# Code Metrics (com recurso ao *plugin* MetricsReloaded para o IntelliJ IDEA)

## Lines of Code

### Explicação das métricas recolhidas

O conjunto de métricas oferecido pelo *plugin* que foi escolhido denomina-se de “*Lines of code metrics*”. Este conjunto foi desenvolvido seguindo a métrica que se denomina pelo mesmo nome – *Source Lines of Code* (SLOC), ou *Lines of Code* (LOC).

Ora, por linha de código entende-se qualquer linha de texto num ficheiro de código que não seja um comentário nem uma linha, mas também linhas de cabeçalho, caso existam número nas declarações ou em fragmentos de declarações dessa linha. Assim, a métrica SLOC é uma métrica de software usada para medir o tamanho de um programa contando, principalmente, com o número de linhas no ficheiro do código.

Para cada caso, observamos que foram medidos valores diferentes. O nome explica por si, no fundo, a medida usada. Temos:

* Para Métodos:
  + Número de linhas em comentário (CLOC)
  + Número de linhas de *Javadoc* (JLOC)
  + Número de linhas de código (LOC)
  + Número de linhas que não são comentário (NCLOC)
  + Percentagem de linhas de código relativas (RLOC)
* Para Classes:
  + mero de linhas em comentário (CLOC)
  + Número de linhas de *Javadoc* (JLOC)
  + Número de linhas de código (LOC)
* Para Interfaces:
  + Número de linhas em comentário (CLOC)
  + Número de linhas de *Javadoc* (JLOC)
  + Número de linhas de código (LOC)
  + Número de linhas que não são comentário (NCLOC)
* Para Pacotes:
  + Número de linhas em comentário (CLOC)
  + Número de linhas em comentário (recursivo) (CLOC(rec))
  + Número de linhas de *Javadoc* (JLOC)
  + Número de linhas de *Javadoc* (recursivo) (JLOC(rec))
  + Número de linhas de código (LOC)
  + Número de linhas de código (recursivo) (LOC(rec))
  + Número de linhas de código-produto (LOCp)
  + Número de linhas de código-produto (recursivo) (LOCp(rec))
  + Número de linhas de código-teste (LOCt)
  + Número de linhas de código-teste (recursivo) (LOCt(rec))
  + Número de linhas que não são comentário (NCLOC)
  + Número de linhas que não são comentário (produto) (NCLOCp)
  + Número de linhas que não são comentário (produto, recursivo) (NCLOCp(rec))
  + Número de linhas que não são comentário (teste) (NCLOCt)
  + Número de linhas que não são comentário (teste, recursivo) (NCLOCt(rec))
* Para Módulos:
  + Número de linhas de *Javadoc* (JLOC)
  + Número de linhas escritas em *Groovy* (L(Groovy))
  + Número de linhas escitas em *HTML* (L(HTML))
  + Número de linhas escritas em *Java* (L(J))
  + Número de linhas escritas em *Kotlin* (L(KT))
  + Número de linhas escritas em *XML* (L(XML))
  + Número de linhas de código (LOC)
  + Número de linhas de código-produto (LOCp)
  + Número de linhas de código-teste (LOCt)
  + Número de linhas que não são comentário (NCLOC)
  + Número de linhas que não são comentário (produto) (NCLOCp)
  + Número de linhas que não são comentário (teste) (NCLOCt)
* Para Tipos de Ficheiro:
  + Número de linhas de código (LOC)
  + Número de linhas que não são comentário (NCLOC)
* Para todo o Projeto:
  + Número de linhas em comentário (CLOC)
  + Número de linhas de *Javadoc* (JLOC)
  + Número de linhas escritas em *Groovy* (L(Groovy))
  + Número de linhas escitas em *HTML* (L(HTML))
  + Número de linhas escritas em *Java* (L(J))
  + Número de linhas escritas em *Kotlin* (L(KT))
  + Número de linhas escritas em *XML* (L(XML))
  + Número de linhas de código (LOC)
  + Número de linhas de código-produto (LOCp)
  + Número de linhas de código-teste (LOCt)
  + Número de linhas que não são comentário (NCLOC)
  + Número de linhas que não são comentário (produto) (NCLOCp)
  + Número de linhas que não são comentário (teste) (NCLOCt)

### Potenciais locais “problemáticos”

* Para Métodos, observamos que **apenas** 4.26% (RLOC) do código são linhas consideravelmente relevantes para o código:

Application

Description automatically generated with low confidence

* Para Módulos, apenas aproximadamente **metade** das linhas de código (LOC) fazem parte de código-produto (LOCp), ou seja, apenas metade é útil:

Graphical user interface, application

Description automatically generated

### Relação mantida com os *Code Smells* identificados

Os problemas apresentados no ponto anterior, em conjunto com análise dos restantes resuldos propostos pelo decorrer da métrica, conferem alguns dos pontos que foram apresentados pelo nosso grupo na fase anterior como *code smells*, tais como:

* “Código antigo em comentário”
* “Comentário a relembrar um raciocínio”
* “TODO comentado”
* “Excesso de comentários”
* “Classe com múltiplos métodos, definida dentro de uma interface”
* “Classe vazia”
* “Classe exaustivamente extensa”
* “Método demasiado grande e complexo”
* “Duplicação de código”
* “Métodos não utilizados (Dead Code)”
* “Valores que deviam ser constantes”