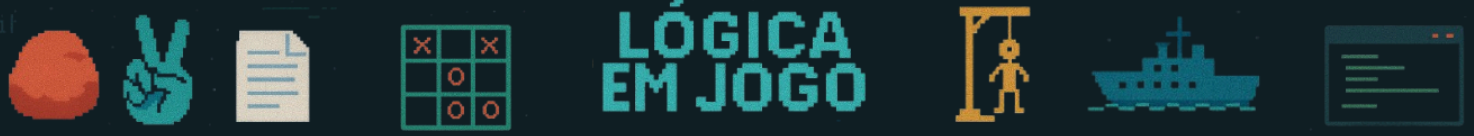

INSTITUTO FEDERAL
Minas Gerais

Sessão 01

O projeto, os jogos e o Github

Professor Thiago Goveia

Objetivos da sessão:

- Compreender a proposta e metodologia do projeto
- Conhecer os quatro jogos que serão implementados
- Entender a importância do controle de versão no desenvolvimento de software
- Criar conta no GitHub e realizar primeiros passos na plataforma

Cronograma:

- **20min:** Abertura e introdução
- **20min:** Apresentação do projeto no GitHub
- **20min:** Organização das equipes
- **40min:** Tutorial: Gerenciando modificações no jogo “Par ou ímpar” com o GitHub

Materiais de apoio:

- Apresentação do projeto (LEJ - Visão geral.pdf)
- Repositório do projeto

Tópicos:

1. Introdução ao GitHub
    - O que é controle de versão?
    - O que é Git? O que é GitHub?
    - Por que usar?
    - Conceitos iniciais: Repositório, Commit e Tag
2. Criação da conta pessoal
3. Criação do seu 1º repositório (teste)
    - O arquivo README.md
4. Compartilhamento do repositório
5. Rastreamento de mudanças por meio de commits
6. Rastreamento de versões por meio de tags
7. Exemplo prático.

Conteúdo da sessão:

INSTITUTO FEDERAL
Minas Gerais

1. Introdução ao GitHub

1.1 O que é controle de versão?

O controle de versão é uma forma de **acompanhar todas as mudanças feitas em um projeto** ao longo do tempo. Ele permite:

- Salvar versões diferentes de um mesmo arquivo (como um "histórico" do código).
- Voltar para versões anteriores se algo der errado.
- Saber **quem fez o quê e quando** em projetos colaborativos.

Um bom exemplo: pense em um **Google Drive com histórico**, mas para código.

1.2 O que é Git? O que é GitHub?

- **Git** é uma ferramenta de controle de versão criada por Linus Torvalds (criador do Linux).

- **GitHub** é uma plataforma online onde você pode **salvar seus repositórios Git na nuvem**, **compartilhar projetos**, **trabalhar em equipe** e **mostrar seu portfólio**.

1.3 Por que usar?

- Trabalhar em equipe sem bagunça
- Revisar o histórico de mudanças
- Aprender boas práticas de desenvolvimento
- Conquistar visibilidade (empregadores e colegas podem ver seu código)
- Evitar perda de dados

1.4 Conceitos iniciais:

- **Repositório (repo)** é onde todo o seu projeto vive:
  - É uma pasta com arquivos, histórico e configurações de controle de versão.
- **Commit**: Um commit é como tirar uma foto do seu projeto num determinado momento.
  - Guarda quais arquivos foram modificados e uma mensagem explicando a mudança.

    - É parte essencial do histórico do repositório.
- **Tag:** Uma tag marca uma versão especial do projeto (ex: v1.0).
    - Serve para indicar momentos importantes: primeira versão finalizada, versão entregue, etc.
    - Ajuda a rastrear o progresso do projeto.

2. Criação da conta pessoal

1. Acesse https://github.com
2. Clique em **Sign up** (ou "Cadastre-se")
3. Escolha um **nome de usuário**, email e senha
4. Confirme o e-mail

3. Criação do seu 1º repositório (teste)

1. Após criar a conta, clique em **New repository**.
2. Nomeie o repositório como **teste**.
3. Marque a opção "**Add a README file**" (criar com README).
4. Clique em **Create repository**.

3.1 O arquivo README.md

- Arquivo de texto com formatação Markdown (.md)
- Explica o que é o projeto, como usar, quem fez, etc.
- É o **cartão de visitas do seu projeto**.
- Exemplo simples:

```
# Meu primeiro repositório (# indica um título)
Este repositório é parte do projeto Lógica em Jogo.
Feito por [Seu Nome].
```

4. Compartilhamento do repositório

- Dentro do repositório **teste**, acesse a aba **Settings**.
- No menu à esquerda, acesso **Collaborators**.
- Inclua a sua dupla como colaborador do seu projeto, informando o e-mail ou o nome de usuário.

5. Rastreamento de mudanças por meio de commits

- Quando você modifica um arquivo (como o README) e salva no GitHub, pode escrever uma mensagem explicando a mudança.
- Exemplos de mensagens de commit:

    - Atualizei o título
    - Corrigi erro de digitação
    - Adicionei instruções para rodar o jogo
- Essas mensagens ajudam você e outras pessoas a **entenderem o que foi feito em cada etapa**.

6. Rastreamento de versões por meio de tags

- Após concluir um jogo, por exemplo, você pode marcar o repositório com uma **tag** como:
    - **v1.0.0** (versão funcional)
    - **v2.0.0** (versão com melhorias)
- Tags tornam fácil **voltar a uma versão estável**, mesmo que o projeto tenha mudado muito depois.

7. Exemplo prático:

8. Em dupla, escolham um dos repositórios “teste” para trabalharem.
9. Crie um arquivo main.c no repositório teste
10. Aluno 1:
    - Inclua neste arquivo a lógica de um programa que lê um número e mostre na tela “1” se ele for par ou “0” se for ímpar.
    - Faça o commit da sua modificação
11. Aluno 2:
    - Da sua máquina, faça uma melhoria no código implementado pelo Aluno 1
    - Faça o commit da sua modificação
12. Aluno 1:
    - Verifique se há alguma melhoria que ainda possa ser feita. Se sim, faça-a.
    - Faça o commit da sua modificação
13. Após as modificações, gere uma tag v0.0.0, com a descrição desta versão.