

 De: O Mentor

*Projetos convivem com incertezas e mudanças. Neste capítulo, a **Gestão de Riscos** oferece um processo para identificá-las, priorizá-las e definir respostas proporcionais. A **Documentação** registra conhecimento e decisões relevantes; ela complementa, mas não substitui, testes, revisões e procedimentos operacionais. Credenciais e chaves de API não devem ser registradas na documentação nem no repositório: use variáveis de ambiente, um gerenciador de segredos e controle de acesso adequado.*

1 Gestão de Riscos: Mapeando Ameaças e Oportunidades

A Figure 1 contextualiza a gestão de riscos apresentada a seguir. Um erro comum de equipes iniciantes é tratar risco como sinônimo de problema. Um problema é algo que já aconteceu e requer tratamento. O **risco** é um evento *incerto* que, se ocorrer, poderá afetar o projeto. A gestão de riscos ajuda a equipe a se preparar e a decidir onde investir esforço. Conforme o PMBOK, riscos podem ser negativos (ameaças) ou positivos (oportunidades).

Apenas listar possibilidades em uma planilha não basta. A equipe precisa avaliar quais ameaças exigem resposta prioritária e quais podem ser monitoradas. A matriz de probabilidade e impacto é uma técnica qualitativa de triagem; suas escalas e limiares devem ser definidos no contexto do projeto.



Figure 1: Ilustração de abertura: observação e priorização de riscos. Fonte: Imagem gerada por IA, com supervisão do autor.

1.1 A Matriz de Probabilidade e Impacto (Pxl)

A equipe pode qualificar os riscos cruzando duas variáveis: a chance do evento ocorrer (**probabilidade**) e o efeito esperado sobre objetivos do projeto (**impacto**). A classificação orienta a conversa e não substitui a análise de contexto.

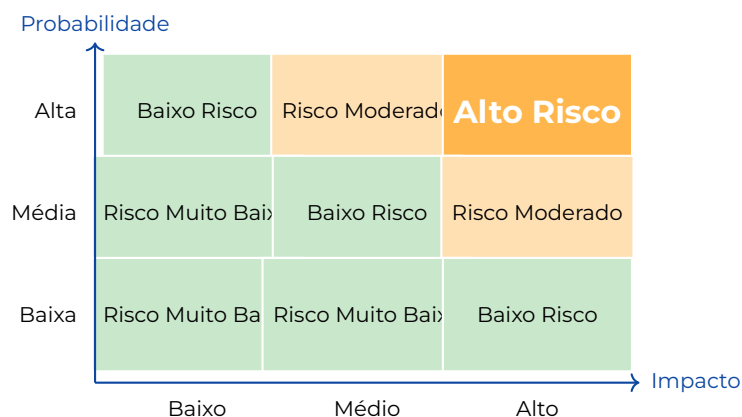


Figure 2: Matriz qualitativa de probabilidade e impacto para triagem de riscos. Fonte: Elaboração do autor.

A Figura 2 sintetiza visualmente o conceito discutido nesta seção.

1.2 Estratégias de Resposta

Para cada risco catalogado no *Risk Register* (Registro de Riscos), a equipe define um plano de ataque. Para as ameaças, temos quatro estratégias clássicas:

- **Evitar:** Alterar o plano para eliminar a ameaça ou sua causa. (Ex.: substituir uma biblioteca sem manutenção por uma alternativa compatível e mantida).
- **Transferir:** Alocar contratualmente a responsabilidade financeira ou operacional por parte do impacto a outra parte. (Ex.: contratar seguro cibernético ou suporte com acordo de nível de serviço para um serviço crítico). A equipe continua responsável por selecionar, configurar e acompanhar o fornecedor; apenas hospedar na nuvem não transfere automaticamente o risco.
- **Mitigar:** Reduzir a probabilidade ou o impacto. (Ex.: criar e testar backups periódicos).
- **Aceitar:** Decidir conscientemente não mudar o plano, quando o custo da resposta não se justifica. A aceitação ativa pode incluir reserva de contingência e plano de resposta.

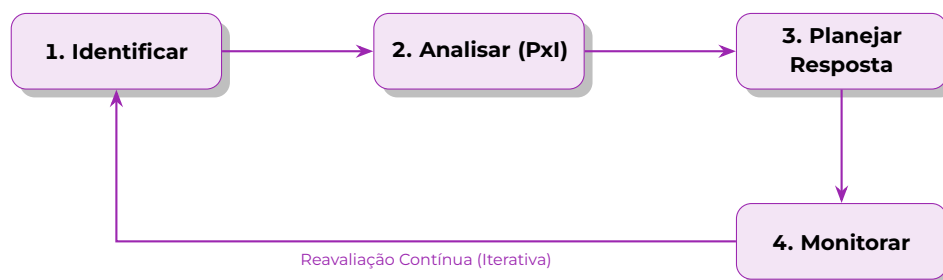


Figure 3: Ciclo iterativo de identificação, análise, resposta e monitoramento de riscos. Fonte: *Elaboração do autor.*

A Figura 3 sintetiza visualmente o conceito discutido nesta seção.

1.3 Riscos Específicos em Projetos de IA (Assistentes LLM)

Quando conectamos um software a uma API de inteligência artificial generativa, acrescentamos riscos específicos relacionados ao comportamento probabilístico do modelo, à integração e aos dados. Sistemas CRUD também podem apresentar riscos de segurança, disponibilidade, privacidade, custo e qualidade; o inventário deve considerar a arquitetura concreta:

- **Alucinação e respostas inadequadas (probabilidade e impacto a estimar):** O LLM pode inventar sintaxe C++ ou apresentar orientação incorreta. Temperatura baixa e *Retrieval-Augmented Generation* (RAG) com fontes selecionadas podem reduzir a variabilidade e melhorar a rastreabilidade,

mas não garantem correção. Avalie respostas em cenários representativos, mostre fontes ao usuário quando aplicável e defina encaminhamento para revisão humana em temas sensíveis.

- **Rate limiting e custos (probabilidade e impacto a estimar):** O provedor pode limitar chamadas ou o uso pode gerar cobranças acima do esperado. Defina cotas e alertas no servidor, registre consumo, use *cache* quando apropriado e aplique *rate limit* no servidor. Controles apenas no *front-end* podem ser contornados.
- **Latência e Timeout (Alta Prob. / Baixo Impacto):** O modelo demorar 15 segundos para responder. *Mitigação:* Usar interfaces assíncronas com indicadores visuais de "carregando" para o usuário não clicar mil vezes no botão.
- **Prompt injection e acesso indevido a dados (probabilidade e impacto a estimar):** Entradas podem tentar alterar instruções ou induzir o sistema a expor dados. Um *system prompt* ajuda a orientar o modelo, mas não é uma fronteira de segurança. A defesa em profundidade combina validação e filtragem no servidor, autenticação e autorização por recurso, isolamento de dados entre usuários, *rate limit*, avaliação de segurança e monitoramento de uso e incidentes.

• Teoria: No Mundo Real: Knight Capital Group (2012)

Em 2012, a Knight Capital Group implantou software que interagiu de forma indevida com mercados norte-americanos. Em cerca de 45 minutos, foram realizadas milhões de ordens, com perda aproximada de US\$ 440 milhões. A empresa recebeu posteriormente um aporte, foi adquirida e não entrou em falência. O episódio envolveu, entre outros fatores, a implantação inconsistente do software em servidores e controles operacionais insuficientes; não deve ser reduzido a uma falha de CI/CD. O caso ilustra a necessidade de mudanças graduais quando possível, validação antes e após a implantação, mecanismos de interrupção e monitoramento operacional.

2 Documentação e Conhecimento Operacional

A **documentação** é um meio de comunicação e preservação de conhecimento que pode reduzir alguns riscos de continuidade e retrabalho. Seu valor depende de estar atualizada, acessível e adequada ao objetivo; ela não substitui controles técnicos, testes, revisão de código ou operação monitorada. Quando apenas uma pessoa sabe configurar ou operar uma parte crítica do sistema, há um Ponto Único de Falha (*Single Point of Failure* – SPOF). A

chamada **dívida documental (Doc Debt)** pode ampliar esse risco ao tornar o conhecimento difícil de localizar ou transferir. Instruções de operação, revisão por pares e treinamento cruzado são respostas complementares.

Sem manutenção, informações importantes podem se perder e o *onboarding* de novos membros pode exigir mais apoio da equipe. Decisões, limitações conhecidas e procedimentos de recuperação merecem registros proporcionais ao risco e à vida útil do projeto.

Uma prática útil é revisar a documentação afetada junto com a mudança. A *Definition of Done (DoD)* pode incluir essa atualização quando ela for relevante ao Incremento, mas a equipe deve definir quais artefatos são necessários em cada caso. Um `README.md` orienta instalação e uso; um *runbook* descreve procedimentos de operação e recuperação; testes verificam comportamento esperado.

2.1 Engenharia Moderna e ADRs

Em alguns projetos, a documentação é tratada como código (*Docs as Code*) e mantida em Markdown (`.md`) no repositório. Para registrar decisões arquiteturais relevantes, a equipe pode usar um **ADR (Architecture Decision Record)**: documento curto que registra contexto, decisão e consequências. ADRs preservam o raciocínio de uma decisão, mas não substituem *runbooks* operacionais, requisitos, testes ou o registro de incidentes.

▷ Exemplo: ADR Exemplo: Escolha do LLM**ADR-001: Seleção do Modelo de Linguagem****Status:** Aceito | **Data:** 2026-03-15**Contexto:** Precisamos de um LLM para o Assistente Sócrático. As opções avaliadas foram modelos de dois provedores, considerando requisitos pedagógicos, custo, privacidade e integração.**Decisão:** Adotaremos o modelo selecionado pela equipe via API REST, por meio de uma camada de integração própria.**Justificativa:**

- Atende aos requisitos de custo e de uso definidos para o protótipo
- Apresentou latência aceitável nos cenários de teste documentados pela equipe
- A camada de integração reduz o acoplamento ao provedor

Trade-off aceito: O modelo escolhido pode ter desempenho diferente em tarefas de código; a equipe acompanhará os resultados da avaliação.**Consequências:** Mudanças de preço, limite ou qualidade podem exigir reavaliação do provedor. O risco e a alternativa de migração são registrados no plano de riscos.**△ Importante: Alerta de Risco: Código Limpo não Substitui Documentação**

Código legível, nomes claros e comentários pontuais facilitam a manutenção, mas não registram todo o contexto. O código tende a mostrar **o que** o sistema faz e **como** faz; ADRs podem registrar **por que** uma decisão foi tomada. README, diagramas, *runbooks* e testes têm finalidades distintas e devem ser usados quando necessários.

► Prática: No GitLab: Riscos e Wiki

Como operacionalizamos segurança no Projeto Integrador? 1. **Gestão de Ameaças:** Crie *Issues* para os riscos mapeados com a etiqueta (*Label*) Risco. Atribua uma "Estratégia de Resposta" nos comentários e marque um responsável. 2. **Instruções de uso:** O arquivo `README.md` pode informar requisitos, instalação, configuração sem segredos e execução. Armazene valores sensíveis fora do repositório e forneça apenas nomes de variáveis ou um arquivo de exemplo sem valores reais. 3. **Base de conhecimento:** A *Wiki* do GitLab ou o repositório podem hospedar ADRs, o Project Charter e atas de reuniões. Procedimentos de implantação, resposta a incidentes e recuperação devem estar em *runbooks* testados e acessíveis a quem opera o sistema.

3 Perguntas Reflexivas

Olhando para o radar de ameaças do seu projeto, me responda:

- 1 Se a equipe decidir não criar testes automatizados porque "tomam muito tempo", isso é um Risco (evento incerto) ou um Problema (certeza de retrabalho)?
- 2 Explique por que a estratégia de "Aceitar" um risco (como a queda temporária de internet do desenvolvedor) pode ser financeiramente mais inteligente do que tentar "Mitigar".
- 3 Qual o risco de depender de "conhecimento tribal", quando a arquitetura do sistema existe apenas na memória de algumas pessoas?

✓ Síntese: Assistente de IA

Ao analisar riscos e conhecimento do projeto:

- **Reduza SPOFs:** Documentação, revisão por pares e treinamento cruzado podem criar redundância operacional.
- **Docs as Code:** Versionar documentação próxima ao código pode facilitar sua revisão e atualização.
- **Gerencie riscos:** Identifique, analise, responda e monitore riscos de acordo com o contexto do projeto.

★ Checkpoint do Projeto: Riscos e Documentação

Verifique as defesas e os artefatos proporcionais ao seu MVP:

- **Riscos mapeados:** Registrem riscos técnicos relevantes em *Issues* no GitLab, com probabilidade, impacto, responsável, resposta e critério de monitoramento.
- **Ponto de atenção – SPOF:** Existe alguma parte da arquitetura que apenas **uma** pessoa sabe operar? Registrem a decisão em ADR quando for arquitetural e criem ou atualizem um *runbook* quando for operacional.
- **Configuração segura:** O README possui instruções de instalação, nomes de variáveis de ambiente e execução local, sem expor senhas, tokens ou chaves de API?

Com riscos identificados e conhecimento organizado, o próximo capítulo apresenta práticas de metodologias ágeis.