

Skills e Versionamento

<LAB365>

SENAI

AGENDA

- Apresentação
- Skills
- Versionamento
- Git

Apresentação

Conte um pouco sobre você:

- Nome
- Idade
- Onde mora

- O que faz atualmente
- Objetivos pessoais
- Expectativas no curso



Considerações iniciais

Para tirar o melhor proveito dos nossos estudos, vamos estabelecer algumas regras:

- Não é uma competição
- Não durma com dúvidas
- Trabalhe em equipe
- Não tenha medo de errar
- Cada um tem seu tempo - respeite

Soft-skills

Hard-skills



[Mentimeter](#)

Soft-skills

- Difícil de qualificar
- Aptidões mentais (sociocomportamental)
 - Proatividade
 - Resolução de conflitos
 - Persuasão
 - Senso de liderança

Hard-skills

- Fácil de qualificar
- Aptidões técnicas do profissional
 - HTML
 - JavaScript
 - CSS
 - SQL

Importância

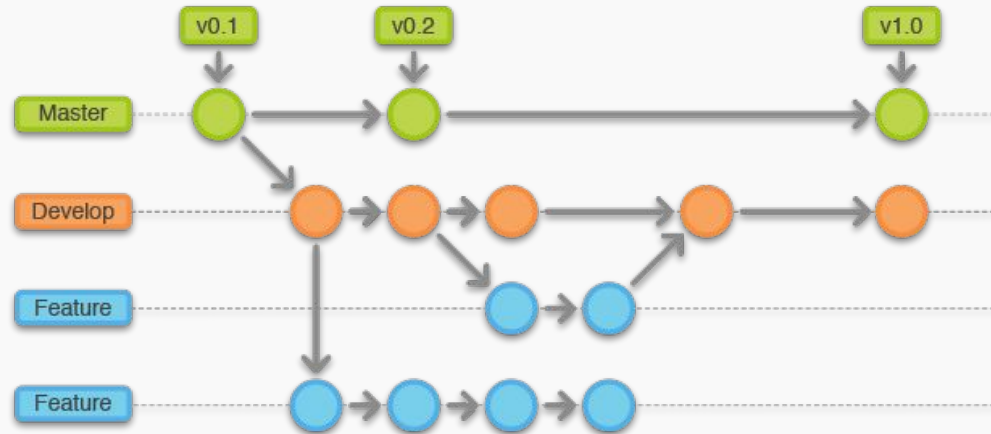
Em toda empresa os diferentes cargos exigem diferentes domínios de habilidades.

- Atendimento ao cliente demanda inteligência emocional
- Suporte demanda de entendimento sobre questões técnicas

Isso garante que todo colaborador seja encaminhado às funções certas, não somente com base em seus conhecimentos gerais.



Versionamento



Versionamento

- O versionamento é o gerenciamento de versões diferentes de um documento de texto qualquer;
- Não precisa ser apenas para código;
- É utilizado no desenvolvimento de software para controlar as diferentes versões e histórico de desenvolvimento do código.

Versionamento

- Imagine que você está escrevendo um programa e precisa editar um arquivo, porém, você quer **manter** uma cópia da **versão anterior** desse arquivo como segurança.
- Desse modo, você pode retomar seu arquivo da versão anterior, caso algo em sua alteração dê errado, ou por algum motivo você queira voltar para a versão anterior.

Versionamento

Vamos supor que o seu arquivo se chama **index.html**. Você já construiu bastante coisa, e não gostaria de perder seu progresso caso algo dê errado. Por isso, você cria uma cópia desse arquivo e a chama de **index_estavel.html**, e continua editando o arquivo **index.html**.

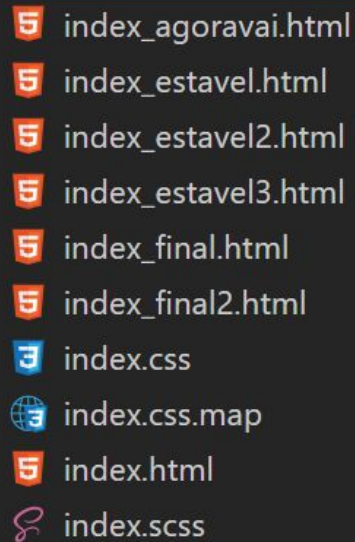
Desta forma você versionou seu código, pois agora existem 2 versões do arquivo index.html.











E qual o problema de versionar código criando cópias físicas para cada alteração?

Imagine que você continuou trabalhando nesse mesmo programa por mais alguns dias e foi criando várias cópias desse arquivo.

Versionamento

Quando você olha para sua pasta, ela está assim:

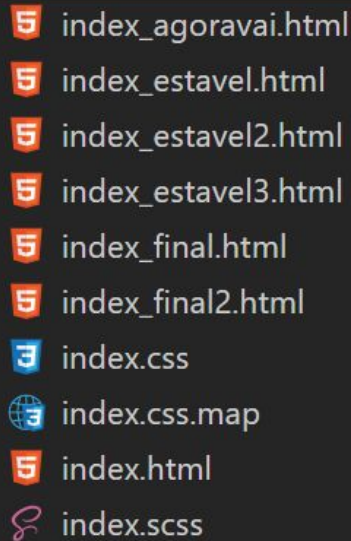


-  index_agoravai.html
-  index_estavel.html
-  index_estavel2.html
-  index_estavel3.html
-  index_final.html
-  index_final2.html
-  index.css
-  index.css.map
-  index.html
-  index.scss

Versionamento

Quando você olha para sua pasta, ela está assim:

Alguns dias depois, você volta a mexer nesse mesmo projeto e se depara com alguns problemas:



A screenshot of a file explorer window with a dark background. It displays a list of files, each preceded by a small icon representing its type. The files are: index_agoravai.html (HTML icon), index_estavel.html (HTML icon), index_estavel2.html (HTML icon), index_estavel3.html (HTML icon), index_final.html (HTML icon), index_final2.html (HTML icon), index.css (CSS icon), index.css.map (Map icon), index.html (HTML icon), and index.scss (SCSS icon).

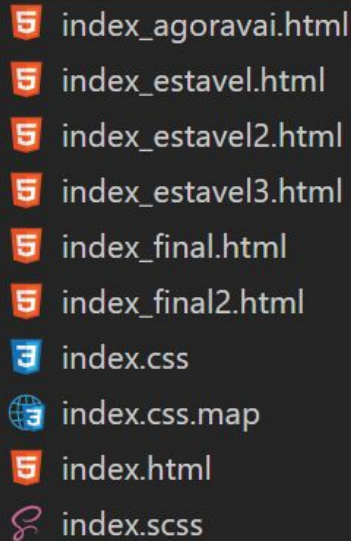
- index_agoravai.html
- index_estavel.html
- index_estavel2.html
- index_estavel3.html
- index_final.html
- index_final2.html
- index.css
- index.css.map
- index.html
- index.scss

Versionamento

Quando você olha para sua pasta, ela está assim:

Alguns dias depois, você volta a mexer nesse mesmo projeto e se depara com alguns problemas:

- Você não sabe mais qual é a versão mais recente do arquivo;



A screenshot of a file explorer interface with a dark background. On the left side, there is a vertical column of version numbers: five '5's, one '3', one '3' with a globe icon, one '5', and one '8'. To the right of these numbers are the following file names: index_agoravai.html, index_estavel.html, index_estavel2.html, index_estavel3.html, index_final.html, index_final2.html, index.css, index.css.map, index.html, and index.scss.

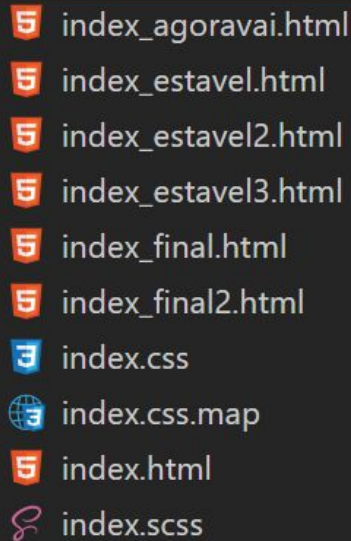
5	index_agoravai.html
5	index_estavel.html
5	index_estavel2.html
5	index_estavel3.html
5	index_final.html
5	index_final2.html
3	index.css
3	index.css.map
5	index.html
8	index.scss

Versionamento

Quando você olha para sua pasta, ela está assim:

Alguns dias depois, você volta a mexer nesse mesmo projeto e se depara com alguns problemas:

- Você não sabe mais qual é a versão mais recente do arquivo;
- Você não sabe qual a diferença entre cada versão do arquivo;



A screenshot of a file explorer interface with a dark background. On the left side, there is a vertical column of small, colored icons representing version numbers. To the right of these icons are the names of the files. The files listed are: index_agoravai.html, index_estavel.html, index_estavel2.html, index_estavel3.html, index_final.html, index_final2.html, index.css, index.css.map, index.html, and index.scss. The icons for the HTML files are orange with a white '5', the CSS file is blue with a white '3', the CSS map file is blue with a white globe icon, and the SCSS file is purple with a white 'S'.

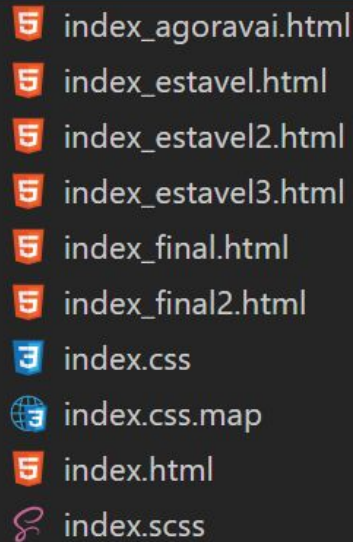
Version Icon	File Name
5	index_agoravai.html
5	index_estavel.html
5	index_estavel2.html
5	index_estavel3.html
5	index_final.html
5	index_final2.html
3	index.css
Globe	index.css.map
5	index.html
S	index.scss

Versionamento

Quando você olha para sua pasta, ela está assim:

Alguns dias depois, você volta a mexer nesse mesmo projeto e se depara com alguns problemas:

- Você não sabe mais qual é a versão mais recente do arquivo;
- Você não sabe qual a diferença entre cada versão do arquivo;
- Você não sabe em qual delas está aquela alteração específica que você fez.



```
index_agoravai.html
index_estavel.html
index_estavel2.html
index_estavel3.html
index_final.html
index_final2.html
index.css
index.css.map
index.html
index.scss
```

GIT

Criado por **Linus Benedict Torvalds** e lançado em **2005**, se tornou a principal ferramenta de controle de versão;

Através dela podemos desenvolver projetos na qual diversas pessoas podem contribuir **simultaneamente** editando e criando novos arquivos e permitindo que os mesmos possam existir sem o risco de suas alterações serem sobrescritas.



Git é um sistema de controle de versão de arquivos;

Com ele, podemos manter um histórico de todas as alterações que foram realizadas em nossos arquivos de código fonte;

Em equipe, podemos visualizar todas as alterações realizadas por todos os membros.

Cada nova funcionalidade (ou correção) que você adiciona na sua aplicação, **você salva nesse sistema de controle;**

Muitas vezes, uma nova funcionalidade/correção significa vários arquivos alterados de uma só vez;

Ao final do desenvolvimento de cada nova funcionalidade ou correção, salvamos o estado dos arquivos naquele momento.

Cada **“salvar”** desses, fica registrado em um **“pacotinho de alterações”**. É o que chamamos de **“commit”**;

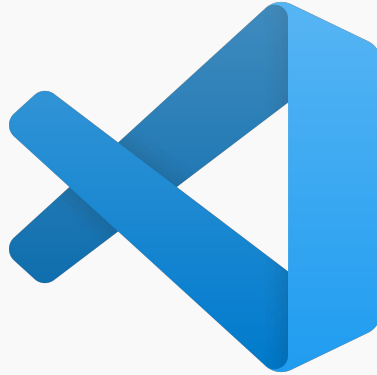
Quando você quiser voltar o estado da sua aplicação para um ponto específico do passado, ou apenas visualizar como era esse código em algum momento passado, você faz isso facilmente.

Seja para comparar uma versão do código que funcionava com outra que não está mais funcionando, resolvendo um bug;

Ou apenas descobrir o que foi alterado de uma versão para outra, por diversas outras razões.

Visual Studio Code | Instalação

Vamos instalar o Visual Studio Code:



<https://code.visualstudio.com/download>

Vamos realizar a instalação do Git. <https://git-scm.com/downloads>

- **Windows:** <https://git-scm.com/download/win>
- **MacOS:** <https://git-scm.com/download/mac>
- **Linux:** <https://git-scm.com/download/linux>



Execute o **comando** para verificar se a instalação foi executada com sucesso:

```
git --version
```

Vamos configurar o nome e e-mail que irão aparecer no commit.

- Para isso utilizamos os seguintes comandos:
 - `git config --global user.name "Seu nome"`
 - `git config --global user.email "email@example.com"`
- Verificar configurações:
 - `git config --list` ou `git config --l`

INTERVALO DE AULA

DEV!

Finalizamos o nosso primeiro período de hoje. Que tal descansar um pouco?!

Nos vemos em 20 minutos.

Início: 20:20

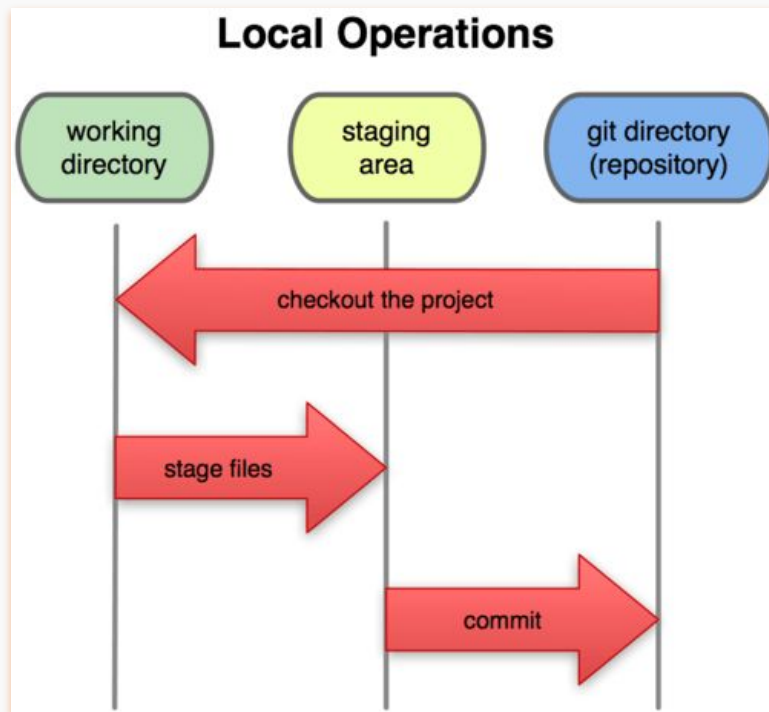
Retorno: 20:40



GIT Local

Fluxo de trabalho:

- Modificar arquivos no seu diretório de trabalho;
- Preparar os arquivos, adicionando eles à sua área de preparo;
- Fazer o commit, que leva os arquivos como estão na área de preparo e armazena de forma permanente no diretório do Git.



Antes de começar, precisamos aprender os **principais comandos Git**:

git init

- Inicia um novo repositório;

git status

- Verifica o status dos arquivos adicionados;

git add {arquivos}

- Adiciona os arquivos que foram modificados.
 - **Ex1:** Adiciona arquivos especificados
`git add index.html`
 - **Ex2:** Adiciona TODOS
`git add .`

git commit -m "Mensagem do commit"

- Comando utilizado para salvar suas informações no repositório.

git log

- Mostra um log de todos os commits feitos.

git revert {hash}

- Utilizado para reverter alterações de um commit. Para descobrir o hash pode ser usado o git log.



Mão na massa...



Vamos praticar!

- Em uma pasta com os arquivos, iniciaremos um repositório git com **git init**;
- Em seguida deve-se preparar os arquivos para adicioná-los de fato ao repositório. Fazemos isso com **git add .**
- Na sequência vamos executar o **git status** para ver o que foi adicionado;
- Por fim criamos um commit integrando o código com **git commit -m "mensagem do commit"**

GIT Remoto

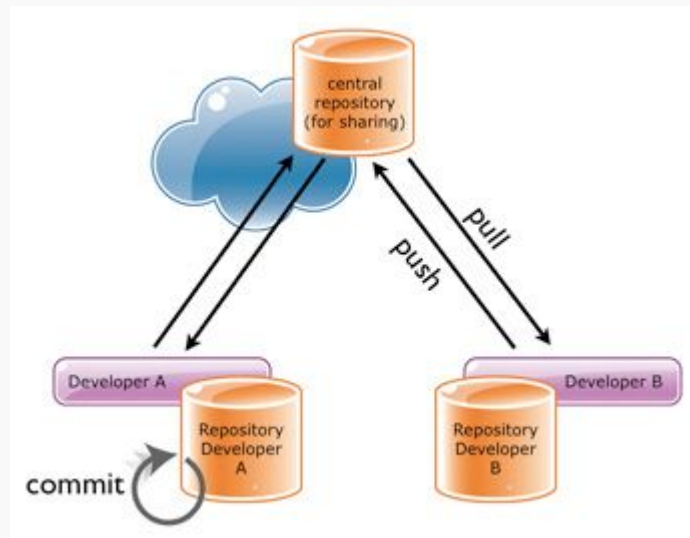
Além do gerenciamento local do repositório, existe o gerenciamento remoto, ou simplesmente repositório remoto;

Esse repositório é o original do qual os locais são uma cópia, e é para onde as alterações dos repositórios locais vão;

Os repositórios podem ser públicos ou privados. Em ambos os casos para alterarmos o repositório principal precisamos de uma permissão correspondente.

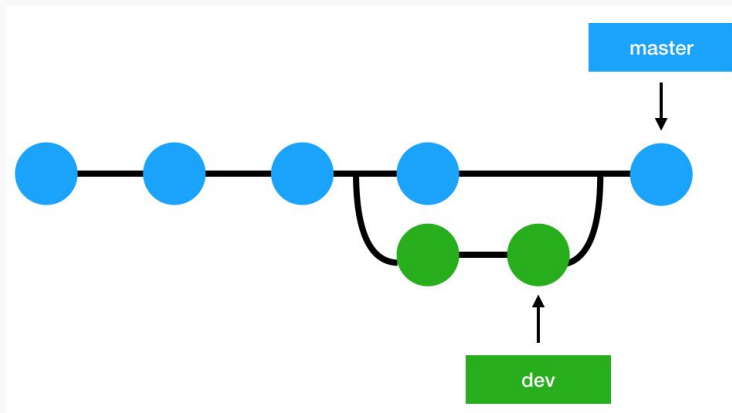
Fluxo de trabalho remoto:

- Obter um repositório remoto
`git clone {url-do-repositório}`
- Obter as alterações de um repositório
`git pull`
- Enviar as alterações de código (*commits*)
`git push`



Outro conceito básico no git é o de branches ou ramificações;

Quando criamos um novo repositório a branch “main” ou “master” é criada.



Na prática, branch começa a nos mostrar como trabalhar de forma colaborativa, onde eu não espero a entrega de outro desenvolvedor para executar e integrar as minhas alterações com o repositório remoto.

Para criar uma nova branch usamos **git checkout -b {nome-da-branch}**:

```
git checkout -b feat/Menu
```

```
git checkout -b fix/Rodape
```



Mão na massa...



Vamos praticar!

- Vamos criar uma branch no repositório clonado com o comando **git checkout -b nome-da-branch**
- Em seguida vamos alterar alguns arquivos e realizar o commit;
- Atualizar as alterações no git remoto com o comando **git push -set-upstream origin nome-da-branch**

Agora é hora de colocar em práticas os conhecimentos vistos na aula

- Atividade em grupo
- Duração: 15~20 minutos
- Versionamento local

Agora é hora de colocar em práticas os conhecimentos vistos na aula

- Atividade em grupo
- Duração: 10~15 minutos
- Versionamento local

Atividade

- Crie uma pasta específica para esta atividade
- Inicie um repositório local
- Crie dois arquivos (txt) em branco
 - *turmas.txt*
 - *alunos.txt*
- Adicione os dois arquivos para área de preparação
- Faça o seu primeiro commit

No arquivo *turmas.txt*

- Insira na primeira linha (Título) o texto "Turmas ativas"
- Adicione ao menos 5 nomes de turmas
- Adicione e faça o *commit*

No arquivo *alunos.txt*

- Insira na primeira linha (Título) o texto "Top alunos"
- Adicione ao menos 15 nomes dos top alunos
- Adicione e faça o *commit*

Para finalizar

- Adicione mais 5 nomes (turmas e alunos) em cada arquivo
- Adicione e faça o *commit* de cada arquivo separadamente

Material Complementar

- Documentation GIT - <https://git-scm.com/doc>
- Git - [woliveiras](#)
- Git - [medium](#)

AVALIAÇÃO DOCENTE

O que você está achando das minhas aulas neste conteúdo?

[Clique aqui](#) ou escaneie o QRCode ao lado para avaliar minha aula.

Sinta-se à vontade para fornecer uma avaliação sempre que achar necessário.





OBRIGADO!

<LAB365>