Relatório Part.2

Engenharia de Software II

Gabriel Pereira 8150115

Nuno Josefino 8150189

Rafael Vieira 8150448



# Índice

[Índice 2](#_Toc535161203)

[Introdução 4](#_Toc535161204)

[Âmbito do Projeto 5](#_Toc535161205)

[No âmbito 5](#_Toc535161206)

[Fora do âmbito 5](#_Toc535161207)

[Diagrama de Use Cases 6](#_Toc535161208)

[Tecnologias Implementadas 6](#_Toc535161209)

[Metódo isFound 7](#_Toc535161210)

[Tabela ECP 7](#_Toc535161211)

[Tabela BVA 8](#_Toc535161212)

[Metódo remove 8](#_Toc535161213)

[Tabela ECP 8](#_Toc535161214)

[Tabela BVA 9](#_Toc535161215)

[Casos de teste 9](#_Toc535161216)

[Caso de teste 1.1 () 9](#_Toc535161217)

[Caso de teste 1.2 () 9](#_Toc535161218)

[Metódo TratamentoPalavras 9](#_Toc535161219)

[Tabela ECP 9](#_Toc535161220)

[Tabela BVA 10](#_Toc535161221)

[Caso de teste 10](#_Toc535161222)

[Caso de teste 3.1 () 10](#_Toc535161223)

[Caso de teste 3.2 () 10](#_Toc535161224)

[Metódo countforQ 10](#_Toc535161225)

[Tabela ECP 10](#_Toc535161226)

[Tabela BVA 11](#_Toc535161227)

[11](#_Toc535161228)

[Metódo matrizQ 11](#_Toc535161229)

[Tabela ECP 11](#_Toc535161230)

[Tabela BVA 12](#_Toc535161231)

[Caso de teste 12](#_Toc535161232)

[Caso de teste 4.1 () 12](#_Toc535161233)

[Caso de teste 4.2 () 12](#_Toc535161234)

[Case de teste 4.3 () 12](#_Toc535161235)

[Caso de teste 4.4 () 12](#_Toc535161236)

[Metódo grauSimilariedade 13](#_Toc535161237)

[Tabela ECP 13](#_Toc535161238)

[Tabela BVA 14](#_Toc535161239)

[Caso de teste 14](#_Toc535161240)

[Caso de teste 5.1 () 14](#_Toc535161241)

[Caso de teste 5.2 () 14](#_Toc535161242)

[Caso de teste 5.3 () 14](#_Toc535161243)

[Metódo ordenado 15](#_Toc535161244)

[Tabela ECP 15](#_Toc535161245)

[Tabela BVA 15](#_Toc535161246)

[Caso de teste 15](#_Toc535161247)

[Caso de teste 6.1 () 15](#_Toc535161248)

[Caso de teste 6.2 () 15](#_Toc535161249)

[Metódo grauAcima 16](#_Toc535161250)

[Tabela ECP 16](#_Toc535161251)

[Tabela BVA 16](#_Toc535161252)

[Caso de teste 16](#_Toc535161253)

[Caso de teste 7.1 () 16](#_Toc535161254)

[Caso de teste 7.2 () 16](#_Toc535161255)

[Caso de teste 7.3 () 16](#_Toc535161256)

[Metódo maximoFicheiros 17](#_Toc535161257)

[Tabela ECP 17](#_Toc535161258)

[Tabela BVA 17](#_Toc535161259)

[Caso de teste 17](#_Toc535161260)

[Caso de teste 8.1 () 17](#_Toc535161261)

[Caso de teste 8.2 () 17](#_Toc535161262)

[Caso de teste 8.3 () 17](#_Toc535161263)

[Conclusão 18](#_Toc535161264)

# 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, tendo sido 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, utilizamos 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 realizamos um estudo prévio das restantes ferramentas que foram utilizadas por nós durante o decorrer do projeto.

Como último objetivo definimos a elaboração de um relatório de testes.

# Âmbito do Projeto

## No âmbito

No âmbito desta fase final do trabalho incluimos a definição de requisitos, realização de tabelas *ECP*, *BVA*, diagrama de casos de uso, codificação dos métodos e criação dos testes e elaboração dos relatório de testes.

De igual forma, incluimos 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, incluimos o desenvolvimento de software funcional bem como a utilização de ferramentas para codificação de testes, revisão de código, ferramentas para implementação dos 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 usamos:

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

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

-O IntelliJ como ambiente de desenvolvimento do código que fomos desenvolvendo e impleementado;

-A linguagem Java que vai ser utilizada onde desenvolvemos todo o código do motor de pesquisa e especificação de testes;

-A framework jUnit para codificação dos testes especificados;

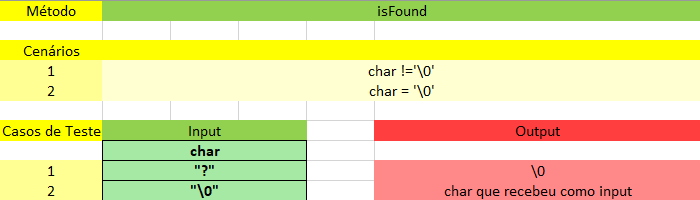
-A plataforma GitHub que serviu para hospedagem do código-fonte e de todos os ficheiros referentes a este projeto.

# Metódo isFound

## 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 caracteres presentes nos ficheiros.

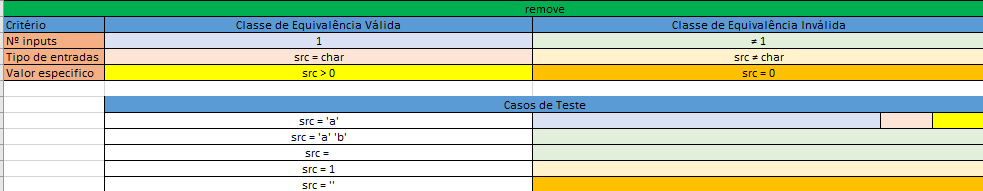
O programa deve devolver os ficheiros sem os dígitos.

## Caso de teste 1.2 ()

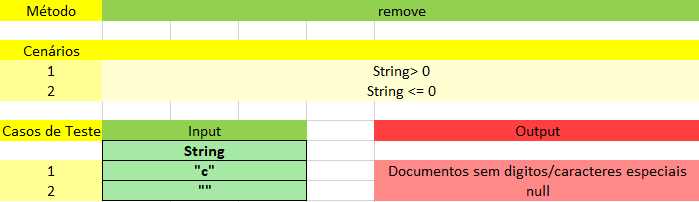
Se não for recebido nenhum ficheiro, o programa deve devolver -1.

# Metódo 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 ()

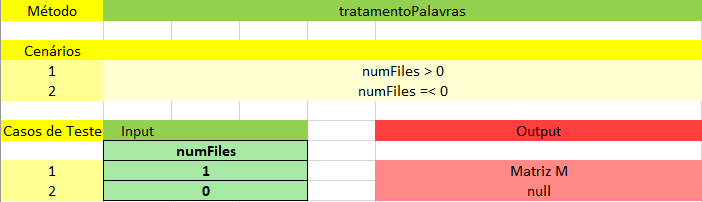
Se não for recebido nenhum ficheiro, o programa deve devolver -1.

# Metódo TratamentoPalavras

## Tabela ECP



## Tabela BVA



## Caso de teste

## Caso de teste 3.1 ()

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

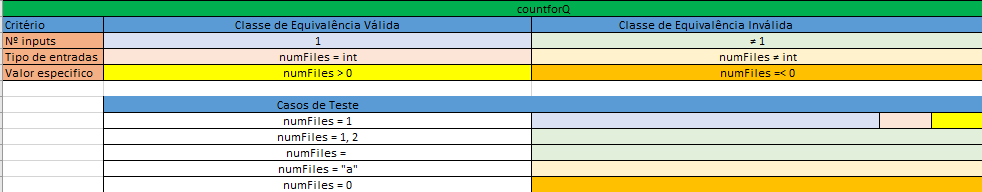
O programa deve devolver a matriz M.

## Caso de teste 3.2 ()

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

# Metódo countforQ

## Tabela ECP

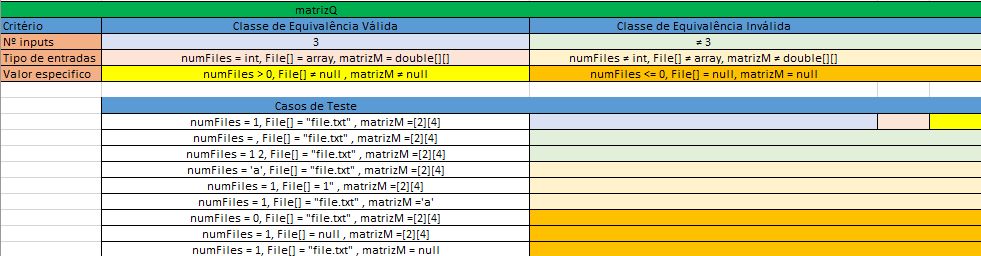


## Tabela BVA

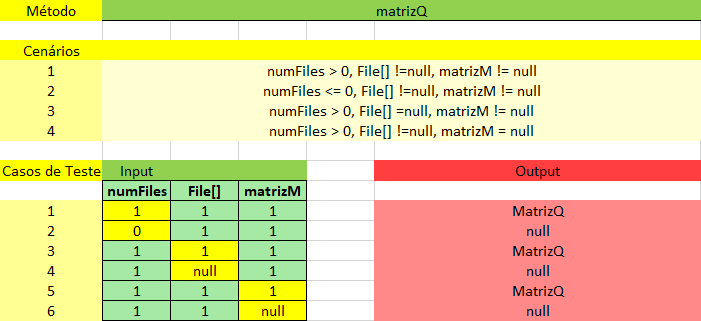
## 

## 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 ()

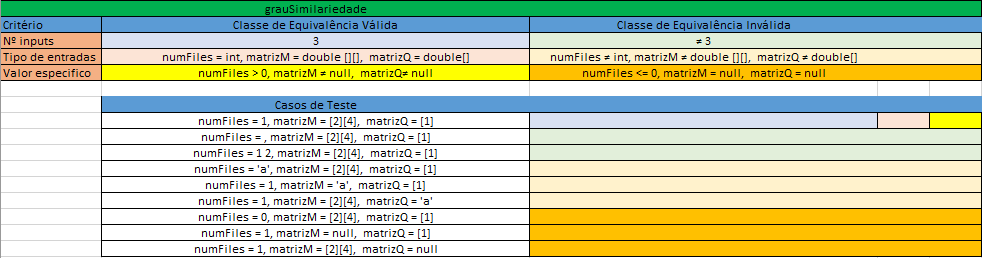
Se não for recebido o numero de documentos, o programa deve devolver -1.

## Caso de teste 4.4 ()

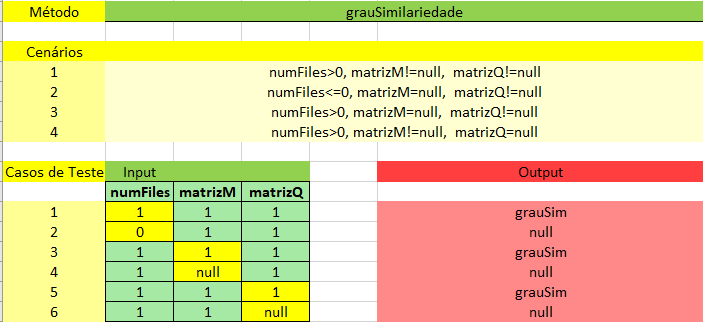
Se não for recebido o numero de documentos em que a palavra p aparece, o programa deve devolver -1.

# 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 grauSimilaridade.

O programa deve devolver o grauSimilariedade.

## Caso de teste 5.2 ()

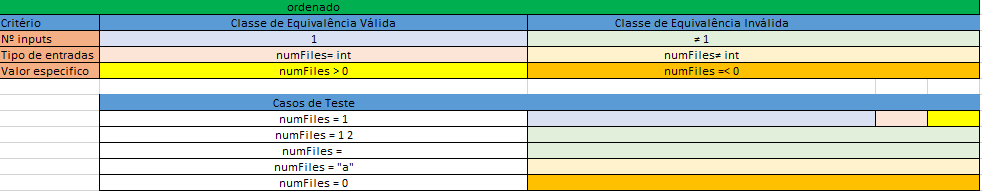
Se não for recebido o arrayQ, o programa deve devolver -1.

## Caso de teste 5.3 ()

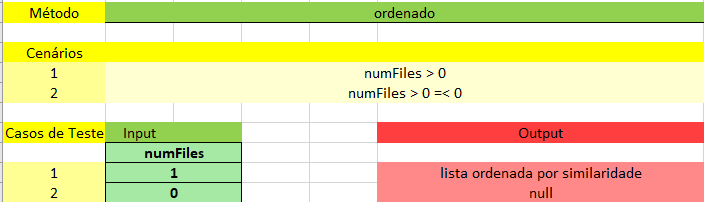
Se não for recebida a matriz M, o programa deve devolver -1.

# 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 (ordenado), o programa deve ordenar a lista pelos respetivos rankings.

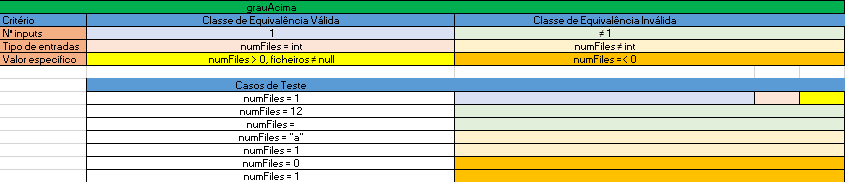
O programa deve devolver a lista ordenada por similaridade.

## Caso de teste 6.2 ()

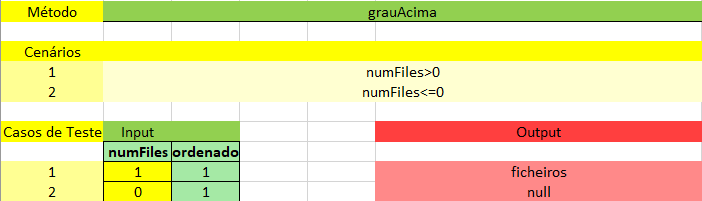
Se não for recebida a lista de ordenado, o programa deve devolver -1.

# Metódo grauAcima

## Tabela ECP



## Tabela BVA



## Caso de teste

## Caso de teste 7.1 ()

Se o programa receber a lista de ordenados e um valor para o grau que se quer definir, o programa cria 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 lista de ordenados, o programa deve devolver -1.

## Caso de teste 7.3 ()

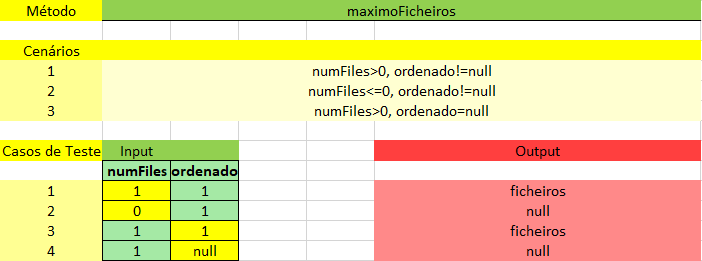
Se o valor do grau para criar a lista for inferior a zero, o programa deve devolver -1.

# Metódo maximoFicheiros

## Tabela ECP



## Tabela BVA



## Caso de teste

## Caso de teste 8.1 ()

Se o programa receber a lista de ordenados e um valor para o máximo de ficheiros que se quer ver, o programa cria 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 lista de ordenados, 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 Trabalho Prático da Unidade Curricular de Engenharia de Software II que irá ser apresentado realizado no dia 13/01/19, 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.