



Pontifícia Universidade Católica de Minas Gerais
Departamento de Engenharia de Software e Sistemas de Informação

Disciplina: Processos e Qualidade de Software
Professor: Lesandro Ponciano

Alunos: Caio César S. & Carlos Eduardo Fonseca

Relatório do desenvolvimento do Projeto de Processo e Qualidade de Software

1. Planejamento do projeto.

1.1 Escolha das dimensões.

Na primeira etapa do trabalho, foi pedido para escolher 2 dimensões para a realização de uma análise, foi escolhido entre a dupla as dimensões: **Documentação por Repositório.**

1.2 Ideia proposta.

Com as dimensões escolhidas, começamos a pensar em alguma ideia envolvendo as 2 dimensões, pensamos então em analisar a quantidade de linhas comentadas em 10 repositórios diferentes, já que se entende, que um código bem comentado, é um código bem documentado na maioria dos casos, assim seria capaz de analisar o quão documentado um repositório é.


2. Desenvolvimento.

2.1 Escolha dos repositórios.

Para começar a desenvolver a ideia, a primeira coisa que tivemos que fazer era quais repositórios poderíamos analisar, depois de pesquisar chegamos à conclusão de que os repositórios analisados seriam os projetos com mais contribuidores segundo o <https://octoverse.github.com/> :

1. <https://github.com/Microsoft/vscode>
2. <https://github.com/facebook/react-native>
3. <https://github.com/npm/cli>
4. <https://github.com/angular/angular-cli>
5. <https://github.com/tensorflow/tensorflow>
6. <https://github.com/FortAwesome/Font-Awesome>
7. <https://github.com/angular/angular>
8. <https://github.com/moby/moby>
9. <https://github.com/jlord/patchwork>
10. <https://github.com/ansible/ansible>

Projects with the most contributors

	MICROSOFT/VSCODE	15K
	FACEBOOK/REACT-NATIVE	8.8K
	NPM/NPM	7.6K
	ANGULAR/ANGULAR-CLI	7.4K
	TENSORFLOW/TENSORFLOW	7.3K
	FORTAWESOME/FONT-AWESOME	6.8K
	ANGULAR/ANGULAR	6K
	DOCKER/DOCKER	6K
	JLORD/PATCHWORK	5.9K
	ANSIBLE/ANSIBLE	5.9K

2.2 Busca de dados.

Após escolher os repositórios, teríamos que pensar em como iríamos buscar os dados do repositório, após algumas pesquisas, descobrimos uma ferramenta chamada CLOC, uma ferramenta que roda no diretório raiz do repositório, e então devolve alguns dados, como a quantidade de linhas de código, a quantidade de linhas em branco, a quantidade de linhas comentadas, e entre outras diversas funções da ferramenta.

```
C:\Users\Caio\Downloads>cloc-1.78.exe C:\Users\Caio\Downloads\cloc-master\cloc-master
387 text files.
375 unique files.
78 files ignored.


github.com/AlDanial/cloc v 1.78 T=0.50 s (728.0 files/s, 95728.0 lines/s)
-----
Language             files      blank   comment      code
-----
Perl                  6          1472       2451       22191
YAML                  189         6         189       4304
Markdown              1          232        26       2261
ANTLR Grammar        2           200         59       1012
R                     3           95        312        698
C/C++ Header         2           191       780        618
C++                   4           132       173        570
Forth                 2           17         84        529
TypeScript            4           53         39        416
Logtalk               1           59         57        368
Windows Message File  2           89          9        348
```

2.3 Analise dos dados.

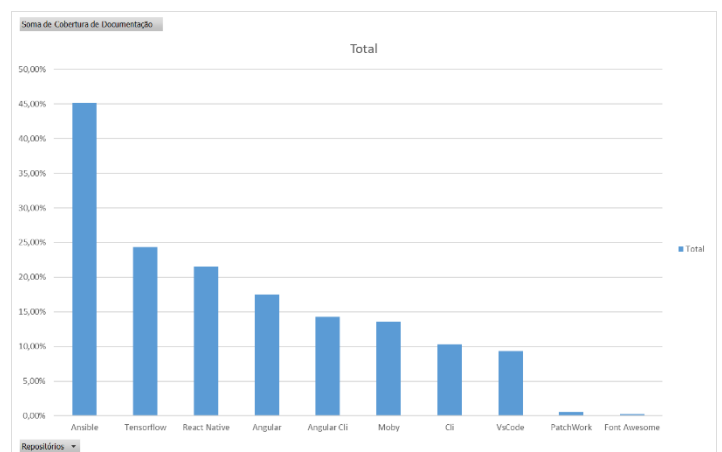
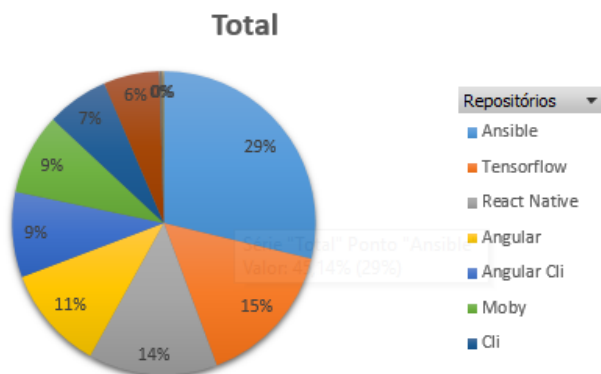
Com os dados em mãos, então partimos para a análise dos dados brutos, para essa análise, fizemos uma planilha no Excel, onde continha uma tabela com as informações buscadas do CLOC, dos repositórios selecionados.

Repo	Arq	Language	Branco	Comentadas	Code	Cobertura
Angular Cli	800	TypeScript	9537	9999	51258	19,51%
Angular Cli	135	Markdown	2315	0	9978	0,00%
Angular Cli	192	JSON	46	0	9469	0,00%
Angular Cli	32	JavaScript	109	169	786	21,50%
Angular Cli	16	HTML	43	30	323	9,29%
Angular Cli	3	YAML	32	43	257	16,73%
Angular Cli	5	EJS	47	47	224	20,98%
Angular Cli	1	Skylark	10	16	54	29,63%
Angular Cli	2	Windows Resource File	7	0	21	0,00%
Angular Cli	2	Sass	2	4	12	33,33%
Angular Cli	4	CSS	0	9	3	300,00%
Angular	3544	TypeScript	55383	63437	306037	20,73%
Angular	184	Markdown	20817	0	35450	0,00%
Angular	314	JSON	281	0	19796	0,00%
Angular	372	JavaScript	2936	3644	16991	21,45%
Angular	359	HTML	1209	985	8004	12,31%
Angular	79	CSS	823	150	4590	3,27%
Angular	49	Sass	657	70	3474	2,01%
Angular	73	Bourne Shell	567	483	2129	22,69%
Angular	16	Skylark	300	734	1392	52,73%
Angular	4	YAML	80	159	781	20,36%
Angular	6	Bourne Again Shell	38	17	204	8,33%
Angular	2	Dockerfile	43	44	156	28,21%
Angular	4	Windows Resource File	36	0	149	0,00%
Angular	1	Pascal	40	130	109	119,27%

A tabela gerada, possui os dados brutos gerados pelo CLOC, para melhor desenvolver as representações gráficas do dado, foi criado então, tabelas dinâmicas que pegam estes dados brutos, e geram os dados que alimentam os gráficos.

Repositórios	Valores					
	 Arquivos	Linguagens	Linhas Comentadas	Linhas de Código	Cobertura de Doc.	
Ansible	7630	19		371180	822278	45,14%
Tensorflow	9067	31		464010	1904537	24,36%
React Native	2938	23		58060	269578	21,54%
Angular	5009	15		69853	399278	17,49%
Angular Cli	1192	11		10317	72385	14,25%
Moby	5031	16		132313	975408	13,56%
Cli	3080	19		33921	327424	10,36%
VsCode	3034	44		69814	748728	9,32%
PatchWork	7	4		2	403	0,50%
Font Awesome	2689	8		394	149121	0,26%
Total Geral	39677	190		1209864	5669140	21,34%

Após criar as tabelas dinâmicas, então geramos as representações gráficas de cada tabela.



Análise e arquivos disponíveis em: <https://github.com/CaioCSOares/AnaliseGIT/>

