GPS - Gestão de Projetos de Software

Aula 15: Apresentação Final e Defesa

Prof. Aléssio Miranda Júnior
alessio@cefetmg.br

CEFET-MG - Campus Timoteo
Dep. Engenharia de Computação

Dezembro de 2026

Sumário

Objetivos da Aula

- Aprender a projetar o **Pitch do Produto** (focando em Valor, não em Código).
- Dominar a estrutura visual e narrativa de uma apresentação técnica.
- Preparar-se para o apocalíptico **Efeito Demo** (Falhas ao Vivo).
- Defender as decisões da equipe na arguição de **Trade-offs**.
- Fechar o ciclo do PMBOK através do **Registro de Lições Aprendidas**.
- Compreender os critérios do **Check 04 Final** (Produto, Processo e Apresentação).

O Mentor Adverte: O Fim da Linha

Comunicação é Sobrevivência

"A engenharia mais espetacular da galáxia não vale um centavo se você gaguejar e não souber explicá-la. O código não se defende sozinho perante um conselho de diretores."

Neste último capítulo de treinamento, vamos focar nas artes ocultas da persuasão técnica. Você vai aprender a justificar as suas decisões arquiteturais de forma inquestionável.

A Tentação Técnica

Quando a equipe chega ao apagar das luzes do projeto, o instinto primitivo do engenheiro é abrir a IDE e exibir a limpeza algorítmica do seu *backend*.

Suprima esse instinto.

Diretores e investidores não se importam com a complexidade assintótica do banco de dados; eles se importam com a **resolução de problemas reais**.

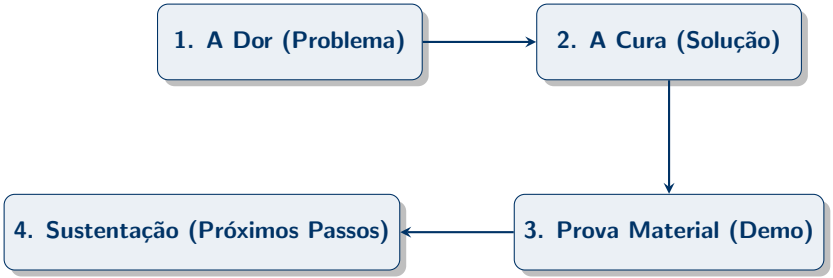
Valor Acima de Código

Sua apresentação deve focar obsessivamente no **Valor de Negócio** (Retorno sobre o Investimento - ROI).

A abordagem Júnior: *"Construímos uma API GraphQL em Node.js com cache no Redis."*
A abordagem do Engenheiro Líder: *"Entregamos um motor de transações que reduz o tempo de espera do usuário final em 85%, blindado contra picos de acesso na Black Friday."*

A Estrutura Visual do Pitch

A estrutura de um *Pitch* de software de sucesso é fixa e direta. Não perca tempo com histórias irrelevantes ou detalhes teóricos vazios.



Passo 1: A Dor (Problema)

O primeiro minuto do *Pitch* define se os diretores vão continuar te escutando ou se vão começar a checar o celular.

- Você precisa fazer o cliente concordar que ele **tem um problema grave**.
- *Exemplo Prático:* "Notamos que os alunos calouros passam até 4 horas da semana tentando achar a regra de jubilamento em um PDF burocrático de 400 páginas."
- Você quantificou o tempo perdido, gerou empatia e preparou o terreno.

Passo 2: A Cura (Solução Conceitual)

Apresente a visão do seu produto de forma simples e de altíssimo nível, sem ofender a inteligência do cliente com jargões (esqueça o nome do banco de dados).

- *Exemplo Prático:* "Para aniquilar essa dor, criamos o Tutor IA. Um assistente que leu os 400 PDFs e responde, com linguagem acessível, qualquer dúvida acadêmica em exatos 3 segundos."
- Você vendeu o benefício direto: tempo poupado.

Passo 3: A Prova Material (A Demonstração)

É aqui que você consome 60% do tempo da sua apresentação.

As promessas de PowerPoint acabaram. O avaliador quer ver o "sistema vivo".

- O engenheiro assume o teclado e projeta a tela do sistema em Staging/Produção.
- Você escreve a pergunta real ao vivo, aperta `Enter` e deixa a magia acontecer na frente da banca.

Passo 4: Sustentação (Roadmap e Escala)

Você provou que o seu sistema resolve o problema *hoje*. Agora você precisa provar que pensou no *amanhã*.

- Apresente as funcionalidades que ficaram no *Backlog* para a Versão 2.0.
- *Exemplo Prático:* "No momento, o bot responde via terminal. Para o próximo ciclo, nosso roadmap inclui a integração via WhatsApp Web e a autenticação SSO do Google para ler o histórico real do aluno."

O Apocalíptico "Efeito Demo"

O núcleo duro do *Pitch* é ver o Software rodando ao vivo. Contudo, há uma maldição clássica na engenharia de software corporativa:

A Lei Termodinâmica da Apresentação

"A probabilidade de um sistema falhar ao vivo é diretamente proporcional ao poder aquisitivo e à hierarquia de quem está assistindo."

Doutrinas de Contenção

Engenheiros seniores não contam com a sorte; eles operam com protocolos rígidos:

- **Cenário Coreografado:** Não clique a esmo na interface! Escreva o fluxo passo a passo (*Login* $\rightarrow$ *Pergunta* $\rightarrow$ *Resposta*) e não desvie do caminho feliz (*Happy Path*).
- **Massa de Dados Limpa:** Apresentar a interface do cliente contendo o registro *"Joãozinho Teste das Couves"* destrói sua credibilidade. Preencha o banco de dados antes da demo com nomes, fotos e valores extremamente realistas.

Sistemas Redundantes (O Plano B)

Assuma a pior premissa possível: **A internet vai cair, e o servidor de Nuvem vai reiniciar.**

Como se defender?

- **Contingência Local:** Tenha um *Deploy* da aplicação e do banco de dados rodando em *localhost* (na sua máquina isolada).
- **O Plano C (Vídeo):** Na pior das hipóteses absolutas, se até o seu computador travar, tenha um **vídeo impecável gravado da tela** executando o *Happy Path*. Dar *play* no vídeo é infinitamente melhor do que pedir desculpas.

O Desastre Aconteceu. E Agora?

Mesmo com redundâncias, um erro de tela vermelha estourou no meio da apresentação. Como um engenheiro sênior se comporta?

- **Nunca culpe um colega ("Ah, o fulano não testou o back-end").** A culpa é sempre do processo de arquitetura.
- **Mantenha a frieza:** "Tivemos um *timeout* no gateway de autenticação. Por isso ambientes distribuídos exigem resiliência. Vou trocar para a nossa contingência local e dar continuidade."
- Uma falha contornada com calma gera mais credibilidade do que um sistema que funcionou por sorte.

O Tribunal da Arguição

Após a *Demo*, a etapa final é a arguição da banca. A banca tentará desestabilizar suas escolhas para testar a maturidade da equipe.

A Pergunta Letal:

"Por que vocês escolheram usar a linguagem/banco de dados X, se o mercado diz que a Y é melhor?"

Dominando Trade-Offs

Responder: *"Porque a gente só sabia programar na ferramenta X"* expõe negligência. A resposta madura baseia-se em **Trade-offs** (Concessões).

O Escudo da Decisão

"Optamos pela Linguagem X porque, apesar da Linguagem Y ter benchmarks 15% mais rápidos, a Linguagem X possui um ecossistema gigantesco de bibliotecas de IA (LlamaIndex, LangChain) que nos permitiu entregar o MVP totalmente funcional no prazo crítico imposto pelo negócio."

O Fim do "Eu Acho"

Em sabatinas de engenharia, a hesitação cheira a sangue na água.

Se o cliente perguntar: *"Por que a IA demorou 12 segundos para responder?"*, banir a frase *"Eu acho que a culpa é da API"*. Na engenharia, nós **medimos**, não **achamos**.

Resposta Certa: *"Nossos relatórios de log apontaram um limite de rateio (Rate Limit) na versão gratuita do modelo neste horário. A versão 2 em produção já prevê um sistema de Cache para zerar esse tempo."*

A Memória Institucional da Equipe

O encerramento formal de um projeto (exigência do PMBOK) requer a entrega do **Registro de Lições Aprendidas** (*Lessons Learned Register*).

Não é um texto burocrático para engavetar; é a base do seu próximo projeto de sucesso. O documento responde com honestidade brutal a três perguntas:

1. O que **repetiremos** no próximo projeto com certeza?
2. O que **nunca mais** faremos?
3. O que **não sabíamos** no início e agora é óbvio?

Exemplos Críticos de Lições

- **A Repetir:** *"A automação de CI/CD via GitLab Runner nos salvou de 3 deploys quebrados de sexta-feira. Vamos usar o `.yml` como template para as próximas disciplinas."*
- **O Nunca Mais:** *"Deixamos o ajuste do System Prompt da IA para o final. A IA começou a alucinar na véspera. Nunca mais deixaremos o Core-Business para a Sprint 3."*
- **A Surpresa:** *"Descobrimos que a Engenharia de Prompts consome muito mais tempo do que plugar a API em si. O esforço é enorme."*

O Alerta do Desperdício

O Projeto que Não Ensina Nada

Se a sua equipe chega ao final da disciplina e não consegue listar pelo menos 5 Lições Aprendidas pesadas e sinceras, algo deu muito errado.

Ou o projeto que vocês fizeram era trivial demais (sem desafios reais), ou a equipe não teve a maturidade para fazer retrospectivas durante as Sprints.

O Veredito do Projeto Integrador

O último Check da disciplina não testa apenas a tela colorida do seu sistema; ele mede a sua sobrevivência aos quatro grandes pilares: Planejamento, Execução, Monitoramento e Comunicação.

Os Três Eixos da Avaliação

1. **O Motor (Produto 50%):** O sistema roda de verdade? Está livre de bugs bloqueadores e atende ao que foi prometido nas Histórias Épicas iniciais?
2. **O Log (Processo 30%):** O GitLab comprova empiricamente a evolução, ou tem 50 *Commits* na madrugada da entrega?
3. **O Escudo (Apresentação 20%):** A postura durante a Demo. Dados limpos, discurso de negócios e habilidade em rebater críticas na arguição.

Perguntas Reflexivas (Pitch)

1. Por que obrigar um *Stakeholder* investidor a olhar para a estrutura do seu banco de dados ou para algoritmos durante o *Pitch* destrói a sua credibilidade profissional?
2. Com base nas estratégias de redundância de uma *Demo*, o que você fará no exato segundo em que o servidor da nuvem (AWS/Azure) reiniciar de surpresa na frente da banca examinadora?
3. A essência da engenharia é a gestão de Concessões (*Trade-offs*). Argumente defendendo uma escolha técnica imperfeita (ex: um banco de dados mais lento) justificando com o ganho em Prazo ou Custo.

A partir de hoje, você não é mais um "digitador de código" passivo.

Você é um Engenheiro(a) de Software que governa o caos, que blinda o escopo e que protege sua equipe com empirismo e dados concretos.

Se as nossas batalhas ao longo das aulas provaram algo, foi que o sucesso do software nunca depende só da qualidade da linguagem... depende das decisões das pessoas!

Obrigado e Boa Sorte!

Leve essas cicatrizes de guerra para o mercado.

Email: alessio@cefetmg.br
Horário de atendimento: Quarta 16h e whatsapp

Preparem a Apresentação Final!

O mercado lá fora não perdoa, mas vocês estão prontos.