GPS - Gestão de Projetos de Software

Aula 4: Escopo e Cronograma

Prof. Aléssio Miranda Júnior
alessio@cefetmg.br

CEFET-MG - Campus Timoteo
Dep. Engenharia de Computação

Setembro de 2026

Sumário

Objetivos da Aula

- Definir as fronteiras do **Escopo do Produto** e do **Escopo do Projeto**.
- Dominar a construção da **EAP** (Estrutura Analítica do Projeto) usando a Regra dos 100%.
- Diferenciar e mitigar o **Scope Creep** e o **Gold Plating**.
- Converter Pacotes de Trabalho em Cronograma (Atividades).
- Aprender a realizar estimativas robustas usando a matemática do **PERT**.
- Identificar o **Caminho Crítico (CPM)** em diagramas de Gantt.
- Transpor o planejamento estático para as ferramentas dinâmicas do GitLab.

A Mensagem do Mentor

O Erro do Amador

"Escuta aqui, garoto. O maior erro de um amador é construir o prédio antes de desenhar a planta."

Neste capítulo, entraremos no núcleo duro do planejamento preditivo. Vamos definir com precisão absoluta *o que* será feito (Escopo) e *quando* será entregue (Cronograma). A matemática começa agora.

O que é o Escopo?

No PMBOK, o escopo não é uma ideia genérica, é um contrato.

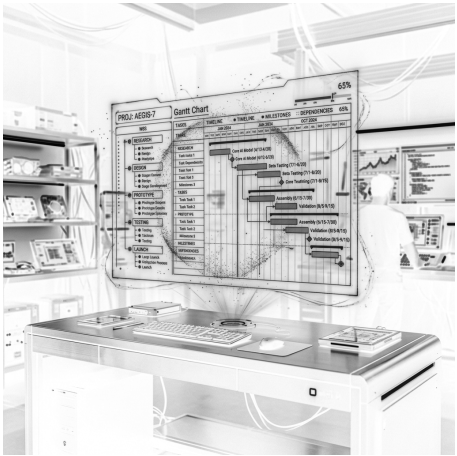
Escopo do Produto:

- Os recursos, funções e características que definem o software.
- *Ex:* "Responder a dúvidas de código com precisão; Interface via Telegram."

Escopo do Projeto:

- Todo o trabalho braçal e gerencial necessário para construir o produto.
- *Ex:* Reuniões semanais, testes unitários, treinamento de *deploy*.

Projeção do Escopo e Tempo



A gestão profissional amarra visualmente o Escopo (eixo Y) ao Tempo (eixo X).

A Ferramenta de Ouro: WBS / EAP

Para garantir que a equipe inteira saiba exatamente o que fazer, a gestão clássica inventou a ferramenta de planejamento mais importante de todas:

EAP (Estrutura Analítica do Projeto)

Em inglês *WBS (Work Breakdown Structure)*. É a decomposição hierárquica e visual do escopo total em partes menores, até chegar ao nível gerenciável.

Imagine uma árvore invertida: no topo (nível 0) temos o Projeto, e nas raízes temos as tarefas diárias.

A Regra dos 100%

Como garantir que a EAP está correta?

A regra suprema do PMI dita que:

- A EAP deve conter **TODO** o trabalho do projeto.
- Se uma funcionalidade está na EAP, ela faz parte do projeto.
- Se uma funcionalidade **NÃO ESTÁ** na EAP, ela não existe no projeto e não receberá financiamento ou tempo.
- Nada a mais, nada a menos.

O Pacote de Trabalho

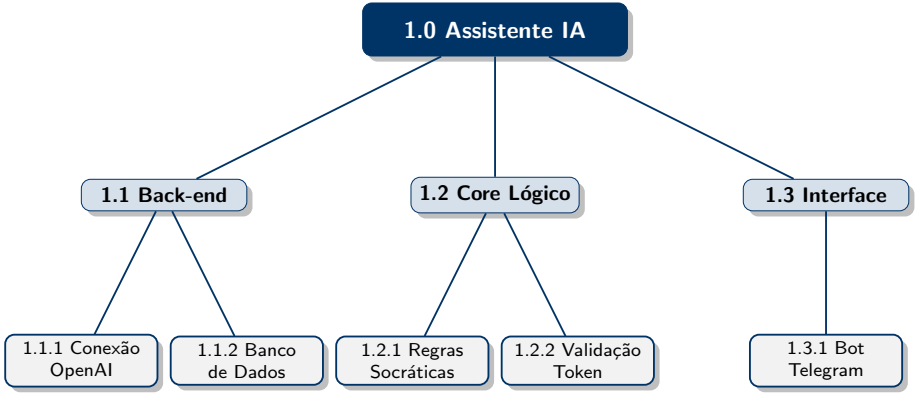
Descendo a hierarquia da EAP, chegamos à folha da árvore: o **Pacote de Trabalho** (*Work Package*).

- É a menor unidade do **Escopo**.
- Diferente de uma tarefa granular de 1 hora, o Pacote é um agrupamento lógico maior.
- Ele é grande o suficiente para ser estimado financeiramente e temporalmente de forma confiável (ex: "Módulo de Login").

No nosso projeto

Seu "Pacote de Trabalho" no papel será exatamente a sua *Issue* no GitLab.

A EAP do Assistente IA



Os Destruidores de Prazo

Você desenhou a EAP, orçou o projeto e iniciou os trabalhos. O que pode dar errado?

Em engenharia de software, o maior risco não é a tecnologia, é a **mudança descontrolada de escopo**. O escopo tende a inflar como um balão de ar se não for ativamente monitorado.

Inimigo 1: Scope Creep

Corrupção de Escopo

É o inchaço gradual e silencioso do projeto provocado por pressões externas, sem revisão de prazo ou custo.

Como ocorre?

- O cliente ou *Product Owner* pede "só mais um botãozinho".
- A equipe atende por educação, sem registrar no *Project Charter*.
- Vários pedidos pequenos somam-se, e no final do mês, o projeto atrasa.

Inimigo 2: Gold Plating

Folheamento a Ouro

É o inchaço do escopo causado pela **própria equipe técnica**, sem solicitação do cliente.

Como ocorre?

- O programador acha que a tela ficaria mais bonita com uma animação 3D.
- Ele decide adicionar o recurso "porque é legal" ou porque quer testar uma "biblioteca nova".
- O cliente não pediu, o escopo não cobre, e o recurso gerará custo de manutenção no futuro.

Transformando Pacotes em Tempo

Com a EAP validada e trancada a sete chaves, o gerente traduz o "O Quê" para o "Quando".

- A EAP dita o Escopo (Pacotes de Trabalho).
- O **Cronograma** decompõe os Pacotes em **Atividades** diárias.
- Regra de Ouro: Uma atividade saudável deve possuir entre **4 e 40 horas** de esforço. Abaixo de 4h gera microgerenciamento. Acima de 40h gera incerteza.

Como Estimar Tempo? Fugindo do Chute

Programadores adoram "chutar" prazos. O gerente profissional usa técnicas:

- **Estimativa Análoga:** Baseada em intuição e projetos passados. (Rápida, péssima precisão).
- **Estimativa Paramétrica:** Baseada em índices estatísticos (Ex: um endpoint REST simples sempre leva cerca de 2 horas na nossa equipe).
- **Estimativa de Três Pontos (PERT):** O método mais profissional, absorvendo a incerteza estatisticamente.

O Método PERT

O PERT impede a mentira do prazo otimista. Pedimos ao desenvolvedor 3 cenários:

A Fórmula Matemática:

$$Te = \frac{O + 4M + P}{6}$$

- **O (Otimista):** Se tudo der certo.
- **M (Mais Provável):** O cenário padrão (peso 4).
- **P (Pessimista):** Se a API falhar, o servidor cair ou houver refatoração pesada.

Exemplo Real de PERT

Pacote: Integrar API da OpenAI

O (Otimista): 8 horas ("Se a doc estiver boa")
M (Mais Provável): 16 horas ("Cenário com tratamento básico de erro")
P (Pessimista): 32 horas ("Se o LLM travar por limite de tokens e exigirmos cache")

Cálculo: $Te = (8 + (4 \times 16) + 32) \div 6$
$Te = (8 + 64 + 32) \div 6 = 104 \div 6 = \mathbf{17{,}3\ horas}$.

Ao montar o cronograma, reserve **18 horas** e ignore as "8 horas" que o estagiário prometeu!

Duração vs. Esforço

Muitos cronogramas falham por confundir grandezas!

- **Esforço (Effort):** Quantidade bruta de horas de trabalho humano necessárias. Ex: A tarefa exige 16 horas de código bruto.
- **Duração (Duration):** Tempo de *calendário* necessário.

O Choque de Realidade

Se o desenvolvedor possui 16 horas de **Esforço** para aplicar, mas ele trabalha e estuda e só consegue alocar **2 horas por dia** no projeto... A **Duração** da atividade será de **8 dias úteis** no cronograma!

A Rede de Dependências

Atividades não ocorrem no vácuo simultaneamente. Elas possuem relacionamentos lógicos ditando a ordem de execução.

A Dependência Fim-para-Início (F-I):

- É o relacionamento mais comum.
- A Atividade B só pode *Inciar* quando a Atividade A *Finalizar*.
- Ex: Só podemos testar as regras de *Prompt* da IA, *depois* que a conexão com a API da Groq/OpenAI estiver feita.

O Caminho Crítico (Critical Path)

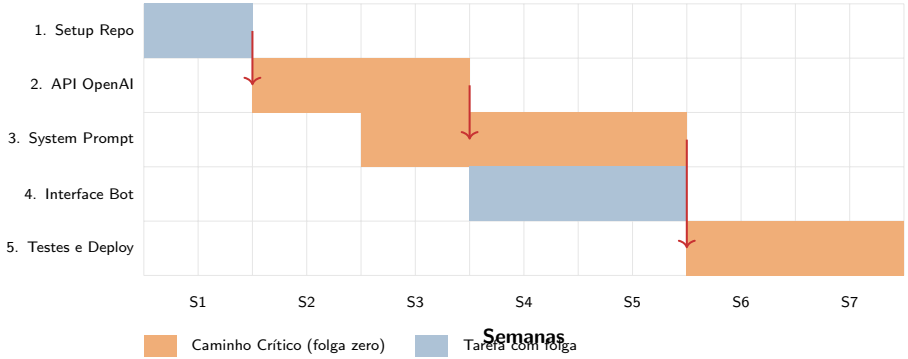
Quando conectamos todas as dependências, descobrimos a malha do projeto.

Definição

O Caminho Crítico é a sequência mais longa de tarefas dependentes do projeto.

A Magia da Folga Zero: As tarefas que estão no caminho crítico ditam a data final do projeto inteiro. Elas **não possuem folga**. Se uma tarefa do caminho crítico atrasar 1 dia, a entrega final para o cliente atrasará 1 dia inevitavelmente.

Desenhando o Caminho Crítico (Gantt)



Entendendo o Gráfico

No diagrama anterior, as tarefas em laranja formam o **Caminho Crítico**:

$$\textit{Setup} \rightarrow \textit{API} \rightarrow \textit{System Prompt} \rightarrow \textit{Deploy}$$

A Folga (Slack): A tarefa azul "Interface do Bot" pode atrasar um pouco sem comprometer a data final do Deploy, pois ela ocorre em paralelo e termina antes da próxima restrição (tem folga). Mas se o "System Prompt" demorar mais, a entrega inteira escorrega.

Monitoramento e Linha de Base (Baseline)

Cronogramas de software não são esculpidos em pedra, mas eles precisam de uma fundação.

- Quando a EAP e o Cronograma são assinados na Fase de Planejamento, geramos a **Linha de Base (Baseline)**.
- A Linha de Base vira o referencial oficial.
- Durante a execução, usamos o **EVM** (que veremos profundamente depois) para cruzar a Linha de Base com a execução real e garantir que o escopo e o tempo estão saudáveis!

No GitLab: A EAP e o Cronograma viram Código

Como aplicamos o peso do PMI no nosso ambiente *Agile* no Projeto Integrador?

O GitLab converte sua teoria em gestão de alto impacto:

1. Cada **Pacote de Trabalho** da sua EAP em papel se torna uma **Issue** aberta no repositório. (Não está no repo? Não está no escopo).
2. **Decomposição:** Utilize a ferramenta de *Checklist* (Markdown) dentro da descrição da *Issue* para fatiá-la em Atividades diárias para os programadores.

Tempo e Dependências no GitLab

3. **Estimativas PERT:** Após calcular o seu *Te* usando a fórmula PERT, aplique o tempo direto na *Issue* do GitLab através do comando oficial: `/estimate [tempo]`. (O sistema cria a métrica de *Weight*).
4. **Dependências Rígidas:** Se a Tarefa A está no caminho crítico bloqueando a Tarefa B, utilize o sistema nativo de vínculo do GitLab para marcar que a "Issue B está **bloqueada por** Issue A".

Missão do Grupo

Execução Prática

1. Finalizem o *Project Charter* definindo bem os limites do produto.
2. Desenhem no papel (ou MIRO) a sua **EAP (WBS)**.
3. Lancem todos os pacotes aprovados no **Board de Issues** da oficina.
4. O professor validará se o seu Caminho Crítico faz sentido.

Reflexões do Capítulo

Antes de fechar a planta da nossa oficina de IA, reflita:

1. O cliente pede reconhecimento de voz no seu chatbot, e a equipe aceita sem renegociar prazo. Qual fenômeno ocorreu?
2. O programador estima "10h otimista", "40h pessimista (API falhar)" e histórico "17h base". Usando PERT, que valor vai para o cronograma?
3. Você tem uma atividade no Caminho Crítico (40h de esforço). Você aloca um estagiário de 20h semanais. Qual a "Duração" e por que ele não pode atrasar de jeito nenhum?

Dúvidas?

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

Não esqueça de responder o questionário!

Próxima aula: Qualidade e Recursos Humanos