

Relatório ESII Project

Docentes:

Cristóvão Sousa
Fábio Silva
Bruno Silva

Realizado por:

Abílio Castro - 8170054
Ricardo Cardoso - 8170278
Vitor Santos - 8170312

Introdução	4
Configuração de Ferramentas	5
Youtrack:	5
Swimlanes:	5
Colunas:	6
Backlog:	8
Sprints:	8
Sprint #1	9
Sprint #2	9
Sprint Entrega Final	9
Git:	9
Máquina Virtual:	10
Jenkins:	11
IntelliJ:	13
Upsource (opcional):	13
O problema	14
Testes	20
Método insertQuery():	20
Tabela ECP:	20
Tabela BVA:	20
Tabela Test Cases:	20
Testes em Java correspondentes às tabelas:	21
Método insertFile():	21
Tabela ECP:	21
Tabela BVA:	21
Tabela Test Cases:	21
Testes em Java correspondentes às tabelas:	22
Método removeDigits():	22
Tabela ECP:	22
Tabela BVA:	22
Tabelas Test Cases:	22
Método removeChars():	23
Tabela ECP:	23
Tabela BVA:	24
Tabela Test Cases:	24
Método uniqueWords():	24
Tabela ECP:	24

Tabela BVA:	25
Tabela Test Cases:	25
Testes em Java correspondentes às tabelas:	25
Método matrizOrganizer():	26
Tabela ECP:	26
Tabela BVA:	26
Tabela Test Cases:	27
Teste em Java correspondentes às tabelas:	27
Método matrizModifier():	28
Tabela ECP:	28
Tabela BVA:	28
Tabela Test Case:	28
Testes em Java correspondentes às tabelas:	29
Método calculoGrauS():	29
Tabela ECP:	29
Tabela BVA:	30
Tabela Test Cases:	30
Testes em Java correspondentes às tabelas:	31
Conclusão	32

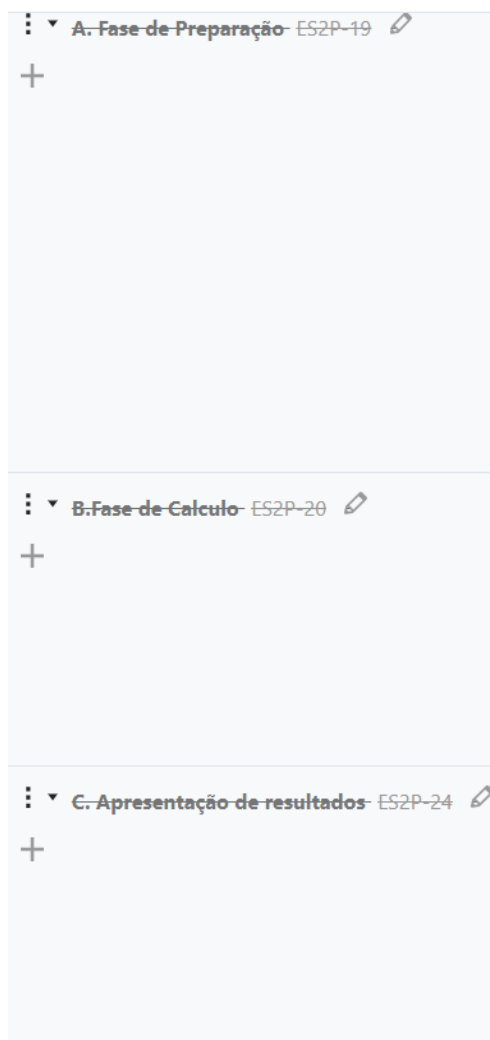
1. Introdução

Este documento tem como objetivo a especificação de conteúdo técnico do software de pesquisa de ficheiros do grupo MichaelSoft Inc., onde vai constar o propósito deste software, o seu domínio, os vários requisitos necessários para o desenvolvimento deste software, os testes efetuados, os issues, entre outros...

2. 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:

< In Progress

ES2P-27 ● Normal Task

Método matrizPesquisa

 Not estimated

+


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

< Code Review

+

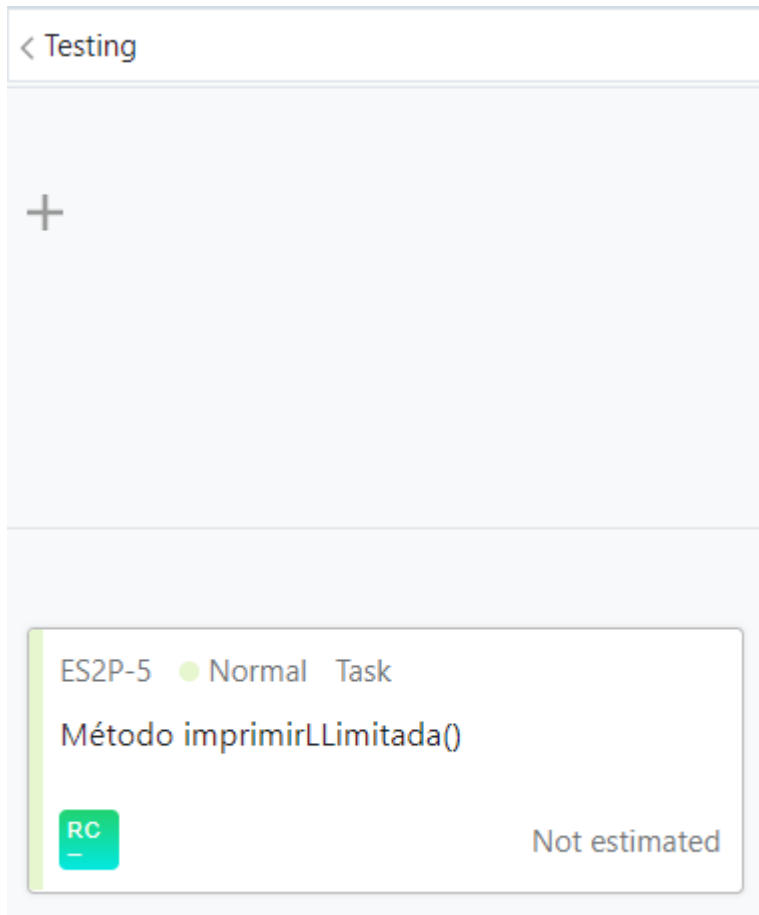
ES2P-28 ● Show-stopper Task

Método calculoGrauS

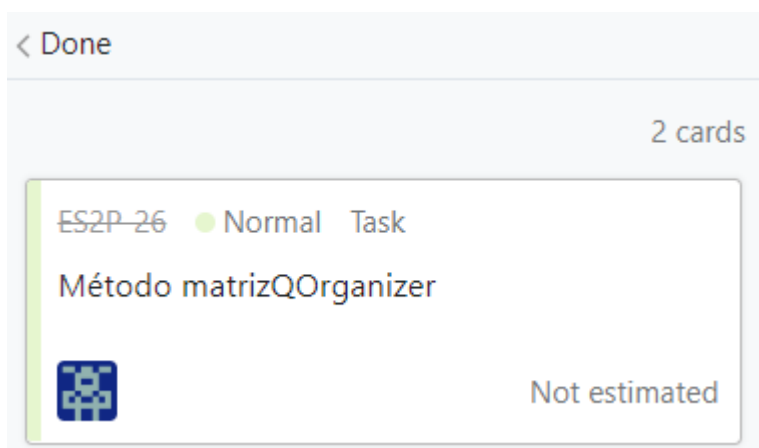
 Not estimated

+

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.

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.

Update Main.java

abiliomg committed 388b483 1 changed file

```
src/main/java/Main.java
1 6 public class Main {
2 7     public static void main(String[] args) {
3 8         public static void main(String[] args) throws IOException {
4 9             System.out.println("Insira a query");
5 10             BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
6 11             String query = br.readLine();
7 12
8 13             FileManager teste=new FileManager(2);
9 14             teste.insertFile("um.txt");
10 15             teste.insertFile("dois.txt");
11 16             String query="um dois tres";
12 17             teste.insertQuery(query);
13 18             query=teste.getQuery();
14 19
15 20 @@ -32,21 +40,31 @@ public class Main {
16 21             System.out.println(i);
17 22             }*/
18 23             grauSim=teste.orderGrau5(grauSim);
19 24             int opcao=2,quantFMostrar=3;
20 25             int opcao=0,quantFMostrar=3;
21 26             double limite=0.2;
22 27
23 28             switch (opcao){
24 29                 case 1:
25 30                     System.out.println(teste.imprimirCompleta(grauSim,teste.getFilesName()));
26 31                     break;
27 32                 case 2:
28 33                     //Escreva o nome do arquivo e o limite entre 0 e 1
29 34
30 35
31 36
32 37
33 38
34 39
35 40
36 41
37 42
38 43
39 44
40 45
41 46
42 47
43 48
44 49
45 50
46 51
47 52
48 53
49 54
50 55
51 56
52 57
53 58
54 59
55 60
56 61
57 62
58 63
59 64
60 65
61 66
62 67
63 68
64 69
65 70
66 71
67 72
68 73
69 74
70 75
71 76
72 77
73 78
74 79
75 80
76 81
77 82
78 83
79 84
80 85
81 86
82 87
83 88
84 89
85 90
86 91
87 92
88 93
89 94
90 95
91 96
92 97
93 98
94 99
95 100
96 101
97 102
98 103
99 104
100 105
101 106
102 107
103 108
104 109
105 110
106 111
107 112
108 113
109 114
110 115
111 116
112 117
113 118
114 119
115 120
116 121
117 122
118 123
119 124
120 125
121 126
122 127
123 128
124 129
125 130
126 131
127 132
128 133
129 134
130 135
131 136
132 137
133 138
134 139
135 140
136 141
137 142
138 143
139 144
140 145
141 146
142 147
143 148
144 149
145 150
146 151
147 152
148 153
149 154
150 155
151 156
152 157
153 158
154 159
155 160
156 161
157 162
158 163
159 164
160 165
161 166
162 167
163 168
164 169
165 170
166 171
167 172
168 173
169 174
170 175
171 176
172 177
173 178
174 179
175 180
176 181
177 182
178 183
179 184
180 185
181 186
182 187
183 188
184 189
185 190
186 191
187 192
188 193
189 194
190 195
191 196
192 197
193 198
194 199
195 200
196 201
197 202
198 203
199 204
200 205
201 206
202 207
203 208
204 209
205 210
206 211
207 212
208 213
209 214
210 215
211 216
212 217
213 218
214 219
215 220
216 221
217 222
218 223
219 224
220 225
221 226
222 227
223 228
224 229
225 230
226 231
227 232
228 233
229 234
230 235
231 236
232 237
233 238
234 239
235 240
236 241
237 242
238 243
239 244
240 245
241 246
242 247
243 248
244 249
245 250
246 251
247 252
248 253
249 254
250 255
251 256
252 257
253 258
254 259
255 260
256 261
257 262
258 263
259 264
260 265
261 266
262 267
263 268
264 269
265 270
266 271
267 272
268 273
269 274
270 275
271 276
272 277
273 278
274 279
275 280
276 281
277 282
278 283
279 284
280 285
281 286
282 287
283 288
284 289
285 290
286 291
287 292
288 293
289 294
290 295
291 296
292 297
293 298
294 299
295 300
300 301
301 302
302 303
303 304
304 305
305 306
306 307
307 308
308 309
309 310
310 311
311 312
312 313
313 314
314 315
315 316
316 317
317 318
318 319
319 320
320 321
321 322
322 323
323 324
324 325
325 326
326 327
327 328
328 329
329 330
330 331
331 332
332 333
333 334
334 335
335 336
336 337
337 338
338 339
339 340
340 341
341 342
342 343
343 344
344 345
345 346
346 347
347 348
348 349
349 350
350 351
351 352
352 353
353 354
354 355
355 356
356 357
357 358
358 359
359 360
360 361
361 362
362 363
363 364
364 365
365 366
366 367
367 368
368 369
369 370
370 371
371 372
372 373
373 374
374 375
375 376
376 377
377 378
378 379
379 380
380 381
381 382
382 383
383 384
384 385
385 386
386 387
387 388
388 389
389 390
390 391
391 392
392 393
393 394
394 395
395 396
396 397
397 398
398 399
399 400
400 401
401 402
402 403
403 404
404 405
405 406
406 407
407 408
408 409
409 410
410 411
411 412
412 413
413 414
414 415
415 416
416 417
417 418
418 419
419 420
420 421
421 422
422 423
423 424
424 425
425 426
426 427
427 428
428 429
429 430
430 431
431 432
432 433
433 434
434 435
435 436
436 437
437 438
438 439
439 440
440 441
441 442
442 443
443 444
444 445
445 446
446 447
447 448
448 449
449 450
450 451
451 452
452 453
453 454
454 455
455 456
456 457
457 458
458 459
459 460
460 461
461 462
462 463
463 464
464 465
465 466
466 467
467 468
468 469
469 470
470 471
471 472
472 473
473 474
474 475
475 476
476 477
477 478
478 479
479 480
480 481
481 482
482 483
483 484
484 485
485 486
486 487
487 488
488 489
489 490
490 491
491 492
492 493
493 494
494 495
495 496
496 497
497 498
498 499
499 500
500 501
501 502
502 503
503 504
504 505
505 506
506 507
507 508
508 509
509 510
510 511
511 512
512 513
513 514
514 515
515 516
516 517
517 518
518 519
519 520
520 521
521 522
522 523
523 524
524 525
525 526
526 527
527 528
528 529
529 530
530 531
531 532
532 533
533 534
534 535
535 536
536 537
537 538
538 539
539 540
540 541
541 542
542 543
543 544
544 545
545 546
546 547
547 548
548 549
549 550
550 551
551 552
552 553
553 554
554 555
555 556
556 557
557 558
558 559
559 560
560 561
561 562
562 563
563 564
564 565
565 566
566 567
567 568
568 569
569 570
570 571
571 572
572 573
573 574
574 575
575 576
576 577
577 578
578 579
579 580
580 581
581 582
582 583
583 584
584 585
585 586
586 587
587 588
588 589
589 590
590 591
591 592
592 593
593 594
594 595
595 596
596 597
597 598
598 599
599 600
600 601
601 602
602 603
603 604
604 605
605 606
606 607
607 608
608 609
609 610
610 611
611 612
612 613
613 614
614 615
615 616
616 617
617 618
618 619
619 620
620 621
621 622
622 623
623 624
624 625
625 626
626 627
627 628
628 629
629 630
630 631
631 632
632 633
633 634
634 635
635 636
636 637
637 638
638 639
639 640
640 641
641 642
642 643
643 644
644 645
645 646
646 647
647 648
648 649
649 650
650 651
651 652
652 653
653 654
654 655
655 656
656 657
657 658
658 659
659 660
660 661
661 662
662 663
663 664
664 665
665 666
666 667
667 668
668 669
669 670
670 671
671 672
672 673
673 674
674 675
675 676
676 677
677 678
678 679
679 680
680 681
681 682
682 683
683 684
684 685
685 686
686 687
687 688
688 689
689 690
690 691
691 692
692 693
693 694
694 695
695 696
696 697
697 698
698 699
699 700
700 701
701 702
702 703
703 704
704 705
705 706
706 707
707 708
708 709
709 710
710 711
711 712
712 713
713 714
714 715
715 716
716 717
717 718
718 719
719 720
720 721
721 722
722 723
723 724
724 725
725 726
726 727
727 728
728 729
729 730
730 731
731 732
732 733
733 734
734 735
735 736
736 737
737 738
738 739
739 740
740 741
741 742
742 743
743 744
744 745
745 746
746 747
747 748
748 749
749 750
750 751
751 752
752 753
753 754
754 755
755 756
756 757
757 758
758 759
759 760
760 761
761 762
762 763
763 764
764 765
765 766
766 767
767 768
768 769
769 770
770 771
771 772
772 773
773 774
774 775
775 776
776 777
777 778
778 779
779 780
780 781
781 782
782 783
783 784
784 785
785 786
786 787
787 788
788 789
789 790
790 791
791 792
792 793
793 794
794 795
795 796
796 797
797 798
798 799
799 800
800 801
801 802
802 803
803 804
804 805
805 806
806 807
807 808
808 809
809 810
810 811
811 812
812 813
813 814
814 815
815 816
816 817
817 818
818 819
819 820
820 821
821 822
822 823
823 824
824 825
825 826
826 827
827 828
828 829
829 830
830 831
831 832
832 833
833 834
834 835
835 836
836 837
837 838
838 839
839 840
840 841
841 842
842 843
843 844
844 845
845 846
846 847
847 848
848 849
849 850
850 851
851 852
852 853
853 854
854 855
855 856
856 857
857 858
858 859
859 860
860 861
861 862
862 863
863 864
864 865
865 866
866 867
867 868
868 869
869 870
870 871
871 872
872 873
873 874
874 875
875 876
876 877
877 878
878 879
879 880
880 881
881 882
882 883
883 884
884 885
885 886
886 887
887 888
888 889
889 890
890 891
891 892
892 893
893 894
894 895
895 896
896 897
897 898
898 899
899 900
900 901
901 902
902 903
903 904
904 905
905 906
906 907
907 908
908 909
909 910
910 911
911 912
912 913
913 914
914 915
915 916
916 917
917 918
918 919
919 920
920 921
921 922
922 923
923 924
924 925
925 926
926 927
927 928
928 929
929 930
930 931
931 932
932 933
933 934
934 935
935 936
936 937
937 938
938 939
939 940
940 941
941 942
942 943
943 944
944 945
945 946
946 947
947 948
948 949
949 950
950 951
951 952
952 953
953 954
954 955
955 956
956 957
957 958
958 959
959 960
960 961
961 962
962 963
963 964
964 965
965 966
966 967
967 968
968 969
969 970
970 971
971 972
972 973
973 974
974 975
975 976
976 977
977 978
978 979
979 980
980 981
981 982
982 983
983 984
984 985
985 986
986 987
987 988
988 989
989 990
990 991
991 992
992 993
993 994
994 995
995 996
996 997
997 998
998 999
999 1000
1000 1001
1001 1002
1002 1003
1003 1004
1004 1005
1005 1006
1006 1007
1007 1008
1008 1009
1009 1010
1010 1011
1011 1012
1012 1013
1013 1014
1014 1015
1015 1016
1016 1017
1017 1018
1018 1019
1019 1020
1020 1021
1021 1022
1022 1023
1023 1024
1024 1025
1025 1026
1026 1027
1027 1028
1028 1029
1029 1030
1030 1031
1031 1032
1032 1033
1033 1034
1034 1035
1035 1036
1036 1037
1037 1038
1038 1039
1039 1040
1040 1041
1041 1042
1042 1043
1043 1044
1044 1045
1045 1046
1046 1047
1047 1048
1048 1049
1049 1050
1050 1051
1051 1052
1052 1053
1053 1054
1054 1055
1055 1056
1056 1057
1057 1058
1058 1059
1059 1060
1060 1061
1061 1062
1062 1063
1063 1064
1064 1065
1065 1066
1066 1067
1067 1068
1068 1069
1069 1070
1070 1071
1071 1072
1072 1073
1073 1074
1074 1075
1075 1076
1076 1077
1077 1078
1078 1079
1079 1080
1080 1081
1081 1082
1082 1083
1083 1084
1084 1085
1085 1086
1086 1087
1087 1088
1088 1089
1089 1090
1090 1091
1091 1092
1092 1093
1093 1094
1094 1095
1095 1096
1096 1097
1097 1098
1098 1099
1099 1100
1100 1101
1101 1102
1102 1103
1103 1104
1104 1105
1105 1106
1106 1107
1107 1108
1108 1109
1109 1110
1110 1111
1111 1112
1112 1113
1113 1114
1114 1115
1115 1116
1116 1117
1117 1118
1118 1119
1119 1120
1120 1121
1121 1122
1122 1123
1123 1124
1124 1125
1125 1126
1126 1127
1127 1128
1128 1129
1129 1130
1130 1131
1131 1132
1132 1133
1133 1134
1134 1135
1135 1136
1136 1137
1137 1138
1138 1139
1139 1140
1140 1141
1141 1142
1142 1143
1143 1144
1144 1145
1145 1146
1146 1147
1147 1148
1148 1149
1149 1150
1150 1151
1151 1152
1152 1153
1153 1154
1154 1155
1155 1156
1156 1157
1157 1158
1158 1159
1159 1160
1160 1161
1161 1162
1162 1163
1163 1164
1164 1165
1165 1166
1166 1167
1167 1168
1168 1169
1169 1170
1170 1171
1171 1172
1172 1173
1173 1174
1174 1175
1175 1176
1176 1177
1177 1178
1178 1179
1179 1180
1180 1181
1181 1182
1182 1183
1183 1184
1184 1185
1185 1186
1186 1187
1187 1188
1188 1189
1189 1190
1190 1191
1191 1192
1192 1193
1193 1194
1194 1195
1195 1196
1196 1197
1197 1198
1198 1199
1199 1200
1200 1201
1201 1202
1202 1203
1203 1204
1204 1205
1205 1206
1206 1207
1207 1208
1208 1209
1209 1210
1210 1211
1211 1212
1212 1213
1213 1214
1214 1215
1215 1216
1216 1217
1217 1218
1218 1219
1219 1220
1220 1221
1221 1222
1222 1223
1223 1224
1224 1225
1225 1226
1226 1227
1227 1228
1228 1229
1229 1230
1230 1231
1231 1232
1232 1233
1233 1234
1234 1235
1235 1236
1236 1237
1237 1238
1238 1239
1239 1240
1240 1241
1241 1242
1242 1243
1243 1244
1244 1245
1245 1246
1246 1247
1247 1248
1248 1249
1249 1250
1250 1251
1251 1252
1252 1253
1253 1254
1254 1255
1255 1256
1256 1257
1257 1258
1258 1259
1259 1260
1260 1261
1261 1262
1262 1263
1263 1264
1264 1265
1265 1266
1266 1267
1267 1268
1268 1269
1269 1270
1270 1271
1271 1272
1272 1273
1273 1274
1274 1275
1275 1276
1276 1277
1277 1278
1278 1279
1279 1280
1280 1281
1281 1282
1282 1283
1283 1284
1284 1285
1285 1286
1286 1287
1287 1288
1288 1289
1289 1290
1290 1291
1291 1292
1292 1293
1293 1294
1294 1295
1295 1296
1296 1297
1297 1298
1298 1299
1299 1300
1300 1301
1301 1302
1302 1303
1303 1304
1304 1305
1305 1306
1306 1307
1307 1308
1308 1309
1309 1310
1310 1311
1311 1312
1312 1313
1313 1314
1314 1315
1315 1316
1316 1317
1317 1318
1318 1319
1319 1320
1320 1321
1321 1322
1322 1323
1323 1324
1324 1325
1325 1326
1326 1327
1327 1328
1328 1329
1329 1330
1330 1331
1331 1332
1332 1333
1333 1334
1334 1335
1335 1336
1336 1337
1337 1338
1338 1339
1339 1340
1340 1341
1341 1342
1342 1343
1343 1344
1344 1345
1345 1346
1346 1347
1347 1348
1348 1349
1349 1350
1350 1351
1351 1352
1352 1353
1353 1354
1354 1355
1355 1356
1356 1357
1357 1358
1358 1359
1359 1360
1360 1361
1361 1362
1362 1363
1363 1364
1364 1365
1365 1366
1366 1367
1367 1368
1368 1369
1369 1370
1370 1371
1371 1372
1372 1373
1373 1374
1374 1375
1375 1376
1376 1377
1377 1378
1378 1379
1379 1380
1380 1381
1381 1382
1382 1383
1383 1384
1384 1385
1385 1386
1386 1387
1387 1388
1388 1389
1389 1390
1390 1391
1391 1392
1392 1393
1393 1394
1394 1395
1395 1396
1396 1397
1397 1398
1398 1399
1399 1400
1400 1401
1401 1402
1402 1403
1403 1404
1404 1405
1405 1406
1406 1407
1407 1408
1408 1409
1409 1410
1410 1411
1411 1412
1412 1413
1413 1414
1414 1415
1415 1416
1416 1417
1417 1418
1418 1419
1419 1420
1420 1421
1421 1422
1422 1423
1423 1424
1424 1425
1425 1426
1426 1427
1427 1428
1428 14
```

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.

The screenshot shows the 'Build Triggers' section of a Jenkins job configuration. It includes the following elements:

- Build Triggers** header.
- Three checkboxes: 'Trigger builds remotely (e.g., from scripts)', 'Build after other projects are built', and 'Build periodically' (which is checked).
- A 'Schedule' field containing the text 'H/15 * * * *'.
- A status message: 'Would last have run at Sunday, January 13, 2019 11:03:15 PM WET; would next run at Sunday, January 13, 2019 11:18:15 PM WET.'
- Three more checkboxes: 'GitHub Pull Request Builder', 'GitHub hook trigger for GITScm polling', and 'Poll SCM'.
- Each checkbox has a corresponding help icon (a question mark in a circle) to its right.

Below the 'Build Triggers' section is the 'Build Environment' section, which contains:

- A checked checkbox: 'Delete workspace before build starts'.
- An 'Advanced...' button at the bottom.

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).

Changes

Console Output

View as plain text

Edit Build Information

Delete Build

Git Build Data

No Tags

Coverage Report

Open Blue Ocean

Previous Build

Executed Gradle Tasks

- Task :compileJava
- Task :processResources
- Task :classes
- Task :jar
- Task :assemble
- Task :compileTestJava
- Task :processTestResources
- Task :testClasses
- Task :test
- Task :jacocoTestReport
- Task :check
- Task :build

Started by timer

Building in workspace /var/lib/jenkins/workspace/ES2Projeto

[WS-CLEANUP] Deleting project workspace...

[WS-CLEANUP] Deferred wipeout is used...

[WS-CLEANUP] Done

Cloning the remote Git repository

Cloning repository <https://github.com/Dumbrica/ESIIProject>

> git init /var/lib/jenkins/workspace/ES2Projeto # timeout=10

Fetching upstream changes from <https://github.com/Dumbrica/ESIIProject>

> git --version # timeout=10

using GIT_ASKPASS to set credentials

> git fetch --tags --progress <https://github.com/Dumbrica/ESIIProject> +refs/heads/*:refs/remotes/origin/* # timeout=10

> git config remote.origin.url <https://github.com/Dumbrica/ESIIProject> # timeout=10

> git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10

> git config remote.origin.url <https://github.com/Dumbrica/ESIIProject> # timeout=10

Fetching upstream changes from <https://github.com/Dumbrica/ESIIProject>

using GIT_ASKPASS to set credentials

> git fetch --tags --progress <https://github.com/Dumbrica/ESIIProject> +refs/heads/*:refs/remotes/origin/* # timeout=10

> git rev-parse refs/remotes/origin/master^{commit} # timeout=10

> git rev-parse refs/remotes/origin/master^{commit} # timeout=10

Checking out Revision b25ff7fc9236f545635c39296557090e4ed80025 (refs/remotes/origin/master)

> git config core.sparsecheckout # timeout=10

> git checkout -f b25ff7fc9236f545635c39296557090e4ed80025

Commit message: "Merge branch 'master' of <https://github.com/Dumbrica/ESIIProject>"

> git rev-list --no-walk b25ff7fc9236f545635c39296557090e4ed80025 # timeout=10

[Gradle] - Launching build.

[ES2Projeto] \$ /var/lib/jenkins/workspace/ES2Projeto/gradlew build

Starting a Gradle Daemon (subsequent builds will be faster)

> Task :compileJava

> Task :processResources NO-SOURCE

> Task :classes

> Task :jar

> Task :assemble

> Task :compileTestJava

> Task :processTestResources NO-SOURCE

> Task :testClasses

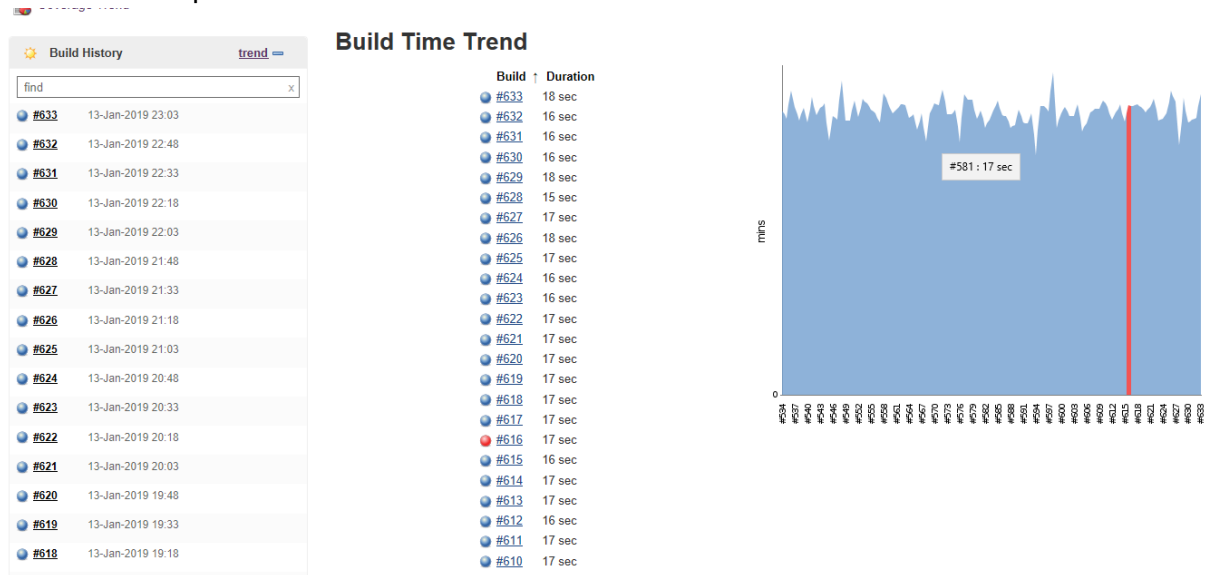
> Task :test

> Task :jacocoTestReport

> Task :check

> Task :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.

Record JaCoCo coverage report

Path to exec files (e.g.:
/target//*.exec, **/jacoco.exec)

build/jacoco/*.exec

Inclusions (e.g.: **/*.class)

Exclusions (e.g.: **/*Test*.class)

**/*test*, **/*Teste*, **/*Main*

Path to class directories (e.g.: **/target/classDir, **/classes)

build/classes/java

Path to source directories (e.g.:
**/mySourceFiles)

src/

Inclusions (e.g.:
**/*.java, **/*.groovy, **/*.gs)

Exclusions (e.g.:
generated/**/*.java)

Jenkins

2

search

Abilio Miguel Gonçalves de Castro | log out

Jenkins

ES2Projeto

#630

Jacoco

Back to Project

Status

Changes

Console Output

Edit Build Information

Delete Build

Git Build Data

No Tags

Coverage Report

Open Blue Ocean

Previous Build

JaCoCo Coverage Report

Download jacoco.exec binary coverage file

Overall Coverage Summary

name	instruction	branch	complexity	line	method	class
all	100%	100%	100%	100%	100%	100%
classes	M: 0 C: 698	M: 0 C: 62	M: 0 C: 52	M: 0 C: 131	M: 0 C: 21	M: 0 C: 1

Coverage Breakdown by Package

name	instruction	branch	complexity	line	method	class
(default)	M: 0 C: 698	M: 0 C: 62	M: 0 C: 52	M: 0 C: 131	M: 0 C: 21	M: 0 C: 1
	100%	100%	100%	100%	100%	100%

Page generated: 13-Jan-2019 22:31:22 WET [REST API](#) [Jenkins ver. 2.150.1](#)

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.

13

3.O problema

Deve ser desenvolvido um motor de pesquisa que usa uma frase de pesquisa (a *query*) para inferir sobre um repositório de ficheiros de texto. Neste sentido será necessário desenvolver uma biblioteca de software com métodos capazes de processar uma coleção de ficheiros por forma a encontrar os ficheiros com maior grau de similaridade com a *query* de pesquisa.

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.

```
private String readFile(String filePath) throws IOException {  
  
    String file = "";  
  
    file = new String(Files.readAllBytes(Paths.get(filePath)));  
  
    return file;  
}  
  
public boolean insertFile(String filePath){  
  
    String aux;  
  
    try {  
        aux = readFile(filePath);  
    } catch (IOException ex){  
        return false;  
    }  
  
    aux = removeDigits(aux);  
    aux = removeChars(aux);  
    aux = aux.toLowerCase();  
    files[filesCount] = aux;  
    fileName[filesCount]=Paths.get(filePath).getFileName().toString();  
    totalWords = totalWords + files[filesCount] + " ";  
    filesCount++;  
  
    return true;  
}
```

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 todas 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 termos como comparação.

[illegible]

```
public Boolean insertQuery(String query){
    if(query.compareTo("")==0) return false;
    query=removeChars(query);
    query=removeDigits(query);
    query=query.toLowerCase();
    totalWords=totalWords + query + " ";
    this.query=query;
    return true;
}
```

```
public String[] uniqueWords(String texto){
    String[] aux=texto.split( regex " ");
    if(aux.length<2)
        return aux;
    Set<String> unique=new LinkedHashSet<>();
    for(String i : aux){
        if(!unique.contains(i)) {
            unique.add(i);
        }
    }
    Iterator itr=unique.iterator();

    String[] aux2=unique.toArray(new String[0]);
    return aux2;
}
```

15

```

public int[][] matrizOrganizer(String[] uniqueWords){
    int numeroDoc=filesCount,numeroPalavras=uniqueWords.length,count,h;
    String[] aux;

    int[][] matrizM=new int[numeroDoc][numeroPalavras];
    for(int i=0;i<numeroDoc;i++){
        aux=files[i].split( regex: "\\s");
        for(int j=0;j<numeroPalavras;j++){
            count=0;
            h=0;
            while(h<aux.length){
                if(uniqueWords[j].compareTo(aux[h])==0){
                    count++;
                }
                h++;
            }
            matrizM[i][j]=count;
        }
    }
    return matrizM;
}

public int[] matrizOrganizer(String query,String[] uniqueWords){
    int count,h;
    int[] queryArray=new int[uniqueWords.length];
    String[] queroA=query.split( regex: "\\s");
    for(int i=0;i<uniqueWords.length;i++){
        count=0;
        h=0;
        while(h<queroA.length){
            if(uniqueWords[i].compareTo(queroA[h])==0){
                count++;
            }
            h++;
        }
        queryArray[i]=count;
    }
    return queryArray;
}

```

Após esta parte estar concluída passamos para o uso da primeira fórmula onde tivermos que salvar 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.

$$M_{ij} = M_{ij} * 1 + \log_{10} \left(\frac{N}{N_p} \right), \text{ onde :}$$

N - n.º total de documentos;

N_p - n.º de documentos que contêm a palavra p, correspondente à coluna j da matriz. (Nota: procure salvar as divisões na fórmula as divisões por zero);


```

public double[][] matrizModifier(int[][] matrizM, String[] totalWordsM) {
    int contadoc=0;
    double[][] matrizOut=new double[filesCount][totalWordsM.length];
    for(int i=0;i<filesCount;i++){
        for(int j=0;j< totalWordsM.length;j++){
            contadoc=0;
            for(int h=0;h<filesCount;h++){
                if(matrizM[h][j]>0) contadoc++;
            }
            if(contadoc== 0){
                matrizOut[i][j]=0;
            }else {
                matrizOut[i][j] = matrizM[i][j] * (1 + Math.log10((filesCount / contadoc)));
            }
        }
    }
    return matrizOut;
}

public double[] matrizModifier(int[] queryArray,int[][] matrizM,String[] totalWordsM){
    double[] matrizOut=new double[totalWordsM.length];
    int contadoc=0;
    for(int i=0;i<totalWordsM.length;i++){
        contadoc=0;
        for(int h=0;h<filesCount;h++){
            if(matrizM[h][i]>0) contadoc++;
        }
        if(contadoc== 0) {
            matrizOut[i] = 0;
        }else {
            matrizOut[i] = queryArray[i] * (1 + Math.log10((filesCount / contadoc)));
        }
    }

    return matrizOut;
}

```

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.

$$GrauSim = \frac{\sum(M_i * Q_i)}{\sqrt{\sum M_i^2} * \sqrt{\sum Q_i^2}}, \text{ onde :}$$

```

public double[] calculoGrauS(double[][] matrizMFiles, double[] matrizMQuery) {
    double[] grauSim=new double[filesCount];
    for(int i=0;i<filesCount;i++){
        double cima=0;
        for(int l=0;l<matrizMQuery.length;l++){
            cima+=matrizMFiles[i][l]*matrizMQuery[l];
        }
        double baixo=0;
        double esq=0,dir=0;
        for(int l=0;l<matrizMQuery.length;l++){
            esq+=Math.pow(matrizMFiles[i][l],2);
            dir+=Math.pow(matrizMQuery[l],2);
        }
        baixo=(Math.sqrt(esq))*(Math.sqrt(dir));
        grauSim[i]=cima/baixo;
    }
    return grauSim;
}

```

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.

```

public double[] orderGrauS(double[] grauS){
    double temp;
    String auxa;
    for (int i = 0; i <= grauS.length; i++)
    {
        for (int j = i+1; j < grauS.length; j++)
        {
            if (grauS[j] > grauS[i])
            {
                temp = grauS[i];
                grauS[i] = grauS[j];
                grauS[j] = temp;

                auxa=fileName[i];
                fileName[i]=fileName[j];
                fileName[j]=auxa;
            }
        }
    }
    return grauS;
}

```

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

```
public String imprimirLCompleta(double[] grauS,String[] files){
    String imprime="Ficheiro | Grau\n";

    for(int i=0;i<grauS.length;i++){
        imprime+=files[i]+" | " + (float)grauS[i]+" \n";
    }
    return imprime;
}

public String imprimirLLimitada(double[] grauS,String[] files,int quant){

    String imprime="Ficheiro | Grau\n";

    for(int i=0;i<grauS.length && i<quant;i++){
        imprime+=files[i]+" | " + (float)grauS[i]+" \n";
    }
    return imprime;
}

public String imprimirLGrauLimite(double[] grauS,String[] files,double limite){
    String imprime="Ficheiro | Grau\n";
    for(int i=0;i<grauS.length && grauS[i]>limite;i++){
        imprime+=files[i]+" | " + (float)grauS[i]+" \n";
    }
    return imprime;
}
```

4. 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:

BOUNDARY VALUE ANALYSIS insertQuery()							
	LOWER BOUNDRY				UPPER BOUNDRY		
VARIABLES	INVALID PARTITION BY BELOW THE BOUNDRY		VALID PARTITION BY ABOVE THE BOUNDRY		VALID PARTITION BY BELOW THE BOUNDRY		INVALID PARTITION BY ABOVE THE BOUNDRY
query	""		" "		NO MAX AMOUNT		NO MAX AMOUNT

Tabela Test Cases:

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

Testes em Java correspondentes às tabelas:

```
//Teste para insertQuery() quando query válida
@Test
public void test10() { assertTrue(fm.insertQuery("Um2! dois Tres")); }

//Teste para insertQuery() quando query inválida
@Test
public void test11() { assertFalse(fm.insertQuery("")); }
```

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:

BOUNDARY VALUE ANALYSIS insertFile()							
		LOWER BOUNDARY			UPPER BOUNDARY		
VARIABLES	INVALID PARTITION BY BELOW THE BOUNDARY		VALID PARTITION BY ABOVE THE BOUNDARY		VALID PARTITION BY BELOW THE BOUNDARY		PARTITION BY ABOVE THE BOUNDARY
filePath	""		" "		NO MAX AMOUNT		NO MAX AMOUNT

Tabela Test Cases:

TEST CASES insertFile()						
TEST CASE ID	TEST CASE	PRE-CONDITIONS	VARIABLES filePath	EXPECTED RESULTS	ACTUAL RESULTS	Test Status
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:

```
//Teste para insertFile com caminho para documento válido()
@Test
public void test1() { assertEquals( expected: true, fm.insertFile( filePath: "DOC.txt")); }

//Teste para insertFile com caminho inválido()
@Test
public void test2() { assertEquals( expected: false, fm.insertFile( filePath: "DO.txt")); }

//Teste para insertFile com caminho inválido()
@Test
public void test3() { assertEquals( expected: false, fm.insertFile( filePath: "")); }
```

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:

BOUNDARY VALUE ANALYSIS removeDigits()							
LOWER BOUNDARY				UPPER BOUNDARY			
VARIABLES	INVALID PARTITION BY BELOW THE BOUNDARY		VALID PARTITION BY ABOVE THE BOUNDARY		VALID PARTITION BY BELOW THE BOUNDARY		INVALID PARTITION BY ABOVE THE BOUNDARY
texto	""		""		NO MAX AMOUNT		NO MAX AMOUNT

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:

```
//Teste para removeDigits()
@Test
public void test4(){
    assertEquals( expected: "um dois tres", fm.removeDigits( texto: "um1 dois2 tres3"));
}
```

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:

BOUNDARY VALUE ANALYSIS removeChars()							
VARIABLES	LOWER BOUNDRY				UPPER BOUNDRY		
	INVALID PARTITION BY BELOW THE BOUNDRY		VALID PARTITION BY ABOVE THE BOUNDRY		VALID PARTITION BY BELOW THE BOUNDRY		BY ABOVE THE BOUNDRY
texto	000		99 99		NO MAX AMOUNT		NO MAX AMOUNT

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:

```
//Teste para removeChars()
@Test
public void test5(){
    assertEquals( expected: "um dois tres",fm.removeChars( texto: "um? dois! tres#"));
}
```

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:

BOUNDARY VALUE ANALYSIS uniqueWords()							
LOWER BOUNDRY				UPPER BOUNDRY			
VARIABLES	INVALID PARTITION BY BELOW THE BOUNDRY		VALID PARTITION BY ABOVE THE BOUNDRY		VALID PARTITION BY BELOW THE BOUNDRY		BY ABOVE THE BOUNDRY
texto	""		" "		NO MAX AMOUNT		NO MAX AMOUNT

Tabela Test Cases:

TEST CASES uniqueWords()						
TEST CASE ID	TEST CASE	PRE-CONDITIONS	VARIABLES texto	EXPECTED RESULTS	ACTUAL RESULTS	Test Status
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:

```

//Teste para uniqueWords()
@Test
public void test6() {
    String[] teste = {"um", "dois", "tres"};
    String[] teste2=fm.uniqueWords( texto: "um dois tres tres");
    assertAll(
        () -> assertTrue( condition: (teste[0].compareTo(teste2[0])) == 0),
        () -> assertTrue( condition: (teste[1].compareTo(teste2[1])) == 0),
        () -> assertTrue( condition: (teste[2].compareTo(teste2[2])) == 0)
    );
}

//Teste para uniqueWords() (aux)
@Test
public void test7(){
    String[] teste = {"HelloWorld"};
    String[] teste2=fm.uniqueWords( texto: "HelloWorld");
    assertTrue( condition: (teste[0].compareTo(teste2[0])) == 0);
}

```

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:

BOUNDARY VALUE ANALYSIS matrizOrganizer()							
	LOWER BOUNDRY				UPPER BOUNDRY		
VARIABLES	INVALID PARTITION BY BELOW THE BOUNDRY		VALID PARTITION BY ABOVE THE BOUNDRY		VALID PARTITION BY BELOW THE BOUNDRY		INVALID PARTITION BY ABOVE THE BOUNDRY
texto	""		" "		NO MAX AMOUNT		NO MAX AMOUNT

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 occurrences of each word in each file	int[][] with the occurrences of each word in each file	Passed

Teste em Java correspondentes às tabelas:

```
//Teste para matrizOrganizer() para ficheiros
@Test
public void test8(){
    fm.insertFile( filePath: "DOC.txt");
    fm.insertFile( filePath: "DOC2.txt");
    int[][] aux=fm.matrizOrganizer(fm.uniqueWords(fm.getTotalWords()));

    assertEquals( expected: 1,aux[0][0]),
    assertEquals( expected: 3,aux[0][1]),
    assertEquals( expected: 1,aux[0][2]),
    assertEquals( expected: 0,aux[0][3]),
    assertEquals( expected: 0,aux[0][4]),
    assertEquals( expected: 2,aux[1][0]),
    assertEquals( expected: 1,aux[1][1]),
    assertEquals( expected: 0,aux[1][2]),
    assertEquals( expected: 1,aux[1][3]),
    assertEquals( expected: 1,aux[1][4])
};
}
```

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:

BOUNDARY VALUE ANALYSIS matrizModifier()							
VARIABLES	LOWER BOUNDARY				UPPER BOUNDARY		
	INVALID PARTITION BY BELOW THE BOUNDARY		VALID PARTITION BY ABOVE THE BOUNDARY		VALID PARTITION BY BELOW THE BOUNDARY		INVALID PARTITION BY ABOVE THE BOUNDARY
matrizM	[0][0]		[1][1]		NO MAX AMOUNT		NO MAX AMOUNT
totalWordsM	[0]		[1]		NO MAX AMOUNT		NO MAX AMOUNT

Tabela Test Case:

TEST CASES matrizModifier()						
TEST CASE ID	TEST CASE	PRE- CONDITIO NS	VARIABLES matrizM, totalWordsM	EXPECTE D RESULTS	ACTUAL RESULTS	Test Statu s
13	valid variables	fm (FileManager) declared, files and query	int[][] with the ocurrences of each word in each file,	double[][] with the result from the first formula	double[][] with the result from the first formula	Passed

		imported and cleaned	String[] with each unique word			
--	--	-------------------------	--------------------------------------	--	--	--

Testes em Java correspondentes às tabelas:

```
@Test
public void test13(){
    fm.insertFile( filePath: "um.txt");
    fm.insertFile( filePath: "dois.txt");
    fm.insertQuery("um dois tres");
    String query=fm.getQuery();
    String[] totalWords =fm.uniqueWords(fm.getTotalWords());
    int[][] matrizQFiles=fm.matrizOrganizer(totalWords);
    int[] matrizQQuery=fm.matrizOrganizer(query,totalWords);

    double [][] matrizMFiles=fm.matrizModifier(matrizQFiles,totalWords);
    assertEquals(1*(1+Math.log10(2/1)),matrizMFiles[0][0]),
    assertEquals(1*(1+Math.log10(2/1)),matrizMFiles[0][1]),
    assertEquals(1*(1+Math.log10(2/2)),matrizMFiles[0][2]),
    assertEquals(0*(1+Math.log10(2/1)),matrizMFiles[0][3]),
    assertEquals(0*(1+Math.log10(2/1)),matrizMFiles[0][4]),
    assertEquals(0*(1+Math.log10(2/1)),matrizMFiles[1][0]),
    assertEquals(0*(1+Math.log10(2/1)),matrizMFiles[1][1]),
    assertEquals(1*(1+Math.log10(2/2)),matrizMFiles[1][2]),
    assertEquals(1*(1+Math.log10(2/1)),matrizMFiles[1][3]),
    assertEquals(1*(1+Math.log10(2/1)),matrizMFiles[1][4])
};
}
```

Método calculoGrauS():

Tabela ECP:

EQUIVALENCE CLASS PARTITIONING calculoGrauS()		
REQUIREMENT S	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:

BOUNDARY VALUE ANALYSIS calculoGrauS()								
LOWER BOUNDARY					UPPER BOUNDARY			
VARIABLES	INVALID PARTITION THE BOUNDARY	BY BELOW		VALID PARTITION BY ABOVE THE BOUNDARY		VALID PARTITION BY BELOW THE BOUNDARY		INVALID PARTITION BY ABOVE THE BOUNDARY
matrizMFiles	[0][0]			[1][1]		NO MAX AMOUNT		NO MAX AMOUNT
matrizMQuery	[0]			[1]		NO MAX AMOUNT		NO MAX AMOUNT

Tabela Test Cases:

TEST CASES calculoGrauS()						
TEST CASE ID	TEST CASE	PRE- CONDI TIONS	VARIABLE S matrizMFiles, matrizMQuery	EXPECT ED RESULT S	ACTUAL RESULTS	Test Statu s
1	valid variables	fm (FileManag er) declared, files and query imported and cleaned, matrizMFi les and matrizMQ ery already passed through the matrizModi fier	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 second formula	double[] with the result from the second formula	Passed

Testes em Java correspondentes às tabelas:

```
//Teste para calculoGraus
@Test
public void test15(){
    fm.insertFile( filePath: "um.txt");
    fm.insertFile( filePath: "dois.txt");
    fm.insertQuery("um dois tres");
    String query=fm.getQuery();
    String[] totalWords =fm.uniqueWords(fm.getTotalWords());
    int[][] matrizQFiles=fm.matrizOrganizer(totalWords);
    int[] matrizQQuery=fm.matrizOrganizer(query,totalWords);
    double[][] matrizMFiles=fm.matrizModifier(matrizQFiles,totalWords);
    double[] matrizMQuery=fm.matrizModifier(matrizQQuery,matrizQFiles,totalWords);

    double[] grauSim=fm.calculoGraus(matrizMFiles,matrizMQuery);

    assertAll(
        () -> assertEquals( expected: 1.0000000000000002,grauSim[0]),
        () -> assertEquals( expected: 0.22803154893427774,grauSim[1])
    );
}
```

5. Conclusão

Em retrospectiva, este trabalho prático foi o maior desafio encontrado até à data, talvez pelo facto de ser obrigatório empregar diversos softwares de diferentes naturezas.

Não só, como também nos ser exigido uma coordenação e organização soberba, que apenas foi possível recorrendo a entreaajuda e elevada comunicação entre os vários membros do grupo.

Foram diversos os momentos em que os objetivos principais do trabalho se tornavam enevoados e pouco concretos, tendo de recorrer aos docentes da unidade curricular para aconselhamento e auxílio. Estes que se expressaram sempre disponíveis para o prestar.