Que dia é hoje?

Módulo 2 - Aula 6 - HARD

RESILIA | Senac

Revisão

Datas

Datas

Datas são parte vital em diversos sistemas e aplicações.
Com elas, podemos além de garantir o bom funcionamento de uma aplicação, analisar o desempenho e estratégias do produto.
Cenários em que datas são importantes:

- Ecommerce
- Notícias
- Redes sociais

Datas: epoch time

Em computação, muitas vezes o referencial de data utilizado é um número inteiro. Isso é feito para que haja facilidade de lidar com datas, padronização e sincronização dos sistemas.
Esse número representa o total de milissegundos decorridos a partir do dia 01/01/1970.

JavaScript: datas

O JavaScript fornece a classe Date para que seja possível trabalhar com datas de forma mais fácil. Uma instância pode ser criada a partir da utilização de diferentes construtores, que recebem diferentes parâmetros.

```
new Date();
new Date(valor); //epoch time (inteiro)
new Date(dataString); //string em formato de data
new Date(ano, mês, dia, hora, minuto, segundo,
milissegundo);
```

Date: atributos e métodos

Um objeto date possui diversos métodos que facilitam a utilização de datas no JavaScript. Além disso, a classe fornece métodos utilitários para lidar com datas.

```
//Métodos de um objeto
const data = new Date(2001, 12, 11);
data.getDay(); //Dia da semana de 0 a 6
data.getFullYear(); //Ano completo (4 dígitos)
data.toISOString(); //String com data completa no formato ISO
//Métodos da classe
Date.now(); //epoch time agora!
Date.UTC(); //epoch time de ano, mes, dia,... como argumento
```

Day.js

Day.js é uma biblioteca, assim como o jQuery, que facilita a manipulação de datas.
Importantando no html (é importante que essa importação esteja antes do arquivo em que ela será utilizada):

```
<script src="https://cdn.jsdelivr.net/npm/dayjs@1/dayjs.min.js"></script>
```

Day.js

Utilizando a biblioteca no arquivo js:

```
const data = dayjs()
console.log(data.format()) // 2022-09-05T08:00:00+08:00
console.log(data.format("YYYY MM DD")) // 2022 09 05
console.log(data.format("DD MM YYYY")) // 05 09 2022
```

Recap

Comandos

Comandos

Comandos básicos do git vistos até agora

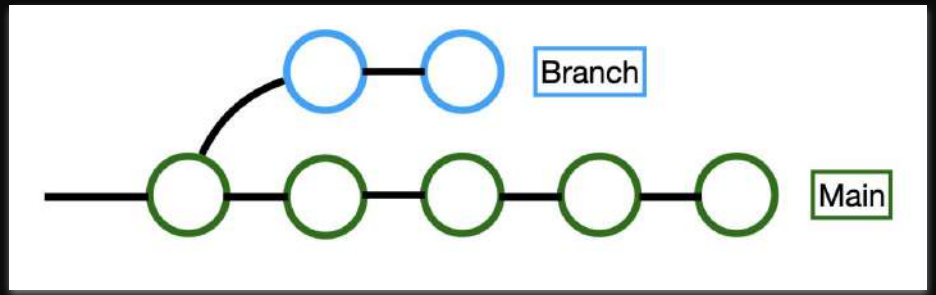
- git init
- git add
- git commit
- git push

Google Drive?

git + Github

git: branches

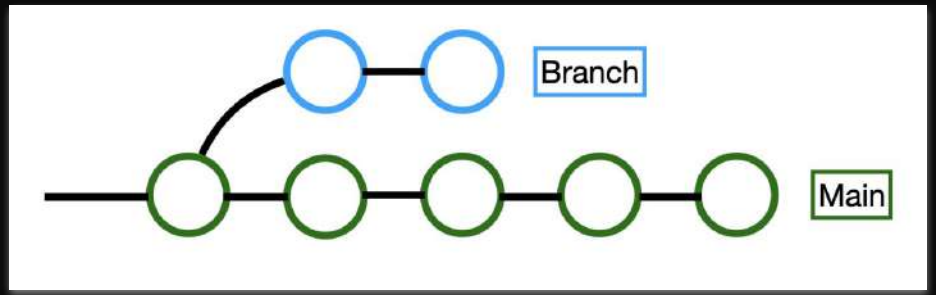
Por padrão, o git inicia um projeto utilizando uma branch (galho) principal (master/main).
Branches diferentes são como linhas temporais diferentes, com históricos diferentes. Geralmente uma nova branch é criada por funcionalidade e após concluído o trabalho uma branch é incorporada a outra.



git: branches - sintaxe

Uma nova branch pode ser criada com o git branch. No entanto, ao utilizar git checkout -b <nome da branch>, além de criar uma nova branch (caso não exista) “pulamos” para ela.
Ex:

$ git checkout -b req_APIs



git: branches - sintaxe

Podemos também listar as branches existentes em um repositório com git branch apenas e, passando como parâmetro -r, verificamos as branches remotas.
O comando git checkout <nome da branch> é utilizado para pular para uma branch específica.

Ex:

```
$ git branch

$ git checkout main
```

git: merge

O processo de merge realiza a junção de duas branches diferentes. Para que seja realizado o merge da branch atual com outra utilizamos o comando git merge <nome da branch>.

Ex:

```
$ git merge cabecalho
```

Cuidado!

git: merge - conflitos

Conflitos podem ocorrer quando tentamos realizar o merge de branches que tenham avançado na história e desenvolvido códigos diferentes nos mesmos arquivos.
Entretanto, o processo de resolução consiste em definir qual trecho deve ser mantido.
Após os conflitos serem resolvidos, é preciso dar continuidade com o merge usando o comando git merge --continue.

Fluxo de trabalho Github

branches remotas

git + github: branches remotas

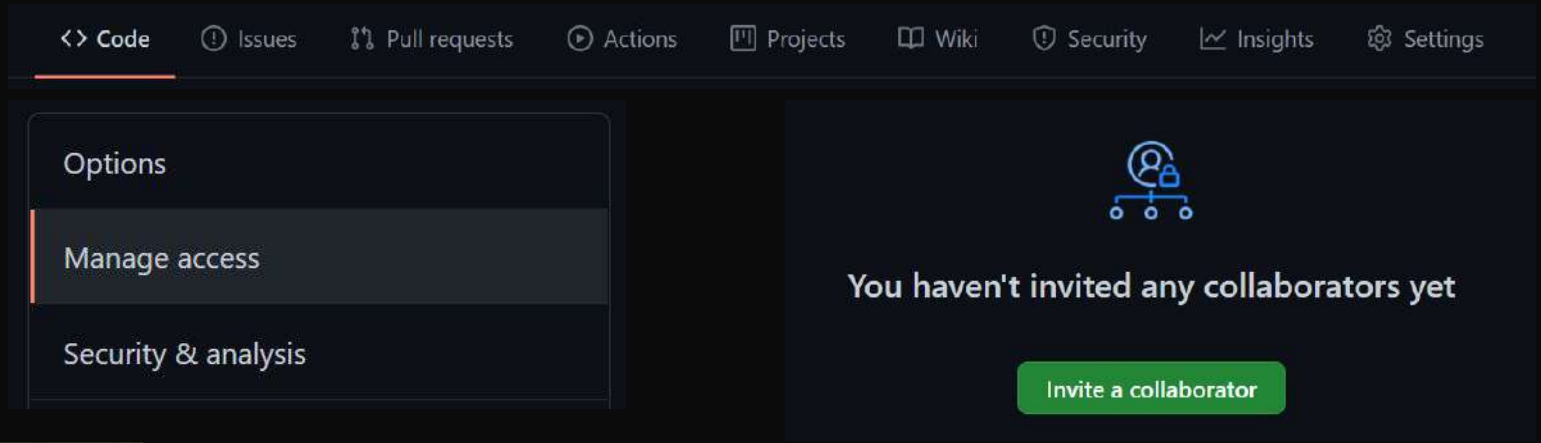
Branches remotas são branches mantidas em um repositório/servidor de forma remota.
Para configurar uma branch remota com o mesmo nome da atual, podemos utilizar o próprio push com -u e o nome desejado.

Ex (estando na branch styles):

```
$ git push -u origin styles
```

github: colaboradores

É possível adicionar colaboradores a um repositório no github.
Para isso utilizamos as configurações do repositório e adicionamos novos colaboradores.



github: pull request

Com uma branch remota devidamente configurada, podemos abrir uma pull request (PR), para que o merge seja feito de forma remota entre duas branches. A pull request pode ser processada e aceita por algum colaborador do repositório.

Fluxo de trabalho: github

O fluxo de trabalho abaixo contém as regras e sugestões para que o menor número de problemas ocorra com um grupo utilizando um repositório git/github para gerenciamento de um projeto:

1. O repositório deve ser criado e todos os integrantes devem ter acesso a ele
2. A branch main/master não deve ser utilizada de forma direta, apenas para consolidar pull requests (uma pessoa deve gerenciar as PRs)
3. Cada pessoa desenvolve uma funcionalidade em um branch diferente e, quando finalizado, realiza o push e cria uma pull request
4. A pull request é incorporada na branch principal
5. A branch principal (local) é atualizada antes que uma nova seja criada (git pull)
6. O ciclo se repete indefinidamente!

ATIVIDADE
quebra-galho

Atividade: quebra-galho

Vamos testar a dinâmica de trabalho utilizando mais de um colaborador em um repositório, com branches e pull requests. Em duplas ou trios:
Objetivo:

- Estabeleça uma pessoa responsável pela criação do repositório e adição de ao menos duas pessoas do grupo como colaboradores;
- Cada integrante adicionado deve:
  - Clonar o repositório (git clone <url repo>);
  - Criar uma nova branch local (git checkout -b <nome>);
  - Criar um novo arquivo e commita-lo
    - Cada estudante deve criar um arquivo diferente;
    - Nesse arquivo cada estudante deve criar um código utilizando manipulação algum método de data visto em aula;
  - Realizar um push da branch atual para a branch remota (git push -u origin <nome>);
  - Utilizar a interface do github para criar uma nova pull request para a branch main/master.
- Usar a interface de code review para analisar e comentar o código da outra pessoa;
- A pessoa que criou o repositório deverá gerenciar e aceitar as pull requests.

ELI5: Explain like I’m five

ELI5: Explain Like I'm five

Relembrando a atividade "Me explique como se eu tivesse 5 anos".

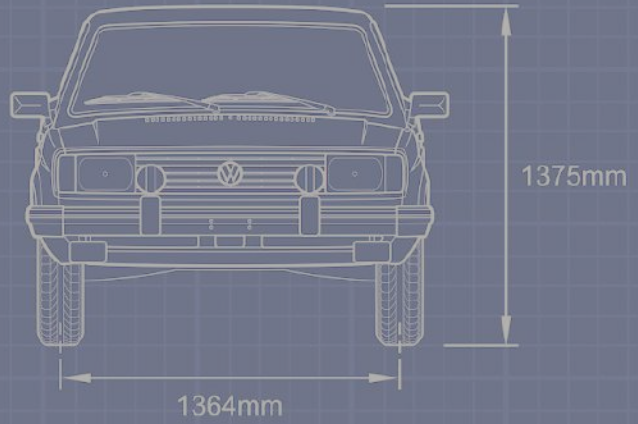
- Explicar o significado de cada palavra/tema como com palavras simples, como se estivesse explicando para uma criança
- Dicas:
  - Use palavras simples;
  - Use e abuse de exemplos e associações, tente relacionar o assunto com algo do dia a dia;
  - Se estiver com dificuldade, relaxa!!! Não é fácil, mesmo!

Vamos lá?

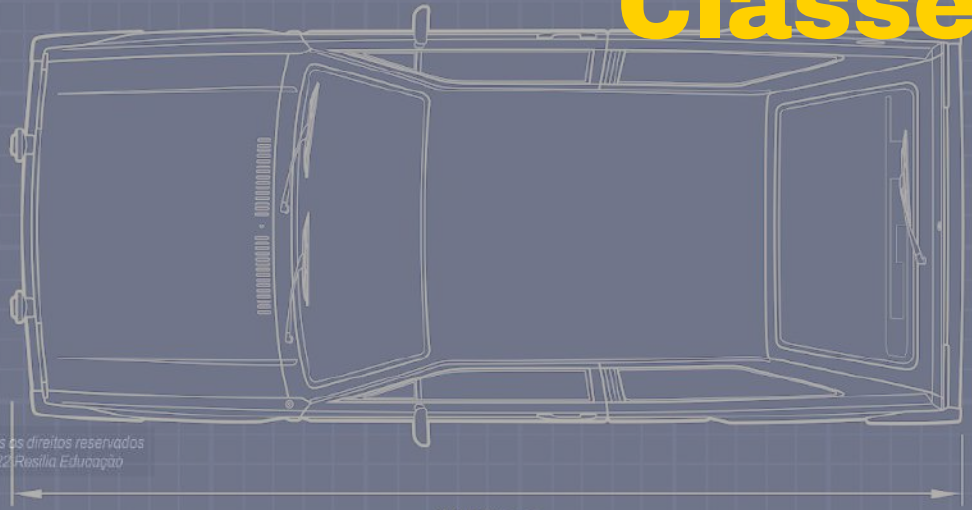
AJAX

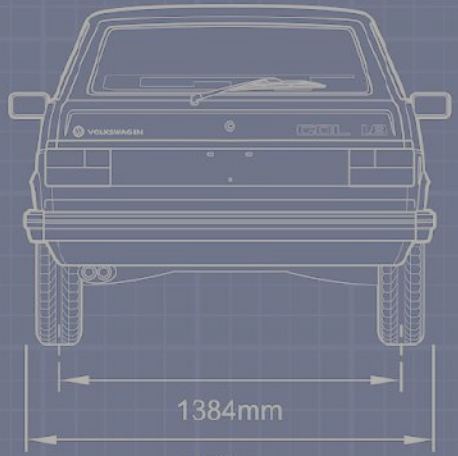
Volkswagen

Gol GT 1984



Classes





Objetos

Atributos

Constructor

Erros

MOMENTO
APRENDIZAGEM EM PARES