**Instituto Politécnico de Coimbra**

***Instituto Superior de Engenharia de Coimbra***

**Disciplina Programação Avançada**

Trabalho Prático – Meta 1

**Docente: Álvaro Santos**

Pedro Jorge Fernandes Morais – 2018020733 – LEI

Maria Beatriz Girão Abreu – 2020130937 – LEI-CE

domingo, 23 de maio de 2021

**Índice**

[1 Opções tomadas na implementação 1](#_Toc102254481)

[2 Diagrama Máquina de Estados 2](#_Toc102254482)

[3 Outros padrões de Programação 3](#_Toc102254483)

[4 Classes Utilizadas 4](#_Toc102254484)

[*4.1* *pt.isec.pa.apoio\_poe.model.cmd* 4](#_Toc102254485)

[4.1.1. ICommand 4](#_Toc102254486)

[4.1.2. CommandManager 4](#_Toc102254487)

[4.1.3. CommandAdapter 4](#_Toc102254488)

[4.1.4. AddAtribuicaoAlunoProposta 4](#_Toc102254489)

[4.1.5. RemoveAtribuicaoAlunoProposta 5](#_Toc102254490)

[4.1.6. AddOrientador 5](#_Toc102254491)

[4.1.7. ChangeOrientador 5](#_Toc102254492)

[4.1.8. RemoveOrientador 5](#_Toc102254493)

[4.2 pt.isec.pa.apoio\_poe.model.data 6](#_Toc102254494)

[4.2.1. Pessoa 6](#_Toc102254495)

[4.2.2. Docente 6](#_Toc102254496)

[4.2.3. Aluno 6](#_Toc102254497)

[4.2.4. AlunoClassificacaoComparator 6](#_Toc102254498)

[4.2.5. Proposta 6](#_Toc102254499)

[4.2.6. Projeto 7](#_Toc102254500)

[4.2.7. Estagio 7](#_Toc102254501)

[4.2.8. Autoproposto 7](#_Toc102254502)

[4.2.9. PropostaAtribuida 7](#_Toc102254503)

[4.2.10. Candidatura 7](#_Toc102254504)

[4.2.11. ApoioPoE 8](#_Toc102254505)

[4.2.12. ApoioPoEManager 8](#_Toc102254506)

[4.3 pt.isec.pa.apoio\_poe.model.exceptionsHandling 8](#_Toc102254507)

[4.3.1. ExceptionsTypes 8](#_Toc102254508)

[4.3.2. ExceptionOccurred 8](#_Toc102254509)

[4.4 pt.isec.pa.apoio\_poe.model.facade 9](#_Toc102254510)

[4.4.1. GestaoManualAtribuicoesManager 9](#_Toc102254511)

[4.4.2. GestaoManualOrientadoresManager 9](#_Toc102254512)

[4.5 pt.isec.pa.apoio\_poe.model.fsm 9](#_Toc102254513)

[4.5.1. ApoioPoEState 9](#_Toc102254514)

[4.5.2. IApoioPoEState 9](#_Toc102254515)

[4.5.3. ApoioPoEContext 9](#_Toc102254516)

[4.5.4. ApoioPoEAdapter 10](#_Toc102254517)

[4.5.5. InicioState 10](#_Toc102254518)

[4.5.6. Fase1State 10](#_Toc102254519)

[4.5.7. Fase1BloqueadaState 10](#_Toc102254520)

[4.5.8. Fase2 10](#_Toc102254521)

[4.5.9. Fase2BloqueadaState 10](#_Toc102254522)

[4.5.10. Fase3 10](#_Toc102254523)

[4.5.11. Fase3MasFase2AbertaState 10](#_Toc102254524)

[4.5.12. Fase3BloqueadaState 11](#_Toc102254525)

[4.5.13. Fase3AtribuicaoAutomaticaState 11](#_Toc102254526)

[4.5.14. Fase4 11](#_Toc102254527)

[4.5.15. Fase5 11](#_Toc102254528)

[4.5.16. GestaoAlunosState 11](#_Toc102254529)

[4.5.17. GestaoDocentesState 11](#_Toc102254530)

[4.5.18. GestaoPropostasState 11](#_Toc102254531)

[4.5.19. GestaoCandidaturasState 11](#_Toc102254532)

[4.5.20. GestaoManualAtribuicoesState 12](#_Toc102254533)

[4.5.21. GestaoManualOrientadoresState 12](#_Toc102254534)

[4.6 pt.isec.pa.apoio\_poe.ui.text 12](#_Toc102254535)

[4.6.1. ApoioPoEUI 12](#_Toc102254536)

[4.7 pt.isec.pa.apoio\_poe.utils 12](#_Toc102254537)

[4.7.1. PAInput 12](#_Toc102254538)

[4.8 pt.isec.pa.apoio\_poe 12](#_Toc102254539)

[4.8.1. Main 12](#_Toc102254540)

[5 Relacionamento entre Classes 13](#_Toc102254541)

[6 Funcionalidades Implementadas 14](#_Toc102254542)

**Índice de Figuras**

**Não foi encontrada nenhuma entrada do índice de ilustrações.**

**Índice de Tabelas**

[Tabela 1 - Funcionalidades Fase 1 6](#_Toc102232731)

[Tabela 2 - Funcionalidades Fase 2 7](#_Toc102232732)

[Tabela 3 - Funcionalidades Fase 3 7](#_Toc102232733)

[Tabela 4 - Funcionalidades Fase 4 9](#_Toc102232734)

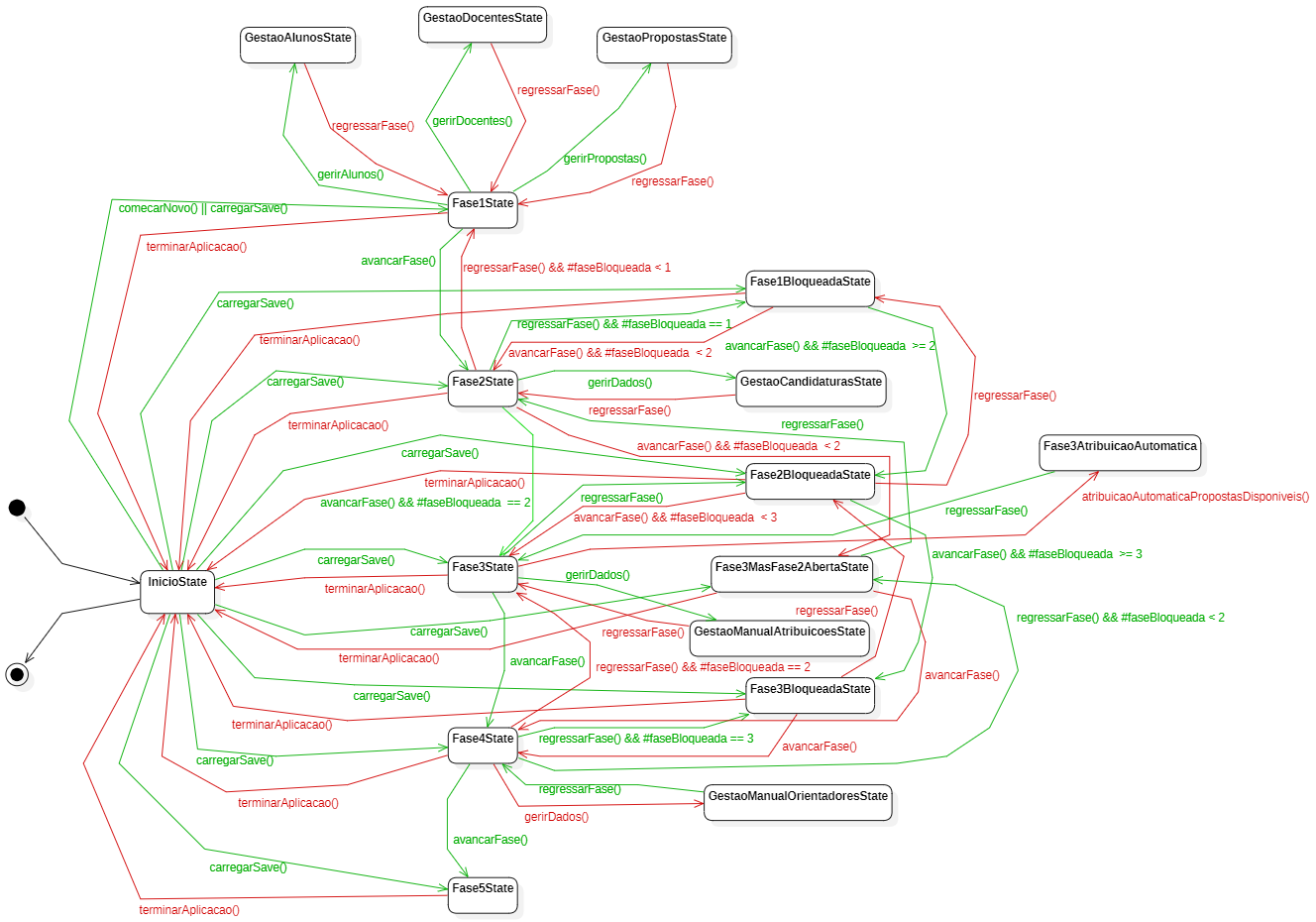
[Tabela 5 - Funcionalidades Fase 5 9](#_Toc102232735)

[Tabela 6 - Outras Funcionalidades 10](#_Toc102232736)

# Opções tomadas na implementação

Introdução

# Diagrama Máquina de Estados

****

# Outros padrões de Programação

Texto 2, referência [[2]](#Ref_2)

# Classes Utilizadas

## *pt.isec.pa.apoio\_poe.model.cmd*

### *ICommand*

A classe *ICommand* é a interface base para todos os comandos. Todos os comandos implementados terão uma ação de *execute()* e outra de *undo()*. A operação *execute()* executa o comando e retorna *true* caso este tenha sido executado com sucesso. A operação *undo()* reverte as alterações que tinham sido realizadas com a operação *execute()*.

### *CommandManager*

A classe *CommandManager* é o gestor de comandos, responsável por executar os comandos, guardar um histórico dos comandos realizados e guardar também os comandos que foram revertidos, podendo voltar a executá-los.

### *CommandAdapter*

A classe *CommandAdapter* tem como objetivo disponibilizar implementações padrão para as operações da interface *ICommand*. Desta forma as classes que estenderem da classe *CommandAdapter* não necessitarão de implementar todas as operações possíveis, apenas terão de redefinir as que realmente necessitam.

### AddAtribuicaoAlunoProposta

A classe *AddAtribuicaoAlunoProposta* representa a implementação de um comando concreto. Esta classe tem como objetivo atribuir um aluno a uma proposta. Nesta classe estão redefinidas as operações de *execute()* e *undo()*, com o objetivo de criar uma nova atribuição ou revertendo o comando, retirando essa atribuição.

### RemoveAtribuicaoAlunoProposta

A classe *RemoveAtribuicaoAlunoProposta* representa a implementação de um comando concreto. O objetivo desta classe é remover uma atribuição de um aluno a uma proposta. Para tal são redefinidas as operações de *execute()* e *undo()*, indo estas remover uma atribuição ou reverter este comando, voltando a adicionar a atribuição.

### AddOrientador

A classe *AddOrientador* representa a implementação de um comando concreto. Esta classe tem como objetivo adicionar a uma proposta atribuída um docente orientador. Para isso redefine as operações de *execute()* e *undo()*, onde estas vão adicionar um docente a uma atribuição que ainda não tenha docente, ou remover o docente orientador na operação de *undo()*.

### ChangeOrientador

A classe *ChangeOrientador* representa a implementação de um comando concreto. O objetivo desta classe é alterar o docente orientador a uma proposta atribuída que já tenha um orientador atribuído. Para tal redefine as operações de *execute()* e *undo()*, onde estas vão alterar o antigo docente orientador pelo novo especificado pelo utilizador, ou reverter esta operação colocando o antigo docente orientador.

### RemoveOrientador

A classe *RemoveOrientador* representa a implementação de um comando concreto. Esta classe tem como objetivo remover o docente orientador de uma proposta atribuída, ficando esta sem docente orientador. Para isso redefine as operações de *execute()* e *undo()*, onde estas vão remover o atual docente orientador, ou revertendo a operação colocando o antigo docente orientador.

## pt.isec.pa.apoio\_poe.model.data

### Pessoa

A classe *Pessoa* representa uma classe abstrata com as informações básicas de uma pessoa no contexto da aplicação. Tem como objetivo servir de classe mãe para as classes *Docente* e *Aluno*.

### Docente

A classe *Docente* representa toda a informação necessária sobre os docentes no contexto desta aplicação. Para tal deriva da classe *Pessoa* de forma a utilizar as funcionalidades já por esta disponibilizadas.

### Aluno

A classe *Aluno* representa um aluno no contexto desta aplicação. Tem como objetivo guardar toda a informação relativa a um aluno. Para isso deriva da classe *Pessoa* com o objetivo de utilizar as funcionalidades já por esta disponibilizadas.

### AlunoClassificacaoComparator

A classe *AlunoClassificacaoComparator* representa um comparador de dois alunos pela classificação dos mesmos. Esta classe tem como objetivo ser utilizada para ordenar os alunos pela sua classificação, útil no momento de atribuição de propostas aos alunos de forma automática. Para tal implementa a interface *Comparator* e o seu método *compare()*.

### Proposta

A classe *Proposta* representa uma classe abstrata com as informações básicas de todas as propostas. Esta classe tem como objetivo servir de classe mãe para as classes *Projeto*, *Estagio*, *Autoproposto* e *PropostaAtribuida*.

### Projeto

A classe *Projeto* representa uma proposta do tipo projeto. Esta classe tem como objetivo guardar os dados referentes a uma proposta de projeto. Para tal deriva da classe *Proposta*, de forma a utilizar as funcionalidades por esta disponibilizadas.

### Estagio

A classe *Estagio* representa uma proposta do tipo estágio. Esta classe tem como objetivo guardar toda a informação referente a uma proposta de estágio. Para isso deriva da classe *Proposta*, com o objetivo de utilizar as funcionalidades que esta disponibiliza.

### Autoproposto

A classe *Autoproposto* representa uma proposta de estágio/projeto apresenta por um aluno. Para tal guarda todas as informações referentes a uma autoproposta de um aluno. Esta classe também deriva da classe *Proposta*, utilizando também as funcionalidades que esta disponibiliza.

### PropostaAtribuida

A classe *PropostaAtribuida* representa a atribuição de uma proposta a um aluno e qual o docente orientador atribuído a esta proposta. Esta classe tem como objetivo guardar toda a informação necessária para representar a atribuição de uma proposta a um aluno e qual o docente orientador para esta proposta.

### Candidatura

A classe *Candidatura* tem como objetivo representar a candidatura de um aluno a uma ou várias propostas. Para isso guarda toda a informação necessária, nomeadamente o aluno candidato e as propostas a que este se está a candidatar.

### ApoioPoE

A classe *ApoioPoE* representa todos os dados do programa. Tem como objetivo guardar os vários alunos, docentes, propostas e candidaturas que são criados no decorrer da execução da aplicação. Disponibiliza também métodos que permitem manipular esta informação.

### ApoioPoEManager

A classe *ApoioPoEManager* representa um *proxy* para a classe *ApoioPoE*. Tem como objetivo fazer a gestão de um objeto *ApoioPoE* disponibilizando métodos semelhantes aos da classe *ApoioPoE* redirecionando todo o processamento para a classe *ApoioPoE*. Nesta classe é também implementado o padrão *Decorator*, onde são disponibilizadas novas funcionalidades à classe *ApoioPoE* sem que esta sofra alterações à sua estrutura, nomeadamente a introdução de dados vinda de ficheiros, bem como a realização do *load* e *save* do estado e dados da aplicação num determinado ponto da sua execução.

## pt.isec.pa.apoio\_poe.model.exceptionsHandling

### ExceptionsTypes

A classe *ExceptionTypes* representa uma classe enumerável. Esta classe tem como objetivo representar as diversas possíveis exceções que podem decorrer durante a execução da aplicação.

### ExceptionOccurred

A classe *ExceptionOccurred* representa uma classe segundo o padrão *Singleton*. Esta classe tem como objetivo representar uma exceção quando esta ocorre, de forma que a interface com o utilizador possa apresentar uma mensagem personalizada conforme o tipo de exceção que tenha ocorrido.

## pt.isec.pa.apoio\_poe.model.facade

### GestaoManualAtribuicoesManager

A classe *GestaoManulaAtribuicoesManager* representa uma classe segundo o padrão *Facade*. Esta classe tem como objetivo fornecer uma interface mais simples para as classes que vão utilizar as funcionalidades que esta esconde, neste caso a criação e remoção de atribuições de propostas a alunos, realizada através do padrão *Command*.

### GestaoManualOrientadoresManager

A classe *GestaoManualOrientadoresManager* representa uma classe segundo o padrão *Facade*. Esta classe tem como objetivo esconder funcionalidades mais complexas, de forma a criar uma interface mais simples para as classes que utilizem estas funcionalidades. Esta classe refere-se a associação, alteração e desassociação de um docente orientador a uma proposta atribuída, implementados segundo o padrão *Command*.

## pt.isec.pa.apoio\_poe.model.fsm

### ApoioPoEState

A classe *ApoioPoEState* representa todos os uma classe enumerável com todos os estados possíveis. Esta classe tem ainda o objetivo de criar instâncias das classes que representam os diversos estados, usando para isso o padrão *Factory*.

### IApoioPoEState

A classe *IApoioPoEState* representa uma interface base para todos os estados da aplicação. Esta interface tem como objetivo criar todos os métodos a serem implementar pelos diversos estados, sendo estes métodos representativos da transição entre estados e também das operações que podem ser realizadas em cada estado.

### ApoioPoEContext

A classe *ApoioPoEContext* representa a classe que será utilizada pelas diversas interfaces com o utilizador para estas receberem e alterarem os dados. Esta classe tem como objetivo garantir que as alterações aos dados são apenas realizadas no contexto de um estado, não possibilitando a alteração direta dos dados.

### ApoioPoEAdapter

A classe *ApoioPoEAdapter* representa as implementações por omissão dos métodos definidos na interface *IApoioPoeState*. Esta classe tem como objetivo criar implementações por omissão, desta forma os diversos estados apenas terão de redefinir os métodos que utilizem. Fornece ainda um método que permite a alterar o estado atual no contexto.

### InicioState

A classe *InicioState* representa a implementação de um estado. Esta classe tem como objetivo dar a possibilidade ao utilizador de criar uma nova execução com os dados vazios, ou então carregar um estado guardado anteriormente.

### Fase1State

A classe *Fase1State* representa a implementação de um estado. Esta classe tem como objetivo permitir ao utilizar alternar entre os modos de gestão de alunos, docentes ou propostas.

### Fase1BloqueadaState

A classe *Fase1BloqueadaState* representa a implementação de um estado. Esta classe tem como objetivo fornecer ao utilizar a possibilidade de consultar e exportar os dados dos alunos, docentes e propostas após a primeira fase ser fechada a novas alterações.

### Fase2State

A classe *Fase2State* representa a implementação de um estado. Esta classe tem como objetivo possibilitar ao utilizar a gestão de candidaturas, bem como consultar dados sobre os alunos e propostas. Podendo ainda regressar à primeira fase e avançar para a terceira fase.

### Fase2BloqueadaState

A classe *Fase2BloqueadaState* representa a implementação de um estado. Esta classe tem como objetivo permitir ao utilizador consultar os dados das candidaturas e poder exportar estes dados, após a segunda fase ter sido fechada a novas alterações.

### Fase3State

A classe *Fase3State* representa a implementação de um estado. Esta classe tem como objetivo proporcionar ao utilizador três diferentes mecanismos para atribuir propostas ao utilizador, dois de forma automática, tendo um deste uma resolução de conflitos entre alunos, e ainda uma outra forma de gerir as atribuições de propostas a alunos de forma manual. Permite ainda obter dados sobre os alunos e sobre as várias propostas.

### Fase3MasFase2AbertaState

A classe *Fase3MasFase2AbertaState* representa a implementação de um estado. Esta classe tem como objetivo apenas fornecer uma forma de atribuir propostas aos alunos, sendo esta totalmente automática, limitando assim as possibilidades do utilizador, devido à segunda fase ainda se encontrar aberta a alterações, podendo ser alteradas as candidaturas.

### Fase3BloqueadaState

A classe *Fase3BloqueadaState* representa a implementação de um estado. Esta classe tem como objetivo apresentar dados sobre os alunos e propostas ao utilizador após a terceira fase ter sido fechada a alterações. Possibilita ainda a exportação de dados sobre os alunos, docente, propostas, candidaturas e propostas atribuídas.

### Fase3AtribuicaoAutomaticaState

A classe *Fase3AtribuicaoAutomaticaState* representa a implementação de um estado. Esta classe tem como objetivo a atribuição automática de alunos a propostas segundo a sua classificação e as suas opções de candidatura. Tem ainda o objetivo de permitir ao utilizador a resolução de conflitos entre dois utilizadores.

### Fase4State

A classe *Fase4State* representa a implementação de um estado. Esta classe tem como objetivo possibilitar ao utilizador duas formas de atribuir docentes orientadores a propostas atribuídas, uma forma automática e uma forma manual. Permite ainda obter dados sobre os alunos e os docentes e exportar dados sobre os alunos, docentes, propostas, candidaturas e propostas atribuídas.

### Fase5State

A classe *Fase5State* representa a implementação de um estado. Esta classe tem como objetivo apresentar informação sobre os alunos, propostas e docentes. Permitindo também exportar dados sobre os alunos, docentes, propostas, candidaturas e propostas atribuídas.

### GestaoAlunosState

A classe *GestaoAlunosState* representa a implementação de um estado. Esta classe tem como objetivo possibilitar ao utilizador a gestão manual de alunos, bem como importar e exportar alunos através de ficheiros *csv*.

### GestaoDocentesState

A classe *GestaoDocentesState* representa a implementação de um estado. Esta classe tem como objetivo permitir ao utilizador gerir manualmente os docentes, bem como a possibilidade de importar ou exportar docentes para ficheiros *csv*.

### GestaoPropostasState

A classe *GestaoPropostasState* representa a implementação de um estado. Esta classe tem como objetivo fornecer ao utilizador uma forma de gerir as propostas de forma manual, e ainda possibilitar que este importe ou exporte os dados das propostas para ficheiros *csv*.

### GestaoCandidaturasState

A classe *GestaoCandidaturasState* representa a implementação de um estado. Esta classe tem como objetivo fornecer ao utilizador uma forma de gerir as candidaturas manualmente, bem como permitir que o utilizador importe ou exporte candidaturas para ficheiros *csv*.

### GestaoManualAtribuicoesState

A classe *GestaoManualAtribuicoesState* representa a implementação de um estado. Esta classe tem como objetivo permitir ao utilizador que este gira as atribuições de alunos a propostas de forma manual. O utilizador tem ainda a possibilitar de reverter ou refazer as alterações aos dados que realizou através de comandos *undo* e *redo*.

### GestaoManualOrientadoresState

A classe *GestaoManualOrientadoresState* representa a implementação de um estado. Esta classe tem como objetivo possibilitar que o utilizador faça a gestão manual da atribuição de docentes orientador a propostas que estejam atribuídas a alunos. O utilizador pode ainda reverter ou refazer as alterações que realizou através de comandos *undo* e *redo*.

## pt.isec.pa.apoio\_poe.ui.text

### ApoioPoEUI

A classe *ApoioPoEUI* representa a interface com o utilizador em modo texto. Esta classe tem como objetivo permitir ao utilizador interagir com a aplicação permitindo a alteração de dados através dos vários estados e métodos criados.

## pt.isec.pa.apoio\_poe.utils

### PAInput

A classe *PAInput* representa uma forma mais simples de pedir dados ao utilizador em formato de texto. Esta classe tem como objetivo fornecer métodos e verificações aos dados inseridos pelo utilizador. Esta classe foi fornecida nas aulas práticas, sendo que foi adicionado um método que permite ao utilizador não inserir nada, útil para a consulta de dados, quando o utilizador pretende ver todos os dados e não apenas de um aluno/docente/proposta/candidatura especifica.

## pt.isec.pa.apoio\_poe

### Main

A classe *Main* representa o início da aplicação. Esta classe tem como objetivo criar a máquina de estados e fornecer esta às várias interfaces que se pretendam executar.

# Relacionamento entre Classes

asd

# Funcionalidades Implementadas

Tabela 1 - Funcionalidades Fase 1

|  |  |  |
| --- | --- | --- |
| Funcionalidade | Implementado | Razão |
| Alternar entre modos de gestão de alunos, docentes e propostas de estágio | Totalmente implementado |  |
| Inserção, Consulta, Edição e Eliminação de dados referentes a alunos | Implementado parcialmente | Apenas foi implementada a consulta de dados, sendo esta a única requerida para a primeira meta do trabalho prático. |
| Inserção, Consulta, Edição e Eliminação de dados referentes a docentes | Implementado parcialmente | Apenas foi implementada a consulta de dados, sendo esta a única requerida para a primeira meta do trabalho prático. |
| Inserção, Consulta, Edição e Eliminação de dados referentes a propostas | Implementado parcialmente | Apenas foi implementada a consulta de dados, sendo esta a única requerida para a primeira meta do trabalho prático. |
| Eliminar todos os dados de alunos, docentes ou propostas | Totalmente implementado |  |
| Importação de dados de alunos, docentes ou propostas a partir de ficheiros *csv* | Totalmente implementado |  |
| Exportação de dados de alunos, docentes ou propostas para ficheiros *csv* | Totalmente implementado |  |
| Fechar a fase | Totalmente implementado |  |
| Avançar para a fase seguinte | Totalmente implementado |  |

Tabela 2 - Funcionalidades Fase 2

|  |  |  |
| --- | --- | --- |
| Funcionalidade | Implementado | Razão |
| Inserção, Consulta, Edição e Eliminação de dados referentes a candidaturas | Implementado parcialmente | Apenas foi implementada a consulta de dados, sendo esta a única requerida para a primeira meta do trabalho prático. |
| Eliminar todos os dados de candidaturas | Totalmente implementado |  |
| Importação de dados de candidaturas a partir de ficheiros *csv* | Totalmente implementado |  |
| Exportação de dados de candidaturas para ficheiros *csv* | Totalmente implementado |  |
| Obter lista de alunos | Totalmente implementado |  |
| Obter lista de propostas | Totalmente implementado |  |
| Fechar a fase | Totalmente implementado |  |
| Regressar à fase anterior | Totalmente implementado |  |
| Avançar para a fase seguinte | Totalmente implementado |  |

Tabela 3 - Funcionalidades Fase 3

|  |  |  |
| --- | --- | --- |
| Funcionalidade | Implementado | Razão |
| Atribuição automática das autopropostas ou propostas de docentes com aluno associado | Totalmente implementado |  |
| Atribuição automática de uma proposta disponível aos alunos sem atribuições definidas, com base nas suas classificações, restrição de acesso a estágios e opções  indicadas no processo de candidatura | Totalmente implementado |  |
| Atribuição manual de propostas disponíveis aos alunos sem atribuição ainda definida | Totalmente implementado |  |
| Remoção manual de uma atribuição previamente realizada ou de todas as atribuições | Totalmente implementado |  |
| Consultar propostas atribuídas | Totalmente implementado |  |
| Operações de *undo* e *redo* na gestão manual | Totalmente implementado |  |
| Importação de dados de candidaturas a partir de ficheiros *csv* | Totalmente implementado |  |
| Exportação de dados de alunos, docentes, propostas, candidaturas ou propostas atribuídas para ficheiros *csv* | Totalmente implementado |  |
| Obter lista de propostas | Totalmente implementado |  |
| Fechar a fase | Totalmente implementado |  |
| Regressar à fase anterior | Totalmente implementado |  |
| Avançar para a fase seguinte | Totalmente implementado |  |

Tabela 4 - Funcionalidades Fase 4

|  |  |  |
| --- | --- | --- |
| Funcionalidade | Implementado | Razão |
| Associação automática dos docentes proponentes de projetos como orientador dos  mesmos | Totalmente implementado |  |
| Atribuição, consulta, alteração e eliminação de um orientador aos alunos com propostas atribuídas | Totalmente implementado |  |
| Operações de *undo* e *redo* na gestão manual | Totalmente implementado |  |
| Exportação de dados de alunos, docentes, propostas, candidaturas ou propostas atribuídas para ficheiros *csv* | Totalmente implementado |  |
| Obtenção de dados diversos sobre atribuição de orientadores | Totalmente implementado |  |
| Fechar a fase | Totalmente implementado |  |
| Regressar à fase anterior | Totalmente implementado |  |
| Avançar para a fase seguinte | Totalmente implementado |  |

Tabela 5 - Funcionalidades Fase 5

|  |  |  |
| --- | --- | --- |
| Funcionalidade | Implementado | Razão |
| Lista de estudantes com propostas atribuídas | Totalmente implementado |  |
| Lista de estudantes sem propostas atribuídas e com opções de candidatura | Totalmente implementado |  |
| Conjunto de propostas disponíveis | Totalmente implementado |  |
| Conjunto de propostas atribuídas | Totalmente implementado |  |
| Número de orientações por docente, em média, mínimo, máximo, e por docente  especificado | Totalmente implementado |  |
| Outros dados | Não implementado | Não foram encontrados outros dados que se considerassem pertinentes mostrar ao utilizador nesta fase. |
| Exportação de dados de alunos, docentes, propostas, candidaturas ou propostas atribuídas para ficheiros csv | Totalmente implementado |  |

Tabela 6 - Outras Funcionalidades

|  |  |  |
| --- | --- | --- |
| Funcionalidade | Implementado | Razão |
| Gravação/carregamento do estado da aplicação usando um formato binário | Totalmente implementado |  |
| Interface em modo Texto | Implementado parcialmente | As interfaces relativas à inserção, consulta e remoção de alunos, docentes, propostas e candidaturas não foram implementadas, devido a estas funcionalidades não terem sido implementadas na primeira meta |
| Interface em modo Gráfico | Não implementado | Apenas requerido para a segunda meta do trabalho prático. |