

# AUTOMAÇÃO FRONT END

---

Git

## Git

---

- **Git**

Sistema de controle de versões. Facilita o trabalho em equipe e o controle de mudanças entre arquivos e diretórios.

- **Github**

Plataforma online de hospedagem para repositórios Git. Existem outras como GitLab e Bitbucket.

## Git Setup

- Instalar o Git

<https://git-scm.com/>

- Configurar Nome

```
$ git config --global user.name "Seu Nome"
```

- Configurar Email

```
$ git config --global user.email "email@gmail.com"
```

## Git Comandos

---

- `$ git init`

Inicia um repositório

- `$ git add style.css`

Adiciona o arquivo style.css ao index do git. Com o `$ git add -A`, adicionamos todos os arquivos.

- `$ git status`

Mostra os arquivos que tiveram mudanças.

- `$ git commit -m 'Descrição'`

Irá fazer o commit do código adicionado com uma mensagem.

## Criar Repositório no Github

- Github

Criar conta: <https://github.com/>

- Novo Repositório

<https://github.com/new>

- Adicionar diretório remoto

```
$ git remote add origin  
https://github.com/seuusuario/seurepositorio.git
```

- Push do primeiro commit

```
$ git push -u origin master
```

- Se for a sua primeira vez

## Branching

---

- Branch

Uma das principais vantagens do git é a possibilidade de criarmos 'ramificações'. Assim podemos trabalhar em funcionalidades adicionais para um projeto, sem modificarmos o 'ramificação principal', o master.

- `$ git branch nomebranch`

Toda vez que formos adicionar uma nova funcionalidade, devemos iniciar criando um novo branch ao invés de fazermos alterações direto no master. O que for modificado no branch não afetará o master.

- `$ git checkout nomebranch`

Irá mudar de branch. Podemos usar o atalho

`$git checkout -b novobranch`, assim ele cria e muda de branch ao mesmo tempo.

- `$ git branch`



## Workflow

- Sempre crie um branch

Toda funcionalidade nova, crie um branch para desenvolver a mesma.

```
$ git checkout -b feature1
```

- Após o desenvolvimento e commit, vá até o master e veja se existem mudanças

```
$ git checkout master e $ git pull
```

- Volte para o branch e dê um merge com o master

```
$ git checkout feature1 e depois $ git merge master
```

- Conflitos

Se existirem conflitos você será avisado e deverá lidar com os mesmos



Após lidar com os conflitos faça o push do branch: `$ git push` e  
`$ git push --set-upstream origin feature1`.

## Lidando com Pull Request

- **No Github**

Agora você possui um novo branch no github e pode fazer o pull request (juntar ao master).

- **Compare e Pull Request**

Pode adicionar comentários. Create Pull Request.

- **Merge Pull Request**

Geralmente é o líder do projeto / responsável por fazer o review do seu código. Pode deletar o branch após o merge com o master.

## Mais Git

---

- **.gitignore**

Lista de arquivos que não devem ser manipulados pelo git. `node_modules` é um bom exemplo.

- **Commit sem texto**

Ao usar o `$ git commit` você entra no modo completo de comentário, com um editor de texto direto na linha de comando. Utilize `esc` + `:wq` para sair do mesmo.

- **Bitbucket**

Permite repositórios privados e gratuitos. <https://bitbucket.org/product>

## Github Pages

- Criar repositório

O nome deve ser `seuusuario.github.io`

- HTML Simples

O site só funcionará em html/css/js simples, sem linguagem de servidor

- Qualquer projeto

Qualquer projeto poderá ter uma página para o mesmo. Vá em Settings > GitHub Pages > selecione master branch e salve. E acesso `seuusuario.github.io/repositorio/`