**Repositorios**

### [alibaba / fescar](https://github.com/alibaba/fescar)

### Sub-pastas do projeto = 275 pastas

### Classes do projeto = 381 classes

### Linhas de codigo do projeto = 46927 linhas

### [spring-projects / spring-boot](https://github.com/spring-projects/spring-boot)

Sub-pastas do projeto = 3235 pastas

Classes do projeto = 4600 classes

Linhas de codigo do projeto = 463080 linhas

### [spring-projects / spring-framework](https://github.com/spring-projects/spring-framework)

Sub-pastas do projeto = 1816 pastas

Classes do projeto = 7066 classes

Linhas de codigo do projeto = 1140182 linhas

### [elastic / elasticsearch](https://github.com/elastic/elasticsearch)

Sub-pastas do projeto = 4821 pastas

Classes do projeto = 11017 classes

Linhas de codigo do projeto = 1914238 linhas

### [iluwatar / java-design-patterns](https://github.com/iluwatar/java-design-patterns)

Sub-pastas do projeto = 1791 pastas

Classes do projeto = 1126 classes

Linhas de codigo do projeto = 75589 linhas

### [kdn251 / interviews](https://github.com/kdn251/interviews)

Sub-pastas do projeto = 48 pastas

Classes do projeto = 515 classes

Linhas de codigo do projeto = 22271 linhas

### [TheAlgorithms / Java](https://github.com/TheAlgorithms/Java)

Sub-pastas do projeto = 26 pastas

Classes do projeto = 158 classes

Linhas de codigo do projeto = 15045 linhas

### [google / guava](https://github.com/google/guava)

Sub-pastas do projeto = 327 pastas

Classes do projeto = 3155 classes

Linhas de codigo do projeto = 773194 linhas

### [apache / incubator-dubbo](https://github.com/apache/incubator-dubbo)

Sub-pastas do projeto = 1714 pastas

Classes do projeto = 1631 classes

Linhas de codigo do projeto = 194710 linhas

### [ityouknow / spring-boot-examples](https://github.com/ityouknow/spring-boot-examples)

Sub-pastas do projeto = 517 pastas

Classes do projeto = 241 classes

Linhas de codigo do projeto = 8117 linhas

### [eugenp / tutorials](https://github.com/eugenp/tutorials)

Sub-pastas do projeto = 12584 pastas

Classes do projeto = 9274 classes

Linhas de codigo do projeto = 436159 linhas

### [TommyLemon / APIJSON](https://github.com/TommyLemon/APIJSON)

Sub-pastas do projeto = 243 pastas

Classes do projeto = 338 classes

Linhas de codigo do projeto = 61062 linhas

[**jhipster**](https://github.com/jhipster)**/**[**generator-jhipster**](https://github.com/jhipster/generator-jhipster)

Sub-pastas do projeto = 590 pastas

Classes do projeto = 2 classes

Linhas de codigo do projeto = 179 linhas

# [rovo89](https://github.com/rovo89)/[XposedBridge](https://github.com/rovo89/XposedBridge)

Sub-pastas do projeto = 52 pastas

Classes do projeto = 70 classes

Linhas de codigo do projeto = 33435 linhas

# [bastillion-io](https://github.com/bastillion-io)/[Bastillion](https://github.com/bastillion-io/Bastillion)

Sub-pastas do projeto = 32 pastas

Classes do projeto = 61 classes

Linhas de codigo do projeto = 11304 linhas

Selecionar repositórios que possuem maior Reconhecimento dentro da própria plataforma do GitHub, uma vez que as estrelas funcionam como ferramenta de qualificação.

Número de forks também é utilizado como critério de seleção de repositórios populares uma vez que mensuram a quantidade de desenvolvedores trabalhando paralelamente em diferentes instancias daquele código.

Foi usado nos gráficos somente duas métricas, por exemplo classe e forks, as estrelas estão nas tabelas como forma de representar popularidade/aceitação da comunidade github.

Em um gráfico de dispersão só se utilizamos duas métricas conforme mostrado abaixo.

Linhas / forks

75589 / 15557 -> teremos 4,85 linhas para cada pessoa que está contribuindo no repositório, criando assim a popularidade do repositório.

Linhas / Classes

75589 / 61 = 1240 linhas por classes, 0,08 linhas por forks ou pessoas que estão trabalhando no repositórios.

Linhas / linhas do manual

75589 / 5663 -> 13,35 linhas ou seja a cada 1 linha do manual, explicamos 13,35 linhas do código.

Linhas / condicionais.

75589 / 967 -> 78,16 linhas, ou seja a cada 78,16 linhas de código, existe uma condicional.

Nem sempre a quantidade de linhas de manual define a qualidade e a popularidade do código, ambas as características podem ser definidas através do código, um código bem endentado e bem comentando, mostrara organização e será de fácil entendimento, fazendo com o que a popularidade aumente e também atraia mais colaboradores para o código.

Logico que o tempo de postagem também influencia na quantidade de estrelas e forks;