MATÉRIA - GESTÃO E QUALIDADE DE SOFTWARE – GQS
PROFESSOR - *Calvetti*
ALUNO - João Luiz da Silva – RA 82420546

Atividade: Pesquisa e apresentação, Individualmente em formato de mapa mental, fazer resumo do solicitado entregá-lo em seu GitHub até a véspera da próxima aula.

Temas:
Verifique o significado de todos os termos que você conhece e os que não conhece apresentados neste material de aula, explicando e dando exemplos para cada um deles.

Mapa Mental – Versionamento de Software e Git

1. Versionamento de Software

- **Definição:** Controle de alterações do código ao longo do tempo.
- **Exemplo:** Passar de versão 2.2 para 2.3 de um sistema.
- **Benefícios:**
  - Reverter mudanças
  - Trabalho em equipe simultâneo
  - Auditoria e rastreabilidade
  - Experimentos seguros (branches)

2. Git

- **Definição:** Sistema de versionamento distribuído, criado por Linus Torvalds.
- **Exemplo:** Criar repositório local do TCC sem Internet.
- **Características:**
  - Repositório distribuído
  - Backup completo em cada clone
  - Histórico detalhado de commits

3. GitHub / GitLab / Bitbucket

- **GitHub:** Hospedagem + colaboração (issues, pull requests, CI/CD)
  - Exemplo: PRs para revisar código e rodar testes automáticos.
- **GitLab:** Foco DevOps completo
  - Exemplo: Pipeline CI/CD automatiza testes e deploy.
- **Bitbucket:** Integração com Jira/Atlassian
  - Exemplo: Vincular commits a tarefas do backlog.

4. Rastreabilidade e Auditoria

- **Commit:** Registro de alteração (autor, data, mensagem)
  - Ex.: "Ana – adicionou validação de senha – 01/09/2025"
- **Rastreabilidade:** Saber quem mudou o quê e por que
- **Auditoria:** Histórico formal de alterações

5. Branches

- **Feature Branch:** Nova funcionalidade isolada
  - Ex.: feature/pagamento-pix
- **Release Branch:** Preparação para entrega
  - Ex.: release/2.1 → só correções de bugs
- **Hotfix Branch:** Correção rápida em produção
  - Ex.: hotfix/erro-login
- **Merge:** União de branches (possível conflito)

6. Fluxos de Trabalho (Workflows)

- **Git Flow:** Branches feature, release, hotfix
- **GitHub Flow:** Simples → main + PRs
- **Trunk-Based Development:** Commits frequentes diretamente no main

7. Versionamento Semântico (SemVer)

- **MAJOR:** Quebra de compatibilidade → Ex.: 3.0.0
- **MINOR:** Novas funcionalidades compatíveis → Ex.: 3.2.0
- **PATCH:** Correção de bugs → Ex.: 3.2.5

8. Práticas Modernas

- **Integração Contínua (CI):** Pipeline roda a cada commit
- **Entrega Contínua (CD):** Pipeline roda a cada entrega
- **Code Review:** Revisão obrigatória antes do merge
- **Pull Requests:** Centraliza discussões, testes e aprovação
- **Commits pequenos e frequentes:** Fácil rastreio de bugs
- **Mensagens claras:** Histórico compreensível
- **Branches curtos:** Menos conflitos

9. Ferramentas Visuais

- **GitKraken:** Grafo visual de branches → facilita merges
- **Sourcetree:** Integra commits a tarefas Jira
- **VS Code + Git:** Commit, branch e sync direto no editor

10. Práticas DevOps e Automatização

- **Pipelines automatizados (CI/CD):** Build + testes + deploy
- **Infraestrutura como Código (IaC):** Arquivos Terraform/Ansible versionados
- **Deploy automatizado:** Push → produção automática

11. Integração com Métodos Ágeis

- **Scrum + Git:** Branches organizam sprints → merge no final
- **Kanban + GitHub Flow:** Branch por tarefa → PR → merge main
- **XP + TDD + CI/CD:** Testes automáticos antes do merge → qualidade garantida

12. Riscos sem versionamento

- Merge malfeito →software instável
- Sem versionamento →retrabalho e perda de histórico
- Branches descontroladas → confusão e atrasos