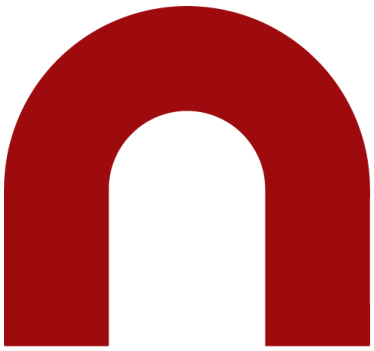


Instituto Superior de Engenharia

Politécnico de Coimbra

Programação Avançada

Licenciatura em Engenharia Informática

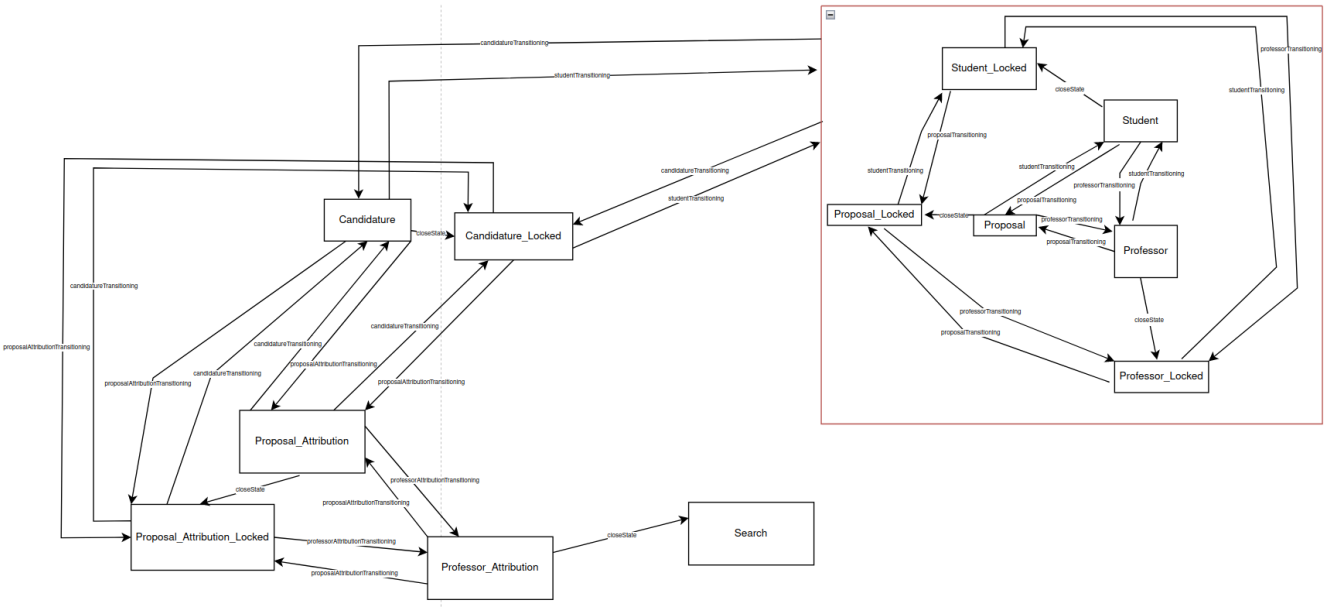
Rafael Gil - 2020136741
Hugo Ferreira - 2020128305

Decisões tomadas

Foram tomadas algumas decisões quanto à implementação do que era pedido no enunciado:

- Apenas é possível atribuir propostas a alunos que pertençam ao ramo desse mesmo aluno (Ex: uma proposta de DA apenas pode ser atribuída a um aluno de DA).
- O programa começa sempre no estado Student.
- A interface gráfica é composta por 3 janelas distintas:
    - POI_MANAGEMENT que é a janela principal, onde se realizam as ações da aplicação
    - DATA que é onde os dados vão ser mostrados
    - LOGGER que é onde vão ser apresentadas as mensagens de erro

Máquina de estados - FSM



A máquina de estados implementada é composta por 12 estados distintos, por os quais se pode transitar durante o programa, havendo algumas restrições quanto às transições entre alguns dos estados:

- **Student** → estado onde se dá toda a gestão relacionada com os alunos. Permite inserir, apagar, editar e consultar os dados referentes aos alunos. Permite transitar para os estados Professor, Proposal, Candidature, Student_Locked, Professor_Locked e Proposal_Locked.
- **Professor** → estado onde se dá toda a gestão relacionada com os professores. Permite inserir, apagar, editar e consultar os dados referentes aos professores. Permite transitar para os estados Student, Proposal, Candidature, Student_Locked, Professor_Locked e Proposal_Locked.

- **Proposal** → estado onde se dá toda a gestão relacionada com as propostas. Permite inserir, apagar, editar e consultar os dados referentes às propostas. Permite transitar para os estados Student, Professor, Candidature, Student_Locked, Professor_Locked e Proposal_Locked.
- **Student_Locked** → este estado representa um "bloqueio" do estado Student, ao qual não é possível regressar uma vez neste estado. Apenas é possível consultar os dados referentes aos alunos. Permite transitar para Professor_Locked, Proposal_Locked, Candidature e Candidature_Locked.
- **Professor_Locked** → este estado representa um "bloqueio" do estado Professor, ao qual não é possível regressar uma vez neste estado. Apenas é possível consultar os dados referentes aos professores. Permite transitar para Student_Locked, Proposal_Locked, Candidature e Candidature_Locked.
- **Proposal_Locked** → este estado representa um "bloqueio" do estado Proposal, ao qual não é possível regressar uma vez neste estado. Apenas é possível consultar os dados referentes às propostas. Permite transitar para Professor_Locked, Student_Locked, Candidature e Candidature_Locked.
- **Candidature** → estado onde se dá toda a gestão relacionada com as candidaturas. Permite inserir, apagar, editar e consultar os dados referentes às candidaturas e ainda obter listagens de alunos. Permite transitar para os estados Student, Student_Locked, Candidature_Locked, Proposal_Attribution e Proposal_Attribution_Locked.
- **Candidature_Locked** → este estado representa um "bloqueio" do estado Candidature, ao qual não é possível regressar uma vez neste estado. Apenas é possível consultar os dados referentes às candidaturas e as listagens de alunos. Permite transitar para Student_Locked, Proposal_Attribution e Proposal_Attribution_Locked.
- **Proposal_Attribution** → estado onde se processa as atribuições de alunos a propostas. É possível fazer atribuições automáticas (alunos associados ou com autopropostas) ou manuais, remover atribuições e obtenção de listagens de alunos e de propostas. Permite transitar para Candidature, Professor_Attribution, Candidature_Locked e Proposal_Attribution_Locked.
- **Proposal_Attribution_Locked** → este estado representa um "bloqueio" do estado Proposal_Attribution, ao qual não é possível regressar uma vez neste estado. Apenas é possível obter as listagens de alunos e de propostas. Permite transitar para Candidature_Locked, Professor.
- **Professor_Attribution** → estado onde se processa as atribuições de professores a propostas. É possível fazer atribuições automáticas (professores associados) ou manuais, remover atribuições e obtenção de listagens de professores. Permite transitar para Professor_Attribution, Proposal_Attribution_Locked e Search.
- **Search** → estado referente a consulta de dados. É possível obter listagens de alunos, propostas. Permite transitar para Candidature, Professor_Attribution, Candidature_Locked e Proposal_Attribution_Locked.

Classes implementadas

Package pt.isec.pa.apoi_poe.model.data

AutoProposal

Representa uma proposta do tipo autoproposta. Estende a classe Proposal. Implementa a interface Serializable.

Internship

Representa uma proposta do tipo estágio. Contém uma String para armazenar o nome da empresa onde se irá realizar. Estende a classe MidProposal. Implementa a interface Serializable.

Project

Representa uma proposta do tipo projeto. Contém uma referência para um Professor, para armazenar o professor responsável pelo projeto. Estende a classe MidProposal. Implementa a interface Serializable.

MidProposal

Classe abstrata que armazena valores em comum das classes Project e Internship (lista com os ramos a que pertencem). Estende a classe Proposal. Implementa a interface Serializable.

Proposal

Classe abstrata que armazena todos os atributos em comum entre AutoProposal e MidProposal (id, título e o estudante da proposta). Implementa a interface Serializable.

Student

Classe que representa um aluno. Contém o id, o curso e o ramo a que pertence, a sua classificação, se pode participar em estágios e se já tem uma proposta atribuída. Estende a classe Person. Implementa a interface Serializable.

Professor

Classe que representa um professor. Contém um valor booleano para indicar se pode ser orientador de projeto. Implementa a interface Serializable.

Person

Classe abstrata que armazena os atributos em comum das classes Student e Professor (nome e email). Implementa a interface Serializable.

Data

Esta classe é a responsável por guardar e realizar todas as operações sobre os dados do programa. Contém dois hashset de Person para guardar os alunos e os professores, um hashset de Proposal para guardar as autopropostas, dois hashsets de MidProposal para guardar os estágios e os projetos, um hashmap onde a key é o id do aluno e o value uma lista de strings correspondentes ao id de propostas para guardar as candidaturas, um hashmap onde a key corresponde aos enumerados definidos em ApplicationState e o value é um valor booleano para guardar quais dos estados se encontram bloqueados, um hashmap onde a key é o id de uma proposta e o value o id de um aluno para guardar as propostas já atribuídas e o aluno a que foi atribuída, um hashmap onde a key é o id do professor e o valor uma lista de ids de propostas para guardar as propostas a que o professor foi atribuído, uma referência para ApplicationState para guardar qual o estado atual e, por fim, um arraylist de strings para servir de log, para guardar os erros gerados na leitura da informação a partir dos ficheiros CSV.

Package pt.isec.pa.apoi_poe.model.fsm

ApplicationContext

Esta classe é responsável pelo controlo da máquina de estados, armazenando uma referência para Data e para IApplicationState, tendo conhecimento dos dados e do estado atual. É com esta classe que a CommandLineUI vai comunicar.

StateAdapter

É a classe que vai implementar os métodos da máquina de estados, definidos na interface. Todas as classes que representam os estados da máquina de estados, vão estender esta classe.

Package pt.isec.pa.apoi_poe.model

FSManager

Classe que representa a *facade* de interação entre entre as classes do package model e as restantes classes do programa.

Package pt.isec.pa.apoi_poe.model.command

CommandManager

Classe que coordena o padrão *Command.*

AddProfAttrib

Classe que representa o comando de adicionar uma atribuição de professores.

AddPropAttrib

Classe que representa o comando de adicionar uma atribuição de proposta.

RemoveProfAttrib

Classe que representa o comando de remover uma atribuição de professor.

RemovePropAttrib

Classe que representa o comando de remover uma atribuição de proposta.

Package pt.isec.pa.apoi_poe.ui.text

CommandLineUI

É a classe responsável pela interação com o utilizador em forma de texto. Todas as ações que podem ser realizadas por esta classe, vão ser redirecionadas para a classe ApplicationContext, a qual vai ser responsável por devolver o resultado de qualquer ação à CommandLineUI, que por sua vez vai apresentar esse resultado ao utilizador.

Package pt.isec.pa.apoi_poe.ui.gui

JFXMain

Classe responsável por iniciar a interface gráfica e criar as janelas desta.

```
@Override
public void start(Stage stage) throws Exception {
    RootPane root = new RootPane(manager);
    Scene scene = new Scene(root, v: 600, v1: 450, Color.INDIGO);
    stage.setScene(scene);
    stage.setTitle("POI_MANAGEMENT");
    stage.setMinWidth(500);
    stage.setMinHeight(150);
    stage.setResizable(false);
    stage.show();

    Stage stage1 = new Stage();
    RootPane2 root2 = new RootPane2(manager);
    Scene scene1 = new Scene(root2, v: 400, v1: 210, Color.INDIGO);
    stage1.setScene(scene1);
    stage1.setX(stage.getX()+stage.getWidth());
    stage1.setY(stage.getY());
    stage1.setTitle("Data");
    stage1.setResizable(false);
    stage1.show();

    Stage stage2 = new Stage();
    LoggerUI loggerUI = new LoggerUI(manager);
    Scene scene2 = new Scene(loggerUI, v: 400, v1: 210, Color.INDIGO);
    stage2.setScene(scene2);
    stage2.setX(stage.getX()+stage.getWidth());
    stage2.setY(stage1.getY()+stage1.getHeight());
    stage2.setTitle("Logger");
    stage2.setResizable(false);

    stage2.show();
}
```

Package pt.isec.pa.apoi_poe.ui.gui.components

RootPane

Classe que representa a janela principal da interface gráfica. É composta por um StackPane, que por sua vez vai ser composta por várias outras cenas (classes), uma por cada estado da máquina de estados, e que vai alternando a visibilidade destas, consoante o estado em que se encontra.

```
private void createViews() {
    StackPane stackPane = new StackPane(new StudentUI(manager), new StudentLockedUI(manager),
            new ProfessorUI(manager), new ProfessorLockedUI(manager),
            new ProposalUI(manager), new ProposalLockedUI(manager),
            new CandidatureUI(manager), new CandidatureLockedUI(manager),
            new ProposalsAttributionUI(manager), new ProposalsAttributionLockedUI(manager),
            new ProfessorAttributionUI(manager), new SearchUI(manager),
            new TieUI(manager));

    this.setCenter(stackPane);

    this.menu = new Menu( s: "File");
    this.menuItem1 = new MenuItem( s: "Save");
    this.menuItem2 = new MenuItem( s: "Load");
    this.menu.getItems().addAll(menuItem1, menuItem2);
    MenuBar menuBar = new MenuBar(menu);

    VBox vBox2 = new VBox(menuBar);
    vBox2.setSpacing(10);
    this.setTop(vBox2);
}
```

RootPane2

Classe que representa a janela onde os dados vão ser apresentados. À semelhança da classe RootPane, também é constituída por um StackPane que é composto por várias cenas, cada uma correspondente a um estado da máquina de estados, em que cada uma dessas cenas é responsável por apresentar os dados evocados no respectivo estado.

```
private void createViews() {
    StackPane stackPane = new StackPane(new StudentDataUI(manager), new ProfessorDataUI(manager), new ProposalDataUI(manager),
                                        new CandidatureDataUI(manager), new ProposalsAttributionDataUI(manager), new ProfessorAttributionDataUI(manager),
                                        new SearchDataUI(manager));

    this.setCenter(stackPane);
}
```

LoggerUI

Classe que representa a janela onde vão ser apresentadas as mensagens de erro (logger). Esta janela apenas é atualizada quando uma nova mensagem de erro surge.

```
private void createViews(){
    this.setStyle("-fx-background-color: white");

    this.lbState = new Label();
    this.setTop(lbState);

    this.lvLogger = new ListView<>();
    this.setCenter(lvLogger);
}

private void registerHandlers(){
    manager.addPropertyChangeListener(FSManager.PROP_DATA, evt -> {
        update();
    });
}

private void update(){
    this.lvLogger.setItems(FXCollections.observableList(manager.getLogger()));
    this.lbState.setText("Errors in: " + manager.getState());
}
```

Classes adicionais

Existem mais uma série de classes neste package, que são as constituintes dos StackPanes mencionados anteriormente. Cada uma delas é responsável por construir uma cena adaptada às funcionalidades que cada estado permite.

Objetivos cumpridos

Meta 1

- Máquina de estados implementada
- Leitura de informação a partir de ficheiros CSV
- Remoção de dados
- Edição de dados
- Implementação das funcionalidades de atribuição de propostas e professores
- Exportação da informação para ficheiros CSV
- Serialização e Deserialização de toda o programa

Meta 2

- Undo
- Redo
- Interface gráfica 100% funcional
- Aplicação de diversos padrões