GPS - Gestão de Projetos de Software

Aula 2: Viabilidade e Estrutura

Prof. Aléssio Miranda Júnior
alessio@cefetmg.br

CEFET-MG - Campus Timoteo
Dep. Engenharia de Computação

Setembro de 2026

Sumário

Relação entre ES e Gestão de Projetos

Definição

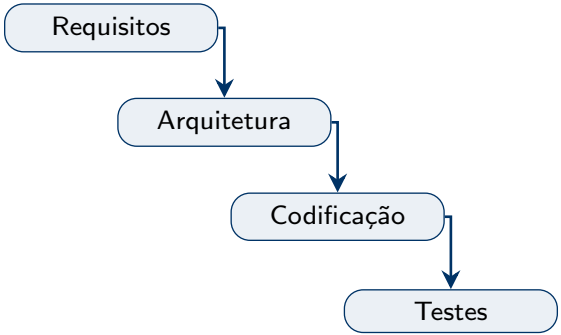
Engenharia de Software é a disciplina que aplica princípios de engenharia ao desenvolvimento, operação e manutenção de software.

Como se relacionam:

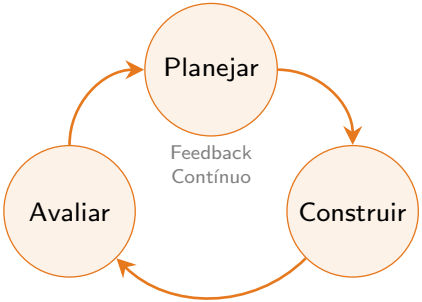
- **ES** fornece **processos** e **metodologias**
- **Gestão de Projetos** aplica esses processos de forma **controlada**
- **ES** define **o que** fazer
- **Gestão** define **como** e **quando** fazer

Modelos de Desenvolvimento: Cascata vs Ágil

Cascata (Preditivo)



Ágil (Iterativo)



Modelos de Desenvolvimento

Modelo Cascata:

- Fases sequenciais
- Documentação extensa
- Controle rigoroso
- Mudanças difíceis
- Adequado para projetos estáveis

Modelo Iterativo:

- Ciclos incrementais
- Feedback contínuo
- Adaptação a mudanças
- Entrega parcial
- Adequado para projetos complexos

Escolha do Modelo

A **gestão do projeto** deve escolher o modelo mais adequado baseado nas características do projeto e da organização.

O que é Viabilidade?

Definição

Viabilidade é a análise que determina se um projeto pode ser executado com sucesso, considerando todos os aspectos técnicos, econômicos, operacionais e legais.

Objetivos da análise:

- **Validar** a ideia do projeto
- **Identificar** riscos e obstáculos
- **Estimar** recursos necessários
- **Comparar** alternativas
- **Tomar decisão** de prosseguir ou não

Tipos de Viabilidade

Viabilidade Técnica:

- Tecnologia disponível
- Habilidades da equipe
- Infraestrutura necessária
- Recursos técnicos
- Complexidade do projeto

Viabilidade Econômica:

- Custos totais
- Benefícios esperados
- Retorno sobre investimento
- Fluxo de caixa
- Análise de rentabilidade

Outros tipos: Operacional, Legal, Temporal, Organizacional

Outros Tipos de Viabilidade

Viabilidade Operacional

- Adequação a processos/rotinas
- Dispon. de equipe p/ operação/suporte
- Treinamento e curva de aprendizagem
- Procedimentos, manuais e manutenção

Viabilidade Temporal

- Prazo vs. esforço estimado
- Dependências e janelas de oportunidade
- Marcos críticos e datas fixas
- Capacidade da equipe no período

Viabilidade Legal

- Conformidade legal/regulatória
- Propr. intelectual (licenças, marcas)
- Contratos/termos/responsabilidades
- LGPD: dados e privacidade

Viabilidade Organizacional

- Alinh. estratégico e patrocínio
- Maturidade de processos e cultura
- Orçamento, pessoas, infraestrutura
- Impacto em áreas/operações

Dica

Use matriz (Peso x Nota) para ponderar e justificar seguir/adiar.

Análise de Viabilidade Técnica

Componentes da análise:

1. **Tecnologia:**
   - API de LLM (ex: Gemini, OpenAI)
   - Python (back-end e lógica)
   - Streamlit ou Telegram (Interface)
   - GitLab (controle de versão e Wiki)
2. **Habilidades da equipe:**
   - Conhecimento de desenvolvimento web
   - Experiência com Git
   - Capacidade de trabalho em equipe
   - Habilidades de gestão

Análise de Viabilidade Econômica

Componentes da análise:

1. **Custos:**
   - Tempo da equipe (15 semanas)
   - Recursos computacionais
   - Ferramentas e licenças
   - Treinamento e capacitação
2. **Benefícios:**
   - Aprendizado prático
   - Portfólio profissional
   - Experiência em gestão
   - Certificação da disciplina

ROI Educacional

O investimento em tempo e esforço resulta em conhecimento prático e portfólio profissional.

Análise de Viabilidade Operacional

Componentes da análise:

1. **Processos e Rotinas**
   - Integração com práticas atuais
   - Adaptações necessárias
2. **Recursos Humanos**
   - Equipe p/ operação e suporte
   - Cobertura e plantão (quando aplicável)
3. **Capacitação e Procedimentos**
   - Treinamento e materiais rápidos
   - Procedimentos e checklists mínimos

Exemplo (Assistente IA)

Garantir que a equipe consiga monitorar se o script Python caiu e reiniciar o bot no servidor (quem é responsável pelo deploy?).

Análise de Viabilidade Legal

Componentes da análise:

1. **Conformidade**
   - Regulatório e institucional
   - Termos de uso e políticas
2. **Propriedade Intelectual**
   - Licenças de temas/dependências
   - Imagens e marcas utilizadas
3. **Privacidade e Dados**
   - LGPD (dados pessoais nos perfis)
   - Consentimento e remoção sob demanda

Exemplo (Assistente IA)

Garantir que os prompts não violem a política de uso da API e não infringir a LGPD nos logs das dúvidas dos alunos.

Análise de Viabilidade Temporal

Componentes da análise:

1. **Prazo vs. Esforço**
   - Estimativas por sprint
   - Redução de escopo (MVP)
2. **Caminho Crítico e Dependências**
   - Tarefas que travam as demais
   - Dependências externas
3. **Capacidade da Equipe**
   - Feriados/provas/ausências
   - Alocação real por semana

Exemplo (Assistente IA)

Planejar Sprints de forma que o MVP (bot respondendo fixo) esteja pronto muito antes da integração real (análise socrática).

Análise de Viabilidade Organizacional

Componentes da análise:

1. **Alinhamento e Patrocínio**
   - Objetivos do curso/time
   - Apoio do professor/cliente
2. **Recursos e Maturidade**
   - Pessoas, orçamento, infraestrutura
   - Processos e cultura para mudança
3. **Impacto Operacional**
   - Interferência em outras atividades
   - Plano de comunicação

Exemplo (Assistente IA)

Validar o escopo e regras com o professor, reservar slots semanais para debater a arquitetura e usar um canal ágil de comunicação.

Matriz de Viabilidade

Critério	Peso	Nota	Pontos
Viabilidade Técnica	0.4	9	3.6
Viabilidade Econômica	0.3	8	2.4
Viabilidade Operacional	0.2	9	1.8
Viabilidade Temporal	0.1	7	0.7
TOTAL	**1.0**		**8.5**

Interpretação

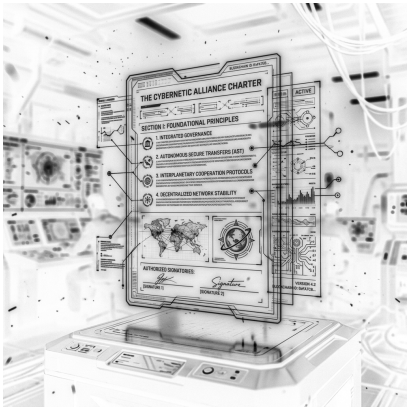
Pontuação 8.5/10: Projeto altamente viável. Recomenda-se prosseguir com o desenvolvimento.

Termo de Abertura (Project Charter)

O **Project Charter** é o documento oficial que formaliza a iniciativa e concede autoridade ao Gerente de Projetos.

Elementos Obrigatórios:

- **Business Case:** Justificativa do projeto.
- **Requisitos de Alto Nível:** O que o sistema fará.
- **Premissas e Restrições:** Custos máximos, infra.
- **Objetivos SMART:** Metas claras e gerenciáveis.



Definindo Objetivos SMART

Para que um objetivo seja alcançável, ele deve ser:

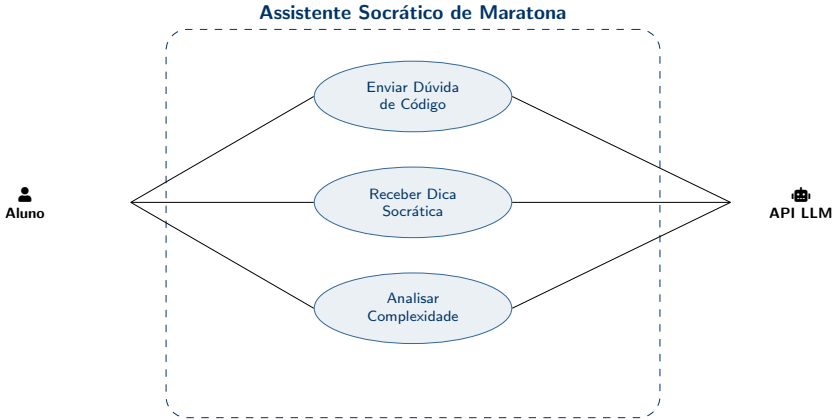
- **S (Specific):** Específico. O que será construído?
- **M (Measurable):** Mensurável. Como medir o sucesso?
- **A (Achievable):** Atingível. É possível de se fazer?
- **R (Relevant):** Relevante. Resolve a dor do usuário?
- **T (Time-bound):** Temporal. Qual o prazo final?

Exemplo do Assistente IA

"Desenvolver um Assistente Socrático via API do Gemini (S, A) capaz de responder a dúvidas com 95% de precisão (M), reduzindo a sobrecarga dos professores (R) até o fim do Sprint 4 (T)."

Requisitos e Modelagem Visual (UML)

A Engenharia de Requisitos organiza **o que** será construído (Funcionais) e **como** deve se comportar (Não-Funcionais).



Identificação de Riscos

Categorias de riscos:

1. **Riscos Técnicos:**
   - Dificuldade com API
   - Problemas com GitLab
   - Conflitos de versão
   - Falhas de CI/CD
2. **Riscos de Equipe:**
   - Conflitos internos
   - Ausência de membros
   - Diferentes níveis de conhecimento
   - Falta de comprometimento

Análise de Probabilidade e Impacto

Risco	Prob.	Impacto	Exposição
Dificuldade com API	Média	Alto	Alto
Conflitos de equipe	Baixa	Alto	Média
Falhas de CI/CD	Média	Médio	Média
Ausência de membros	Alta	Médio	Alto

Legenda: Baixa (1-3), Média (4-6), Alta (7-9)

Estratégias de Mitigação

Para riscos de alta exposição:

1. **Dificuldade com API:**
   - Treinamento prévio
   - Documentação detalhada
   - Suporte técnico
   - Exemplos práticos
2. **Ausência de membros:**
   - Distribuição equilibrada de tarefas
   - Documentação compartilhada
   - Reuniões regulares
   - Backup de responsabilidades

Fases do Projeto

Duração total: 15 semanas (semestre letivo)

Cronograma Detalhado - Semanas 1-5

Semana 1-2:

- Formação das equipes
- Configuração do GitLab
- Instalação das bibliotecas da API
- Definição de escopo

Entregas: Estrutura básica funcionando

Semana 3-5:

- Estrutura básica do site
- Página principal
- Templates básicos
- Configuração CI/CD

Cronograma Detalhado - Semanas 6-10

Semana 6-8:

- Desenvolvimento de perfis
- Conteúdo das páginas
- Estilização CSS
- Responsividade

Entregas: Site completo e funcional

Semana 9-10:

- Funcionalidades avançadas
- Otimizações
- Testes de usabilidade
- Correções de bugs

Cronograma Detalhado - Semanas 11-15

Semana 11-13:

- Testes finais
- Documentação
- Preparação da apresentação
- Refinamentos finais

Semana 14-15:

- Apresentação final
- Demonstração do produto
- Entrega da documentação
- Encerramento do projeto

Entregas: Produto final e documentação completa

Composição da equipe:

1. **Scrum Master (1):**
   - Facilita reuniões
   - Remove impedimentos
   - Garante processo ágil
   - Monitora progresso
2. **Product Owner (1):**
   - Define prioridades
   - Valida funcionalidades
   - Representa stakeholders
   - Aprova entregas
3. **Desenvolvedores (1):**
   - Implementam funcionalidades
   - Criam conteúdo
   - Testam funcionalidades
   - Documentam código

Responsabilidades da Equipe

Modelo de trabalho colaborativo:

Flexibilidade

Todos os membros podem executar qualquer tarefa do projeto, promovendo:

- Aprendizado colaborativo
- Redundância de conhecimento
- Flexibilidade operacional – Suporte mútuo

Responsabilidade Formal

Cada tarefa tem um **responsável formal** designado que:

- Garante a qualidade da entrega
- Coordena a execução
- Reporta o progresso
- Assume a responsabilidade final

Modelo Alternativo: Não Colaborativo

Modelo de trabalho especializado:

Especialização

Cada membro tem responsabilidades específicas e bem definidas:

- Scrum Master: apenas gestão de processo
- Product Owner: apenas definição de requisitos
- Desenvolvedor: apenas implementação técnica

Vantagens do Modelo Não Colaborativo

- **Especialização profunda** em cada área
- **Responsabilidades claras** e bem definidas
- **Menos conflitos** de responsabilidade
- **Processo mais previsível** e estruturado

Recursos Técnicos

Ferramentas necessárias:

1. **Desenvolvimento:**
   - Hugo (gerador de sites)
   - Editor de código (VS Code)
   - Git (controle de versão)
   - Navegador web
2. **Gestão:**
   - GitLab (repositório)
   - Google Docs (documentação)
   - Discord/Slack (comunicação)
   - Trello/Asana (tarefas)

Importante

Todas as ferramentas são gratuitas ou já disponíveis na instituição.

Recursos de Infraestrutura

Requisitos mínimos:

1. **Hardware:**
   - Computador com 4GB RAM
   - 10GB de espaço livre
   - Conexão com internet
   - Navegador atualizado
2. **Software:**
   - Sistema operacional atualizado
   - Git instalado
   - Hugo instalado
   - Editor de código
3. **Ambiente:**
   - Acesso ao GitLab
   - Conta Google (Forms)
   - Espaço de trabalho colaborativo
   - Backup de dados

Métricas de Qualidade

Critérios técnicos:

1. **Funcionalidade:**
   - Site acessível e funcional
   - Todas as páginas carregam
   - Links funcionam corretamente
   - Formulários operacionais
2. **Usabilidade:**
   - Interface intuitiva
   - Navegação clara
   - Responsivo (mobile)
   - Tempo de carregamento ¡ 3s
3. **Conteúdo:**
   - Informações completas
   - Texto sem erros
   - Imagens otimizadas
   - Links funcionais

Métricas de Processo

Critérios de gestão:

1. **Planejamento:**
   - Cronograma respeitado
   - Entregas no prazo
   - Mudanças controladas
   - Riscos mitigados
2. **Colaboração:**
   - Trabalho em equipe
   - Comunicação efetiva
   - Distribuição de tarefas
   - Resolução de conflitos
3. **Documentação:**
   - Plano de projeto
   - Documentação técnica
   - Relatórios de progresso
   - Lições aprendidas

O que Vem na Próxima Aula?

Aula 3: Planejamento no GitLab

- **Conteúdo Teórico:**
  - Iniciação formal do projeto
  - Elaboração do Project Charter
  - Configuração de repositórios
- **Conteúdo Prático:**
  - Criação de repositórios no GitLab
  - Configuração de issues
  - Definição de milestones
  - Configuração de CI/CD

Tarefas para a Próxima Aula

1. **Definir estrutura detalhada** do Assistente IA
2. **Criar wireframes** das páginas principais
3. **Definir paleta de cores** e identidade visual
4. **Instalar bibliotecas (ex: python-telegram-bot)** e criar projeto local
5. **Configurar GitLab** e criar repositório
6. **Responder questionário** da Aula 2

Lembrete

Questionário Google Forms ao final da aula para validação de presença e aprendizado.

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: Planejamento no GitLab

Dinâmica em Sala: Apresentação das Equipes

Objetivo: cada equipe apresenta seu critério de formação e por que se considera equilibrada.

1. **Critérios de Formação** (1 min)
   - Como definiram os membros? (disponibilidade, interesse, experiência)
   - Regras ou acordos estabelecidos
2. **Equilíbrio da Equipe** (1 min)
   - Distribuição de habilidades (técnicas, gestão, comunicação)
   - Cobertura de papéis (PO, SM, Dev, QA) – mesmo que preliminar
3. **Plano Inicial** (1 min)
   - Próximos passos até a próxima aula
   - Riscos percebidos e como pretendem mitigar

Condução

3 minutos por equipe. Um representante deve se manifestar. Feedback breve do professor/colegas.

Checklist Rápido para as Equipes

- Nome da equipe e canal de comunicação definido
- Papéis preliminares atribuídos (podem mudar)
- Disponibilidades mapeadas (janelas semanais de trabalho)
- Repositório criado no GitLab e acesso garantido a todos
- Tema/estrutura inicial do Hugo escolhidos (ou candidatos)