TEMA 1 – Fundamentos do Controle de Versão e História do Git

Objetivo: Compreender o conceito de controle de versão e sua evolução até o Git moderno.

Perguntas:

1. O que é controle de versão e por que ele é essencial no desenvolvimento de software?
2. Qual a diferença entre os modelos Local, Centralizado e Distribuído?
3. Como e por que Linus Torvalds criou o Git em 2005?
4. Quais são os conceitos básicos de commits, snapshots e hash SHA-1?
5. O que torna o Git mais eficiente e seguro do que sistemas anteriores como SVN?

Projeto real para análise:

- Pesquisar o **repositório oficial do Kernel Linux** no GitHub e analisar como o versionamento é estruturado (branches, histórico e contribuidores).

Prática GitHub:

- Criar um repositório chamado **fundamentos-git-equipe1.**
- Fazer commits simulando versões evolutivas de um arquivo **README.md** explicando o histórico do Git.

TEMA 2 – Instalação, Configuração e Conceitos Fundamentais

Objetivo: Aprender a instalar, configurar e criar repositórios locais e remotos.

Perguntas:

1. Quais são os passos para instalar o Git em Windows, Linux e macOS?
2. Como configurar o nome, e-mail e editor padrão do Git?
3. O que são as áreas *Working Directory*, *Staging Area* e *Repository*?
4. Quais os comandos básicos para iniciar e conectar repositórios (init, clone, remote, push)?

5. O que é e qual a função do arquivo .gitignore?

Projeto real para análise:

- Avaliar um repositório open-source que utiliza **.gitignore** de forma eficiente (ex.: React, Node.js).

Prática GitHub:

- Criar um repositório **configuracao-git-equipe2** com as configurações iniciais e um **.gitignore** funcional.

TEMA 3 – Ciclo de Vida dos Arquivos e Comandos Essenciais

Objetivo: Dominar o ciclo completo de versionamento local e remoto.

Perguntas:

1. Quais são os estados de um arquivo (untracked, modified, staged, committed)?
2. Quais comandos são usados para mover arquivos entre esses estados?
3. Como visualizar histórico, diffs e reverter commits?
4. Como e por que usar mensagens de commit claras e padronizadas?
5. Quais boas práticas garantem um histórico limpo e organizado?

Projeto real para análise:

- Examinar o histórico de commits de um projeto open-source (ex.: *Flask*, *Arduino IDE*) e identificar padrões de mensagens de commit.

Prática GitHub:

- Criar o repositório **ciclo-git-equipe3** e fazer commits intencionais com boas mensagens e tags.

TEMA 4 – Branches, GitFlow e Colaboração no GitHub

Objetivo: Aprender a criar e gerenciar branches, pull requests e merges.

Perguntas:

1. O que é uma branch e qual a sua importância?
2. Qual a diferença entre merge e rebase?
3. Como funciona o modelo GitFlow e quais suas vantagens?
4. Como criar e revisar Pull Requests no GitHub?
5. Quais boas práticas evitam conflitos em projetos colaborativos?

Projeto real para análise:

- Estudar o fluxo de branches e PRs do projeto *TensorFlow* ou *FreeCodeCamp* no GitHub.

Prática GitHub:

- Criar o repositório **gitflow-equipe4** e simular o GitFlow (branch **develop, feature/, hotfix/**).

TEMA 5 – Integração Contínua (CI/CD) com GitHub Actions

Objetivo: Entender como automatizar o processo de integração e entrega contínua.

Perguntas:

1. O que é CI/CD e por que é usado?
2. Quais são os principais componentes de um workflow no GitHub Actions?
3. Como criar um pipeline simples que execute testes automáticos?
4. Quais eventos (triggers) podem iniciar um workflow?
5. Como visualizar logs e falhas no painel “Actions”?

Projeto real para análise:

- Examinar o repositório do *Django* ou *Node.js* e observar workflows YAML no diretório **.github/workflows.**

Prática GitHub:

- Criar repositório **ci-cd-equipe5** com um workflow YAML simples que execute npm test ou pytest.

TEMA 6 – GitHub Actions Avançado e DevOps

Objetivo: Explorar pipelines mais complexos e integração com cloud.

Perguntas:

1. Quais são os conceitos de ambientes, secrets e triggers múltiplos?
2. Como integrar GitHub Actions com AWS, Azure ou Docker Hub?
3. O que é o ciclo DevOps e como o Git se encaixa nele?
4. Quais boas práticas aumentam a segurança dos pipelines?
5. Como monitorar e versionar o pipeline em si?

Projeto real para análise:

- Estudar um pipeline completo do *Microsoft Azure SDK* ou *AWS Samples*.

Prática GitHub:

- Criar repositório **devops-equipe6** com pipeline de build + lint + deploy simulado.

TEMA 7 – Git Avançado e Segurança

Objetivo: Garantir que o versionamento ocorra de forma segura e rastreável.

Perguntas:

1. O que são chaves SSH e tokens pessoais de acesso (PAT)?
2. Como criar e vincular uma chave SSH ao GitHub?
3. Por que o GitHub substituiu senhas por tokens?

4. Quais práticas garantem segurança em equipes (branch protection, 2FA)?

5. Como auditar alterações e acessos em um repositório?

Projeto real para análise:

- Pesquisar como empresas open-source configuram *branch protection* e autenticação (ex.: Mozilla, Apache).

Prática GitHub:

- Criar repositório **seguranca-git-equipe7** com README explicando SSH e PAT.

TEMA 8 – Tendências Recentes e IA no Versionamento

Objetivo: Explorar o impacto da inteligência artificial e da automação no futuro do Git e GitHub.

Perguntas:

1. O que é o GitHub Copilot e como ele auxilia no desenvolvimento?

2. Como funciona o GitHub Codespaces e quais suas vantagens?

3. Como a IA está sendo usada para sugerir commits e revisar código?

4. O que é o DVC e por que é importante em IA e ciência de dados?

5. Como essas tecnologias estão moldando o futuro do versionamento?

Projeto real para análise:

- Estudar repositórios que utilizam Copilot ou Codespaces (ex.: GitHub Copilot Labs, OpenAI examples).

Prática GitHub:

- Criar repositório **tendencias-git-equipe8** com README descrevendo uso de IA e automação em versionamento.