## Imagine que esteja iniciando um novo projeto de software:

- 1. Qual seria sua estratégia para identificar as versões? Justifique.
- 2. Como nomearia a primeira versão para o público?
- 3. Após liberado o projeto, uma nova funcionalidade foi requisitada e implementada. Como nomearia esta nova versão?
- 4. Considerando a sequência anterior com o esquema de versionamento SemVer. Como ficaria o histórico de versões?
- 5. A partir da versão indicada antes, uma série de 3 correções, seguida por duas funcionalidades compatíveis com a versão atual e mais outras 2 correções foram publicadas em série. Qual seria a versão mais recente?
- 6. Uma versão nova exigiu mudanças críticas na API, foi lançada e na sequências houveram três novas versões: uma com adição de funcionalidades, seguidas de 2 com correções de bugs. Como fica o histórico?
- 7. Pesquise exemplos representativos de versões reais de software considerando cada um dos termos apontados.

## 1. Estratégia para identificar versões:

- Usaria o Versionamento Semântico (SemVer). Ele é claro, amplamente adotado e facilita a compreensão das mudanças entre as versões.
- A estrutura é MAIOR. MENOR. PATCH.
- MAIOR: Mudanças incompatíveis na API.
- MENOR: Adição de funcionalidades compatíveis com a versão anterior.
- PATCH: Correções de bugs.
- Justificativa: Permite que os usuários entendam o impacto da atualização antes de aplicá-la, evitando surpresas e facilitando a gestão de dependências.

## 2. Nome da primeira versão para o público:

- 1.0.0.
- Indica que a API está estável e pronta para uso em produção.

- 3. Nova versão após adicionar funcionalidade:
- 1.1.0.
- O número MENOR é incrementado, indicando a adição de uma funcionalidade compatível com a versão anterior.

#### 4. Histórico de versões com SemVer:

- 1.0.0 (Versão inicial)
- 1.1.0 (Nova funcionalidade)

## 5. Versão mais recente após correções e funcionalidades:

- Partindo de 1.1.0:
- 3 correções: 1.1.1, 1.1.2, 1.1.3
- 2 funcionalidades: 1.2.0, 1.3.0
- 2 correções: 1.3.1, 1.3.2
- A versão mais recente seria 1.3.2.

## 6. Histórico após mudanças críticas na API:

- 2.0.0 (Mudança crítica na API)
- 2.1.0 (Adição de funcionalidade)
- 2.1.1 (Correção de bug)
- 2.1.2 (Correção de bug)

### 7. Exemplos de versões reais de software:

- Linux Kernel: 5.15.0 (Grandes projetos de código aberto)
- Node.js: 16.13.0 (Plataformas de desenvolvimento)
- Angular: 13.0.0 (Frameworks web)

# ATIVIDADE:: REPOSITÓRIOS E PRIMEIROS PASSOS

- Pesquise alguns projetos no GitHub e liste ao menos três de seu interesse:
- Escolha uma pasta e clone um desses projetos, observando o quer foi gerado;
- 3. crie uma pasta "hellogit" e inicialize um novo repositório git nele; verifique o que foi criado na pasta;
- Crie um arquivo em branco hello.java re registre no repositório;
- Complete com o código para gerar "Hello World" e repita o processo indicando as alterações feitas.

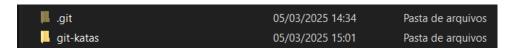
63

#### 1. Projetos do GitHub:

- Flutter (flutter/flutter): Um SDK para criar aplicativos móveis e web desenvolvido pelo Google3.
- TensorFlow: Uma biblioteca de aprendizado de máquina desenvolvida pelo Google, muito usada em projetos de inteligência artificial.
- freeCodeCamp: Um projeto que oferece desafios de codificação interativos para ajudar desenvolvedores a melhorar suas habilidades.

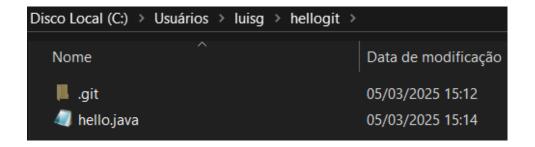
## 2. Clonando projeto:

```
C:\Users\luisg\meuprojeto>git clone https://github.com/eficode-academy/git-katas Cloning into 'git-katas'...
remote: Enumerating objects: 2537, done.
remote: Counting objects: 100% (1310/1310), done.
remote: Compressing objects: 100% (540/540), done.
remote: Total 2537 (delta 855), reused 770 (delta 770), pack-reused 1227 (from 1)
Receiving objects: 100% (2537/2537), 2.82 MiB | 7.56 MiB/s, done.
Resolving deltas: 100% (1110/1110), done.
```



📮 .git	05/03/2025 15:01	Pasta de arquivos
github	05/03/2025 15:01	Pasta de arquivos
3-way-merge	05/03/2025 15:01	Pasta de arquivos
advanced-rebase-interactive	05/03/2025 15:01	Pasta de arquivos
alias	05/03/2025 15:01	Pasta de arquivos
amend	05/03/2025 15:01	Pasta de arquivos
bad-commit	05/03/2025 15:01	Pasta de arquivos
basic-branching	05/03/2025 15:01	Pasta de arquivos
basic-cherry-pick	05/03/2025 15:01	Pasta de arquivos
basic-cleaning	05/03/2025 15:01	Pasta de arquivos
basic-commits	05/03/2025 15:01	Pasta de arquivos
basic-revert	05/03/2025 15:01	Pasta de arquivos
basic-staging	05/03/2025 15:01	Pasta de arquivos
basic-stashing	05/03/2025 15:01	Pasta de arquivos
bisect	05/03/2025 15:01	Pasta de arquivos
change-author	05/03/2025 15:01	Pasta de arquivos
commit-on-wrong-branch	05/03/2025 15:01	Pasta de arquivos
commit-on-wrong-branch-2	05/03/2025 15:01	Pasta de arquivos
configure-git	05/03/2025 15:01	Pasta de arquivos
detached-head	05/03/2025 15:01	Pasta de arquivos
diff-advance	05/03/2025 15:01	Pasta de arquivos
docs	05/03/2025 15:01	Pasta de arquivos
ff-merge	05/03/2025 15:01	Pasta de arquivos
git-attributes	05/03/2025 15:01	Pasta de arquivos
📙 git-tag	05/03/2025 15:01	Pasta de arquivos
📙 ignore	05/03/2025 15:01	Pasta de arquivos
images	05/03/2025 15:01	Pasta de arquivos
investigation	05/03/2025 15:01	Pasta de arquivos

# 4. Arquivo "hellojava":



## 5. Cod para hello world:

```
public class Hello {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

# ATIVIDADE:: IGNORANDO ARQUIVOS

 Descreva o resultado esperado para um repositório com um arquivo .ignore contendo:

```
!lib.a
/TODO
build/
doc/*.txt
doc/**/*.pdf
```

- Inclua um arquivo hello.tmp no repositório criado na atividade anterior.
- 3. Observe o "status" deste arquivo, sem incluí-lo no repositório.
- 4. Altere o repositório para que passe a ignorar este arquivo.

```
C:\Users\luisg\hellogit>git status
On branch main
Changes not staged for commit:
   (use "git add <file>..." to update what will be committed)
   (use "git restore <file>..." to discard changes in working directory)
        modified: hello.java

Untracked files:
   (use "git add <file>..." to include in what will be committed)
        .gitignore
        hello.tmp

no changes added to commit (use "git add" and/or "git commit -a")
```

```
.gitignore - Bloco de Notas

Arquivo Editar Formatar Exibir Ajuda

!lib.a
/TODO
build/
doc/*.txt
doc/**/*.pdf
```

# ATIVIDADE:: ESTADOS E DIFERENÇAS

- 1. Altere o arquivo hello.java para que gere dez saídas "Hello Git!";
- Observe o "status" e diferenças, a seguir registre a versão atualizada no repositório;
- Observe a diferença entre os arquivos. Repita o processo fazendo novas modificações;
- Crie um arquivo "hello.txt", inclua um texto com algumas orientações sobre o Git e adicione ao repositório;
- 5. Renomeie o arquivo para "hello.md";
- 6. Remova o arquivo do repositório.

## 1. Alterando o hello.java:

```
hello.java - Bloco de Notas

Arquivo Editar Formatar Exibir Ajuda

public class Hello {
    public static void main(String[] args) {
        for (int i = 0; i < 10; i++) {
            System.out.println("Hello Git!");
        }
    }
}</pre>
```

2. Status:

```
C:\Users\luisg\hellogit>git status
On branch main
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
Untracked files:
 (use "git add <file>..." to include in what will be committed)
no changes added to commit (use "git add" and/or "git commit -a")
C:\Users\luisg\hellogit>git status
On branch main
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
Untracked files:
 (use "git add <file>..." to include in what will be committed)
no changes added to commit (use "git add" and/or "git commit -a")
```

## 3. Diferença entre os arquivos:

```
C:\Users\luisg\hellogit>git add hello.java
C:\Users\luisg\hellogit>git commit -m "Modificando hello java para aperecer hellogit 10 vezes"
[main 412c33b] Modificando hello java para aperecer hellogit 10 vezes
1 file changed, 7 insertions(+)
C:\Users\luisg\hellogit>git add hello.java
```

### 4. Hello.txt:

:∖Users\luisg\hellogit>echo "Algumas orientações sobre Git: versionamento, controle de código, colaboração." > hello.txt

### 5. Renomeando Hello.txt:

```
C:\Users\luisg\hellogit>git add hello.txt
C:\Users\luisg\hellogit>git mv hello.txt hello.md
```

## 6. Removendo o arquivo:

```
C:\Users\luisg\hellogit>git commit -m " renomeando arquivo"
[main 624b16d] renomeando arquivo
  1 file changed, 1 insertion(+)
  create mode 100644 hello.md

C:\Users\luisg\hellogit>git rm hello.md
rm 'hello.md'
```