RELATÓRIO

Trabalho Prático

Programação Avançada

Licenciatura em Engenharia Informática

Desenvolvimento de uma aplicação de apoio ao processo de gestão de projetos e estágios do Departamento de Engenharia Informática e de Sistemas do ISEC.

Trabalho realizado por:

Tomás Gomes Silva - 2020143845

Rafael Gerardo Couto - 2019142454

Índice

Índice **2**
Introdução **3**
Implementação **4**
Estruturas de Dados ..... **Erro! Marcador não definido.**
Diagrama da Máquina de Estados **5**
Classes **6**
Main 6
PoEAluno 6
PoEDocente 6
PoEProposta 6
PoEEstagio 6
PoEProjeto 6
PoEAutoproposto 7
PoECandidatura 7
PoEOrientador 7
PoEData 7
PoEContext 7
PoEStateAdapter 7
ConfigState 8
ApplicationOptState 8
PropAttributionState 8
OriAttributionState 8
ReviewState 8
PoEUI 8
PoEMenu 8
Relacionamento entre as classes 9
Funcionalidades Implementadas **10**

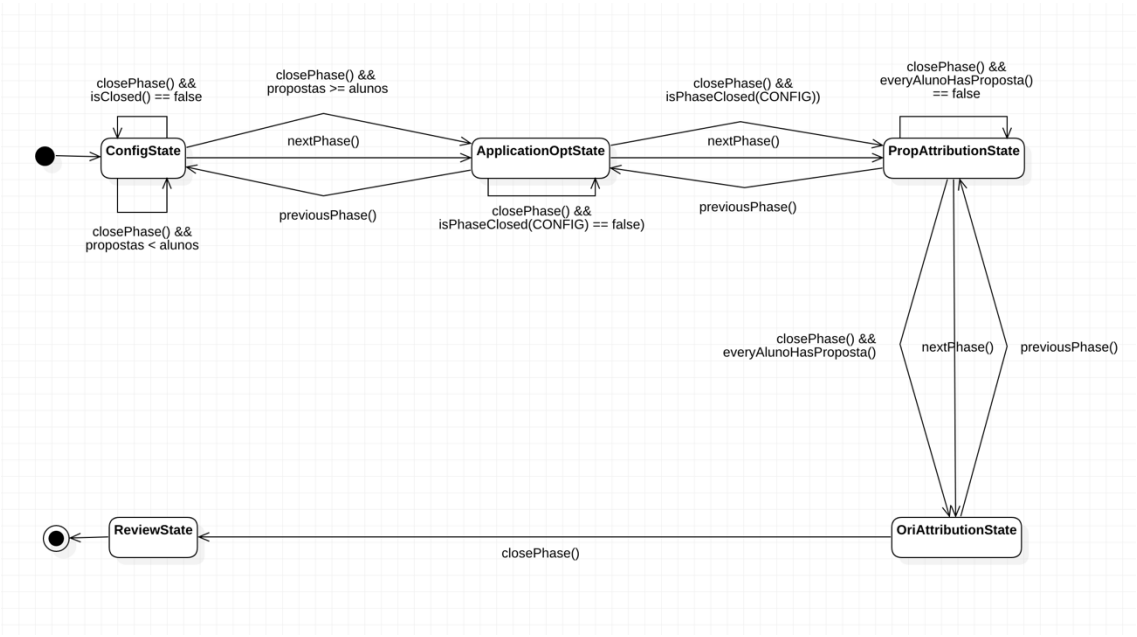
Introdução

O trabalho prático de Programação Avançada consiste na criação de uma aplicação, em Java, que sirva de apoio ao processo de gestão de estágios e projetos do Departamento de Engenharia Informática e Sistemas do ISEC.
A aplicação está dividida em várias fases em que o utilizador vai inserindo informação e o programa manipula essa mesma informação de modo a facilitar todo o processo de atribuição de estágios e projetos.
Na primeira meta esta aplicação conta com uma interface em modo de consola mas pretende-se implementar uma interface gráfica mais à frente com recurso a JavaFX.

Implementação

A aplicação que estamos a desenvolver baseia-se numa máquina de estados finita (FSM – Finite State Machine) e, portanto, começámos por implementar as classes que iam representar cada estado. Precisámos também de criar uma enumeração com os vários estados possíveis, um adaptador para as várias classes-estado, uma interface com os métodos que podem fazer com que a FSM mude de estado e um contexto para a máquina de estados que permitisse a interface interagir com a mesma.
De seguida, começámos a trabalhar na interface em modo de texto mas precisávamos de classes que guardassem os dados e que representassem as várias entidades existentes, tais como os alunos e os docentes, então foi mesmo isso que fizemos.
A classe de dados contém coleções (*ArrayList*) que guardam os objetos das várias classes que compõem e representam os dados e criámos métodos que pudessem obter e alterar esses valores.
Para a primeira fase da máquina de estados implementámos os métodos para ler um ficheiro CSV com os dados dos alunos, dos docentes e das propostas e criámos objetos de cada tipo à medida que íamos lendo cada linha do ficheiro. Na segunda fase fizemos o mesmo só que para as candidaturas.
A terceira fase foi a mais trabalhosa pois tivemos de implementar um algoritmo que atribuísse automaticamente as propostas aos vários alunos. A quarta fase foi essencialmente a mesma coisa só que em vez de atribuir propostas tivemos de atribuir orientadores de forma automática e também manual.
Quanto à quinta fase, esta foi bastante mais simples visto que apenas foi necessário implementar pesquisas simples relativas a todo o processo.
Para que fosse possível guardar o estado da aplicação quando esta fosse fechada fizemos com que todas as classes do modelo implementassem a interface *Serializable* e utilizámos métodos da classe *ObjectInputStream* e *ObjectOutputStream* para ler e escrever o contexto da máquina de estados.

Diagrama da Máquina de Estados



Classes

Main

Ponto de entrada da aplicação. Inicia a máquina de estados (FSM) e a interface com o utilizador (UI).

PoEAluno

Classe que representa um aluno inscrito na unidade de PoE.
Guarda as seguintes informações:

- Nome do aluno, número, email, curso, ramo, classificação, possibilidade de aceder a estágios, candidatura, proposta atribuída.

Esta classe possui métodos que permitem obter e modificar certos dados.

PoEDocente

Classe que representa um docente do ISEC.
Guarda as seguintes informações:

- Nome do docente, email, papel

Esta classe possui métodos que permitem obter e modificar certos dados.

PoEProposta

Classe que representa uma proposta existente na lista de propostas disponíveis para os alunos poderem escolher.
Guarda as seguintes informações:

- Número da proposta, título, número de aluno atribuído, candidaturas efetuadas para a proposta e orientador da proposta.

Esta classe possui métodos que permitem obter e modificar certos dados.

PoEEstagio

Classe derivada de **PoEProposta** que representa uma proposta de estágio.
Guarda as seguintes informações:

- Ramos de destino, entidade acolhedora

Esta classe possui métodos que permitem obter e modificar certos dados.

PoEProjeto

Classe derivada de **PoEProposta** que representa uma proposta de projeto.
Guarda as seguintes informações:

- Ramos de destino, docente proponente

Esta classe possui métodos que permitem obter e modificar certos dados.

PoEAutoproposto

Classe derivada de **PoEProposta** que representa uma autoproposta efetuada por um aluno.
Guarda as seguintes informações:

- Aluno proponente

Esta classe possui métodos que permitem obter e modificar certos dados.

PoECandidatura

Classe que representa uma candidatura efetuada por um aluno.
Guarda as seguintes informações:

- Número do estudante candidato, propostas de preferência

Esta classe possui métodos que permitem obter e modificar certos dados.

PoEOrientador

Classe que representa um orientador do ISEC, que é um docente na sua essência.
Guarda as seguintes informações:

- Docente orientador, propostas que o docente vai orientar

Esta classe possui métodos que permitem obter e modificar certos dados.

PoEData

Classe que representa e armazena todos os dados existentes no programa.
Guarda as seguintes informações:

- Alunos inscritos, docentes do ISEC, propostas, candidaturas efetuadas, orientadores do ISEC e fases da máquina de estados que se encontram fechadas

Esta classe possui métodos que permitem obter e modificar certos dados. Existem métodos "extra" que permitem obter um objeto de uma certa classe que respeitem uma certa condição.

PoEContext

Classe que representa o contexto da máquina de estados e serve de ligação entre o modelo e a interface com o utilizador.
Guarda as seguintes informações:

- Estado da máquina de estados, dados da aplicação

Esta classe possui métodos que permitem alterar o estado da máquina de estados bem como outros métodos que permitem manipular os dados existentes.

PoEStateAdapter

Classe abstrata que implementa a interface IPoEState e que contém os vários métodos que a máquina de estados possui que podem fazer alterar o estado da mesma
Guarda as seguintes informações:

- Contexto da máquina de estados, dados da aplicação

Esta classe contém um método que permite alterar o estado da máquina de estados para um estado em específico.

ConfigState

Classe derivada de **PoEStateAdapter** que representa o primeiro estado da máquina de estados (fase de configuração).
Esta classe possui métodos que permitem alterar o estado da máquina de estados bem como outros métodos que permitem manipular os dados existentes.

ApplicationOptState

Classe derivada de **PoEStateAdapter** que representa o segundo estado da máquina de estados (fase de opções de candidaturas).
Esta classe possui métodos que permitem alterar o estado da máquina de estados bem como outros métodos que permitem manipular os dados existentes.

PropAttributionState

Classe derivada de **PoEStateAdapter** que representa o terceiro estado da máquina de estados (fase de atribuição de propostas).
Esta classe possui métodos que permitem alterar o estado da máquina de estados bem como outros métodos que permitem manipular os dados existentes.

OriAttributionState

Classe derivada de **PoEStateAdapter** que representa o quarto estado da máquina de estados (fase de atribuição de orientadores).
Esta classe possui métodos que permitem alterar o estado da máquina de estados bem como outros métodos que permitem manipular os dados existentes.

ReviewState

Classe derivada de **PoEStateAdapter** que representa o último estado da máquina de estados (fase de consulta).
Esta classe possui métodos que permitem alterar o estado da máquina de estados bem como outros métodos que permitem manipular os dados existentes.

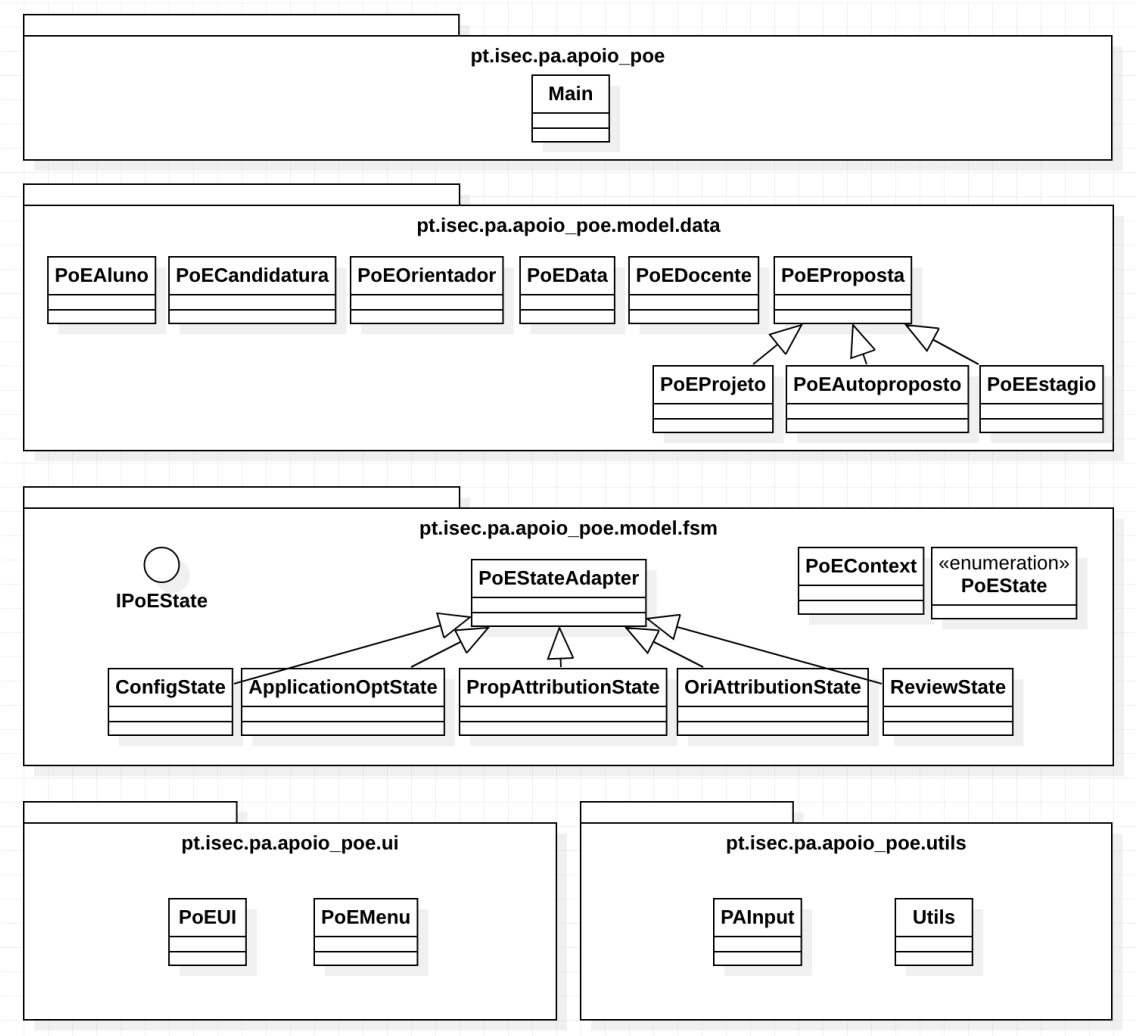
PoEUI

Classe que disponibiliza métodos que permitem ao utilizador interagir com o contexto da máquina de estados e consequentemente o resto das funcionalidades que estão disponíveis em cada fase.

PoEMenu

Classe que disponibiliza métodos estáticos que o a interface de texto pode utilizar para mostrar menus e sub-menus.

Relacionamento entre as classes



Funcionalidades Implementadas

Todas as funcionalidades relativas à primeira meta foram implementadas exceto a consulta de alunos com autoproposta associada e consulta de propostas de alunos autopropostos.
De resto, as 5 fases da máquina de estados encontram-se implementadas, bem como guardar e retomar o estado da aplicação.