Instituto Superior de Engenharia de Coimbra

Licenciatura em Engenharia Informática

Programação Avançada - 2021/2022

Relatório do Trabalho Prático:

Fase 2

Turma Prática Nº1

Beatriz Pinto | 2020144095

Eduardo Correia | 2020139576

Índice

1. Descrição sintética acerca das opções e decisões tomadas na implementação

A aplicação apresenta diversas classes que foram colocadas nos respetivos packages, de acordo com as suas funções específicas.

No package data, encontram-se todos os dados relativos ao sistema de gestão de projetos e estágios do Departamento de Engenharia Informática e de Sistemas do ISEC, presentes na classe Sistem, classe Students, classe Teachers, classe Proposals e classes que derivam de Proposals temos as classes Stages, Projects e SelfProposalProject .

Por sua vez, no package fsm, existe uma interface ISistemState que implementa os diversos métodos a serem executados nas diferentes fases. A classe abstrata SistemStateAdapter, vai implementar a interface ISistemState, e tem uma referência para dados do sistema, de modo que os diferentes estados consigam aceder a essa informação. Para além dessa classe abstrata encontram-se também as classes de cada estado possível e uma classe SistemContext que permite redirecionar o processamento.

No package memento, encontram-se a interface e as classes que dizem respeito ao processo de gravação, carregamento, undo e redo.

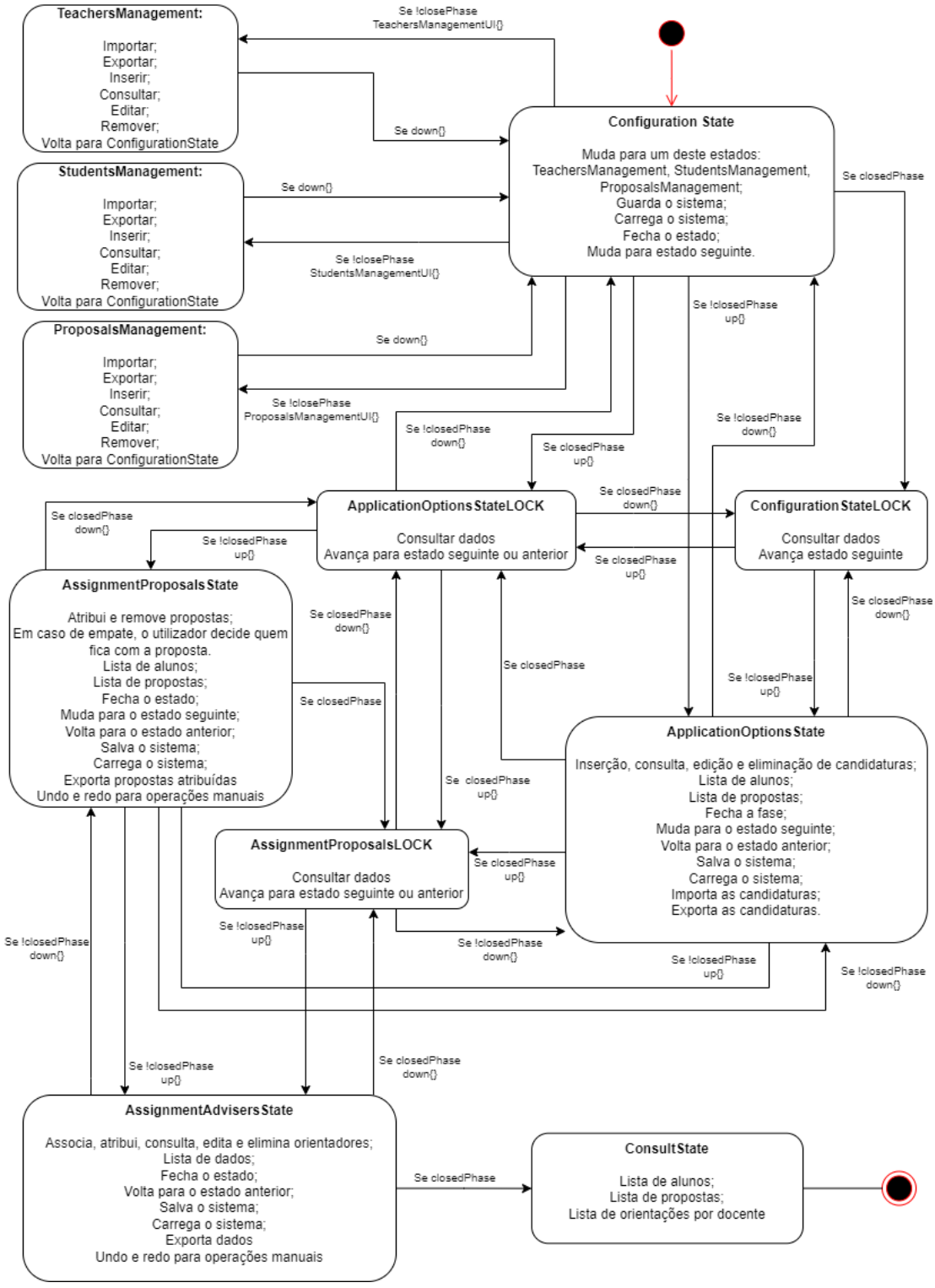
Ainda dentro do package model foi implementado uma classe SistemManager que possui o SistemContext e um PropertyChangeSupport, que é usado para lançar um FirePropertyChange para informar sobre as atualizações que ocorreram no sistema.

Na package ui, onde se encontra a package text, estão presentes todas as classes responsáveis pela interação com o utilizador.

Ainda na package ui, tenho a package gui que armazena todos as classes responsáveis pela parte gráfica do trabalho. Como por exemplo, os diversos estados mas desta vez implementado com ajuda do JavaFX.

Em relação aos estados em si, e de acordo com a máquina de estados que implementamos, achamos necessária a criação de 11 estados indispensáveis para a implementação da lógica do jogo, estes vão ser explicados no próximo tópico.

2. Diagrama da máquina de estados



A máquina de estados vai começar por definir o estado atual, como sendo o estado **ConfigurationState**. Neste estado é possível alternar entre os modos de gestão de alunos (**studentsManagement**), docentes

(**teachersManagement**) e propostas (**proposalsManagement**), ainda neste estado é possível alterar para o estado seguinte, **ApplicationOptionsState**, que é responsável pelas gestão candidaturas dos alunos. Avançando para o estado seguinte, **AssignmentProposalsState**, que faz a atribuição das propostas. O quarto estado, **AssignmentAdvisersState**, é responsável pela atribuição de orientadores. No último estado, **ConsultState**, e não sendo possível regressar a nenhum dos estados anteriores, este é responsável por fornecer a consulta dos dados.

3. Descrição das classes utilizadas no programa

Package pt.isec.pa.apoio_PoE

Classe Main: É responsável por lançar toda a aplicação.

Package apoio_PoE.model:

Classe SistemaManager: Classe que fazendo o redirecionando para os respetivos métodos, dispara também um alerta sobre as atualizações feitas no sistema.

Package apoio_PoE.model.data:

Classe Projects: Tal como a classe Mãe, classe Proposals, permite a criação de propostas mas que por sua vez são projetos.

Classe Proposals: Esta classe vai servir para a criação de propostas, com os seus respetivos dados, fornecendo também gets e sets para serem usados na classe Sistem. Ainda nesta classe, foi criado um arrayList de propostas, onde é possível guardar as que forem criadas.

Classe SelfProposedProject: Tal como a classe mãe, classe Proposals ,esta classe permite a criação de estágios/projetos autopropostos.

Classe Sistem: Contém todos os dados relativos ao sistema de gestão de estágios e que vão permitir o seu funcionamento. É onde está presente a lógica e onde são verificadas todas as condições relativas ao correto funcionamento do sistema.

Classe Stages: Tal como a classe Mãe, classe Proposals, esta permite a criação de um tipo de propostas que são os estágios.

Classe Students: Esta classe vai servir para a criação de estudantes, com os seus respetivos dados, fornecendo também gets e sets para serem usados na classe Sistem. Ainda nesta classe, foi criado um arrayList de estudantes, para ser possível guardar os estudantes criados.

Classe Teachers: Esta classe vai servir para a criação de professores, com os seus respetivos dados, fornecendo também gets e sets para serem usados na classe Sistem. Ainda nesta classe, foi criado um arrayList de professores, para ser possível guardar os que forem criados.

Classe GuardaCarrega: Classe que contém os métodos que permitem guardar e carregar o jogo.

Package apoio_PoE.model.fsm:

Classe ApplicationOptionsLOCK: Classe que corresponde ao segundo estado mas que por estar fechada apenas permite a consulta dos dados.

Classe ApplicationOptionsState: Classe que corresponde ao segundo estado e que é responsável por gerir as candidaturas.

Classe AssignmentAdvisersState: Classe que corresponde ao quarto estado e que é responsável pela atribuição de orientadores.

Classe AssignmentProposalsLOCK: Classe que corresponde ao terceiro estado mas que por estar fechada apenas permite a consulta dos dados.

Classe AssignmentProposalsState: Classe que corresponde ao terceiro estado e que é responsável pela atribuição de propostas aos alunos.

Classe ConfigurationState: Classe que corresponde ao primeiro estado e que é responsável por alterar entre os modos de gestão de estudantes, docentes e propostas.

Classe ConfigurationStateLOCK: Classe que corresponde ao primeiro estado mas que por estar fechada apenas permite a consulta dos dados.

Classe ConsultState: Classe que corresponde ao quinto e último estado e disponibiliza a consulta dos dados.

Interface ISistemState: A interface contém os métodos que permitem a transições entre estados.

Classe ProposalsManagement: Classe que é responsável pela gestão das propostas existentes.

Classe SistemContext: Classe responsável por redirecionar o processamento para os respetivos métodos.

Classe SistemState: Classe responsável por criar os estados possíveis.

Classe SistemStateAdapter: Implementa as funções chamadas na Interface ISistemState e tem uma referência para o sistema, responsável por manipular os dados.

Classe StudentsManagement: Classe que é responsável pela gestão dos estudantes.

Classe TeachersManagement: Classe que é responsável pela gestão dos docentes.

Package apoio_PoE.model.memento:

Classe Memento: Classe que permite guardar o estado para que mais tarde possa ser reposto. Através da criação de snapshots do estado e mantendo um histórico do mesmo, será possível implementar operações de undo e redo.

Classe CareTaker: É a partir desta classe que são despoletadas as operações de criação de memento, mantendo um histórico dos mesmos.

Interface IMementoOriginator: Interface que permite representar/guardar os snapshots de um determinado momento.

Package apoio_PoE.model.ui.gui

Classe ApplicationOptionsLOCKUI: Classe responsável pela parte gráfica da fase ApplicationOptionsLOCK, disponibilizando de forma interativa para o utilizador, todas as funções disponibilizadas por esta fase, respetivas à consulta de dados.

Classe ApplicationOptionsUI: Classe responsável pela parte gráfica da fase ApplicationOptions, disponibilizando de forma interativa para o utilizador, todas as funções disponibilizadas por esta fase.

Classe AssignmentAdvisersUI: Classe responsável pela parte gráfica do estado AssignmentAdvisers, disponibilizando de forma interativa para o utilizador, todas as funções disponibilizadas por esta fase.

Classe AssignmentProposalsLOCKUI: Classe responsável pela parte gráfica do estado AssignementProposalsLOCK, disponibilizando de forma interativa para o utilizador, todas as funções disponibilizadas por esta fase, respetivas à consulta de dados.

Classe AssignmentProposalsUI: Classe responsável pela parte gráfica da fase AssignmentProposals, disponibilizando de forma interativa para o utilizador, todas as funções disponibilizadas por esta fase.

Classe ConfigurationLOCKUI: Classe responsável pela parte gráfica da fase ConfigurationLOCK, disponibilizando de forma interativa para o utilizador, todas as funções disponibilizadas por esta fase, respetivas à consulta de dados.

Classe ConfigurationStateUI: Classe responsável pela parte gráfica da fase ConfigurationState, disponibilizando de forma interativa para o utilizador, todas as funções disponibilizadas por esta fase.

Classe ConsultStateUI: Classe responsável pela parte gráfica da fase ConsultState, disponibilizando de forma interativa para o utilizador, todas as funções disponibilizadas por esta fase.

Classe MainJFX: Classe que é responsável pela criação da parte gráfica.

Classe ProposalsManagementUI: Classe responsável pela parte gráfica da fase ProposalsManagement, disponibilizando de forma interativa para o utilizador, todas as funções disponibilizadas por esta fase.

Classe RootPane: Classe responsável por criar a janela que vai ser apresentada ao utilizador de acordo com o estado em que este se encontra.

Classe StudentsManagementUI: Classe responsável pela parte gráfica da fase StudentsManagement, disponibilizando de forma interativa para o utilizador, todas as funções disponibilizadas por esta fase.

Classe TeachersManagementUI: Classe responsável pela parte gráfica da fase TeachersManagement, disponibilizando de forma interativa para o utilizador, todas as funções disponibilizadas por esta fase.

Package apoio_PoE.model.ui.gui.resources

Classe CSSManager: Classe responsável pelo estilos.

Classe ImageManager: Classe que é responsável pelo estilo, assim como o ficheiro styles.css.

Package apoio_PoE.model.ui.gui.resources.util:

Classe ToastMessage: Classe existe para lançar um pop-up, servindo como uma notificação para utilizador.

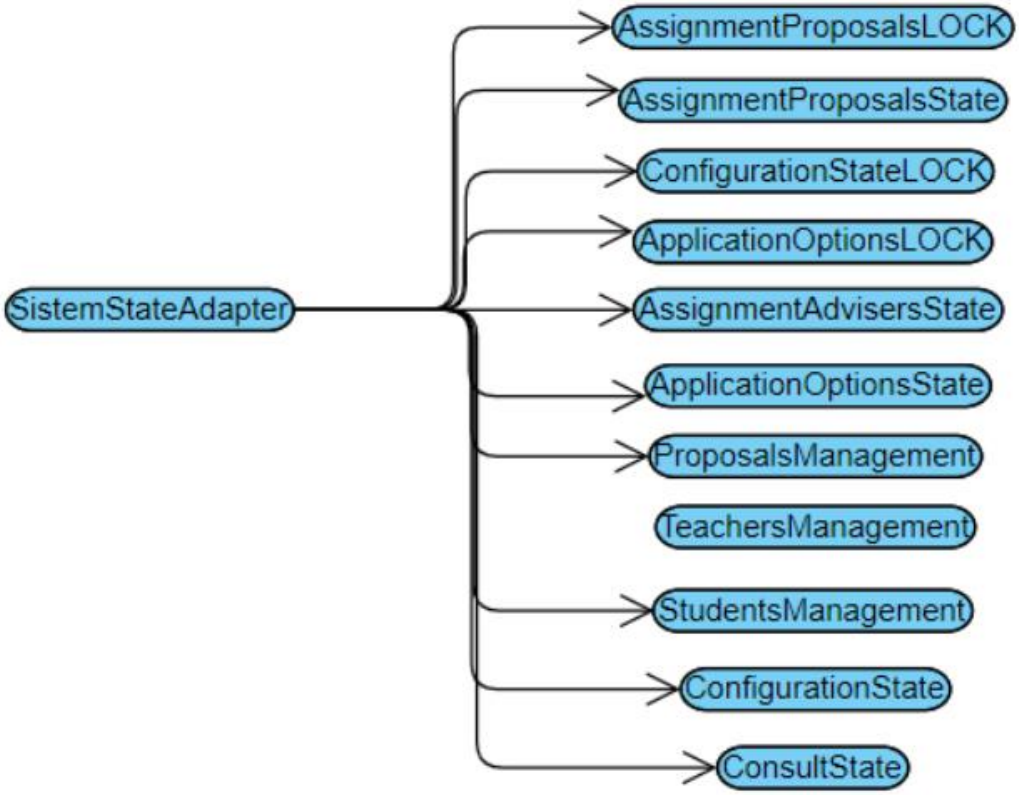
Package apoio_PoE.model.ui.text

Classe SistemUI: Classe responsável pela interação com o utilizador.

Package apoio_PoE.model.ui.text.utils

Classe PAInput: Classe que fornece ajuda no que diz respeito à iteração com o utilizador.

4. Descrição do relacionamento entre as classes



A classe SistemStateAdapter implementa a interface ISistemState, definindo implementações vazias para os métodos da mesma. As classes que representam estados são extends desta classe.

5. Funcionalidades implementadas

Funcionalidades	**Implementada**	**Implementada parcialmente**	**Não implementada**
Fase 1	X		
Fase 2	X		
Fase 3	X		
Fase 4	X		
Fase 5	X		
Importação Dados	X		
Exportação Dados	X		
Interface Modo Texto	X		
Gravação/Carregamento	X		
Interface Gráfica	X		
Undo e Redo	X		