|  |  |
| --- | --- |
|  | *Relatório de inspeção de código*  *Projeto de desenvolvimento de software de configuração anti-spam*  BSC/msc em [meti]  Ano acadêmico 2017/2018-1 º semestre  Engenharia de software I  Grupo 47  61829, Tomás Pires, METI  64454, Joana Antunes, METI  65106, Soraia Fernandes, METI  70082, Bruno Mataloto, Mattie  ISCTE-IUL, Instituto Universitário de Lisboa  1649-026 Lisboa  Portugal  20 de dezembroth 2017 |

**Índice**

[Introdução 3](#_Toc498465002)

[Inspeção de código – nome do componente que está a ser inspecionado 3](#_Toc498465003)

[Lista de verificação de inspeção de código 3](#_Toc498465004)

[Defeitos encontrados 3](#_Toc498465005)

[Medidas correctivas 3](#_Toc498465006)

[Conclusões do processo de inspecção 3](#_Toc498465007)

# Introdução

*O software produzido tem como objetivo configurar um filtro antispam para uma Leisure Box , em que o utilizador insere o ficheiro de regras rules.cf,o ficheiro ham.log e o ficheiro spam.log e o software gera automaticamente uma configuração ideal, possivel de ser editada pelo utilizador. Permite ainda a criação de uma configuração totalmente manual ou aleatória.*

# Inspeção de código – Filtro AntiSpam

*Este relatório resulta da inspeção ao código desenvolvido pelo grupo.*

|  |  |
| --- | --- |
| *Data da reunião:*  *Duração da reunião:*  *Moderador:*  *Produtor:*  *Inspetor:*  *Gravador:* | *20/12/2017*  *40 minutos*  *Joana Antunes*  *Bruno Mataloto*  *Soraia Fernandes*  *Tomás Pires* |
| *Nome do componente (package/class/método):* | *Package : AntiSpamFilter* |
| *Cobertura de testes alcançada:* | *81,4%* |

# Lista de verificação de inspeção de código

A lista de verificação para a inspeção de código Java usada neste projeto está disponível em [http://www.cs.Toronto.edu/~SME/CSC444F/handouts/java\_checklist.pdf](http://www.cs.toronto.edu/~sme/CSC444F/handouts/java_checklist.pdf) e no blackboard ES1 página.

# Defeitos encontrados

|  |  |  |  |
| --- | --- | --- | --- |
| **ID do defeito encontrado** | **Package, Class, Method, Line** | **Categoria de defeito** | **Descrição** |
| 1 | Class: Controller | 1 | Atributo String da classe Rule que nomeava a regra, tinha a mesma designação que a própria Rule na classe Controller o que geráva alguam confusão |
| 2 | Class: GUI | 2 | Existem métodos extra desnecessários, que foram criados para permitir testar ActionListeners presentes na classe |
| 3 | Class: Email e Controller | 14 | Loop de pesquisa das regras no email era muito complexo e atrasava a geração da configuração automática, tendo sido adicionado os indices das regras correspondentes a cada email tornando o ciclo extremamente mais simples e rápido |
| 4 | Package : AntiSpamFilter | 10 | A linguagem utilizada para os Java Docs em alguns casos encontra-se em Inglês e Português |
| 5 | Class Controller | 13 | Começámos por utilizar Vetores, mas devido á dificuldade de gestão de espaço para outros ficheiros, optámos por mudar para ArrayLists |

# Medidas correctivas

*Alguns destes defeitos foram facilmente corrigidos no sprint respetivo, no entanto o problema 3 necessitou de mais tempo pois foi necessário alterar métodos de várias classes. Para a geração de testes JUnit os métodos extra foram necessários para melhorar a cobertura do código, uma vez que possuímos bastante código dentro de ActionListeners que de outra forma nunca seriam testados.*

# Conclusões do processo de inspecção

*Podemos concluir depois de efetuarmos a análise ao código produzido, que apesar de executar todas as funcionalidades necessárias, existem sempre formas de melhorar a sua organização, tornar a interpretação por parte de outras pessoas mais simples através de código mais claro e conciso. Nem sempre são adotadas todas as medidas necessárias para obter uma boa qualidade geral, no entanto a inspeção do código permite detetar situações que passariam despercebidas.*