* **Clonar um repositorio do GitHub**

**git clone [url]**

**Exemplo: git clone https://github.com/luc-git-projects/comandos.git**

* **Ver situação dos arquivos**

**git status**

* **Rastrear arquivos**

**git add [nome-arquivo]**

**Exemplo: git add index.html**

* **Rastrear todos os arquivos**

**git add .**

* **Commitar arquivos**

**git commit –m “[aqui fica a mensagem do commit]”**

**Exemplo: git commit –m “Arquivo inicial de citacoes”**

* **Rastrear e commitar arquivos de uma vez**

**git commit –a –m “[aqui fica a mensagem do commit]”**

**Exemplo: git commit -a -m "Inserindo titulo e diminuindo tamanho da pagina"**

**git commit –am “[aqui fica a mensagem do commit]”**

**Exemplo: git commit -am "Inserindo titulo e diminuindo tamanho da pagina"**



* **Buscar alterações para o GitHub**

**git pull**

* **Enviar alterações para o repostório remoto**

**git push**

* **Ver historico de alterações gravadas no repositório**

**git log**

* **Mostrar quantidade de logs que quisermos (são mostrados os ultimos commits)**

**git log –n [quantidade de logs a serem mostrados]**

**Exemplo: git log –n 2 (mostra os dois ultimos commits)**

* **Mostrar commits bem consisos**

**git log - -oneline**

* **Mostrar resumo de arquivos alterados e quantidade de linhas adiconadas e removidas**

**git log - -stat**

* **Mostrar as diferenças entre os arquivos anteriores a modificação e os arquivos que foram alterados que se encontram no diretório de trabalho**

**git diff**

* **Mostrar as diferenças entre um arquivo especifico antes da modificação e o arquivo que foi alterado que se encontra no diretório de trabalho**

**git diff [nome do arquivo]**

**Exemplo: git diff index.html**

* **Mostrar as diferenças entre os arquivos que estão na area de stage**

**git diff - -staged**

* **Mostrar as diferenças em um arquivo especifico na area de stage**

**git diff - –staged [nome do arquivo]**

**Exemplo:** **git diff - -staged index.html**

* **Mostrar as diferenças entre um arquivo especifico que foi comitado anteriormente e o arquivo alterado**

**git diff [nome do arquivo]**

**Exemplo: git diff index.html**

* **Mostrar as diferenças nos arquivos no diretorio de trabalho e na area de stage**
* **Primeiro temos que conseguir o código do último commit**
* **git log –n 1 - -oneline**
* **Depois usar o comando diff com o código do comit**
* **Git diff [código do commit]**
* **Exemplo: git diff 039e627**
* **Mostrando as diferenças entre intervalos de commits (arquivos tem que estar commitados)**

**git diff [código do commit anterior]..[código do commit posterior]**

**Exemplo: git diff 039e627..877c170**

* **Mostrando as diferenças entre intervalos de commits (mostra qualquer alteração, mesmo os arquivos não sendo commitados e eles podem estar no diretorio de trabalho ou na área de stage)**

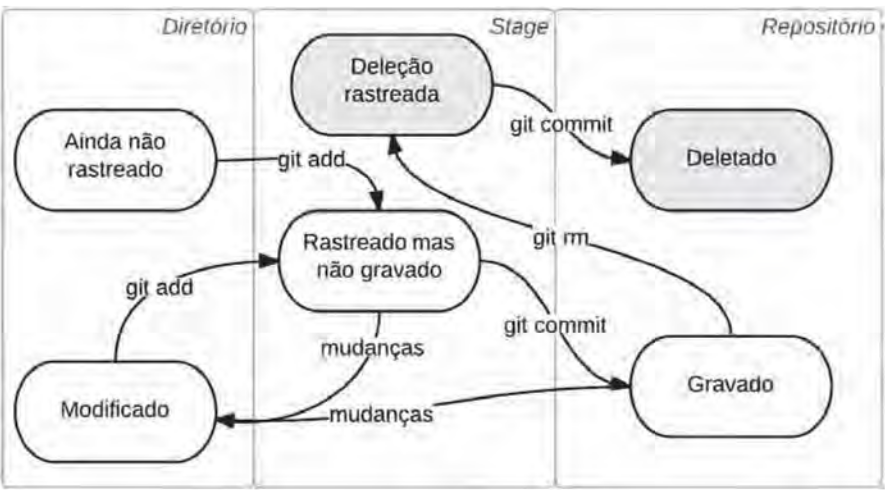
**git diff [codigo do ultimo commit que deseja ver]~[quantidade de commits anteriores]**

**Exemplo: git diff 877c170~2**

* **Remoção do arquivo, mas arquivo ja tem que estar commitado**

**git rm [nome do arquivo]**

**Exemplo: git rm produtos.html**



* **Mudar nome de arquivo**

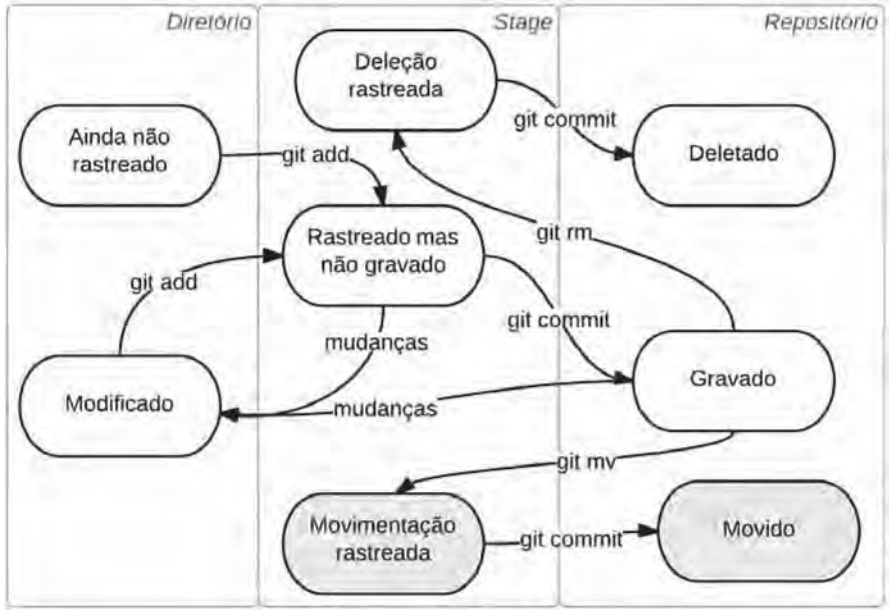
**git mv [nome do arquivo que vai ser mudado] [novo nome do arquivo]**

**Exemplo: git mv estilos.css principal.css**

* **Movendo arquivo para outro diretório (diretório já tem que existir)**

**git mv [nome do arquivo] [diretório/nome do arquivo]**

**Exemplo: git mv principal.js js/principal.js**



* **Desfazendo aterações não rastreadas no diretório de trabalho (arquivos dentro da área de stage não serão afetados)**

**git checkout - - [nome do arquivo]**

**Exemplo: git checkout -- index.html**

* **Recuperar arquivos que foram removidos acidentalmente**

**git checkout - - [nome do arquivo]**

**Exemplo: git checkout -- index.html**

* **Ggit branch alterações de arquivos na área de stage (retira o arquivo da área de stage, mas as mudanças efetuadas no arquivo seram preservadas)**

**git reset - - [nome do arquivo]**

**Exemplo: git reset -- index.html**

* **Removendo todas as aterações de todos os arquivos na área de stage (o repositório volta como estava no último commit)**

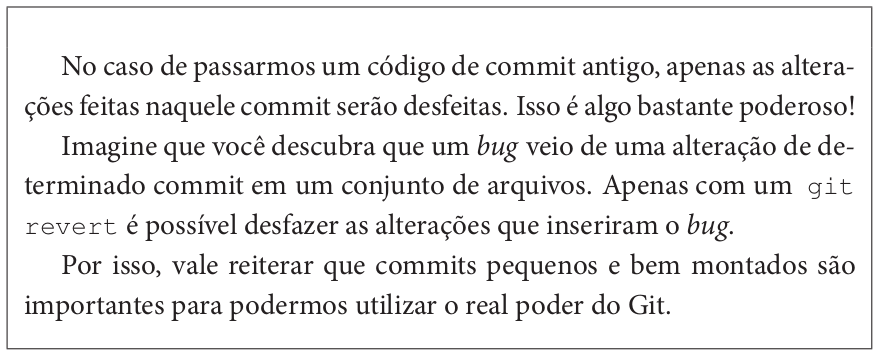
**‘ git reset - -hard**

* **Removendo todas as alterações de todos os arquivos que já foram commitados (se não colocarmos a opção - -no-edit um editor de texto sera aberto para editarmos a mensagem do novo commit)**

**git revert - -no-edit [ código do último commit ou utilizar HEAD]**

**Exemplo: git revert - -no-edit caa57e4**

**Exemplo: git revert - -no-edit HEAD**

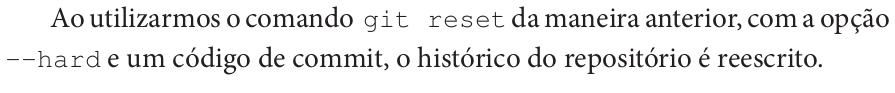


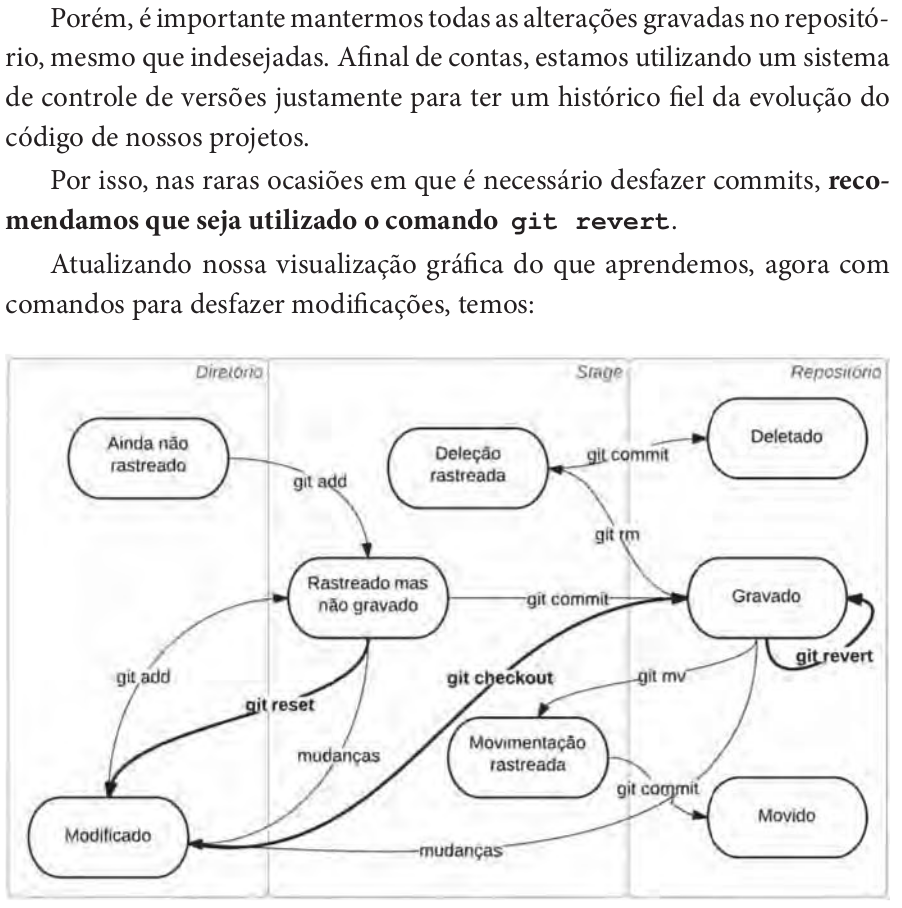
* **Removendo todas as alterações de todos os arquivos que foram commitados. (Vai voltar ao commit anterior)**

**git revert - -hard [ código do último commit ou utilizar HEAD]**

**Exemplo: git revert - -hard caa57e4**

**Exemplo: git revert - -hard HEAD**





* **Listar branchs no nosso repositório**

**git branch**

* **Listar últimos commits associados com as branchs**

**git branch -v**

* **Listar commits e seus respectivos pais (commit pai é o commti que foi efetuado ateriormente)**

**git log --parents**

* **Listar commits e seus respectivos pais de uma fora mais concisa (commit pai é o commti que foi efetuado ateriormente)**

**git log --oneline --decorate --parents**

**Resposta:**

**b92285b a5b9fce (HEAD -> main) Revert "Adicionando texto peculiar"**

**a5b9fce 2259f55 Adicionando texto peculiar**

**2259f55 d211fac Movendo principal.js**

**d211fac 515da0a Renomeando CSS**

**515da0a e9ca4c6 Removendo página de produtos**

**e9ca4c6 793e567 Página de produtos**

**793e567 0f2b749 Diminuindo intervalo de troca de banner**

**0f2b749 7e1c195 Banner ao abrir pagina**

**7e1c195 9a20490 Inserindo arquivo principal.js**

**9a20490 83b60c3 Script de troca de banner**

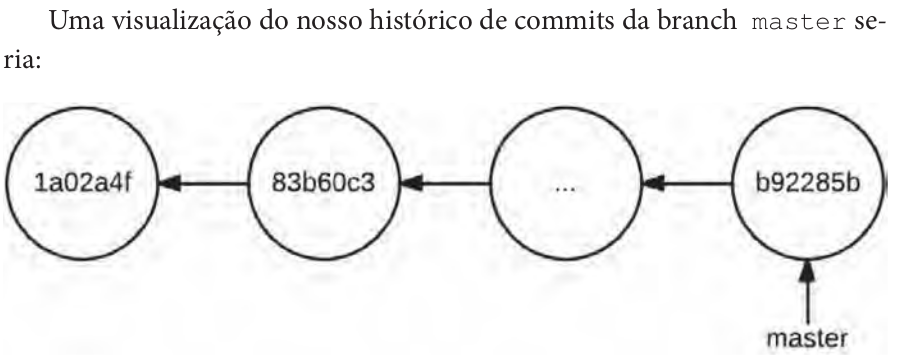
**83b60c3 1a02a4f Inserindo titulo e diminuindo tamanho da pagina**

**1a02a4f Commit inicial**

**Obs: Observe que a resposta do comando anterior apresenta três colunas: o código do commit, o código do pai e a mensagem do commit. Por exemplo, podemos ver que o commit pai do commit d211fac (“Renomeando CSS”) foi o commit 515da0a (“Removendo página de produtos”)**

**Obs: Note que o nosso primeiro commit 1a02a4f (“Commit inicial”) possui apenas duas colunas, indicando que esse commit não possui um commit pai**

**Obs: Além disso podemos ver que a branch main aponta para o último commit, de código b92285b. Não se preocupe com o HEAD, já que veremos a seguir seu significado**



* **Criando uma nova branch**

**git branch [ nome da branch que vai ser criada ]**

**Exemplo: git branch design**

**Obs: Não sera exibida nenhuma resposta. Se listarmos as branchs com o comando**

**git branch**

**As branchs serão listadas**

**design**

**\* main**

**Temos duas branchs, a branch atual que esta em uso esta destacada com um**

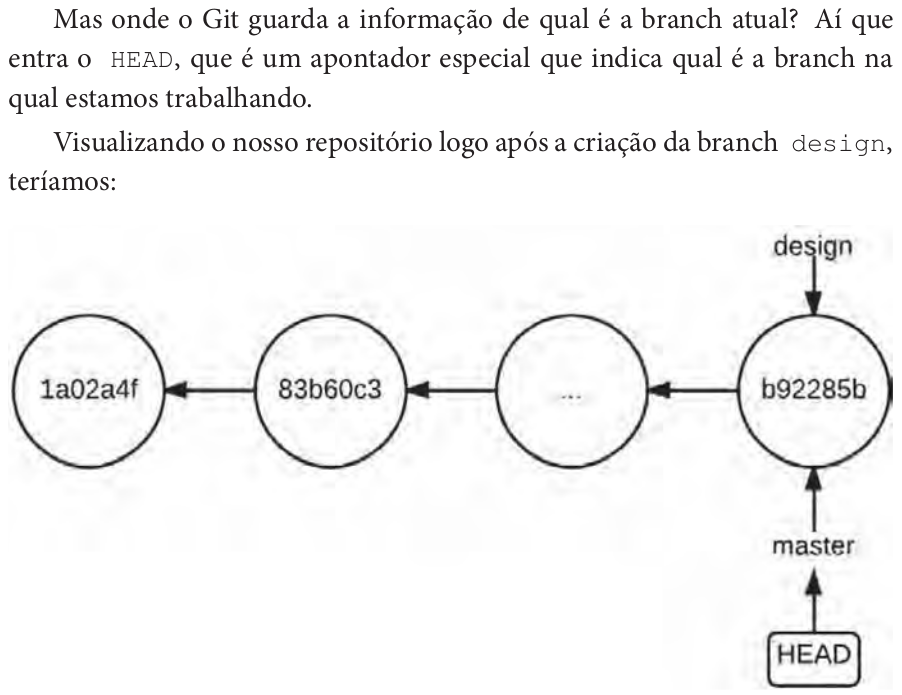
**asterisco ( \* )**

**A nova branch que que acabanos de criar ( design ) paonta para o mesmo commit que a branch que estávamos anteriormente ( main ) podemos verificar isso com o comando:**

**git branch -v**

**design b92285b Revert "Adicionando texto peculiar"**

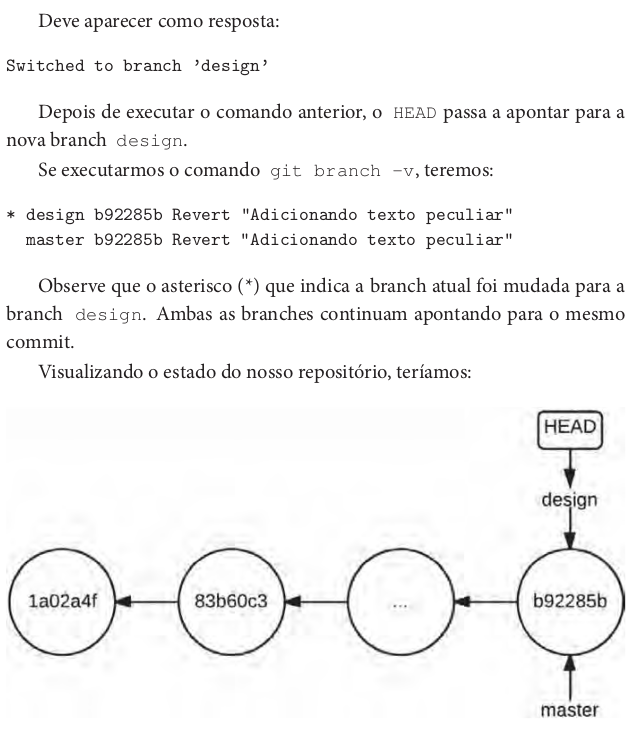
**\* master b92285b Revert "Adicionando texto peculiar"**

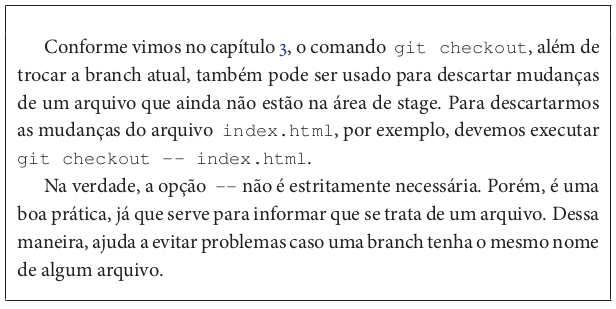


* **Trocando de branch**

**git checkout [ nome da branch para que se deseja trocar ]**

**Exemplo: git checkout design**





* **Criando e já trocando de branch**

**git checkout -b [ nome da branch que se deseja criar e já trocar para essa branch]**

**Exemplo: git checkout -b loja**

* **Deletando uma branch**
* **Para deletar uma branch não podemos estar na branch, ou seja, ela não pode ser nossa branch atual, então vamos listar as branchs e ver em qual das branchs nos estamos, a branch que nos estamos, ou seja, a branch atual, possui um asterisco ( \* ) então vamos listar as branchs com o comando abaixo:**

**git branch -v**

**Obs: se estivermos na branch que queremos deletar temos que trocar de branch, para trocar de branch usamos o comando abaixo:**

**git checkout [ nome da branch para que se deseja mudar ]**

**Exemplo: git checkout main**

* **Para deletar uma branch sem commits usamos o comando abaixo:**

**git branch -d [ nome da branch que deseja deletar ]**

**Exemplo: git branch -d loja**

* **Para deletar uma branch que possui commits usamos o comando abaixo:**

**git branch -D [ nome da branch que deseja deletar ]**

**Exemplo: git branch -D loja**

* **Commitando código em uma nova branch**
* **Para commitar código em uma nova branch, temos que saber se estamos na branch que queremos commitar o código, ou seja, ela tem que ser nossa branch atual, a branch atual possui um asterisco ( \* ) então vamos listar as branchs com o comando abaixo:**

**git branch -v**

**Obs: se não estivermos na branch que queremos commitar o código, temos que trocar para essa branch com o comando abaixo:**

**git checkout [ nome da branch que você quer fazer o commit ]**

**Exemplo: git checkout design**

**Obs: ao executarmos o comando abaixo:**

**git log -n 3 --oneline --decorate --parents**

**Teríamos a seguinte saida:**

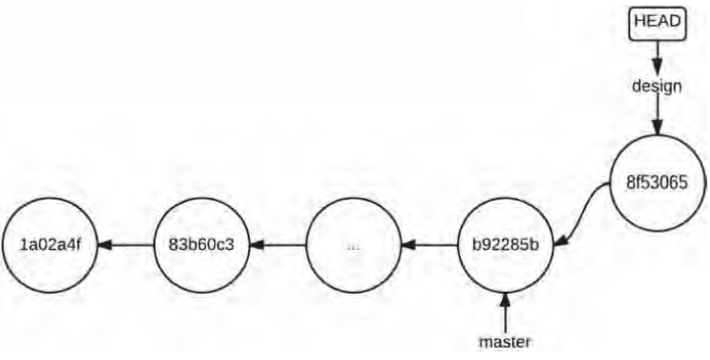
**8f53065 b92285b (HEAD, design) Adicionando Bootstrap**

**b92285b a5b9fce (main) Revert "Adicionando texto peculiar"**

**a5b9fce 2259f55 Adicionando texto peculiar**

**Note que a branch main ainda continua apontando para o commit de código b92285b (Revert "Adicionando texto peculiar")**

**Já a branch design passou a apontar para o novo commi, de código 8f53065 (Adicionando Bootstrap). Observe que o HEAD aponta para o mesmo commit**



**Obs: Preste atenção que quando fazemos checkout para um outra branch com o comando abaixo.**

**git checkout main**

**O HEAD se move para o ultimo commit da branch para a qual mudamos você pode observar isso dando o comando de log abaixo:**

**git log -n 3 - -oneline**

* **Verificando branches não mescladas (temos que estar na branch main)**

**git branch - –no-merged**

* **Mesclando branchs com a branch main**

**OBS: Temos que estar na branch main para fazermos a mesclagem, para isso você deve verificar em qual branch você esta atualmente. A branch em que você esta atualmente possui um asterisco ( \* ) na frente dela. Para verificar em qual branch você se encontra atualmente use o comando abaixo:**

**git branch**

**Exemplo de resultado do comando acima:**

**design**

**\* main**

**OBS: Note acima que a branch atual, ou seja, a branch em que estamos é a branch main, então podemos, fazer a mesclagem da branch main com a branch design, caso estivessemos na branch design teriamos que ir para branch main com o comando abaixo:**

**git checkout main**

* **Comando de commit**

**git merge [branch que vai mergiar] -m “[ mensagem de commit ]”**

**Exemplo: git merge design -m "Mesclando com a branch design"**

**OBS: Usando o comando abaixo que já foi explicado anteriormente, podemos ver os últimos 4 commits de forma concisa, mostrando o código do commit, o código do pai do commit e a mensagem do commit**

**git log -n 4 --oneline --decorate --parents**

**Exemplo de resultado:**

**e6b2f35 2223859 8f53065 (HEAD, main) Mesclando com a branch design**

**‘ 2223859 b92285b Alterando textos do banner**

**8f53065 b92285b (design) Adicionando Bootstrap**

**b92285b a5b9fce Revert "Adicionando texto peculiar"**

**Observe que a branch main e o HEAD apontam para o commit de**

**merge, que é o último commit realizado, e que tem o código e6b2f35 e a**

**mensagem que havíamos informado.**

**Já a branch design continua apontando para o mesmo commit que es-**

**tava anteriormente.**

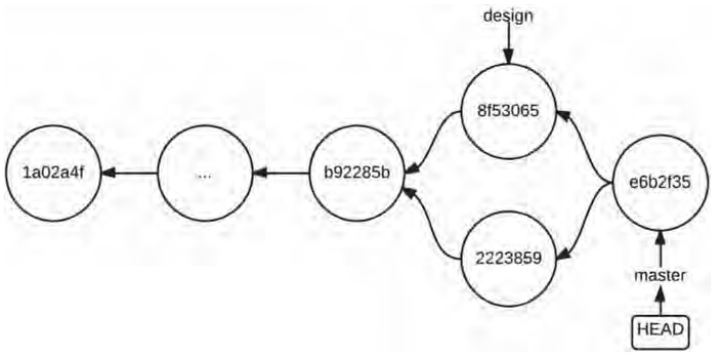
**É importante notar que o commit de merge ( e6b2f35) tem dois commits**

**pai: o commit 2223859, de mensagem “Alterando textos do banner”, e o**

**commit 8f53065, de mensagem “Adicionando Bootstrap”.**

**Visualizando o estado do nosso repositório depois do merge da branch**

**design na branch master, teríamos:**



**“teste”SSS**