de carros.
- Implementar a funcionalidade de exibição da lista de carros cadastrados.

Exemplo fictício de um scrum loja de carros

Daily Scrum

Reuniões curtas todos os dias para discutir o progresso, planejar o dia de trabalho e identificar possíveis impedimentos.

Sprint Review

Entrega ao Cliente: Uma demonstração do cadastro e listagem de carros. Feedback do cliente é coletado.

Sprint Retrospective

Discussão sobre o que funcionou bem, o que pode ser melhorado e planos de ação para a próxima Sprint.

Exemplo fictício de um scrum loja de carros

Sprint 2 (Duração: 2 semanas)

Objetivo da Sprint: Implementar as funcionalidades de pesquisa e edição de carros.

Sprint Planning

Tarefas Escolhidas:

- Desenvolver a interface de usuário para pesquisa de carros.
- Implementar a lógica de busca no backend.
- Criar a interface de usuário para edição de carros.
- Implementar a funcionalidade de atualizar informações de carros no backend.

Exemplo fictício de um scrum loja de carros

Daily Scrum

Continuação das reuniões diárias para coordenação e resolução de impedimentos.

Sprint Review

Entrega ao Cliente: Uma demonstração das funcionalidades de pesquisa e edição de carros. Coleta de feedback do cliente para ajustes e melhorias.

Sprint Retrospective

Avaliação do processo da Sprint, identificando sucessos, desafios e áreas para melhoria antes da próxima Sprint.

Resumo

- Este exemplo simplificado mostra como uma equipe pode organizar o desenvolvimento de um software de cadastro de loja de carros usando Scrum. Através da divisão do trabalho em Sprints, a equipe consegue focar em objetivos específicos, adaptar-se com base no feedback do cliente, e melhorar continuamente seus processos.

Certificações

- PSM (Professional Scrum Master)
- CSM (Certified Scrum Master)
- PSPO (Assessments Professional Scrum Product Owner)
- PSD (Profissioanal Scrum Developer)

Vantagens:

Flexibilidade e Adaptabilidade: A capacidade de se adaptar a mudanças nos requisitos é uma grande vantagem, permitindo que o projeto evolua com as necessidades do cliente.

Entrega Incremental: A entrega de partes do projeto em sprints permite uma verificação constante da qualidade e funcionalidade, além de possibilitar ajustes frequentes.

Colaboração e Comunicação: Promove um alto nível de colaboração entre os membros da equipe e com os stakeholders, melhorando a satisfação do cliente e a qualidade do produto.

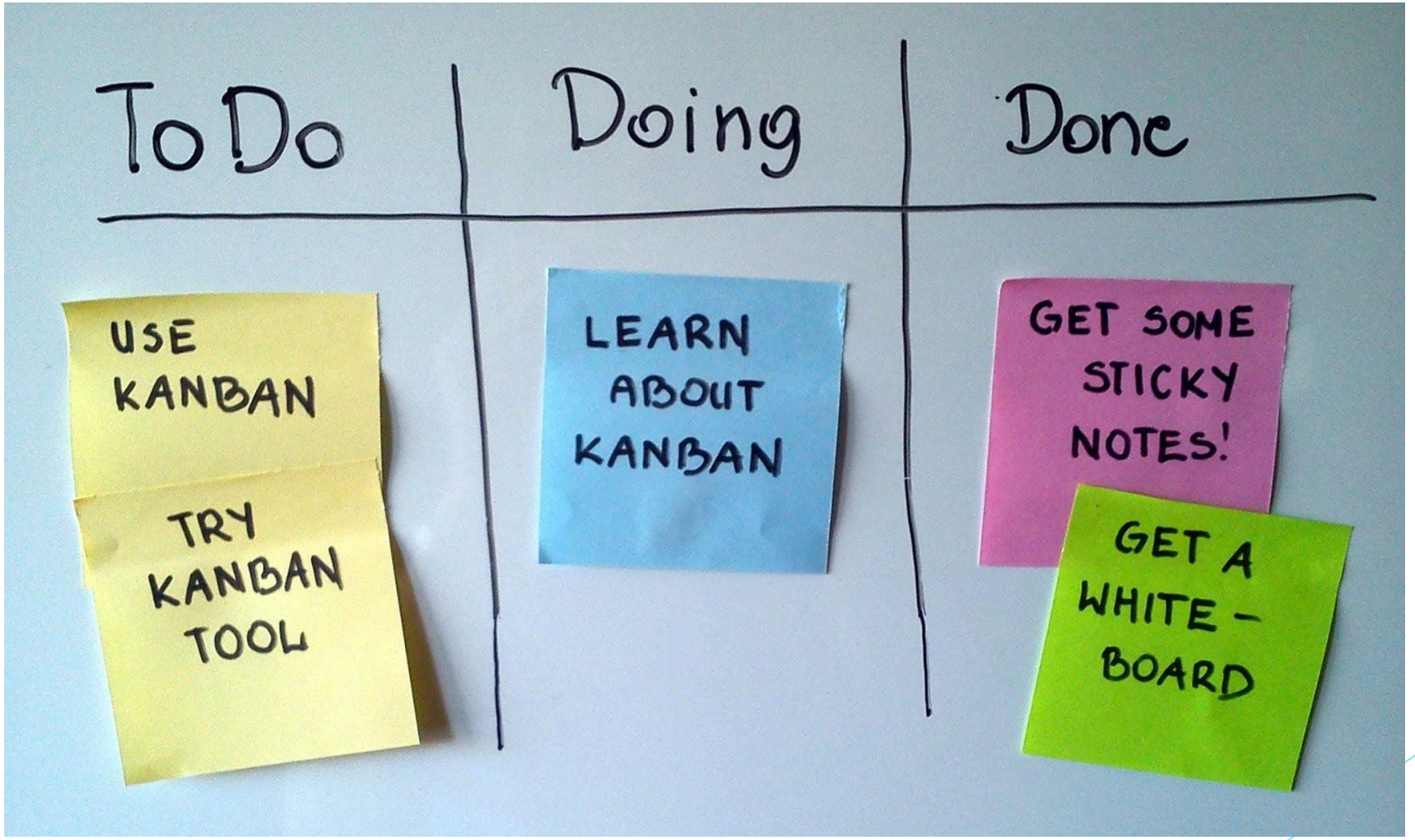
Desvantagens:

Planejamento Menos Previsível: A flexibilidade pode levar a um escopo de projeto menos previsível e a dificuldades em estimar o custo total e o tempo de entrega.

Dependência da Equipe: Requer uma equipe altamente disciplinada e comprometida; sem isso, o projeto pode facilmente sair dos trilhos.

Sobrecarga de Reuniões: As várias reuniões (daily scrums, sprint reviews, etc.) podem ser vistas como onerosas e reduzir o tempo disponível para trabalho efetivo.

kanban


ToDo
Doing
Done
USE KANBAN
TRY KANBAN TOOL
LEARN ABOUT KANBAN
GET SOME STICKY NOTES!
GET A WHITE-BOARD

kanban

- O *kanban* usa muitos dos mesmos princípios vistos no *scrum*. É uma metodologia ágil e, por isso, tem o objetivo de otimizar o processo de desenvolvimento de *software*.
- Diferentemente do *scrum*, o *kanban* não trabalha com *sprints*, trabalha com um quadro para representar o fluxo de desenvolvimento. Esse quadro contém, no mínimo, três colunas: “para fazer”, “fazendo” e “feito”. Nesse cenário sem *sprints*, não consta também o *sprint backlog*. Logo, entenda que a coluna “para fazer” é igual ao *product backlog*, visto no *scrum*.

Scrum x kanban

Característica	Scrum	Kanban
Ciclos de Trabalho	Sprints fixos (geralmente 2-4 semanas)	Fluxo contínuo, sem ciclos fixos
Papéis Definidos	Papéis específicos (Scrum Master, Product Owner, Time de Desenvolvimento)	Sem papéis definidos; a equipe é mais flexível
Quadro de Tarefas	Quadro é resetado no início de cada Sprint	Quadro contínuo, com tarefas movendo-se através de colunas
Mudanças Durante o Ciclo	Mudanças são desencorajadas durante o Sprint	Mudanças podem ser feitas a qualquer momento
Medição de Produtividade	Velocidade da equipe ao longo das Sprints	Lead time (tempo desde o início até a conclusão da tarefa)

Vantagens:

Melhoria Contínua: Facilita a identificação de gargalos e a implementação de melhorias contínuas no processo de trabalho.

Flexibilidade de Tarefas: Permite mudanças a qualquer momento, tornando mais fácil adaptar-se às mudanças nas prioridades ou nos requisitos do projeto.

Visualização do Trabalho: O uso de um quadro Kanban oferece uma visualização clara do progresso do trabalho, facilitando o gerenciamento de tarefas e a distribuição de carga de trabalho.

Desvantagens:

Foco no Processo, Não no Produto: Pode haver uma tendência a focar mais na otimização do processo do que no produto ou resultado final em si.

Menos Estrutura: A falta de prazos e estruturas fixas pode levar a atrasos se a equipe não for auto-motivada ou se houver priorização inadequada de tarefas.

Risco de Sobrecarga de Trabalho: Sem limites claros para as fases ou sprints, membros da equipe podem se tornar sobrecarregados, especialmente se não houver uma gestão eficaz do fluxo de trabalho.

Atividade 2

Product Owner	
Scrum Master	
Desenvolvimento	
Stakeholders	

Atividade 2 – sistema fictício de exemplo sistema de 1 mercado

Product Owner (PO): Maria

Responsabilidades: Define as características do sistema, prioriza o backlog do produto com base nas necessidades do negócio e na feedback dos stakeholders, e garante que o valor do trabalho da equipe de desenvolvimento esteja alinhado com os objetivos do projeto.

Por que Maria?: Ela é a gerente de operações do mercado e tem uma compreensão profunda das necessidades do negócio e dos desafios enfrentados pela equipe de vendas e gerenciamento de estoque.

Atividade 2 – sistema fictício de exemplo sistema de 1 mercado

Scrum Master: Pedro

Responsabilidades: Facilita as reuniões Scrum (Daily Scrum, Sprint Planning, Sprint Review, Sprint Retrospective), ajuda a equipe a remover impedimentos e garante que o processo Scrum seja seguido.

Por que Pedro?: Ele tem experiência em gerenciar equipes ágeis e é conhecido por sua habilidade em promover um ambiente de trabalho colaborativo e produtivo.

1. **Product Backlog**: Maria, como PO, reúne os requisitos do sistema e organiza o backlog do produto, incluindo funcionalidades como rastreamento de estoque, alertas de validade e relatórios de vendas.
2. **Sprint Planning**: Pedro, o Scrum Master, coordena uma reunião de planejamento da Sprint com Maria e a equipe de desenvolvimento. Juntos, eles selecionam itens do backlog para a próxima Sprint, focando em funcionalidades prioritárias como o módulo de rastreamento de estoque.
3. **Desenvolvimento**: Durante a Sprint, Ana, Lucas e Rafael trabalham nas tarefas selecionadas, como o design da interface para o módulo de rastreamento e a implementação da lógica de backend. Eles se reúnem diariamente no Daily Scrum para sincronizar o progresso e identificar possíveis impedimentos.
4. **Sprint Review**: Ao final da Sprint, a equipe apresenta o trabalho realizado aos stakeholders, recebendo feedback crucial que Maria utilizará para ajustar o backlog do produto e priorizar as próximas funcionalidades.
5. **Sprint Retrospective**: Pedro facilita uma retrospectiva com a equipe de desenvolvimento para discutir o que funcionou bem, o que pode ser melhorado e como aumentar a eficiência nas próximas Sprints.

Atividade 2 – sistema fictício de exemplo sistema de 1 mercado

Equipe de Desenvolvimento: Ana (desenvolvedora), Lucas (designer de UI/UX) e Rafael (analista de dados)

Responsabilidades: Trabalham juntos para construir o sistema de gerenciamento de estoque, desde a interface do usuário até o backend e a análise de dados.

Por que essa equipe?: Eles possuem habilidades complementares que cobrem todos os aspectos técnicos necessários para desenvolver o sistema.

Atividade 2 – sistema fictício de exemplo sistema de 1 mercado

Equipe de Desenvolvimento: Ana (desenvolvedora), Lucas (designer de UI/UX) e Rafael (analista de dados)

Responsabilidades: Trabalham juntos para construir o sistema de gerenciamento de estoque, desde a interface do usuário até o backend e a análise de dados.

Por que essa equipe?: Eles possuem habilidades complementares que cobrem todos os aspectos técnicos necessários para desenvolver o sistema.

Atividade 2 – sistema fictício de exemplo sistema de 1 mercado

Stakeholders: Donos do mercado, equipe de vendas, funcionários do estoque

Responsabilidades: Fornecem feedback sobre o sistema, ajudam a identificar requisitos críticos e são os principais beneficiários das soluções desenvolvidas.

Por que são importantes?: Seu input é crucial para garantir que o sistema atenda às necessidades reais do mercado e melhore os processos existentes.

d) Partindo das prioridades do sistema, sugira três tarefas (*user stories*) para a primeira semana de desenvolvimento. Em cada tarefa, indique para quem servirá a funcionalidade e o motivo da tarefa.

Eu, como um:
Quero:
De modo que:

Exemplo fictício mercado - stackholders

1. **Cadastro de Produtos**
    1. **Eu, como um:** Gerente do mercado
    2. **Quero:** Adicionar novos produtos ao sistema
    3. **De modo que:** Possa gerenciar o estoque de produtos disponíveis para venda, incluindo detalhes como nome, preço, e quantidade.

2. **Pesquisa de Produtos**
    1. **Eu, como um:** Funcionário do caixa
    2. **Quero:** Pesquisar produtos por nome ou código de barras
    3. **De modo que:** Possa rapidamente verificar preços e detalhes para os clientes durante o checkout.