Aluna: Maria Eduarda Meirelles dos Santos Passos

Turma: 2.3

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

1. Análise de um Histórico de Commits (Individual)

Repositório analisado:

https://github.com/facebook/react (Projeto de biblioteca de interface do Facebook)

Commits escolhidos:

1. Fix typo in comments

2. Improve error messages in development

3. Remove unused dependencies

Análise dos commits:

Commit Mensagem Arquivos modificados Contribuição

1 Corrige erro de digitação em comentários do código. Comentários em arquivos JS. Melhora a legibilidade e manutenção do código.

2 Melhora mensagens de erro para desenvolvedores. Módulos de mensagens e logs. Ajuda a depurar erros mais facilmente durante o desenvolvimento.

3 Remove dependências não utilizadas. package.json, yarn.lock Torna o projeto mais leve e organizado.

Sequência e evolução:

Esses commits mostram uma preocupação constante com a clareza e organização do código. As mudanças são pequenas, mas acumuladas, elas ajudam a manter o projeto sustentável e eficiente ao longo do tempo.

---

2. A Importância das Mensagens de Commit (Individual)

Importância das mensagens claras:

Mensagens de commit bem escritas são fundamentais para manter o histórico de mudanças compreensível. Elas permitem que qualquer desenvolvedor entenda o que foi alterado sem precisar ler o código inteiro. Isso também facilita revisões de código, identificação de bugs e colaboração.

Boas práticas observadas:

Utilizar verbos no imperativo (“Adiciona”, “Corrige”, “Remove”).

Ser direto e específico.

Evitar mensagens vagas como “update” ou “coisas”.

Exemplos de mensagens eficazes:

Add dark mode toggle button

Fix crash when user profile is empty

Update README with new installation guide

Exemplos que poderiam ser melhorados:

ajustes → Melhor: Adjust padding in navigation menu

coisas feitas → Melhor: Implement search bar functionality on homepage

Justificativa:

Mensagens vagas atrapalham o entendimento coletivo. Mensagens específicas indicam exatamente o que foi feito e ajudam no rastreio de erros e melhorias.

---

3. O Conceito de Branching (Individual)

O que é branch no Git:

Branch é um “ramo” de desenvolvimento. É como criar uma linha paralela onde você pode fazer alterações sem afetar a linha principal (geralmente chamada de main ou master).

Finalidade principal:

Permitir o desenvolvimento de novas funcionalidades, correções de bugs ou experimentos sem comprometer o projeto principal.

Vantagens:

Permite que times trabalhem em paralelo.

Reduz o risco de conflitos diretos no código principal.

Ajuda a organizar melhor os ciclos de desenvolvimento (ex: feature/login, bugfix/header, hotfix/security).

Cenário hipotético:

Imagine que um time está criando um novo sistema de comentários para um blog. Criar uma branch chamada feature/comments permite desenvolver, testar e revisar esse sistema de forma isolada. Só depois de pronto e aprovado ele será mesclado ao main, garantindo estabilidade.

---