SERVIÇO NACIONAL DE APRENDIZAGEM INDUSTRIAL – SENAI

Turma: T DESI 2024/1 N1
Unidade Curricular: Programação de Aplicativos
Professor: Gustavo Garcia de Amo

Alunos: Carlos Jhonne e Brendha Victória

1. O que é versionamento de software e por que ele é importante?

Versionamento de software é basicamente um jeito de registrar todas as mudanças feitas no código ao longo do tempo. Ele é super importante porque permite que a gente acompanhe as alterações, volte para uma versão antiga se algo der errado e colabore com outras pessoas sem bagunçar o projeto. Em resumo, ele organiza o desenvolvimento e facilita muito o trabalho em equipe.

2. Por que usar sistemas de controle de versão, tipo o Git, em projetos de desenvolvimento?

Algumas vantagens são:

- Rastrear cada alteração e poder voltar atrás, se precisar.
- Facilitar o trabalho colaborativo, então vários desenvolvedores podem editar o código sem pisar no trabalho um do outro.
- Guardar um histórico completo das mudanças, ajudando a resolver problemas futuros e a entender o que foi feito em cada fase do projeto.

3. Quais são os principais sistemas de controle de versão hoje?

Os principais são:

- Git: O mais usado atualmente, popular e muito eficiente.
- SVN (Subversion): Ainda bastante comum, especialmente em empresas mais tradicionais.
- Mercurial: Parecido com o Git, mas menos usado.
- Outros como Perforce e ClearCase, geralmente em grandes empresas.

4. Como criar e gerenciar um repositório em um sistema de controle de versão?

Criação: Você começa criando um repositório local no seu projeto (ex: git init).

- Conexão remota: Se quiser compartilhar, liga com um repositório remoto (ex: GitHub, GitLab).
- Adição e commits: Adiciona arquivos e registra as mudanças com um commit.
- Sincronização: Para trabalhar em equipe, envia (push) e recebe (pull) atualizações do repositório remoto.

5. O que são commits e como eles ajudam?

Commits são registros que você cria toda vez que salva uma versão nova do código. Eles são super úteis porque guardam o que foi alterado e por que, então, dá pra ver o histórico do projeto, identificar bugs, ou até voltar a uma versão anterior se precisar.

6. O que são branches e quais os benefícios de trabalhar com eles?

- Branches são "ramificações" onde você cria um novo caminho de desenvolvimento sem afetar a linha principal do projeto. São muito úteis porque:
- Isolam as mudanças, o que evita conflitos no código.
- Permitem desenvolver várias funcionalidades em paralelo.
- Facilitam a revisão e testes antes de unir tudo na linha principal.

7. O que é um merge e como funciona?

Merge é o processo de juntar as mudanças de uma branch com outra (geralmente a principal). Ele cria um novo commit que incorpora as alterações de uma branch para outra. No Git, é feito com o comando git merge.

8. Quais desafios podem surgir quando vários devs trabalham juntos e como o versionamento ajuda?

Desafios comuns:

- Conflitos de código: Quando duas pessoas mudam a mesma parte do código.
- Sobrescrever trabalho: Sem controle, um desenvolvedor pode sobrescrever o código de outro.
- Comunicação das mudanças: É fácil se perder sem uma ferramenta que registre quem fez o quê. O versionamento resolve isso porque guarda tudo, permite ver e resolver conflitos e organiza o trabalho em equipe.

9. Melhores práticas para escrever mensagens de commit que façam sentido

Seja direto e claro: Descreva o que foi feito e por quê.

- Use presente: Escreva “Corrige bug” em vez de “Corrigiu bug”.
- Refira tarefas ou issues: Se ajuda no contexto, cite o número da tarefa ou issue.
- Seja breve: Uma linha geralmente é o ideal; se precisar, adicione detalhes na linha seguinte.

10. Como resolver conflitos de merge?

Para resolver:

- Identifique o conflito: O sistema mostra os arquivos com conflitos.
- Revise e ajuste manualmente: Edite para manter o que faz sentido.
- Teste o código atualizado: Certifique-se de que o merge não quebrou nada.
- Finalize: Salve e faça o commit final do merge.