

# Monitoria – Métodos Avançados Em Programação

## VISÃO GERAL E OBJETIVO

O objetivo principal desta avaliação é de testar os conhecimentos teóricos dos alunos sobre conceitos avançados em programação, no que importa sobre versionamento de código, computação em nuvem, frameworks, arquitetura de sistemas e padrões de projeto. Os participantes deverão responder às questões em um arquivo Markdown (**README.md**) e submeter suas respostas COLABORANDO EM um repositório no GitHub.

## MATERIAIS NECESSÁRIOS

Os alunos devem ter acesso aos seguintes recursos:

- 1 - Computador com acesso à internet
- 2 - Conta no GitHub
- 3 - Git instalado
- 4 - Editor de Texto
- 5 - Ferramentas de Diagramação (Ex.: Draw.io, LucidChart, Whimsical, etc)

## ORIENTAÇÕES



Para realizar a entrega da avaliação, o participante deverá criar uma nova ramificação no repositório git, e submeter o(s) arquivo(s) por essa branch. Considere que para qualquer submissão será levado em consideração o uso correto de boas práticas, uso correto de inglês, código limpo, refatoração e commits/branches semânticos.

Caso o participante queira submeter algum arquivo além do Readme.md, indique a equivalência da resposta para com o arquivo que será submetido. Ex.: Questão2 -> arquivo: responseToQuestion2.



Uma vez iniciada às 13h, teremos 4h até o encerramento da avaliação, portanto o horário final de entrega será até as 17h.

## Repositório

<https://github.com/GiovannaT/MAP-monitoring>

## ATIVIDADE

1. O que é Git e qual a diferença entre `git merge` e `git rebase`?
2. Explique a importância do uso de pull requests em projetos colaborativos. Como funciona o fluxo típico de um pull request no GitHub?
3. O que é computação em nuvem e quais são suas principais vantagens?
4. O que é um framework e como ele difere de uma biblioteca?
5. Explique a diferença entre uma arquitetura monolítica e uma baseada em microsserviços.
6. O que são APIs REST?
7. O que são padrões de projeto e qual sua importância?
8. O que é um **conflito de merge** em Git? Como resolvê-lo?

### Diretrizes para as respostas das questões 9 e 10:

- Defina cada padrão e explique seu propósito.
  - Compare as principais diferenças entre eles.
  - Dê exemplos práticos de uso para cada um.
  - Desenhe os diagramas UML correspondentes.
9. Os padrões de projeto **Adapter** (estrutural) e **Observer** (comportamental) são amplamente utilizados no desenvolvimento de software.
  10. Explique os padrões de projeto **Template Method** e **Factory Method**.