GPS - Gestão de Projetos de Software

Aula 9: Histórias de Usuário e Épicos

Prof. Aléssio Miranda Júnior
alessio@cefetmg.br

CEFET-MG - Campus Timoteo
Dep. Engenharia de Computação

Novembro de 2026

Sumário

Objetivos da Aula

- Compreender o fim da "Especificação de 300 páginas" (IEEE 830).
- Aprender a fatiar a complexidade: **Épicos** vs **User Stories**.
- Dominar a sintaxe Connextra (*Como* / *Quero* / *Para*).
- Aplicar a Matriz **INVEST** como filtro de qualidade.
- Diferenciar **Critérios de Aceitação** de **Definition of Done (DoD)**.
- Entender a pipeline de automação **DevOps (CI/CD)**.
- Transpor o formato ágil de requisitos para as *Issues* do GitLab (Check 03).

A Máquina Exige Clareza

"Garoto, como traduzir os delírios do cliente em algo que os nossos engenheiros consigam codificar? Chega de calhamaços de especificações que ninguém lê."

Neste nono capítulo, vamos aprender a escrever **Histórias de Usuário** e fatiar problemas absurdos em **Épicos**. Se a equipe técnica não sabe exatamente o que é a linha de chegada ("Pronto"), ela nunca vai parar de programar, e o seu orçamento vai virar poeira.

A Engenharia de Software no passado (IEEE 830)

No modelo Cascata, a praxe era contratar um Analista de Requisitos.

- Ele passava de 3 a 6 meses entrevistando o cliente.
- Produzia um "Documento de Especificação de Requisitos" em Word com 300 páginas.
- Entregava o documento na mesa dos Desenvolvedores e ia embora.

O Fracasso da Especificação Exaustiva

Qual o problema prático dessa abordagem?

1. Ninguém lê um PDF de 300 páginas antes de começar a codar.
2. Quando lêem, a interpretação de texto falha miseravelmente, gerando o fenômeno do "telefone sem fio" arquitetural.
3. No instante em que o PDF é impresso, ele já está obsoleto porque o negócio do cliente já mudou.

Requisitos Vivos e Cartões

A abordagem Ágil aboliu o contrato de papel engessado.

O Cartão é um Convite

No Scrum, não documentamos o software exaustivamente antes de construí-lo. Nós criamos **Cartões (Issues)**. Um cartão não é uma especificação técnica; é um *convite para uma conversa ativa* entre quem programa e o Product Owner.

O Problema da Complexidade Maciça

Para gerenciar o tamanho do software, a engenharia ágil divide os requisitos em camadas de abstração.

- O que é um **Épico?** É uma iniciativa estratégica corporativa, grande demais para ser desenvolvida em apenas uma Sprint de 14 dias.
- *Exemplo:* **"Criar Módulo de Tutor IA Socrático"**.

O Perigo do Épico na Sprint

Não coloque um Épico no Backlog da Sprint!

Se você empurrar um Épico inteiro para o seu *Sprint Backlog*, a equipe passará 14 dias brigando com a API e o banco de dados e não conseguirá integrar NADA usável. A Sprint falhará 100% das vezes.

O Épico precisa obrigatoriamente ser decomposto (*Fatiado*) até atingir uma granularidade minúscula chamada **História de Usuário (User Story)**.

Visualizando a Árvore de Decomposição

O que é a User Story?

A *User Story* é a menor fatia de trabalho que entrega **valor real e perceptível** ao cliente no fim de uma Sprint.

Para garantir que o programador não perca o foco no *porquê* de estar codificando aquilo (evitando programar coisas desnecessárias), utilizamos um template linguístico universal (Padrão Connextra):

Como `[Persona / Tipo de Usuário]`
Eu quero `[Ação ou Funcionalidade]`
Para `[Um Valor / Benefício de Negócio]`

Exemplo Ruim vs Exemplo Bom

O Exemplo Ruim (Foco na Máquina)

"Criar rota POST com JWT para enviar texto para a API da OpenAI e devolver um objeto JSON com a Response."

(Isto é uma Tarefa Técnica. O Desenvolvedor pode codar a rota perfeitamente, mas a IA pode devolver uma resposta inútil pro aluno).

O Exemplo Bom (Foco no Usuário)

"Como **estudante**, *eu quero* **que o bot leia o meu código** *para* **receber uma dica sobre o laço for sem receber a resposta final**."

O "Para Quê" Salva Projetos

O **Valor de Negócio (Para Quê)** é a parte mais importante do cartão.

- O Professor pede: "Quero um botão para exportar todo o histórico de chat da IA em formato PDF".
- O programador gasta 5 dias sofrendo com bibliotecas de PDF no Node.js.
- *O Segredo:* Se o desenvolvedor perguntar o **"Para Quê"**, e o professor responder: "Para eu ler o PDF, jogar no ChatGPT e pedir um resumo do aluno"...
- O programador percebe o desperdício! O próprio código poderia resumir e exibir na tela em 1 hora, dispensando o PDF!

A Matriz de Qualidade INVEST (I-N-V)

Antes de colocar a História na *Sprint*, o PO aplica o filtro **INVEST**:

- **I (Independent):** Pode ser programada sem depender *criticamente* que outra história termine no mesmo dia.
- **N (Negotiable):** Não é um contrato blindado. Se a API for muito cara, a equipe pode negociar a lógica.
- **V (Valuable):** Entrega valor visual ao cliente final (não é apenas refatorar banco de dados interno).

A Matriz de Qualidade INVEST (E-S-T)

- **E (Estimable):** A equipe de código entende a regra o suficiente para saber se leva 2 ou 10 dias.
- **S (Small):** É pequena o bastante para caber com folga dentro de uma única Sprint (máx 14 dias).
- **T (Testable): Crucial.** É possível provar logicamente que ela foi concluída? (Se for "Deixar o app mais bonito", não é testável).

A Síndrome dos 90% Prontos

Um dos erros mais letais para equipes imaturas é a temida "Síndrome dos 90% prontos".

- A *Issue* fica presa em um limbo eterno (Doing), recebendo ajustes diários porque o desenvolvedor e o *Product Owner* nunca concordaram sobre qual era a linha de chegada.
- No universo Ágil verdadeiro, o código é **binário**: ou ele atende 100% e vai para produção, ou ele gera 0% de Valor Agregado.

Como Matar a Ambiguidade?

Para aniquilar a desculpa do *"mas na minha máquina funciona"*, o Scrum exige duas camadas de checklist no GitLab:

1. **Critérios de Aceitação (Local):** Regras aplicadas **apenas a uma** história específica. (O que o botão Login precisa fazer?).
2. **Definition of Done - DoD (Global):** Regras de qualidade corporativa aplicadas a **todas** as histórias do repositório, sem exceção. (O que significa "entregar código" na nossa empresa?).

Exemplo: Critérios de Aceitação

Exemplo para a História do "Login de Maratonista via SUAP":

Acceptance Criteria (Dentro da Issue)

- [X] O redirecionamento via OAuth2 deve ocorrer em menos de 2s.
- [X] O nome e e-mail do aluno devem ser salvos no banco.
- [ ] Retornar Tela 401 amigável se a senha do SUAP estiver incorreta.

Sem o terceiro checkbox, a *Issue* não passa da *Review*.

Exemplo: Definition of Done (DoD)

Exemplo para **todas** as *Issues* do projeto Assistente IA:

O Definition of Done (Na Wiki)

Nenhuma Issue avança para "Closed" sem cumprir TODOS os itens:

1. Código aprovado por ao menos 1 colega via *Merge Request*.
2. Pipeline de integração contínua rodando sem falhas (Linter verde).
3. O Prompt de sistema (IA) versionado no repositório.
4. Código não quebra o *build* da branch *Main*.

Se a *Issue* cumpriu a Aceitação do Login, mas **violou o Linter** (DoD), a entrega é rejeitada!

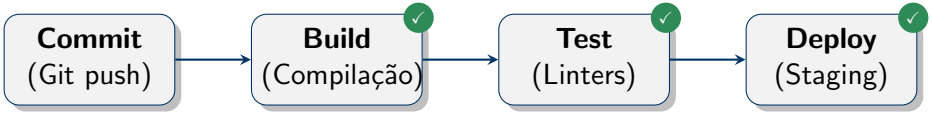
A Cultura DevOps

Uma *User Story* perfeita, mas que demora semanas esperando o administrador de redes publicar o código, é valor represado.

Aqui entra o **DevOps**: a união agressiva entre o Desenvolvimento (Dev) e as Operações de Infraestrutura (Ops) para automatizar o caminho entre a digitação do código e o servidor final.

O Pipeline Automatizado

O mecanismo central do DevOps é o **Pipeline de CI/CD**.



CI vs CD

O que a máquina faz pelo time?

- **Integração Contínua (CI):** A cada *push*, um Runner no GitLab acorda, baixa o código, compila e roda *Linters* / Testes Unitários. O dev descobre em 2 minutos se quebrou o sistema.
- **Entrega Contínua (CD):** Se o CI passar verdinho, a máquina hospeda o site provisório em um ambiente seguro (*Staging*) para o Cliente testar.
- **Deploy Contínuo:** Se os testes passarem, o código é injetado diretamente no servidor de produção (Onde a Netflix vive).

O Check 03 vai caçar o engajamento metodológico do grupo.

Como o professor avaliará suas Issues?

1. O título da Issue é apenas "Banco de Dados" ou é a Ação (ex: "Histórico de IA")?
2. A descrição da Issue traz a taxonomia formal: "Como / Quero / Para"?
3. Existem *Critérios de Aceitação* injetados no corpo da Issue na forma de Checklists do Markdown (`- [ ]`)?
4. Vocês fecharam o **Definition of Done (DoD)** corporativo do time em uma página da Wiki?

DevOps Básico no Projeto

Para comprovar Maturidade em Qualidade (QA), seu projeto **precisa**:

- Possuir um arquivo `.gitlab-ci.yml` na raiz do projeto.
- Quando um *Merge Request* for aberto, ele deve acionar um "Job" no GitLab CI.
- Esse *Job* pode ser simplesmente o comando que verifica se o código front/back-end compila, ou um comando de *Linter* (Eslint / Flake8).
- Proibido aprovar Merge Requests com Pipeline Vermelho!

Reflexões do Capítulo

1. Por que injetar um "Épico" gigantesco de Inteligência Artificial para ser programado dentro de uma única Sprint de 2 semanas é a receita exata para destruir a moral dos engenheiros?
2. Analisando o acrônimo INVEST, explique corporativamente por que aprovar uma *User Story* que é classificada como "Não Testável" gera um buraco negro financeiro na Sprint.
3. Se o desenvolvedor construiu toda a interface do LLM perfeitamente (Critérios de Aceitação), mas pulou a criação do Docker / Linter imposta pelo *DoD* e causou falha no Pipeline CI, o *Scrum Master* deve permitir que ele feche a *Issue*?

As Suas Histórias de Usuário Estão Testáveis?

Email: alessio@cefetmg.br
Horário de aula: Segunda, 7h às 8:40h
Horário de atendimento: Quarta 16h e whatsapp

Preparem os pipelines para o Check 03!

Próxima aula: Como gerenciar métodos ágeis fora da ditadura do *timebox* do Scrum? Entraremos no mundo pragmático do **Kanban**.