UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA

Gestão de propostas de formação avançada no sistema Fenix

Francisco Daniel Eleutério Caeiro

Mestrado em Engenharia Informática
Especialização em Arquitetura, Sistemas e Redes de Computadores

Trabalho de Projeto orientado por:
Prof.a Doutor Maria Dulce Pedroso Domingos
Prof. Doutor Mário João Barata Calha

2021

Agradecimentos

Em primeiro lugar quero agradecer à Faculdade de Ciências e à Reitoria da Universidade de Lisboa pela oportunidade da realização do mestrado. Aos meus orientadores, a professora Dulce Domingos e o professor Mário Calha, por toda a ajuda, orientação dada e disponibilidade ao longo do trabalho. Aos meus supervisores, Ana Catarina Silva e Ricardo Rito. A todo o Departamento de Informática da Reitoria, nomeadamente à Ana Rute, Daniela Mendes, Daniel Vitoriano, Fábio Ferreira, Nuno Heitor, Nuno Jesus, Pedro Moita, André Filipe, José Lima e Tânia Castelo.

Um agradecimento à Quorum Born IT (QubIT), em particular ao Nadir Amin pelo tempo disponibilizado e por todas as dúvidas esclarecidas.

Aos colegas de mestrado que me acompanharam e ajudaram, Ana Espinheira, Diana Marques, Diogo Godinho, Mafalda Luz e Pedro Ladino.

Aos meus amigos de faculdade por toda a ajuda ao longo dos anos, Alexandre Chicharo, António Estriga, Bruno Andrade, Diogo Ferreira, Ricardo Cruz e Tomás Cunha.

Um obrigado à Olga Simões por toda a ajuda e motivação dada.

Quero agradecer a toda a minha família por todo apoio, paciência e sacrifícios, em especial aos meus pais, Henrique Caeiro e Ana Eleutério.

Por fim, um agradecimento especial à Mariana Luz, que me apoiou nos momentos mais difíceis e por tudo o que me ajudou a crescer.

"*There is no problem so bad that you can't make it worse*"
- Chris Hadfield

Resumo

A gestão dos processos iniciais de trabalhos de formação avançada relativos à etapa final dos cursos de 1º, 2º e 3º Ciclo consiste na criação de proponentes, criação de propostas por parte dos proponentes, aprovação das propostas, escolha por parte dos alunos das propostas que têm interesse e por fim a atribuição de uma proposta a um aluno.

Esta gestão inicial é frequentemente feita de forma manual e recorre muitas das vezes aos serviços académicos de cada escola. Na data da escrita do documento não existe um apoio informático que auxilie a gestão deste processo no sistema Fenix. Além disto a gestão manual dos processos é bastante demorada, mesmo para as escolas mais pequenas, e pode em certos casos levar a atrasos no início dos trabalhos de formação avançada.

Para combater esta dificuldade, algumas escolas e departamentos, nomeadamente o Departamento de Informática da Faculdade de Ciências da Universidade de Lisboa (DI-FCUL) criou um sistema próprio (PEIpal/CuCo) que permite uma informatização deste processo. No entanto, este sistema apenas soluciona o problema relativo às especificidades do seu departamento.

As escolas da Universidade de Lisboa usam um sistema de gestão académica designado por Fenix. Para que exista uma ferramenta de gestão dos processos iniciais acima descritos comum a todas as escolas da Universidade de Lisboa (ULisboa), faz sentido que esta seja oferecida como mais uma peça do sistema Fenix que resolva este problema.

Neste projeto foi desenvolvido, para o sistema Fenix, um modelo de processo de gestão de propostas de formação avançada e de um processo de gestão de candidaturas às propostas, utilizado como base os procedimentos em uso em algumas faculdades. Para suportar este modelo foi desenvolvido um módulo que permite a gestão de propostas e candidaturas. Este módulo permite que cada escola, departamento, curso ou outro componente da estrutura orgânica da Universidade de Lisboa tenha a capacidade de criar os seus próprios modelos de processo (*workflow*) de maneira a satisfazer as suas necessidades académicas.

Palavras-chave: Formação avançada, Fenix, *Workflow*, Propostas, Universidade de Lisboa

Abstract

The management of the initial processes of the advanced formation work related to the final stage of the 1st, 2nd, and 3rd cycle courses consists in the creation of proponents, the creation of proposals by said proponents, the approval of said proposals, the student choice of the proposals in which they have interest, and finally the assignment of a proposal to a student.

This initial management process is often done manually and relies on the academic services of each school, without computing support to assist them. Furthermore, this manual process management is quite time-consuming, even for smaller schools, and in certain cases can lead to delays at the beginning of the advanced formation work.

To overcome this difficulty, some schools and departments, namely the Informatics Department of the Faculty of Science of the University of Lisbon, created their own system (PEIpal/CuCo) which allows for the computerization of this process. However, this system only solves the problem related to the specifications of this department.

The schools at the University of Lisbon use an academic management system called Fenix. To allow the managing of the previously described processes by a common mechanism, it makes sense to offer a new tool capable of solving this problem as part of the Fenix system. As such, a model was developed for the management of proposals for advanced formation work, a model for the management process for the candidacy of said proposals, based on the procedures in use in some schools. To support both models, a module that allows the management of proposals and candidacies was developed. This module allows each school, department, course, or other component of the University of Lisbon's organic structure, to have the ability to create their own process models (workflow) in order to meet their academic needs.

Keywords: Advanced work, Fenix, Workflow, Proposals, University of Lisbon

Conteúdo

Lista de Figuras

Lista de excertos de código

Capítulo 1

Introdução

O Sistema Integrado de Gestão Académica (SIGA) Fenix [26] é a plataforma informática que dá suporte à gestão académica das escolas da Universidade de Lisboa (ULisboa) [15]. Este sistema está implementado, a partir do ano letivo 2020/2021, nas 18 escolas, incluindo os serviços centrais da reitoria da ULisboa.

Com este trabalho pretende-se que o sistema Fenix dê o suporte necessário para o processo inicial de formação avançada, nomeadamente na gestão de propostas e de candidaturas às mesmas.

Outras teses relativas ao sistema Fenix podem ser encontradas aqui [11, 20, 23, 25].

1.1 Motivação

A gestão dos processos iniciais de trabalhos de formação avançada, relativos à etapa final dos cursos do 1º, 2º e 3º ciclos, consiste na criação e validação das propostas e candidaturas às mesmas. Este processo inclui a criação e gestão de proponentes (pessoas que podem criar propostas), a criação de propostas (por parte dos proponentes), a aprovação das propostas por parte das pessoas responsáveis de cada âmbito (curso ou unidade curricular), a candidatura por parte dos alunos às propostas do seu interesse e a atribuição final de uma proposta a cada aluno que dará origem a um trabalho de formação avançada.

Todo o procedimento anterior é realizado maioritariamente de forma manual e recorre frequentemente aos serviços académicos de cada escola. Devido à complexidade e diferenças nos processos entre as escolas da ULisboa, e entre os vários departamentos da mesma escola, não existe ainda um recurso informático que facilite a gestão destes processos. Esta lacuna torna todo o procedimento muito mais trabalhoso e demorado, chegando a ter consequências ao nível do início dos próprios trabalhos de formação avançada por parte dos alunos.

O Departamento de Informática da Faculdade de Ciências da Universidade de Lisboa (DI-FCUL) combateu esta dificuldade ao criar um sistema (PEIpal/CuCo) que permite uma informatização deste processo. Este sistema apenas soluciona o problema das espe-

cificidades do seu departamento.

Para garantir uma coerência entre todas as escolas na gestão destes processos, e evitar que os vários sistemas externos diferentes façam esta gestão de maneira diferente, faz sentido que exista um desenvolvimento de um módulo no sistema Fenix, que dê apoio à gestão dos processos de maneira a ser utilizado pelas escolas.

Neste trabalho, o conceito de 'modalidades' incluí os vários tipos de trabalho de formação avançada: dissertação, estágio ou projeto, entre outros. O conceito de 'títulos' incluí os títulos (títulos concretos) e temas (área mais abrangente) das propostas.

1.2 Objetivos

Este projeto tem como objetivo a criação de um módulo que permita a realização de uma gestão informatizada dos processos de formação avançada no sistema Fenix. Este módulo tem de cumprir os requisitos comuns entre as escolas, assim como alguns requisitos mais específicos. Os principais requisitos a satisfazer são:

- Criação de propostas

  Permitir que as pessoas criem propostas de formação avançada.

- Validação de propostas

  Permitir que as propostas sejam validadas pelas pessoas encarregues.

- Escolha de candidatos

  Permitir que as pessoas que criaram as propostas escolham quais os candidatos para realizar as propostas.

- Gestão de proponentes

  Gerir as pessoas que podem criar propostas de formação. Esta gestão inclui a criação dos proponentes e a gestão das entidades a que estão associados.

- Criação de candidaturas

  Permitir que os alunos criem as suas candidaturas às propostas de formação avançada.

- Escolha de propostas por parte dos alunos

  Permitir que os candidatos escolham um conjunto de propostas que tenham interesse em realizar.

1.3 Contribuições

As contribuições deste projeto a destacar são:

- Um modelo de processo de gestão de propostas de formação avançada e de um processo de gestão de candidaturas às propostas, utilizado como base os procedimentos em uso em algumas faculdades.
- Um módulo para o sistema Fenix que permite a gestão de propostas e candidaturas na formação avançada para as escolas da Universidade de Lisboa.
- Cada escola, departamento, curso ou outro componente da estrutura orgânica da Universidade de Lisboa tem a capacidade de criar os seus próprios modelos de processo (*workflow*) de maneira a satisfazer as suas necessidades académicas.

1.4 Estrutura do documento

O documento tem a seguinte estrutura:

Capítulo 1 - Introdução

O primeiro capítulo apresenta a introdução do documento. Esta introdução inclui a motivação do trabalho, os objetivos, a contribuição e a estrutura do documento.

Capítulo 2 - Contexto: Sistema Fenix

Este capítulo explica o Sistema Integrado de Gestão Académica (SIGA) Fenix e algumas das suas tecnologias, *frameworks* e ferramentas. Esta explicação inclui a *framework* OMNIS e o motor de execução de processos (motor de *workflow*).

Capítulo 3 - Levantamento de requisitos

Este capítulo explica o levantamento de requisitos feito com as escolas que participaram nesta fase, Faculdade de Ciências da Universidade de Lisboa (FCUL), Departamento de Informática da Faculdade de Ciências da Universidade de Lisboa (DI-FCUL), Faculdade de Farmácia da Universidade de Lisboa (FFUL), Faculdade de Direito da Universidade de Lisboa (FDUL) e o Instituto de Geografia e Ordenamento do Território da Universidade de Lisboa (IGOT-UL).

O capítulo aborda também o sistema de gestão de propostas de formação avançada PEIpal/CuCo usado pelo DI-FCUL.

Capítulo 4 - Análise e Desenho

Este capítulo apresenta a análise de requisitos e os desenhos do módulo. Esta análise inclui quais os utilizadores intervenientes no módulo, os casos de uso e os requisitos não funcionais. Os desenhos dividem-se em três categorias, o diagrama entidade associação, o desenho do modelo de gestão de propostas e o desenho do modelo do processo de candidatura às propostas.

Capítulo 5 - Desenvolvimento e Testes

Este capítulo explica os problemas e as soluções encontradas com a integração com o *workflow* e a flexibilidade do módulo, o diagrama de classes, o desenvolvimento do módulo com ênfase na DSL e PSL e novos componentes criados.

Capítulo 6 - Conclusão e trabalho futuro

Este capítulo final apresenta as conclusões do trabalho realizado.

Capítulo 2

Contexto: Sistema Fenix

Neste capítulo é apresentado o Sistema Integrado de Gestão Académica (SIGA) Fenix e algumas das ferramentas, tecnologias e *frameworks* utilizadas, com especial foco na *framework* OMNIS e no motor de execução de processos (motor de *Workflow*). Apesar do sistema Fenix ter muito mais componentes, estes dois, em particular, são essenciais para a realização deste projeto.

2.1 Introdução

O SIGA Fenix foi desenvolvido no Instituto Superior Técnico (IST) para realizar a gestão académica da escola, e mais tarde foi implementado nas dezoito (18) escolas que pertencem à ULisboa, incluindo a reitoria da ULisboa. Este sistema permite fazer a gestão dos procedimentos académicos que cada escola necessita para o seu funcionamento. Estes procedimentos incluem a gestão de processos de candidaturas dos ciclos existentes, gestão de Unidades Curriculares (UCs), docentes, alunos, colaboradores, períodos letivos, pagamento de propinas, lançamento de notas, marcação de exames, entre outros.

2.2 Tecnologias e Frameworks

O sistema Fenix utiliza várias tecnologias e *frameworks* para o desenvolvimento das suas funcionalidades. Estas tecnologias incluem linguagens de programação, compiladores de código, servidores, bases de dados, entre outros. As *frameworks* servem de suporte no desenvolvimento do sistema com funcionalidades de apoio aos programadores ou de gestão de segurança do sistema.

Existe ainda duas ferramentas presentes no sistema Fenix, a OMNIsBar e o motor *workflow*. Estas duas ferramentas serão explicadas em mais detalhes nas secções seguintes.

2.2.1 Tecnologias

O sistema Fenix está desenvolvido na linguagem de programação *Java* [13]. É utilizado o *Maven* [17] para a compilação e gestão das classes do projeto. O *TomCat* [19] é utilizado como servidor *Java* para correr instâncias locais dos ambientes de desenvolvimento. O sistema Fenix utiliza como base de dados o *MySQL* [14]. Por fim, o serviço *Git* [27] serve como o controlador de versões do código desenvolvido. As versões apresentadas são as versões mais recentes utilizadas no desenvolvimento, sendo possível que tenham sido utilizadas versões mais recentes ou antigas para o desenvolvimento de outras funcionalidades.

2.2.2 Frameworks

O sistema Fenix utiliza três *frameworks*: a *framework* Fenix, a *framework* Bennu e a *framework* OMNIS.

A *framework* Fenix é a mais antiga do sistema e é a base do mesmo. Tem uma linguagem de domínio própria, a *Domain Model Language (DML)*. Esta linguagem é onde são descritos os objetos, os seus atributos e as relações de multiplicidade com outros objetos. Esta *framework* é ainda responsável pelo modelo transacional e do modelo de persistência.

A *framework* Bennu é focada em ferramentas de autenticação, escalonamento e das aplicações web. É com esta *framework* que é possível fazer login de utilizadores com credenciais válidas, permitir que as instâncias dos servidores consigam suportar a carga de pedidos e é ainda responsável por tratar de pedidos de *SOAP* e *Rest*.

Por fim, a *framework* OMNIS é a *framework* mais recente para o desenvolvimento de módulos e funcionalidade no sistema Fenix. Esta *framework* permite o desenvolvimento da camada de domínio e da camada de apresentação, através de duas linguagens: a *Domain Specific Language* (DSL) e a *Presentation Specific Language* (PSL), respetivamente. A DSL é responsável pela modelação do domínio dos objetos, dos atributos dos mesmos e das relações com outros objetos. A PSL é responsável pela modelação dos ecrãs de apresentação. Esta baseia-se no padrão de desenho *Create, Read, Update, Delete* (CRUD) para a gestão dos ecrãs no sistema. É possível determinar ainda ações e eventos dos ecrãs criados para uma melhor gestão dos objetos de domínio. Por outras palavras, é com o auxílio desta *framework* que são criados automaticamente, com a compilação do ficheiro da PSL, os ecrãs de procura, criação, visualização dos detalhes e edição dos objetos que correspondem aos objetos de domínio criados pela DSL.

Domain Specific Language (DSL)

A *Domain Specific Language* (DSL) é a linguagem utilizada na modelação da camada de domínio da *framework* OMNIS. Esta linguagem permite que qualquer pessoa consiga

criar as classes para os seus próprios objetos no sistema Fenix, sem conhecer como funciona a sua geração interna das mesmas, feita pela *Domain Model Language* (DML). Esta linguagem permite ainda que o programador organize as suas classes de domínio num único ficheiro, com uma estrutura pré-definida.

Na estrutura da DSL é possível definir a localização (*path*) das classes nas pastas do projeto, os nomes das classes, o tipo e nome dos atributos, as relações entre outras classes de domínio e possíveis heranças (extensões) de outras classes. É na DSL onde se criam enumerados, caso sejam necessários, e onde se associa a utilização de classes próprias do sistema, como é o caso da *LocalizedString*. Esta classe é um tipo de dados que dá suporte à língua de apresentação no sistema. É um mapeamento de *String* e língua (Português ou Inglês) para se apresentar dinamicamente os valores corretos das *Strings* de acordo com a linguagem de apresentação escolhida.

É apresentado um exemplo de uma classe da DSL na *Figura 2.1*.

```
enum com.qubit.solution.fenixedu.module.demo.domain.ulawp.enums.ExampleEnum as ExempleEnum;

valueType com.qubit.terra.framework.tools.primitives.LocalizedString as ExampleLocalizedString;

reserved entity pt.ist.fenixframework.DomainRoot channels (WebJava) {
    oneToMany ExamplePerson examplePersons mappedBy domainRoot;
    oneToMany ExampleContact exampleContacts mappedBy domainRoot;
}

reserved entity org.fenixedu.academic.domain.CurricularCourse channels ( WebJava ) {
    oneToMany ExamplePerson examplePerson mappedBy courses;
}

reserved entity org.fenixedu.academic.domain.Person channels ( WebJava ) {
    oneToOne ExamplePerson examplePerson mappedBy person;
}

entity demo.contact.ExampleContact channels (WebJava) {
    manyToOne DomainRoot domainRoot;
    ExampleLocalizedString contactDescription;
    manyToOne ExamplePerson person;
    ExampleEnum contactEnum;
}
entity demo.person.ExamplePerson channels (WebJava) {
    manyToOne DomainRoot domainRoot;
    String name;
    Boolean isStudent;
    Integer age;
    DateTime birthdate;
    oneToMany ExampleContact contacts mappedBy person;
    oneToOne Person person;
    manyToMany (CurricularCourseExamplePerson) CurricularCourse courses;
}
```

Figura 2.1: Exemplo DSL

Para designar a localização de cada classe, basta colocar o caminho desejado separado por pontos finais. Este caminho faz com que sejam criadas pastas e sub-pastas até ao nome da nova classe, e por fim é criada a classe em si.

Para garantir que as novas classes criadas não tenham o mesmo nome que outras já existentes no domínio do Fenix, é utilizado um prefixo para as designar. Este prefixo é, por regra geral, uma abreviatura ou sigla do módulo desenvolvido. No caso deste projeto

a sigla utilizada é ***Ulawp*** (Universidade de Lisboa Advanced Work Proposals). Caso se utilize uma classe já existente, utiliza-se a etiqueta "*reserved entity*", seguido do caminho desta entidade. Se o caso for a criação de uma nova classe, utiliza-se apenas a etiqueta "*entity*".

As relações entre as classes seguem o padrão utilizado em relações direcionais presentes em bases de dados: de um para um (***oneToOne***), um para muitos (***oneToMany***), muitos para um (***manyToOne***) ou muitos para muitos (manyToMany). Na relação de muitos para muitos é necessário ainda inserir uma especificação do nome da relação entre as classes de maneira a evitar colisões na base de dados.

Os atributos das classes podem ser de tipos primitivos (Boolean, Integers, Char) ou não primitivos (Strings, DateTime ou outras classes do mesmo módulo, ou já existentes no sistema) mas não é permitido utilizar listas ou arrays na declaração dos atributos. Para ultrapassar isto, são utilizadas as relações de multiplicidade acima descritas. No entanto, nada impede que dentro das classes dos objetos não possam existir estas estruturas de dados para a gestão e manipulação dos objetos.

Para a explicação da criação das classes DSL irá ser apresentado a título de exemplo a classe **ExamplePerson** apresentada na *Figura 2.1*. A geração das classes é feita em duas fases. Na primeira fase são criadas as classe base (ExamplePerson_ Base.java) e na segunda parte são criadas as classes individuais (ExamplePerson.java). As classes individuais estendem sempre a sua classe base. As classes individuais são responsáveis pela gestão dos objetos. Por outras palavras, é nestas classes que se escreve a totalidade do código que irá ser necessário para a gestão dos objetos desta classe. Esta gestão inclui a criação dos objetos, edição e remoção do objeto, a gestão de atributos, regras de negócio ou até mesmo a gestão de métodos auxiliares da classe. As classes base é onde estão os métodos para a gestão básica, como *getters* e *setters*, os métodos para adição ou remoção de objetos de relações de multiplicidade, a ligação ao objeto ***domainRoot*** do sistema, eliminação do objeto do sistema, entre outros.

A *framework* Fenix, a cada compilação do código, verifica se as classes individuais já existem no projeto. Se já existirem, estas não sofrem qualquer alteração, mas caso não existam é criada uma classe nova de raiz. As classes base são sempre criadas de raiz a cada compilação do código, mesmo que não tenham havido alterações. Isto garante que a funcionalidade base está sempre de acordo com o que está na DSL.

Tomando a classe **ExamplePerson** apresenta na *Figura 2.1* é possível ver um exemplo da relação da classe individual e da respetiva classe base na *Figura 2.2*. Nesta figura existe a classe individual **ExamplePerson.java** que estende a sua classe base **ExamplePerson_ Base.java**. Apesar de não estarem apresentados os métodos, classe base tem os métodos de gestão de negócio, o construtor, os métodos de edição e remoção, entre outros. No entanto, a chamada destes métodos na classe individual invoca a chamada dos mesmos metodos na classe base que trata de toda a lógica, permitindo que o programador apenas

tenha de saber quais os métodos que quer invocar e não tenha de programar os métodos mais básicos.

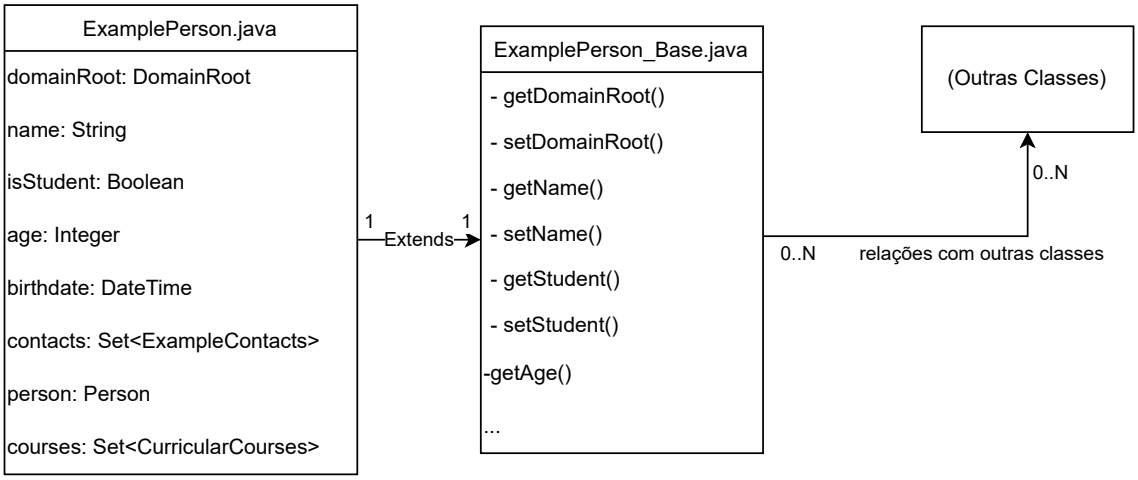

Figura 2.2: Classes da DSL

Presentation Specific Language (PSL)

A *Presentation Specific Language* (PSL) é a linguagem de modelação utilizada para a gestão da camada de apresentação da *framework* OMNIS. Esta linguagem permite que qualquer programador consiga criar os seus próprios ecrãs e fluxos de apresentação para visualizar os objetos criados pela DSL no sistema Fenix, sem conhecer o funcionamento interno da camada de apresentação do sistema.

Esta linguagem permite ainda que o programador organize os seus ecrãs de apresentação num único ficheiro, com uma estrutura única. Nesta estrutura, à semelhança com a DSL, é possível definir a localização dos ecrãs nas pastas do projeto, o nome do ecrã, o tipo de ecrã, os fluxos de apresentação, qual o tipo de objeto para que o ecrã será utilizado, os atributos dos objetos a serem apresentados no ecrã, assim como os nome das operações e dos eventos. É possível ver um exemplo de um ficheiro da PSL na *Figura 2.3*.

```
flow demo.examplePerson channel ( WebJava ) {
    searchScreen searchExamplePersons [ ExamplePerson ] (fields name isStudent age courses birthdate person contacts) (searchFields name isStudent age) {
        createEvent -> createExamplePerson
        selectMultipleEvent deleteMultiple -> searchExamplePersons
        dialog event addPersonalInformation -> addPersonalInformation
        viewAction -> readExamplePerson
        deleteAction -> searchExamplePersons
        action exampleAction
    }
    createScreen createExamplePerson [ ExamplePerson ] (fields name isStudent age courses birthdate person contacts) {
        createEvent -> searchExamplePersons
        cancelEvent -> searchExamplePersons
    }
    updateScreen updateExamplePerson [ ExamplePerson ] (fields name isStudent age courses birthdate person contacts) {
        updateEvent -> readExamplePerson
        cancelEvent -> readExamplePerson
    }
    readScreen readExamplePerson [ ExamplePerson ] (fields name isStudent age courses birthdate person contacts) {
        backEvent -> searchExamplePersons
        updateEvent -> updateExamplePerson
        deleteEvent -> searchExamplePersons
    }
    screen addPersonalInformation {
        event create
        event cancel
    }

}
```

Figura 2.3: Exemplo PSL

Para designar a localização do ecrã, basca colocar o caminho desejado separado por pontos finais. Este caminho faz com que sejam criadas pastas e sub-pastas até ao novo ecrã de apresentação e por fim é criada a classe do ecrã.

Existem cinco tipos de ecrãs: de pesquisa de objetos (*SearchScreen*), de criação de objetos (*CreateScreen*), de visualização dos detalhes do objeto (*ReadScreen*), de edição do objeto (*UpdateScreen*) e um ecrã genérico (*Screen*). Após o tipo segue-se o nome do ecrã, que irá ser o nome das classes geradas pela tecnologia OMNIS.

O tipo do objeto para qual o ecrã irá ser utilizado é colocado entre chavetas após o nome do ecrã (*searchExamplePersons* **[ExamplePerson]**). De seguida são apresentados os atributos do objeto que são necessários para o funcionamento do ecrã (*fields* ***name isStudent*** ...). Estes atributos têm de ter o mesmo nome que os atributos do objeto definidos na DSL. No caso de ser um ecrã de pesquisa é ainda necessário colocar os atributos pelos quais irá ser possível fazer a pesquisa (*searchFields* ***name isStudent*** ...). Estes atributos serão apresentados no ecrã seguindo as regras de formatação dos respetivos tipos de ecrãs, fazendo com que o programador apenas se preocupe com a ordem dos atributos e não com a estrutura de apresentação.

Por fim temos a definição do fluxo de ecrãs com o uso de eventos e ações. O uso de eventos cria botões para o propósito do evento, enquanto que o uso de ações cria um texto clicável com diferentes possibilidades de avanço no fluxo. Primeiro são definidos os eventos e depois as ações.

Para visualização do uso dos eventos e ações abaixo descritos, existe um exemplo de cada um na *Figura 2.3*.

A lista abaixo apresenta alguns tipos de eventos e uma breve descrição do seu uso:

- *CreateEvent*

  Utilizado para navegar para o ecrã de criação de objetos. Este evento é utilizado maioritariamente no ecrã de pesquisa para ir para o ecrã de criação ou no ecrã de

criação para confirmar a criação de um novo objeto.

- *UpdateEvent*

  Utilizado para navegar para o ecrã de edição de objetos. Este evento é utilizado no ecrã de leitura para ir para o ecrã de edição ou no ecrã de edição para guardar as alterações feitas ao objeto.

- *DeleteEvent*

  Utilizado para apagar um objeto que está a ser visualizado no ecrã de leitura. Este evento abre uma janela de confirmação para apagar o objeto.

- *BackEvent*

  Este evento retrocede para o ecrã precedente de navegação, ou seja, apenas retrocede para o ecrã de onde veio. Este evento é geralmente utilizado no ecrã de leitura para voltar ao ecrã de pesquisa.

- *CancelEvent*

  Este evento permite cancelar a criação ou edição de objetos. Isto permite voltar atrás e evitar criar ou editar objetos por engano.

- *Event*

  Este evento é um evento genérico utilizado para criar novos eventos. É necessário que todo o seu comportamento seja programado na classe em que está presente.

- *SelectMultipleEvent*

  Idêntico ao evento genérico, este permite a criação de um evento personalizado. No entanto, este evento permite selecionar vários objetos e realizar o comportamento programado em todos os objetos selecionados em vez de um a um. Isto pode ser utilizado para apagar vários objetos em bloco, em vez de os apagar um a um, ou associar vários objetos a outros, semelhante à população de listas em objetos.

A lista abaixo apresenta alguns tipos de ações e uma breve descrição do seu uso:

- *ViewAction*

  Esta ação permite visualizar os detalhes de um objeto. Existe um botão de "Detalhes" em cada objeto na grelha de resultados presente do ecrã pesquisa. Ao clicar neste texto, o fluxo avança para o ecrã de visualização do objeto selecionado.

- *DeleteAction*

  Esta ação permite apagar o objeto ao qual o evento está associado. Existe um botão de "Apagar" em cada objeto na grelha de resultados presente do ecrã pesquisa. Ao

clicar no texto, é pedida a confirmação de eliminação do objeto. Ao confirmar o objeto é apagado e a lista de resultados da pesquisa é atualizada.

- *Action*

  À semelhança do "*Event*", esta ação é uma ação genérica, caso seja necessário criar uma nova ação para além das existentes. Desta maneira, a sua programação tem de ser desenvolvida no ecrã associado para garantir o seu correto funcionamento.

Por norma os eventos e as ações realizam uma mudança no fluxo dos ecrãs. No entanto, é possível realizar todas as ações e eventos numa janela de "*popup*", conhecida como uma janela de diálogo. Para isto, basta colocar a etiqueta ***Dialog*** antes da ação ou evento para que tenha este comportamento.

No sistema Fenix é possível configurar quaisquer fluxos entre os ecrãs, contudo existem algumas regras a ter em conta.

- Só deve ser possível ir para o ecrã de visualização de detalhes a partir do ecrã de pesquisa;
- Só deve ser possível ir para o ecrã de criação a partir do ecrã de pesquisa;
- Só deve ser possível ir para o ecrã de edição a partir do ecrã de visualização.
- O ecrã principal dos objetos é o ecrã de pesquisa.

Estas regras representam o padrão normal de fluxo de navegação que é utilizado no Fenix. No entanto, é possível ter outros pontos de entrada, ou outras maneiras de estruturar o fluxo passando por outros ecrãs. Isto significa que o programador pode escolher qual o fluxo que faz mais sentido para o seu módulo, mesmo que não seja um fluxo normal. É possível ver um esquema do fluxo normal de ecrãs na *Figura 2.4*.

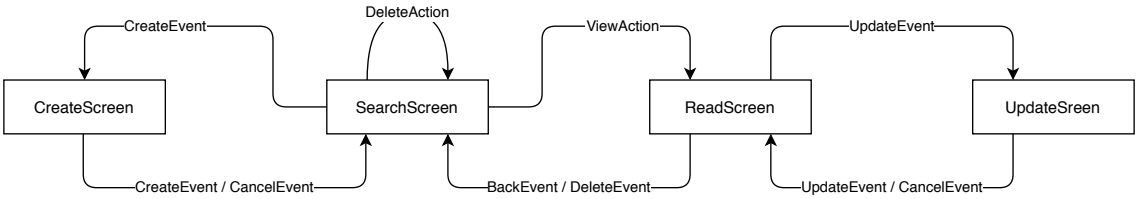

Figura 2.4: Fluxo geral da PSL

A criação das classes de apresentação é feita em duas fases, semelhantes às fases de criação das classes de domínio. Na primeira fase são criadas as classe base (SearchExample_ Base.java) e na segunda fase as classes individuais (SearchExample.java). As classes individuais estendem sempre a sua classe base que por sua vez estendem a classe que representa o tipo de ecrã: (*Search*, *Create* , *Read* ou *Update*).

As classes base dos ecrãs de apresentação tratam da criação e gestão de componentes visuais para os mesmos, como grelhas de resultados, botões, eventos, ações, caixas de diálogo (*popup*) ou os campos de texto dos objetos.

A *framework* OMNIS, a cada compilação do código, verifica se as classes individuais já existem no projeto. Se não existirem é criada uma classe nova de raiz, se existirem não sofrem alterações. As classes base são sempre criadas de raiz a cada compilação do código, mesmo que não tenham havido alterações na PSL sobre essa classe.

As classes de apresentação tratam da estrutura de visualização dos atributos dos objetos. Como cada ecrã tem um propósito, (procurar, criar, visualizar, editar) este tem de ter os eventos para chamar os métodos correspondentes nos objetos de domínio. Esta chamada aos métodos pode ser feita através do uso de eventos ou ações. É da responsabilidade do programador alterar o comportamento dos eventos ou ações caso deseje que estes tenham um comportamento diferente ao comportamento padrão definido na classe base.

É possível ver um exemplo da relação das classes individuais, das classe base e das de tipo de ecrã na *Figura 2.5*.

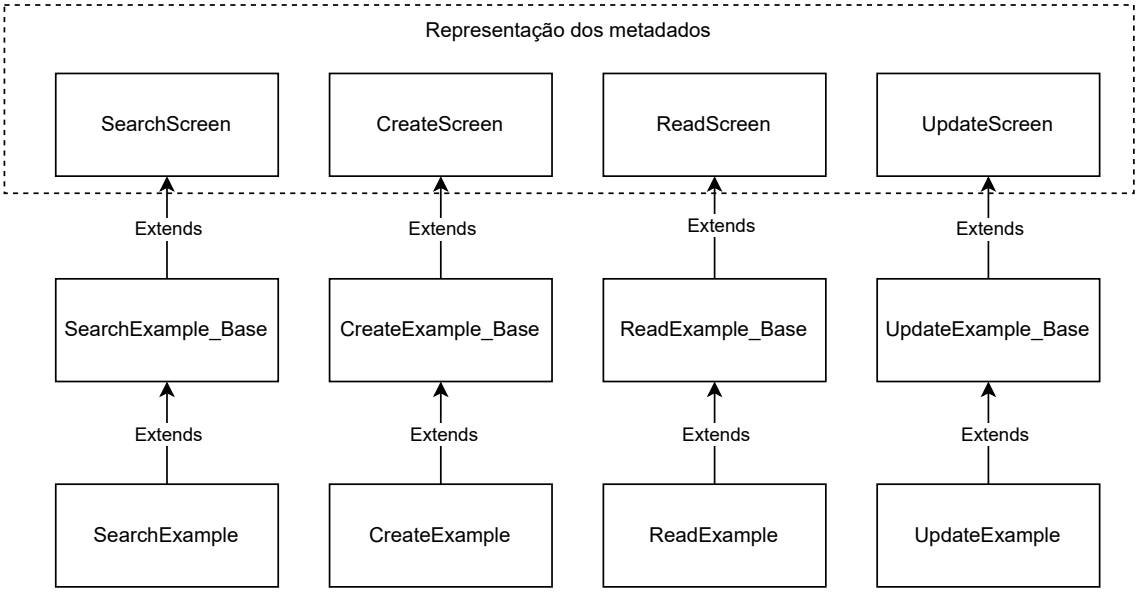

Figura 2.5: Classes da PSL

Interação entre a *dsl* e *psl*

A interação entre a DSL, PSL e o sistema Fenix seguem o padrão de ***MVC*** (*Model View Controller*) [22]. Como é possível ver na *Figura 2.6*, o padrão de *MVC* tem três componentes que interagem entre si para a apresentação e manipulação dos dados do sistema. O utilizador utiliza com o **controlador**, que por sua vez manipula os dados utilizando as classes de **domínio**. Esta manipulação obriga a que sejam atualizados os dados na camada de **apresentação** de maneira a que o utilizador veja as alterações realizadas.

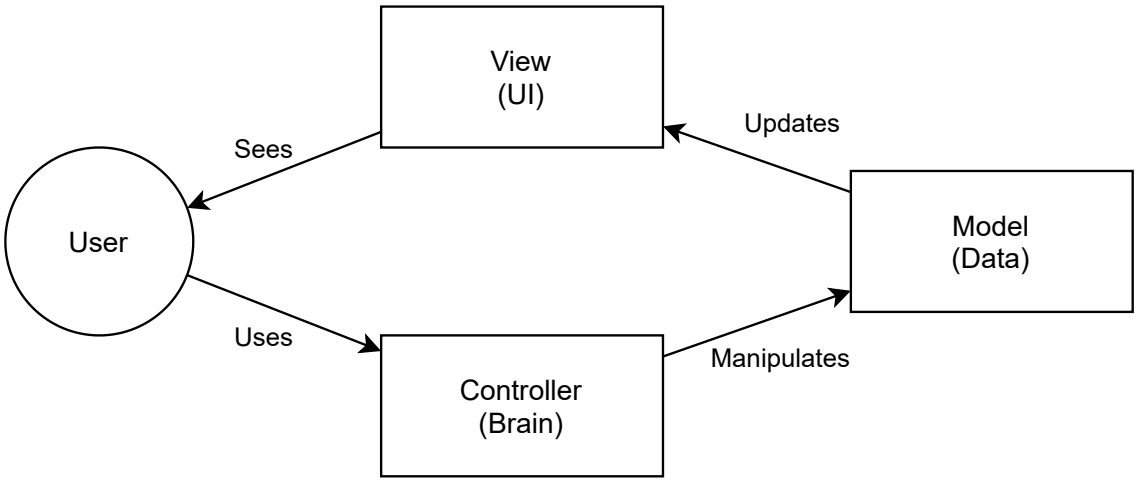

Figura 2.6: Diagrama *MVC*

No sistema Fenix esta apresentação é feita com três componentes, as classes de domínio (**dsl**), as classes de apresentação (**psl**) e o sistema Fenix (**controlador**). O utilizador interage com o sistema Fenix que por sua vez faz as alterações na camada de domínio. No entanto, é o sistema Fenix que realiza as atualizações na camada de apresentação, em vez da camada de domínio, utilizando os novos dados. Quando estes dados de apresentação são alterados o utilizador consegue visualizar as alterações feitas. Este padrão *MVC* do sistema Fenix está apresentado na *Figura 2.7*.

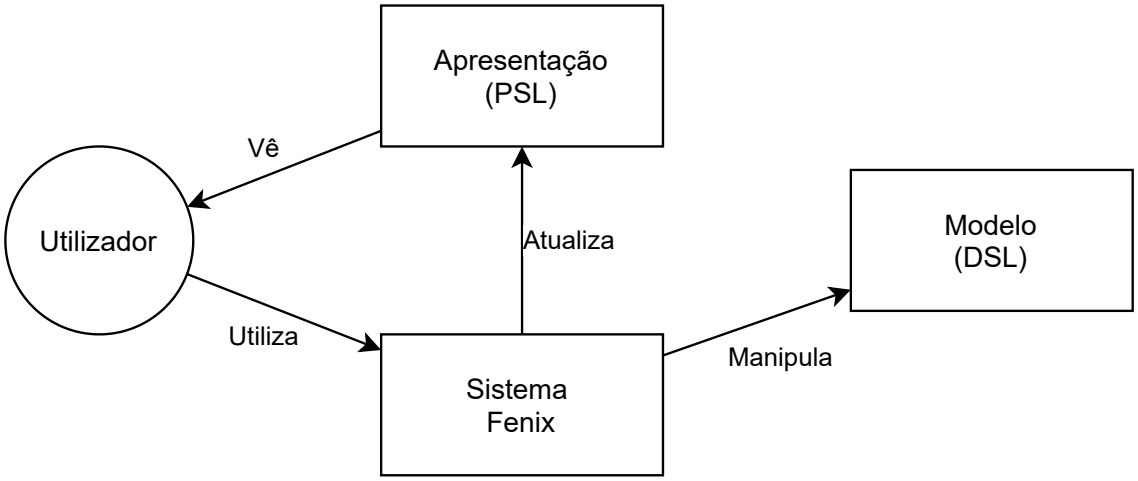

Figura 2.7: Diagrama *MVC* do Fenix

2.3 OmnisBar

A OmnisBar oferece funcionalidades ao programador para facilitar o desenvolvimento de funcionalidades para o sistema Fenix.

Algumas destas funcionalidades permitem que, em certos casos, o programador tenha acesso a atributos especiais dos ecrãs de apresentação, consiga ativar/desativar a opção de

debug em tempo real ou realizar alteração de etiquetas (*labels*) de texto nos ecrãs, sem compilar a totalidade do código.

Uma das funcionalidades mais importantes é a possibilidade de recarregar ecrãs em tempo real. Isto permite ao programador realizar as alterações nas classes dos ecrãs e atualizar no servidor essa classe sem que seja necessário uma recompilação do código ou o arranque de um novo servidor web. Esta recarga só é possível em classes da camada da apresentação ou serviços dos sistema. Qualquer alteração feita a qualquer classe de domínio ou às classes da DSL e PSL necessitam de recompilação do código e de um arranque de um nova instância de servidor web.

Outra funcionalidade de auxílio ao programador é a gestão de etiquetas de texto nos ecrãs de apresentação. Esta funcionalidade permite editar as etiquetas no servidor web e guardá-las para um ficheiro. Se forem feitas alterações nos ecrãs ou alterada a base de dados, basta carregar o ficheiro para o servidor em vez de alterar as etiquetas manualmente uma a uma.

Estas duas funcionalidades acima descritas permitem cortar bastante tempo entre a realização das alterações feitas no código e a sua visualização nos ecrãs, tornando o desenvolvimento mais eficaz.

Algumas das funcionalidades oferecidas pela OmnisBar, como a recarga de código dos ecrãs ou a gestão de etiquetas, necessitam de um *plugin* especial no editor de texto *Eclipse* [16] chamado *OmnisEclipseIntegration*.

Esta barra está presente no final dos ecrãs e tem quatro menus, OmnisBar, Introspeção, Desenvolvimento e Localização, como é possível ver na *Figura 2.8*.

Figura 2.8: OmnisBar

1. OmnisBar

   Este menu permite ver os artefactos instalados e as suas informações, como a data de compilação ou qual a versão de cada módulo instalado. Existem várias operações a realizar sobre a listas de artefactos, como exportar os artefactos ou ver um grafo de dependências. Este menu está apresentado na *Figura 2.9*

Figura 2.9: OmnisBar - OmnisBar

2. Introspeção

   Este menu permite ativar um modo de *debug*, visualizar a composição de classes presentes em cada ecrã (semelhante à funcionalidade de *Web Developer Tools* de um browser) ou visualizar quais os estados dos *stickies* em cada ecrã. Estes *stickies* servem para passar objetos entre os ecrãs. Este menu está apresentado na *Figura 2.10*

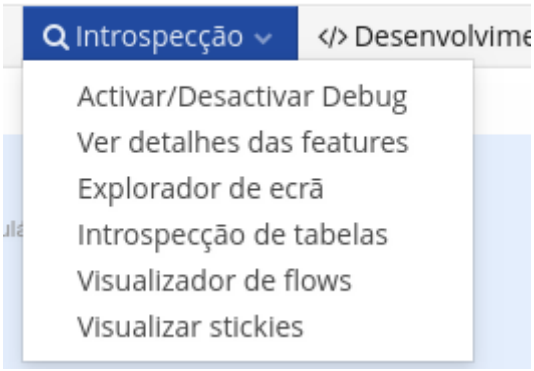

Figura 2.10: OmnisBar - Introspeção

3. Desenvolvimento

   Este menu é o mais importante e utilizado no desenvolvimento de novos ecrãs. Este permite abrir o código do ecrã no eclipse, para uma maior facilidade de desenvolvimento, configurar os caminhos de cada artefacto referente ao seu código local. Existe uma funcionalidade que possibilita que qualquer ecrã seja alterado para a sua versão mais atualizada sem que seja necessário recompilar todo o código (uma versão de software equivalente a um *HotSwap* [1]). É possível utilizar esta alteração para outros artefactos, como as implementações de componentes de ecrãs ou operações de *workflows*. Também é possível recarregar o ecrã em que se está, desde que o artefacto esteja associado corretamente. Estas alterações só são possíveis quando se altera as classes relevantes a classes de apresentação e não a classes de domínio. Este menu está apresentado na *Figura 2.11*

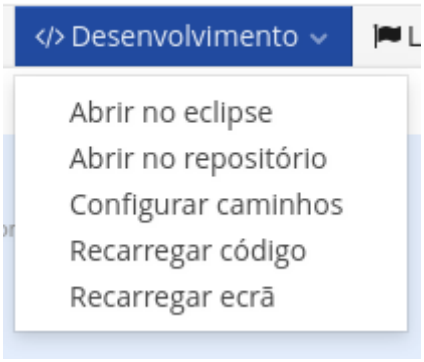

Figura 2.11: OmnisBar - Desenvolvimento

4. Localização

   Este menu permite editar o texto dos rótulos que aparecem nos ecrãs, e guarda automaticamente para um *i18n bundle* (pacote de internacionalização e localização [2]). Também existe a possibilidade de ver, editar e guardar o caminho dos *bundles*. Isto permite exportar as labels presentes no *bundle* para um ficheiro. Este menu está apresentado na *Figura 2.12*

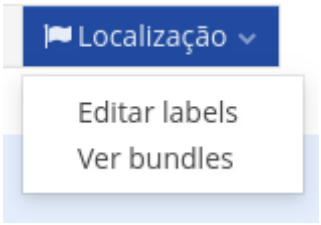

Figura 2.12: OmnisBar - Localização

2.4 Workflow

O sistema Fenix incorpora um motor de *workflow*, que tem um comportamento semelhante a uma máquina de estados. Este motor permite modelar diversos fluxos de processos. Um exemplo de um modelo de processo está presente na *Figura 2.13*. Este exemplo serve apenas para exemplificar como é a estrutura e não o seu funcionamento. A empresa QubIT [21], responsável pelo desenvolvimento do sistema Fenix, disponibilizou alguns manuais sobre o funcionamento do motor de *workflow*. Não é possível apresentar esses manuais por impedimento de divulgação, cópia e disponibilização sem autorização.

Os modelos de processos contêm as operações base, vários estados manuais ou automáticos, sub-processos, operações gerais, eventos, temporizadores e separadores.

- Operações base

Figura 2.13: Workflow - Exemplo Geral

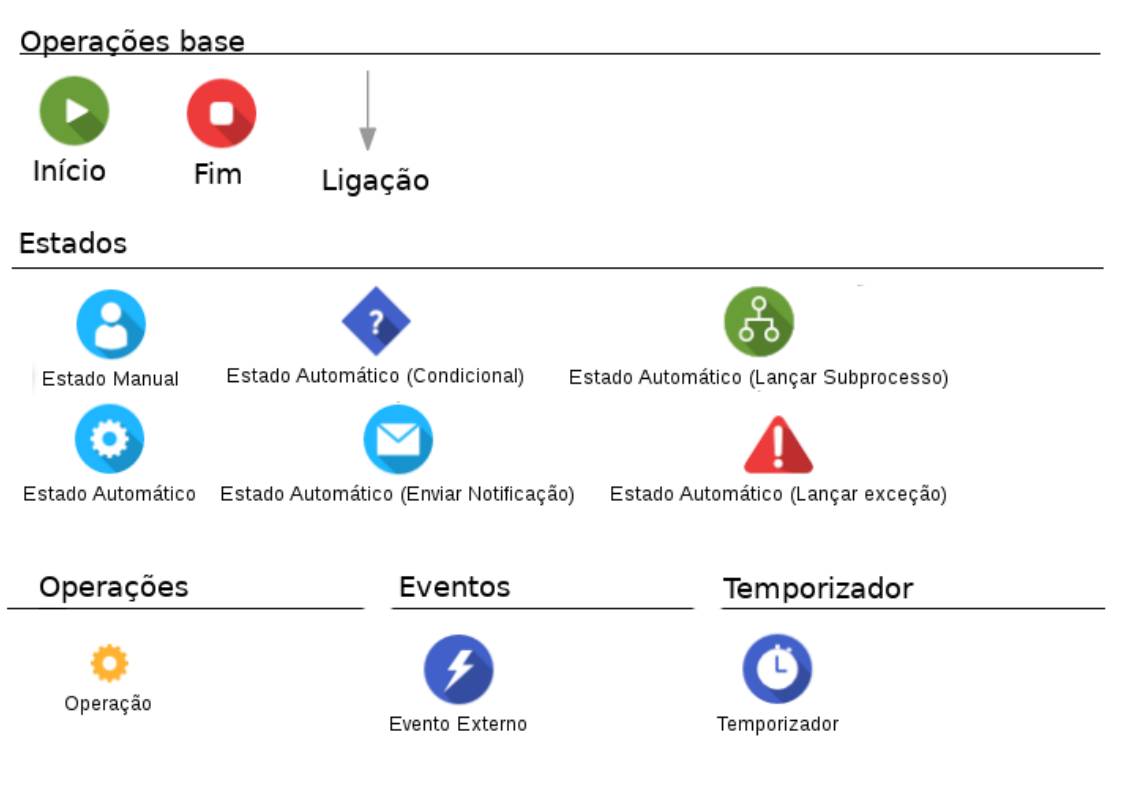

Figura 2.14: Workflow - Componentes

As operações base fazem parte da estruturação base do *workflow*. Estas operações permitem saber quais os pontos de início e de fim do *workflow*, assim como o caminho de transição entre os estados do processo. As operações de início e de ligação de estados são sempre obrigatórias, enquanto a operação de fim pode não ser utilizada se o modelo do processo não tiver um fim definido.

- Início

  O início do *workflow* é definido pelo seu estado inicial, podendo ter várias maneiras de ser despoletado (eventos externos, temporizadores ou criação manual do processo). Esta operação inicial está representada pelo ícone '**Início**' na *Figura 2.14*.

- Fim

  O fim do *workflow* é definido pelo seu estado final, ou de término, podendo ser alcançado através de qualquer operação ou estado. Esta operação final está representada pelo ícone '**Fim**' na *Figura 2.14*.

- Ligação entre estados

  As ligações do *workflow* fazem a ligação unidirecional entre estados e operações. Isto permite definir quais os possíveis caminhos no modelo do processo. Este elemento de ligação está representado pelo ícone '**Ligação**' na *Figura 2.14*.

– Estados

Existem dois tipos de estados: estados manuais (requerem interação de um utilizador) e estados automáticos (são realizados pelo sistema). Os estados são pontos no processo onde é possível realizar alterações nos objetos associados ao processo, como a edição de objetos ou a associação de novos objetos.

* Manuais

Os estados manuais requerem que o utilizador faça as alterações de forma manual no sistema. Isto implica que um utilizador clique em algum botão, faça algum tipo de validação manual relativo a um tipo de informação ou importe algum documento. Estas operações dependem dos separadores presentes no estado. O estado manual é representados pelo ícone '**Evento Manual**' na *Figura 2.14*.

* Automáticos

Os estados automáticos realizam validações ou operações e transitam automaticamente de estado. Estas validações ou operações podem ser mais específicas como estados condicionais, o lançamento de sub-processos, o envio de notificações ou o lançamento de exceções. Quando o estado automático não é um destes 4 estados específicos é representado pelo ícone '**Estado Automático**' na *Figura 2.14*.

Os quatro estados automáticos mais específicos são destacados pelos seus ícones de apresentação:

· Estado Condicional

O estado automático condicional verifica uma condição booleana, por exemplo se um utilizador pertence a um determinado grupo, e faz com que o fluxo apenas siga por um dos dois caminhos possíveis, **Sim** ou **Não**. Este estado automático é representados pelo ícone '**Estado Automático (Condicional)**' na *Figura 2.14*.

· Enviar Notificação

O estado automático de enviar notificação, como o nome indica, permite enviar notificações personalizadas a um ou vários utilizadores. É possível utilizar os grupos de participantes para enviar a notificação para um ou mais grupos. Esta notificação pode ser enviada a pedido de um utilizador, a entrar ou sair do estado. Este estado automático é representados pelo ícone '**Estado Automático (Enviar Notificação)**' na *Figura 2.14*.

· Lançar Exceção

O estado automático de lançar exceção envia uma mensagem de erro ao utilizador quando o processo atinge este estado. À semelhança do estado final, não é possível sair deste estado depois de ser atingido.

Isto serve para parar o processo sem que ele atinja um estado final. Por outras palavras, o processo termina, mas não num estado considerado correto de término. Não é habitualmente utilizado, sendo mais comum para casos excecionais. Se na criação de uma instância do processo existir uma operação condicional e o processo siga o caminho do estado de lançar exceção, é apresentada a mensagem de exceção ao utilizador mas a instância não é criada. Este estado automático é representados pelo ícone '**Estado Automático (Lançar exceção)**' na *Figura 2.14*.

· Sub-processo
Este estado automático permite lançar novos modelos de processos dentro do processo principal, e só segue para o próximo estado do processo principal quando o sub-processo ficar concluído. Em vez de modelar repetidamente o mesmo fluxo, ou partes dele, este estado permite fazer uma modelação de operações e estados que serão repetidos em vários passos e/ou processos. Ao momento deste trabalho é usado um sub-processo de votação no processo de distribuição e recolha dos pareceres dos membros dos júris para os trabalhos de formação avançada de 2º Ciclo. Este estado automático é representados pelo ícone '**Estado Automático (Lançar Sub-processo)**' na *Figura 2.14*.

- Operações gerais

  As operações gerais, ou apenas operações, realizam alterações nos objetos do sistema além de permitirem escolher para qual o estado do fluxo vai transitar. Estas alterações podem ser a troca de atributos, associação/remoção de objetos ou a criação de novos objetos. No entanto, as operações servem muitas das vezes para fazer a ligação unidirecional entre os estados do processo de maneira a avançar o processo. As operações podem ser desencadeadas de forma manual pelo utilizador ou de forma automática através de temporizadores ou de eventos externos. A forma manual é despoletada quando o utilizador carrega no botão da operação, quando este tem as permissões para o fazer, e avança o processo. As operações podem ser despoletadas automaticamente por eventos externos, que são provocados por um evento no sistema, como uma geração de uma referência de pagamento, entre outras possibilidades. Por fim as operações podem ainda ser despoletadas por temporizadores, que ao fim de um tempo pré-definido fazem avançar o fluxo para o estado seguinte. As operações são representadas pelo ícone '**Operação**' na *Figura 2.14*.

- Evento Externo

Os eventos externos despoletam a execução das operações, servindo como uma forma de fazer avançar o processo tendo em conta as alterações externas ao processo. Um exemplo deste evento é quando o fluxo do processo fica à espera de receber um pagamento, como um pagamento de propina, para poder avançar para o próximo estado. Os eventos externos são representados pelo ícone '**Evento Externo**' na *Figura 2.14*.

- Temporizador

  Os temporizadores também despoletam a execução das operações, como os eventos externos, mas podem funcionar como envio de lembretes. Um exemplo deste evento é quando o fluxo do processo fica à espera que passe uma semana da data de abertura de um tipo de candidaturas para poder terminar este período automaticamente e avançar para o próximo estado. Os temporizadores podem ser configurados de maneira a permitir o envio repetido e automático de lembretes em condições que estejam previstas no modelo de negócio. Estes lembretes são resultado do temporizador ativar repetidamente uma operação que envia a notificação. Os temporizadores são representados pelo ícone '**Temporizador**' na *Figura 2.14*.

- Separadores

  Para a apresentação da modelação dos processos é necessário apresentar ecrãs para que os utilizadores realizem as tarefas associadas. Para isto são criados separadores que representam os ecrãs. Estes separadores estão associados a classes dos módulos ou a ecrãs já existentes no sistema. Existem 3 tipos de separadores, separadores customizados, separadores de documentos e ecrãs dinâmicos.

  - Separadores Customizados

    Os separadores customizados necessitam de uma implementação de ecrãs com uso PSL para gerir a apresentação dos componentes que se quer apresentar. Este tipo de separador permite realizar um desenvolvimento mais específico do que se quer apresentar no ecrã. Este tipo de separador é o tipo mais utilizado neste projeto.

  - Separadores de Documentos

    Nos separadores de documentos é possível carregar um documento para o processo. Este documento tem de estar de acordo com tipo de documento criado no modelo de processo. É possível escolher qual dos vários tipos de ficheiros é o documento (*pdf*, *png*, ficheiros *excel*, entre outros), qual o número máximo de documentos que se pode carregar e o tamanho máximo do documento. Este separador pode ser utilizado, por exemplo, por uma aluno para carregar um *Curriculum Vitae* (CV) para um processo de candidatura.

– Ecrã Dinâmico

  Este separador permite que seja criado um ecrã em estilo de um formulário onde é possível escolher as perguntas e o tipo de respostas. É possível criar perguntas com resposta aberta, com escolha múltipla, caixa de texto ou criar um grupo de questões. É possível ainda associar uma cotação para cada pergunta.

Para conseguir escolher quem tem acesso a cada separador basta associar um grupo de participantes para ter acesso. Os separadores têm um campo de permissões que permite dizer qual o grupo que tem acesso e qual o tipo de acesso, escrita ou leitura. Se tiver permissões de leitura pode apenas consultar o separador, se tiver permissões de escrita pode executar ações ou eventos que estejam presentes no separador. Existe ainda um campo de validação que verifica se o separador está num estado correto e não permite avançar no processo se esta validação não estiver correta. É possível ver um exemplo das permissões dos separadores na *Figura 2.15*.

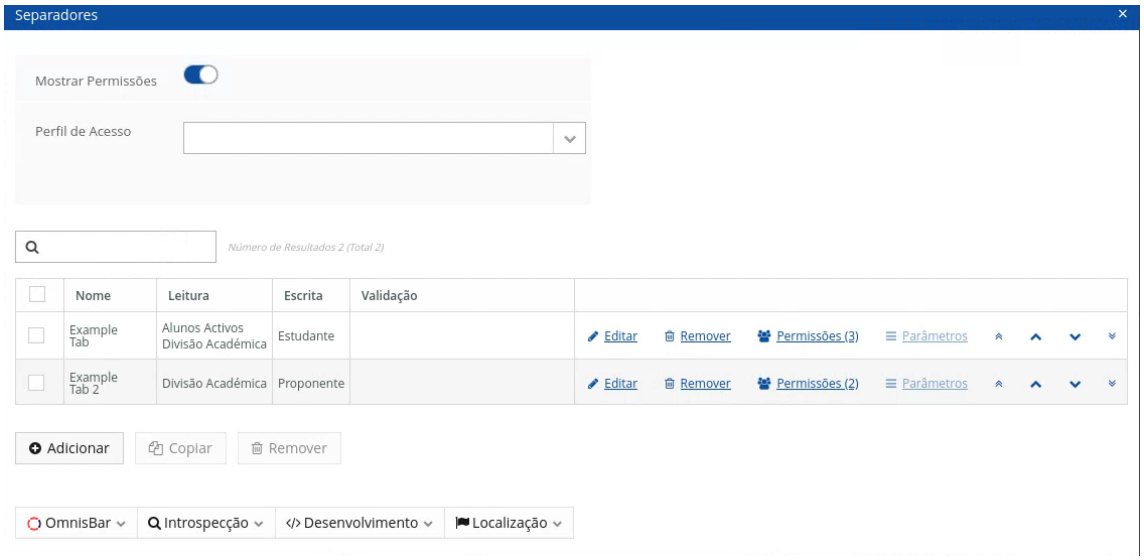

Figura 2.15: Workflow - Separadores (Permissões)

2.5 Sumário

Este capítulo dá um contexto das ferramentas e tecnologias presentes no sistema Fenix, no qual o projeto foi desenvolvido. No capítulo seguinte é apresentado o levantamento de requisitos onde se identifica quais os artefactos e passos necessários que as escolas participantes utilizam ao realizarem o seu processo de gestão de propostas de formação avançada. Esta identificação permite distinguir os componentes chave necessários existentes e a desenvolver nas tecnologias apresentadas neste capítulo.

Capítulo 3

Levantamento de requisitos

Neste capítulo é apresentado o levantamento de requisitos feito a quatro escolas da ULisboa que se voluntariaram para participar nesta etapa do projeto. É apresentado ainda o levantamento de requisitos feito com o Departamento de Informática da Faculdade de Ciências da Universidade de Lisboa (DI-FCUL), visto que este já utiliza um sistema informático interno ao departamento que faz uma gestão das proposta e das candidaturas a trabalhos de formação avançada (PEIpal).

3.1 Introdução

Foram efetuadas reuniões com o Instituto de Geografia e Ordenamento do Território da Universidade de Lisboa (IGOT-UL), com a Faculdade de Farmácia da Universidade de Lisboa (FFUL) e com a Faculdade de Direito da Universidade de Lisboa (FDUL). Nestas reuniões estiveram presentes várias pessoas que estão familiarizadas com o processo de gestão de propostas e candidaturas de formação avançada da sua escola. A Faculdade de Ciências da Universidade de Lisboa (FCUL) disponibilizou um documento que contém informação relativa ao processo de atribuição de alunos a trabalhos de formação avançada. Por fim, foram realizadas três reuniões com três pessoas do Departamento de Informática da Faculdade de Ciências da Universidade de Lisboa (DI-FCUL). A primeira reunião foi feita com o professor Mário Calha, o responsável pelo desenvolvimento do sistema PEIpal/CuCo, a segunda reunião foi realizada com o professor Luís Moniz, responsável pelo sistema PEIpal, e a terceira reunião com o professor Luís Correia, coordenador do Doutoramento em Informática do DI-FCUL.

Para uma melhor compreensão dos conceitos apresentados neste capítulo o conceito de **modalidades** inclui qualquer tipo de trabalho de formação avançada: dissertação, estágio, projeto, investigação, entre outros. O conceito de **títulos** inclui os títulos e temas de propostas (os títulos são mais específicos enquanto os temas são sobre áreas mais abrangentes).

3.2 Faculdade de Ciências da Universidade de Lisboa

O levantamento de requisitos feito com a FCUL foi diferente do processo feito com as outras escolas. Este levantamento de requisitos foi feito através da disponibilização de um documento onde estão enumerados os requisitos que a FCUL necessita para o funcionamento do seu processo de gestão de propostas e de candidaturas de formação avançada. Este documento está apresentado no *Anexo A.3*.

Em contraste com muitas das escolas da ULisboa, a FCUL já utiliza os modelos de processo existente no Fenix para a fazer a gestão dos trabalhos de formação avançada. A utilização destas funcionalidades do SIGA Fenix permite que tenham uma melhor compreensão sobre o que pode ser ou já é feito no sistema. Este conhecimento é evidente na documentação disponibilizada, visto que inclui funcionalidades que a FCUL gostaria de ter disponíveis para realizar a sua gestão de propostas de formação avançada.

3.3 Departamento de Informática da Faculdade de Ciências da Universidade de Lisboa

O levantamento de requisitos com o DI-FCUL foi composto por duas reuniões, ambas feitas no dia 10/01/2020, uma com o coordenador e responsável pela gestão do sistema PEIpal, o professor Luís Moniz, e a outra com o responsável pelos Doutoramentos do DI-FCUL, o professor Luís Correia. Estiveram presentes em ambas as reuniões dois elementos do Departamento de Informática da Reitoria da Universidade de Lisboa (DI-UL) (Francisco Caeiro e André Farias). O objetivo destas reuniões foi compreender como é feita a gestão dos processos iniciais de formação avançada em mestrados e doutoramentos no DI-FCUL.

2º Ciclo - Mestrados

No início do ano letivo, o docente responsável pela Unidade curricular (UC) de Disciplina de Projeto de Engenharia Informática (DPEI) faz várias configurações para o processo de gestão de propostas de formação avançada. É criado um documento interno, fora de sistema, com as estratégias e definição das regras para o ano letivo. O coordenador envia um e-mail a todos os proponentes (internos e externos) a divulgar o início do período de submissão de propostas. Para a criação de proponentes internos, o coordenador tem de criar manualmente os utilizadores no sistema quando um docente entra para o departamento. Para os proponentes externos, existe um perfil externo associado a uma entidade externa que pode criar propostas de formação avançada. Este perfil externo é responsável pela criação de utilizadores externos, designados por supervisores. Os supervisores não podem criar propostas, apenas podem ser atribuídos a propostas criadas pelo perfil externo que o criou. Para carregar os alunos, o coordenador vai ao sistema Fenix e exporta uma

lista com todos os alunos inscritos no primeiro ano do mestrado, e carrega esta informação proveniente de um ficheiro excel para o sistema. Caso o aluno não reúna as condições necessárias para realizar um trabalho de formação avançada, o coordenador tem de retirar o aluno manualmente. Pode haver casos em que alunos que tenham no máximo 3 UCs por fazer, realizem o trabalho de formação avançada com um requerimento. Quem faz a gestão de todos os intervenientes externos, proponentes, supervisores e propostas, é o coordenador da DPEI.

Existem duas modalidades para o trabalho final, dissertação ou projeto. Não é possível submeter temas nem áreas de estudo, apenas títulos (concretos) de propostas. Pode existir um aluno pré-selecionado para realizar a proposta, mas no sistema aparece apenas se existe ou não um aluno pré-selecionado, não é divulgado qual é o aluno. Isto não impede que a proposta tenha vários interessados, nem impede que o aluno que é pré-selecionado se candidate a outras propostas. As propostas submetidas por proponentes externos podem não ter orientadores internos atribuídos, para ultrapassar esta situação o coordenador faz um pedido aos docentes para orientarem a proposta no caso de esta ter sido atribuída a algum aluno. Não é possível os alunos criarem as próprias propostas (auto-propostas), são sempre os proponentes internos e externos a criar e submeter propostas.

As propostas têm de ser validadas pelo coordenador e pelo Conselho Científico (CC). Durante o período de validação, o coordenador pode editar erros que não alterem a propostas, como correções das datas de início ou fim, duração, ou erros ortográficos. É o coordenador que faz a validação da proposta em sistema (uma a uma). É obrigatório a proposta ter um orientador interno. Esta obrigatoriedade tem de ser cumprida no caso da proposta ser atribuída a um aluno, mas não é obrigatório ter um orientador interno quando a proposta é criada. Pode existir ainda um orientador externo e um supervisor/tutor associado à proposta, e este tem de obrigatoriamente ser doutorado.

Durante o período de candidaturas os alunos podem escolher até seis propostas ordenadas por ordem de preferência. Enquanto este período estiver aberto, o aluno pode alterar a ordem da sua escolha ou trocar as propostas escolhidas.

No final do período de candidaturas o coordenador envia um e-mail aos proponentes para estes irem ao sistema verem quais os alunos que se candidataram a cada uma das suas propostas. Os proponentes têm de seriar e escolher os alunos para as suas propostas, contudo os critérios das escolhas é da inteira responsabilidade de cada um, sendo estas diferentes entre eles. Quando as propostas são atribuídas a alunos passam a designar-se de trabalhos de formação avançada.

Após o aluno e o proponente terem entrado em acordo, o aluno tem de confirmar a escolha da proposta no sistema. O coordenador informa o gabinete de estudos de pós graduados para este realizar os protocolos e acordos com as instituições externas para as quais os alunos vão realizar os trabalhos de formação avançada. Estes protocolos interinstitucionais e acordos específicos são todos em papel. Os alunos podem cancelar a sua

candidatura ou trabalho enviando um e-mail com as razões ao coordenador, e este indica no sistema a desistência do aluno. Quando um aluno desiste da proposta, esta fica bloqueada e não pode ser escolhida por outro aluno. Os alunos que pretendam fazer uma troca de trabalho, enviam um e-mail ao coordenador com as razões pelas quais querem trocar o trabalho e o coordenador troca o estado dos alunos para candidato para que estes possam selecionar uma nova proposta para realizarem. Quando não existem orientadores internos associados às propostas escolhidas ou já atribuídas por parte dos alunos, o coordenador faz uma divulgação destas propostas para que os docentes possam escolher quais pretendam orientar. Após os docentes escolherem quais a propostas que querem orientar, enviam um e-mail ao coordenador e este assinala em sistema qual o orientador interno para cada proposta. Esta divulgação é feita no final do processo.

3º Ciclo - Doutoramentos

No caso dos doutoramentos, são os alunos que submetem as suas próprias propostas de formação avançada (auto-proposta) com o título, orientação e alguma informação como um resumo do trabalho, os objetivos e o plano de trabalho. Para as propostas de doutoramento só existe a modalidade de dissertação. Normalmente as propostas de doutoramento já têm uma bolsa associada, na maior parte dos casos, esta bolsa é proveniente da Fundação para a Ciência e a Tecnologia (FCT).

O aluno entrega o documento de proposta ao coordenador do doutoramento que a entrega à Comissão de Acompanhamento (CA). Esta Comissão de Acompanhamento é composta pelo coordenador do doutoramento, 3 docentes das várias áreas científicas que podem ser elementos do Conselho Científico.

O Conselho Científico é que valida as propostas que foram entregues à Comissão de Acompanhamento.

O plano de trabalho entregue tem de ser aprovado pelo Conselho Científico.

À data da reunião não foi possível saber quem faz a validação da proposta do alunos, se são os elementos do CC ou da CA. Também não é possível saber quem é a equipa de orientação do aluno. **Nota:** Esta informação não é possível determinar **no sistema Fenix**, apesar de ser conhecida pelos intervenientes internos do departamento.

É obrigatório a proposta ter um orientador interno, e no caso de existir um orientador externo este tem de ser doutorado.

No caso dos doutoramentos realizados neste departamento é necessário que o aluno apresente e defenda a sua proposta de tese. Se for aprovada a defesa então o aluno poderá começar a realizar o seu trabalho de formação avançada. Nesta defesa existe um júri que é constituído pela Comissão de Acompanhamento, coordenador do doutoramentos, orientador e outros possíveis participantes. Caso o aluno não obtenha aprovação na defesa, este não poderá realizar a sua tese.

A submissão da tese é feita após a defesa da proposta da tese.

3.4 Sistema PEIpal/CuCo

O PEIpal é o sistema de gestão de propostas de formação avançada de 2º Ciclo criado e utilizado pelo Departamento de Informática da Faculdade de Ciências da Universidade de Lisboa (DI-FCUL). A primeira versão, criada em 2010, ainda em uso no DI-FCUL trouxe uma gestão informatizada da propostas de formação avançada dentro do departamento. Isto permite que os proponentes e os alunos tenham uma maior facilidade na realização das suas tarefas durante o processo, ao mesmo tempo que dá ao coordenador da DPEI uma visão global no decorrer do processo.

A segunda versão (CuCo), criada em 2013, nunca chegou a ser implementada no departamento. No entanto, existe documentação a explicar os intervenientes, processos e funcionalidades que este sistema iria fornecer aos seus utilizadores. Esta documentação foi fornecida pelo DI-FCUL e permitiu ficar a saber que operações iriam ser ou não possíveis realizar, como são feitos os processos de candidaturas, os processos de gestão das propostas e a validação das mesmas, assim como a gestão interna do sistema.

Para complementar a informação enviada existiu uma reunião com o professor Mário Calha, o responsável pelo desenvolvimento do sistema PEIpal, no dia 05/11/2019 e com dois membros do DI-UL (Francisco Caeiro, José Lima) que permitiu esclarecer dúvidas sobre o funcionamento dos dois sistemas. Esta reunião, juntamente com mais duas reuniões detalhadas na *Secção 3.3 - Departamento de Informática da Faculdade de Ciências da Universidade de Lisboa*, permitiu ter uma visão do funcionamento do sistema PEIpal/-CuCo, assim como os seus requisitos.

As principais funcionalidades do sistema foram identificadas e agrupadas em 3 categorias, a gestão de configurações anuais, a gestão de propostas de formação avançada e a gestão de candidaturas às propostas. Dentro de cada categoria estão explicados os detalhes de cada uma das funcionalidades.

1. Gestão de configurações anuais

   No início de cada ano civil, o administrador do sistema e o coordenador têm de criar configurações no sistema que serão utilizadas na gestão dos processos do ano letivo seguinte.

   I Criar utilizador com perfil de coordenador

   O administrador do sistema consegue criar utilizadores com o perfil de coordenador e atribuí-lo ao coordenador da UC de DPEI do ano letivo corrente. Este perfil de coordenador tem permissões para criar utilizadores com perfis de proponente interno, externo, consegue validar os utilizadores externos e a validar as propostas criadas pelos proponentes. No caso do DI-FCUL só existe um coordenador.

   II Criar utilizadores de perfis de proponentes externos

A criação de utilizadores com perfil de proponentes externos é feito quando uma entidade externa designa uma pessoa responsável que vai criar propostas em nome da entidade externa. Este utilizador pode criar e editar propostas de formação avançada enquanto o período de submissão de propostas estiver aberto. Quando cria as propostas pode associar um supervisor externo à proposta. Este utilizador externo é o ponto de comunicação entre a entidade externa e o DI-FCUL.

III Criar utilizadores com perfil de supervisor externo

A criação dos utilizadores com este perfil pressupõe a existência de um utilizador com perfil externo. O utilizador com perfil de supervisor externo pode ser associado às propostas criadas pelo utilizador com perfil externo. Este supervisor externo pode ser considerado o orientador externo da proposta.

IV Criar utilizadores com perfil de proponente interno (docente)

A criação de um utilizador com perfil de proponente interno é feita assim que um docente entra no departamento. Este utilizador pode criar e editar propostas de formação avançada enquanto o período de submissão de propostas estiver aberto. Os docentes têm permissões para ver todas as propostas criadas, quem se candidatou e qual a ordem de preferência. Esta visão engloba propostas de proponentes internos e externos.

2. Gestão de propostas de formação avançada

Após as criações de configurações e utilizadores dá-se início ao processo de gestão das propostas. Este processo inclui a criação e validação das mesmas.

V Criar propostas de formação avançada

Os proponentes, internos e externos, têm permissões no sistema para criar e editar propostas de formação avançada. Estas propostas podem ter duas modalidades, dissertação ou projeto, e devem ter preenchida a informação necessária à sua validação. Esta informação inclui o título, introdução, objetivos, plano de trabalho, duração do trabalho, os responsáveis (orientadores, proponente, supervisores/tutores), quais os cursos a que se aplica a proposta entre outras informações opcionais como a remuneração ou a existência de um aluno pré-selecionado.

VI Validar propostas de formação avançada

No fecho do período de submissão de propostas, e antes da validação oficial das propostas, o coordenador faz uma filtragem manual das mesmas para garantir que estão de acordo com as normas do departamento. Caso não estejam, em vez de avançarem para a fazer de validação, os proponentes são informados (através de um e-mail) quais os campos ou informações que não cumprem

as normas. Caso pretendam editar, realizam as alterações e enviam novamente a proposta ao coordenador para seguir para validação. Se não realizarem as edições, ou após a edição ainda não esteja de acordo com as normas, a proposta é rejeitada.

A validação das propostas é feita através de pareceres individuais dos membros do CC e do coordenador da UC. Quando existe uma proposta que não esteja de acordo com os requisitos, o coordenador envia um e-mail informativo ao proponente sobre as informações que estejam em falta ou mal especificadas, para que o proponente possa editar as inconformidades. Esta última edição é relativa ao conteúdo da proposta, como o tema, os objetivos ou relativo à complexidade do trabalho, e não relativo a informações da proposta como o tempo de trabalho, local de trabalho, etc.

3. Gestão de candidaturas a propostas de formação avançada

No final do período de gestão de propostas estas ficam disponíveis para os aluno se candidatarem. Existe uma primeira fase de candidaturas para os alunos que tenham concluído o primeiro ano do mestrado. Existe ainda uma segunda fase para alunos que não ficaram com propostas atribuídas na primeira fase ou para alunos com um estatuto especial.

VII Candidatura às propostas pelos alunos – 1ª fase

Quando o período de candidaturas às propostas abre, os alunos que estejam em condições de estar inscritos na DPEI (i.e., 48 *European Credit Transfer and Accumulation System* (ECTS) feitos no primeiro ano) podem visualizar todas as propostas atribuídas ao curso em que estão inscritos. Os alunos podem escolher até seis propostas de trabalhos avançados ordenadas pela sua preferência. Enquanto o período de escolha estiver aberto, os alunos podem trocar as propostas ou a ordem de preferência.

VIII Seriação dos alunos e das propostas escolhidas – 1ª fase

A seriação dos alunos para cada proposta fica ao critério do proponente, ou seja, é ele que decide quais são os critérios para escolher o aluno candidato que vai realizar cada proposta. Esta seriação pode ser feita através de entrevistas com os alunos ou através de uma seleção automática de acordo com diversos parâmetros, como média do primeiro ano de mestrado, licenciatura, etc.

Quando o proponente escolhe um aluno, informa o mesmo da sua escolha. No entanto, é o aluno que indica no sistema qual a proposta escolhida. Após esta indicação é que o aluno fica atribuído a uma proposta de formação avançada e dá-se início ao trabalho de formação avançada. Este trabalho é gerido pelo

sistema Fenix que, no caso da FCUL, já tem modelos de processo presentes no sistema.

Nota: No sistema CuCo estava previsto que o proponente selecionasse o aluno no sistema para cada proposta, e o aluno indicasse a proposta que escolheu, também no sistema. Só quando existir esta dupla confirmação (*double-lock*) é que o aluno ficaria com a proposta atribuída.

IX Candidatura às propostas pelos alunos – 2ª fase

A segunda fase de candidaturas às propostas é diferente da primeira. Nesta fase é o aluno que entra em contacto com os proponentes e tenta arranjar uma proposta para realizar, ao entrar em acordo com os proponentes. Quando estiverem ambos em acordo, o aluno tem de ir ao sistema e indicar qual a proposta que escolheu. Nesta fase não é necessário que o proponente faça nada. Se o aluno se enganar na escolha da proposta, ou o aluno ou o proponente avisam o coordenador do sucedido, que resolve a situação de forma manual.

Os alunos que vão a segunda fase de candidaturas a propostas são aqueles que não ficaram com nenhuma proposta atribuída em primeira fase ou tenham estatuto de Lic+Mestrado[1] com condições de transitar para o 2º ano do 2º Ciclo.

Em caso de algum erro de negócio no sistema é o coordenador que está encarregue de o corrigir manualmente, com operações *Create, Read, Update, Delete* (CRUD) diretamente no sistema. A inserção de alunos em segunda fase é feita manualmente em sistema por parte do coordenador.

3.5 Faculdade de Farmácia da Universidade de Lisboa

A reunião para o levantamento de requisitos com a FFUL realizou-se no dia 11/12/2019 com a presença de vários elementos da FFUL (Pedro Russo, Inês Agostinho e Raquel Patrício) e dois elementos do DI-UL (Francisco Caeiro e André Farias). Esta reunião permitiu identificar os vários pontos chave que fazem parte do procedimento interno da gestão de propostas de trabalhos de formação avançada.

2º Ciclo - Mestrados

Os cursos têm três modalidades, estágio, dissertação e projeto, a modalidade de projeto apenas existe no Mestrado Integrado em Ciências Farmacêuticas (MICF). Este mestrado

[1] **Estatuto Lic+Mestrado:** Alunos que estejam a fazer pelo menos uma cadeira de Licenciatura e estejam a fazer unidades curriculares isoladas. Só podem ir a segunda fase de escolha de propostas caso obtenham a conclusão da licenciatura e aprovação de pelo menos oito cadeiras (48 ECTS) nas unidades curriculares isoladas à data da abertura da 2ª fase.

tem um processo idêntico ao processo de gestão de propostas feito pelo DI-FCUL, explicado na *Secção 3.3 - Departamento de Informática da Faculdade de Ciências da Universidade de Lisboa*.

Além da criação de propostas por parte dos docentes, a FFUL permite que os alunos criem as suas próprias propostas.

É permitido existir uma pré-seleção de alunos para as propostas. Esta pré-seleção é feita com o nome do aluno descrito na proposta.

Os alunos têm de preencher e entregar um formulário de "Submissão do Tema de Dissertação" na secretária académica para submeterem o trabalho de formação avançada. Esta submissão contém as seguintes informações:

- Do aluno:
  - Nome, número de aluno, telefone, e-mail, morada, código postal e localidade.
- Dos orientadores (interno e externo):
  - Nome, categoria, instituição, e-mail, tipo de documento de identificação e número do documento de identificação.

Esta submissão inclui ainda qual é o mestrado, ano letivo, o título, a língua (Português ou Inglês), se é a primeira submissão ou não, uma síntese do trabalho, objetivos, pertinência do estudo, metodologia, cronograma, local de trabalho e 5 a 10 referências bibliográficas. Esta submissão é assinada pelo aluno, orientadores e coordenador do mestrado. Ambos os orientadores têm de ser doutorados e tem de haver pelo menos um orientador interno. É entregue em anexo um CV do aluno devidamente atualizado e assinado, e um CV do orientador externo.

Ao contrário de outras escolas, a proposta só é validada em CC após ser submetida. Para fazer pequenas alterações no trabalho, como correções ortográficas no título, têm de ser feitas pelo orientador. Para realizar trocas de conteúdo, como a troca do título, é necessário fazer uma nova submissão de proposta no sistema e esta tem de ser aprovada em CC novamente.

A lista de propostas é colocada no moodle para os alunos visualizarem todas as propostas disponíveis. A escolha das propostas e dos locais de estágio por parte dos alunos são feitas por ordem de nota mais alta para a mais baixa, o aluno com a melhor nota escolhe primeiro, depois o segundo, e assim sucessivamente. Os alunos têm até 48 horas para alterar a escolha da proposta.

A comunicação entre proponentes externos e docentes é feita de forma informal fora do sistema, através de e-mails ou chamadas telefónicas, e são os docentes que propõem a proposta em nome pessoa externa, como orientadores internos. Existe uma troca de protocolos informalmente com as entidades para onde os alunos vão fazer a dissertação.

Erasmus

Todos os alunos de todos os cursos podem ir fazer o trabalho de formação avançada em Erasmus mas são maioritariamente alunos do MICF.

Os alunos do MICF vão 3 meses em trabalho, enquanto os outros cursos podem ir até 1 ano letivo, sendo que esse ano também pode ser para realizar UCs.

Os alunos vão maioritariamente para escolas (faculdades e universidades) ou entidades que estejam relacionadas com estas, como laboratórios de investigação. No entanto, nada impede que não possam ir para qualquer outra entidade.

Existe a possibilidade de os alunos não saberem à partida qual o título da proposta antes de irem. Esta divulgação desta informação depende da entidade externa.

A lista de entidades externas para quais os alunos podem ir realizar o trabalho de formação avançada é divulgada no portal da FFUL.

Existem acordos de intercâmbio com as entidades externas, com o número de alunos que podem ir ou vir.

O critério normal para a seriação é a média das UCs dos 3 anos, mas podem ser utilizados outros ou mais critérios além das médias.

Os alunos para irem de Erasmus têm de estar no 4º ano.

3º Ciclo - Doutoramentos

No caso dos doutoramentos só existem propostas feitas por parte dos alunos e, na maioria dos casos, os alunos já têm associada uma bolsa FCT para a sua tese. As propostas têm de ser aprovadas pelo CC.

3.6 Faculdade de Direito da Universidade de Lisboa

A reunião de levantamento de requisitos com a FDUL ocorreu no dia 10/01/2020, com dois elementos da divisão académica (Bertolino Campaniço e João Cardoso) e dois elementos do DI-UL (Francisco Caeiro e José Lima). O objetivo desta reunião foi fazer um levantamento de requisitos sobre o processo de gestão das propostas de formação avançada nos mestrados e doutoramentos.

2º Ciclo - Mestrados

Na FDUL só existem duas modalidades para os trabalhos finais de curso de mestrado, dissertação ou estágio. Cada modalidade tem uma UC associada. Para os alunos escolherem que modalidade da proposta querem realizar, estes têm de se inscrevem na UC respetiva da modalidade. Como existem quatro mestrados existem oito UCs, das quais os alunos só se podem inscrever em uma de duas possíveis.

Existem a possibilidade de serem os alunos a criarem e submeterem as suas próprias. As propostas têm de ter título, ano letivo e orientador e este tem de aprovar as propostas para que foi selecionado para as poder orientar. Depois do orientador aceitar a orientação da proposta, esta vai a CC para ser aprovada. Só depois desta aprovação é que o aluno fica com a proposta atribuída. O aluno pode trocar de modalidade da proposta enviando um e-mail com um requerimento à divisão académica.

Quando o aluno tem uma proposta atribuída é informado pelo gabinete de estágios para escolher o local de trabalho para realizar o seu trabalho. É o aluno que escolhe o local de trabalho e informa o gabinete de estudos da escolha através de um e-mail. A seriação dos alunos para a ordem de escolha de local de trabalho é feita tendo em conta as notas, médias e com a ponderação de cada proponente. Para cancelar o seu trabalho o aluno envia um e-mail com um requerimento para a divisão académica, e é esta que faz o cancelamento.

O aluno pode alterar o trabalho final enviando um e-mail com um requerimento para a divisão académica. Se a troca for apenas de informação que não altere o âmbito do trabalho, e.g. erros ortográficos, a divisão académica faz a alteração. Se for uma troca de trabalho ou orientação, o aluno tem de cancelar e criar uma nova proposta que irá dar origem a um novo trabalho. Esta nova proposta tem de passar novamente pela validação do orientador atribuído e pelo CC.

3º Ciclo - Doutoramentos

No caso dos doutoramentos na FDUL apenas existe auto-proposta e o aluno tem de indicar o título e o orientador. Nos doutoramentos só existe a modalidade de dissertação. Os alunos podem fazer o doutoramento fora da FDUL, mas são poucos os casos em que tal acontece.

3.7 Instituto de Geografia e Ordenamento do Território da Universidade de Lisboa

A reunião de levantamento de requisitos ocorreu no dia 27/11/2019 com quatro elementos da Unidade de Gestão Académica (Rita Matos, Elisabete Nunes, Elisabete Almeida e Joana Santos) e três elementos do DI-UL (Francisco Caeiro, José Lima e André Farias).

2º Ciclo - Mestrados

No IGOT-UL existe um coordenador (docente) responsável por cada curso de mestrado e um docente responsável pelas três UCs relativas ao trabalho final de Dissertação / Estágio / Projeto (DEP). Estas três UCs estão separadas para permitir separar a escolha das modalidades. Se um aluno quiser realizar um trabalho de formação avançada em determinada

modalidade, o aluno inscreve-se na UC respetiva. O coordenador do curso é o responsável pela comunicação com os proponentes e as entidades externas, feita de forma informal, através de trocas de e-mails.

O coordenador do curso envia um e-mail aos proponentes para que estes lhe enviem as suas propostas. Quando são criadas as propostas, por parte dos docentes ou entidades externas, não existe uma pré-seleção do aluno para cada proposta.

O IGOT-UL permite também que sejam os alunos a criarem as suas próprias propostas apresentando um de plano de trabalho, a equipa de orientação e o local de trabalho.

As propostas necessitam que o coordenador do curso faça uma validação prévia antes de serem avaliadas pelo CC. Depois de ser aprovado pelo CC é que a proposta pode ser apresentada aos alunos.

Quando um aluno tem interesse numa proposta, este comunica o seu interesse ao coordenador do curso ou ao coordenador da UC de DEP. Esta comunicação de interesse resulta, na maioria das vezes, a atribuição da proposta ao aluno. No entanto, o IGOT-UL agiliza o processo de atribuição e escolha de propostas de maneira a que todos os alunos fiquem com uma proposta associada.

O aluno tem de entregar uma folha de inscrição do trabalho de DEP com um plano de trabalho, a descrição do contexto, a introdução à relevância do tema, os objetivos, a metodologia, uma bibliografia inicial e um cronograma. Se o aluno desejar realizar o trabalho numa língua diferente de Português, tem de apresentar um certificado de língua. Quando um aluno quer fazer a troca de alguma informação do trabalho é necessário que apresente uma nova submissão com o título, qual a modalidade e a submissão anterior. Estas submissões têm de ser aprovadas pelo coordenador de curso e de UC e finalmente pelo CC.

Para o início dos trabalhos é necessário que o aluno entregue e assine um protocolo com a entidade externa responsável pela proposta. Para isto tem de existir um protocolo prévio com essa entidade que permita a colaboração entre ambas.

O IGOT-UL realiza todo este processo sem usar o sistema académico Fenix, isto faz com que o processo seja único, visto que são os responsáveis do curso que decidem como fazer o processo.

3º Ciclo - Doutoramentos

A submissão de propostas é feita previamente pelos alunos. Normalmente a proposta é referente a um trabalho que já tem uma bolsa financiada pela FCT ou que venceu algum concurso. Nestes casos, o aluno só tem de submeter a proposta de trabalho e indicar a informação necessária para a submissão (igual à de 2º Ciclo).

3.8 Sumário

Neste capítulo, foi descrito o processo de levantamento de requisitos individuais de cada escola que participou nesta fase, assim como uma breve explicação do funcionamento do sistema PEIPal/CuCo. Este capítulo serve para saber quais são os casos de uso das escolas e quais são as funcionalidades críticas nos processos de gestão de propostas de formação avançada de cada uma. No capítulo seguinte será realizada a análise dos requisitos. Esta análise parte da informação obtida neste capítulo onde serão definidos os pontos chaves necessários para o funcionamento correto deste projeto nas escolas da ULisboa.

Capítulo 4

Análise e Desenho

Neste capítulo é apresentada a análise de requisitos e os desenhos do sistema. Esta análise tem como objetivo encontrar as funcionalidades (casos de uso) do sistema. Estas funcionalidades têm de responder às necessidades das escolas que participaram no levantamento de requisitos. Neste capítulo são identificados os utilizadores do sistema, os casos de uso, os requisitos não funcionais, o desenho de entidade associação [3] e dos processos dos dois grupos principais, gestão de propostas e candidaturas às propostas.

4.1 Utilizadores

Os principais utilizadores do SIGA Fenix que irão interagir com as funcionalidades deste módulo são:

- **Aluno** - Utilizador com conta Fenix ativa, na escola a que pertence, com condições de realizar um trabalho final de curso. Este utilizador pode criar propostas (caso a escola o permita), candidatar-se a propostas e escolher qual a proposta a realizar, quando os respetivos períodos estiverem abertos.

- **Coordenador** – Docente responsável por uma UC ou por um curso. O coordenador é responsável pela gestão das propostas de formação avançada e das regras de negócio individuais de cada escola, departamento, curso ou UC. O coordenador pode fazer parte do grupo de responsáveis pela validação das propostas criadas pelos proponentes.

- **Proponente** – Perfil de uma pessoa interna ou externa que pode criar e editar as suas propostas de formação avançada no sistema. A quem é atribuído este perfil depende das regras de cada escola, podendo até ser alunos. O proponente é responsável ainda pela escolha do aluno candidato que ficará atribuído às suas propostas.

- **Conselho científico** – Grupo de docentes da escola responsáveis pela validação das propostas de formação avançada, através de um parecer.

- **Administrador de sistema** – Responsável pela criação e gestão das configurações que permitem realizar o processo de gestão de propostas de formação avançada e do processo de candidaturas às propostas de formação avançada. Este administrador pode ser ainda responsável pela correção de erros que possam surgir no decorrer dos processos.

4.2 Casos de uso

Foram identificados 9 casos de uso, representados por um código (UC1, UC2, etc). Estes casos de uso foram separados em dois grupos, o primeiro grupo é relativo à gestão das propostas e o segundo é sobre o processo de candidaturas às propostas. Todos os casos de uso abaixo descritos têm como **pré-condição** que o utilizador tem uma conta no sistema Fenix e que já realizou o *login* na mesma. Os utilizadores apresentados em baixo são os esperados que realizem os casos de uso e não existe nenhuma restrição que outros utilizadores não possam realizar os casos de uso, se a escola o permitir.

4.2.1 Gestão de propostas

O primeiro grupo do problema consiste em fazer a gestão inicial do processo de criação e validação de propostas de formação avançada, recorrendo ao sistema Fenix e às ferramentas necessárias para esta gestão inicial.

Caso de Uso - UC1: Criar configuração de abertura de submissão de propostas

Para a correta utilização do módulo nas escolas, optou-se por definir este caso de uso para permitir configurar os componentes do módulo de forma a responder aos requisitos específicos de cada escola.

Este requisito serve para fazer uma ponte entre as regras de negócio específicas de cada escola e o módulo. Este caso de uso permite que exista a flexibilidade necessária para a utilização do módulo nas diferentes instâncias dos sistemas de cada escola e nos diferentes ciclos de estudo. Desta maneira o módulo permite que cada escola o adapte às suas necessidades específicas.

Sumário: O utilizador cria uma configuração para um novo período de submissão de propostas de formação avançada.

Utilizadores: Administrador do sistema.

Cenário Principal de Sucesso:

1. O utilizador acede ao portal de configurações de abertura à submissão propostas de formação avançada.

2. O utilizador inicia o processo de criação de uma nova configuração fornecendo as informações para a sua criação, por exemplo, a escolha o âmbito da configuração (cursos ou UCs, modalidades disponíveis ou o ano de execução.

3. O utilizador indica quais são os cursos ou UCs para adicionar à configuração.

4. Por fim tem de selecionar os responsáveis pela validação das propostas e escolher as possíveis modalidades. Cada curso ou UC associado tem uma lista de responsáveis e modalidades própria.

5. O utilizador confirma a finalização da configuração de abertura.

Pré-condições: Já existem UCs ou cursos abertos na instância do sistema para o ano letivo em que o utilizador pretende criar a configuração.

Pós-condições: Fica criada uma configuração de abertura com UCs ou cursos associados. As modalidades e responsáveis pela validação das propostas de cada curso ou UC também ficam configurados.

Regras de negócio: Não aplicável.

Caso de Uso - UC2: Criação de propostas de formação avançada

Sumário: O utilizador cria uma proposta de formação avançada.

Utilizadores: Pessoa com permissões no sistema Fenix que pode criar propostas de formação avançada (proponente). Esta pessoa pode ser aluno, docentes ou pessoas externas.

Cenário Principal de Sucesso:

1. O utilizador acede ao portal de criação de propostas de formação avançada.

2. Inicia o processo de criação de uma nova proposta de formação avançada fornecendo os campos obrigatórios para a criação da mesma:

    - Qual a configuração para que quer propor;
    - Quais os cursos ou UCs para quais o proponente quer propor;
    - Modalidade;
    - Título;
    - Introdução;
    - Objetivos;
    - Plano de trabalho;
    - Número de vagas (caso seja possível existirem vagas);
    - Data de início e fim de trabalho.

3. Confirma a criação da proposta com a informação preenchida nos campos.

Pré-condições: O responsável pela configuração e manutenção das instância Fenix já definiu quais os perfis que podem submeter propostas de formação avançada.

Pós-condições: A proposta de formação avançada é criada.

Regras de negócio: As permissões de quem pode criar propostas é definida pela escola.

Caso de Uso - UC3: Visualização de propostas de formação avançada

Sumário: O utilizador vê as propostas que criou.

Utilizadores: Pessoa com permissões no sistema Fenix que já criou propostas de formação avançada.

Cenário Principal de Sucesso:

1. O utilizador acede ao portal de propostas e vê as propostas que criou.
2. O utilizador seleciona a proposta que quer ver em maior detalhe.
3. O utilizador vê os detalhes da proposta criada.

Cenário alternativo:

1. Semelhantes ao cenário principal de sucesso, mas neste caso o utilizador pode ver as propostas a que tem acesso. Este cenário é mais comum para os utilizadores que sejam responsáveis por validar propostas.

Pré-condições: O utilizador já criou pelo menos uma proposta de formação avançada.

Pós-condições: O utilizador consegues ver os detalhes da proposta de formação avançada.

Regras de negócio: Dependendo da política da escola, o utilizador apenas pode ver as suas propostas ou as propostas existentes, ou caso seja responsável por validação ou docente pode ter uma visão de mais propostas no sistema.

Caso de Uso - UC4: Edição de propostas de formação avançada

Sumário: O utilizador edita as suas propostas.

Utilizadores: Pessoa com permissões no sistema Fenix que já criou propostas de formação avançada.

Cenário Principal de Sucesso:

1. O utilizador vai ao portal de propostas e vê as propostas que criou.
2. O utilizador seleciona a proposta que quer editar.
3. O utilizador edita as informações da proposta que quer alterar.

4. Por fim, o utilizador confirma a edição da proposta e são guardadas as alterações.

Cenário alternativo:

1. Semelhante ao cenário principal de sucesso, mas neste caso o utilizador pode editar as propostas a que tem acesso. Este cenário é mais comum para os utilizadores que sejam responsáveis por validar propostas.

Pré-condições: O utilizador já criou pelo menos uma proposta de formação avançada.
Pós-condições: A proposta de formação avançada é editada.
Regras de negócio: Dependendo da política da escola, o utilizador apenas pode editar as suas propostas ou as propostas existentes.

Caso de Uso - UC5: Validação das propostas de formação avançada

Neste caso de uso entende-se que quando uma proposta é validada é apenas para um curso ou uma UC. Quando uma proposta é para vários cursos ou UCs esta tem que ser validada individualmente para cada um. No entanto, se por alguma razão a proposta não faz sentido para um curso ou UC ou não seja validada, não interfere com a validação já feita para os outros cursos ou UCs. A validação da proposta pode ter um parecer positivo (validada) ou um parecer negativo (rejeitada).

Sumário: O utilizador valida as propostas pelas quais é responsável por validar.
Utilizadores: Os utilizadores responsáveis pela validação das propostas são atribuídos na configuração de abertura. Desta maneira, a escola pode delegar esta responsabilidade a qualquer utilizador do sistema Fenix, incluindo docentes, alunos, elementos da secretária académica, membros do conselho científico, entre outros.
Cenário Principal de Sucesso:

1. O utilizador acede ao portal de validação de propostas onde vê as propostas das quais ele é responsável por validar.

2. O utilizador escolhe quais as propostas que quer validar (uma ou mais).

3. O utilizador escolhe quais os cursos ou UCs, das propostas selecionadas, que quer validar ou rejeitar (um ou mais). Esta escolha é possível caso o utilizador tenha sido atribuído a mais do que um curso ou UC na configuração de abertura.

4. O utilizador confirma a validação ou rejeitar dos cursos das propostas selecionadas.

5. Por fim, as propostas ficam com os cursos e UCs escolhidas, no passo anterior, validadas ou rejeitadas.

Cenário alternativo: Não aplicável.

Pré-condições: Foi atribuído um responsável pela validação da proposta.

Pós-condições: A proposta é validada, tendo apenas três resultados, validada, rejeitada ou caso o curso ou UC não tenha sido alterada, fica com o estado inicial.

Regras de negócio: Não aplicável.

4.2.2 Candidatura a propostas

O segundo grupo do problema é o processo de candidatura às propostas criadas. É aqui que são apresentadas aos alunos as propostas que foram aprovadas e que estão disponíveis para se candidatarem. Nesta fase também será necessário uma interação do proponente para selecionarem os candidatos para cada uma das suas propostas. Dependendo de cada escola poderá existir uma, duas ou mais fases de candidaturas a propostas. No entanto, só serão apresentadas apenas duas fases, que será o esperado na maior parte das escolas.

Caso de Uso - UC6: Criar configuração de abertura de candidaturas às propostas de formação avançada

Semelhante à criação de configuração para a gestão das proposta de formação avançada, este caso de uso surge para permitir a flexibilidade do módulo de maneira a conseguir responder aos requisitos específicos de cada escola.

Sumário: O utilizador cria uma configuração para um novo período de candidaturas de propostas de formação avançada.

Utilizadores: Administrador do sistema.

Cenário Principal de Sucesso:

1. O utilizador acede ao portal de configurações de abertura às candidaturas propostas de formação avançada.
2. O utilizador inicia o processo de criação de uma nova configuração fornecendo os campos necessários, descrição, se existe ordem de preferência de escolhas, o número máximo de escolhas de propostas, o Workflow a utilizar e escolher o âmbito da configuração (cursos ou UC).
3. O utilizador adiciona os cursos ou UCs para à configuração.
4. O utilizador confirma a finalização da configuração de abertura.

Pré-condições: Já existem UCs ou cursos abertos na instância do sistema para o ano letivo em que o utilizador pretende criar a configuração.

Pós-condições: Fica criado uma configuração de candidaturas para alunos dos cursos ou das UCs escolhidos.

Regras de negócio: Não aplicável.

Caso de Uso - UC7: Candidatar às propostas de formação avançada

Sumário: O utilizador escolhe quais as propostas de formação avançada disponíveis que gostaria de realizar.

Utilizadores: Aluno.

Cenário Principal de Sucesso:

1. O utilizador vê as propostas que estão disponíveis para se candidatar.
2. O utilizador escolhe quais as propostas que se quer candidatar (uma ou mais depende do nº de escolhas) por ordem de preferência.
3. O utilizador pode alterar a ordem de preferência das propostas ou trocar as propostas em que está interessado.
4. Por fim, o utilizador fica com as propostas escolhidas atribuídas à candidatura.

Cenário alternativo: Não existe escolha na ordem de preferência das propostas e o aluno apenas escolas as propostas que se que candidatar, respeitando o número de escolhas possíveis. Não pode alterar a ordem mas pode trocar as propostas escolhidas por outras que tenha mais interesse.

Pré-condições: O aluno reúne as condições necessárias para estar inscrito numa cadeira de formação avançada ou está em condições de transitar para o ano de realização de um trabalho de formação avançada.

Pós-condições: O aluno escolhe, por ordem de preferência, as propostas de formação avançada.

Regras de negócio: O aluno pode escolher, no máximo, o número de propostas definido na configuração de abertura das candidaturas de propostas de formação avançada. O aluno pode ainda editar a escolha das propostas e a sua ordem de preferência, desde que esteja dentro do prazo de candidatura de propostas.

Caso de Uso - UC8: Selecionar aluno candidato para proposta de formação avançada

Sumário: O utilizador faz uma seleção dos alunos interessados nas suas propostas e escolhe qual o aluno que quer que realize a proposta. Esta escolha é feita para cada uma das propostas do proponente.

Utilizadores: Proponente.

Cenário Principal de Sucesso:

1. O utilizador vê os alunos que se candidataram às suas propostas.

2. O utilizador realiza um processo de seleção ao seu critério para escolher o aluno para cada proposta (realizado fora do sistema).

3. O utilizador escolhe, para cada proposta, qual o aluno candidato que a irá realizar. Caso a proposta tenha mais do que uma vaga, o utilizador pode escolher vários candidatos, nunca ultrapassando o máximo de vagas permitido.

4. O utilizador confirma a escolha dos candidatos às propostas.

Cenário alternativo: Caso o proponente não selecione nenhum candidato a proposta passa à próxima fase de candidaturas, caso esta exista.

Pré-condições: Existe, pelo menos, um aluno que se candidatou a, pelo menos, uma proposta do proponente.

Pós-condições: O proponente escolhe os candidatos que ficaram com as propostas.

Regras de negócio: Quando a fase de escolha de propostas por parte dos alunos termina, a informação relativa a cada proposta é disponibilizada a cada proponente. Estes podem entrar em contacto com os alunos, marcando entrevistas. A escolha dos alunos para as propostas é de total responsabilidade dos proponentes. No final, os proponentes têm de selecionar qual o aluno a atribuir a cada proposta.

Caso de Uso - UC9: Escolha final da proposta de formação avançada

Este caso de uso surge do sistema PEIPAL/Cuco, que um acordo mútuo entre a escolha da proposta por parte do aluno e a escolha do candidato por parte do proponente para que a proposta dê origem a um trabalho de formação avançada. Caso o aluno não escolha uma proposta ou caso o proponente não escolha o candidato, não dá origem a um trabalho de formação avançada. Isto permite evitar escolhas erradas ou precipitadas de ambas as partes. Caso exista um acordo de ambas as partes, no final desde caso de uso que dá-se a passagem do processo de candidaturas para o processo de trabalhos de formação avançada.

Sumário: O utilizador realiza uma escolha final das propostas de formação avançada.

Utilizadores: Aluno.

Cenário Principal de Sucesso:

1. O utilizador vê as propostas para quais foi escolhido por parte dos proponentes (dentro do conjunto das propostas que escolheu anteriormente).

2. O utilizador escolhe qual a proposta de formação avançada que quer realizar.

3. Por fim, o utilizador confirma a escolha da proposta que irá realizar.

4. No final da escolha é criado automaticamente um trabalho de formação avançada baseado na proposta escolhida pelo aluno.

Cenário alternativo: Não aplicável.

Pré-condições: O aluno foi selecionado para uma proposta a que se candidatou. O aluno não pode ter nenhuma proposta de formação avançada já atribuída.

Pós-condições: O aluno fica com uma proposta atribuída. A atribuição da proposta origina a criação e atribuição de um trabalho de formação avançada ao aluno.

Regras de negócio: O aluno é quem dá o passo final no processo da escolha da proposta e criação do trabalho de formação avançada.

4.3 Requisitos não funcionais

Os requisitos não funcionais encontrados após a análise de requisitos foram:

1. O módulo deve ser flexível para facilitar a sua adaptação e utilização nas escolas da ULisboa.
2. A interação do utilizador com o sistema deve seguir padrões semelhantes aos padrões de interação já feita pelo mesmo.
3. O código do módulo deve manter a estrutura de organização do Fenix.
4. O módulo tem de ser rápido de maneira a não atrasar o desempenho do sistema.
5. O módulo deve ter em conta questões de segurança de maneira a não permitir que erros de código passem para o utilizador, assim como garantir que cada utilizador pode apenas realizar as funções que lhe compete.

4.4 Desenho

Nesta secção é apresentado o diagrama entidade associação [3] relativo às entidades principais e os modelos para os dois processos, o processo de gestão de propostas de formação avançada e o processo de candidaturas às propostas. Ambos os modelos dos processos devem ser tomados como sugestão e não como um modelo final, visto que as escolas têm necessidades e regras de negócio diferentes, podendo alterar qualquer parte do processo de maneira a responder às suas necessidades.

4.4.1 Diagrama entidade associação

O diagrama de entidade associação [3] está apresentado na *Figura 4.1*. Visto que este trabalho tem o principal foco na gestão de propostas, a entidade da proposta (*Proposal*) é a entidade mais importante do módulo. Esta entidade terá várias associações com outras entidades e será responsável por guardar as informações da proposta. Estas informações incluem o título, os objetivos, a modalidade, duração de trabalho, entre outras.

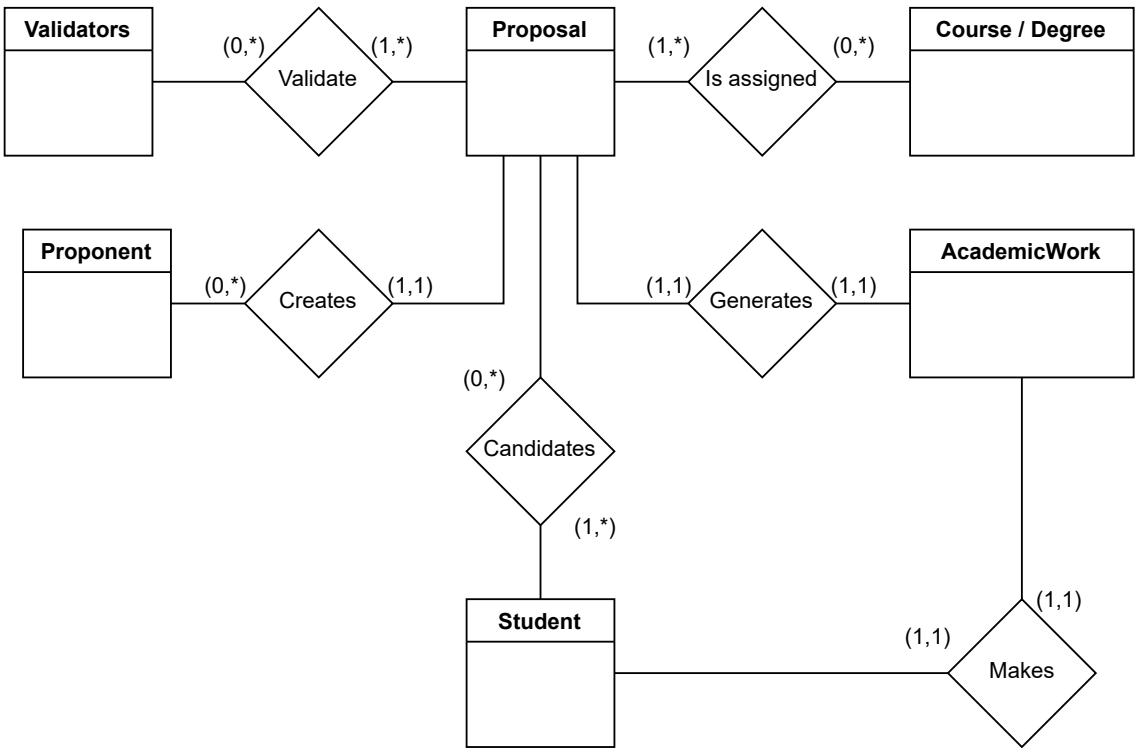

Figura 4.1: Diagrama de Entidade Associação

Cada proposta irá estar ligada a pelo menos um curso ou UC, representados pela entidade *Course/Degree*, o que resulta nas ligações de pelo menos um (Proposal –>Course/Degree) e de zero a muitos (Course/Degree –>Proposal), já que podem existir cursos/UCs que não estejam associados a nenhuma proposta ou associados a várias propostas.

A proposta tem de ser criada por um proponente, originando a relação de um e apenas um (Proposal –>Proponent). Da mesma maneira os proponentes podem criar várias propostas ou até mesmo não criar nenhuma, originando uma relação de zero a muitos (Proponent –>Proposal). Os proponentes poderão ser pessoas internas (professores, investigadores ou alunos), ou pessoas externas (empresas ou instituições), isto irá depender das regras de negócio da escola. Esta separação é apresentada em mais detalhe no *Capítulo 5 - Desenvolvimento e Testes*.

A proposta necessita de ser validada pelos responsáveis antes de ser apresentada aos alunos. Para isto existe uma entidade que representa os responsáveis pela validação (Validators). Estes responsáveis podem ser docentes ou qualquer outra pessoa que a escola entenda que deve ter as permissões para validar as propostas. Estas entidades têm numa associação de pelo menos um (Proposal –>Validators) já que a proposta tem de ser validada por pelo menos um responsável. Tem a associação de zero a muitos (Validators –>Proposal) visto que o responsável pode não ter nenhuma proposta por validar.

O aluno poderá candidatar-se a várias proposta, até a um determinado limite. Este limite é dependente do número máximo que cada escola permite que os alunos se candidatem. Esta escolha associa o alunos às propostas em pelo menos um (Student –>Proposal),

já que os alunos se candidatam a pelo menos uma proposta e de zero a muitos (Proposal –>Student) visto que a proposta pode não receber candidatos.

Por fim existe a entidade de trabalhos académicos (*AcademicWork*). Estas entidade surge quando uma proposta é atribuída ao aluno originando um trabalho académico. Cada proposta só pode dar origem a uma trabalho académico, resultando numa relação de um para um (Proposal<–>AcademicWork). Como cada trabalho académico só pode ser realizado por um aluno o que também dá origem a uma relação de um para um (Student<–>AcademicWork).

4.4.2 Modelo de processo

Seguindo a divisão do problema, como foi feitos nos casos de uso, este fica dividido em dois, a gestão de propostas e as candidaturas às mesmas. Para isto são necessários dois modelos de processo para que se consiga resolver os dois grupos do problema. Estes processos foram modelados para estarem de acordo com os casos de uso acima apresentados. Apesar de serem apresentados os modelos dos dois processos, as escolas podem alterar ou até mesmo criarem os seus próprios processos, como assim entenderem, já que a linguagem de modelação escolhida para representar os modelos de processo foi a do modelo de *Workflow*.

Processo inicial para as propostas

O fluxo inicial para o processo das propostas e das suas candidaturas está apresentado na *Figura 4.2*. Este fluxo é uma simplificação da complexidade do processo de gestão de propostas e de candidaturas de formação avançada.

No início deste fluxo, e quando o período de criação de propostas é aberto, podem ser criadas propostas de formação avançada por parte dos proponentes. As propostas criadas são guardadas na base de dados dos sistema. O fluxo avança para um estado onde é possível editar a proposta, caso tenham sido inseridas informações erradas à sua criação. Após a verificação das informações da proposta, esta é enviada aos responsáveis de validação para a validarem. Neste estado existem duas possibilidades, ou a proposta é rejeitada e é enviada de volta ao proponente para alterar alguma informação, ou é aceite e é marcada para mostrar aos alunos candidatos. Se a proposta for rejeitada e editada, pode ser novamente validada.

Quando o período de candidatura é aberto, os alunos podem criar as suas candidaturas às propostas aceites. O aluno visualiza as propostas disponíveis no sistema e realiza a sua escolha das propostas que tem interesse em realizar. Após esta escolha são apresentados aos proponentes das propostas escolhidas quais os alunos que se candidataram. O proponente pode escolher um aluno candidato para realizar a proposta. Após a escolha por parte do proponente, são apresentas aos candidatos quais as propostas para que foram escolhidos. Os candidatos podem escolher apenas uma proposta para realizar. Esta escolha é

denominada como escolha final. Após a escolha final da proposta por parte do candidato, dá-se início a um processo de trabalho de formação avançada, ou trabalho académico.

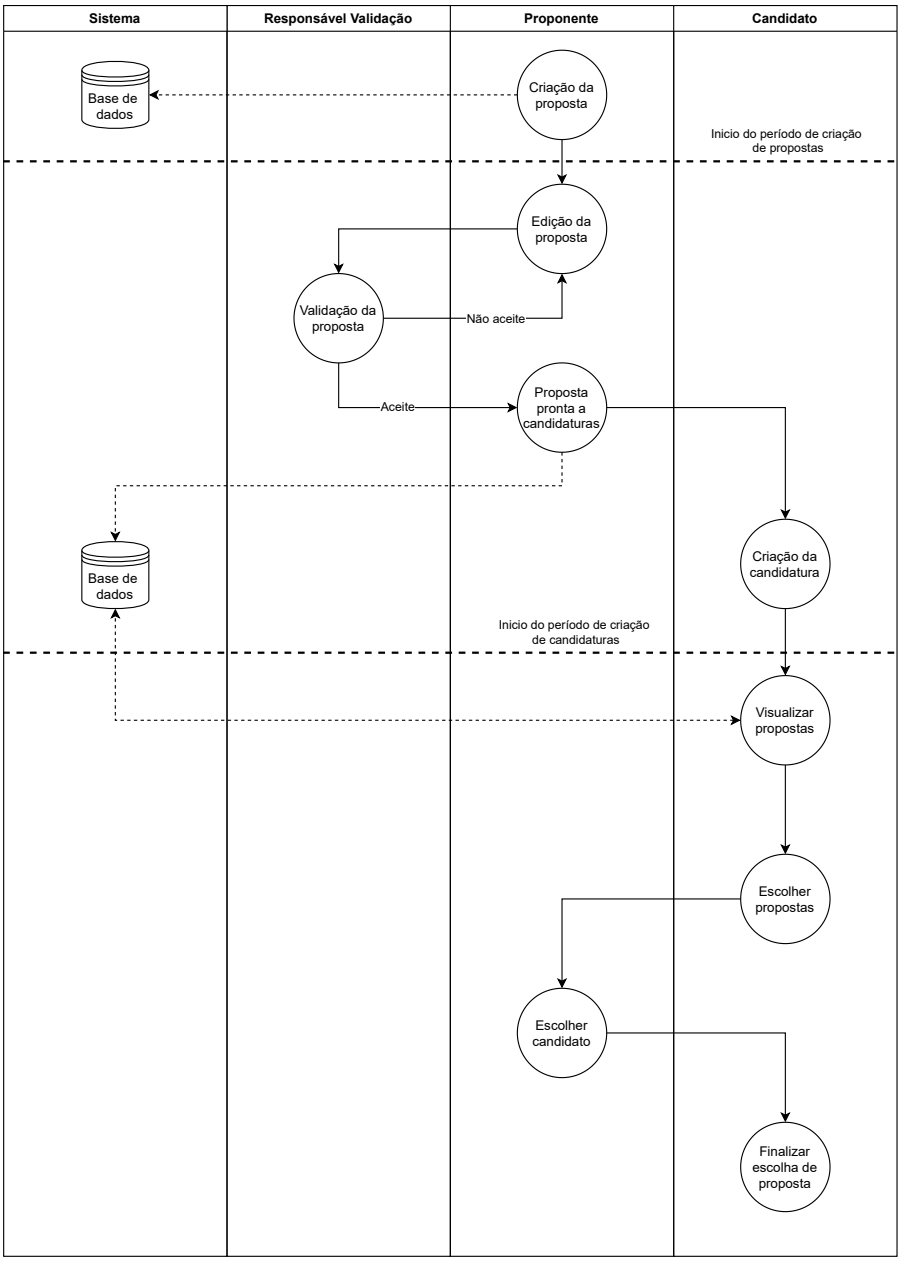

Figura 4.2: Fluxo base do processo

Processo de gestão de propostas

O processo para a gestão de propostas está apresentado na *Figura 4.3*. Este processo começa por verificar se a pessoa que está a tentar criar uma nova proposta de formação avançada tem um perfil que a escola definiu como válido para propor (*Consegue Propor?*). Caso não tenha a proposta não é criada. Caso tenha permissões, o processo avança para um estado em que o proponente pode editar alguma informação que se tenha enganado

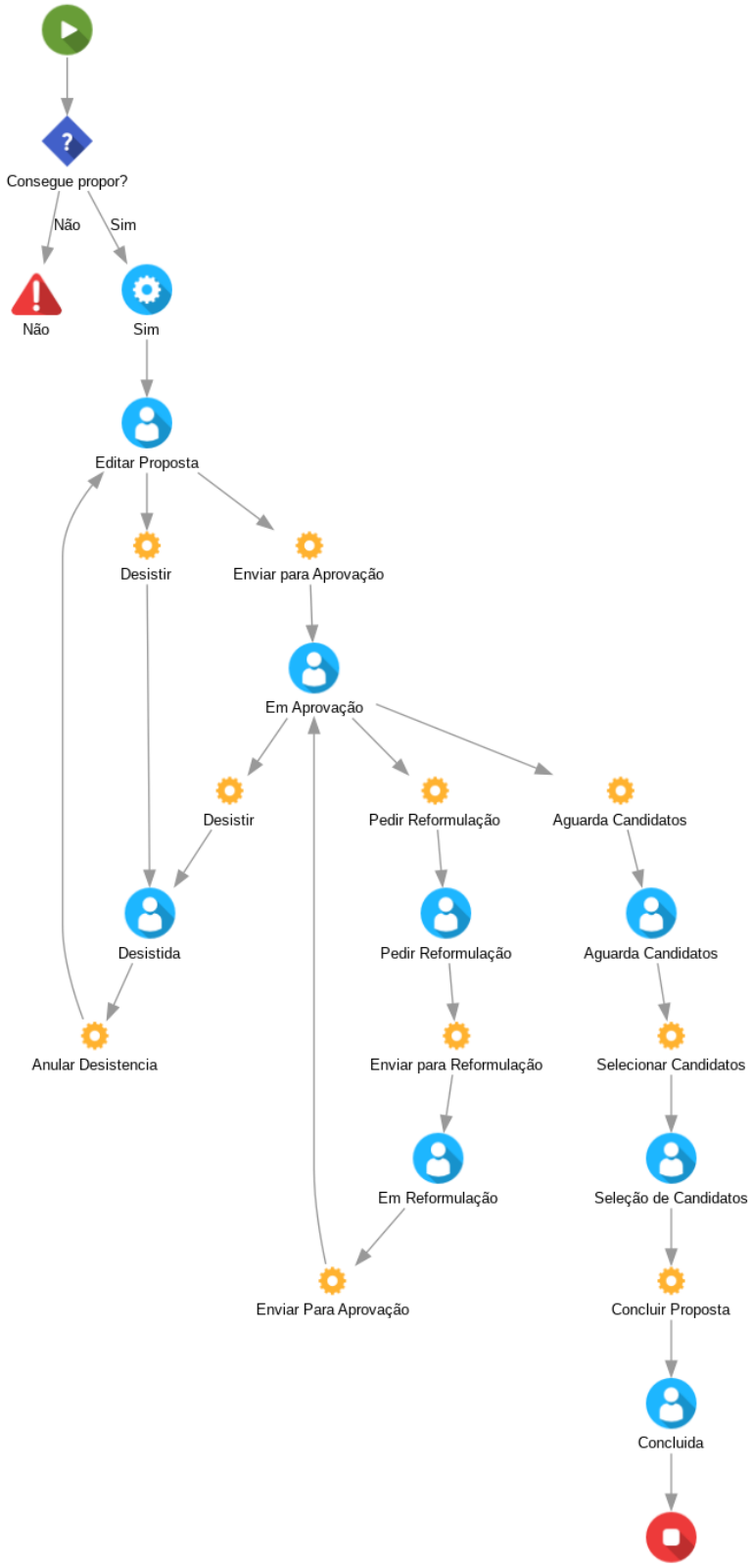

Figura 4.3: Modelo do processo de gestão de propostas

na criação da proposta (*Editar Proposta*). Isto permite que exista uma correção rápida assim que a proposta é criada, evitando que o proponente tenha de apagar a proposta e crie uma proposta nova no caso de algum erro, poupando tempo. A criação de proposta é feita no menu do sistema, mas fora do modelo de *Workflow*. Isto permite que se existir erros na sua criação, como é o caso de não ter permissões para criar propostas, a instância do processo não é criada, evitando propostas erradas ou mal criadas no sistema.

Depois da confirmação das informações da proposta, o proponente envia a proposta para ser validada (*Em Aprovação*). Esta validação é feita pelos utilizadores que tenham sido atribuídos como responsáveis pela validação das propostas. Caso exista alguma informação errada ou em falta, ou até mesmo se o responsável pela validação decidir que a proposta tem o seu conteúdo mal explícito, este pode pedir uma reformulação da proposta. Neste estado (*Pedir Reformulação*) o responsável indica qual a informação a reformular e as razões da mesma e envia a proposta de volta ao proponente para reformulação. É neste estado (*Em Reformulação*) que o proponente pode realizar, caso queira, as alterações sugeridas pelo responsável. Caso não faça as alterações necessárias corre o risco da proposta não ser aceite na validação. Após as alterações a proposta é devolvida ao responsável para realizar uma nova validação (*Em Aprovação*).

Quando a proposta é validada em pelo menos um curso ou UC e o período de validação de propostas termina, a proposta avança para o estado onde aguarda que os alunos se candidatem às propostas (*Aguarda Candidatos*). A candidatura às propostas é feita utilizando outro modelo de processo, pelo que a proposta ficará neste estado até que o outro processo termine.

Após o término do processo de candidaturas, este processo avança para o estado em que o proponente escolhe qual o aluno a que quer atribuir a proposta (*Seleção de Candidatos*). Para isto é apresentada uma lista dos alunos que escolheram a sua proposta. Nesta lista irá aparecer as informações necessárias do aluno para este tomar a sua decisão. Nesta versão do trabalho a escolha dos alunos fica da completa responsabilidade do proponente, podendo este escolher qualquer critério que desejar. Este pode entrar em contacto com os alunos se assim desejar, marcando entrevistas ou escolher com base nas informações apresentadas.

No final da escolha a gestão da proposta termina (*Concluída*). No entanto, ainda falta que o aluno finalize a escolha da proposta que quer realizar no modelo de processo das candidaturas para que seja despoletado o início de um trabalho académico.

Apesar de não estar modelado no processo apresentado, pode acontecer que nenhum aluno selecione a proposta e esta avance para uma fase de candidaturas extra. Visto que as operações existentes permitem a escolha de propostas e candidatos, a sua adição para mais casos poderá ser modelada. Esta modelação faz com que a resolução deste caso específico fique a cargo de cada escola e das suas regras de negócio.

Por fim, e apesar de só estar modelado em dois estados (*Editar Proposta* e *Em aprovação*)

o proponente pode desistir da proposta. Quando a proposta está neste estado (*Desistida*) é considerada como desistida. Esta modelação permite às escolas escolherem onde é permitido que o proponente desista da proposta, podendo ser em alguns estados ou até mesmo em todos. Caso o proponente queira anular a desistência da proposta, basta realizar a operação de anulo da desistência (*Anular Desistência*) e modelar para que estado é que deverá passar.

Processo de candidaturas

O processo para as candidaturas está apresentado na *Figura 4.4*. Em contraste com o modelo anterior, o modelo de candidaturas é bastante mais simples, visto que a gestão das mesmas já foi realizado. Um aluno começa por abrir o seu processo de candidatura. Após a abertura do processo são apresentadas ao aluno as propostas a que se pode candidatar (*Escolher Propostas*). O aluno realiza a escolha de um conjunto de propostas a que se quer candidatar, com limite pré-definido, e confirma a sua escolha. As propostas apresentadas têm de ter em conta o curso ou as UCs que o aluno está inscrito.

Após a confirmação da escolha o processo avança para um estado onde fica à espera que os proponentes façam a seleção dos candidatos (*Aguarda Seleção*). O aluno tem de esperar que esta seleção termine.

Ao terminar o processo avança para o estado em que o aluno finaliza a sua escolha da proposta que quer realizar (*Finalizar Escolha de Proposta*). É apresentada uma lista ao aluno das propostas que foi escolhido pelo proponentes. O aluno só pode escolher uma proposta neste estado. Após a escolha da proposta o processo avança novamente de estado e fica assim concluído.

Existe a possibilidade de o aluno não ter sido aceite para nenhuma proposta escolhida por parte dos proponentes e cabe à escola decidir como tratar deste caso específico. O aluno pode ir para uma segunda fase de candidaturas ou ser atribuída uma proposta que não tenha sido atribuída a nenhum outro aluno. Ambos os casos são possíveis de modelar no processo.

Por fim, e à semelhança com o processo de gestão de propostas, a escola decide se é possível o aluno desistir da candidatura e em que estados o pode fazer. No entanto, neste modelo apenas é permitido que o aluno desista na escolha inicial das propostas (*Escolher Propostas*). Também é da responsabilidade da escola decidir se é possível reverter a desistência e para qual o estado que irá avançar.

4.5 Sumário

Neste capítulo foi apresentada a análise de requisitos necessária para compreender as funcionalidades a serem desenvolvidas. Foram apresentados ainda alguns dos desenhos e modelos a ter em conta no desenvolvimento do trabalho, nomeadamente o diagrama de

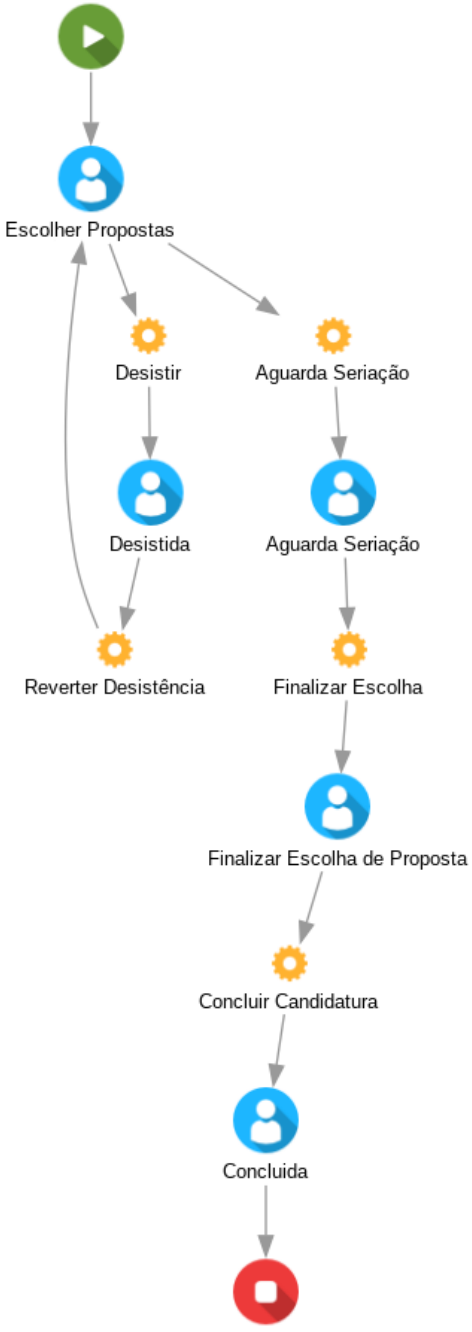

Figura 4.4: Modelo do processo de candidaturas

classe e os modelos de processos. O capítulo seguinte abordará o desenvolvimento das funcionalidades e componentes necessários à utilização do módulo assim como os testes de usabilidade realizados.

Capítulo 5

Desenvolvimento e Testes

Neste capítulo é apresentado o desenvolvimento realizado no módulo de gestão de propostas de formação avançada. É explicado o ambiente de desenvolvimento, os problemas encontrados com o desenvolvimento do módulo para o sistema Fenix e a integração com o *workflow*, assim como soluções encontradas, o diagrama de classes [5] do módulo, as entidades da DSL e os ecrãs PSL, os separadores de *workflow*, as operações de *workflow* criadas. No final do capítulo são apresentados os testes de usabilidade realizados aos docentes (proponentes) e aos alunos (candidatos).

5.1 Ambiente de desenvolvimento

Para o ambiente de desenvolvimento foi criada uma máquina virtual [6] utilizando o sistema operativo *Xubuntu* [12] (v18.04.6 LTS), 16Gb de memória *ram*, processador com 4 *cores* e aproximadamente 100Gb de armazenamento. Foi necessário instalar o serviço de *git* [27], o *OpenJDK* [13] (v11.0.2), o *Maven* [17] (v3.6.x), o *Tomcat* [19] (v9.0.33) e o *Eclipse* [16] (v4.14.0). Para o armazenamento dos dados foi utilizada uma base de dados remota [14] previamente carregada com os dados do ambiente de qualidade da FCUL. A instalação dos componentes para o desenvolvimento do sistema fenix seguem o padrão apresentado neste guia: `https://fenix-ashes.ist.utl.pt/fenixWiki/GettingStartedFenix`.

A empresa *Quorum Born IT* (QubIT) [21] criou uma configuração individual do módulo a desenvolver e bastou realizar o *clone* do projeto, através do *Bitbucket* [7], para o ambiente de desenvolvimento e importar o ficheiro ***pom*** [18] do projeto para o *eclipse*. Foi necessário fazer um *clone* do projeto de *web-app* da empresa, que permite criar e arrancar o servidor de desenvolvimento Fenix com recurso ao *TomCat* [19].

O desenvolvimento do projeto foi realizado utilizando a linguagem *Java*. Foi utilizada a organização em ramos (*branches*), pelo serviço de *Git* [27] separado do ramo principal (*master*) visto que este recebe ramo atualizações e correção de erros pela *QubIT*.

Foram desenvolvidas 144 classes para o módulo com aproximadamente 11500 linhas

de código.

5.2 Diagrama de classes

Ao longo do desenvolvimento do módulo foi necessário realizar algumas alterações ao diagrama de entidade associação apresentado na *Sub-Secção 4.4.1* de maneira a se conseguir garantir os requisitos de flexibilidade levantados. Disto resultou o novo diagrama de classe que apresentado na *Figura 5.5* e na *Figura 5.6*.

Estas alterações surgiram para ultrapassar vários problemas encontrados no decorrer do desenvolvimento do projeto e dividem-se em três partes, a integração com o *workflow*, a flexibilidade de configuração para os requisitos de cada escola e, por fim, a necessidade de transformar alguns atributos em entidades. As relações representadas com as linhas normais são conexões criadas no módulo, enquanto as conexões com linhas tracejadas representam conexões já existentes entre as entidades.

5.2.1 Integração com o workflow

O desenvolvimento deste módulo foi feito para ser integrado com o motor de execução de processos (motor de *workflow*), explicado na *Secção 2.4 - Workflow*, para permitir que cada escola possa modelar os seus processos de maneira a corresponder às suas necessidades.

Para se garantir a correta integração do módulo com o motor de *workflow*, existem algumas relações entre as várias entidades que são necessárias. O excerto do diagrama de classes apresentado na (*Figura 5.1*) permite perceber quais são estas relações.

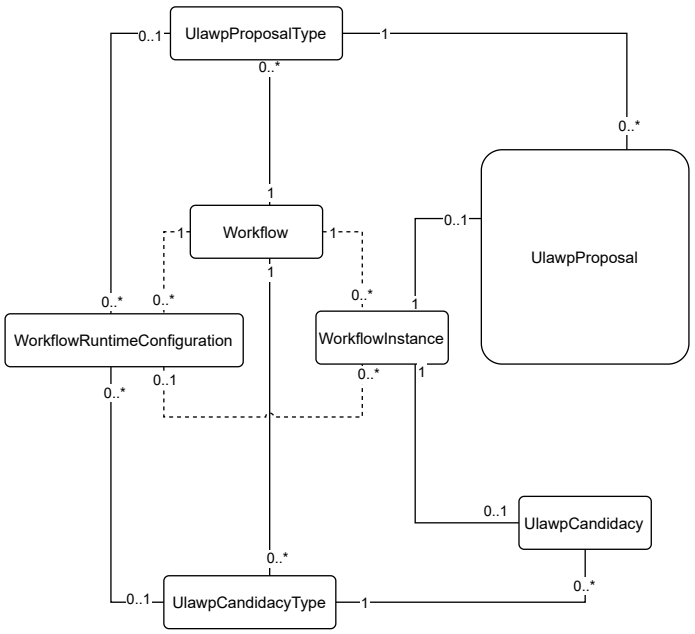

Figura 5.1: Integração com o workflow

Workflow

A entidade de *workflow* tem de ter associações com as configurações de execução (*WorkflowRuntimeConfiguration*), com as instâncias de *workflow* e com os tipos das entidades que serão usadas no *workflow*, a proposta e a candidatura. É no *workflow* que se gere os estados, separadores, transições e operações do modelo de processo.

WorkflowInstance

A instância de *workflow* tem associações com o *workflow*, que é o modelo de processo que irá seguir, com uma configuração de execução, que contém as informações de execuções como as datas de abertura e término ou geradores de números. A instância tem ainda ligações com os entidades que vai gerir, neste caso é com a proposta e com a candidatura.

WorkflowRuntimeConfiguration

A configuração de execução permite às escolas criarem várias execuções dos *workflow* de maneira a utilizar o mesmo modelo de processo em diferentes altura do ano. Nesta configuração é possível especificar qual é o tipo de proposta/candidatura da execução, o seu ano letivo e quais são os geradores de números a utilizar (gere o código único de cada proposta). É possível especificar os períodos a utilizar. Estes permitem criar intervalos de tempo para situações mais detalhadas do processo, como a criação de uma data para uma transição de estados automática.

Proposta/Candidatura

Cada proposta (*UlawpProposal*), ou candidatura (*UlawpCandidacy*), tem de estar associada ao seu tipo. A proposta está associada ao *UlawpProposalType* e a candidatura ao *UlawpCandidacyType*. Estas associações permitem que a proposta e a candidatura tenha acesso às configurações do seu tipo. A proposta e candidatura têm associadas a sua própria instância de *workflow* (*WorkflowInstance*) que as gere.

Tipo de proposta/candidatura

Os tipos de proposta e de candidatura têm associados quais são os modelos de *workflow* a utilizar, as suas configurações de execução e a propostas e candidatura criadas para cada tipo respetivo.

5.2.2 Flexibilidade do módulo

Um dos requisitos do módulo é a flexibilidade para a sua utilização nas várias escolas da ULisboa. Para isto é necessário conseguir configurar o módulo de maneira a responder às

várias necessidades. Para isso foram criadas seis entidades próprias, três para a gestão de propostas e três para as candidaturas.

Para a caso da gestão das propostas foram criadas as entidades *UlawpProposalType*, *UlawpProposalTypeEntry* e *UlawpProposalEntry*. A entidade do tipo de proposta (*UlawpProposalType*) guarda várias informações para a configuração dos tipos de proposta na gestão das propostas. Estas informações incluem qual o número de vagas (por omissão é 1), se o tipo de proposta é para cursos ou é para unidades curriculares e quais as durações e tipos de remuneração possíveis para as propostas criadas para esse tipo. Estas informações permitem definir mais detalhadamente cada tipo de proposta de acordo com o exigido.

A entidade de entrada de tipo de proposta (*UlawpProposalTypeEntry*) representa a associação entre um curso (*Degree*) ou uma unidade curricular (*CompetenceCourse*) e um tipo de proposta. Por outras palavras, é esta entidade que indica quais são âmbitos que estão associados a cada tipo de proposta. Esta entidade guarda uma lista de pessoas responsáveis pela validação das propostas criadas para o âmbito. Isto permite que cada âmbito tenha uma lista individual de responsáveis pela validação. Cada entrada de tipo de proposta tem associada uma lista de modalidades para permitir a escolha das modalidades permitidas por cada âmbito. Por fim, esta entidade tem associada uma lista de entradas de propostas.

A entidade de entrada de proposta (*UlawpProposalEntry*) representa a ligação entre a proposta e um âmbito. Esta entrada tem uma associação com uma entrada de tipo de proposta, que por sua vez está associada a um âmbito. Esta entidade guarda a validação da proposta para o âmbito. É esta validação que indica se a proposta foi ou não aceite para cada âmbito.

Estas entidades e as suas relações estão apresentadas na *Figura 5.2*, juntamente com o respetivo excerto do diagrama entidade associação.

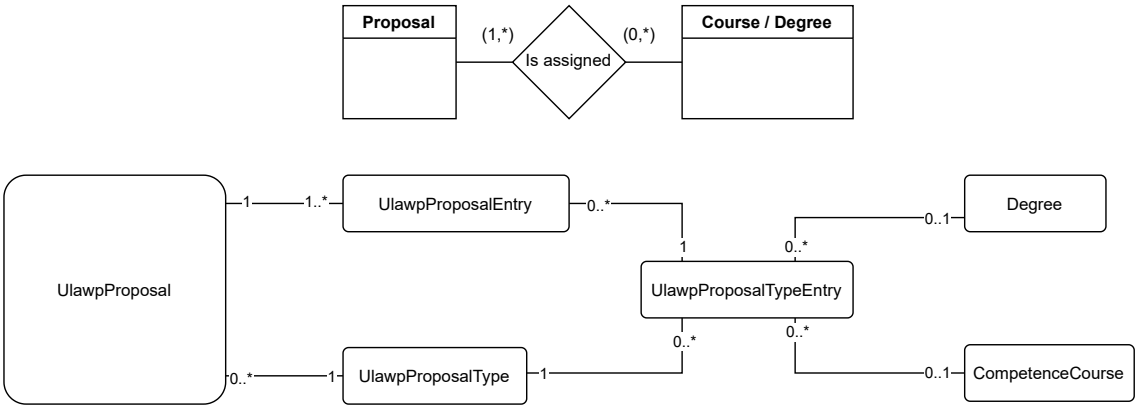

Figura 5.2: Flexibilidade com os tipos de Proposta

Para o caso das candidaturas foram criadas três entidades, a *UlawpCandidacyType*, a *UlawpCandidacyTypeEntry* e a *UlawpCandidacyEntry*.

A entidade de tipo de candidatura (*UlawpCandidacyType*), à semelhança com os tipos de proposta, serve para guardar as várias configurações dos tipos de candidatura. Esta entidade guarda se existe ordenação na escolhas de propostas, qual o número máximo de propostas que um candidato pode escolher, se o tipo de candidatura é para cursos ou para unidades curriculares, e uma lista de entradas de tipo de proposta.

A entidade de entrada de tipo de candidatura (*UlawpCandidacyTypeEntry*) representa a associação entre os tipos de candidatura e os seus âmbitos. Cada entrada de tipo de proposta representa um curso ou uma unidade curricular. Esta entidade serve ainda para filtrar os tipos de candidatura que um aluno se pode candidatar, que obriga a que um aluno só se possa candidatar a um tipo de candidatura que tenha o curso ou unidades curriculares a que está inscrito.

Por fim, existe a entidade de entrada de candidatura (*UlawpCandidacyEntry*). Esta entidade representa a escolha de uma proposta na candidatura de um aluno. Esta entidade guarda a proposta, a candidatura associada, qual a ordem da escolha (caso exista) e dois boleanos, um que representa se o candidato foi aceite pelo proponente da proposta, e outro que representa se o aluno escolheu a proposta como escolha final. Estes dois últimos atributos permitem que seja criado um *double-lock* que garante uma mútua escolha entre o proponente e o aluno, que resultará na criação de um trabalho de formação avançada.

Estas entidades e as suas relações estão apresentadas na *Figura 5.3*, juntamente com o respetivo excerto do diagrama entidade associação.

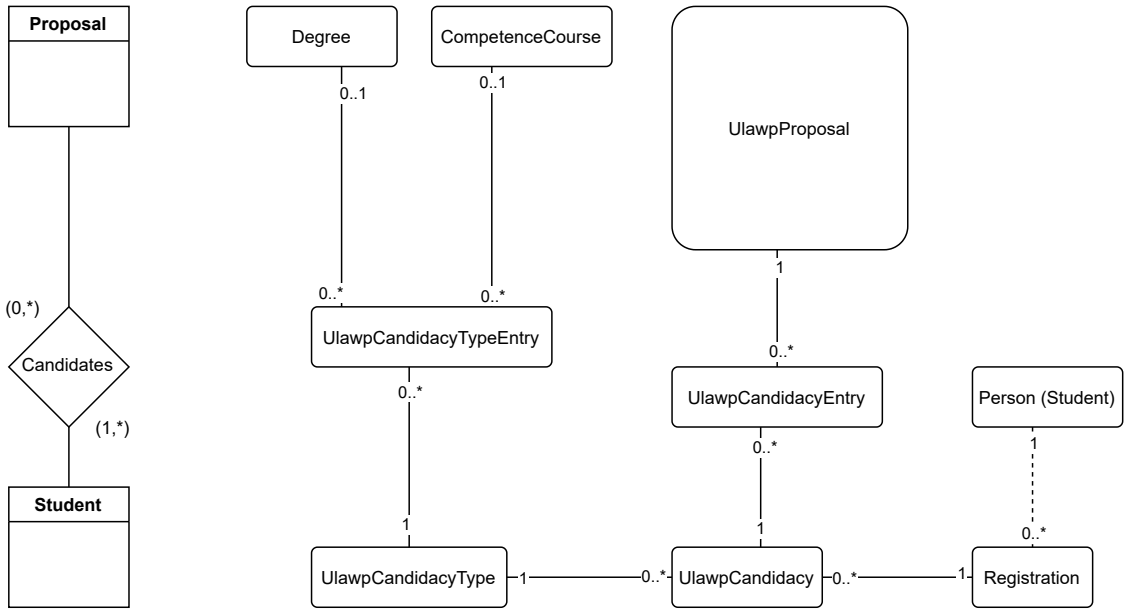

Figura 5.3: Flexibilidade com os tipos de Candidatura

5.2.3 Atributos e novas entidades

Como explicado no início da secção, o desenvolvimento do módulo obrigou a várias alterações nas associações e respetivas entidades. Uma destas alterações foi a criação de novas entidades para substituir atributos. Um destes exemplos é a entidade que representa uma proposta. Esta entidade tinha vários atributos como a modalidade, a remuneração e duração como atributos simples (*Strings, Integers, etc*). Para permitir que cada escola conseguisse designar cada um destes atributos de maneira a serem reconhecidos pelas pessoas intervenientes nos processos, foram criadas várias entidades que representam estes atributos. A entidade de modalidade (*UlawpModality*) para representar as modalidades de uma proposta (trabalho, projeto, estágio, etc). A entidade de remuneração (*UlawpSalary*), que permite atribuir uma remuneração pré-definida às propostas (bolsas FCT, propostas sem remunerações, etc). Estas remunerações pré-definidas são criadas e geridas pela entidade de tipos de remuneração (*UlawpSalaryType*). É a entidade dos tipos de remuneração que permite que cada escola crie as suas próprias remunerações para os candidatos. A entidade de duração da proposta (*UlawpDuration*) representa a duração da proposta (3 meses, 6 meses, 1 ano, etc). Existe ainda a entidade que representa a validação de uma proposta (*UlawpValidationEntry*), que apesar de não estar apresentada na *Figura 5.4* é idêntica às entidades descritas nesta subsecção. Esta entidade guarda dois campos adicionais, se é uma validação inicial, que serve como validação por omissão e se é uma validação que representa que a proposta foi aceite/aprovada. Não é possível ter mais do que um tipo de validação inicial e aceite, de maneira a não surgirem erros no sistema, visto que estes campos são utilizados automaticamente pelo código.

Estas entidades e as suas relações estão apresentadas na *Figura 5.4*, juntamente com o respetivo excerto do diagrama entidade associação.

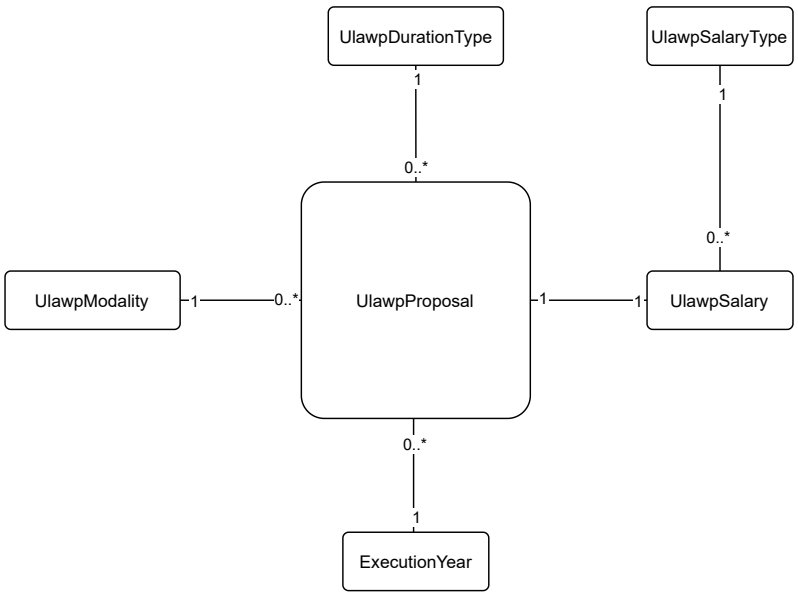

Figura 5.4: Separação das Entidades

5.2.4 Diagrama de classes do módulo

As alterações realizadas ao modelo de entidade associação, feitas para a garantir os requisitos de flexibilidade e a integração com o *workflow*, resultam no diagrama de classes apresentado na *Figura 5.5* e *Figura 5.6*.

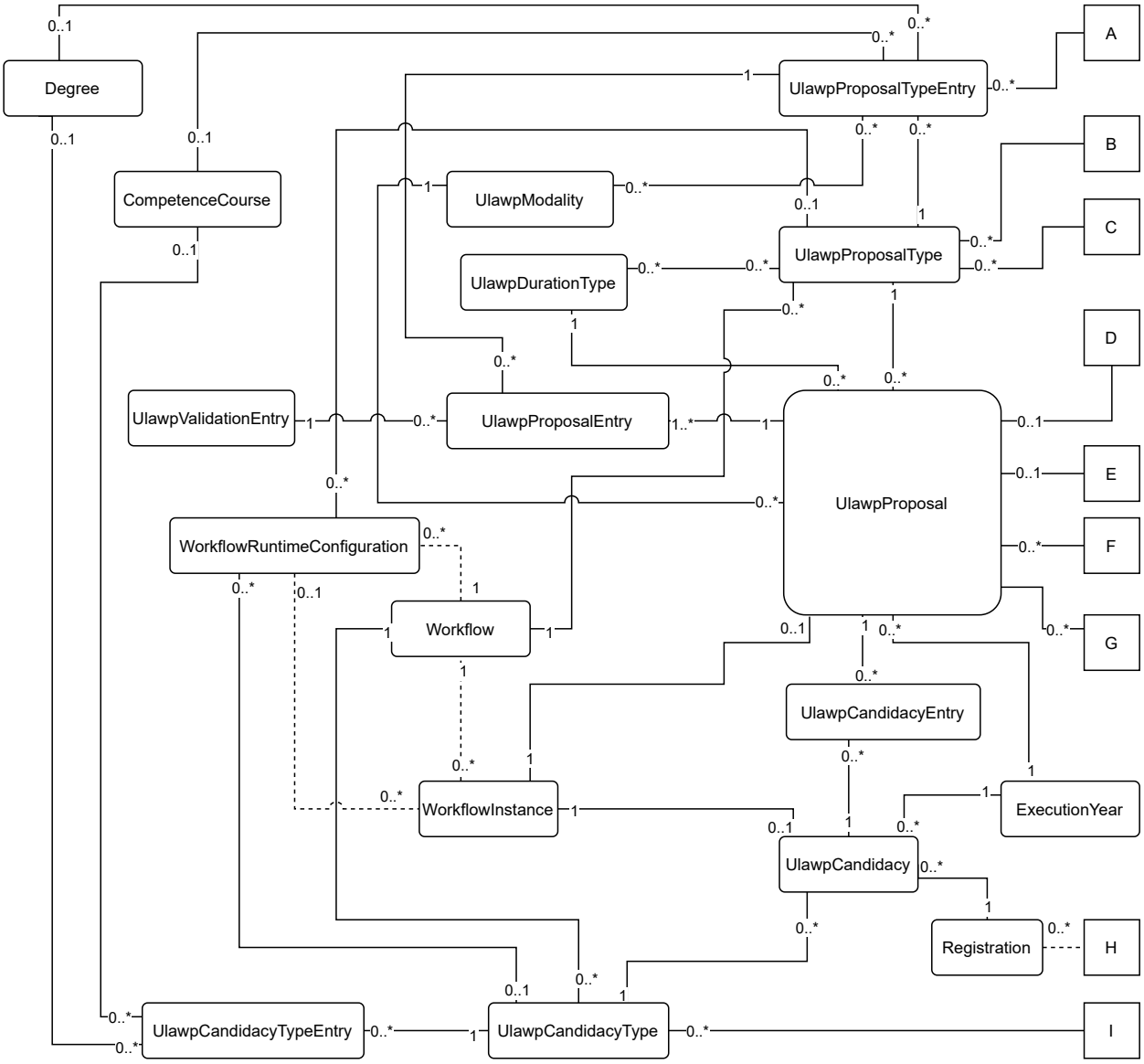

Figura 5.5: Diagrama de Classes

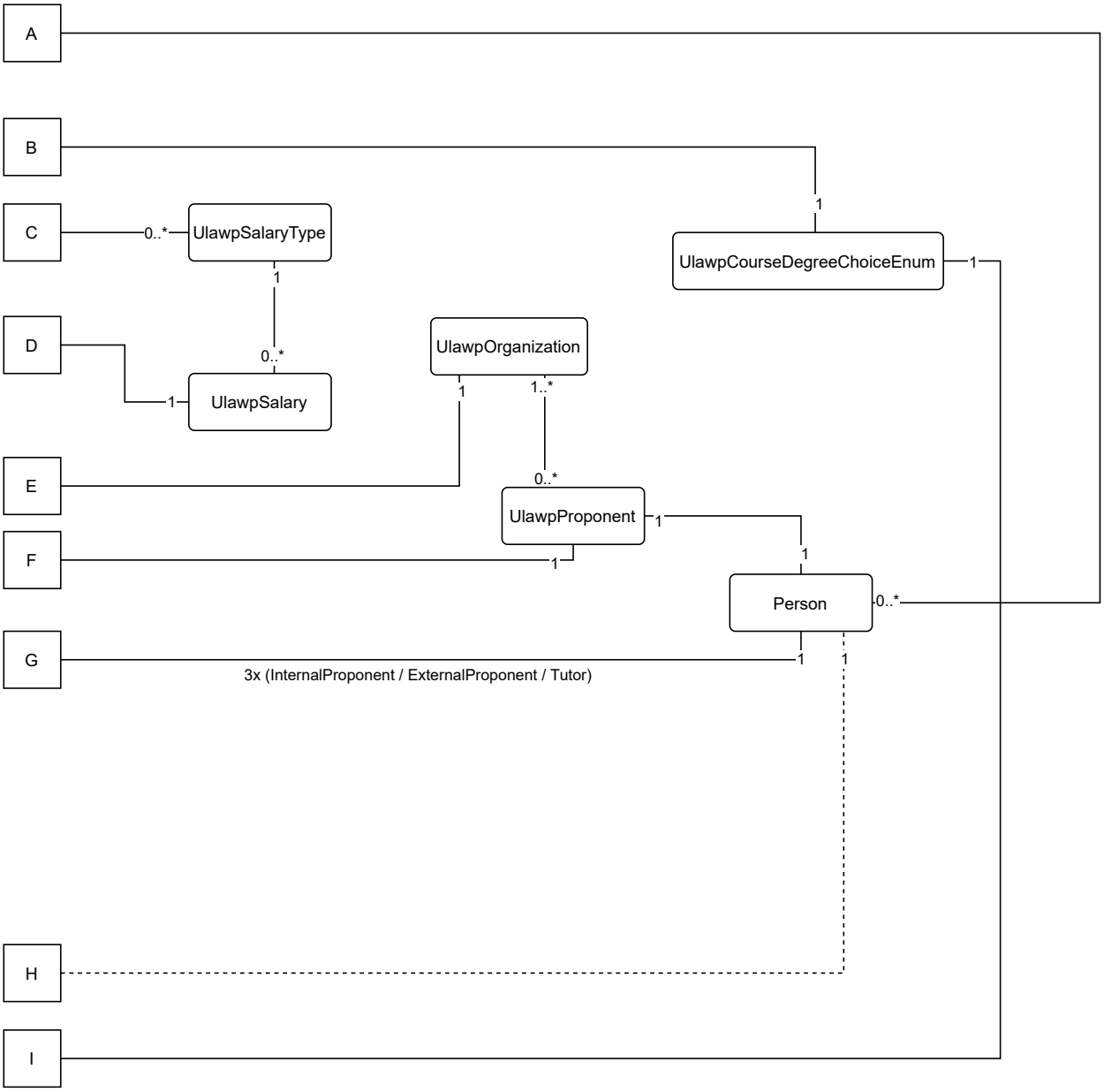

Figura 5.6: Diagrama de Classes

Para se conseguir perceber o diagrama, este não apresenta nem os atributos nem os métodos em cada uma das classes. Estes atributos e métodos estão apresentados na lista[1] abaixo:

- Atributos da proposta –> *Figura A.1*.
- Métodos da proposta –> *Figura A.2*.
- Tipo de proposta –> *Figura A.3*.
- Entrada de tipo de proposta –> *Figura A.4*.
- Entrada proposta –> *Figura A.5*.
- Modalidade –> *Figura A.6*.
- Duração –> *Figura A.7*.
- Validação –> *Figura A.8*.
- Tipo de remuneração –> *Figura A.9*.
- Remuneração –> *Figura A.10*.
- Entidade proponente –> *Figura A.11*.
- Proponente –> *Figura A.12*.
- Candidatura –> *Figura A.13*.
- Entrada de candidatura –> *Figura A.14*.
- Tipo de candidatura –> *Figura A.15*.
- Entrada de tipo de candidatura –> *Figura A.16*.

As classes têm vários métodos comuns como o seu construtor, o *init*, o *verifyInvariant*, o *edit*, o *delete*, o *create* e o *findAll*. Os construtores dos métodos são protegidos de maneira a verificar a informação de criação. Para se criar um objeto utiliza-se o *create*, que por sua vez chama o construtor e o *init*, que introduz (*set*) as informações no objeto. O *verifyInvariant* faz a validação das informações.

[1] As figuras têm a seguinte nomenclatura: Os atributos e métodos públicos (*public*) são representados com "+". Os métodos protegidos (*protected*) são representados com "#". Os métodos estáticos (*static*) estão sublinhados.

5.3 DSL

Nesta secção é apresentada a DSL do módulo e são explicadas as suas entidades. Esta secção é dividida em três partes, a primeira parte explica como foram utilizadas as entidades já existentes no sistema Fenix, a segunda parte explica as entidades criadas para a gestão de propostas, e a terceira parte explica as entidades criadas para as candidaturas às propostas.

As entidades são referidas apenas pelo último nome dado, ignorando o resto do caminho (org.fenixedu.academic.domain.**Person** –> **Person**).

Algumas das entidades têm uma *string* (*code*) e duas *DateTime* (*creationDate* e *lastModificationdate*). Esta *string* serve como identificador único de cada objeto, enquanto as duas datas servem para saber, respetivamente, quando o objeto foi criado e a data da sua última alteração.

Nesta secção as ”entradas”, ou *entry*, representam uma associação entre duas entidades onde se guarda algum tipo de informação.

5.3.1 Entidades do Fenix

No desenvolvimento do módulo foram utilizadas várias entidades já existentes no sistema Fenix. Estas entidades estão apresentadas no *Excerto de código - A.1*.

A *UlawpLocalizedString* possibilita a utilização das *LocalizedString*[2] no módulo.

A entidade *DomainRoot* é única e é a ponto central de ligação de todas as entidades do Fenix, ou seja, todas as entidades do sistema Fenix têm de estar ligadas à *DomainRoot*.

As entidades *Person*, *Degree*, *CompetenceCourse* e *ExecutionYear* são utilizadas por entidades do módulo e representam respetivamente pessoas, cursos, unidades curriculares e anos de execução. São utilizadas ainda as entidades de *WorkflowInstance*, *Workflow* e *WorkflowRuntimeConfiguration* para permitir a implementação do módulo na ferramenta de *workflow*.

As entidades de sistema têm de ter as associações às entidades do novo módulo para garantir uma associação bidirecional.

5.3.2 Entidades de gestão de propostas

Nesta secção são apresentadas as entidades da DSL criadas para a gestão das propostas de formação avançada.

[2] ***LocalizedString*:** Faz uma correspondência entre o valor de uma *string* e língua (Português ou Inglês). Isto permite apresentar dinamicamente os valores corretos das *strings* de acordo com a linguagem de apresentação escolhida no sistema.

Enumerado

O enumerado *UlawpCourseDegreeChoiceEnum* serve para indicar qual o tipo de âmbito que os tipos de proposta e os tipos de candidatura irão utilizar, se são cursos ou unidades curriculares.

Tipo de proposta

A entidade de tipo de propostas é umas das principais entidades que fornece uma flexibilidade de configuração do módulo. Esta entidade tem vários atributos, um nome que será utilizado para fazer a distinção de cada tipo de proposta, o número máximo de vagas que é permitido por proposta (por omissão é um) e um enumerado que indica qual o tipo de âmbito (cursos ou para unidades curriculares). Não é possível um tipo de proposta estar relacionado a cursos e unidades curriculares simultaneamente.

O tipo de proposta está associado às entidades das propostas, que foram criadas para este tipo, e as várias configurações de execução do *workflow*. Esta associação permite que o mesmo tipo de proposta seja utilizado várias vezes em execuções diferentes, evitando que se crie um tipo de proposta novo para cada execução. Cada tipo de proposta tem associado um conjunto de durações e um conjunto de tipos de remuneração. Estes conjuntos servem para limitar quais as possíveis durações e remunerações que cada proposta possa ter. Existe ainda a associação com a entidade de *workflow*. Este *workflow* é o modelo de processo que será utilizado. Por fim, existe a associação com as entradas de tipo de propostas. Estas entradas representam quais os âmbitos (cursos ou unidades curriculares) que cada tipo de proposta terá.

O código da DSL referente ao tipo de proposta está apresentado no *Excerto de código - A.2*.

Entrada de tipo de proposta

A entidade de entrada de tipo de proposta representa a associação entre o tipo de proposta e um curso ou unidade curricular. Esta entidade está associada a um curso ou a uma unidade curricular que a representa, a uma lista de pessoas que são as pessoas encarregues pela validação das propostas que sejam criadas para esse âmbito. Estas entradas têm ainda associada uma lista de modalidades de maneira a limitar quais as modalidades permitidas em cada âmbito. Por fim, existe a associação com as entradas de proposta que representam a ligação entre a proposta e os âmbitos.

O código da DSL referente às entradas de tipo de proposta está apresentado no *Excerto de código - A.4*.

Proposta

A entidade da proposta é a maior entidade em termos de atributos e associações do módulo. Esta entidade tem vários atributos que servem para guardar informação da proposta, como o título, a introdução, os objetivos, o plano de trabalho, observações que o proponente ache relevante, as pré condições para a escolha dos candidatos e o local de trabalho.

A entidade tem ainda outras informações como o número de vagas disponíveis (uma vaga por omissão), o número de vagas preenchidas, se existe um aluno pré-selecionado ou não (só indica se existe ou não, não indica qual é o aluno), se a proposta foi desistida ou não (*active*), qual a data de início do trabalho e se a gestão da proposta já está concluída. Esta última informação serve para saber se a proposta já foi devidamente avaliada e se pode ser apresentada aos alunos. A proposta tem associações com outras entidades como é o caso da modalidade, a duração, a organização proponente, o tipo de proposta, o proponente e a remuneração. A proposta está ainda ligada a várias entidades existentes no sistema Fenix, como é o caso do ano letivo, a instância do *workflow* e as pessoas (Person) que representam o orientador interno, orientador externo e tutor/supervisor. Existe por fim uma lista de entradas de proposta, que representam a associação entre proposta e os âmbitos, e uma lista de entradas de candidatura, que representa as escolhas dos alunos.

O código da DSL relativo à proposta está no *Excerto de código - A.3*.

Entrada de proposta

A entidade de entrada de proposta representa uma ligação entre a proposta a que está associada e a entrada do tipo de proposta (que representa o âmbito). Esta entidade tem um atributo que indica qual a validação da proposta para o âmbito a que está associado.

O código da DSL relativo às entradas de proposta está no *Excerto de código - A.5*.

Validação

Quando as pessoas encarregues pela validação de propostas dão o seu parecer às mesmas, é necessário guardar esta informação. Isto requer que existam objetos que representem estas validações. A entidade de validação guarda o nome da validação, se é uma validação de aceitação ou se é uma validação inicial. A validação inicial é usada para popular as entradas de proposta quando a proposta é criada. A validação de aceitação serve para representar um validação de parecer positivo. Não é permitido, em termos de código, que uma escola crie mais do que uma validação inicial e mais do que uma validação de aceitação. Existe ainda a associação com a entrada de proposta. É esta associação que indica o estado de validação da entrada de proposta.

O código da DSL relativo às validações está presente no *Excerto de código - A.6*.

Modalidade

As propostas têm de ter uma modalidade associada. Esta modalidade indica se a proposta será uma proposta de projeto, estágio, trabalho, dissertação ou outros tipos que a escola permita. Esta entidade está associada às entradas dos tipos de propostas e às propostas. A associação com as entradas dos tipos de proposta permite configurar as entradas dos tipos de proposta mais detalhadamente, enquanto a associação com a proposta apenas representa qual é a modalidade da proposta.

O código da DSL relativo às modalidades está presente no *Excerto de código - A.7*.

Duração

A entidade de duração representa qual será a duração de realização do trabalho de formação avançada. Esta entidade tem apenas um atributo (nome) que a representa. Este nome é apenas textual e não faz nenhuma validação de duração. Isto permite às escolas criarem durações sem qualquer restrição (”3 meses”, ”1 ano”, ”24 horas”, etc). Esta entidade tem associações com os vários tipos de propostas e com as propostas. A associação aos tipos de proposta permite definir nos vários tipos de proposta quais as durações permitidas, enquanto a associação com a proposta representa a sua duração.

O código da DSL relativo às durações está presente no *Excerto de código - A.8*.

Remuneração e tipo de remuneração

A entidade de remuneração representa a remuneração associada a cada proposta enquanto o tipo de remuneração permite que cada escola crie os seus próprios tipos de remuneração.

A entidade de tipo de remuneração guarda o valor da remuneração, e dois atributos boleanos, um que indica se o valor da remuneração é editável ou não, e outro que indica se o valor da remuneração é visível ou não. Isto permite a criação de tipos de remuneração em que o valor é editável, ou seja, é o proponente a definir o valor da remuneração, ou não editável, no caso das remunerações com valor fixo (bolsas de investigação). O campo de visível permite esconder o valor numérico da remuneração. Esta entidade está ligada aos tipos de proposta para se configurar quais os tipos de remuneração permitidos em cada tipo de proposta. A entidade de tipo de remuneração está ainda ligada à entidade de remuneração.

A entidade de remuneração faz uma associação entre um tipo de remuneração e uma proposta. É esta associação que representa a remuneração individual de cada proposta.

Os códigos da DSL relativos aos tipos de remuneração e remuneração estão presentes no *Excerto de código - A.9* e no *Excerto de código - A.10*, respetivamente.

Proponente

As propostas de formação avançada só podem ser criadas por pessoas que tenham as devidas permissões escolares para o fazer, ou seja, só podem ser criadas por pessoas que a escola indique. Estes proponentes podem ser pessoas internas à escola (docentes, investigadores, alunos ou outros) ou pessoas externas à escola, ligadas a instituições que aceitem alunos na realização de trabalhos de formação avançada. Para gerir estes proponentes foi criada a sua própria entidade. Esta entidade está associada à pessoa do Fenix que o proponente representa. Guarda o nome da pessoa (para pesquisas mais rápidas), uma a lista de propostas criadas e uma lista das entidades proponentes a que está associado. Um proponente tem de estar associado a pelo menos uma entidade proponente para conseguir criar propostas. O proponente pode ser considerado como interno ou externo, dependendo das entidades proponentes a ele associadas. É considerado como proponente interno se pelo menos uma entidade proponente associada for interna, caso contrário é considerado como um proponente externo.

O código da DSL relativo aos proponente está presente no *Excerto de código - A.11*.

Entidades proponentes

Como explicado anteriormente, os proponentes têm de estar associados a entidades proponente. Para isso foi criada a sua própria entidade. Esta entidade guarda as informações da entidade proponente como o nome, um endereço de email, uma morada, um número de telefone e a indicação se é uma entidade interna à escola ou não. Existem associações com os seus proponentes e com as propostas por estes criadas. Isto permite associar diretamente a proposta à entidade e saber mais rapidamente quais foram as propostas que cada entidade proponente criou.

O código da DSL relativo às entidades proponentes está presente no *Excerto de código - A.12*.

5.3.3 Entidades de candidatura a propostas

Nesta secção são apresentadas as entidades da DSL criadas para as candidaturas às propostas de formação avançada.

Tipo de candidatura

A entidade de tipo de candidatura, semelhante à entidade de tipo de proposta, permite a criação de várias configurações para as candidaturas às propostas, oferecendo uma flexibilidade de configuração às escolas. Esta entidade tem vários atributos como o nome, utilizado para distinguir os diferentes tipos de candidatura, um número máximo de escolhas, que representa o número máximo de propostas a que o aluno se pode candidatar (por omissão é 1), e dois boleanos um que determina a possibilidade dos alunos ordenarem a

escolha de propostas, e outro que permite aos proponentes verem a escolhas dos alunos que se candidataram às suas propostas. À semelhança do tipo de proposta, esta entidade tem um atributo enumerado para indicar se o tipo de candidatura é para cursos ou para unidades curriculares. Esta entidade está associada às entradas de tipo de candidatura, explicadas abaixo, às candidaturas dos alunos, às configurações de gestão de execução do *workflow*, permitindo a utilização do mesmo tipo de candidatura em execuções diferentes. Por fim, existe ainda a associação com um *workflow*, sendo este o modelo de processo utilizado pelo tipo de candidatura.

O código da DSL referente aos tipos de candidatura está apresentado no *Excerto de código - A.13*.

Entrada de tipo de candidatura

A entidade de entrada de tipo de candidatura representa uma associação entre um âmbito e um tipo de candidatura. Esta entidade têm apenas duas associações, uma com o curso ou unidade curricular que representam e outra com o tipo de candidatura.

O código da DSL referente às entradas de tipo de candidatura está apresentado no *Excerto de código - A.14*.

Candidatura

A entidade candidatura é a entidade que representa as candidaturas dos alunos a um conjunto de propostas. Esta entidade tem dois boleanos, um para indicar se a candidatura foi ou não desistida pelo aluno (*active*), e outro para indicar se a candidatura foi aceite ou não. Uma candidatura é considerada aceite quando o aluno escolheu uma proposta como escolha final e o proponente dessa proposta seleciona o aluno para a realizar. Esta entidade está associada à matricula do aluno candidato (*Registration*), ao ano letivo da candidatura, à sua instância do *workflow* e ao seu tipo de candidatura. Existe ainda uma associação com as entradas de candidatura.

O código da DSL relativo à candidatura está no *Excerto de código - A.15*.

Entrada de candidatura

A entidade das entradas de candidatura representam a escolha de uma proposta na candidatura do aluno. Esta entidade tem associações com a candidatura e com a proposta. Caso o tipo de candidatura permita ordenar a escolha das propostas, é nesta entidade onde se guarda essa informação. Existem ainda dois boleanos, um que indica se o proponente aceitou o aluno para realizar a proposta e outro que indica se o aluno escolheu a proposta como final.

O código da DSL relativo às entradas de candidatura está no *Excerto de código - A.16*.

5.4 PSL

Nesta secção é apresentada a PSL do módulo juntamente com a explicação dos ecrãs e dos separadores de *workflow* criados. Esta apresentação é dividida em três partes, as duas primeiras partes explicam os ecrãs para a gestão de propostas e para as candidaturas às mesmas, enquanto a terceira parte explica os separadores de *workflow* usados na modelação de ambos os processos.

Os ecrãs são referenciados apenas pelo último nome dado, ignorando o resto do caminho (*flow ulawp.proposalManagement.proposal.**searchUlawpProposalGlobal** –> **searchUlawpProposalGlobal***).

5.4.1 Gestão de propostas

Nesta secção são apresentados os ecrãs da PSL criados para a gestão das propostas de formação avançada.

Proposta

A entidade da proposta têm vários ecrãs para a realização da sua gestão. O código da PSL dos ecrãs da proposta está no *Excerto de código - A.17*. Alguns dos ecrãs que tenham as *viewAction* a apontar para o *ManageProposalUI* implicam que a sua ação de visualização foi alterada para entrar diretamente na sua instância de *workflow*.

O ecrã *searchUlawpProposalGlobal* serve para os serviços académicos fazerem a sua gestão das propostas. Esta gestão inclui a procura de propostas, a criação de novas propostas, a alteração do estado do *workflow* da proposta, a visualização do processo e a eliminação individual ou em massa de propostas.

O ecrã *seachUlawpProposalProponent* serve para os proponentes visualizarem as suas propostas criadas, criarem novas propostas, visualizarem o processo individual de cada proposta e eliminarem propostas.

O ecrã *searchUlawpProposalSplitScreen* é o ecrã onde as pessoas encarregues pela validação de propostas a realizam. Este ecrã está dividido verticalmente em duas partes. O lado esquerdo corresponde à procura e a validação em massa das propostas, enquanto o lado direito apresenta as informações detalhadas da proposta selecionada e permite a realizar uma validação individual. Quando se clica numa proposta da lista apresentada no lado esquerdo, a informação dessa proposta aparece no lado direito. A validação em massa utiliza um ecrã (*informationMassValidation*) que apresenta os títulos de todas as propostas selecionadas e pede ao utilizador que confirme da ação de validação.

O ecrã *createUlawpProposal* é o ecrã de criação de novas propostas. Este ecrã apresenta ou esconde alguns dos campos apresentados dependendo do ecrã de acesso anterior. Caso o acesso tenha sido feito através do ecrã de secretária académica são apresentados

mais campos do que os que seriam apresentados se o acesso fosse feito pelo ecrã dos proponentes. Quando a criação da proposta é cancelada, o fluxo segue para o ecrã anterior, apesar de não estar especificado na PSL.

O ecrã *executeUlawpProposalWorkflowOperation* permite avançar estados de *workflow* das propostas sem que seja necessário entrar individualmente em cada uma e realizar a operação. Para isto selecionam-se as propostas que se quer que avancem de estado e é apresentado neste ecrã os estados comuns às propostas. Quando se seleciona o estado, todas as propostas selecionadas avançam para esse estado. Caso não haja nenhum estado em comum, é apresentado uma mensagem a indicar o problema.

O ecrã *confirmationDeleteProposal* pede ao utilizador que confirme a ação de eliminação de uma ou mais propostas quando este as tenta apagar. Este ecrã aparece quando se tenta apagar as propostas em algum dos ecrãs onde é possível esta ação.

Execuções de propostas

O ecrã *manageUlawpProposalTypesExecutions* faz a gestão de execuções dos tipos de propostas. Esta gestão é feita através de um módulo de gestão de execuções de modelos de processos já existente no sistema Fenix. Esta gestão é realizada no ecrã *SearchWorkflowRuntimeConfiguration*.

O código da PSL relativo à gestão de execuções da proposta está no *Excerto de código - A.19*.

Tipos de proposta

O ecrã *searchUlawpProposalType* permite procurar os tipos de proposta criados, visualizar os detalhes de cada um e eliminar tipos de proposta existentes.

O ecrã *createUlawpProposalType* permite a criação de novos tipos de proposta. Existe um enumerado que não está na PSL que representa o tipo de âmbito (cursos ou unidades curriculares) a utilizar no novo tipo de proposta.

O ecrã *readUlawpProposalType* permite visualizar os detalhes de cada tipo de proposta. Este ecrã tem um separador interno (*searchUlawpProposalTypeEntry*) onde é possível realizar a gestão das entradas dos tipos de proposta. Este ecrã é explicado mais abaixo.

O ecrã *updateUlawpProposalType* permite editar o tipo de proposta. Só é possível editar os campos que foram utilizados na sua criação, à exceção do enumerado.

O código da PSL relativo aos tipos de proposta está no *Excerto de código - A.22*.

Entradas do tipos de proposta

O ecrã *searchUlawpProposalTypeEntry*, que está presente como um separador no ecrã de detalhes dos tipos de proposta, é o ecrã onde é realizada a gestão das entradas dos

tipos de proposta. Neste ecrã é possível adicionar entradas de tipos de proposta tendo em conta o âmbito do tipo de proposta. Caso o âmbito utilizado seja de cursos, é utilizado o ecrã *addDegreesProposalType* para procurar os cursos e adicioná-los ao tipo de proposta. Caso o âmbito utilizado seja de unidades curriculares é utilizado o ecrã *addCoursesProposalType* para procurar e adicionar as unidades curriculares ao tipo de proposta. Os eventos de adicionar cursos ou unidades curriculares são mostrados ou escondidos dependendo do âmbito escolhido. Cada entrada de tipo de proposta tem um evento que permite a sua eliminação.

Para adicionar responsáveis de validação nas entrada de tipo de proposta é utilizado um evento próprio, que chama o ecrã *addResponsibleEntry*, para procurar e associar as pessoas. É possível associar os responsáveis a cada entrada de tipo de proposta individualmente ou em massa, selecionando várias entradas e associar os responsáveis. Nos dois casos é possível associar um ou mais responsáveis ao mesmo tempo.

Para remover uma pessoa como responsável de uma entrada de tipo de proposta basta selecionar a entrada e utilizar o evento de remoção de responsável. Este evento utiliza o ecrã *removeResponsibleEntry* onde é apresentada a lista de responsáveis dessa entrada. Seleciona-se o responsável que se que remover e confirma-se a ação. É possível remover os responsáveis de várias entradas ao mesmo tempo, desde que estes estejam em todas as entradas selecionadas. Se não existir uma pessoa em comum nas entradas selecionadas, não é apresentado nenhum responsável para remover.

As modalidades das entradas dos tipos de proposta são geridas pelo ecrã *addModalityEntry*. Este ecrã permite adicionar modalidades às entradas de forma individual ou em massa. Isto permite que os cursos ou unidades curriculares especifiquem quais as modalidades que autorizam, partilhando o mesmo tipo de proposta e o mesmo modelo de *workflow* com os outros âmbitos. Para a remoção de modalidades é utilizado o ecrã *removeModalityEntry*. Semelhante à remoção dos responsáveis, a remoção de modalidades pode ser feita de forma individual, escolhendo qual a entrada e a modalidade a remover, ou de forma massiva, em que selecionam várias entradas e as modalidades em comum a remover.

O código da PSL relativo às entradas dos tipos de proposta está no *Excerto de código - A.23*.

Remuneração

O ecrã *searchUlawpSalaryType* permite procurar os tipos de remuneração, ir para o ecrã de criação de novos tipos de remuneração e ir para o ecrã de detalhe de cada um.

O ecrã *createUlawpSalaryType* serve para a criação de novos tipos de remuneração.

O ecrã *readUlawpSalaryType* serve para visualizar os detalhes de cada tipo de remuneração e para navegar para o ecrã de edição.

O ecrã *updateUlawpSalaryType* serve para editar o tipo de remuneração.

O código da PSL relativo aos tipos de remuneração está apresentado no *Excerto de código - A.20*.

Duração

O ecrã *searchUlawpDurationType* permite procurar as durações, navegar para o ecrã de criação de novas durações e navegar para o ecrã de detalhe de cada uma.

O ecrã *createUlawpDurationType* serve para a criação de novas durações.

O ecrã *readUlawpDurationType* serve para visualizar os detalhes de cada duração e navegar para o ecrã de edição.

O ecrã *updateUlawpDurationType* serve para editar a duração.

O código da PSL relativo às durações está apresentado no *Excerto de código - A.24*.

Modalidade

O ecrã *searchUlawpModality* permite procurar as modalidades, ir para o ecrã de criação de novas modalidades e ir para o ecrã de detalhe de cada uma.

O ecrã *createUlawpModality* serve para a criação de novas modalidades.

O ecrã *readUlawpModality* serve para visualizar os detalhes de cada modalidade e ir para o ecrã de edição.

O ecrã *updateUlawpModality* serve para editar a modalidade.

O código da PSL relativo às modalidades está apresentado no *Excerto de código - A.25*.

Validação

O ecrã *searchUlawpValidationEntry* permite procurar as validações, navegar para o ecrã de criação de novas validações e navegar para o ecrã de detalhe de cada uma.

O ecrã *createUlawpValidationEntry* serve para a criação de novas validações. Só é possível ter uma validação inicial e uma validação aceite no sistema Fenix. Isto previne a utilização da validação errada na validação de uma proposta, por parte dos responsáveis.

O ecrã *readUlawpValidationEntry* serve para visualizar os detalhes de cada validação e navegar para o ecrã de edição.

O ecrã *updateUlawpValidationEntry* serve para editar a validação.

O código da PSL relativo às modalidades está apresentado no *Excerto de código - A.26*.

Proponente

O ecrã *searchUlawpProponent* permite procurar os proponentes do sistema, ir para o ecrã de criação de novos proponentes e navegar para o ecrã de detalhe de cada um.

O ecrã *createUlawpProponent* serve para a criação de novos proponentes. Na criação dos proponentes é obrigatório associar as pessoas a pelo menos uma entidade proponente.

O ecrã *readUlawpProponent* serve para visualizar os detalhes de cada proponente e ir para o ecrã de edição.

O ecrã *updateUlawpProponent* serve para a edição do proponente. Esta edição serve apenas para alterar as entidades proponentes associadas.

O código da PSL relativo aos proponente está apresentado no *Excerto de código - A.27.*

Entidades proponentes

O ecrã *searchUlawpOrganization* permite procurar as entidades proponentes do sistema, navegar para o ecrã de criação de novas entidades proponentes e navegar para o ecrã de detalhe de cada uma.

O ecrã *createUlawpOrganization* serve para a criação de novas entidades proponentes.

O ecrã *readUlawpOrganization* serve para visualizar os detalhes de cada entidade proponente e navegar e navegar para o ecrã de detalhes de cada uma. Neste ecrã existem dois eventos, um que permite adicionar proponentes à entidade e outro que permite remover os proponentes associados. Para adicionar os proponentes é utilizado o seu próprio ecrã (*addProponents*), explicado abaixo. No caso da remoção de proponentes, basta selecionar os proponentes da listra apresentada e realizar a ação.

O ecrã *updateUlawpOrganization* serve para a edição da entidade proponente.

O ecrã *addProponents* serve para adicionar novos proponentes à entidade proponente. Para adicionar os proponentes pesquisa-se o nome das pessoas e selecionam-se as pessoas a associar como proponente da entidade. Este ecrã é apresentado em quando é chamado o evento de adicionar proponentes no ecrã de detalhes da entidade proponente (*readUlawpOrganization*).

O código da PSL relativo às entidades proponentes está apresentado no *Excerto de código - A.28.*

5.4.2 Candidaturas a propostas

Nesta secção são apresentados os ecrãs da PSL criados para a candidatura às propostas de formação avançada.

Candidatura

Os ecrãs que tenham as *viewAction* a apontar para o ecrã *ManageCandidacyUI* implicam que a ação de visualização foi alterada para entrar diretamente na instância de *workflow*, semelhante aos ecrãs da proposta.

O ecrã *searchUlawpCandidacyGlobal* serve para os serviços académicos fazerem a sua gestão das candidaturas dos alunos. Isto inclui a procura de candidaturas, a criação de novas candidaturas, a alteração de estados de *workflow* das candidaturas, a visualização do processo e a eliminação, individual ou em massa, de candidaturas.

O ecrã *seachUlawpCandidacyCandidate* serve para os alunos visualizarem a sua candidatura. Caso não tenham uma candidatura ativa, e esteja no período em que podem criar candidaturas, é neste ecrã onde podem navegar para o ecrã de criação de candidatura. Este ecrã permite ao aluno entrar dentro da instância do processo de *workflow* ou eliminarem a candidatura, se desejarem.

O ecrã *createUlawpCandidacy* é o ecrã de criação de candidaturas de alunos para se candidatarem às propostas de formação avançada. Este ecrã esconde ou apresenta alguns campos dependendo do acesso anterior. Caso o acesso tenha sido realizado pelo ecrã de secretária académica apresenta mais campos do que pelo ecrã dos candidatos. Quando a criação é cancelada é voltado para o ecrã de onde veio, apesar de não estar especificado na PSL.

O ecrã *excecuteUlawpCandidacyWorkflowOperation* permite que as candidaturas avancem de estado sem que seja necessário entrar na instância de cada uma. Este ecrã é utilizado pela secretária académica que seleciona as candidaturas para avançarem de estado. São apresentados, neste ecrã, os estados comuns às candidaturas selecionadas. Quando se seleciona o estado, as candidaturas avançam para esse estado. Caso as candidaturas selecionadas não tenham nenhum estado em comum, é apresentada uma mensagem que indica o problema.

O ecrã *confirmationDeleteCandidacy* serve para confirmar a eliminação das candidaturas.

O código da PSL relativo à candidatura está no *Excerto de código - A.33*.

Execuções de candidaturas

O ecrã *manageUlawpCandidacyTypesExecutions* faz a gestão de execuções dos tipos de candidatura criados ao longo dos anos letivos, semelhante à gestão de execuções de propostas. Esta gestão é feita através de um módulo de gestão de execuções de modelos de processos já existente no sistema Fenix. Esta gestão é realizada no ecrã *SearchWorkflowRuntimeConfiguration*.

O código da PSL relativo à gestão de execuções da candidatura está no *Excerto de código - A.29*.

Tipos de candidatura

O ecrã *searchUlawpCandidacyType* permite procurar os tipos de candidatura, navegar para o ecrã de visualização dos detalhes de cada um e eliminar tipos de candidatura criados.

O ecrã *createUlawpCandidacyType* permite criar novos tipos de candidatura. Existe um atributo no ecrã de criação, que não está na PSL, que é o enumerado que indica qual o tipo de âmbito (cursos ou para unidades curriculares) a usar no tipo de candidatura, semelhante ao ecrã de criação de tipos de proposta.

O ecrã *readUlawpCandidacyType* permite visualizar os detalhes de cada tipo de candidatura. Este ecrã tem um separador interno, o ecrã *searchUlawpCandidacyTypeEntry*, onde é possível fazer a gestão das entradas dos tipos de candidatura, semelhante ao ecrã de detalhes dos tipos de proposta. Este ecrã é explicado mais à frente.

O ecrã *updateUlawpCandidacyType* permite editar o tipo de candidatura. Só é possível editar os campos que foram utilizados na sua criação, à exceção do enumerado do tipo de âmbito.

O código da PSL relativo aos tipos de candidatura está no *Excerto de código - A.30*.

Entradas de tipos de candidaturas

O ecrã *searchUlawpCandidacyTypeEntry*, que está presente como separador no ecrã de detalhes dos tipos de candidatura, é o ecrã onde é feita a gestão das entradas dos tipos de candidatura. Neste ecrã é possível adicionar entradas de tipos de candidatura dependendo do âmbito escolhido na criação do tipo de candidatura. Caso o âmbito seja cursos, é utilizado o ecrã *addDegreesCandidacyType* para procurar os cursos e adicioná-los ao tipo de candidatura. Caso seja unidades curriculares é utilizado o ecrã *addCoursesCandidacyType* para procurar as unidades curriculares e adicioná-las ao tipo de candidatura. Os eventos de adicionar cursos ou unidades curriculares são mostrados ou escondidos dependendo do âmbito escolhido no tipo de candidatura. Cada entrada de tipo de candidatura tem um evento que permite a sua remoção.

O código da PSL relativo às entradas dos tipos de proposta está no *Excerto de código - A.31*.

5.4.3 Separadores

Os separadores de *workflow* são criados na PSL como os outros ecrãs. Estes ecrãs têm mais um caminho (*ulawp.proposalManagement.proposal.**insideWorkflow***) que indica que serão usados como separadores. Cada classe individual criada pela PSL tem de implementar a ***IUILayerWorkflowScreenEditor*** para ter os métodos necessários para um correto funcionamento dentro do *workflow*.

Proposta

O separador *readUlawpProposalInsideWorflow* é o separador de visualização dos detalhes da proposta. Neste separador são apresentados todas as informações da proposta. No fim

deste separador é apresentada a lista de âmbitos da proposta e a respetiva validação de cada um.

O separador *updateUlawpProposalInsideWorkflow* é o separador que permite editar as informações da proposta dentro do *workflow*. Este separador é um ecrã de leitura (*readScreen*) e não de edição (*updateScreen*). Isto serve para quando se entrar no separador os campos de edição não estejam automaticamente abertos. Para se editar a informação da proposta basta clicar no botão de edição para abrir os campos, realizar as alterações necessárias e guardar as alterações realizadas.

O separador *validateScreenInsideWorkflow* serve para a validação dos âmbitos da proposta. Este separador é também utilizado para se poder marcar no modelo de processo qual é o estado em que se pode realizar as validações dos âmbitos das propostas. Na configuração do separador é possível atribuir grupos de participantes que indica quais são os grupos de pessoas têm permissões para realizar a validação.

O separador *readUlawpCandidateInsideWorkflow* serve para os proponentes visualizarem e escolherem os alunos que se candidataram à proposta. Neste separador é apresentada uma grelha com os candidatos. Esta grelha tem quatro campos, o nome dos candidatos, a ordem da escolha da proposta (caso exista), se o proponente já aceitou o aluno e se a proposta foi a escolha final do candidato. Existe um campo extra que permite ao proponente ver mais informações sobre o candidato. Esta visualização das informações apresenta os separadores do processo de candidatura. Ao selecionar um separador este é apresentado numa janela de *pop-up*. Existem ainda dois eventos debaixo da grelha que permite ao proponente aceitar ou rejeitar os candidatos que selecione.

O código da PSL relativo aos separadores das propostas está no *Excerto de código - A.18*.

Remuneração

O separador *searchUlawpSalaryInsideWorkflow* é o separador de *workflow* que trata da remuneração associada à proposta. Este separador mostra qual a remuneração da proposta. Se na criação da proposta não foi atribuído uma remuneração é possível atribuir uma remuneração, neste ecrã, utilizado o evento de criação de remuneração. Se já existir remuneração associada, é possível editar esta remuneração neste separador.

O separador *createUlawpSalaryInsideWorkflow* serve para a criação de uma remuneração para a proposta.

O separador *updateUlawpSalaryInsideWorkflow* serve para editar a remuneração da proposta.

O código da PSL relativo aos separadores das remunerações está no *Excerto de código - A.21*.

Candidatura

O separador *readUlawpCandidacyInsideWorkflow* serve para ver os detalhes da candidatura. Neste separador é apresentado as informações da candidatura, incluindo uma lista das propostas selecionadas pelo aluno no final do separador.

O separador *searchUlawpChosenProposalCandidacySplit* serve para os candidatos verem as propostas de formação avançada disponíveis para se candidatarem. Este separador está dividido verticalmente em duas partes. A parte da esquerda serve para o candidato procurar as propostas a que se pode candidatar, selecionar e escolher quais as propostas a que se quer candidatar. O aluno não pode escolher um número propostas maior do que o número de escolhas máximo definido no tipo de candidatura. Se o tipo de candidatura permitir ordenação de escolhas de proposta, é neste separador que o candidato pode alterar a ordem das suas escolhas. O candidato pode ainda remover as propostas que já não tenha interesse. Ao clicar numa proposta é apresentado na parte direita do ecrã os detalhes da proposta, semelhante à apresentação da informação da proposta no ecrã de validação de propostas.

O separador *searchFinalizeCandidacyChoiceSplit* permite ao candidato finalizar a escolha das propostas na sua candidatura. Este separador apresenta as propostas a que o aluno se candidatou e para as quais foi escolhido, ou seja, os proponentes aceitaram o aluno para a realizar. O candidato só pode escolher apenas uma proposta final. Semelhante ao separador anterior, este também é dividido em duas partes, onde a parte da esquerda apresenta as propostas escolhidas e permite a realização da escolha fina, enquanto a parte da direita apresenta os detalhes da proposta, quando se clica numa proposta.

O código da PSL relativo aos separadores das candidaturas está no *Excerto de código - A.32*.

5.5 Componentes desenvolvidos

Nesta secção são apresentados os componentes criados para a módulo de gestão de propostas de formação avançada. Foram criados várias operações de *workflow* e um componente de apresentação de informação de ecrã.

5.5.1 Componente

No sistema Fenix existem vários componentes que tratam da apresentação dos dados dos objetos presentes no sistema. Este componentes são componentes textuais, componente de multi língua (*LocalizedString*), componente de seleção de datas, entre outros. Para a apresentação das informações da proposta nos ecrãs foi necessário a criação de um novo componente que apresenta todos os detalhes da proposta devidamente organizados. Este componente apresenta o título da proposta destacado no topo do ecrã, seguido das

informações das proposta, como a introdução, os objetivos, modalidades, etc. Caso a informação não exista é apresentado a abreviatura "*N/a*"(Não apresentado / *Not Available*). No final do componente são apresentadas as duas tabelas com os âmbitos da proposta e a respetiva validação. Se a pessoa que está a visualizar o componente não tem permissões para validar os âmbitos são apresentados numa tabela de "Outros âmbitos". Caso a pessoa tenha permissões para validar as propostas, os âmbitos estão apresentados numa tabela mais abaixo que permite a validação dos mesmos.

O código da PSL relativo ao componente de apresentação da proposta está no *Excerto de código - A.34*.

5.5.2 Operações

No sistema Fenix já existem várias operações de *workflow*, porém foi necessário criar seis novas operações para uma melhor gestão do módulo.

Conclusão da gestão da proposta

Esta operação altera o estado da proposta que indica se a proposta ainda está em fase de gestão ou não. Se a proposta já não estiver em fase de gestão significa que já está pronta a ser apresentada aos alunos candidatos.

Desistir da proposta

Esta operação altera o campo boleano ativo (*active*) da proposta para falso e permite que o proponente desista da mesma. Esta operação não apaga a proposta do sistema, apenas altera o boleano da proposta para permitir que se mantenha um histórico das propostas criadas.

Desistir da candidatura

Esta operação altera o campo boleano ativo (*active*) da candidatura para falso do aluno e permite que este desista da candidatura. Esta operação, semelhante à operação anterior, não apaga a candidatura do sistema, apenas troca o boleano de maneira a manter um histórico das candidaturas dos alunos.

Pedido de reformulação

Esta operação é utilizada para realizar um pedido de reformulação da proposta. Esta reformulação pode ser para alguma informação que se encontra incorreta ou incompleta. Como a validação da proposta é feito ao nível de cada âmbito individual, ao existir uma reformulação essas validações têm de ser novamente verificadas com as novas alterações. Esta operação reverte automaticamente as validações da proposta para o valor definido como valor inicial.

Reverter a desistência da proposta

Esta operação altera o campo boleano ativo (*active*) da proposta para verdadeiro e reverte a desistência da mesma. Esta operação não deve ser utilizada pelo proponente mas sim por alguém responsável pelos processos internos da escola, como a secretaria académica ou as pessoas encarregues pela validação de propostas.

Reverter a desistência da candidatura

Esta operação altera o campo boleano ativo (*active*) da candidatura do aluno e reverte a desistência da mesma. Esta operação, à semelhança com a operação anterior, não deve ser utilizada pelo aluno mas sim por alguém responsável pelos processos internos da escola como a secretaria académica.

5.6 Testes

Para a avaliação do módulo foram realizados vários testes de usabilidade utilizando a escala numérica de usabilidade, o ***SUS*** [9, 10] (*System Usability Scale*). Estes testes foram divididos em duas partes, a criação de propostas e a candidatura a propostas.

Para a parte de criação de propostas foram realizados seis testes com seis docentes da FCUL. Neste testes os docentes seguiram um conjunto de instruções para a criação de propostas. Foi pedido que criassem várias propostas para testar as diferentes possibilidades de criação.

Na parte de candidatura às propostas foram realizados seis testes com seis alunos da ULisboa. Nestes testes foi pedido que os alunos seguissem as instruções dadas para se conseguirem candidatar a propostas de formação avançada, previamente criadas para estes testes.

Após a realização dos testes, cada utilizador respondeu a um questionário *SUS* para medir a usabilidade do módulo. Este questionário é composto pelas suas dez questões padrão:

- Q1: Acho que gostaria de utilizar o sistema com frequência.
- Q2: Considerei o sistema mais complexo que o necessário.
- Q3: Achei o sistema fácil de utilizar.
- Q4: Acho que necessitaria de ajuda de um técnico para utilizar o sistema.
- Q5: Considerei que as várias funcionalidades do sistema estavam bem integradas.
- Q6: Achei que o sistema tem muitas inconsistências.

- Q7: Suponho que a maioria das pessoas aprenderia rapidamente a utilizar este sistema.
- Q8: Considerei o sistema muito complicado de utilizar.
- Q9: Senti-me muito confiante ao utilizar o sistema.
- Q10: Precisei de aprender várias coisas novas antes de conseguir usar o sistema.

Em cada pergunta o utilizador dá uma reposta numérica de um (discordo totalmente) a cinco (concordo plenamente). O resultado de cada questionário é calculado através da seguinte formula:

$$\begin{aligned} resultado = ((Q1-1)+(5-Q2)+(Q3-1)+(5-Q4)+(Q5-1) \\ +(5-Q6)+(Q7-1)+(5-Q8)+(Q9-1)+(5-Q10))*2,5 \end{aligned} \tag{5.1}$$

Os resultados dos questionários estão apresentados na *Tabela 5.1* e na *Tabela 5.2*. A coluna final apresenta a média de resultados de cada questionário. Esta média é o resultado final da avaliação *SUS* [8]. Ambos os resultados atingem a nota máxima (**A**), visto que ultrapassam um resultado de 80,3.

Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	**Resultado**	**Total**
5	2	5	1	5	1	5	1	5	1	**97,5**	**85,42**
4	4	2	1	3	2	4	1	5	1	**72,5**	
5	1	5	1	5	1	5	1	5	1	**100**	
5	2	5	1	4	2	4	1	5	1	**90**	
3	2	3	2	3	2	3	3	2	1	**60**	
5	1	4	1	4	1	5	1	4	1	**92,5**	

Tabela 5.1: Resultado do *SUS* dos docentes

Candidato	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	**Resultado**	**Total**
C1	4	2	5	1	4	1	5	1	5	1	**92,5**	**87,5**
C2	4	3	4	1	4	1	3	2	4	1	**77,5**	
C3	5	1	5	1	5	1	5	1	5	1	**100**	
C4	4	1	5	1	4	1	5	1	5	1	**95**	
C5	4	1	4	1	5	1	5	1	5	1	**95**	
C6	2	3	2	1	3	2	4	2	4	1	**65**	

Tabela 5.2: Resultado do *SUS* dos alunos

A *Tabela 5.2* divide os alunos em três categorias, os alunos que utilizaram o PEIpal mas não têm conhecimento interno de Fenix (verde), os alunos que utilizaram o PEIpal e têm conhecimento interno de Fenix (ciano) e os alunos que não utilizaram o PEIpal nem têm conhecimentos de Fenix (laranja). Esta divisão permite verificar se existe uma

diferença na experiência de usabilidade do módulo dependendo o grau de conhecimento do aluno questionado. Os alunos que têm mais experiência com o sistema Fenix não encontraram tantas dificuldades na usabilidade quando comparados com os alunos que não têm essa experiência.

5.7 Sumário

Neste capítulo foi apresentado todo o desenvolvimento realizado ao longo deste projeto que permite às escolas da ULisboa a utilização de um novo módulo integrado no sistema Fenix e no motor de *workflow* para criem os seus próprios processos de gestão de propostas e candidaturas de formação avançada. No capítulo seguinte, e último deste projeto, são apresentadas as conclusões e o trabalho futuro do projeto.

Capítulo 6

Conclusão e trabalho futuro

6.1 Conclusão

Atualmente, o Sistema Integrado de Gestão Académica Fenix da Universidade de Lisboa (**Fenix**) não suporta a gestão dos processos relativos às propostas e às candidaturas de formação avançada. Desta maneira, cada escola da ULisboa tem de realizar as suas próprias gestões, de forma manual ou criando um sistema próprio, como é o caso do Departamento de Informática da Faculdade de Ciências da Universidade de Lisboa.

Deste projeto resulta a criação de um modelo de processo de gestão de propostas de formação avançada e de um processo de gestão de candidaturas às propostas, utilizado como base os procedimentos em uso em algumas faculdades. Para suportar este modelo foi desenvolvido um módulo que permite a gestão de propostas e candidaturas. Este módulo permite que cada escola, departamento, curso ou outro componente da estrutura orgânica da Universidade de Lisboa tenha a capacidade de criar os seus próprios modelos de processo (*workflow*) de maneira a satisfazer as suas necessidades académicas.

O projeto foi constituído por várias fases, a formação sobre o sistema académico, o levantamento de requisitos, o desenvolvimento e os testes de usabilidade. Numa fase inicial existiram várias formações sobre os vários sistemas, ferramentas e *frameworks* do sistema Fenix dadas pelos serviços centrais da Universidade de Lisboa e pela QubIT. O levantamento de requisitos realizou-se com quatro escolas da Universidade de Lisboa, a Faculdade de Ciências, a Faculdade de Farmácia, a Faculdade de Direito e o Instituto de Geografia e Ordenamento do Território. Foi ainda feito um levantamento de requisitos com o Departamento de Informáticas da FCUL. Durante a fase de desenvolvimento foram utilizadas as *frameworks* Fenix, *Bennu* e *OMNIS*, sendo esta última a de maior importância, por ter permitido uma maior rapidez no desenvolvimento do código do projeto. Neste desenvolvimento utilizou-se a linguagem *Java* e linguagens de modelação, nomeadamente a *Domain Specific Language* e a *Presentation Specific Language* num ambiente de desenvolvimento utilizando tecnologias como o *Maven* e o *TomCat*. Também foram elaborados modelos de *workflow* no sistema Fenix, recorrendo a funcionalidades exis-

tentes e criadas neste projeto, para os processos de criação e gestão das propostas de formação avançada e as respetivas candidaturas. Por fim, realizaram-se os testes de usabilidade aos utilizadores finais e avaliaram-se os resultados dos mesmos, recorrendo ao *SUS*, onde se obteve um resultado bastante positivo.

Este projeto foi importante na expansão das minhas capacidades e competências de desenvolvimento de software. Uma das áreas que aprimorei mais foi o desenvolvimento de código na linguagem *Java*.

6.2 Trabalho Futuro

Dada a dimensão do projeto desenvolvido, existem algumas melhorias e funcionalidades que não se enquadram no foco deste projeto, tais como:

- Realizar pesquisas nos ecrãs respetivos de maneira mais rápida - pode ser feito através de filtros pré-definidos ou limitando o número de resultados apresentados, como apresentar os primeiros 50 resultados em vez de serem apresentados todos os resultados;
- Melhoramentos na apresentação de alguns ecrãs onde os utilizadores sentiram dificuldades na navegação, nomeadamente no ecrã de escolha de propostas.
- A criação de pré-propostas - serviria para os proponentes poderem criar propostas de formação avançada antes dos períodos indicados por cada escola, e quando estes períodos abrirem ser possível submeter as pré-propostas criadas, como propostas de formação avançada;
- Nos pedidos de reformulação de proposta, criar entidades que representam um pedido de reformulação que não obriga a reverter todas as validações dos âmbitos da proposta - permite que os responsáveis pela validação de propostas aceitem o pedido de reformulação em vez de realizarem uma nova validação a cada pedido de reformulação;
- A gestão de pessoas e entidades proponentes externas pode ser feita através do sistema Fenix e não apenas com o módulo desenvolvido - o sistema Fenix já tem implementado uma estrutura organizacional de cada escola, departamento e cursos, porém não é possível gerir entidades proponentes e as suas respetivas pessoas externas;
- A gestão de acordos específicos entre as escolas e as entidades para as quais os alunos vão realizar o seu trabalho de formação avançada;
- A criação de trabalhos de formação avançada, ou trabalhos académicos, de forma automática aquando da conclusão do processo de candidatura de formação avançada.

Apêndice A

Apêndices

A.1 DSL

```
enum com.qubit.solution.fenixedu.module.fenixeduUlisboaAcademicWorkProposals.domain.
    ulawp.proposalManagement.enums.UlawpCourseDegreeChoiceEnum as
    UlawpCourseDegreeChoiceEnum;

valueType com.qubit.terra.framework.tools.primitives.LocalizedString as
    UlawpLocalizedString;

reserved entity pt.ist.fenixframework.DomainRoot channels ( WebJava ){
    oneToMany UlawpProposal ulawpProposals mappedBy domainRoot;
    oneToMany UlawpProposalType ulawpProposalType mappedBy domainRoot;
    oneToMany UlawpProponent ulawpProponent mappedBy domainRoot;
    oneToMany UlawpModality ulawpModality mappedBy domainRoot;
    oneToMany UlawpSalaryType ulawpSalaryType mappedBy domainRoot;
    oneToMany UlawpSalary ulawpSalary mappedBy domainRoot;
    oneToMany UlawpCandidacy ulawpCandidacies mappedBy domainRoot;
    oneToMany UlawpCandidacyType ulawpCandidacyTypes mappedBy domainRoot;
    oneToMany UlawpCandidacyTypeEntry ulawpCandidacyTypeEntries mappedBy domainRoot;
    oneToMany UlawpDurationType ulawpDurationType mappedBy domainRoot;
    oneToMany UlawpCandidacyEntry ulawpCandidacyEntries mappedBy domainRoot;
    oneToMany UlawpProposalTypeEntry ulawpProposalTypeEntry mappedBy domainRoot;
    oneToMany UlawpProposalEntry ulawpProposalEntry mappedBy domainRoot;
    oneToMany UlawpValidationEntry ulawpValidationEntry mappedBy domainRoot;
    oneToMany UlawpOrganization UlawpOrganization mappedBy domainRoot;
}

reserved entity org.fenixedu.academic.domain.Person channels ( WebJava ) {
    oneToOne UlawpProponent ulawpProponent mappedBy person;
    oneToMany UlawpProposal ulawpInternalAdvisor  mappedBy internalAdvisor;
    oneToMany UlawpProposal ulawpExternalAdvisor  mappedBy externalAdvisor;
    oneToMany UlawpProposal ulawpTutor mappedBy tutor;
    manyToMany (UlawpProposalTypeEntryPerson) UlawpProposalTypeEntry
    ulawpProposalTypeEntry mappedBy responsible;
}

reserved entity org.fenixedu.academic.domain.Degree channels ( WebJava ){
    oneToMany UlawpProposalTypeEntry ulawpProposalTypeEntry mappedBy degree;
    oneToMany UlawpCandidacyTypeEntry ulawpCandidacyTypeEntry mappedBy degree;
}

reserved entity org.fenixedu.academic.domain.CompetenceCourse channels ( WebJava ) {
    oneToMany UlawpProposalTypeEntry ulawpProposalTypeEntry mappedBy course;
    oneToMany UlawpCandidacyTypeEntry ulawpCandidacyTypeEntry mappedBy course;
}

reserved entity org.fenixedu.academic.domain.ExecutionYear channels ( WebJava ) {
    oneToMany UlawpProposal ulawpProposals mappedBy executionYear;
    oneToMany UlawpCandidacy ulawpCandidacy mappedBy executionYear;
```

```
}

reserved entity com.qubit.qubEdu.module.workflow.domain.runtime.WorkflowInstance
    channels ( WebJava ) {
    oneToOne UlawpProposal ulawpProposal mappedBy workflowInstance;
    oneToOne UlawpCandidacy ulawpCandidacy mappedBy workflowInstance;
}

reserved entity com.qubit.qubEdu.module.workflow.domain.Workflow channels ( WebJava ) {
    oneToMany UlawpProposalType proposalTypes mappedBy workflow;
    oneToMany UlawpCandidacyType ulawpCandidacyType mappedBy workflow;
}

reserved entity com.qubit.qubEdu.module.workflow.domain.WorkflowRuntimeConfiguration
    channels ( WebJava ) {
    manyToOne UlawpProposalType ulawpProposalType;
    manyToOne UlawpCandidacyType ulawpCandidacyType;
}

reserved entity org.fenixedu.academic.domain.student.Registration channels ( WebJava ) {
    oneToMany UlawpCandidacy ulawpCandidacy mappedBy registration;
}
```

Excerto de código A.1: Reserved Entities

```
entity ulawp.proposalManagement.proposal.workflow.UlawpProposalType channels ( WebJava )
    {
    manyToOne DomainRoot domainRoot;
    String code;
    UlawpLocalizedString name;
    Integer numberOfOpenings;
    UlawpCourseDegreeChoiceEnum courseDegreeChoice
    oneToMany UlawpProposal proposals mappedBy proposalType;
    oneToMany WorkflowRuntimeConfiguration workflowRuntimeConfiguration mappedBy
    ulawpProposalType;
    oneToMany UlawpProposalTypeEntry entry mappedBy proposalType;
    manyToMany (UlawpProposalTypeUlawpSalaryType) UlawpSalaryType salaryType;
    manyToMany (UlawpProposalTypeUlawpDurationType) UlawpDurationType durationType;
    manyToOne Workflow workflow;
    DateTime creationDate;
    DateTime lastModificationDate;
}
```

Excerto de código A.2: Proposal Type

```
entity ulawp.proposalManagement.proposal.UlawpProposal channels ( WebJava ) {
    manyToOne DomainRoot domainRoot;
    String code;
    UlawpLocalizedString title;
    UlawpLocalizedString introduction;
    UlawpLocalizedString objectives;
    UlawpLocalizedString workPlan;
    UlawpLocalizedString observation;
    UlawpLocalizedString preCondition;
    UlawpLocalizedString workplace;
    Integer numberOpenings;
    Integer openingsFilled;
    Boolean preSelectedStudent;
    Boolean active;
    Boolean managementConcluded;
    LocalDate beginWorkProposalDate;
    manyToOne UlawpModality modality;
    manyToOne UlawpDurationType durationType;
    manyToOne UlawpOrganization proposingOrganization;
    oneToMany UlawpProposalEntry proposalEntries mappedBy proposal;
    oneToMany UlawpCandidacyEntry candidacyEntry mappedBy proposal;
    manyToOne UlawpProposalType proposalType;
    manyToOne UlawpProponent proponent;
```

```
    oneToOne UlawpSalary salary;
    manyToOne Person internalAdvisor;
    manyToOne Person externalAdvisor;
    manyToOne Person tutor;
    DateTime creationDate;
    DateTime lastModificationDate;
    manyToOne ExecutionYear executionYear;
    oneToOne WorkflowInstance workflowInstance;
}
```

Excerto de código A.3: Proposal

```
entity ulawp.proposalManagement.entry.UlawpProposalTypeEntry channels ( WebJava ) {
    manyToOne DomainRoot domainRoot;
    manyToMany (UlawpProposalTypeEntryPerson) Person responsible;
    manyToOne UlawpProposalType proposalType;
    manyToOne Degree degree;
    manyToOne CompetenceCourse course;
    manyToMany (UlawpProposalTypeEntryUlawpModality) UlawpModality modalities;
    oneToMany UlawpProposalEntry proposalEntry mappedBy proposalTypeEntry;
    DateTime creationDate;
    DateTime lastModificationDate;
}
```

Excerto de código A.4: Proposal Type Entry

```
entity ulawp.proposalManagement.entry.UlawpProposalEntry channels ( WebJava ) {
    manyToOne DomainRoot domainRoot;
    manyToOne UlawpValidationEntry validation;
    manyToOne UlawpProposalTypeEntry proposalTypeEntry;
    manyToOne UlawpProposal proposal;
}
```

Excerto de código A.5: Proposal Entry

```
entity ulawp.proposalManagement.entry.UlawpValidationEntry channels ( WebJava ) {
    manyToOne DomainRoot domainRoot;
    String code;
    UlawpLocalizedString name;
    Boolean acceptedValidation;
    Boolean initialValidation;
    oneToMany UlawpProposalEntry proposalEntry mappedBy validation;
    DateTime creationDate;
    DateTime lastModificationDate;
}
```

Excerto de código A.6: Validation Entry

```
entity ulawp.proposalManagement.modality.UlawpModality channels ( WebJava ) {
    manyToOne DomainRoot domainRoot;
    String code;
    UlawpLocalizedString name;
    manyToMany (UlawpProposalTypeEntryUlawpModality) UlawpProposalTypeEntry entries
     mappedBy modalities;
    oneToMany UlawpProposal proposals mappedBy modality;
    DateTime creationDate;
    DateTime lastModificationDate;
}
```

Excerto de código A.7: Modality

```
entity ulawp.proposalManagement.duration.UlawpDurationType channels ( WebJava ) {
    manyToOne DomainRoot domainRoot;
```

```
3     manyToMany (UlawpProposalTypeUlawpDurationType) UlawpProposalType proposalType
      mappedBy durationType;
4     oneToMany UlawpProposal proposals mappedBy durationType;
5     String code;
6     UlawpLocalizedString name;
7     DateTime creationDate;
8     DateTime lastModificationDate;
9 }
```

Excerto de código A.8: Duration Type

```
1 entity ulawp.proposalManagement.salary.UlawpSalaryType channels ( WebJava ) {
2     manyToOne DomainRoot domainRoot;
3     oneToMany UlawpSalary salaries mappedBy salaryType;
4     manyToMany (UlawpProposalTypeUlawpSalaryType) UlawpProposalType proposalType
      mappedBy salaryType;
5     String code;
6     UlawpLocalizedString name;
7     Boolean editableValue;
8     DateTime creationDate;
9     DateTime lastModificationDate;
10 }
```

Excerto de código A.9: Salary Type

```
1 entity ulawp.proposalManagement.salary.UlawpSalary channels ( WebJava ) {
2     manyToOne DomainRoot domainRoot;
3     manyToOne UlawpSalaryType salaryType;
4     Integer value;
5     oneToOne UlawpProposal proposal mappedBy salary;
6     DateTime creationDate;
7     DateTime lastModificationDate;
8 }
```

Excerto de código A.10: Salary

```
1 entity ulawp.proposalManagement.proponent.UlawpProponent channels ( WebJava ){
2     manyToOne DomainRoot domainRoot;
3     oneToOne Person person;
4     String proponentName;
5     oneToMany UlawpProposal proposalList mappedBy proponent;
6     manyToMany (UlawpProponentUlawpOrganization) UlawpOrganization organization;
7     DateTime creationDate;
8     DateTime lastModificationDate;
9 }
```

Excerto de código A.11: Proponent

```
1 entity ulawp.proposalManagement.organizations.UlawpOrganization channels ( WebJava ){
2     manyToOne DomainRoot domainRoot;
3     String code;
4     UlawpLocalizedString name;
5     String emailAddress;
6     UlawpLocalizedString physicalAddress;
7     String phoneNumber;
8     Boolean internalOrganization;
9     manyToMany (UlawpProponentUlawpOrganization) UlawpProponent proponent mappedBy
      organization;
10    oneToMany UlawpProposal proposal mappedBy proposingOrganization;
11    DateTime creationDate;
12    DateTime lastModificationDate;
13 }
```

Excerto de código A.12: Organization

```
entity ulawp.candidacyManagement.candidacy.workflow.UlawpCandidacyType channels (
    WebJava ){
    manyToOne DomainRoot domainRoot;
    String code;
    UlawpLocalizedString name;
    Boolean choicesExist;
    Integer maximumChoices;
    Boolean seeAllCandidacyChoices;
    UlawpCourseDegreeChoiceEnum courseDegreeChoice;
    oneToMany UlawpCandidacy candidacies mappedBy candidacyType;
    oneToMany UlawpCandidacyTypeEntry entries mappedBy candidacyType;
    DateTime creationDate;
    DateTime lastModificationDate;
    oneToMany WorkflowRuntimeConfiguration workflowRuntimeConfiguration mappedBy
    ulawpCandidacyType;
    manyToOne Workflow workflow;
}
```

Excerto de código A.13: Candidacy Type

```
entity ulawp.candidacyManagement.entry.UlawpCandidacyTypeEntry channels ( WebJava ){
    manyToOne DomainRoot domainRoot;
    manyToOne UlawpCandidacyType candidacyType;
    manyToOne Degree degree;
    manyToOne CompetenceCourse course;
    DateTime creationDate;
    DateTime lastModificationDate;
}
```

Excerto de código A.14: Candidacy Type Entry

```
entity ulawp.candidacyManagement.candidacy.UlawpCandidacy channels ( WebJava ){
    manyToOne DomainRoot domainRoot;
    manyToOne Registration registration;
    Boolean active;
    Boolean accepted;
    oneToMany UlawpCandidacyEntry chosenEntries mappedBy candidacy;
    manyToOne UlawpCandidacyType candidacyType;
    manyToOne ExecutionYear executionYear;
    oneToOne WorkflowInstance workflowInstance;
    DateTime creationDate;
    DateTime lastModificationDate;
}
```

Excerto de código A.15: Candidacy

```
entity ulawp.candidacyManagement.entry.UlawpCandidacyEntry channels ( WebJava ){
    manyToOne DomainRoot domainRoot;
    Integer choiceOrder;
    Boolean proponentAccepted;
    Boolean candidateAccepted;
    manyToOne UlawpCandidacy candidacy;
    manyToOne UlawpProposal proposal;
}
```

Excerto de código A.16: Candidacy Entry

A.2 PSL

```
flow ulawp.proposalManagement.proposal channel (WebJava)  {
```

```
    searchScreen searchUlawpProposalGlobal [UlawpProposal] (fields executionYear
    modality title proponent beginWorkProposalDate durationType) (searchFields
    executionYear modality proponent) {
        createEvent -> createUlawpProposal
        dialog selectMultipleEvent executeOperation ->
    executeUlawpProposalWorkflowOperation
        dialog selectMultipleEvent deleteProposals -> confirmationDeleteProposal
        viewAction //-> ManageProposalsUI
        deleteAction -> searchUlawpProposalGlobal
    }
    searchScreen searchUlawpProposalPropononent [UlawpProposal] (fields active
    executionYear modality title beginWorkProposalDate durationType) (searchFields
    executionYear modality) {
        createEvent -> createUlawpProposal
        dialog selectMultipleEvent deleteProposals -> confirmationDeleteProposal
        viewAction //-> ManageProposalsUI
        deleteAction hidden
    }
    searchScreen searchUlawpProposalSplitScreen [UlawpProposal] (searchFields
    executionYear modality) {
        dialog selectMultipleEvent massValidateProposals -> informationMassValidation
    }
    screen informationMassValidation {
    }
    createScreen createUlawpProposal [UlawpProposal] (fields proposalType
    proposalEntries executionYear modality title introduction objectives workPlan
    numberOpenings beginWorkProposalDate durationType observation preCondition workplace
     preSelectedStudent) {
        createEvent // -> ManageProposalUI
        cancelEvent
    }
    screen executeUlawpProposalWorkflowOperation {
    }
    screen confirmationDeleteProposal {
    }
}
```

Excerto de código A.17: Proposal

```
flow ulawp.proposalManagement.proposal.insideWorkflow channel(WebJava) {
    readScreen readUlawpProposalInsideWorkflow [UlawpProposal] (fields executionYear
    modality title introduction objectives workPlan numberOpenings beginWorkProposalDate
     durationType observation preCondition workplace preSelectedStudent proponent
    proposingOrganization salary creationDate lastModificationDate proposalEntries) {
        backEvent hidden
        updateEvent hidden
    }
    readScreen updateUlawpProposalInsideWorkflow [UlawpProposal] (fields modality title
    introduction objectives workPlan numberOpenings beginWorkProposalDate durationType
    observation preCondition workplace preSelectedStudent) {
        backEvent hidden
        updateEvent
    }
    readScreen validateScreenInsideWorkflow [UlawpProposal] (fields active executionYear
     modality title introduction objectives workPlan numberOpenings
    beginWorkProposalDate durationType observation preCondition workplace
    preSelectedStudent proponent proposingOrganization creationDate lastModificationDate
     proposalEntries) {
    }
    readScreen readUlawpCandidateInsideWorkflow [UlawpProposal] (fields candidacyEntries
    ) {
        backEvent hidden
        updateEvent hidden
        deleteEvent hidden
        event acceptCandidates
        event declineCandidates
    }
}
```

Excerto de código A.18: Proposal Inside Workflow

```
flow ulawp.proposalManagement.proposal.execution channel(WebJava) {
    searchScreen manageUlawpProposalTypesExecutions [UlawpProposalType] (fields name
    workflow) {
        backEvent hidden
        createEvent hidden
        viewAction //-> SearchWorkflowRuntimeConfiguration
        deleteAction
    }
}
```

Excerto de código A.19: Proposal Execution

```
flow ulawp.proposalManagement.salary channel (WebJava)  {
    searchScreen searchUlawpSalaryType [UlawpSalaryType] (fields name editableValue) (
    searchFields editableValue) {
        createEvent -> createUlawpSalaryType
        viewAction -> readUlawpSalaryType
        deleteAction -> searchUlawpSalaryType
    }
    createScreen createUlawpSalaryType [UlawpSalaryType] (fields name editableValue) {
        createEvent -> searchUlawpSalaryType
        cancelEvent -> searchUlawpSalaryType
    }
    readScreen readUlawpSalaryType [UlawpSalaryType] (fields code name editableValue
    creationDate lastModificationDate) {
        backEvent -> searchUlawpSalaryType
        updateEvent -> updateUlawpSalaryType
        deleteEvent -> searchUlawpSalaryType
    }
    updateScreen updateUlawpSalaryType [UlawpSalaryType] (fields name editableValue ) {
        updateEvent -> readUlawpSalaryType
        cancelEvent -> readUlawpSalaryType
    }
}
```

Excerto de código A.20: Salary

```
 flow ulawp.proposalManagement.salary.insideWorkflow channel(WebJava) {
    searchScreen searchUlawpSalaryInsideWorkflow [UlawpSalary] (fields salaryType
    creationDate lastModificationDate) {
        dialog createEvent -> createUlawpSalaryInsideWorkflow
        deleteAction
        dialog action edit -> updateUlawpSalaryInsideWorkflow
    }
    createScreen createUlawpSalaryInsideWorkflow [UlawpSalary] (fields salaryType value)
     {
        createEvent
        cancelEvent
    }
    updateScreen updateUlawpSalaryInsideWorkflow [UlawpSalary] (fields salaryType value)
     {
        updateEvent
        cancelEvent
    }
}
```

Excerto de código A.21: Salary Inside Workflow

```
flow ulawp.proposalManagement.proposalType channel (WebJava) {
    searchScreen searchUlawpProposalType [UlawpProposalType] (fields name workflow) (
    searchFields workflow) {
```

```
3         createEvent -> createUlawpProposalType
4         viewAction -> readUlawpProposalType
5         deleteAction -> searchUlawpProposalType
6     }
7     createScreen createUlawpProposalType [UlawpProposalType] (fields name workflow
      numberOfOpenings durationType salaryType) {
8         createEvent -> readUlawpProposalType
9         cancelEvent -> searchUlawpProposalType
10    }
11    readScreen readUlawpProposalType [UlawpProposalType] (fields code name workflow
      numberOfOpenings creationDate lastModificationDate) {
12        backEvent -> searchUlawpProposalType
13        updateEvent -> updateUlawpProposalType
14        deleteEvent -> searchUlawpProposalType
15    }
16    updateScreen updateUlawpProposalType [UlawpProposalType] (fields name workflow
      numberOfOpenings salaryType durationType) {
17        updateEvent -> readUlawpProposalType
18        cancelEvent -> readUlawpProposalType
19    }
20 }
```

Excerto de código A.22: Proposal Type

```
1  flow ulawp.proposalManagement.proposalType.entry channel (WebJava){
2     searchScreen searchUlawpProposalTypeEntry [UlawpProposalTypeEntry] (fields
      modalities responsible) {
3         dialog selectMultipleEvent addModality -> addModalityEntry
4         dialog selectMultipleEvent removeModality -> removeModalityEntry
5         dialog selectMultipleEvent addResponsible -> addResponsibleEntry
6         dialog selectMultipleEvent removeResponsible -> removeResponsibleEntry
7         dialog event addCourses -> addCoursesProposalType
8         dialog event addDegrees -> addDegreesProposalType
9         viewAction hidden
10        deleteAction -> readUlawpProposalType@ulawp.proposalManagement.proposalType
11    }
12    searchScreen addCoursesProposalType [CompetenceCourse] (fields code) (searchFields
      code name) {
13        selectMultipleEvent addCourse
14    }
15    searchScreen addDegreesProposalType [Degree] (fields code) (searchFields code) {
16        selectMultipleEvent addDegree
17    }
18    searchScreen addResponsibleEntry [Person] (fields presentationName) (searchFields
      displayName) {
19        event addResponsible
20    }
21    searchScreen removeResponsibleEntry [Person] (fields presentationName) {
22        event removeResponsible
23    }
24    searchScreen addModalityEntry [UlawpModality] (fields name) {
25        event addModalities
26    }
27    searchScreen removeModalityEntry [UlawpModality] (fields name) {
28        event removeModalities
29    }
30 }
```

Excerto de código A.23: Proposal Type Entry

```
1  flow ulawp.proposalManagement.duration channel (WebJava)  {
2     searchScreen searchUlawpDurationType [UlawpDurationType] (fields name) {
3         createEvent -> createUlawpDurationType
4         viewAction -> readUlawpDurationType
5         deleteAction -> searchUlawpDurationType
6     }
7     createScreen createUlawpDurationType [UlawpDurationType] (fields name) {
8         createEvent -> searchUlawpDurationType
```

```
9           cancelEvent -> searchUlawpDurationType
10      }
11      readScreen readUlawpDurationType [UlawpDurationType] (fields code name creationDate
        lastModificationDate) {
12          backEvent -> searchUlawpDurationType
13          updateEvent -> updateUlawpDurationType
14          deleteEvent -> searchUlawpDurationType
15      }
16      updateScreen updateUlawpDurationType [UlawpDurationType] (fields name) {
17          updateEvent -> readUlawpDurationType
18          cancelEvent -> readUlawpDurationType
19      }
20  }
```

Excerto de código A.24: Duration

```
1   flow ulawp.proposalManagement.modality channel (WebJava)  {
2       searchScreen searchUlawpModality [UlawpModality] (fields name) {
3           createEvent -> createUlawpModality
4           viewAction -> readUlawpModality
5           deleteAction -> searchUlawpModality
6       }
7       createScreen createUlawpModality [UlawpModality] (fields name) {
8           createEvent -> searchUlawpModality
9           cancelEvent -> searchUlawpModality
10      }
11      readScreen readUlawpModality [UlawpModality] (fields code name creationDate
        lastModificationDate) {
12          backEvent -> searchUlawpModality
13          updateEvent -> updateUlawpModality
14          deleteEvent -> searchUlawpModality
15      }
16      updateScreen updateUlawpModality [UlawpModality] (fields name) {
17          updateEvent -> readUlawpModality
18          cancelEvent -> readUlawpModality
19      }
20  }
```

Excerto de código A.25: Modality

```
1   flow ulawp.proposalManagement.validationEntry channel (WebJava)  {
2       searchScreen searchUlawpValidationEntry [UlawpValidationEntry] (fields name
        acceptedValidation initialValidation) (searchFields acceptedValidation
        initialValidation) {
3           createEvent -> createUlawpValidationEntry
4           viewAction -> readUlawpValidationEntry
5           deleteAction -> searchUlawpValidationEntry
6       }
7       createScreen createUlawpValidationEntry [UlawpValidationEntry] (fields name
        acceptedValidation initialValidation) {
8           createEvent -> searchUlawpValidationEntry
9           cancelEvent -> searchUlawpValidationEntry
10      }
11      readScreen readUlawpValidationEntry [UlawpValidationEntry] (fields code name
        acceptedValidation initialValidation creationDate lastModificationDate) {
12          backEvent -> searchUlawpValidationEntry
13          updateEvent -> updateUlawpValidationEntry
14          deleteEvent -> searchUlawpValidationEntry
15      }
16      updateScreen updateUlawpValidationEntry [UlawpValidationEntry] (fields name
        acceptedValidation initialValidation) {
17          updateEvent -> readUlawpValidationEntry
18          cancelEvent -> readUlawpValidationEntry
19      }
20  }
```

Excerto de código A.26: Validation Entry

```
flow ulawp.proposalManagement.proponent channel (WebJava)  {
    searchScreen searchUlawpProponent [UlawpProponent] (fields proponentName
    organization) (searchFields proponentName organization) {
        createEvent -> createUlawpProponent
        viewAction -> readUlawpProponent
        deleteAction -> searchUlawpProponent
    }
    createScreen createUlawpProponent [UlawpProponent] (fields person organization) {
        createEvent -> searchUlawpProponent
        cancelEvent -> searchUlawpProponent
    }
    readScreen readUlawpProponent [UlawpProponent] (fields proponentName organization
    proposalList creationDate lastModificationDate) {
        backEvent -> searchUlawpProponent
        updateEvent -> updateUlawpProponent
        deleteEvent -> searchUlawpProponent
    }
    updateScreen updateUlawpProponent [UlawpProponent] (fields organization) {
        updateEvent -> readUlawpProponent
        cancelEvent -> readUlawpProponent
    }
}
```

Excerto de código A.27: Proponent

```
flow ulawp.proposalManagement.organizations channel (WebJava)  {
    searchScreen searchUlawpOrganization [UlawpOrganization] (fields name
    internalOrganization) (searchFields internalOrganization) {
        createEvent -> createUlawpOrganization
        viewAction -> readUlawpOrganization
        deleteAction -> searchUlawpOrganization
    }
    createScreen createUlawpOrganization [UlawpOrganization] (fields name emailAddress
    physicalAddress phoneNumber internalOrganization) {
        createEvent -> searchUlawpOrganization
        cancelEvent -> searchUlawpOrganization
    }
    readScreen readUlawpOrganization [UlawpOrganization] (fields code name emailAddress
    physicalAddress phoneNumber internalOrganization proponent creationDate
    lastModificationDate) {
        backEvent -> searchUlawpOrganization
        updateEvent -> updateUlawpOrganization
        deleteEvent -> searchUlawpOrganization
        dialog event addProponents -> addProponents
        dialog event removeProponents
    }
    updateScreen updateUlawpOrganization [UlawpOrganization] (fields name emailAddress
    physicalAddress phoneNumber internalOrganization){
        updateEvent -> readUlawpOrganization
        cancelEvent -> readUlawpOrganization
    }
    searchScreen addProponents [Person] (fields presentationName) (searchFields
    displayName) {
        event addProponents
    }
}
```

Excerto de código A.28: Organizations

```
flow ulawp.candidacyManagement.candidacy.execution channel (WebJava) {
    searchScreen manageUlawpCandidacyTypesExecutions [UlawpCandidacyType] (fields name
    workflow) {
        backEvent hidden
        createEvent hidden
        viewAction //-> SearchWorkflowRuntimeConfiguration
        deleteAction
    }
```

```
}
```

Excerto de código A.29: Candidacy Execution

```
flow ulawp.candidacyManagement.candidacyType channel (WebJava) {
    searchScreen searchUlawpCandidacyType [UlawpCandidacyType] (fields name workflow) (searchFields workflow) {
        createEvent -> createUlawpCandidacyType
        viewAction -> readUlawpCandidacyType
        deleteAction -> searchUlawpCandidacyType
    }
    createScreen createUlawpCandidacyType [UlawpCandidacyType] (fields name choicesExist maximumChoices seeAllCandidacyChoices workflow) {
        createEvent -> readUlawpCandidacyType
        cancelEvent -> searchUlawpCandidacyType
    }
    readScreen readUlawpCandidacyType [UlawpCandidacyType] (fields code name choicesExist maximumChoices seeAllCandidacyChoices workflow creationDate lastModificationDate) {
        backEvent -> searchUlawpCandidacyType
        updateEvent -> updateUlawpCandidacyType
        deleteEvent -> searchUlawpCandidacyType
    }
    updateScreen updateUlawpCandidacyType [UlawpCandidacyType] (fields name choicesExist maximumChoices seeAllCandidacyChoices workflow) {
        updateEvent -> readUlawpCandidacyType
        cancelEvent -> readUlawpCandidacyType
    }
}
```

Excerto de código A.30: Candidacy Type

```
flow ulawp.candidacyManagement.candidacyType.entry channel (WebJava) {
    searchScreen searchUlawpCandidacyTypeEntry [UlawpCandidacyTypeEntry]  {
        dialog event addCoursesCandidacy -> addCoursesCandidacyType
        dialog event addDegreesCandidacy -> addDegreesCandidacyType
        viewAction hidden
        deleteAction -> readUlawpCandidacyType@ulawp.candidacyManagement.candidacyType
    }
    searchScreen addCoursesCandidacyType [CompetenceCourse] (fields code) (searchFields code name) {
        selectMultipleEvent addCourse
    }
    searchScreen addDegreesCandidacyType [Degree] (fields code) (searchFields code) {
        selectMultipleEvent addDegree
    }
}
```

Excerto de código A.31: Candidacy Type Entry

```
flow ulawp.candidacyManagement.candidacy.insideWorkflow channel (WebJava) {
    readScreen readUlawpCandidacyInsideWorkflow [UlawpCandidacy] (fields active accepted candidacyType chosenEntries executionYear creationDate lastModificationDate) {
        backEvent hidden
    }
    searchScreen searchChosenProposalCandidacySplit [UlawpProposal] {
    }
    searchScreen searchFinalizeCandidacyChoiceSplit [UlawpCandidacyEntry] {
    }
}
```

Excerto de código A.32: Candidacy Inside Workflow

```
flow ulawp.candidacyManagement.candidacy channel (WebJava)  {
```

```
2     searchScreen searchUlawpCandidacyGlobal [UlawpCandidacy] (fields active accepted
      executionYear candidacyType) (searchFields active accepted executionYear
      candidacyType) {
3         createEvent -> createUlawpCandidacy
4         dialog selectMultipleEvent executeCandidaciesOperation ->
      executeUlawpCandidacyWorkflowOperation
5         dialog selectMultipleEvent deleteCandidacies -> confirmationDeleteCandidacy
6         viewAction //-> ManageCandidacyUI
7         deleteAction
8     }
9     searchScreen searchUlawpCandidacyCandidate [UlawpCandidacy] (fields active accepted
      executionYear candidacyType) (searchFields active accepted executionYear
      candidacyType) {
10        createEvent -> createUlawpCandidacy
11        dialog selectMultipleEvent deleteCandidacies -> confirmationDeleteCandidacy
12        viewAction //-> ManageCandidacyUI
13        deleteAction hidden
14    }
15    createScreen createUlawpCandidacy [UlawpCandidacy] (fields executionYear) {
16        createEvent // -> ManageCandidacyUI
17        cancelEvent
18    }
19    screen executeUlawpCandidacyWorkflowOperation {
20    }
21    screen confirmationDeleteCandidacy {
22    }
23 }
```

Excerto de código A.33: Candidacy

```
1 flow UlawpProposalDetailsComponent channel(WebJava) {
2     screen UlawpProposalDetailsComponentUI {
3     }
4 }
```

Excerto de código A.34: Proposal Component

Figura A.1: Classe: Proposta (Atributos)

Figura A.2: Classe: Proposta (Métodos)

Figura A.3: Classe: Tipo de proposta

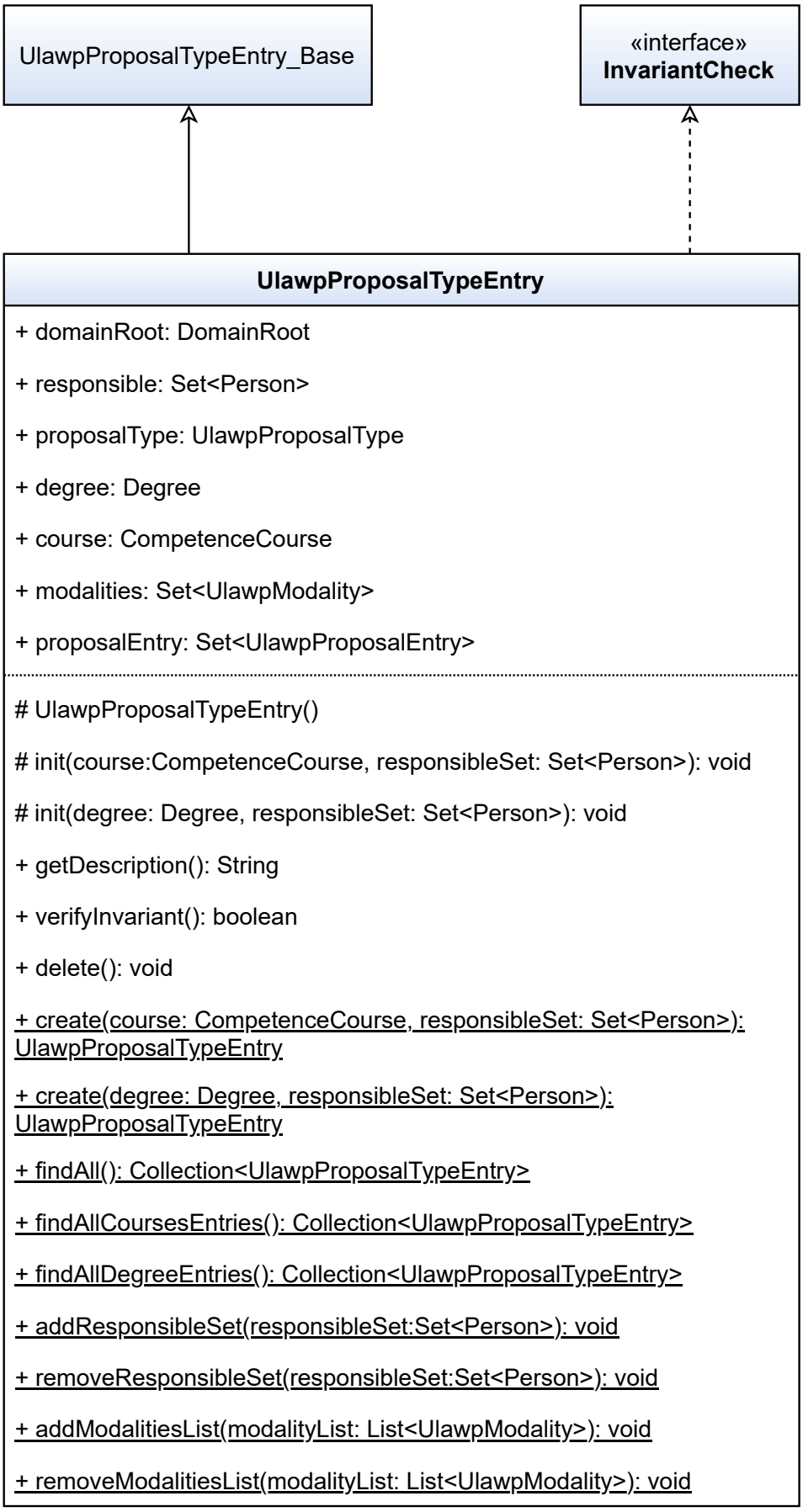

Figura A.4: Classe: Entrada de tipo de proposta

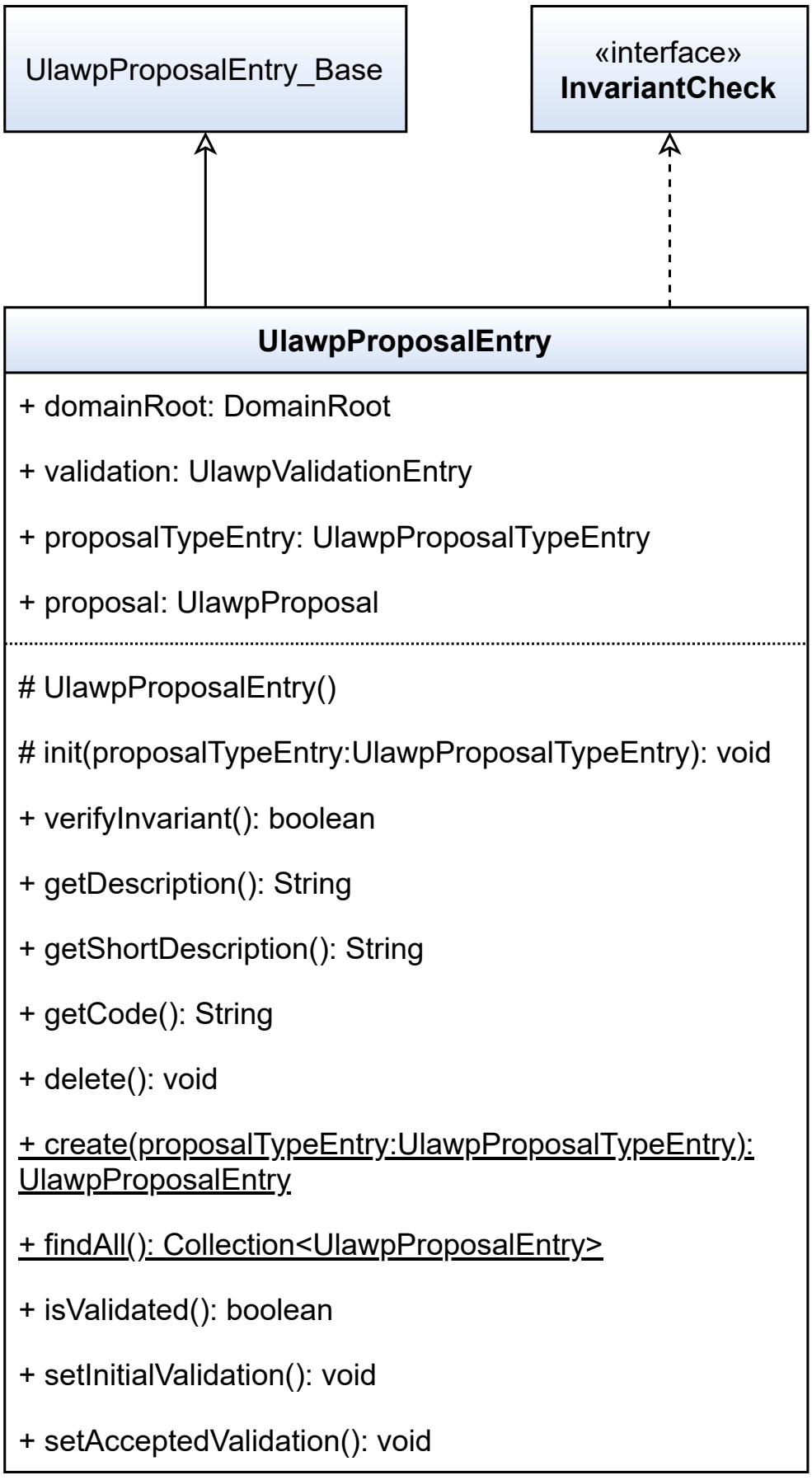

Figura A.5: Classe: Entrada de proposta

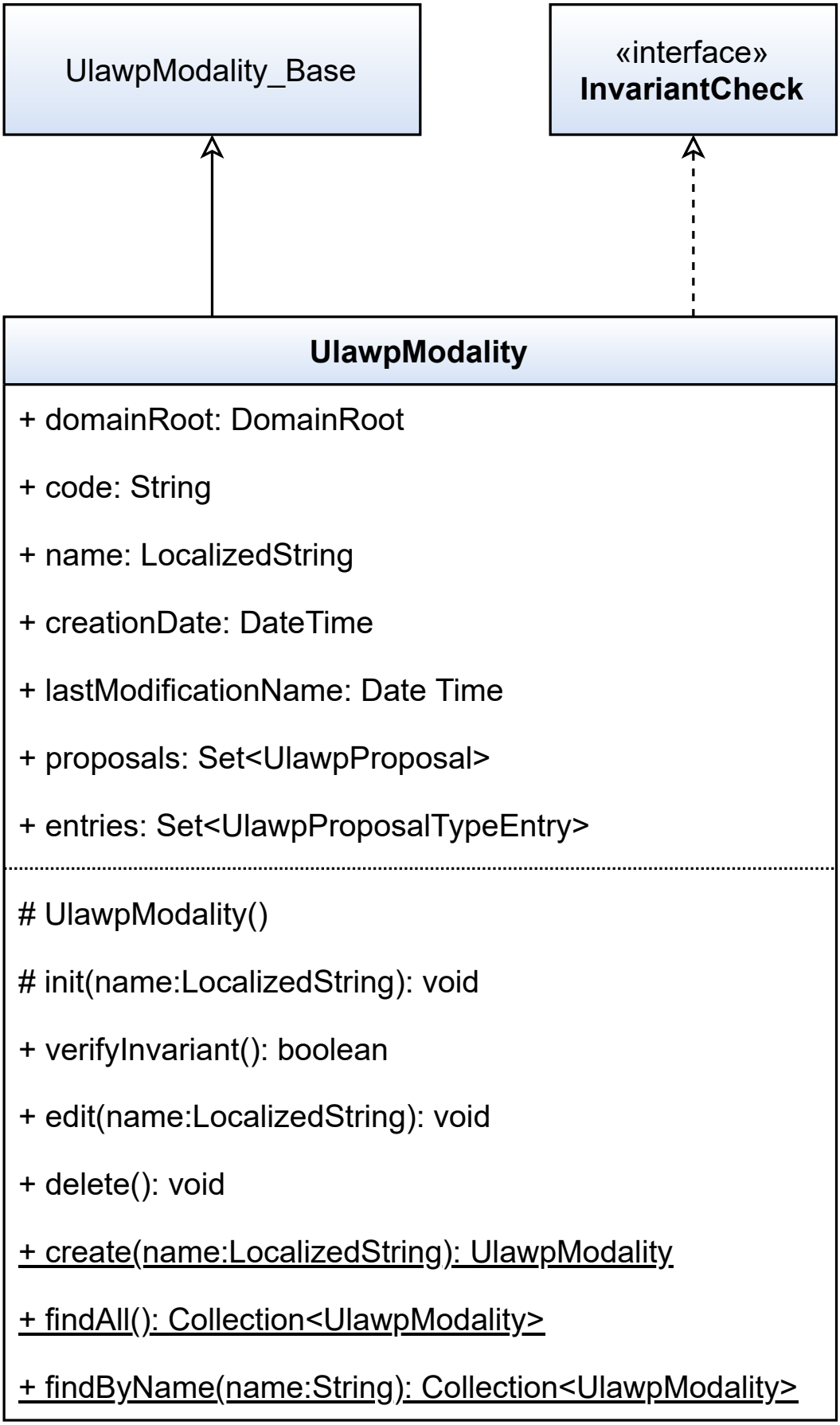

Figura A.6: Classe: Modalidade

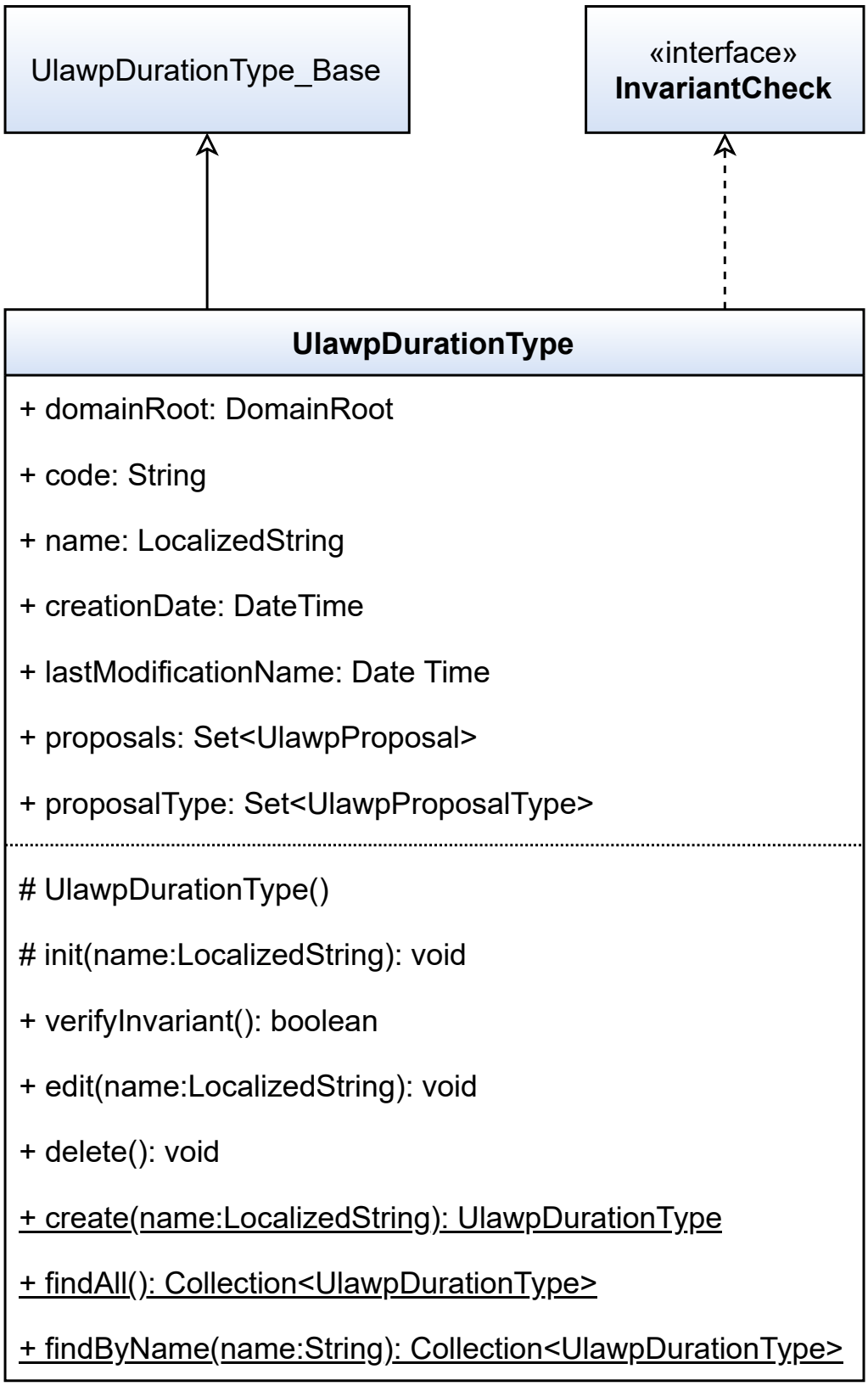

Figura A.7: Classe: Duração

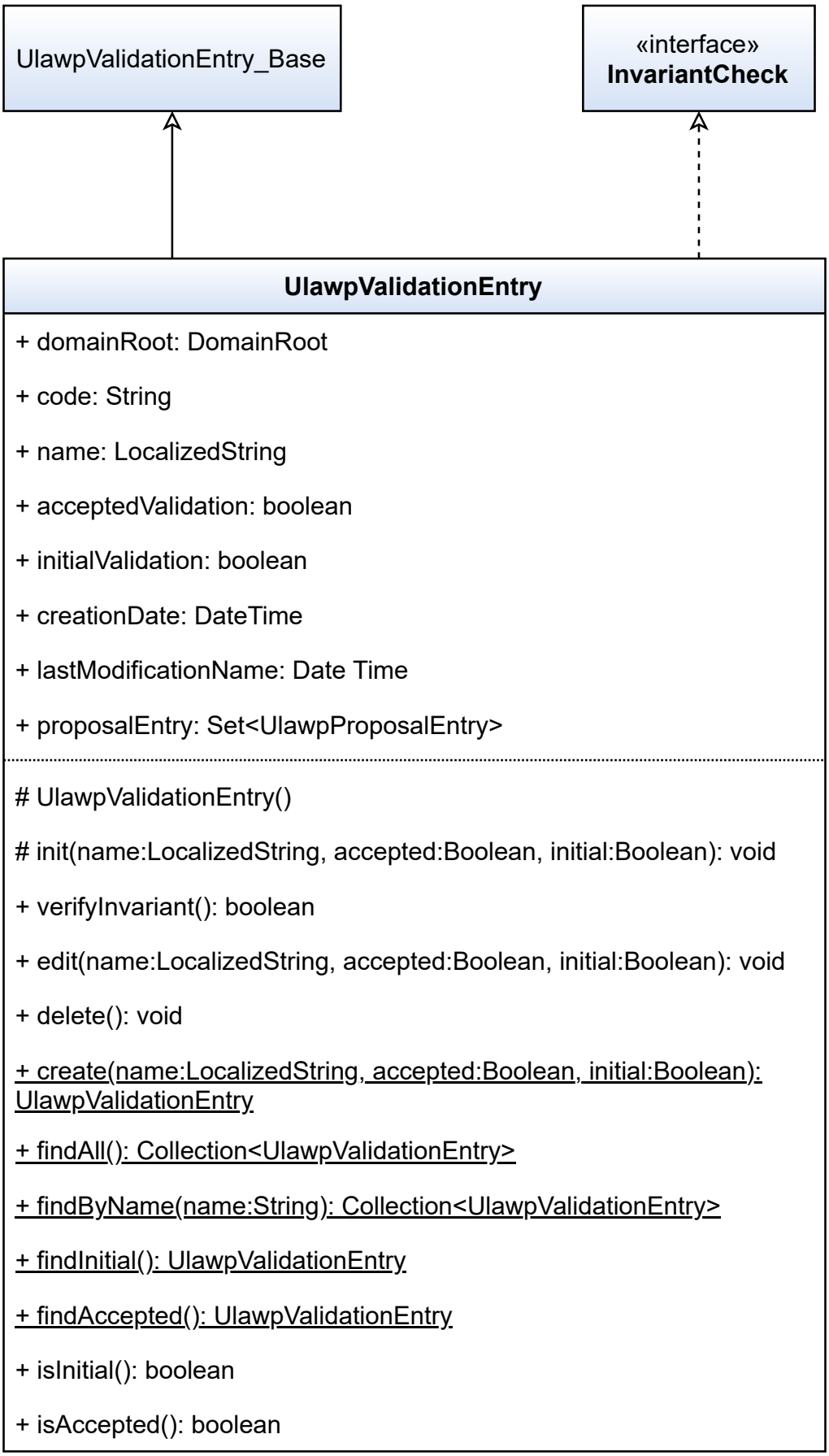

Figura A.8: Classe: Validação

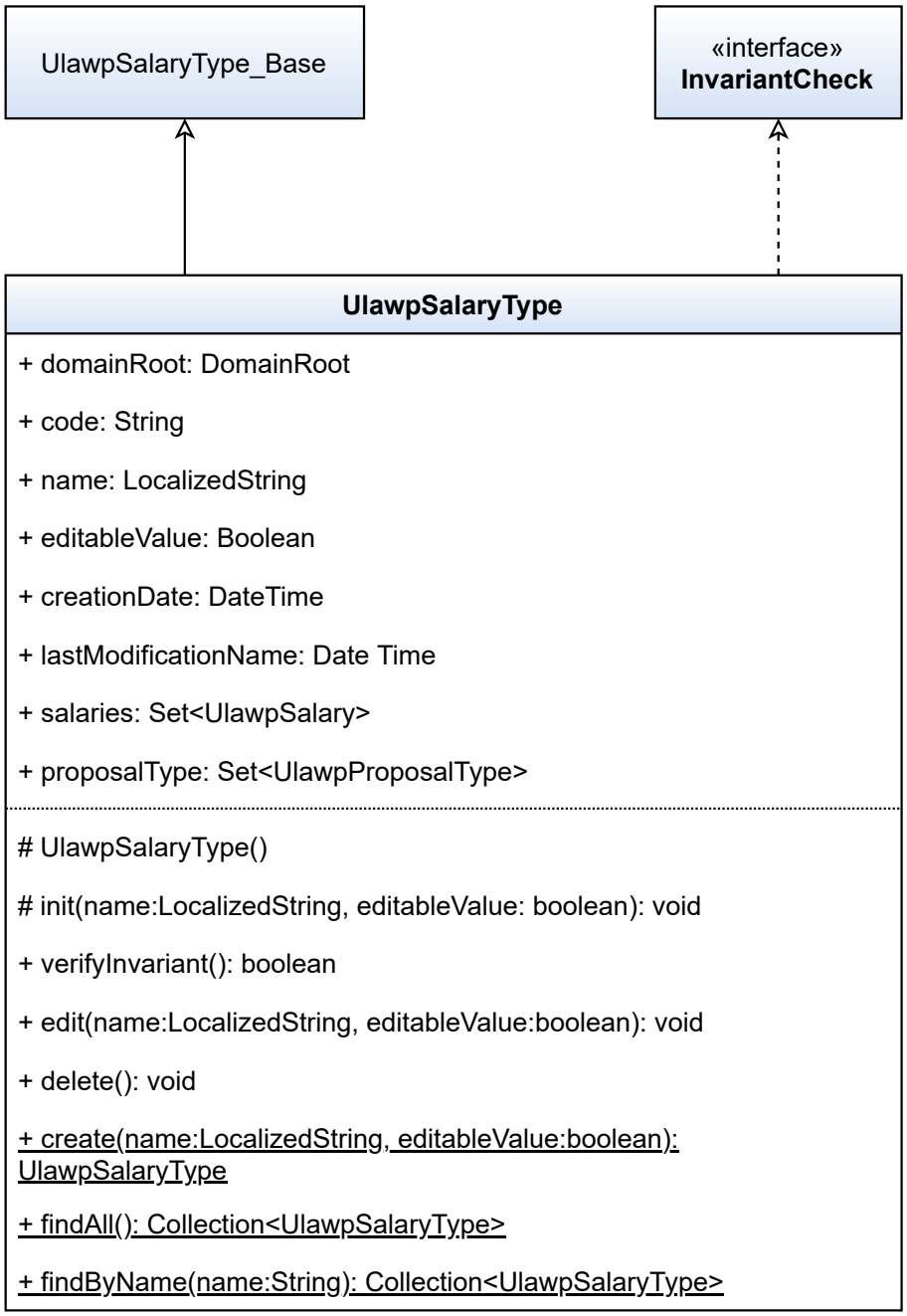

Figura A.9: Classe: Tipo de Remuneração

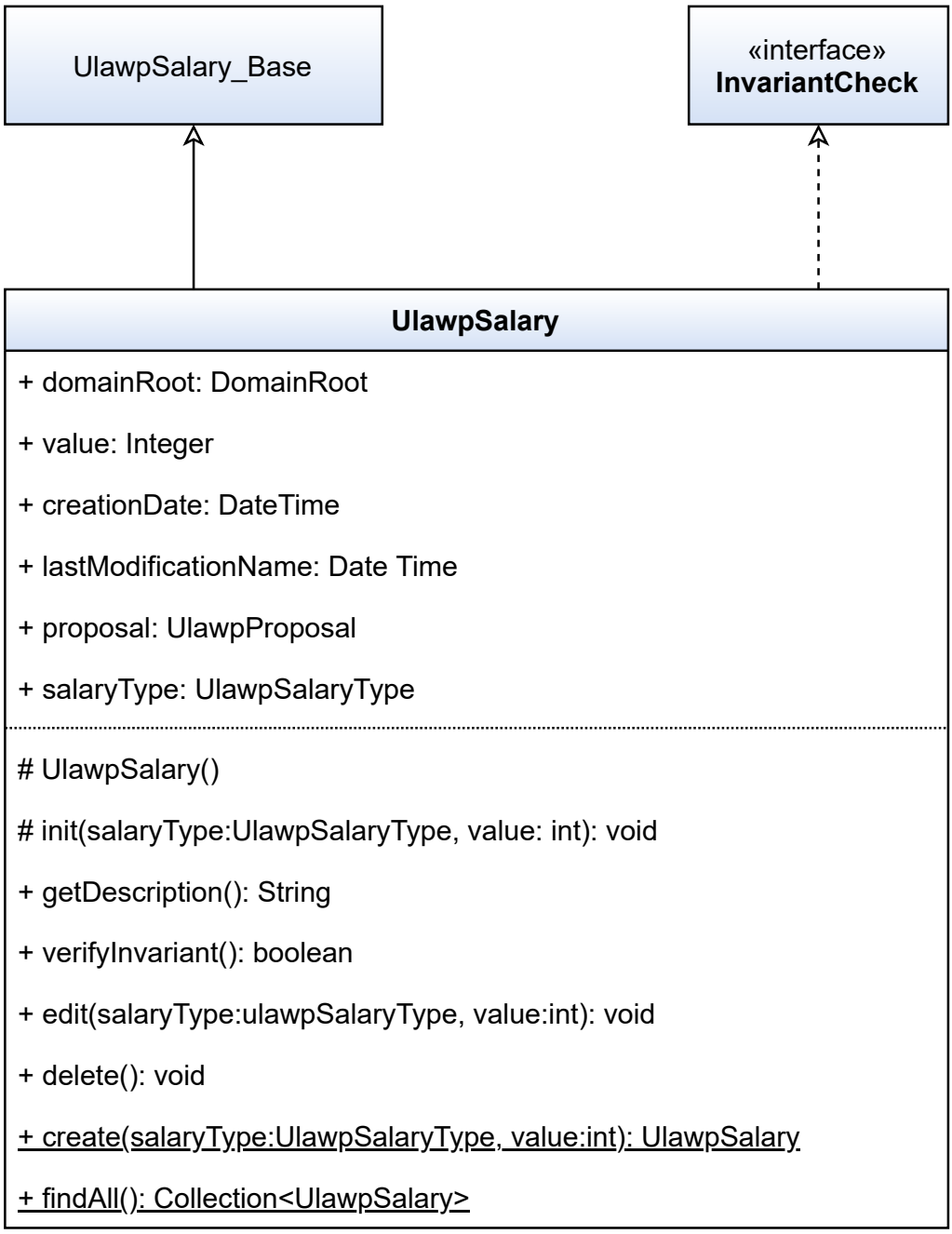

Figura A.10: Classe: Remuneração

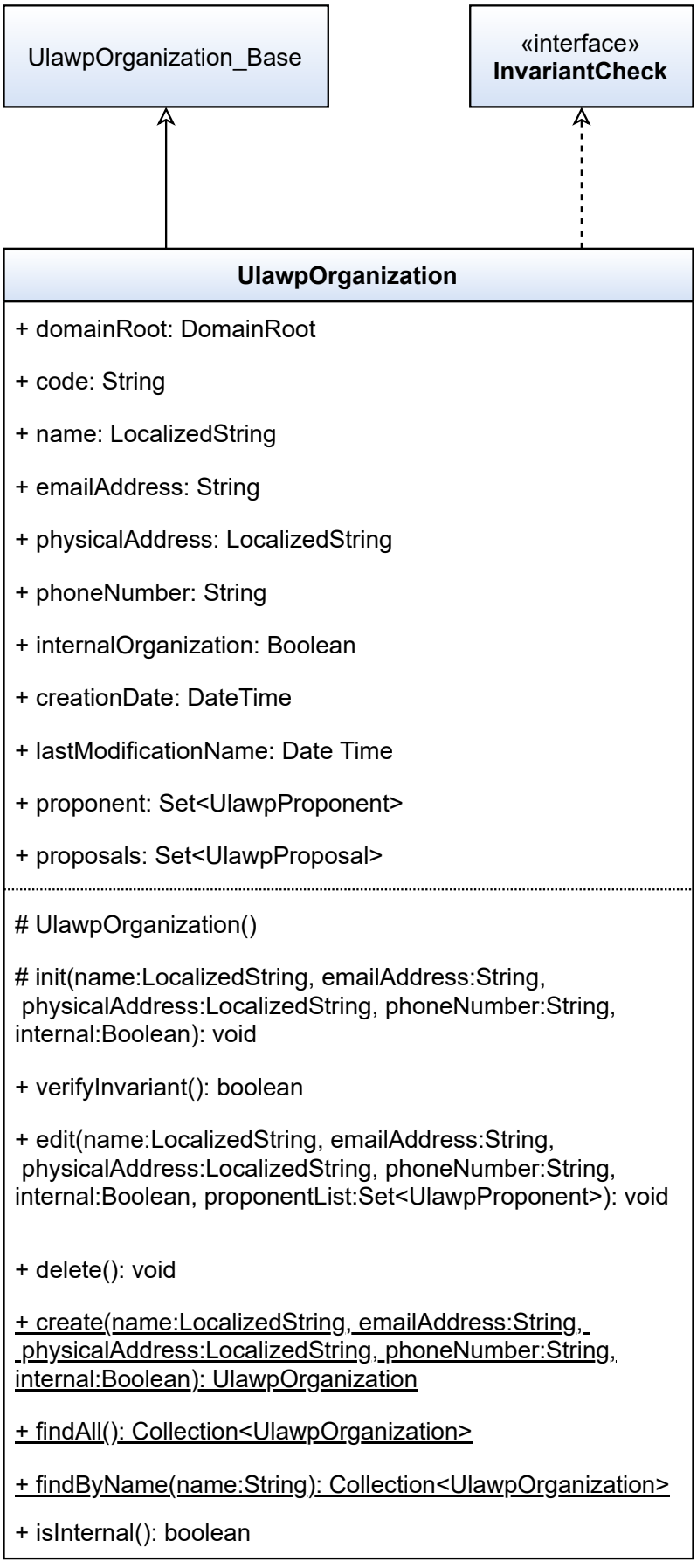

Figura A.11: Classe: Entidade Proponente

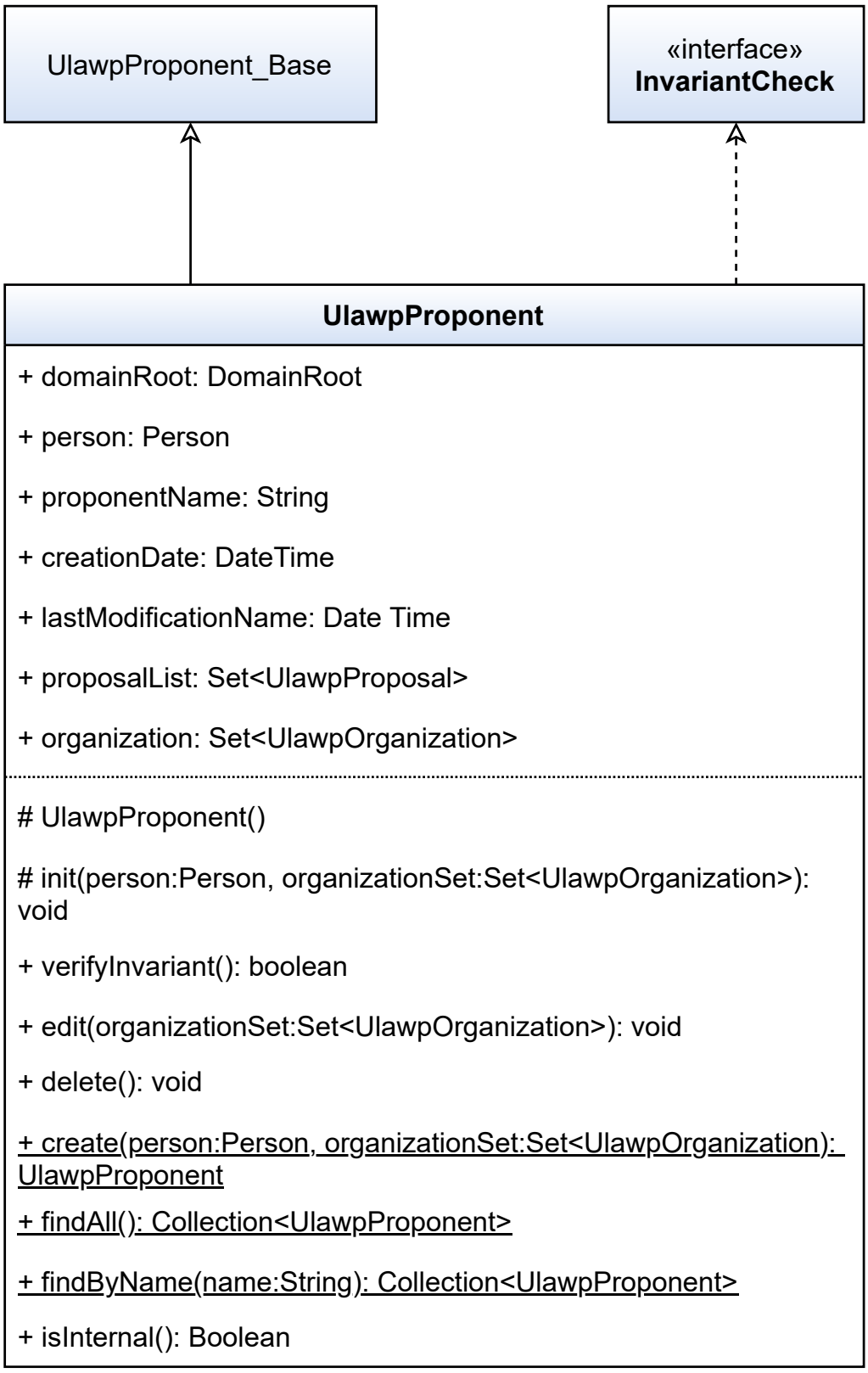

Figura A.12: Classe: Proponente

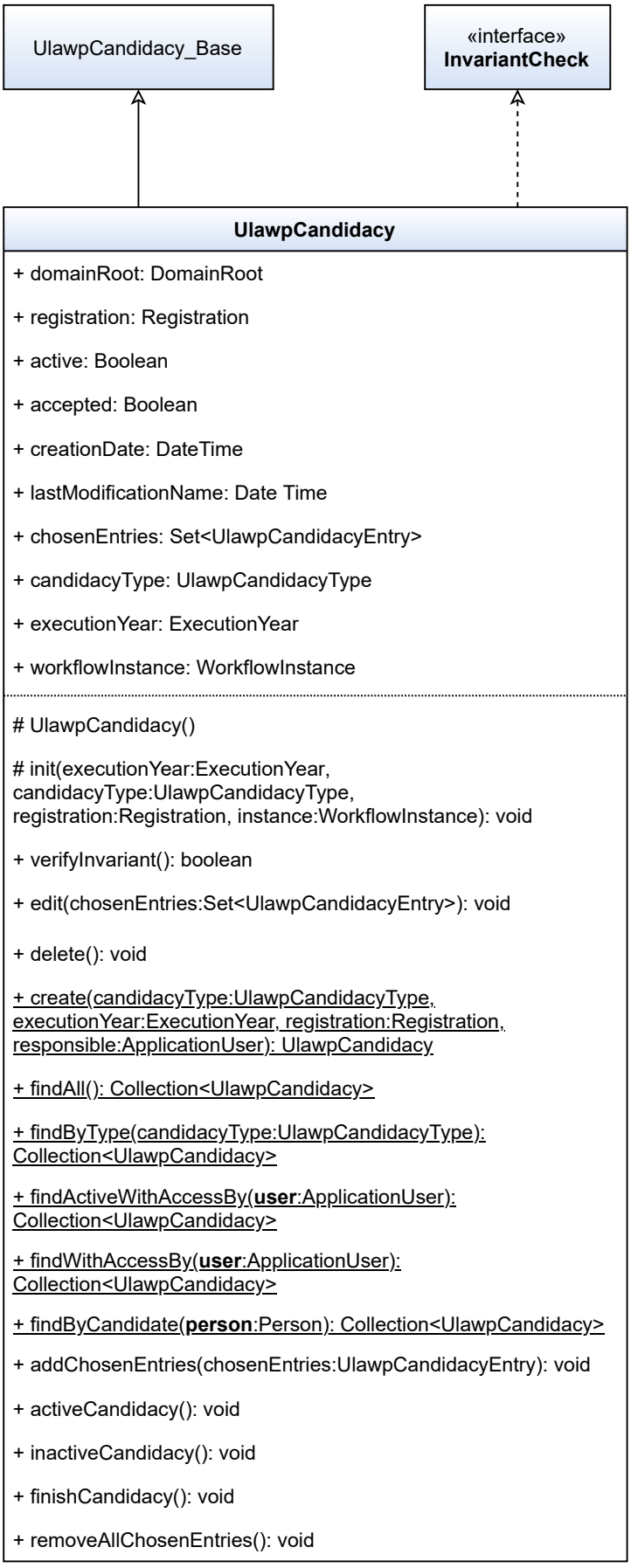

Figura A.13: Classe: Candidatura

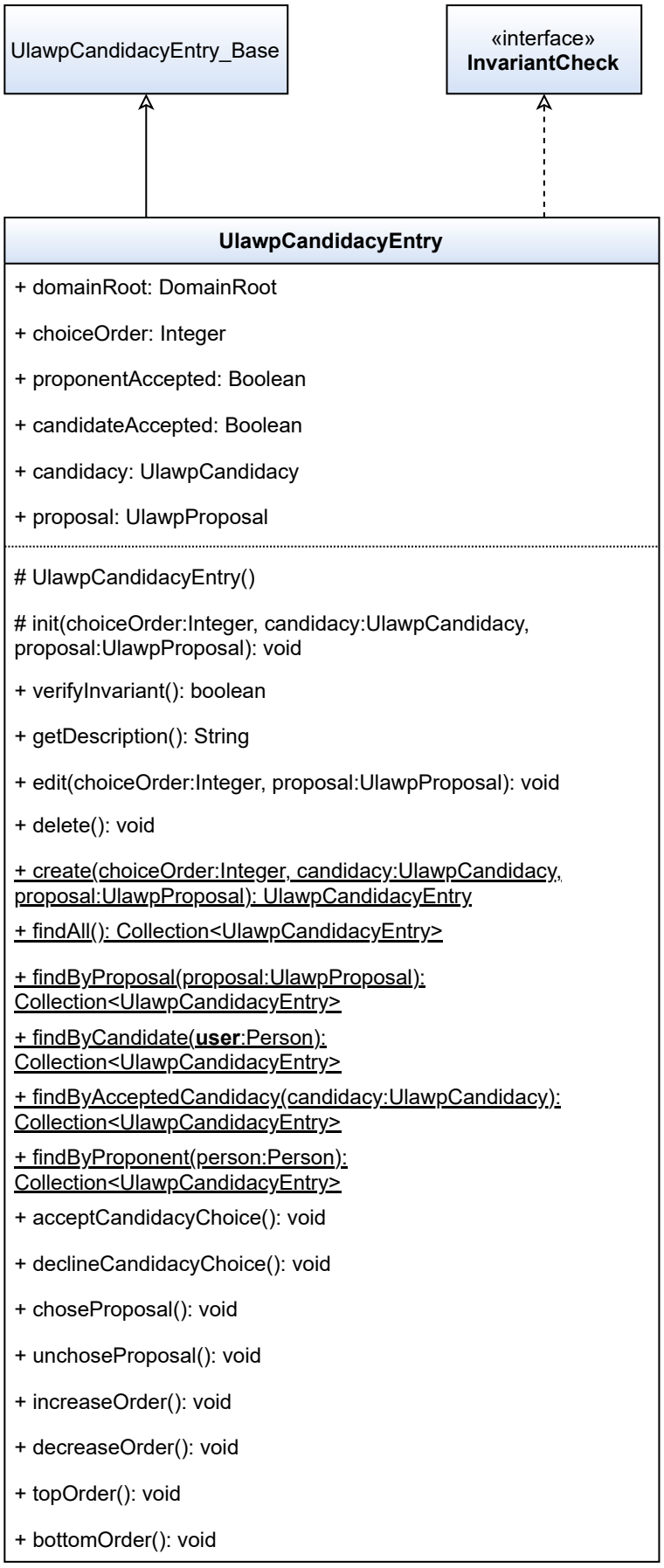

Figura A.14: Classe: Entrada de candidatura

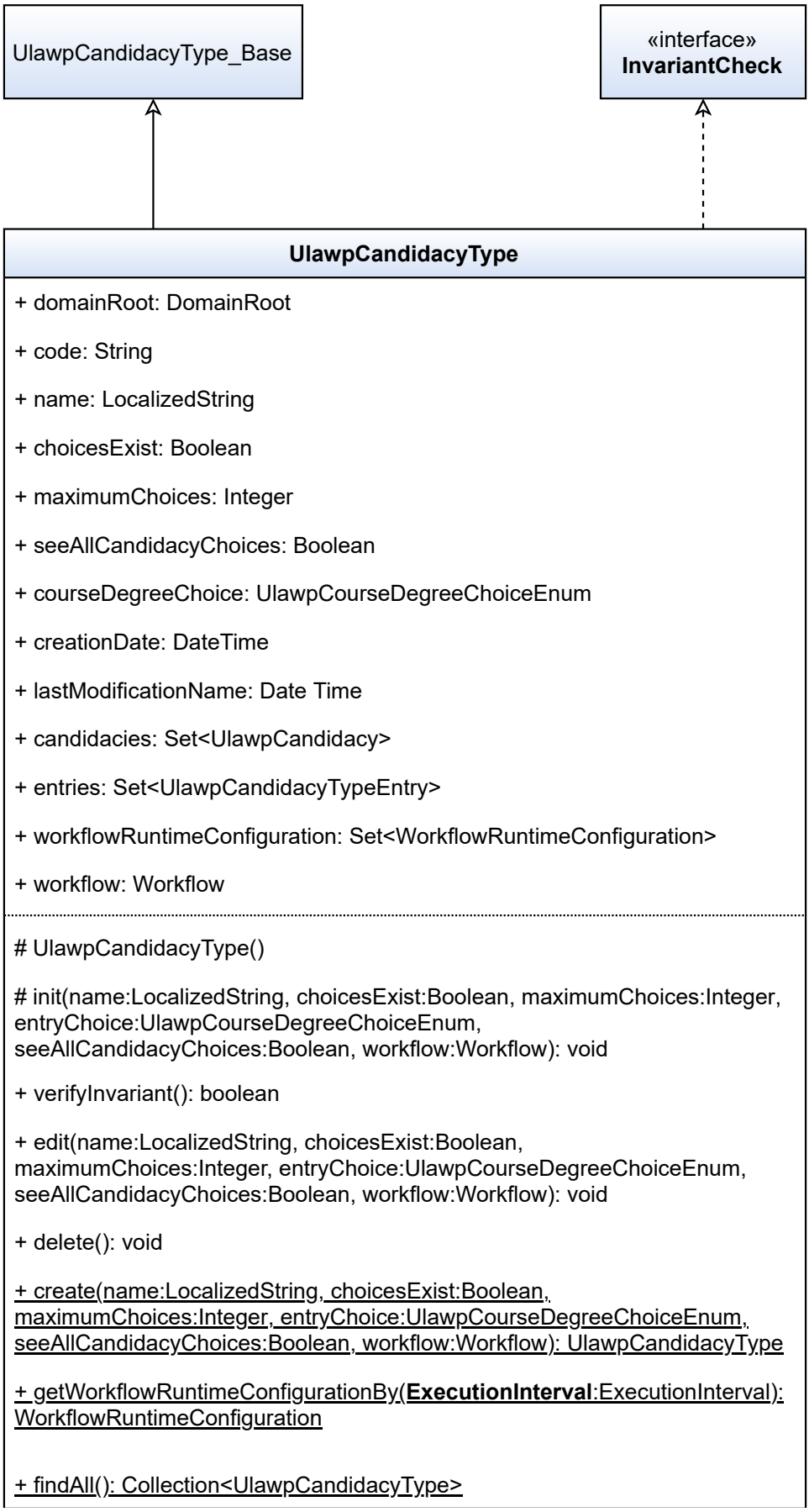

Figura A.15: Classe: Tipo de Candidatura

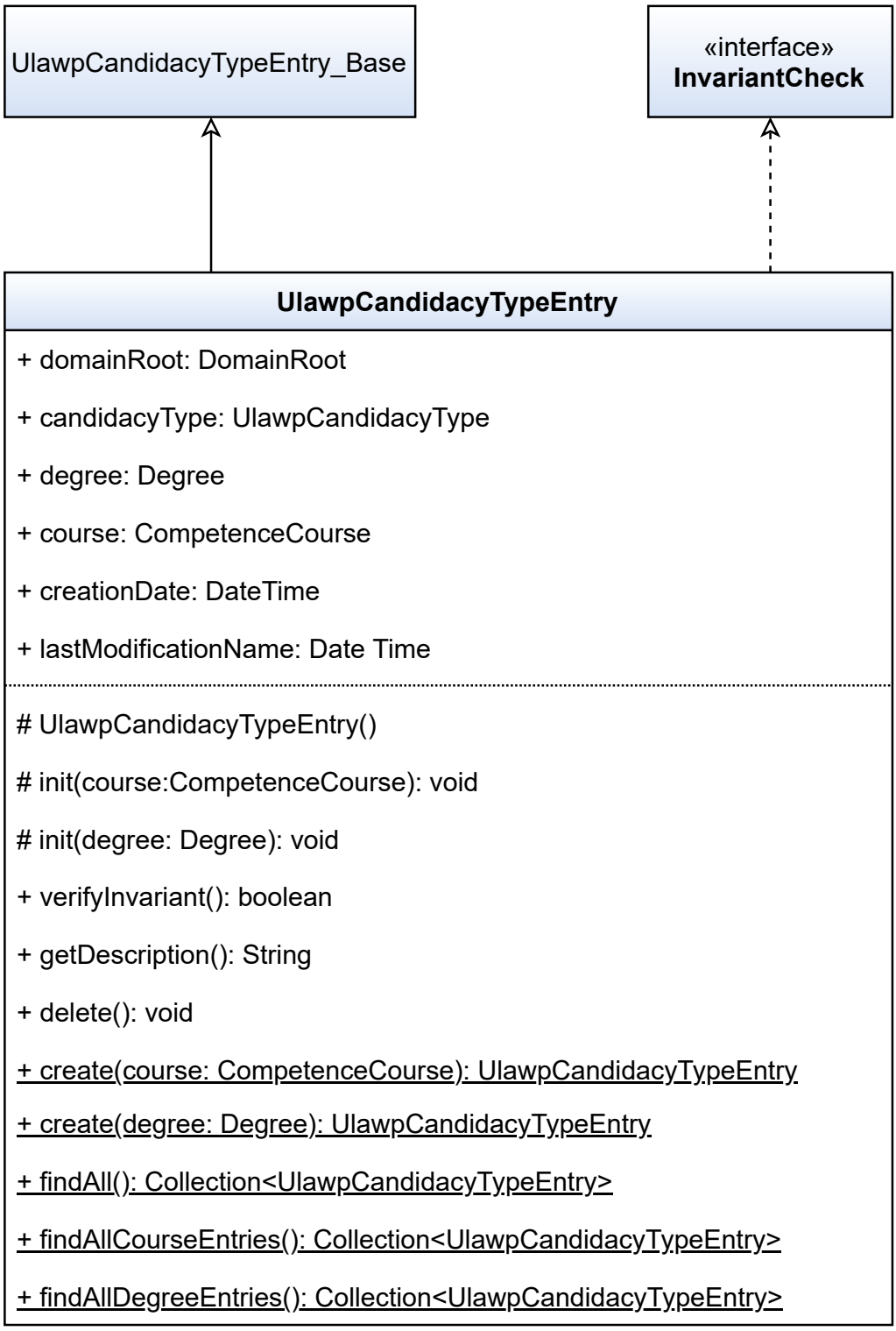

Figura A.16: Classe: Entrada de tipo de candidatura

A.3 Levantamento de requisitos - FCUL

Processo de atribuição de alunos a estágios - modelação no Fénix

Hugo Miranda e coordenadores de ciclo de estudo de Ciências

28 de Novembro de 2019

1 Introdução

O processo tem como objetivo diminuir o esforço dos coordenadores de ciclos de estudo na atribuição de estágios a alunos, agregando numa única plataforma as ofertas de estágio e as manifestações de interesse por parte dos alunos e gerindo o respetivo processo de afetação de candidatos a propostas. O processo é especialmente relevante para o 2º ano dos 2º ciclos de Ciências, mas podem também surgir necessidades de outros ciclos de estudos. Espera-se que o coordenador **intervenha principalmente na fase anual de configuração do processo** mas deverá poder, ao longo de todo o processo **supervisionar o seu andamento** e **gerir as exceções** que naturalmente vão emergindo e **recolher informação estatística** sobre a forma como decorreu o processo em cada ano.

Quando aplicável, a informação recolhida deverá **alimentar o processo de gestão de dissertações**, também ele em definição no Fénix.

A apresentação de requisitos realizada em seguida aborda cada uma das fases do processo.

2 Configuração genérica e definições

2.1 Instâncias

Assume-se que existirão simultaneamente **múltiplas instâncias da plataforma**, de forma a que elas possam evoluir independentemente entre si numa relação de **uma plataforma para um ou mais cursos**. Cada plataforma terá **1 ou mais gestores**, que serão necessariamente os coordenadores de cada curso coberto pela instância da plataforma mas também terceiros, cuja nomeação depende da organização interna do curso e do departamento. Será responsabilidade dos gestores já existentes adicionar outros gestores.

Cada instância da plataforma deve disponibilizar aos gestores indicadores de gestão sobre o andamento do processo: número de propostas e entidades proponentes, número de propostas aceites e rejeitadas, propostas com candidatos, número de candidatos sem proposta atribuída, etc.[1] Além disso, e para acomodar as inúmeras exceções que surgem ao longo deste processo, o gestor deverá ter a possibilidade de realizar ações ignorando os prazos definidos para os participantes.

[1] cf. peipal

Deve-se assumir que a plataforma **não será utilizada por todos os ciclos de estudos nem por todos os alunos num ciclo de estudos que a use**. Nessa medida, será importante que a sua existência ou utilização num ciclo de estudo não bloqueie modelos alternativos no que toca à sua integração com o processo de tratamento de dissertações que está a ser desenvolvido autonomamente. Por outro lado, os ciclos de estudo que a utilizam deverão ter a sua **integração com o processo de tratamento de dissertações** tão simplificado quanto possível.

2.2 Proponentes

Os **proponentes** poderão ser entidades internas ou externas a Ciências pelo que a plataforma terá que suportar um mecanismo de **criação de contas e autenticação de entidades externas**. Para respeitar o RGPD, os proponentes terão que assumir algumas responsabilidades no tratamento dos dados que lhe são confiados pelo que ou o mecanismo de autenticação assegura a não repudiação ou será necessário carregar documentos em formato digital e associá-los aos proponentes. Os proponentes deverão ser registados numa base de dados perene (independente dos anos letivos) por forma a registar pelo menos a seguinte informação histórica:

- Nome da entidade e pessoa de contacto
- Dados para contacto (telefone e e-mail)
- Histórico de propostas, alunos e seus resultados
- Entradas de texto livre que permitam manter um registo interno das interações com a entidade (o que correu bem e mal)

2.3 Suporte à Comunicação

A plataforma deverá permitir que, em qualquer momento, o gestor envie mensagens de correio eletrónico para os seguintes grupos:

- os proponentes que tenham submetido propostas
- os proponentes que estejam registados na base de dados de proponentes
- os proponentes com propostas atribuídas
- os proponentes sem propostas atribuídas
- os potenciais candidatos
- os candidatos que tenham submetido candidaturas
- os candidatos com proposta atribuída
- os candidatos sem proposta atribuída

3 Configuração anual

Para efeitos de simplificação do processo, será importante disponibilizar uma facilidade que permita a cópia prévia dos parâmetros de configuração de um ano para o seguinte.

São parâmetros de configuração anual da proposta:

- Número de propostas máximo a que cada candidato se poderá candidatar (opcional)
- Número de propostas que cada proponente poderá submeter (opcional)
- Se as propostas devem ser ordenadas por ordem de preferência
- Se são permitidas propostas com indicação do candidato (atribuição por pré-acordo entre o candidato e o proponente)
- A forma de indicar a remuneração mensal das propostas, existindo 3 possibilidades:
  - Não consta da proposta
  - Lista de textos fechada, para que o proponente possa selecionar um
  - Campo numérico

3.1 Calendarização

Alguns cursos identificaram a necessidade de flexibilizar o prazo de candidaturas por proposta. Ou seja, em tempo de configuração anual, o gestor deverá poder identificar se aquando da submissão de propostas, o proponente poderá definir uma data limite de candidatura distinta da pré-definida. Mesmo com esta possibilidade, será importante assegurar que todas as propostas apresentam uma data limite de candidatura que interceta o prazo de candidaturas. Nos casos em que a data limite de candidatura seja inferior ao prazo regular de candidaturas, a informação deverá ser entregue aos proponentes logo que é encerrado o prazo de candidatura a essa proposta.

3.2 Informação aos proponentes

O gestor seleciona um conjunto de informação que autoriza que seja disponibilizada ao proponente nos termos seguintes:

1. Ouvidos os gestores e mediante disponibilidade técnica, o Fénix é carregado com uma lista de dados que podem ser extraídos diretamente sobre cada aluno. Exemplos são:
   - Endereço de correio eletrónico
   - Telefone/telemóvel
   - média atual de curso,
   - nº de matrículas
   - classificação em determinadas disciplinas

2. Os gestores decidem na fase de configuração anual que dados dessa lista os proponentes podem solicitar nas candidaturas. É responsabilidade do gestor garantir a defesa da privacidade dos alunos, selecionando apenas o conjunto de dados que possa ser, justificadamente, considerado relevante para a tomada de decisão por parte do proponente.

3. Na fase de receção de propostas, os proponentes irão selecionar da lista aprovada pelo gestor quais os dados que pretendem.

4. Na fase de apresentação de candidaturas os alunos são informados para cada proposta da lista dos dados que irão ser enviados ao proponente, dando consentimento explícito e afirmativo à sua entrega. Se o aluno não der consentimento, não pode candidatar-se a essa proposta.

3.3 Lista de prerequisitos

O gestor define uma lista de pré-requisitos que os proponentes poderão selecionar para que os alunos do curso se possam candidatar. Exemplos são a média atual de curso, número de disciplinas em atraso, etc. O sistema deverá impedir os candidatos de concorrerem aos estágios para os quais não satisfaçam os requisitos.[2]

4 Receção de propostas

No período definido pelo gestor, o Fénix passa a aceitar propostas de estágio. Sem diferenciação no processo, estas propostas poderão ser submetidas por empresas, docentes, unidades de investigação. A informação a submeter para cada proposta inclui pelo menos [3]:

- Título
- Objetivos
- Plano de trabalhos
- Envolvimento do aluno no trabalho
- Incorporação em equipa com objetivo identico (S/N)
- Data de início/fim
- Remuneração, de acordo com os critérios definidos na configuração anual.

Alguns parâmetros poderão depender de outras configurações (ver por exemplo Calendarização).

[2] A atribuição por pré-acordo entre o candidato e o proponente poderá ser considerado um pré-requisito, simplificando desta forma a codificação do sistema.

[3] cf. peipal

5 Validação das propostas

Após a submissão de cada proposta, os gestores são notificados para acederem à plataforma e procederem à sua validação. Deverá ser possível ao proponente corrigir a proposta, caso em que será resubmetida para aprovação do gestor. Para tal, o gestor poderá adicionar comentários que serão remetidos pelo Fènix para o proponente.

O gestor poderá também editar partes da proposta, em particular adicionando prérequisitos.

6 Seleção de propostas pelos alunos

Cada candidato carrega uma única vez o seu CV, que será enviado para os proponentes responsáveis pelas propostas a que ele se candidate.

As propostas são apresentadas aos alunos, selecionando os alunos um número máximo (se aplicável, dependendo da configuração anual) daquelas para as quais o aluno satisfaça os prerequisitos. O aluno indica (se aplicável de acordo com a configuração anual)a ordem de preferência das propostas.

Junto a cada proposta deve ainda constar o conjunto de informação solicitada pelo proponente por forma a que o aluno possa decidir de forma informada se pretende ou não candidatar-se enviando esses dados.

6.1 Proteção de Dados

O processo implica o tratamento e partilha de dados pessoais dos candidatos com terceiros pelo que será necessário prestar todas as informações a que o RGPD obriga e obter o seu consentimento **explícito e afirmativo** para o tratamento.

As informações a prestar serão um texto fixo, a definir em tempo de configuração genérica sem possibilidade de edição por parte dos gestores.

Os dados pessoais que serão partilhados são o CV e a informação solicitada pelo proponente pelo que irão variar de acordo com as propostas a que o candidato concorra. Antes da submissão, o candidato deverá ser informado de forma clara dos dados que serão enviados ao proponente e da necessidade de aceitar para submeter a candidatura.[4]. Perante a informação apresentada, o candidato deverá ter as opções "Aceito"e "Não Aceito".

7 Seleção de candidatos e atribuição

O momento de abertura da fase de seleção de candidatos é determinado pelo gestor. Esta regra irá permitir que o gestor faça previamente a atribuição de alguns (ou todos) os candidatos de acordo com critérios não contemplados pela plataforma. Para estas atribuições o gestor deverá poder consultar as candidaturas e o currículo académico de cada candidato e a lista de propostas. O processo deverá ser facilitado, apresentando numa tabela a informação básica de suporte à decisão (por exemplo a média atual de cada aluno) e a lista de propostas ainda disponíveis.

[4]Os textos serão definidos mais próximo da conclusão do projeto

Existindo ainda candidatos e propostas não atribuídas quando o gestor determina a abertura da fase de seleção, é disponibilizado a cada proponente a lista de alunos que a selecionou, juntamente com o seu CV e a restante informação solicitada. Por forma a garantir a confidencialidade dos dados, esta informação não pode ser enviada por correio eletrónico, tendo que ficar disponível para consulta ou download pelo proponente na plataforma. O processo de seleção é deixado ao cuidado do proponente e dos candidatos.

Independentemente da forma de atribuição, o candidato e o proponente encerram o processo indicando na plataforma respetivamente a proposta e o candidato selecionado. Este passo é importante para garantir a aceitação por ambas as partes.

A plataforma deverá suportar uma repetição de todo o processo a partir da receção de propostas por forma a permitir uma 2ª fase que permita:

- a submissão de novas propostas
- a manutenção de propostas já existentes na 1ª fase
- a inscrição de novos candidatos
- nova atribuição de candidatos a propostas, permitindo resolver casos que ficaram pendentes da 1ª fase e alunos que só posteriormente reuniram as condições para realizar a candidatura.

8 Orientador interno

Cada candidato com uma proposta atribuída terá que ter um orientador interno. Em última instância, é responsabilidade do gestor garantir esta atribuição pelo que este deverá ter a informação necessária para a realizar, nomeadamente os dados de todos os pares (proposta,candidato), a indicação de terem ou não orientador e, para cada docente do departamento, o número de orientações em curso e a possibilidade de fazer atribuições de orientações internas.

Simultaneamente, a plataforma deverá facilitar esta atribuição, disponibilizando aos docentes do departamento a lista de pares (proposta,candidato) a aguardar orientador interno e permitindo que estes se auto-nomeiem orientadores.

A informação relevante das atribuições (título e orientador) será utilizada para instanciar um processo de gestão de dissertações, em desenvolvimento em Ciências.

9 Acordo específico

Cada par (proposta,candidato) requer a existência de um acordo formal assinado pelo orientador interno, o supervisor externo, o aluno e o coordenador do curso. A plataforma deverá simplificar a gestão deste processo, sinalizando ao gestor que acordos já foram recebidos e elaborando a partir de uma template e dos dados da proposta e do candidato uma versão muito próxima da final do acordo para assinatura.

10 Entrega de relatório preliminar

A plataforma deverá disponibilizar um suporte para que os alunos possam submeter o relatório preliminar dos trabalhos. Este relatório deverá ficar disponível para o aluno, o orientador, o proponente e os gestores. Deverá ser possivel ao gestor listar todos os alunos, indicando quais submeteram o relatório e comunicar com aqueles que não submeteram.

11 Disponibilidade e Eliminação de dados

Ao longo de todo o processo, em particular durante o decorrer do estágio, toda a informação gerada durante o processo deverá ficar disponível para o orientador, proponente e gestores.

Por forma a satisfazer os requisitos do RGPD, a informação que não é gerada automaticamente, em particular os CVs dos candidatos, deverá ser eliminada do sistema logo que deixe de ser considerada necessária.[5]

[5] O prazo será definido mais próximo da conclusão do projeto

Abreviaturas

CA Comissão de Acompanhamento.

CC Conselho Científico.

CRUD *Create, Read, Update, Delete*.

DEP Dissertação / Estágio / Projeto.

DI-FCUL Departamento de Informática da Faculdade de Ciências da Universidade de Lisboa.

DI-UL Departamento de Informática da Reitoria da Universidade de Lisboa.

DML *Domain Model Language*.

DPEI Disciplina de Projeto de Engenharia Informática.

DSL *Domain Specific Language*.

ECTS *European Credit Transfer and Accumulation System*.

FCT Fundação para a Ciência e a Tecnologia.

FCUL Faculdade de Ciências da Universidade de Lisboa.

FDUL Faculdade de Direito da Universidade de Lisboa.

FFUL Faculdade de Farmácia da Universidade de Lisboa.

IGOT-UL Instituto de Geografia e Ordenamento do Território da Universidade de Lisboa.

IST Instituto Superior Técnico.

MICF Mestrado Integrado em Ciências Farmacêuticas.

PSL *Presentation Specific Language*.

SIGA Sistema Integrado de Gestão Académica.

UC Unidade curricular.

ULisboa Universidade de Lisboa.

Bibliografia

[1] Hot swapping. https://en.wikipedia.org/wiki/Hot_swapping. (Acedido em 20/09/2020).

[2] Internacionalização. https://en.wikipedia.org/wiki/Internationalization_and_localization. (Acedido em 20/09/2020).

[3] Modelo entidade associação. https://java-programming.mooc.fi/part-11/1-class-diagrams. (Acedido em 23/04/2020).

[4] Organization structures. https://martinfowler.com/apsupp/accountability.pdf. (Acedido em 23/04/2020).

[5] Unified modeling language. https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/. (Acedido em 23/04/2020).

[6] Virtual machine. https://en.wikipedia.org/wiki/Virtual_machine. (Acedido em 25/11/2021).

[7] Atlassian. Bitbucket. https://bitbucket.org/product. (Acedido em 14/01/2020).

[8] Aaron Bangor, Philip Kortum, and James Miller. An empirical evaluation of the system usability scale. *Journal of Human–Computer Interaction*, 24(6):574–594, 2008.

[9] Aaron Bangor, Philip Kortum, and James Miller. Determining what individual sus scores mean: Adding an adjective rating scale. *Journal of usability studies*, 4(3):114–123, 2009.

[10] John Brooke. Sus: A quick and dirty usability scale. *Journal of Human–Computer Interaction*, 1995.

[11] João Cachopo. *Development of rich domain models with atomic actions.* PhD thesis, Instituto Superior Técnico, 2007.

[12] Canonical. Xubuntu. https://xubuntu.org/release/18-04/. (Acedido em 14/01/2020).

[13] Oracle Corporation. Java 11. https://www.oracle.com/pt/java/technologies/javase/jdk11-archive-downloads.html. (Acedido em 14/01/2020).

[14] Oracle Corporation. Mysql. https://www.mysql.com/. (Acedido em 14/01/2020).

[15] Uiversidade de Lisboa. Sobre nós ulisboa. https://www.ulisboa.pt/sobre-nos. (Acedido em 25/11/2021).

[16] Eclipse Foundation. Eclipse. https://www.eclipse.org/eclipseide/2019-12/. (Acedido em 14/01/2020).

[17] The Apache Software Foundation. Maven project. https://maven.apache.org/docs/history.html. (Acedido em 14/01/2020).

[18] The Apache Software Foundation. Pom. https://maven.apache.org/guides/introduction/introduction-to-the-pom.html. (Acedido em 14/01/2020).

[19] The Apache Software Foundation. Tomcat. https://tomcat.apache.org/download-90.cgi. (Acedido em 14/01/2020).

[20] Diogo Godinho. Cursos partilhados no sistema fenix - análise comparativa de alternativas de concretização. Master's thesis, Faculdade de Ciências da Universidade de Lisboa, 2020.

[21] Quorum Born IT. Qubit. https://qub-it.com/. (Acedido em 25/11/2021).

[22] Glenn E. Krasner and Stephen T. Pope. A description of the model-view-controller user interface paradigm in the smalltalk-80 system. 1998.

[23] Diana Marques. Gestão de alunos erasmus no sistema fenix. Master's thesis, Faculdade de Ciências da Universidade de Lisboa, 2021.

[24] Fundação para a ciência e a tecnologia. Fenix framework. https://fenix-framework.github.io/. (Acedido em 25/11/2021).

[25] Daniel Pires. Controlo de acesso no sistema académico fenix. Master's thesis, Faculdade de Ciências da Universidade de Lisboa, 2019.

[26] António Rito Silva, Artur Ventura, Carlos Ribeiro, Fernando Mira da Silva, João Cachopo, Luís Cruz, and Susana Fernandes. *The FenixEdu Project: an Open-Source Academic Information Platform*. Instituto Superior Técnico, 2011.

[27] Linus Torvalds. Git. `https://git-scm.com/`. (Acedido em 14/01/2020).

[28] Instituto Superior Técnico. Fenixedu. `https://confluence.fenixedu.org/display/FENIXEDU/Welcome`. (Acedido em 25/11/2021).