

NExT

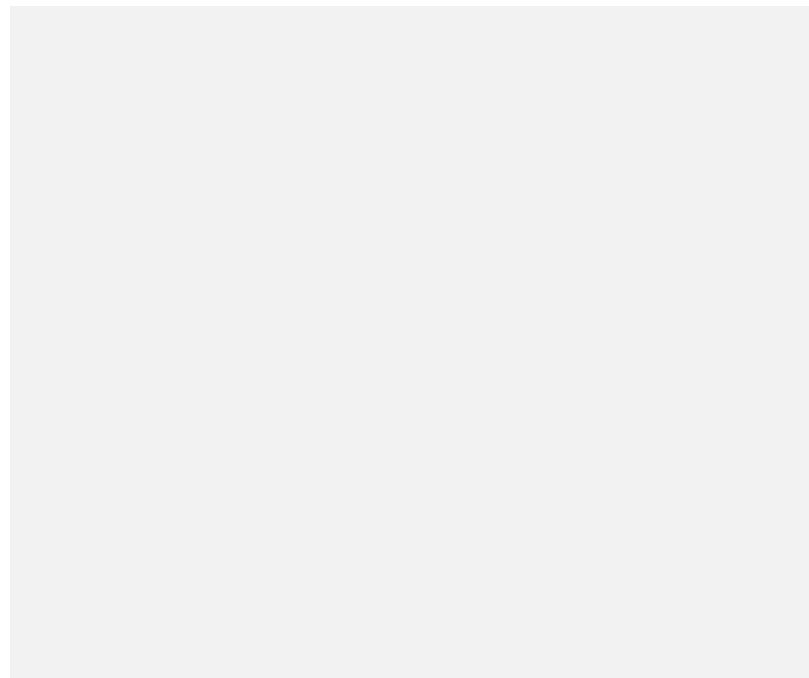
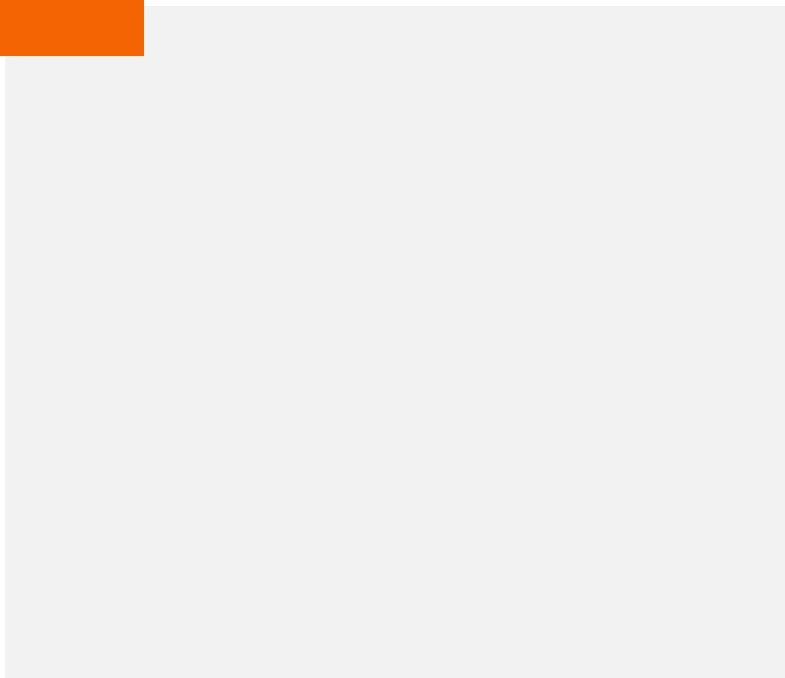
Nova Experiência de Trabalho

Ellen Coelho
ecxc@cesar.org.br

Ronnie Santos
ress@cesar.org.br



MONITORES



Agenda

1ª Aula (07/11)

Introdução a
Gerência de
Configuração

Introdução ao
Git

Comandos Git

2ª Aula (08/11)

Explorando o
GitHub

Criando
Read.me

Criando
Repositório e
adicionando
projetos ao
Github

3ª Aula (09/11)

Aplicação do
Fluxo de
Trabalho no
GitHub

Mesclando
Branches

Review Process


Conflitos

4ª Aula (10/11)

Gerência de
configuração e
Qualidade de
software

Integração
contínua

Testes
automatizados
e Github

Todo nosso material de aula, exercícios e chamada vão ser disponibilizados diariamente via Google Classroom 

Classroom

Código da sala:

qjr4fg2



Qualidade e Integração Contínua

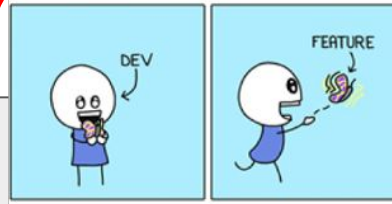
Processo de Desenvolvimento de Software

Requisitos



Análise e Projeto

Programação



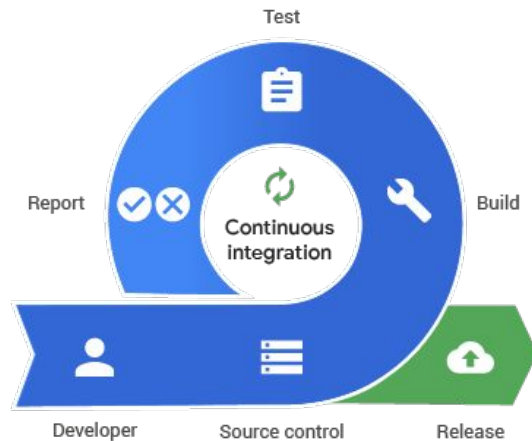
Testes



Release

Integração Contínua

As metodologias ágeis propõem que o processo de desenvolvimento de software seja realizado com entregas mais frequentes. Como consequência, o trabalho necessário para reunir, integrar e testar todo o código desenvolvido pela equipe em um repositório central também se tornou mais frequente.



Build de Teste

Branches Individuais

O teste de software é a ação de examinar os artefatos e o comportamento de um software em desenvolvimento através de sua verificação e validação.

Teste de Software



Níveis de Teste

- **Teste de unidade:** teste de uma função de um desenvolvedor.
- **Teste de componentes:** teste para uma classe ou pacote de vários desenvolvedores
- **Teste de integração:** teste para mais de 2 componentes juntos.
- **Teste de sistema:** teste do software em sua configuração final
- **Teste de aceitação:** teste do software no ambiente do usuário.



E como é que
testa?



Manual



Automatizado

Teste Automático

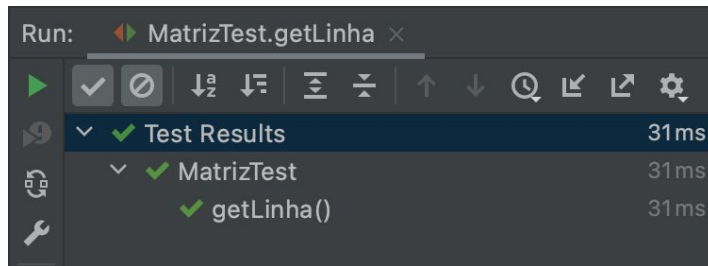
```
/**  
 * @return a quantidade de linhas matriz.  
 */  
public int getLinha() {  
    return linha;  
}
```



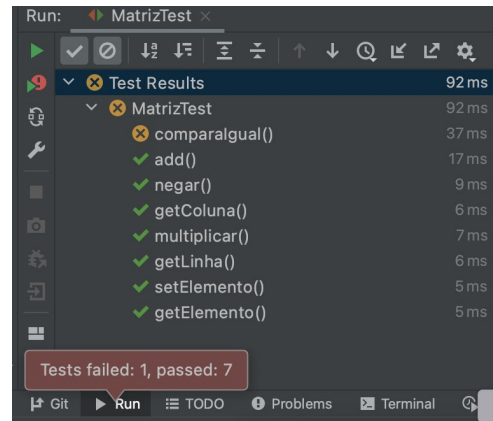
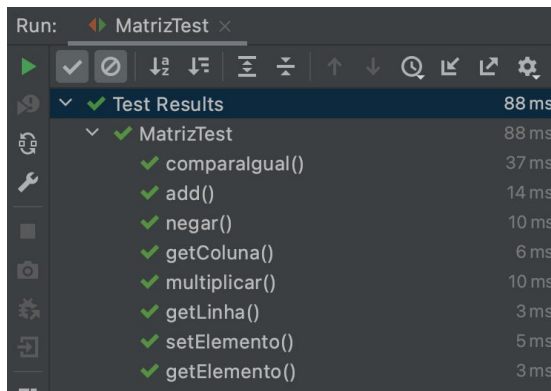
```
private final static String matrizSimples = "2 2 1 2 3 4"; // 1 2  
                                           // 3 4  
  
@Test  
void getLinha() {  
    Matriz m = new Matriz(new Scanner(matrizSimples));  
    assertEquals( expected: 2, m.getLinha(), message: "numero de linhas incorreto");  
}
```



Teste Automático e Integração Contínua



1 teste



N testes

Processo de Teste Resumido

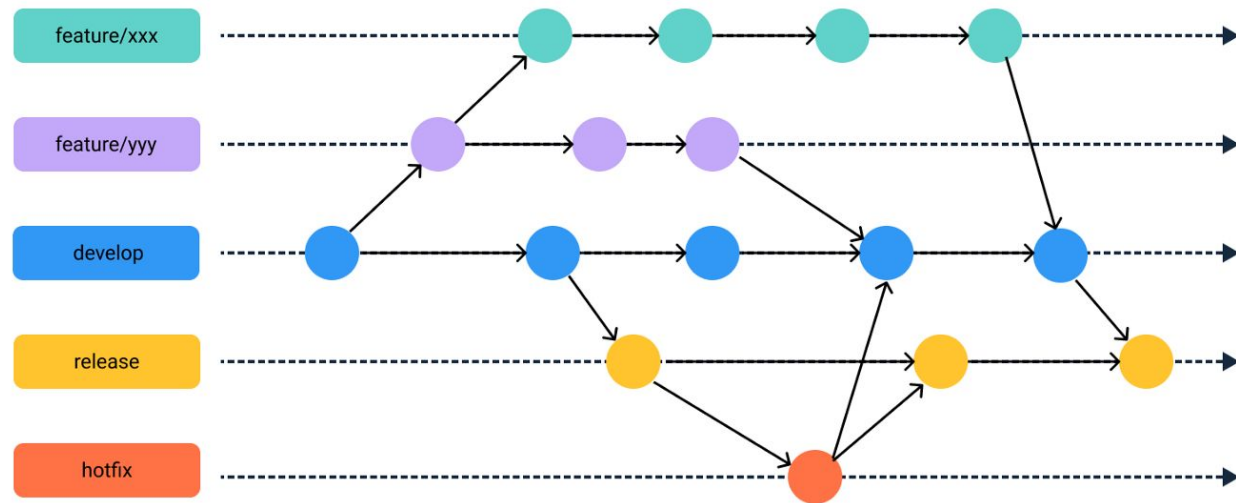


Testar



Debuggar

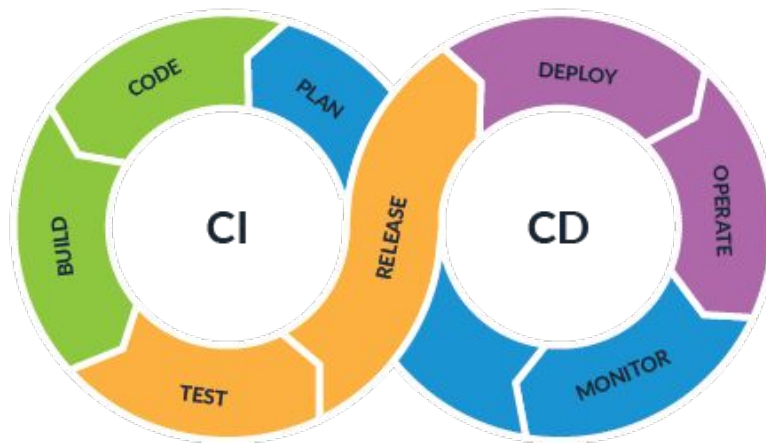
E o git com isso?



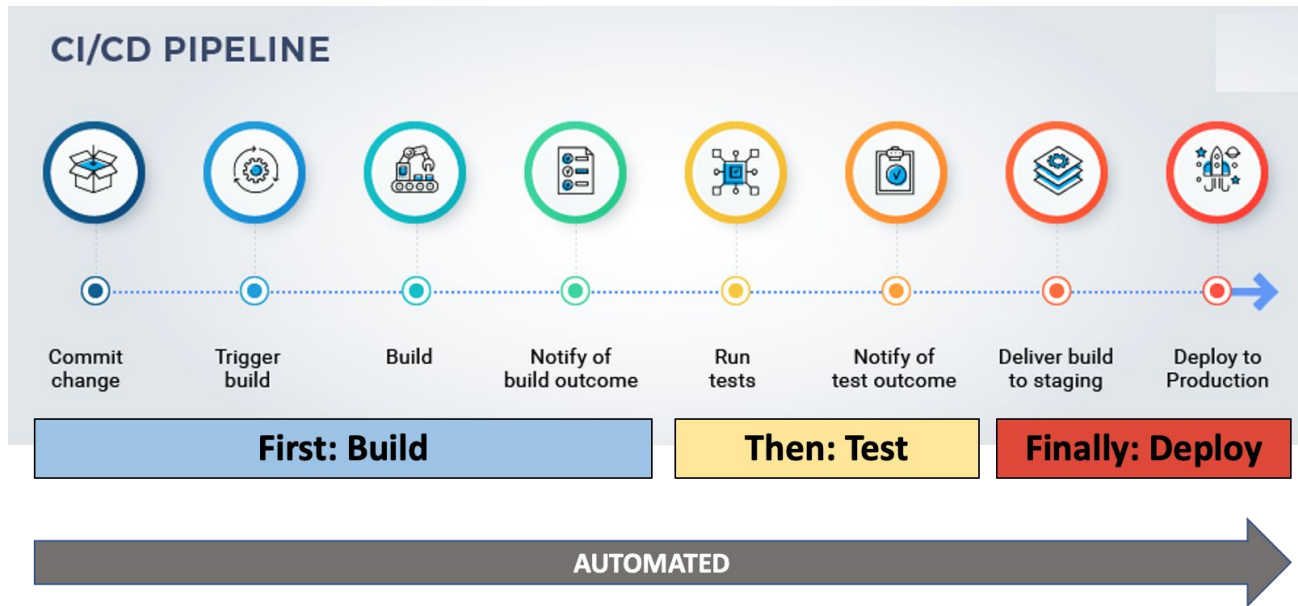
Entrega Contínua

A entrega contínua visa garantir que o código esteja apto para entrar em ambiente de produção. Para isso, é necessário que:

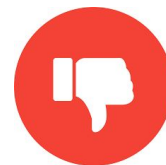
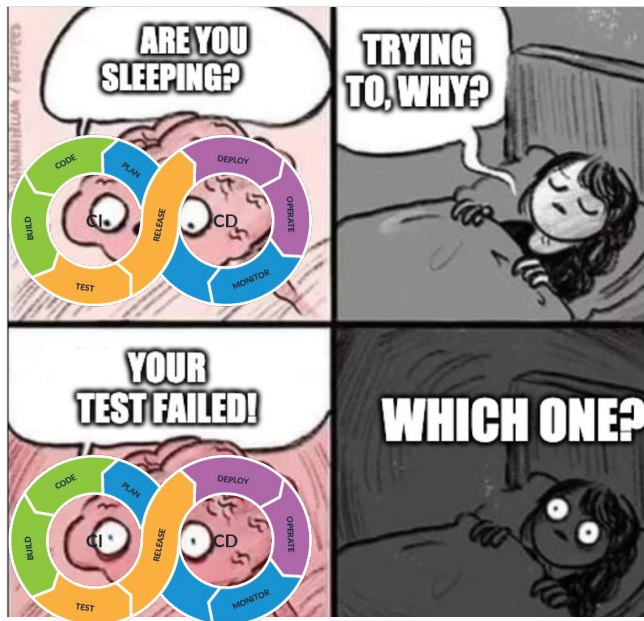
- 1) o deploy do código nos ambientes de homologação e desenvolvimento possa ser realizado de forma simplificada;
- 2) testes complementares aos unitários e inspeções finais devem ser executadas o mais rápido possível.



Integração Contínua



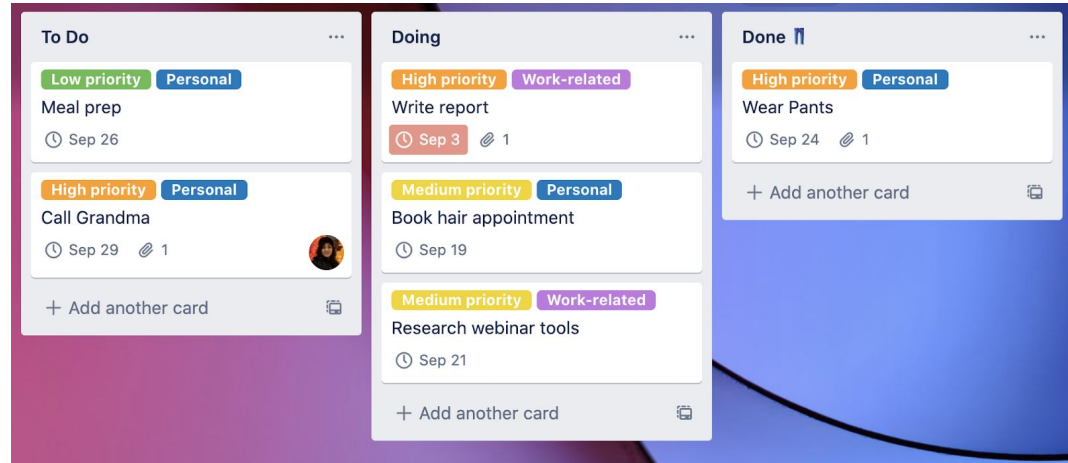
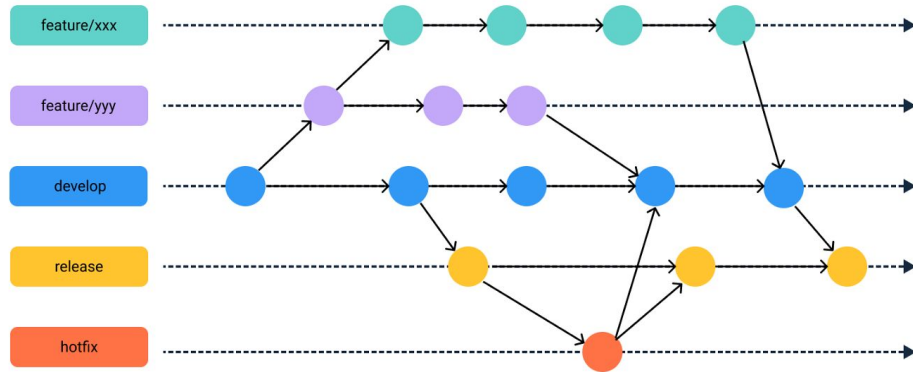
E como lidar
com tudo
isso?





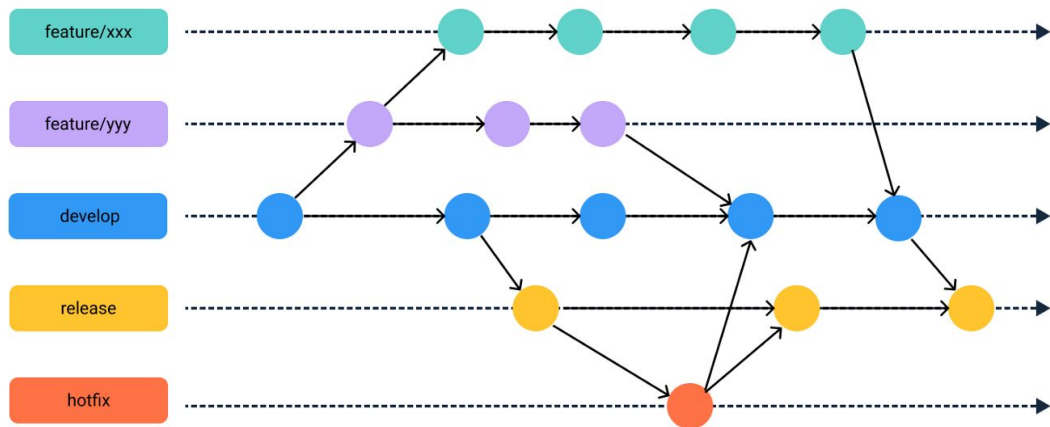
QUALIDADE E GERÊNCIA DE CONFIGURAÇÃO

Organizando Branches





Planejando Branches



<https://www.saucedemo.com/>

1 - Use o Trello para definir um esquema de branches

2 - Identifique e classifique falhas no site e crie cards de acordo.

3 - Proponha 3 features para o site e crie cards de acordo.



.gitignore

O que é?

Pode haver arquivos que você não deseja incluir ao confirmar um conjunto de modificações, como configurações privadas e arquivos de sistema ocultos.

O **Git não tem um comando ignore**, mas você pode usar um **arquivo .gitignore** para esta tarefa.

Um arquivo .gitignore é um arquivo de texto que especifica quais arquivos e pastas o Git deve ignorar em sua árvore de trabalho.

Normalmente existe no diretório raiz de um projeto.



Quais arquivos podem e devem ser ignorados?

Geralmente são arquivos gerados por máquina ou artefatos construídos.

Porém qualquer arquivo pode ser ignorado.

Os exemplos mais comuns incluem:

- Arquivos de sistema ocultos, como .DS_Store e Thumbs.db;
- Arquivos gerados durante o tempo de execução, como .log e .temp;
- Caches de dependência, como o conteúdo de /node_modules e /packages;
- Arquivos de configuração do IDE pessoal, como .idea/workspace.xml;
- Crie diretórios de saída, como /bin, /out e /target;

/packages



.idea/workspace.xml



.temp



/bin



Como criar?

Para criar um arquivo .gitignore, crie um arquivo de texto simples usando qualquer editor de texto como o Bloco de Notas e nomeie-o como .gitignore. Adicione os arquivos de destino conforme necessário.

```
.gitignore
1  HELP.md
2  target/
3  */.mvn/
4  !**/src/main/**/target/
5  !**/src/test/**/target/
6
7  ### STS ###
8  .apt_generated
9  .classpath
10 .factorypath
11 .project
12 .settings
13 .springBeans
14 .sts4-cache
15
16 ### IntelliJ IDEA ###
17 .idea
18 *.iws
19 *.iml
20 *.ipr
21
22 ### NetBeans ###
23 /nbproject/private/
24 /nbbuild/
25 /dist/
26 /nbdist/
27 /.nb-gradle/
28 build/
29 !**/src/main/**/build/
30 !**/src/test/**/build/
31
32 # VS Code ###
33 .vscode/
```

```
# compiled output
/dist
/tmp
/out-tsc
# Only exists if Bazel was run
/bazel-out

# dependencies
/node_modules
/Frontend/npi-front/node_modules

# profiling files
chrome-profiler-events*.json
speed-measure-plugin*.json

# IDEs and editors
/.idea
.c9/
*.launch
.settings/
*.sublime-workspace

# IDE - VSCode
.vscode/*
!.vscode/settings.json
!.vscode/tasks.json
!.vscode/launch.json
!.vscode/extensions.json
.history/*

# misc
/.sass-cache
```

Como ignorar?



A diagram showing a dark rectangular box representing a .gitignore file. Inside the box, the following text is written in a monospaced font: `nomedoarquivo.txt`, `diretório/`, `*.html`, and `*~`. An arrow points from the text `.gitignore` located above and to the right of the box to the top-right corner of the box.

```
nomedoarquivo.txt
diretório/
*.html
*~
```

1. Nomes de arquivo literais – ignore um nome de arquivo específico, como `nomedoarquivo.txt`.
2. Diretórios – ignore um diretório inteiro adicionando um símbolo de barra (/) no final do nome do diretório, por exemplo, `diretório/`.
3. Curinga – ignore qualquer arquivo que termine com uma extensão específica adicionando um asterisco (*) antes do nome da extensão, por exemplo, `*.html`. Além disso, `*~` solicitará ao Git que ignore qualquer arquivo que termine com `~`, como `index.html~`.
4. Você também pode especificar determinados arquivos a serem excluídos da lista `.gitignore` usando o `!` prefixo.

Criar exceções

Por exemplo, você não quer que o Git ignore `example.html`, mesmo que todos os outros arquivos com extensão `.html` sejam ignorados. Nesse caso, adicione isso à sua lista `.gitignore`:



```
nomedoarquivo.txt
diretório/
*.html
!example.html
*~
```

Lembre-se que é possível ignorar um arquivo dentro de outro diretório, basta adicionar o caminho do diretório.

GitHub

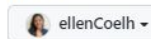
.gitignore

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?

[Import a repository.](#)

Owner *

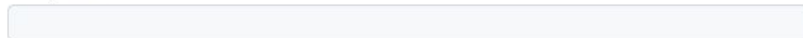


Repository name *



Great repository names are short and memorable. Need inspiration? How about [reimagined-spoon?](#)

Description (optional)



Public

Anyone on the internet can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.

Initialize this repository with:

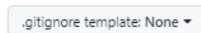
Skip this step if you're importing an existing repository.

☐ Add a README file

This is where you can write a long description for your project. [Learn more.](#)

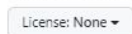
Add .gitignore

Choose which files not to track from a list of templates. [Learn more.](#)



Choose a license

A license tells others what they can and can't do with your code. [Learn more.](#)



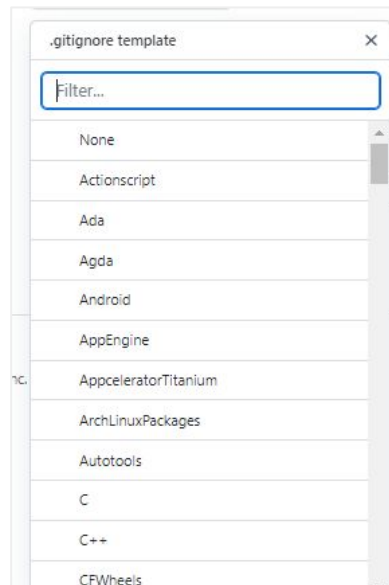
You are creating a public repository in your personal account.

Create repository

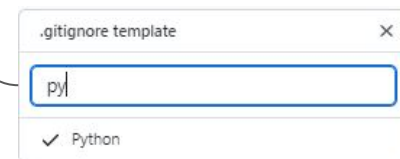
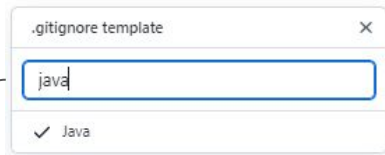
Selecionar quais arquivos não rastrear.

GitHub

.gitignore

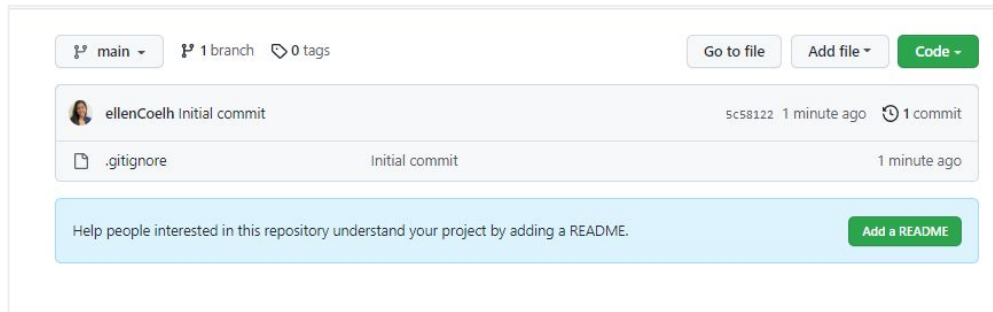


Seleciona-se a principal




Várias opções de template


Ao criar o repositório







GitHub .gitignore

Já possui os arquivos que devem ser ignorados, baseado no template selecionado

 main [ignore / .gitignore](#) Go to file ...

 **ellenCoelh** Initial commit Latest commit 5c58122 2 minutes ago [History](#)

 1 contributor

23 lines (18 sloc) | 278 Bytes Raw Blame   

```
1 # Compiled class file
2 *.class
3
4 # Log file
5 *.log
6
7 # BlueJ files
8 *.ctxt
9
10 # Mobile Tools for Java (J2ME)
11 .mtj.tmp/
12
13 # Package Files #
14 *.jar
15 *.war
16 *.nar
17 *.ear
18 *.zip
19 *.tar.gz
20 *.rar
21
22 # virtual machine crash logs, see http://www.java.com/en/download/help/error_hotspot.xml
23 hs_err_pid*
```

[Give feedback](#)

**Crie 3 repositórios diferentes escolhendo um tipo de template, observe os arquivos de devem ser ignorados para cada um dos casos.
São os mesmos?**

To practice 



C . E . S . A . R

Pessoas impulsionando inovação.
Inovação impulsionando negócios.



Ellen Coelho

<https://github.com/ellenCoelh>



Ronnie de Souza Santos

drdesouzasantos.ca

