|  |  |
| --- | --- |
|  | *Code Inspection Report*  *‘Bom Dia Academia’ Software Development Project*  BSc/MSc in [LEI | LIGE | METI]  Academic Year 2018/2019 - 1º Semester  Software Engineering I  Grupo 88  60396, André Mota, EIC1-PL  24227, Elvino Monteiro, EIC2-PL  69016, Bernardo Kelly, IC2  ISCTE-IUL, Instituto Universitário de Lisboa  1649-026 Lisbon  Portugal  November 2018 |

**Table of Contents**

[Introduction 3](#_Toc498465002)

[Code inspection – Name of the component being inspected 3](#_Toc498465003)

[Code inspection checklist 3](#_Toc498465004)

[Found defects 3](#_Toc498465005)

[Corrective measures 3](#_Toc498465006)

[Conclusions of the inspection process 3](#_Toc498465007)

# Introduction

*O projecto consiste no desenvolvimento de software para a integração de informação académica com origem em vários sistemas (fontes de informação académica). A aplicação chama-se de Bom Dia Academia (BDA) e deverá suportar funcionalidades para agregação de conteúdos académicos.*

*Permite o acesso à informação académica disponibilizada através dos canais, nomeadamente, Email, Facebook, Twitter.*

*A aplicação deverá permitir que o utilizador não só aceda/consulte, mas também possa enviar informação através desses canais (resposta a emails, posts Facebook, retweets).*

# Code inspection – Name of the component being inspected

*Description of the software component being inspected*

|  |  |
| --- | --- |
| *Meeting date:*  *Meeting duration:*  *Moderator:*  *Producer:*  *Inspector:*  *Recorder:* | *??/??/2017*  *?? minutes*  *Elvino*  *André Mota & Bernardo Kelly*  *André Mota*  *Andrré Mota* |
| *Component name (Package/Class/Method):* | *es.projecto* |
| *Component was compiled:* | *YES* |
| *Component was executed:* | *YES* |
| *Component was tested without errors:* | *YES* |
| *Testing coverage achieved:* | *86%* |

# Code inspection checklist

1. Variable, Attribute, and Constant Declaration Defects (VC)

* Are descriptive variable and constant names used in accord with naming conventions?
* Are there variables or attributes with confusingly similar names?
* Is every variable and attribute correctly typed?
* Is every variable and attribute properly initialized?
* Could any non-local variables be made local?
* Are all for-loop control variables declared in the loop header?
* Are there literal constants that should be named constants?
* Are there variables or attributes that should be constants?
* Are there attributes that should be local variables?
* Do all attributes have appropriate access modifiers (private, protected, public)?
* Are there static attributes that should be non-static or vice-versa?

1. Method Definition Defects (FD)

* Are descriptive method names used in accord with naming conventions?
* Is every method parameter value checked before being used?
* For every method: Does it return the correct value at every method return point?
* Do all methods have appropriate access modifiers (private, protected, public)?
* Are there static methods that should be non-static or vice-versa?

1. Class Definition Defects (CD)

* Does each class have appropriate constructors and destructors?
* Do any subclasses have common members that should be in the superclass?
* Can the class inheritance hierarchy be simplified?

1. Data Reference Defects (DR)

* For every array reference: Is each subscript value within the defined bounds?
* For every object or array reference: Is the value certain to be non-null?

1. Computation/Numeric Defects (CN)

* Are there any computations with mixed data types?
* Is overflow or underflow possible during a computation?
* For each expressions with more than one operator: Are the assumptions about order of evaluation and precedence correct?
* Are parentheses used to avoid ambiguity?

1. Comparison/Relational Defects (CR)

* For every boolean test: Is the correct condition checked?
* Are the comparison operators correct?
* Has each boolean expression been simplified by driving negations inward?
* Is each boolean expression correct?
* Are there improper and unnoticed side-effects of a comparison?
* Has an "&" inadvertently been interchanged with a "&&" or a "|" for a "||"?

1. Control Flow Defects (CF)

* For each loop: Is the best choice of looping constructs used?
* Will all loops terminate?
* When there are multiple exits from a loop, is each exit necessary and handled properly?
* Does each switch statement have a default case?
* Are missing switch case break statements correct and marked with a comment?
* Do named break statements send control to the right place?
* Is the nesting of loops and branches too deep, and is it correct?
* Can any nested if statements be converted into a switch statement?
* Are null bodied control structures correct and marked with braces or comments?
* Are all exceptions handled appropriately?
* Does every method terminate? 8. Input-Output Defects (IO)
* Have all files been opened before use?
* Are the attributes of the input object consistent with the use of the file?
* Have all files been closed after use?
* Are there spelling or grammatical errors in any text printed or displayed?
* Are all I/O exceptions handled in a reasonable way?

1. Input-Output Defects (IO)

* Have all files been opened before use?
* Are the attributes of the input object consistent with the use of the file?
* Have all files been closed after use?
* Are there spelling or grammatical errors in any text printed or displayed?
* Are all I/O exceptions handled in a reasonable way?

1. Module Interface Defects (MI)

* Are the number, order, types, and values of parameters in every method call in agreement with the called method's declaration?
* Do the values in units agree (e.g., inches versus yards)?
* If an object or array is passed, does it get changed, and changed correctly by the called method?

1. Comment Defects (CM)

* Does every method, class, and file have an appropriate header comment?
* Does every attribute, variable, and constant declaration have a comment?
* Is the underlying behavior of each method and class expressed in plain language?
* Is the header comment for each method and class consistent with the behavior of the method or class?
* Do the comments and code agree?
* Do the comments help in understanding the code?
* Are there enough comments in the code?
* Are there too many comments in the code?

1. Layout and Packaging Defects (LP)

* Is a standard indentation and layout format used consistently?
* For each method: Is it no more than about 60 lines long?
* For each compile module: Is no more than about 600 lines long?

1. Modularity Defects (MO)

* Is there a low level of coupling between modules (methods and classes)?
* Is there a high level of cohesion within each module (methods or class)?
* Is there repetitive code that could be replaced by a call to a method that provides the behavior of the repetitive code?
* Are the Java class libraries used where and when appropriate?

1. Storage Usage Defects (SU)

* Are arrays large enough?
* Are object and array references set to null once the object or array is no longer needed?

1. Performance Defects (PE)

* Can better data structures or more efficient algorithms be used?
* Are logical tests arranged such that the often successful and inexpensive tests precede the more expensive and less frequently successful tests?
* Can the cost of recomputing a value be reduced by computing it once and storing the results?
* Is every result that is computed and stored actually used?
* Can a computation be moved outside a loop?
* Are there tests within a loop that do not need to be done?
* Can a short loop be unrolled?
* Are there two loops operating on the same data that can be combined into one?
* Are frequently used variables declared register?
* Are short and commonly called methods declared inline?

# Found defects

Identify and describe found defects, opinions and suggestions.

|  |  |  |  |
| --- | --- | --- | --- |
| **Found defect Id** | **Package, Class, Method, Line** | **Defect category** | **Description** |
| 1 | es.projecto.facebook.conection;  FacebookBDAClient | CM | Não tem comentários em nenhuma parte do código, tirando javadc |
| 2 | es.projecto.email; EmailClient | PE | Podiaser melhorada a eficiencia de codigo |
| 3 |  |  |  |
| ... | ... | … | ... |

# Corrective measures

*Found defect Id, how/when/who will correct the identified defect.*

# Conclusions of the inspection process

*O Codigo em geral está bom, deveria ter mais comentarios, tem indentação correcta e nomes apropriados, sem erros gramaticais a nível de output etc.*

**Note: Each group may adapt the current template according to its needs and preferences or adopt a different template. Decisions on template adaptations or the use of a different template must be justified.**