Apresentação síncrona: UC00617
Utilização dos Serviços Git e GitHub (25 Horas)

João Silva / Talentus

July 3, 2025

Conteúdo do Curso

Sessão 1: Desmistificando o Controlo de Versões e Primeiros Passos com Git

Sessão 2: GitHub e a Ponte para o Mundo Remoto

Sessão 3: Gerir o Histórico e Clientes Gráficos

Sessão 4: Branches e Colaboração

Sessão 5: Resolução de Conflitos e Boas Práticas Profissionais

Trabalho Prático Final

Introdução ao Controlo de Versões:

O que é e porquê é indispensável (rastreabilidade, colaboração).
Sistemas Centralizados vs. Distribuídos (DVCS).
Por que usar **Git**.

Instalação e Configuração do Git:

Guia passo a passo para Windows, macOS, Linux.
Configuração inicial (`user.name`, `user.email`).

Primeiros Comandos Essenciais:

Ciclo de vida: Working Directory, Staging Area, Local Repository.
Comandos: `git init`, `git status`, `git add`, `git commit`, `git log`.

Sessão 1: Exercício Prático

O seu Primeiro Repositório Git Local

Criar e inicializar uma pasta como repositório Git.

Criar e modificar ficheiros (`info_pessoal.txt`, `hobbies.txt`).

Realizar múltiplos commits com mensagens descritivas.

Explorar o histórico de commits com `git log`.

Introdução ao GitHub:

O que é e funcionalidades chave (hospedagem, Pull Requests, Issues).
Criação de conta gratuita e visão geral da interface.

Criar e Gerir Repositórios Remotos:

Definição de repositório remoto.
Criar um novo repositório no GitHub (opções: público/privado, README, gitignore).
Associar repositório local a remoto (`git remote add origin`).

Sincronização de Repositórios:

Enviar alterações: `git push`.
Obter e integrar alterações: `git pull`.
Fluxo de trabalho básico de sincronização.

Sessão 2: Exercício Prático

Publicando o seu Projeto e Clonando

Publicar um repositório local existente no GitHub.

Clonar o seu próprio repositório (simulando um colega).

Fazer alterações no clone e enviá-las para o remoto.

Puxar as alterações de volta para o repositório original.

Análise e Edição do Histórico de Commits:

Opções avançadas de `git log` (`--oneline`, `--graph`, `-p`).
Desfazer alterações com `git revert` (seguro).
Marcar versões com `git tag` (`git push origin --tags`).

Ficheiros `README.md`:

Importância para documentação do projeto.
Sintaxe básica de Markdown (títulos, listas, negrito, links, código).

Clientes Gráficos Git:

O que são, vantagens e desvantagens.
Demonstração de ferramentas populares (GitKraken, Sourcetree, extensões VS Code).
Realizar operações essenciais via interface gráfica.

Sessão 3: Exercício Prático

Documentar e Experimentar GUI

Explorar o histórico de commits.

Marcar uma versão importante com uma tag.

Criar um `README.md` profissional usando Markdown.

Realizar operações Git básicas usando um cliente gráfico à escolha.

Conceitos de Branches:

O que são branches e por que usá-las (isolamento, colaboração paralela).
Comandos de gestão: `git branch`, `git checkout`, `git switch`, `git branch -d`.
Fluxo de trabalho básico com branches.

Fusão de Branches (`git merge`):

Processo de combinação de alterações.
Tipos de merge: Fast-Forward vs. 3-Way Merge (recursive).

Colaboração com Pull Requests (PRs):

O papel dos PRs no GitHub (revisão de código, discussão, testes).
Ciclo de vida de um Pull Request (demonstração na UI do GitHub).

Sessão 4: Exercício Prático

Simulando Desenvolvimento em Equipa

Criar branches para novas funcionalidades e correções (`feat/`, `fix/`).

Desenvolver em branches separadas.

Fundir branches na `main` usando `git merge`.

Visualizar o histórico com branches e merges.

Resolução de Conflitos:

O que são e como ocorrem conflitos de merge.

Identificação: mensagens do terminal, `git status`, marcas nos ficheiros.

Processo de resolução manual na linha de comando e via cliente gráfico.

Boas Práticas e Colaboração Eficaz:

Comunicação, frequência de commits, `git pull` antes de trabalhar e antes de `push`.

Importância da revisão de código.

Etiqueta no desenvolvimento colaborativo.

Normas e Regulamentos Aplicáveis:

Propriedade intelectual e licenças de software (MIT, GPL).

Segurança e privacidade: **NUNCA commitar credenciais**.

Uso de `.gitignore` para excluir ficheiros sensíveis/temporários.

Respeito por regras de contribuição (`CONTRIBUTING.md`).

Sessão 5: Exercício Prático

Desafio de Resolução de Conflitos

Preparar um cenário onde dois formandos (ou o formando e o formador) alteram a mesma linha em branches diferentes.

Provocar um conflito de merge intencionalmente.

Resolver o conflito passo a passo, usando o editor de texto ou um cliente gráfico.

Fazer o commit do merge e verificar o histórico.

Trabalho Prático Final

Consolidação de Competências

Criar um **novo repositório Git** para um "Mini-Projeto Pessoal" (ex: site, calculadora, diário de estudos).

Publicar no **GitHub**.

Demonstrar **histórico de commits limpo e significativo**.

Utilizar **pelo menos duas branches** para desenvolvimento, fundindo-as na `main`.

Incluir um `README.md` **bem estruturado** com Markdown.

Opcional/Desafio: Simular e resolver um pequeno conflito no seu próprio projeto.

Adicionar um ficheiro `.gitignore`.

Escolher e adicionar uma **Licença** (ex: MIT License).

Entrega: Link do repositório GitHub do projeto.