**Relatório ESII Project**

**Docentes:**

Cristóvão Sousa

Fábio Silva

Bruno Silva

**Realizado por:**

Abílio Castro - 8170054

Ricardo Cardoso - 8170278

Vitor Santos - 8170312

[**Processo de aprendizagem**](#_7puvb4oem7as) **4**

[**Configuração de Ferramentas**](#_2hkenvx8hd4g) **5**

[Youtrack:](#_bnrwnuiq8u6q) 5

[Swimlanes:](#_5t3txzbbgv5l) 5

[Colunas:](#_sv055d196mc7) 6

[Backlog:](#_ujtd5lavpo04) 8

[Sprints:](#_yjr92vxrhntu) 8

[Sprint #1](#_ep9qsxtg4rw0) 9

[Sprint #2](#_eor8ey2vmtez) 9

[Sprint Entrega Final](#_jc9nhfi04xru) 9

[Git:](#_td06ht58gf96) 9

[Máquina Virtual:](#_tj71xglrjbsa) 10

[Jenkins:](#_3qkvhnji6rrr) 11

[IntelliJ:](#_ukydzaifghfk) 13

[Upsource (opcional):](#_qhn4vbjmtghj) 13

[**O problema**](#_h2d4wcm6t3j3) **14**

[**Testes**](#_gath1gr04peg) **20**

[Método insertQuery():](#_94khqkz4p0o5) 20

[Tabela ECP:](#_gd8z1gw69jbv) 20

[Tabela BVA:](#_jwfo3fu8m563) 20

[Tabela Test Cases:](#_q04w8jltx58d) 20

[Testes em Java correspondentes às tabelas:](#_di0i40y4czhy) 21

[Método insertFile():](#_6w45tho8tz3t) 21

[Tabela ECP:](#_x4ju48ke7vak) 21

[Tabela BVA:](#_ukc3tkguywgo) 21

[Tabela Test Cases:](#_fft4tagkw33q) 21

[Testes em Java correspondentes às tabelas:](#_83p6cyf2lyd7) 22

[Método removeDigits():](#_3l7dopspkx1o) 22

[Tabela ECP:](#_4yks8w4p09vy) 22

[Tabela BVA:](#_e1yarrr4d1y8) 22

[Tabelas Test Cases:](#_aewil3ld7exv) 23

[Método removeChars():](#_45a79hdbsi2k) 23

[Tabela ECP:](#_iqcejl1dgvh) 23

[Tabela BVA:](#_2kf67zxifroe) 24

[Tabela Test Cases:](#_koekvpqhwlee) 24

[Método uniqueWords():](#_e71yjkjdekns) 25

[Tabela ECP:](#_75omk1by5l2h) 25

[Tabela BVA:](#_zcd8sah0q6tl) 25

[Tabela Test Cases:](#_htbl0eebj8cr) 25

[Testes em Java correspondentes às tabelas:](#_mmo2n3d6izri) 26

[Método matrizOrganizer():](#_8w3sclw3z4m8) 26

[Tabela ECP:](#_9dgby91ng6py) 26

[Tabela BVA:](#_p3zp7tfgno7q) 27

[Tabela Test Cases:](#_fv8x7gl9lq0j) 27

[Teste em Java correspondentes às tabelas:](#_41d88oqbz1qo) 28

[Método matrizModifier():](#_dwedcg7g5fs8) 28

[Tabela ECP:](#_ciobvvt6z7te) 28

[Tabela BVA:](#_7ctkp461bvkf) 29

[Tabela Test Case:](#_btyo495p9lqq) 29

[Testes em Java correspondentes às tabelas:](#_2y2iqgq7e4rl) 30

[Método calculoGrauS():](#_vzso4mhuqjs9) 30

[Tabela ECP:](#_ho7988b65yt7) 30

[Tabela BVA:](#_kwy5c3mg27pp) 31

[Tabela Test Cases:](#_wmqpjbykr7lf) 31

[Testes em Java correspondentes às tabelas:](#_s1yya9ogv44) 32

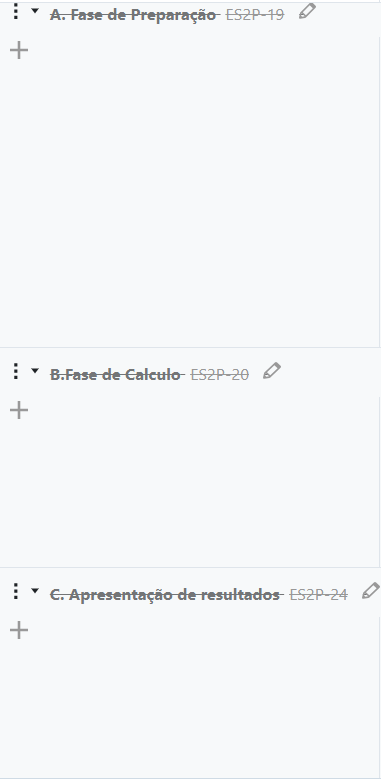
# **Processo de aprendizagem**

# **Configuração de Ferramentas**

## **Youtrack:**

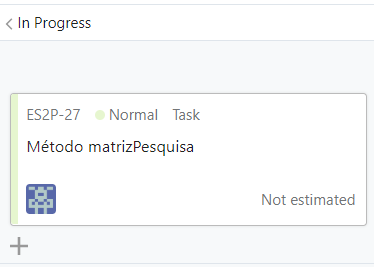
Primeiramente adicionamos todos os elementos do grupo, de seguida, adicionamos colunas e swimlanes de modo a que melhor se adaptasse ao nosso projeto:

### Swimlanes:

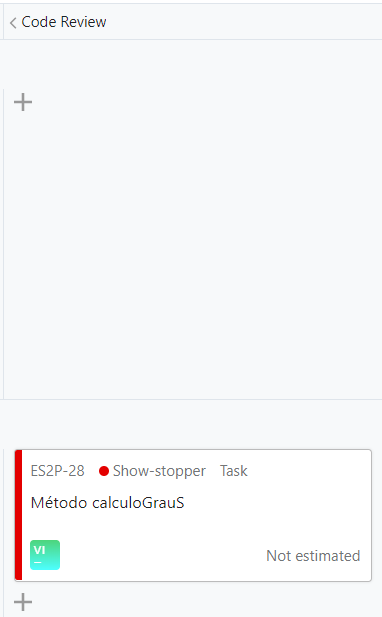


Cada Swimlane equivale a um ponto do enunciado, sendo que a fase de preparação equivalente ao ponto A do enunciado, a fase de cálculo equivalente ao ponto B do enunciado e a fase de apresentação de resultados equivalente ao ponto C do enunciado.

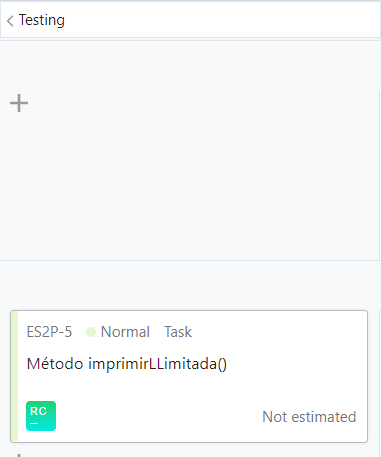
### Colunas:



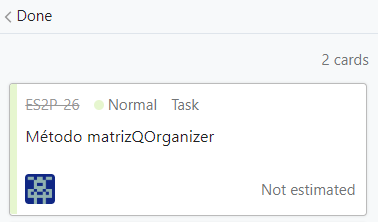
A coluna In Progress conforma-se aos métodos/funcionalidades que se encontram em desenvolvimento.



A coluna Code Review acomoda os métodos/funcionalidades que estão a ser revistas pelo code Reviewer.



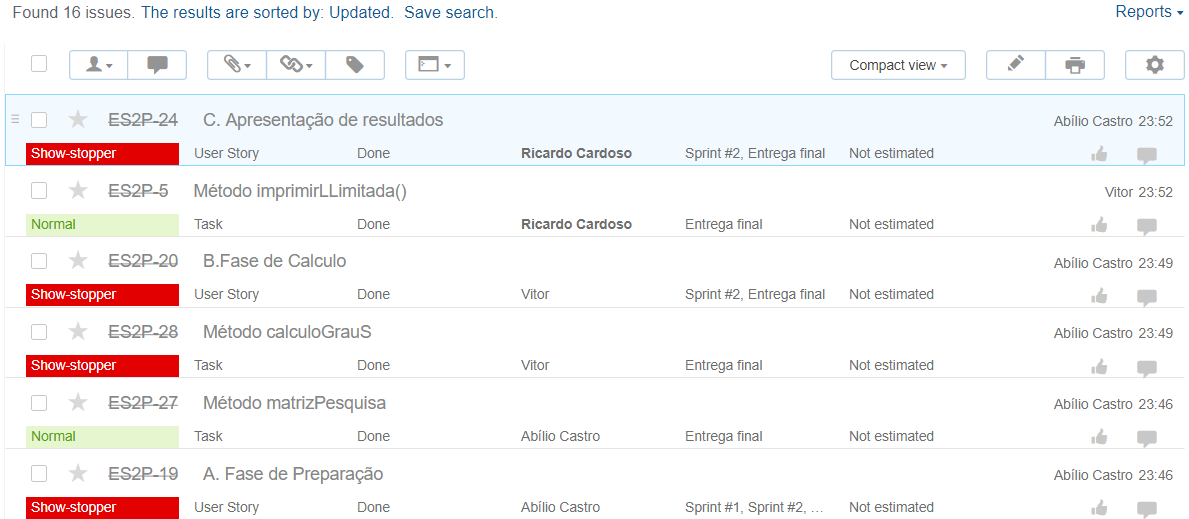
A coluna Testing é para os métodos/funcionalidades que estão a ser construídos os teste em JUnit.



A coluna Done é a última coluna, após dos métodos/funcionalidades passarem por todas as outras colunas com sucesso, são arrastados para aqui e são dados como completos.

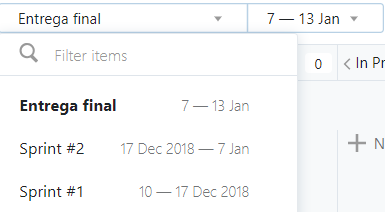
Depois da configuração do Youtrack para o nosso projeto, foi criado o Backlog e os Sprints.

### Backlog:



Nesta imagem podemos contemplar alguns dos nossos Issues no Backlog.

### Sprints:



Naquele momento criamos sprints de modo a coincidir com as revisões de sprints e com a entrega final.

Já era tarde demais quando nos apercebemos que o período dos sprints deveria ser sempre idêntico entre eles e cometemos a imprecisão do período de tempo correspondente ao sprint #1 ser de uma semana, o do sprint #2 ser de três semanas e o sprint Entrega final ser de uma semana.

### Sprint #1

Durante este Sprint abordamos a compreensão do problema, a configuração da máquina virtual e o método insertFile.

### Sprint #2

Neste Sprint foram desenvolvidos e concluídos os métodos cleanDigits, cleanChars e matrizOrganizer.

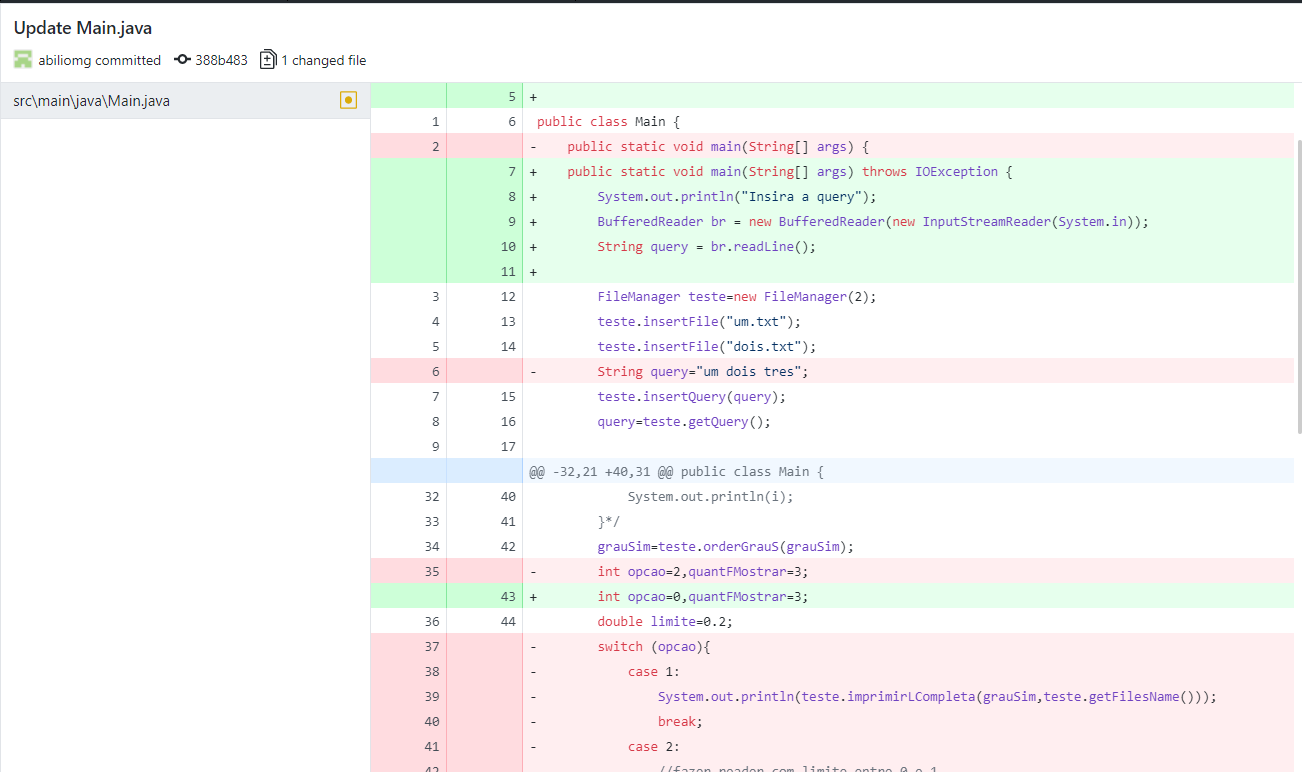
### Sprint Entrega Final

Sendo este o último Sprint, foram desenvolvidos todos os métodos que seriam necessários para o funcionamento do programa como: matrizQOrganizer, matrizPesquisa, calculoGrauS, imprimirLCompleta, imprimirLPercentagem e imprimirLLimitada.

## Git:

O GitHub foi o software utilizado para o controlo de versões, como backup na cloud e como método de transferência de dados entre diferentes developers.

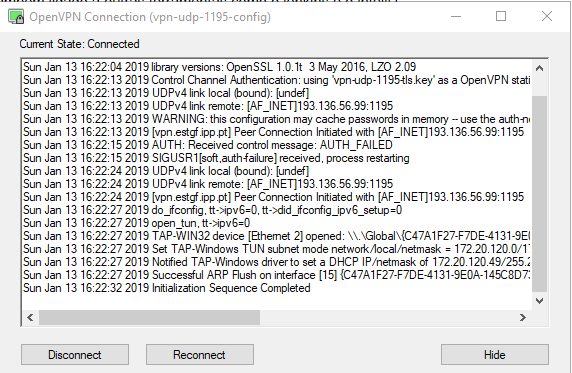
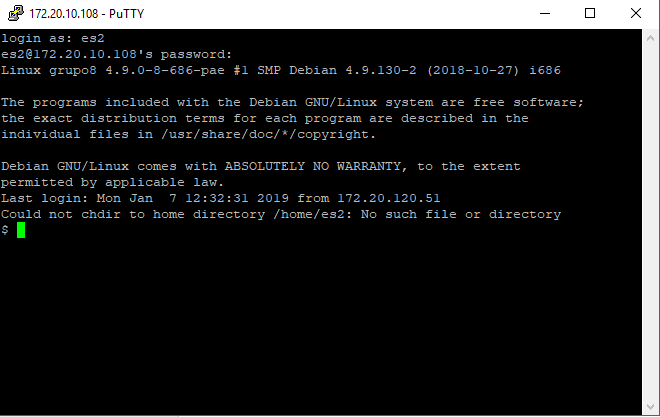
Foi efetuado também o download de um software chamado GitHub Desktop para ser mais fácil executar commits, push, pull e revert do código, desta maneira seria também mais fácil ver o histórico de alterações no código.



O repositório foi ligado a outras ferramentas como o Jenkins e o IntelliJ.

Repositório Git: https://github.com/Dumbrica/ESIIProject

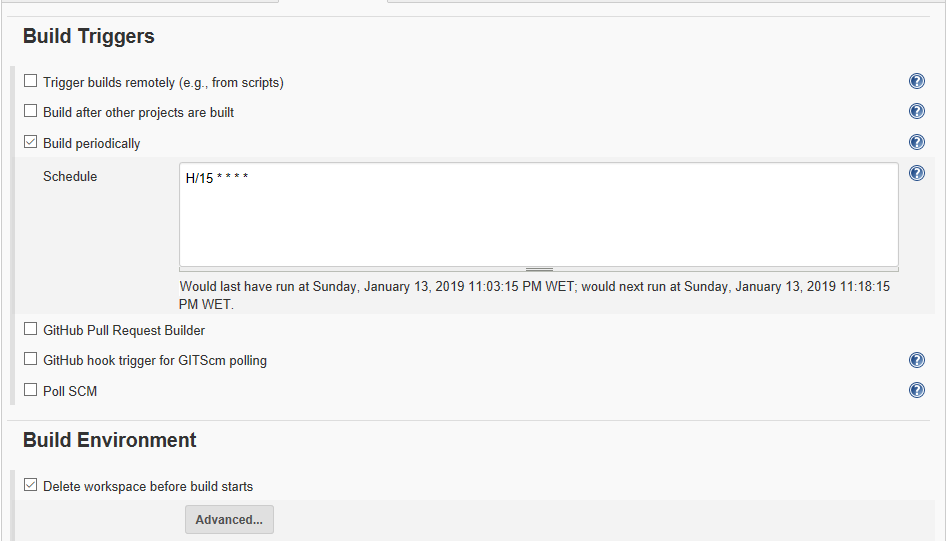
## Máquina Virtual:

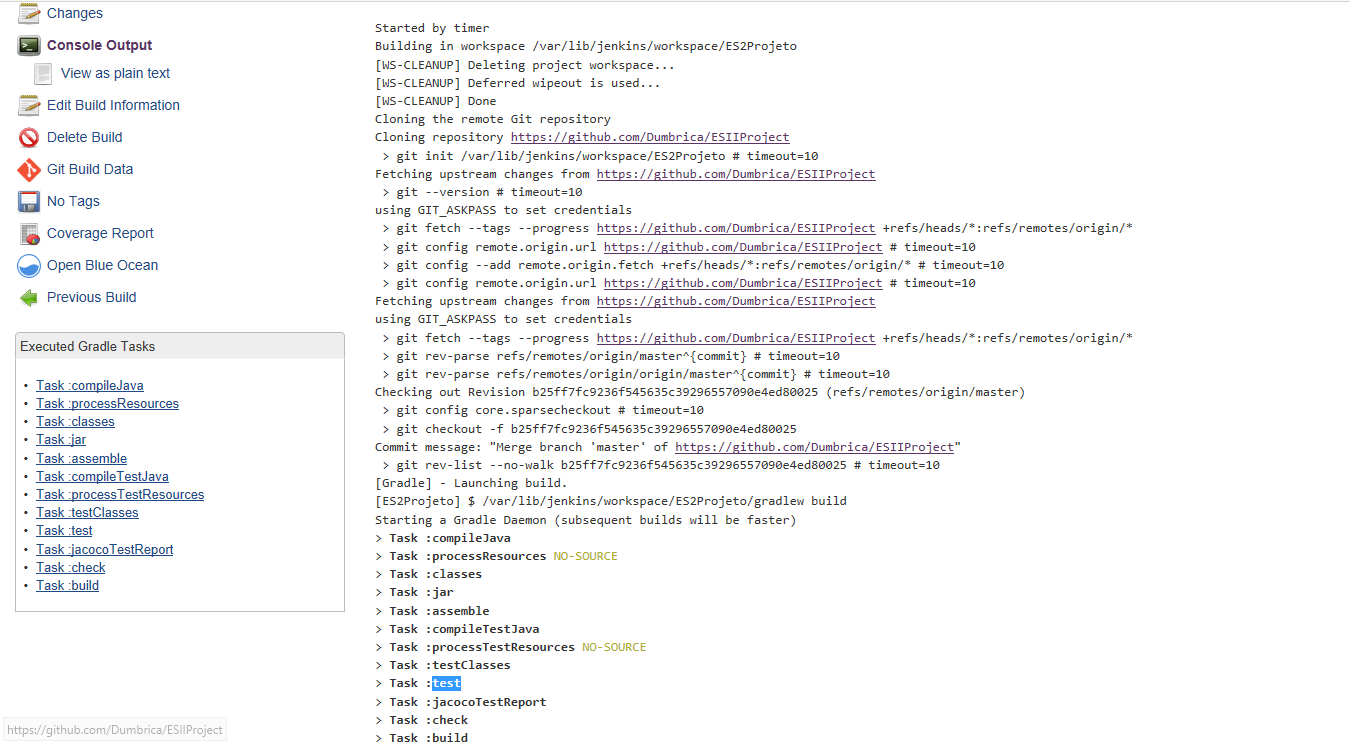
O primeiro passo foi instalar o software auxiliar Putty para podermos aceder à nossa máquina virtual, inicialmente tivemos vários problemas em relação a ligações e configurações, mas depois de muitas tentativas fomos melhorando. Nesta máquina está instalado o jenkins que é o principal uso. Tentamos instalar o upsource mas sem sucesso devido a falta de permissões. Para termos possibilidade de trabalhar fora da escola na máquina ou no jenkins, tivemos que utilizar o vpn da escola. 

## Jenkins:

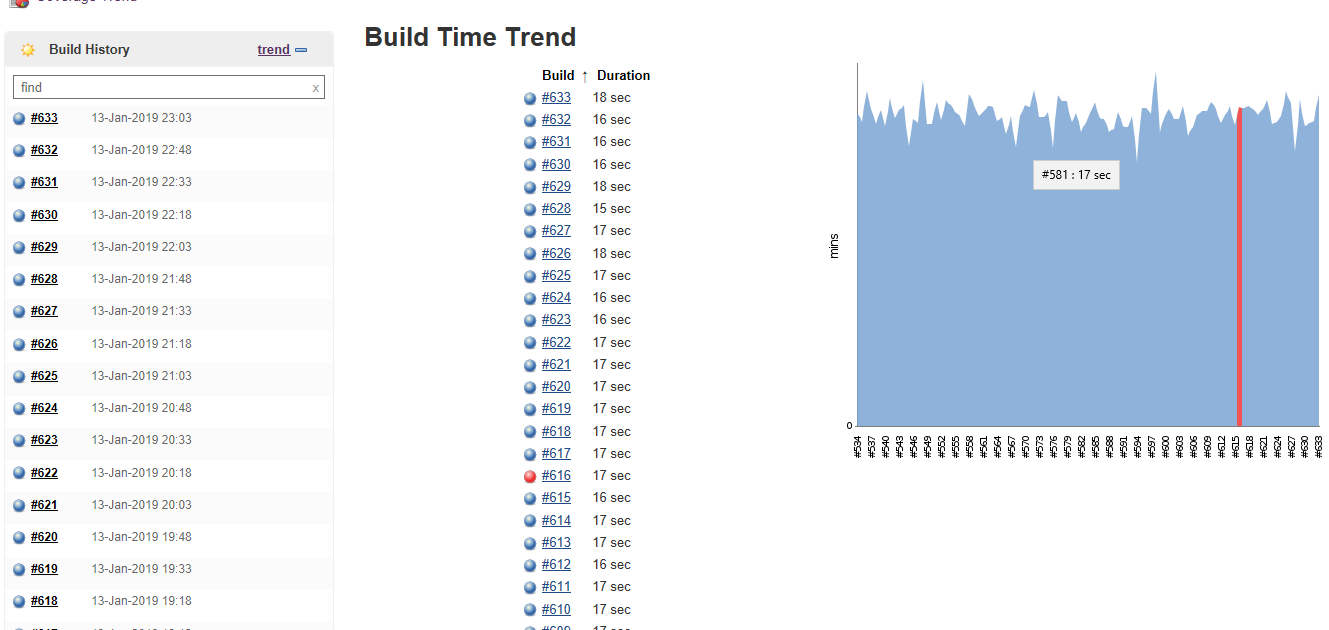
Inicialmente tivemos problemas na criação de pipelines devido a faltas de permissões, no qual decidimos criar um Freestyle project.

Definimos builds periódicas de modo a automatizar o processo, são feitas builds de 15 em 15 minutos.



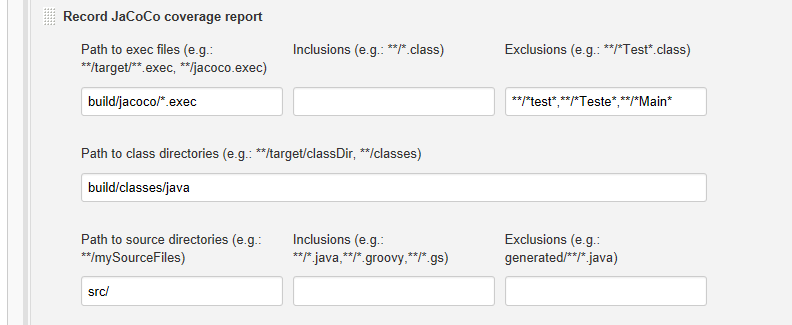
Como podemos ver no output da consola de cada build, o workspace é limpo para que ao ir buscar os ficheiros ao git não haja conflitos, é utilizado as configurações do nosso build.gradle e são feitas as tasks automáticas (test, jacocoTestReport, build). 

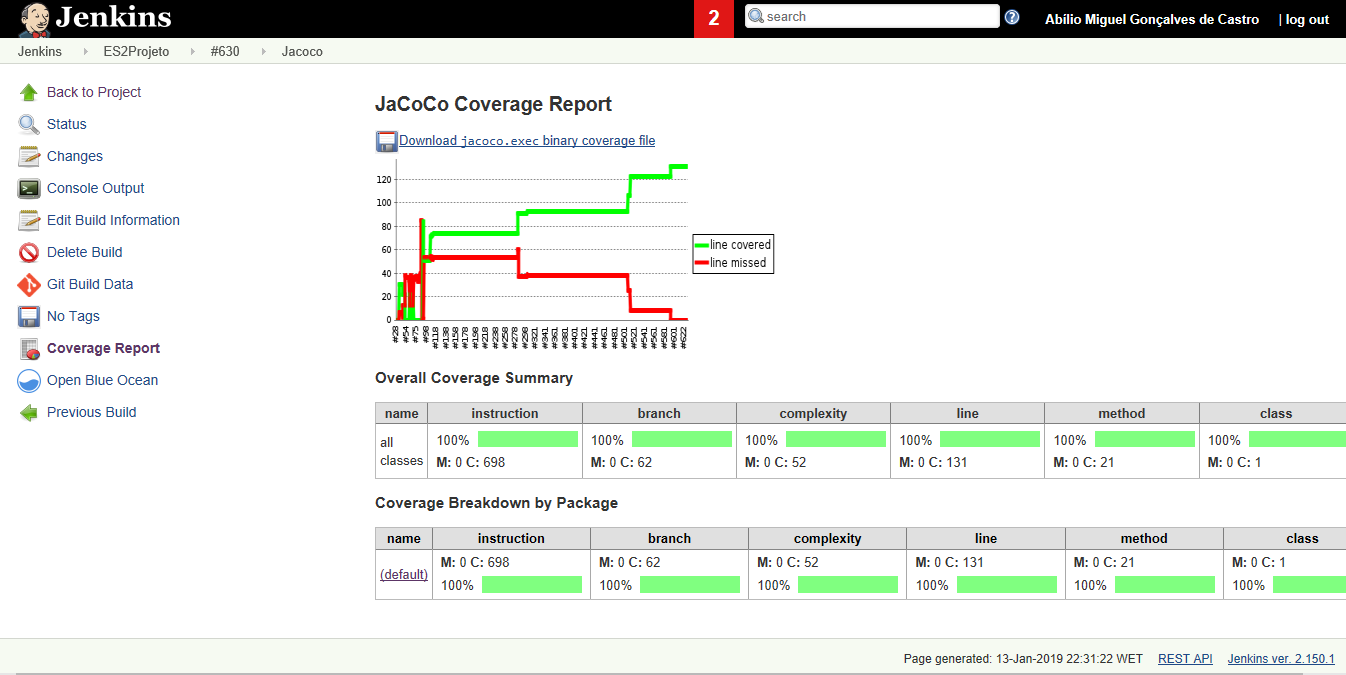
No jenkins temos acesso a trend das builds, onde podemos ver as builds que tiveram success e as que tiverem failed.



Toda a configuração foi feita com o auxílio dos powerpoints do moodle e da página de wiki do jenkins, para nos ajudar na revisão dos testes instalamos um plugin auxiliar que nos mostra um gráfico de progresso na revisão de código de todo o projeto, e também temos acesso a que métodos é que ficaram por testar.

Utilizador para uso pelos professores login : ES2 password : password123.





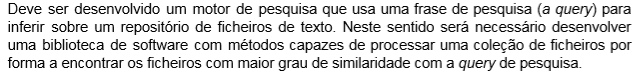
## IntelliJ:

O IntelliJ foi o IDE que utilizamos para o desenvolvimento do projeto, a sua utilização foi bastante importante pois permitiu-nos de maneira fácil criar um projeto com Gradle e também fazer commits,push e pull através do próprio IDE.

## Upsource (opcional):

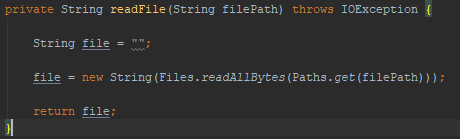
Como foi mencionado anteriormente não tivemos sucesso a instalar o Upsource na máquina virtual, e por causa disso descartamos a sua utilização.

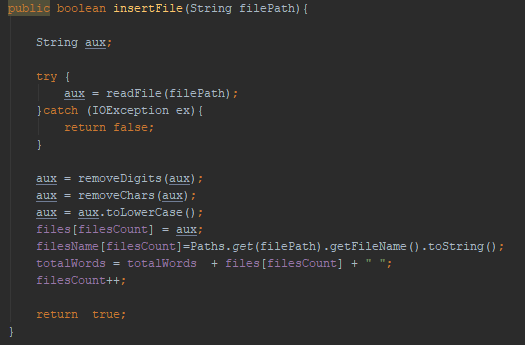
# **O problema**

****

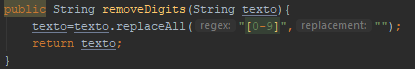
Inicialmente a interpretação do problema não foi instantânea, mas após várias discussões sobre o problema, decidimos recolher os requisitos necessários e dividir o problema em vários módulos que resolvemos com vários métodos.

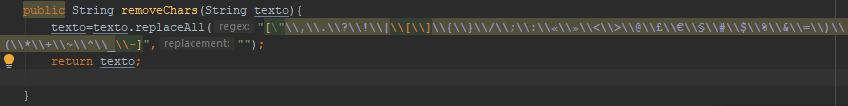
Primeiramente a leitura dos ficheiros, sempre que inserimos um ficheiro guardávamos toda a informação numa string.

****

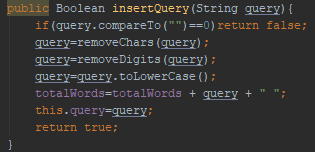
****

À string com os dados do ficheiro é removido os dígitos (removeDigits), os caracteres especiais (removeChars), passado tudo para letra minúscula para fácil comparação e armazenamos a string num array com todos as informações de todos os ficheiros e o nome do ficheiro para uso mais tarde, temos uma string onde concatenamos todas as palavras dos ficheiros, para teremos como comparação.

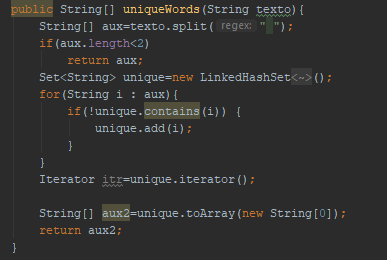


O método remove chars está a remover todos os caracteres que conseguimos encontrar no nosso teclado, apesar de noutros idiomas haver mais caracteres não encontramos uma lista de todos. 

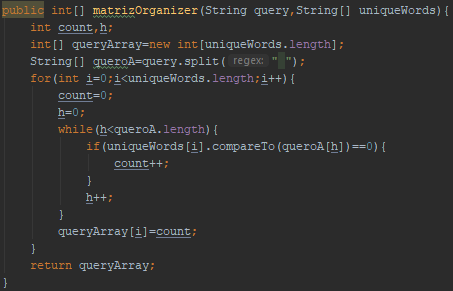
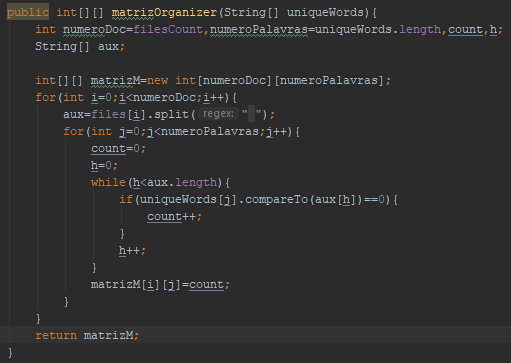
Depois de inserir todos os ficheiros inserimos a query de pesquisa que é questionada ao utilizador. O método de insertQuery faz o mesmo que o insertFile, limpa dígitos e caracteres, passa para letras minúscula e adiciona a string com todas as palavras.

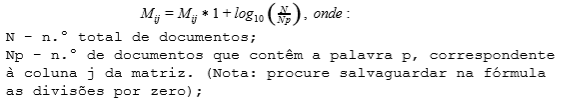
****

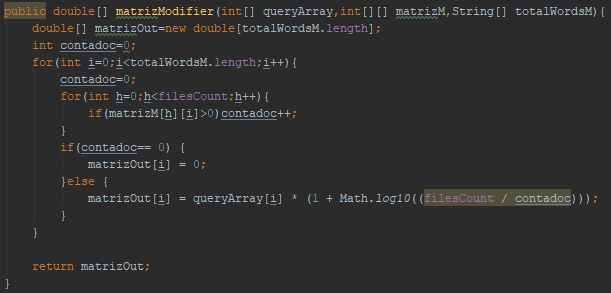
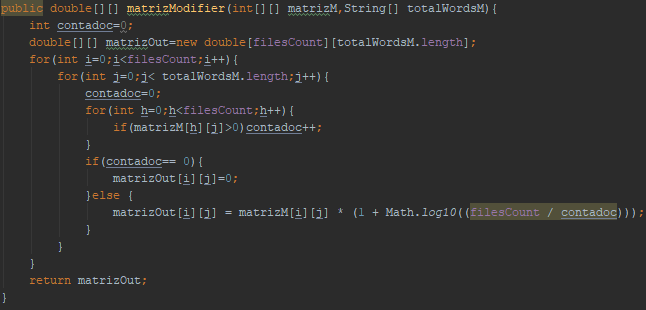
Após esta parte de inserção de dados passamos a utilização de um método auxiliar que serve para retirarmos as palavras únicas da string com todas as palavras referida acima, para termos um array de comparação com os dados dos ficheiros e da query.



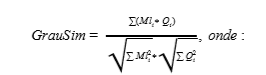
De seguida utilizamos o método matrizOrganizer para termos uma matriz e um array de ocorrências das palavras dos ficheiros e da query. A lógica por trás desta implementação é a comparação do array de palavras uniqueWords com as palavras de cada ficheiro e da query para obtermos as ocorrências.

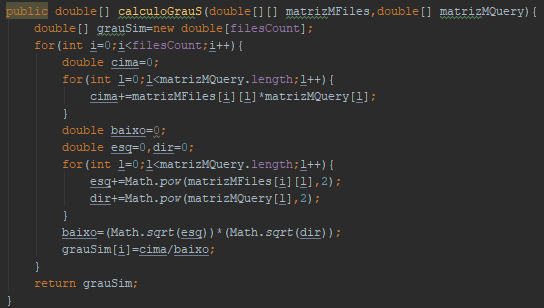


Após esta parte estar concluída passamos para o uso da primeira fórmula onde tivermos que salvaguardar as divisões por zero, que é quando é pesquisada uma palavra que não aparece em nenhum ficheiro. Dividimos em dois métodos para um ser para os ficheiros e o outro para a query. ****

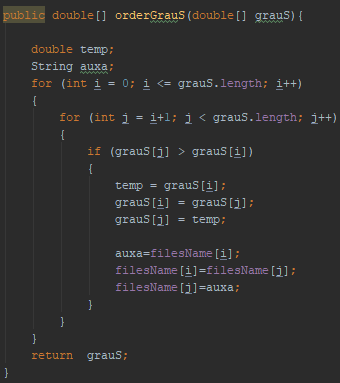
****

Posteriormente é feito o cálculo do grau de semelhança com a segunda fórmula fornecida, a implementação do método acreditamos não ter sido a mais otimizada devido a quantidade de ciclos, mas foi a que conseguimos pensar e implementar.

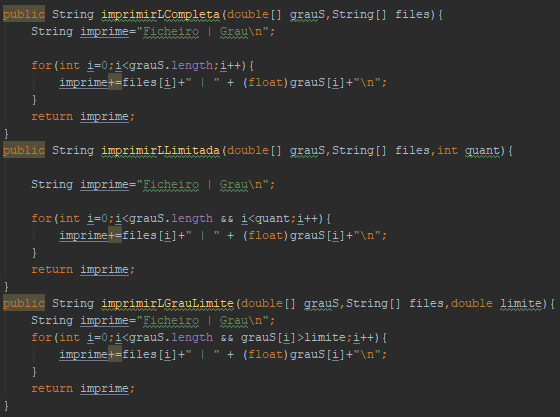
****

****

Logo em seguida é feita uma ordenação dos graus de semelhança para fácil impressão, isto é feito em conjunto com o array acima com os nomes dos ficheiros para as posições coincidirem.



E finalmente é pedido ao utilizar que tipo de impressão quer completa, limitada por ficheiros, limitada por grau semelhança.



# **Testes**

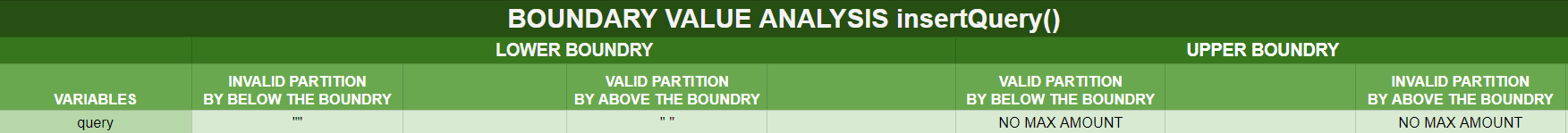
Na nossa opinião foi particularmente difícil a criação de casos de teste para o nosso problema porque estávamos a trabalhar com Strings sendo impossível definir limites superiores o que diminui muito a quantidade de casos de teste.

## Método insertQuery():

### Tabela ECP:

|  |  |  |
| --- | --- | --- |
| **EQUIVALENCE CLASS PARTITIONING insertQuery()** | | |
| **REQUIREMENTS** | **VALID CLASS** | **INVALID CLASS** |
| **Nº INPUTS** | 1 | !=1 |
| **INPUT TYPES** | query: String | query != String |
| **SPECIFIC X VALUE** | query: "anything" | query: "" |

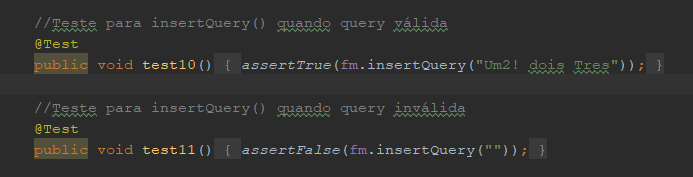
### Tabela BVA:



### Tabela Test Cases:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **TEST CASES insertQuery()** | | | | | | |
| **TEST CASE ID** | **TEST CASE** | **PRE-CONDITIONS** | **VARIABLES** | **EXPECTED RESULTS** | **ACTUAL RESULTS** | **Test Status** |
| query |
| 11 | query inválida | fm (FileManager) being declared | "" | False | False | Passed |
| 10 | query válida | "anything" | True | True | Passed |

### Testes em Java correspondentes às tabelas:

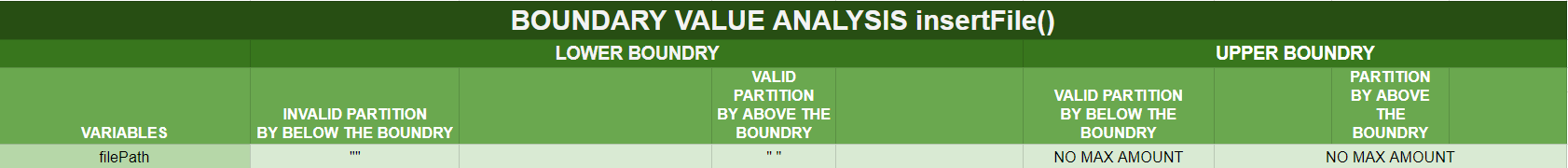


## Método insertFile():

### Tabela ECP:

|  |  |  |
| --- | --- | --- |
| **EQUIVALENCE CLASS PARTITIONING insertFile()** | | |
| **REQUIREMENTS** | **VALID CLASS** | **INVALID CLASS** |
| **Nº INPUTS** | 1 | !=1 |
| **INPUT TYPES** | filePath: String | filePath!= String |
| **SPECIFIC X VALUE** | filePath: "anything" | filePath: "" |

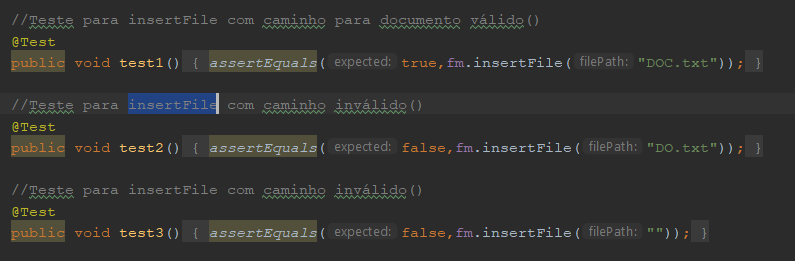
### Tabela BVA:



### Tabela Test Cases:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **TEST CASES insertFile()** | | | | | | |
| **TEST CASE ID** | **TEST CASE** | **PRE-CONDITIONS** | **VARIABLES** | **EXPECTED RESULTS** | **ACTUAL RESULTS** | **Test Status** |
| filePath |
| 2 | Invalid Path | fm (FileManager) being declared | Invalid path | False | False | Passed |
| 3 | Invalid Path | Null path | False | False | Passed |
| 1 | Valid Path | Valid path | True | True | Passed |

### Testes em Java correspondentes às tabelas:

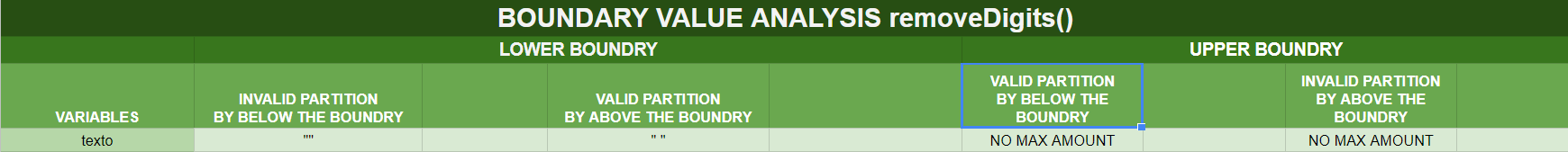


## Método removeDigits():

### Tabela ECP:

|  |  |  |
| --- | --- | --- |
| **EQUIVALENCE CLASS PARTITIONING removeDigits()** | | |
| **REQUIREMENTS** | **VALID CLASS** | **INVALID CLASS** |
| **Nº INPUTS** | 1 | !=1 |
| **INPUT TYPES** | texto: String | texto!= String |
| **SPECIFIC X VALUE** | texto: "anything" | texto: "" |

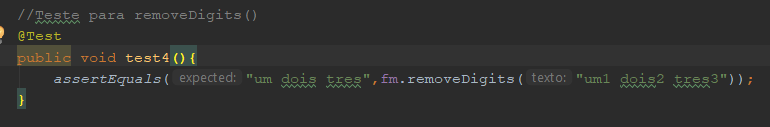
### Tabela BVA:



### Tabelas Test Cases:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **TEST CASES removeDigits()** | | | | | | |
| **TEST CASE ID** | **TEST CASE** | **PRE-CONDITIONS** | **VARIABLES** | **EXPECTED RESULTS** | **ACTUAL RESULTS** | **Test Status** |
| texto |
| 4 | Digits Being Removed | fm (FileManager) being declared && String has numbers | texto with Numbers | String With No Numbers | String With No Numbers | Passed |

Teste em Java correspondentes às tabelas:

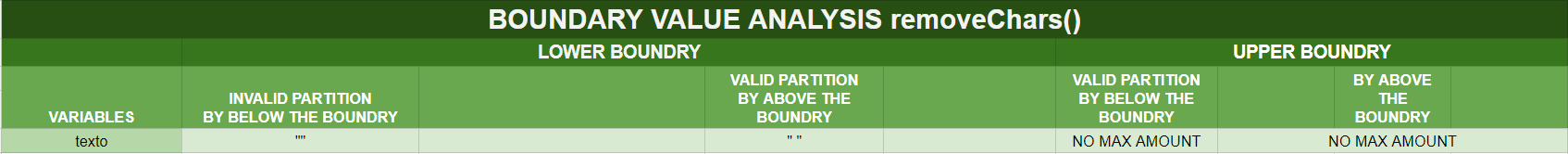


## Método removeChars():

### Tabela ECP:

|  |  |  |
| --- | --- | --- |
| **EQUIVALENCE CLASS PARTITIONING removeChars()** | | |
| **REQUIREMENTS** | **VALID CLASS** | **INVALID CLASS** |
| **Nº INPUTS** | 1 | !=1 |
| **INPUT TYPES** | texto: String | texto != String |
| **SPECIFIC X VALUE** | texto: "anything" | texto: "" |

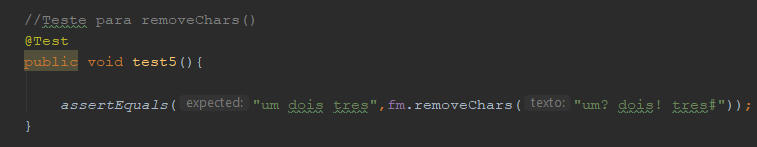
### Tabela BVA:



### Tabela Test Cases:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **TEST CASES removeChars()** | | | | | | |
| **TEST CASE ID** | **TEST CASE** | **PRE-CONDITIONS** | **VARIABLES** | **EXPECTED RESULTS** | **ACTUAL RESULTS** | **Test Status** |
| texto |
| 5 | Special Characters Being Removed | fm (FileManager) being declared && String has special characters | texto with Special Characters | String With No Special Characters | String With No Characters | Passed |

Testes em Java correspondentes à tabela:



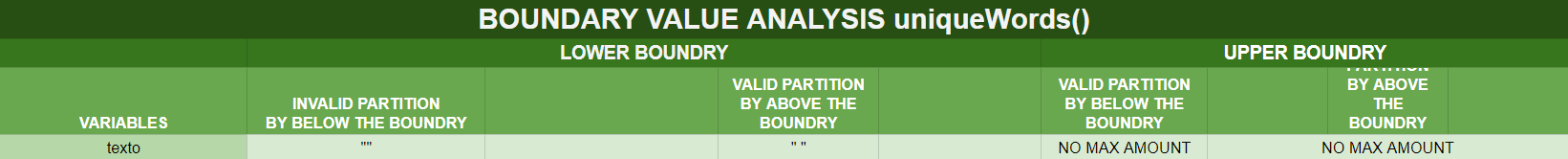
## 

## Método uniqueWords():

### Tabela ECP:

|  |  |  |
| --- | --- | --- |
| **EQUIVALENCE CLASS PARTITIONING uniqueWords()** | | |
| **REQUIREMENTS** | **VALID CLASS** | **INVALID CLASS** |
| **Nº INPUTS** | 1 | !=1 |
| **INPUT TYPES** | texto: String | texto!= String |
| **SPECIFIC X VALUE** | texto: "anything" | texto: "" |

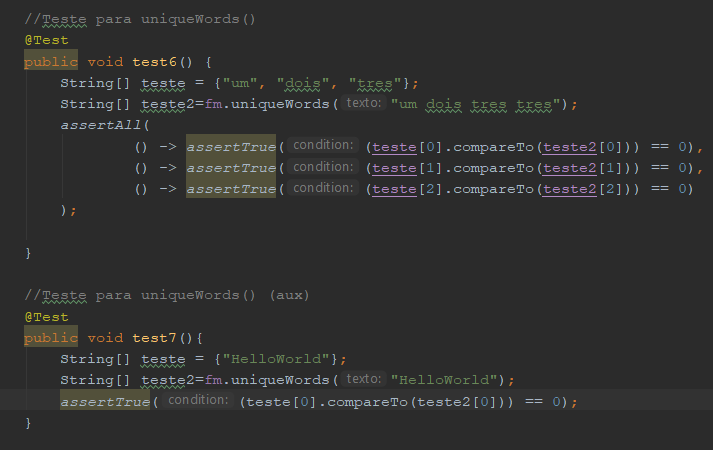
### Tabela BVA:



### Tabela Test Cases:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **TEST CASES uniqueWords()** | | | | | | |
| **TEST CASE ID** | **TEST CASE** | **PRE-CONDITIONS** | **VARIABLES** | **EXPECTED RESULTS** | **ACTUAL RESULTS** | **Test Status** |
| texto |
| 6 | Extra words Being Removed | fm (FileManager) being declared | texto with duplicated words | String[] With No duplicated words | String[] With No duplicated words | Passed |
| 7 | No extra words | fm (FileManager) being declared | texto with only 1 word | String[] With only 1 position | String[] With only 1 position | Passed |

### Testes em Java correspondentes às tabelas:

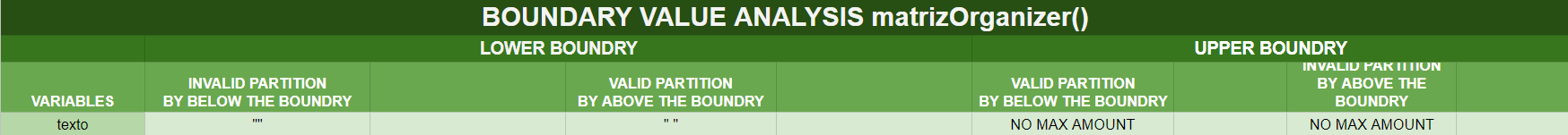


## Método matrizOrganizer():

### Tabela ECP:

|  |  |  |
| --- | --- | --- |
| **EQUIVALENCE CLASS PARTITIONING matrizOrganizer()** | | |
| **REQUIREMENTS** | **VALID CLASS** | **INVALID CLASS** |
| **Nº INPUTS** | 1 | !=1 |
| **INPUT TYPES** | uniqueWords: String[] | uniqueWords!= String[] |
| **SPECIFIC X VALUE** | uniqueWords: {'teste1'} | uniqueWords: {} |

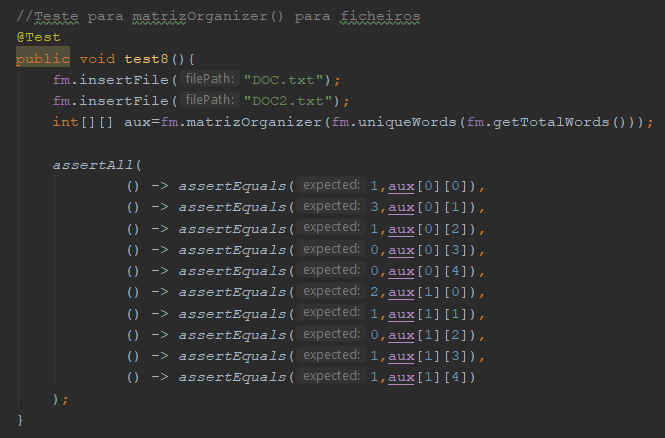
### Tabela BVA:



### Tabela Test Cases:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **TEST CASES matrizOrganizer()** | | | | | | |
| **TEST CASE ID** | **TEST CASE** | **PRE-CONDITIONS** | **VARIABLES** | **EXPECTED RESULTS** | **ACTUAL RESULTS** | **Test Status** |
| uniqueWords |
| 8 | Válid uniqueWords | fm (FileManager) declared, files and query imported and cleaned | String[] with no duplicated words | int[][] with the ocurrencies of each word in each file | int[][] with the ocurrencies of each word in each file | Passed |

### Teste em Java correspondentes às tabelas:

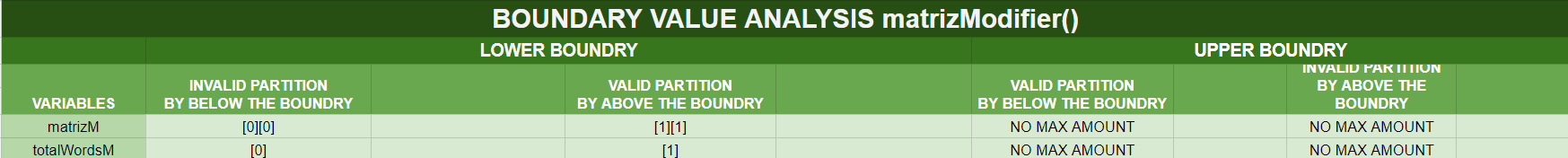


## Método matrizModifier():

### Tabela ECP:

|  |  |  |
| --- | --- | --- |
| **EQUIVALENCE CLASS PARTITIONING matrizModifier()** | | |
| **REQUIREMENTS** | **VALID CLASS** | **INVALID CLASS** |
| **Nº INPUTS** | 2 | !=2 |
| **INPUT TYPES** | matrizM: int[][], totalWordsM:String[] | matrizM != int[][], totalWordsM!=String[] |
| **SPECIFIC X VALUE** | matriz: [1][1], totalWordsM: [1] | matriz: [0][0], totalWordsM[0] |

### Tabela BVA:



### Tabela Test Case:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **TEST CASES matrizModifier()** | | | | | | |
| **TEST CASE ID** | **TEST CASE** | **PRE-CONDITIONS** | **VARIABLES** | **EXPECTED RESULTS** | **ACTUAL RESULTS** | **Test Status** |
| matrizM, totalWordsM |
| 13 | valid variables | fm (FileManager) declared, files and query imported and cleaned | int[][] with the ocurrencies of each word in each file, String[] with each unique word | double[][] with the result from the first formula | double[][] with the result from the first formula | Passed |

### Testes em Java correspondentes às tabelas:

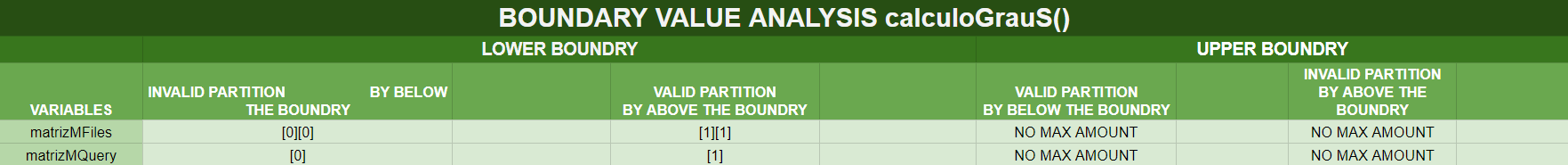


## Método calculoGrauS():

### Tabela ECP:

|  |  |  |
| --- | --- | --- |
| **EQUIVALENCE CLASS PARTITIONING calculoGrauS()** | | |
| **REQUIREMENTS** | **VALID CLASS** | **INVALID CLASS** |
| **Nº INPUTS** | 2 | !=2 |
| **INPUT TYPES** | matrizMFiles: double[][], matrizMQuery:double[] | matrizMFiles!= double[][], matrizMQuery!=double[] |
| **SPECIFIC X VALUE** | matrizMFiles: [1][1], matrizMQuery: [1] | matrizMFiles: [0][0], matrizMQuery[0] |

### Tabela BVA:



### Tabela Test Cases:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **TEST CASES calculoGrauS()** | | | | | | |
| **TEST CASE ID** | **TEST CASE** | **PRE-CONDITIONS** | **VARIABLES** | **EXPECTED RESULTS** | **ACTUAL RESULTS** | **Test Status** |
| matrizMFiles, matrizMQuery |
| 1 | valid variables | fm (FileManager) declared, files and query imported and cleaned, matrizMFiles and matrizMQuery already passed through the matrizModifier | double[][] with the result from the first formula for files, double[] with the result from the first formula for query | double[] with the result from the secund formula | double[] with the result from the secund formula | Passed |

### Testes em Java correspondentes às tabelas:

