Atividade 1: Desvendando a História dos Projetos: A Lógica do Git

1. Análise de um Histórico de Commits (Individual)

Repositório analisado: https://github.com/twbs/bootstrap (Projeto Bootstrap)

Commits escolhidos:

Commit 1:

- Mensagem: “Fix typo in alert.scss”
- Arquivos modificados: scss/_alert.scss
- Intenção: Corrigir um erro de digitação (ex: erro de estilo ou variável incorreta).
- Contribuição: Melhora a qualidade do código, evita problemas futuros.

Commit 2:

- Mensagem: “Update docs for new grid system”
- Arquivos modificados: Arquivos de documentação em /docs
- Intenção: Informar mudanças na estrutura de grid.
- Contribuição: Ajuda usuários e desenvolvedores a entenderem novas funcionalidades.

Commit 3:

- Mensagem: “Add dark mode support”
- Arquivos modificados: Vários arquivos de CSS/JS e configuração.
- Intenção: Adicionar funcionalidade.

- Contribuição: Incrementa a experiência do usuário, mostra evolução do projeto.

Sequência dos commits:

Esses commits mostram uma progressão típica: pequenos ajustes → documentação → nova funcionalidade. Essa ordem contribui para um desenvolvimento sustentável.

---

2. A Importância das Mensagens de Commit (Individual)

- Importância: Mensagens claras tornam o histórico legível e útil. Facilitam a revisão e manutenção do projeto.

- Mensagens bem escritas ajudam a:
    - Identificar rapidamente o que foi alterado.
    - Entender o propósito da mudança.
    - Evitar mal-entendidos entre desenvolvedores.

- Exemplos de boas mensagens:
    - ☑ “Fix login redirect bug when session expires”
    - ☑ “Add responsive navbar for mobile view”

- Exemplos que podem melhorar:
    - ✕ “Update” → Vago, não diz o que foi atualizado.
    - ✕ “changes made” → Não ajuda em nada na revisão.

---

3. O Conceito de Branching (Individual)

- O que é uma branch no Git?
    - É uma ramificação do projeto principal onde desenvolvedores podem trabalhar isoladamente sem afetar a linha principal (geralmente chamada de main ou master).
- Finalidade:
    - Permite o desenvolvimento paralelo de novas funcionalidades, correções de bugs ou testes de ideias.
- Benefícios:
    - Evita conflitos diretos com a versão principal.
    - Garante estabilidade no código principal.
    - Ajuda equipes a trabalhar simultaneamente.
- Exemplo de cenário:
    - Um time está desenvolvendo uma nova interface. Criar a branch nova-interface permite fazer alterações extensas sem interromper o funcionamento da versão atual. Após os testes, ela pode ser integrada à branch principal com segurança.