Resumo da aula 12-06-25

Resumo da Aula: Gerência de Configuração e Git/GitHub

A aula abordou a importância da Gerência de Configuração (GC) no desenvolvimento de software, focando em sistemas de controle de versão (como o Git) e sistemas de controle de modificações (como Issues). O objetivo é garantir um mesmo nível de conhecimento sobre GC antes de introduzir ferramentas.

Tópicos Chave e Falas Importantes:

- **Gerência de Configuração (GC):**
  - É uma área da Engenharia de Software preocupada em gerenciar a evolução de artefatos de software.
  - Gerencia o controle de alterações do primeiro ao centésimo dia de um software em desenvolvimento.
  - Evita problemas como "versão final revisão 22, comentários, correções" de trabalhos sem controle.
- **Sistemas de Controle de Versão (SCM - Software Configuration Management):**
  - **Função:** Controlam e registram modificações.
  - **Commit:** Ato de enviar modificações.
    - Registra: autor, horário (dia, minuto, segundo), e a alteração feita no arquivo (o que foi acrescentado ou tirado).
    - Permite um "controle extremamente preciso".
  - **Git:** O sistema de controle de versão mais famoso.
    - Criado por Linus Torvalds (criador do Linux) para ter um sistema distribuído, pois os existentes (CVS, Subversion) não atendiam suas necessidades.
    - **Distribuído:** Cada desenvolvedor mantém uma cópia completa do projeto em sua máquina. Isso permite trabalhar offline e fazer commits, integrando as modificações posteriormente.
    - "Conflito do Git é conflito físico": ocorre quando há alteração na mesma linha/região do código, não entende sintaxe ou semântica.
- **Plataformas de Repositório Centralizado:**
  - **GitHub:** Plataforma web mais conhecida para versionamento de código e repositório centralizado na internet.
    - Adquirido pela Microsoft.
    - Permite contas gratuitas e repositórios privados.
    - Usado para análise de repositórios, investigando métricas como número de *merges*, conflitos, etc.
    - Pesquisas mostram que: quanto mais pessoas trabalham nos mesmos ramos, maior a chance de conflito; quanto mais arquivos são mexidos, maior a chance de conflito.
    - Projetos no GitHub frequentemente renovam seus especialistas ao longo do tempo.
  - **Repositório:** Onde os artefatos ficam armazenados, como um "banco de dados" ou um "servidor na nuvem".

    - **Boa Prática:** Manter o repositório sincronizado (fazer pull frequentemente) evita conflitos. Quanto maior o tempo de isolamento, maiores as chances de conflito.
- **Sistemas de Controle de Modificações (Issues):**
    - Segundo tipo de sistema mais conhecido em Gerência de Configuração, depois do controle de versão.
    - Também chamados de "CR" (Change Requesters).
    - Gerenciam problemas (*bugs*), melhorias (*enhancements*) e outras tarefas a serem feitas no código.
    - Exemplos: GitHub Projects (integrado com o controle de versão).
    - **Integração com Commits:** É uma boa prática vincular commits a *issues* (ex: #número_da_issue na mensagem do commit).
        - Cria rastreabilidade: um link mostra que a demanda (card/issue) foi atendida por aquele commit, e o commit mostra todas as modificações.
        - Oferece transparência total do trabalho.
        - É uma "documentação mínima" e fácil de fazer.
        - A ordem é: primeiro criar a *issue* (demanda/card) no projeto, depois fazer o commit da alteração.

Perguntas Específicas:

1. **Qual a principal função de um sistema de controle de versão?** A principal função de um sistema de controle de versão é gerenciar e registrar a evolução e as modificações dos artefatos de um software ao longo do tempo, mantendo um controle preciso de quem fez o quê, quando e o que foi alterado.
2. **Quem é o criador do Git e por que ele o criou?** O Git foi criado por Linus Torvalds, o mesmo criador do Linux. Ele o criou porque achava que nenhum sistema de controle de versão existente na época (como CVS ou Subversion) atendia à sua necessidade de ter um sistema distribuído, ideal para a forma como o Linux é desenvolvido com contribuições de várias pessoas.
3. **Qual a diferença entre "conflito físico" e "conflito sintático/semântico" no contexto do Git?** O Git lida apenas com **conflitos físicos** : quando diferentes modificações ocorrem na mesma região (mesmas linhas) de um arquivo com textos ou caracteres diferentes. Ele não é "inteligente" para entender **conflitos sintáticos ou semânticos** , que seriam problemas relacionados à lógica da linguagem ou à funcionalidade do código, mesmo que em partes diferentes do arquivo.