Relatório Part.2

Engenharia de Software II

Gabriel Pereira 8150115

Nuno Josefino 8150189

Rafael Vieira 8150448



# Índice

**­­**[Índice 2](#_Toc532771235)

[Introdução 4](#_Toc532771236)

[Âmbito do Projeto 5](#_Toc532771237)

[No âmbito 5](#_Toc532771238)

[Fora do âmbito 5](#_Toc532771239)

[Metódo removeDigitos 6](#_Toc532771240)

[Tabela ECP 6](#_Toc532771241)

[Tabela BVA 6](#_Toc532771242)

[Casos de teste 6](#_Toc532771243)

[Caso de teste 1.1 () 6](#_Toc532771244)

[Caso de teste 1.2 () 6](#_Toc532771245)

[Metódo removeCaracteresPont 7](#_Toc532771246)

[Tabela ECP 7](#_Toc532771247)

[Tabela BVA 7](#_Toc532771248)

[Caso de teste 7](#_Toc532771249)

[Caso de teste 2.1 () 7](#_Toc532771250)

[Caso de teste 1.2 () 7](#_Toc532771251)

[Metódo matrizOcorrencias 8](#_Toc532771252)

[Tabela ECP 8](#_Toc532771253)

[Tabela BVA 8](#_Toc532771254)

[Caso de teste 8](#_Toc532771255)

[Caso de teste 3.1 () 8](#_Toc532771256)

[Caso de teste 1.2 () 8](#_Toc532771257)

[Metódo matrizQuery 9](#_Toc532771258)

[Tabela ECP 9](#_Toc532771259)

[Tabela BVA 9](#_Toc532771260)

[Caso de teste 9](#_Toc532771261)

[Caso de teste 4.1 () 9](#_Toc532771262)

[Caso de teste 4.2 () 9](#_Toc532771263)

[Case de teste 4.3 () 10](#_Toc532771264)

[Caso de teste 4.4 () 10](#_Toc532771265)

[Metódo verificarSemelhanca 11](#_Toc532771266)

[Tabela ECP 11](#_Toc532771267)

[Tabela BVA 11](#_Toc532771268)

[Caso de teste 11](#_Toc532771269)

[Caso de teste 5.1 () 11](#_Toc532771270)

[Caso de teste 5.2 () 11](#_Toc532771271)

[Caso de teste 5.3 () 11](#_Toc532771272)

[Metódo listaSim 12](#_Toc532771273)

[Tabela ECP 12](#_Toc532771274)

[Tabela BVA 12](#_Toc532771275)

[Caso de teste 12](#_Toc532771276)

[Caso de teste 6.1 () 12](#_Toc532771277)

[Caso de teste 6.2 () 12](#_Toc532771278)

[Metódo listaSimGrau 13](#_Toc532771279)

[Tabela ECP 13](#_Toc532771280)

[Tabela BVA 13](#_Toc532771281)

[Caso de teste 13](#_Toc532771282)

[Caso de teste 7.1 () 13](#_Toc532771283)

[Caso de teste 7.2 () 13](#_Toc532771284)

[Caso de teste 7.3 () 13](#_Toc532771285)

[Metódo listaSimMax 14](#_Toc532771286)

[Tabela ECP 14](#_Toc532771287)

[Tabela BVA 14](#_Toc532771288)

[Caso de teste 14](#_Toc532771289)

[Caso de teste 8.1 () 14](#_Toc532771290)

[Caso de teste 8.2 () 14](#_Toc532771291)

[Caso de teste 8.3 () 14](#_Toc532771292)

[Conclusão 15](#_Toc532771293)

# Introdução

O presente trabalho surge no âmbito da Unidade Curricular de Engenharia de Software II e em especifico da parte pratica da mesma.

Foi-nos apresentado um desafio no dominio da engenharia de software e em particular na áre da qualidade de software.

Desafio este que consiste em desenvolver um motor de pesquisa que através da introdução de uma frase na query, infere sobre um repositório de ficheiros de texto, sendo necessário o desenvolvimento de uma bibilioteca de software capaz de processar coleções de ficheiros de modo a encontrar ficheiro com um grau de similariedade elevado através de uma query de pesquisa.

Esta parte inicial tem como objetivo a especificação de casos de teste utilizando as técnicas *Equivalence Class Partitioning* e *Boundary Value Analysis* no nível de testes unitários especificando casos de teste para cada método. Deverá ser descrito igualmente *test inputs*, *execution conditions* e *expected outputs*, assegurando que os casos de teste cobrem *Valid equivalence classes* e *Invalid equivalence classes*.

Para elaboração deste projeto, pretendemos utilizar ferramentas com o Github para a hospedagem de todos os ficheiros do nosso projeto e o Youtrack para gerenciar todos os passos a ser tomados no nosso projeto, desde o que falta concretizar ao que irá ser elaborado e se tal irá ocorrer dentro dos prazos a ser estimados.

Prentende-se ainda realizar um estudo das restantes ferramentas a serem utilizadas por nós durante o decorrer do projeto.

Como último objetivo define-se a elaboração de um relatório de testes.

# Âmbito do Projeto

## No âmbito

No âmbito desta fase do trabalho inclui-se a definição de requisitos, realização de tabelas *ECP*, *BVA*, diagrama de casos de uso e relatório de testes.

De igual forma, inclui-se a criação de um repositório *GIT* e a sua configuração, o estudo e criação de um projeto no *Youtrack* enquanto ferramenta de *issue tracking* para o desenvolvimento do projeto.

É incluído também o estudo das restantes ferramentas que irão ser configuradas e utilizadas nos sprints seguintes deste projeto.

## Fora do âmbito

Fora do âmbito desta fase do projeto, inclui-se desenvolvimento de software funcional bem como a utilização de ferramentas para codificação de testes, revisão de código, ferramentas para implementar os conceitos de *CI* durante o desenvolvimento do projeto como *builds* e testes automáticos ou linguagens de programação usada para o desenvolvimento do motor de pesquisa e especificação de testes.

# Diagrama de Use Cases

Uma imagem com texto, mapa

Descrição gerada com confiança muito alta

# 

# Tecnologias Implementadas

Como ferramentas vamos ter:

-O YouTrack, ferramenta issue tracker para desenvolvimento usando metodologias ágeis;

-O Jenkins que vai permitir implementar conceitos de Continuous Integration como builds e testes automáticos;

-O IntelliJ como ambiente de desenvolvimento do código a ser impleementado;

-A linguagem Java que vai ser utilizada para o desenvolvimento do motor de pesquisa e especificação de testes;

-A framework jUnit para codificação dos testes especificados;

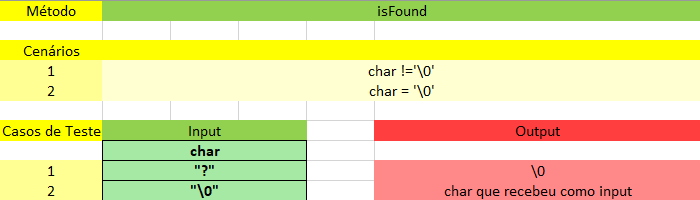
-A plataforma GitHub que serve para hospedagem do código-fonte e de todos os ficheiros referentes a este projeto.

# Metódo isFound

## Tabela ECP



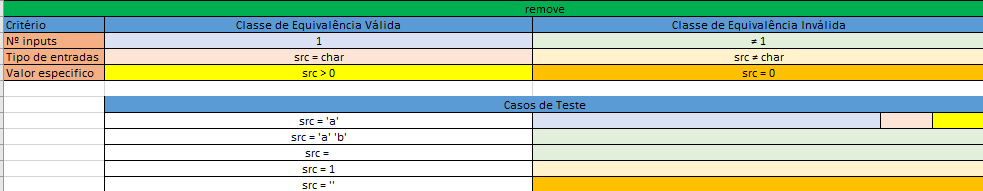
## Tabela BVA



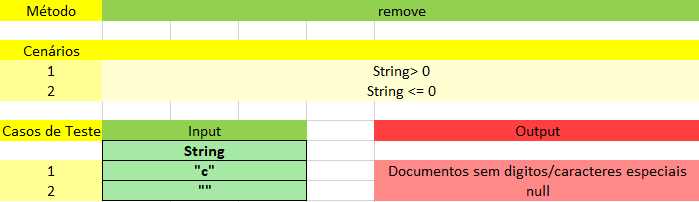
# 

# Metódo remove

## Tabela ECP



## Tabela BVA



## Casos de teste

## Caso de teste 1.1 ()

Se forem recebidos ficheiros, o programa deve ler os ficheiros e remover os dígitos presentes nos ficheiros.

O programa deve devolver os ficheiros sem os dígitos.

## Caso de teste 1.2 ()

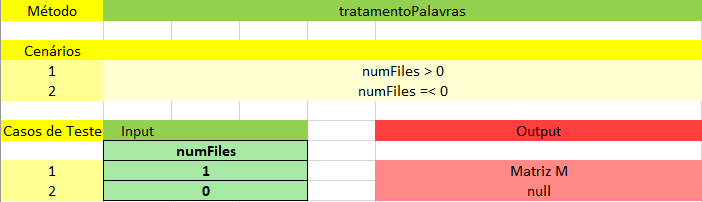
Se não for recebido nenhum ficheiro, o programa deve devolver -1.

# Metódo TratamentoPalavras

## Tabela ECP



## Tabela BVA



## Caso de teste

## Caso de teste 3.1 ()

Se o forem recebidos documentos, o programa deve ler os documentos e criar a matriz M que contem o numero de ocorrências de cada palavra em cada documento.

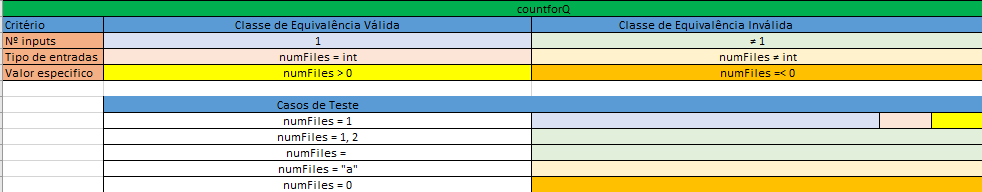
O programa deve devolver a matriz M.

## Caso de teste 3.2 ()

Se não for recebido nenhum documento, o programa deve devolver -1.

# Metódo countforQ

## Tabela ECP

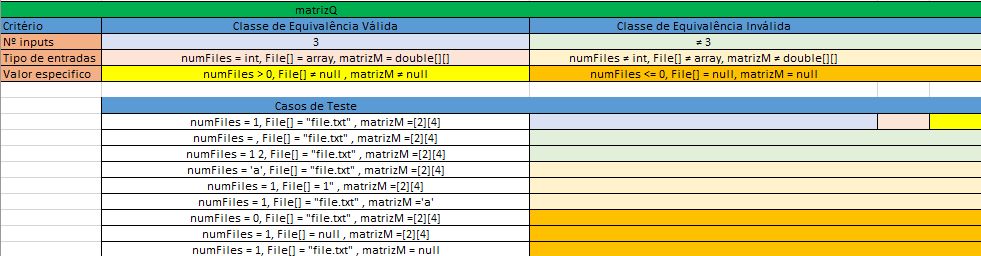


## Tabela BVA

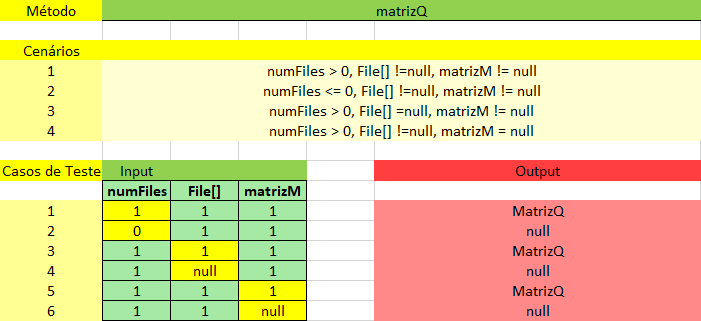
## 

## Metódo matrizQ

## Tabela ECP



## Tabela BVA



## Caso de teste

## Caso de teste 4.1 ()

Se o programa receber a matriz M, o numero de documentos e o numero de documentos em que a palavra p aparece, o programa deve, através da formula designada, criar a matriz arrayQ.

O programa deve devolver a matriz arrayQ.

## Caso de teste 4.2 ()

Se não for recebida a matriz M, o programa deve devolver -1.

## 

## Case de teste 4.3 ()

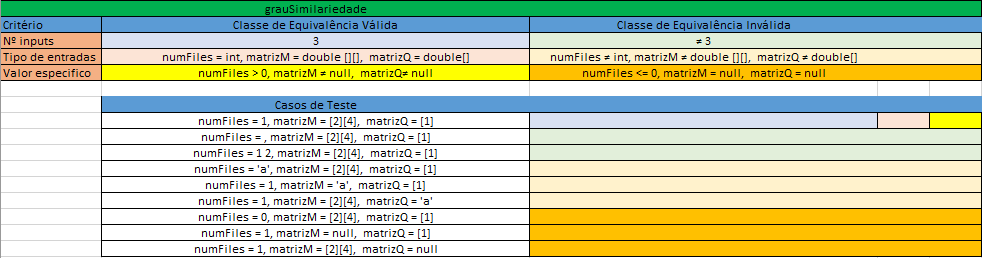
Se não for recebido o numero de documentos, o programa deve devolver -1.

## Caso de teste 4.4 ()

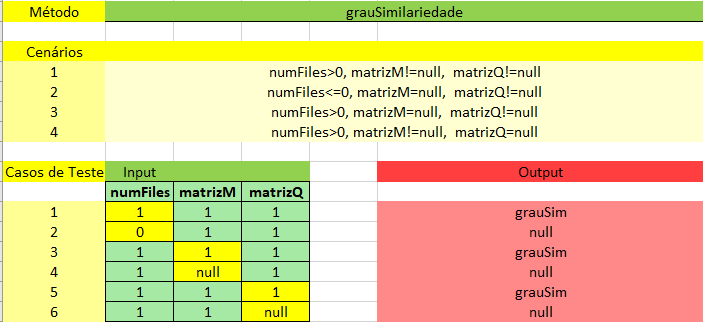
Se não for recebido o numero de documentos em que a palavra p aparece, o programa deve devolver -1.

# Metódo grauSimilariedade

## Tabela ECP



## Tabela BVA



## Caso de teste

## Caso de teste 5.1 ()

Se o programa receber a matriz M e a matriz array Q, o programa deve, através da formula designada, calcular o grau de similaridade (grauSim).

O programa deve devolver o grauSim.

## Caso de teste 5.2 ()

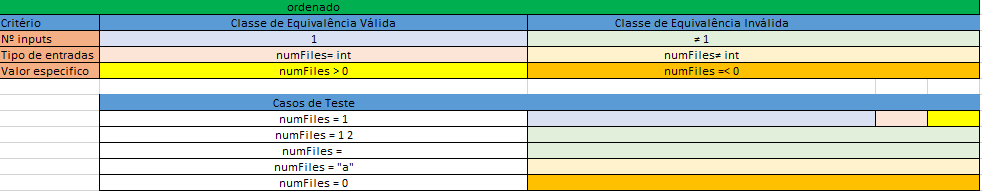
Se não for recebido o arrayQ, o programa deve devolver -1.

## Caso de teste 5.3 ()

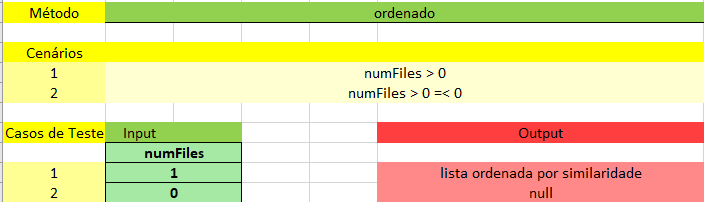
Se não for recebida a matriz M, o programa deve devolver -1.

# Metódo ordenado

## Tabela ECP



## Tabela BVA



## Caso de teste

## Caso de teste 6.1 ()

Se o programa receber a lista com os rankings de similaridade (listaRanking), o programa deve ordenar a lista pelos respetivos rankings.

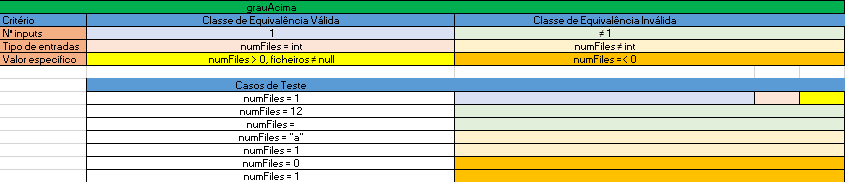
O programa deve devolver a lista ordenada por similaridade.

## Caso de teste 6.2 ()

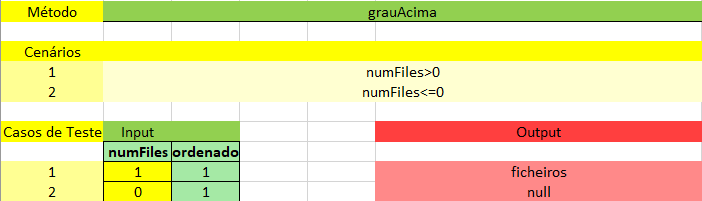
Se não for recebida a listaRanking, o programa deve devolver -1.

# Metódo grauAcima

## Tabela ECP



## Tabela BVA



## Caso de teste

## Caso de teste 7.1 ()

Se o programa receber a listaRanking e um valor para o grau que se quer definir, o programa criar uma lista com os ficheiros que têm um grau de similaridade acima do definido.

O programa deve devolver a lista com similaridade acima do grau.

## Caso de teste 7.2 ()

Se não for recebida a listaRanking, o programa deve devolver -1.

## Caso de teste 7.3 ()

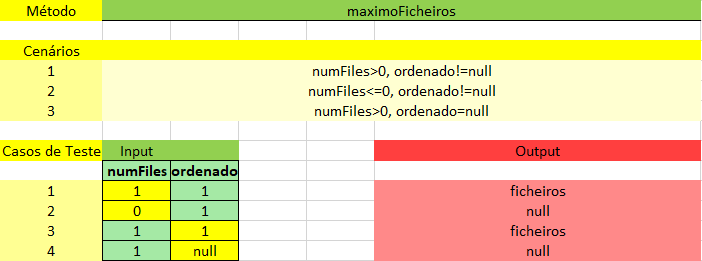
Se o valor do grau para criar a lista for inferior a zero, o programa deve devolver -1.

# Metódo maximoFicheiros

## Tabela ECP



## Tabela BVA



## Caso de teste

## Caso de teste 8.1 ()

Se o programa receber a listaRanking e um valor para o máximo de ficheiros que se quer ver, o programa criar uma lista com o número de ficheiros, ordenados pela similaridade, igual ao valor máximo definido.

O programa deve devolver a lista com máximo de ficheiros.

## Caso de teste 8.2 ()

Se não for recebida a listaRanking, o programa deve devolver -1.

## Caso de teste 8.3 ()

Se o valor máximo de ficheiros que se quer ver for igual ou menor que zero, o programa deve devolver -1.

# Conclusão

Este projeto vem dar resposta a uma solicitação referente ao sprint #1 da Unidade Curricular de Engenharia de Software II que irá ser realizado no dia 17/12/18, para tal usamos as técnicas *Equivalence Class Partitioning* e *Boundary Value Analysis,* descrição de *test inputs, execution conditions* e *expected outputs,* assegurando que os casos de teste cobrem *Valid equivalence classes* e *Invalid equivalence classes.*

Realizamos ainda um estudo das ferramentas que irão ser utilizadas posteriormente no decorrer deste projecto.