**1. O Squash**

* Comando: git rebase -i HEAD~5

a1b2c3d (HEAD -> main) Feature finalizada

4e5f6a7 Ops, corrige erro de digitacao da linha 3

8b9c0d1 Adiciona linha

3 d4e5f6a Adiciona mais trabalho

7f8g9h0 Começa a funcionalidade em file.txt

* + O Git executou o "squash". Abriu o editor de texto, contendo *todas* as mensagens dos 5 commits e pediu para criar a *mensagem única*.
  + Mensagens alterada por “Agrupamento de 5 commits”

**2. Verificando o Log**

* How does git log look now?
  + Comando: git log --oneline --graph
  + O log impo.

**3. A Limpeza**

A limpeza final (remover \n do file.txt), mas sem adicionar um novo commit ao histórico. A solução para isso é "emendar" (amend) o commit que acabamos de criar.

* Passo 1: Faça a mudança
  + Abra o file.txt no seu editor.
  + Remova os caracteres \n indesejados.
  + Salve o arquivo.
* Passo 2: Adicione ao Staging
  + Comando: git add file.txt
* Passo 3: "Emende" o Commit Anterior
  + Comando: git commit --amend
* O comando --amend (emendar/corrigir) não cria um commit. Em vez disso, ele pega suas novas mudanças (a limpeza do \n) e as *adiciona ao commit anterior* (o commit "squashed" que acabamos de criar), substituindo-o por um novo commit que contém *ambas* as mudanças.
* No final rodar git log --oneline de novo, o histórico *continua* parecendo ter apenas aquele um novo commit, mas agora ele inclui a limpeza do \n.

\*a1b2c3d (HEAD -> main) Feature finalizada

|

\*4e5f6a7 Ops, corrige erro de digitacao da linha 3

|

\*8b9c0d1 Adiciona linha

|

\*3 d4e5f6a Adiciona mais trabalho

|

\*7f8g9h0 Começa a funcionalidade em file.txt