UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA

GESTÃO DE ALUNOS ERASMUS NO SISTEMA FENIX

Diana Paiva Marques

Mestrado em Informática

Trabalho de Projeto orientado por:
Prof. Doutora Maria Dulce Pedroso Domingos
Prof. Doutor Carlos Nuno da Cruz Ribeiro

2021

Agradecimentos

Primeiramente gostaria de agradecer aos meus orientadores Professora Dulce Domingos e Professor Carlos Ribeiro pelo apoio e paciência ao longo deste projeto.

À Reitoria da Universidade de Lisboa por me ter proporcionado a oportunidade de desenvolver aqui a minha tese de mestrado.

Ao Departamento de Informática da Reitoria pela ajuda e disponibilidade, especialmente aos colegas do Núcleo de Desenvolvimento de *Software* e do Núcleo de Gestão de Sistemas Académicos, nomeadamente à Ana Rute, Daniela Mendes, Daniel Vitoriano, Fábio Ferreira, Nuno Heitor, Nuno Jesus, Pedro Moita, Ricardo Rito, Andreia Rainha, André Faria, Tânia Castelo, Tânia Crespo e um obrigada ainda mais especial à minha mentora Catarina Silva, à Teresa Gouveia e ao José Lima por todo o apoio que me deram ao longo deste projeto.

Ao Núcleo de Mobilidade do Departamento de Relações Externas e Internacionais da Reitoria, nomeadamente à Maria João Antunes, Carlos Pereira, Rute Pimenta, Sandra Brito e Tânia Aboim, pela ajuda e tempo disponibilizado para este projeto.

Aos meus colegas bolseiros, que entraram neste aventura com o mesmo objetivo que eu e que muito me ajudaram durante todo o tempo que estivemos juntos, nomeadamente à Ana Espinheira, Diogo Godinho, Francisco Caeiro, Mafalda Luz e Pedro Ladino. Sem vocês isto não tinha sido possível, ou pelo menos não tinha sido tão divertido!

Não poderia deixar de agradecer à empresa Quorum Born IT, nomeadamente ao Nadir Amin por toda a ajuda, apoio, paciência e tempo disponibilizado para tirar todas as minhas dúvidas por mais parvas que parecessem.

À Faculdade de Ciências por me ter proporcionado tantos momentos espetaculares e pessoas incríveis!

Aos meus amigos que sempre me apoiaram, ouviram e aturaram com muito amor e paciência, especialmente à Ana Fraga, Filipa Miranda, João Polido e Marta Pinto Coelho.

Por último, mas não menos importante, quero agradecer à minha família, especialmente aos meus pais, Ana Paiva e Luís Marques, pela paciência, apoio e suporte ao longo de todo o meu percurso académico, e ao Joey por todo o carinho e sanidade mental que me dá todos os dias.

Espero não me ter esquecido de alguém. Todos vocês foram fundamentais para concluir este trabalho, por isso, a todos um grande obrigada!

"Welcome to the real world! It sucks. You're gonna love it."
Monica Geller, *Friends*

Resumo

O sistema Fenix é um projeto que começou em 2002 no Instituto Superior Técnico com o objetivo de criar um sistema de informação académica integrado. Neste momento, o sistema Fenix já está implementado na reitoria e em dezassete escolas da Universidade de Lisboa e conta com múltiplas funcionalidades com o intuito de agilizar e facilitar a gestão académica para professores, serviços académicos e alunos.

Apesar de já haver várias funcionalidades no sistema Fenix, não existe uma que contemple todo o processo da mobilidade. A parte do processo que passa pelos serviços centrais é feita manual e presencialmente, recorrendo a papel de forma significativa e com a comunicação entre a escola e a reitoria a ser feita via correio eletrónico ou pastas partilhadas, o que resulta em perdas de eficácia e eficiência.

O objetivo deste projeto é automatizar os processos de mobilidade *outgoing* que passam pela Reitoria da Universidade de Lisboa (RUL), isto é, a celebração dos contratos, a gestão das bolsas e o processamento da chegada dos alunos. Para tal, foi necessário criar uma nova funcionalidade no Fenix da RUL, a qual permite que o processo de emissão de contratos seja feito automaticamente, que as assinaturas de documentos sejam feitas digitalmente e que o processamento de chegada seja feito de forma mais automatizada, de modo a diminuir a taxa de erros e a facilitar o procedimento, além de diminuir o consumo de papel.

Até à data de entrega deste relatório foram inicializados 1957 processos de mobilidade *outgoing* e assinados 1255 contratos de mobilidade utilizando o módulo implementado.

Palavras-chave: Sistema Fenix, Mobilidade *European Region Action Scheme for the Mobility of University Students* (ERASMUS), Desmaterialização, Engenharia de processos

Abstract

The Fenix System is a project that started in 2002 at Instituto Superior Técnico. It's main goal was to develop an integrated academic information system. At this point, the Fenix system is already implemented in the central services and seventeen schools of the University of Lisbon. This system already has multiple features with the purpose of facilitate and streamline the academic management for teachers, academic services and students.

Even though there are already several features in the Fenix system, there is none specifically for all the mobility process. So the process that passes through the central services is manual and in person with great use of paper and with the communication between the schools and central services being made by e-mail or dropbox, resulting in loss of effectiveness and efficiency.

The aim of this project is to bring the outgoing mobility processes, that pass through the central services of the University of Lisbon, to the digital era. That is, the management of the mobility schollarships, the execution of the contracts and the process of arrival. This required the development of a new mobility module in the central services Fenix. With this new module, the execution of contracts and the managment of the mobility schollarships is made automatically, the signatures can be digital and the process of arrival is more automated, hence we decrease the error rate, facilitate the process and decrease paper consumption during the mobility process.

Until the delivery date of this report 1957 outgoing mobility processes were initialized and 1255 mobility contracts were signed using the implemented module.

Keywords: Fenix, Mobility, ERASMUS, Dematerialization, Process engineering

Conteúdo

Índice de figuras

Índice de listagens

Índice de tabelas

Capítulo 1

Introdução

Este documento descreve o projeto desenvolvido para automatizar o processo de gestão de alunos ERASMUS que é realizado nos serviços centrais da RUL,incluindo as tarefas para a sua entrada em produção na instância Fenix da RUL. O projeto foi desenvolvido no âmbito da disciplina Dissertação/Projeto do Mestrado em Informática da Faculdade de Ciências da Universidade de Lisboa (ULisboa).

O primeiro capítulo deste documento descreve a motivação, o trabalho desenvolvido e as contribuições deste projeto, a instituição onde o foi realizado e a estrutura deste documento.

1.1 Motivação

Atualmente a gestão dos processos de mobilidade efetuada nos serviços centrais da ULisboa é realizada manual e presencialmente, recorrendo a tabelas excel, utilizando muito papel e recursos humanos e com a comunicação entre entidades a ser feita via e-mail ou *dropbox*.

Os problemas relativos ao processo da mobilidade são vários, sendo o mais grave o atraso nos pagamentos das bolsas aos alunos deslocados em países estrangeiros, chegando mesmo a provocar a sua desistência, por não terem possibilidades financeiras para irem sem a bolsa paga previamente. Uma das principais fontes de atraso neste processo é a falta de coerência na comunicação entre as escolas e a RUL, que provém da complexidade da gestão dos diferentes programas de mobilidade e dos meios utilizados.

Para além do acima referido, também existe o problema do processo de assinaturas de contratos de bolsas entre o estudante e a ULisboa, que requer uma gestão complexa, visto que, nessa altura, são chamados cerca de cem alunos por dia para assinarem os contratos presencialmente na RUL ou, nos casos em que já não estão no país porque o ano curricular já começou no país de destino, têm de constituir procuradores para irem assinar por eles, o que gera um grande descontentamento.

A questão do Regulamento Geral sobre a Proteção de Dados (RGPD) é também um

problema, uma vez que a informação pessoal dos alunos, pais e procuradores se encontra distribuída por vários computadores pessoais.

Adicionalmente, a quantidade de papel que é consumido durante este processo é também um grande problema, entre contratos, planos de estudo, *Transcript of Records* (TR), informações de cabimento e compromisso, declarações de estada, etc., cerca de mil e quinhentas a mil e seiscentas resmas de papel são consumidas por ano apenas nos processos de mobilidade.

1.2 Objetivos

Devido à dimensão do projeto de mobilidade, os objetivos deste trabalho tiveram de ser bem definidos à partida, uma vez que não iria ser possível concluir todo o projeto no tempo estipulado para a escrita deste documento.

Assim, o objetivo deste trabalho é automatizar o processo ERASMUS que passa pelos serviços centrais da ULisboa. Este processo inclui a criação e assinaturas dos contratos de mobilidade, os cálculos dos dias de mobilidade e do valor da bolsa a que o aluno tem direito e o processamento da chegada aquando do regresso do aluno ao país de origem.

1.3 Trabalho desenvolvido

Inicialmente foi realizado o levantamento e a análise de requisitos que permitiu identificar as cinco funcionalidades principais do processo:

- **Criação de contratos de mobilidade**, que engloba a receção dos dados de mobilidade enviados pelas escolas assim como a atribuição de números de compromisso;
- **Assinaturas do contrato de mobilidade**, que compreende as assinaturas do aluno e do Vice-Reitor;
- **Pagamento da primeira tranche**, que inclui as autorizações, pareceres e assinaturas necessárias ao pagamento;
- **Processamento de chegada**, que engloba a verificação do aproveitamento segundo o TR e do cumprimento do número de dias do contrato;
- **Pagamento da segunda tranche**, que inclui as autorizações, pareceres e assinaturas necessárias ao pagamento da segunda tranche, assim como o encerramento do processo de mobilidade.

Desta forma, foi possível criar uma solução e implementar o novo módulo de mobilidade com base nas especificações identificadas.

O desenvolvimento deste módulo seguiu uma metodologia ágil, de modo a permitir a entrada em produção das funcionalidades em iterações distintas e de acordo com os tempos em que foram necessárias. Em cada iteração a funcionalidade foi implementada, testada e colocada em produção.

1.4 Contribuições

Com o desenvolvimento do módulo de mobilidade este processo foi significativamente automatizado. O processo é, agora, concentrado num único sistema, onde os contratos são gerados automaticamente, podendo ser assinados digitalmente a partir do sistema, e o processamento de chegada é feito de forma automática, verificando o aproveitamento do aluno e calculando os dias de mobilidade e os valores de bolsa.

Com este módulo foi possível diminuir a existência de erros humanos, o que torna o processo mais coerente e mais rápido. O problema das assinaturas serem presenciais como referido na secção 1.1 deixa de existir, uma vez que os alunos podem assinar o contrato de mobilidade com chave móvel digital no sistema sem que tenham de se deslocar à RUL.

O processo de mobilidade passou a ser, também, mais ecológico: não só as assinaturas passaram a ser digitais como também deixou de existir a necessidade de imprimir os documentos referentes aos processos de mobilidade, uma vez que os documentos passaram a ficar guardados em sistema.

1.5 Enquadramento institucional

Este projeto foi desenvolvido para a ULisboa, a maior Universidade de Portugal e uma das maiores da Europa. Esta universidade integra 18 escolas, entre faculdades e institutos, comportando, no total, 406 cursos conferentes de grau no ano letivo 2019/2020 e mais de 50 mil alunos inscritos no ano letivo 2020/2021 [18]. A ULisboa integra ainda unidades especializadas e de investigação, estruturas e serviços que atuam de forma coordenada entre si [15].

A ULisboa tem vindo a desenvolver e implementar um conjunto de sistemas de informação com o objetivo de tornar a sua administração mais eficiente, rápida e eficaz. As questões relacionadas com as tecnologias de informação e comunicação da ULisboa estão a cargo do Departamento de Informática dos serviços centrais da RUL. Este projeto foi elaborado no Núcleo de Desenvolvimento de *Software* (NDS) desse departamento, com o apoio do NM do Departamento de Relações Externas e Internacionais (DREI) da RUL.

1.6 Estrutura do documento

Este documento tem a seguinte estrutura:

- **Capítulo 1 – Introdução**
  Inclui a motivação, trabalho desenvolvido, contribuições e enquadramento institucional do projeto.

- **Capítulo 2 – Sistema Integrado de Gestão Académica Fenix**
  Apresentação do sistema Fenix, contemplando as tecnologias, plataformas e arquitetura do sistema, assim como uma descrição da ferramenta de gestão de *workflows* da Omnis.

- **Capítulo 3 – Análise e desenho**
  Descrição do levantamento e análise de requisitos para o projeto de mobilidade, assim como a arquitetura e o desenho escolhidos para o implementar.

- **Capítulo 4 – Implementação e avaliação**
  Inclui os principais passos efetuados durante a implementação do projeto, assim como os passos para o projeto ser colocado em produção e a avaliação efetuada.

- **Capítulo 5 – Conclusão**
  O último capítulo apresenta a conclusão deste projeto, assim como o trabalho futuro a realizar.

Capítulo 2

Sistema Integrado de Gestão Académica Fenix

Este capítulo apresenta o sistema integrado de gestão académica Fenix, estando dividido em duas partes. A primeira parte explica as tecnologias, plataformas e arquitetura deste sistema, enquanto a segunda parte descreve a ferramenta de gestão de *workflows* da plataforma Omnis.

2.1 Introdução

O sistema Fenix é um projeto que começou em 2002 no Instituto Superior Técnico com o objetivo de criar um sistema integrado de informação académica. Depois da fusão da Universidade Técnica de Lisboa com a Universidade Clássica de Lisboa, em 2015, este sistema integrado de gestão académica foi estendido e neste momento já está implementado na RUL e em dezassete escolas da ULisboa. O sistema conta com múltiplas funcionalidades com o intuito de agilizar e facilitar a gestão académica para professores, serviços académicos e alunos [4].

2.2 Tecnologias

O sistema Fenix utiliza um conjunto de plataformas criadas especificamente para este sistema, que serão abordadas na secção seguinte, e um conjunto de tecnologias, entre elas:

- **Java 11**
  Java é uma linguagem de programação orientada a objetos, que traz grandes vantagens. É uma linguagem bastante utilizada em todo o mundo, existe uma grande quantidade de comunidades Java, ou seja, é muito fácil obter documentação, ou encontrar fóruns sobre esta linguagem, tornando-se mais fácil programar com Java. Existem também bastantes *frameworks* que facilitam o desenvolvimento em Java, e

uma das suas características é que, quando se compila em Java, é gerado um byte-code que é interpretado numa *Java Virtual Machine* (JVM), o que faz com que seja possível executar Java em qualquer sistema operativo que tenha uma JVM [9].

- **Maven**
  Maven é uma ferramenta *open-source* que tem como objetivo simplificar a construção de projetos Java, isto é, com esta ferramenta não precisamos de nos preocupar com as dependências uma vez que esse processo é feito automaticamente através de um ficheiro *EXtensible Markup Language* (XML), designado por *Project Object Model* (POM), que contém todos os detalhes do projeto [8].

- **MySQL**
  MySQL é um sistema *open-source* de gestão de bases de dados relacionais, ou seja, com esta ferramenta é possível armazenar os dados em tabelas separadas com regras próprias, sendo que o conjunto dessas tabelas é a nossa base de dados. O sistema impõe estas regras quando a base de dados é populada de modo a nunca haver inconsistências [10].

- **Git**
  Git é um sistema de controlo de versões distribuído que permite gerir várias versões durante o desenvolvimento de um projeto, assim como manter um histórico dessas versões. Este sistema é bastante utilizado em desenvolvimento de *software* pois possibilita a criação de *branches*, o que permite fazer "experiências" com o código já criado, assim como organizar o código por funcionalidades e possibilita, ainda, que vários utilizadores trabalhem no mesmo projeto ao mesmo tempo sem problemas [6].

2.3 Plataformas

Nesta secção são apresentadas as duas plataformas do sistema Fenix utilizadas no módulo de mobilidade.

2.3.1 Plataforma Fenix

A plataforma Fenix suporta o desenvolvimento de aplicações em java que necessitam de um modelo de domínio persistente e transacional, permitindo que a equipa de desenvolvimento se abstraia dos requisitos do sistema relacionados com a persistência e a sincronização e se foque na concretização das regras do negócio. Para tal, foi criada uma linguagem de domínio, a *Domain Modeling Language* (DML), onde são descritas as definições das entidades de domínio, como as suas classes, tipos de valores e relações entre entidades.

Esta plataforma contém um gerador de classes que, quando executado, cria duas classes java para cada entidade da DML, uma classe ”entidade_base.java”, onde está representada a estrutura e o tipo de dados desta entidade, e a classe ”entidade.java”, onde é implementado o seu comportamento. Desta forma, a estrutura do domínio fica separada do seu comportamento.

A plataforma Fenix também garante coerência, uma vez que possibilita a gestão das transações feitas na base de dados. É utilizada uma *Java Versioned Software Transactional Memory*, que a partir de *Versioned Boxes* mantém o histórico das versões dos objetos. Sempre que existe uma mudança no objeto, essa mudança cria uma nova versão do objeto, ao invés de escrever por cima da versão já existente. Quanto à leitura, cada transação lê o número de versão do objeto, certificando-se de que cada leitura é feita na versão mais recente na altura da transação, garantindo assim que não há conflitos [5].

2.3.2 Plataforma Omnis

A plataforma Omnis possibilita a abstração da DML e da camada de apresentação, utilizando a estrutura *Create, Read, Update, Delete* (CRUD) para uma fácil utilização.

Esta plataforma utiliza uma linguagem específica de apresentação, a PSL, que representa os ecrãs e os fluxos de ecrã. O compilador da PSL cria as classes Java correspondentes aos ecrãs definidos na PSL, as quais podem ser modificadas para customizar os ecrãs.

A plataforma Omnis inclui também uma ferramenta de modelação de domínio e o seu compilador. Esta ferramenta utiliza uma linguagem específica de domínio, a DSL, onde são definidas as entidades, os seus campos e ainda as relações entre entidades. O compilador da DSL gera a DML, mapeando as entidades e as suas relações.

A grande diferença entre a DML, referida na plataforma Fenix, e a DSL é a forma como são definidas as relações entre entidades, enquanto na DML as relações são representadas fora das entidades, onde é especificada a multiplicidade das relações, na DSL essas relações são descritas dentro das entidades. São utilizadas as duas linguagens de modelação de domínio por razões de compatibilidade de versões e porque a DSL e a PSL não são linguagens independentes, logo, de forma a utilizarmos a PSL para representar os ecrãs na camada de apresentação, é necessário utilizar a DSL para modelar o domínio.

Esta plataforma inclui ainda outras ferramentas, como é o caso da ferramenta de gestão de *workflows*, que é abordada na secção 2.5.

2.4 Arquitetura do Fenix

A arquitetura do Fenix utiliza o padrão arquitetural *Model-View-Controller* (MVC) que é, neste momento, bastante comum na implementação de aplicações *web*. Este modelo divide a arquitetura em três camadas logicamente separadas, permitindo a colaboração

entre as camadas e ignorando a forma como cada uma está implementada. Desta forma, cada camada pode ser atualizada independentemente das outras, o que simplifica a manutenção da aplicação e facilita a sua evolução.

Esta secção apresenta as três camadas em que está organizada a arquitetura do Fenix, nomeadamente a camada de persistência, a camada de negócio e a camada de apresentação [3].

Camada de persistência

A camada de persistência é concretizada pela plataforma Fenix.

De modo a garantir a persistência dos objetos na base de dados relacional, é utilizada a técnica *Object Relational Mapper*, ou seja, as classes de domínio representam as tabelas da base de dados e cada instância da classe representa uma linha da tabela. Desta forma, o programador não necessita de implementar código na linguagem SQL.

Camada de negócio

A camada de negócio do Fenix é o elo de ligação entre a camada de persistência e a camada de apresentação.

Esta camada é responsável por receber os pedidos da camada de apresentação, fazer a sua análise e tratamento, podendo fazer pedidos de leitura e/ou escrita à camada de persistência, e devolver os dados à camada de apresentação. A camada de negócio é concretizada nas classes ”entidade.java” geradas pela DML.

Camada de apresentação

A camada de apresentação suporta a interação com o utilizador e faz a ponte entre os ecrãs e os objetos da camada de negócio.

Desde o início do desenvolvimento do sistema Fenix têm sido utilizadas várias ferramentas na implementação desta camada. No desenvolvimento do módulo da mobilidade são utilizadas as ferramentas da plataforma Omnis, nomeadamente a PSL.

2.5 *Workflows*

Workflow é um termo inglês que significa ”fluxo de trabalho”, isto é, uma sequência de passos necessários para automatizar processos de negócio.

A ferramenta de gestão de *workflows* da Omnis é uma ferramenta de modelação de processos baseada nos conceitos de estados e transições para definir a lógica de negócio. Esta ferramenta é composta por um modelador de processos e por um motor de execução, o qual é responsável por interpretar e executar o *workflow*. A figura 2.1 apresenta um exemplo de um *workflow* criado a partir da ferramenta de *workflows* da plataforma Omnis.

O modelador de processos é uma ferramenta de modelação gráfica que possibilita a definição de estados, transições, operações, notificações, controlo de acesso, formulários, separadores e documentos. A figura 2.2 representa os elementos da linguagem de modelação de *workflows* do Fenix.

De seguida são apresentados os elementos principais da linguagem de modelação de *workflows* da plataforma Omnis: estados, separadores, operações, permissões e notificações.

Estados

A execução de um *workflow* começa no estado inicial e passa por vários estados através da execução de operações até ao estado final. Os estados iniciais e finais têm representações próprias, como é visível na primeira linha da figura 2.2.

Existem dois tipos de estados, os automáticos e os manuais. Os estados automáticos são não-interativos, isto é, realizam operações automaticamente sem precisarem de *inputs* de utilizadores e transitam de estado depois de realizarem a operação. Estes estados podem ser condicionais, enviar notificações, lançar exceções ou executar operações. Os estados automáticos podem ter várias representações consoante a operação que executem, como é visível na segunda e terceira linhas da figura 2.2. Já os estados manuais podem ter ecrãs ou separadores, em que são necessárias ações por parte do utilizador, e necessitam que uma operação seja executada para transitarem para o estado seguinte. Este estado é representado por um círculo azul com um símbolo de uma pessoa a branco, como pode ser visualizado na segunda linha da figura 2.2.

Separadores

Separadores são ecrãs dos estados manuais que podem conter informação, formulários ou documentos da instância. Existem três tipos de separadores:

- **Separadores customizados** – São separadores que podem servir, por exemplo, para mostrar informação sobre o processo, para atualizar atributos do processo, ou até para monitorizar o processo. Existem alguns separadores já disponibilizados pela ferramenta de *workflows* da Omnis que podem ser utilizados em qualquer *workflow*. O separador de monitorização do processo é exemplo de um desses separadores.

- **Separadores de documentos** – São separadores onde é possível fazer o *upload/-download* de ficheiros na instância. Um separador de documentos pode ter um ou mais documentos associados. Estes documentos podem ser, ou não, obrigatórios para aquele separador.

- **Ecrãs dinâmicos** – Estes ecrãs permitem a criação de formulários sem ser necessário haver uma implementação do ecrã. Desta forma é possível recolher informação do utilizador sem ser necessária uma implementação em código.

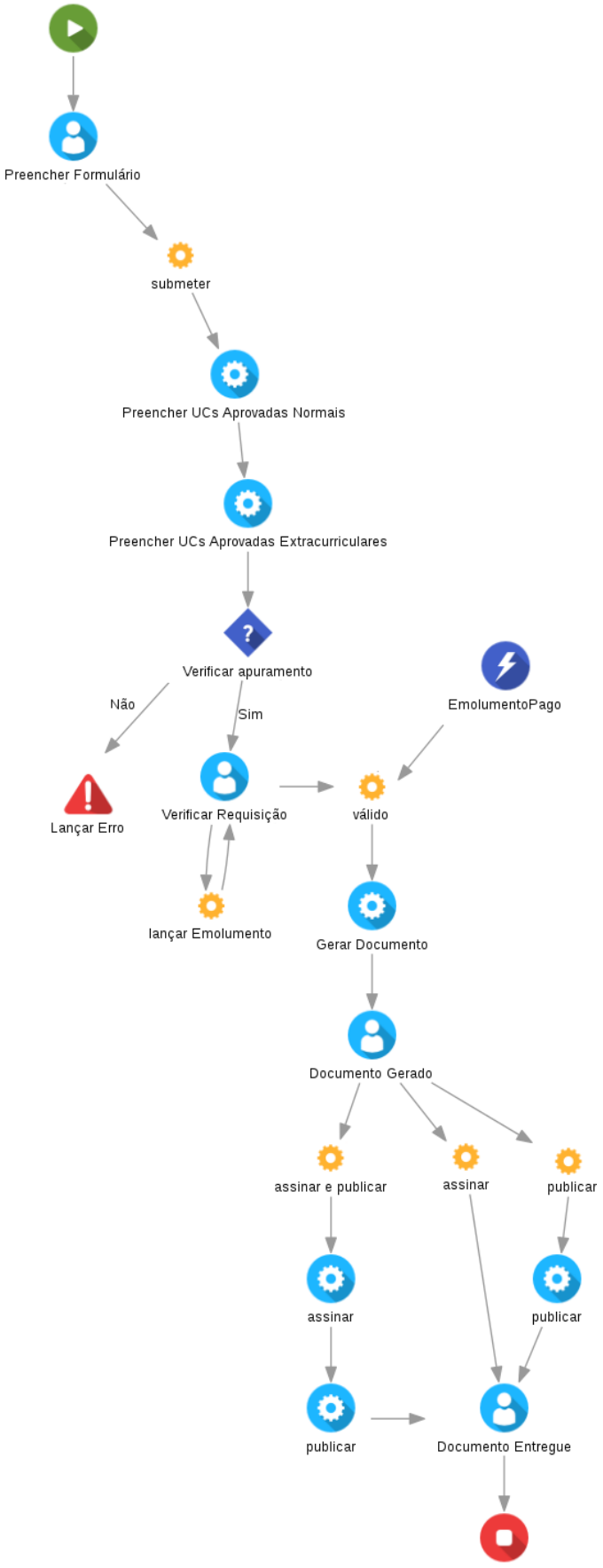

Figura 2.1: Exemplo de um *workflow* de requisição de certidão de conclusão

Figura 2.2: Elementos da linguagem de modelação de *workflows* do Fenix

Os separadores podem ter validadores, que servem para não deixar o processo avançar enquando o separador está inválido. Também existem alguns validadores já disponibilizados pela ferramenta de *workflows* da plataforma Omnis, como é o caso do validador de documentos. Se um separador de documentos possuir um validador de documentos, o processo não pode avançar enquanto não for feito o *download* dos documentos obrigatórios para aquele separador.

Operações

As operações permitem transitar de um estado manual para um estado manual ou automático. A sua execução pode ser desencadeada pelo utilizador ao premir um botão, por um evento, ou por um temporizador. Nestes dois últimos casos não é necessário ao utilizador indicar que deseja executar a operação, uma vez que esta é executada assim que o evento for chamado (como o evento ”EmolumentoPago” no exemplo da figura 2.1), ou assim que o tempo estipulado pelo temporizador for ultrapassado. A representação destes elementos pode ser visualizada na linha final da figura 2.2.

As operações podem ser do tipo ”transitar estado”, em que apenas transitam de um estado para outro, ou de outro tipo, em que podem fazer qualquer tipo de operação, como mudar dados da instância, gerar um documento pdf, ou até enviar notificações.

As operações podem ainda validar o estado. Neste caso, a operação só pode ser executada se não existir nenhum separador inválido. Caso a operação não valide o estado, esta pode ser executada, mesmo na presença de separadores inválidos.

Para além das configurações anteriores, também é possível acrescentar uma mensagem de confirmação; assim, quando se executa a operação, aparece essa mensagem e duas opções, para o utilizador poder confirmar ou cancelar a execução da operação.

Permissões

As permissões permitem que certos perfis apenas tenham acesso a separadores e a operações específicas. Para isso é necessário definir em cada uma das operações do *workflow* os perfis que têm permissões de execução, e em cada separador de cada estado os perfis que têm permissões de leitura e de escrita. Deste modo, as operações são apenas visíveis aos perfis que tenham permissões para as executar. Em cada estado, perfis diferentes podem ter permissões de leitura ou escrita para separadores diferentes.

Notificações

As notificações podem ser enviadas a perfis específicos à entrada de um estado, à saída de um estado ou a pedido e são recebidas por correio eletrónico. No caso em que a notificação é enviada a pedido, é desencadeada por uma operação do tipo ”enviar notificação”, ou por um estado automático do mesmo tipo.

2.6 Sumário

Este capítulo apresenta o sistema Fenix, assim como as tecnologias e plataformas utilizadas neste sistema. Foi ainda descrita a ferramenta de gestão de *workflows* da plataforma Omnis, os seus elementos da linguagem de modelaçao e as suas configurações. No capítulo seguinte são apresentados a anáilse e desenho do projeto, partindo do levantamento de requisitos.

Capítulo 3

Análise e desenho

Este capítulo engloba o levantamento e análise de requisitos deste projeto, assim como a arquitetura e desenho propostos para o módulo de mobilidade.

3.1 Levantamento de requisitos

O levantamento de requisitos foi feito com o NM do DREI da RUL em três reuniões, complementadas com outras sessões onde foram abordados alguns aspectos dos processos com mais detalhe. Na primeira reunião foi apresentado o processo normal de mobilidade que passa pela RUL. Após a análise de requisitos inicial deste processo, foi feita uma segunda reunião onde estes requistos foram revistos e detalhados. As exceções que podem acontecer durante o decorrer do processo de mobilidade foram analisadas durante a terceira reunião.

O fluxo normal do processo de mobilidade, abordado nas primeira e segunda reuniões, divide-se em três processos: a emissão e assinatura de contratos, o pagamento da primeira tranche e o fecho e pagamento da segunda tranche, detalhados nas subsecções 3.1.1, 3.1.2, 3.1.3, respectivamente. A secção 3.1.4 detalha os casos de exceção.

3.1.1 Emissão e assinatura de contratos

O processo começa quando o NM recebe das escolas os dados e os documentos dos alunos que já foram previamente aceites para fazerem mobilidade. Esses documentos são: a ficha de estudante, a cópia do documento de identificação, o comprovativo de dados bancários e o *Learning Agreement* (LA). O LA é um documento com a correspondência entre as unidades curriculares que o aluno irá frequentar na escola de destino e as unidades curriculares a que o aluno irá ter equivalência na escola de origem. Este documento tem de ser assinado por três partes: o aluno, a escola de origem e a escola de destino. Estas assinaturas podem demorar bastante tempo, principalmente a assinatura por parte da escola de destino, o que acaba por provocar atrasos no processo de emissão dos contratos.

Assim que são recebidos os dados, é revisto o orçamento de forma a verificar se este

foi respeitado por cada escola. Se não for esse o caso, os dados são devolvidos às escolas para serem corrigidos. Depois dos dados estarem corrigidos e existir a confirmação de que o orçamento é respeitado, o NM passa para a criação de uma tabela excel (tabela A) com os dados dos alunos de todas as escolas que pretendem fazer mobilidade. Durante a montagem desta tabela, o NM verifica se os alunos têm direito ao ciclo de estudos no estrangeiro a que se estão a propor, uma vez que cada aluno só tem direito a doze meses, se estiver num curso de licenciatura, ou a vinte e quatro meses, se estiver num curso de mestrado integrado.

De seguida, o NM atribui as bolsas aos alunos. Os dias de mobilidade podem não ser pagos na íntegra; garante-se o pagamento de, pelo menos, cento e cinquenta dias de mobilidade. Quando os alunos não têm direito a bolsa, porque não querem, ou porque não houve verba suficiente para lhes ser atribuída uma bolsa, esses alunos são denominados de ”alunos com bolsa zero”.

Com tudo verificado e as bolsas atribuídas, é enviado um e-mail para a área financeira com uma tabela excel (tabela B) com os dados dos alunos, para se dar início aos pedidos de cabimento, à atribuição de compromisso e aos compromissos plurianuais (caso a mobilidade abranja dois anos civis diferentes) para cada aluno. Enquanto isso se passa do lado da área financeira, no NM faz-se o registo dos alunos na plataforma *Online Linguistic Support* (OLS), para o envio automático de licenças de testes de línguas e curso de línguas.

Quando os números de compromisso e os encargos plurianuais são recebidos, são passados para a tabela A. De seguida é feita a impressão dos contratos de mobilidade, que podem ter *templates* diferentes consoante o tipo de mobilidade (de estudos ou de estágio), se é uma mobilidade com consórcio ou não, ou se o aluno tem algum tipo de bolsa diferente [bolsa zero, bolsa Necessidades educativas especiais (NEE), bolsa Serviços de Ação Social (SAS)]. Para finalizar o processo, o contrato é validado e assinado pelo Vice-Reitor responsável pela área académica e de mobilidade da RUL.

O passo seguinte é a assinatura dos contratos pelos alunos, o qual depende das assinaturas prévias dos LAs pelas três partes. Os alunos que têm os LAs assinados são convocados para uma sessão de assinaturas, que acontece na RUL (são convocados cerca de cem alunos por dia para assinarem os contratos presencialmente).

Os alunos que não têm o LA assinado, muitas vezes acabam por ir para mobilidade sem terem os contratos assinados e tem de ser um procurador a ir assinar o contrato, quando o LA estiver assinado.

3.1.2 Pagamento da primeira tranche

Com os contratos assinados, passa-se para a elaboração da informação de pagamento da primeira tranche (oitenta por cento do valor da bolsa). É criado um documento word com a informação de pagamento de vários alunos e é ainda criada mais uma tabela excel

(tabela C) com os dados de pagamento dos alunos. De seguida, esta informação é verificada pelo NM, que regista os dados numa outra tabela excel (tabela C'), igual à tabela C, mas com menos dados. Depois de verificada a informação, é necessária a assinatura da diretora do DREI, para de seguida a tabela C' ser enviada por correio eletrónico para o Vice-Reitor responsável pela área académica e de mobilidade da RUL e para a área financeira. O Vice-Reitor e o conselho de gestão verificam e autorizam os pagamentos e é enviada a informação autorizada para a área financeira para a realização dos pagamentos.

O NM ainda digitaliza os contratos assinados, para disponibilização à área financeira, e regista os alunos na *Mobility Tool+* (MT+) para fechar este processo.

3.1.3 Fecho e pagamento da segunda tranche

Este processo inicia aquando do envio, pelos gabinetes de relações internacionais das escolas, dos documentos de final de mobilidade, sendo estes a declaração de estada, com as datas da mobilidade, e o TR. Este último é um documento com as notas e os *European Credit Transfer System* (ECTS) das unidades curriculares a que o aluno se inscreveu na escola de destino.

O NM começa o processo verificando se os alunos submeteram o relatório final de mobilidade na MT+ e se fizeram o segundo teste na OLS. Se os alunos não tiverem feito alguma destas duas tarefas, o NM envia um e-mail a relembrá-los que têm de as fazer. De seguida registam as datas de mobilidade da declaração de estada na tabela A. Com essa informação, a tabela faz, automaticamente, a contagem do número de dias de mobilidade realizada e é registado o número de dias de bolsa a que o aluno tem direito. De seguida, é validado o aproveitamento do aluno, isto é, os alunos quando assinam contrato comprometem-se a ter um aproveitamento de pelo menos seis ECTS, em caso de mobilidade semestral, ou doze ECTS, nos casos de mobilidade anual. Se os alunos não tiverem aproveitamento segundo os ECTS do TR, o NM contacta as escolas de origem de modo a perceber se existe aproveitamento segundo os ECTS da escola de origem. Se houver aproveitamento segundo um destes dois métodos, considera-se que o aluno teve aproveitamento. Se o aluno não teve aproveitamento, tem de devolver o valor da primeira tranche na íntegra e não recebe o valor da segunda tranche, o que implica um pedido de devolução através da emissão de um ofício, que terá de ser posteriormente assinado pelo Vice-Reitor responsável pela área académica e de mobilidade da RUL.

Para os alunos com aproveitamento, se o aluno cumpriu o número de dias estabelecido no contrato, a bolsa mantém-se. Se o aluno reduziu o número de dias de mobilidade em relação ao contrato, considera-se o novo número de dias para recalcular a bolsa efetiva a que o aluno tem direito. Neste caso, se o valor da bolsa final for maior do que o valor pago na primeira tranche, ajusta-se o valor da segunda tranche, caso contrário, se o valor da bolsa final for menor do que o valor pago na primeira tranche, emite-se um pedido de devolução.

Depois de todos os ajustes, o NM volta a verificar, na MT+, se o aluno submeteu o relatório final e na OLS se o aluno realizou o segundo teste de língua, ficando apenas com a indicação, na tabela A, de que o aluno fez ou não o teste da OLS e, nos casos em que falta submeter o relatório final na MT+, o processo fica em espera até à sua submissão.

Caso haja lugar ao pagamento da segunda tranche, repetem-se os passos do pagamento da primeira tranche, desde a elaboração da informação de pagamento à realização do pagamento, e é enviado um e-mail aos alunos a informar que o processo de ERASMUS foi encerrado.

No caso de haver alterações nas datas de mobilidade, essas alterações também são indicadas na MT+, assim como o número de ECTS reconhecidos pela escola.

No final de todo o processo, o NM, tem ainda de fazer um relatório para o Registo de Alunos Inscritos e Diplomados do Ensino Superior (RAIDES) com os dados estatísticos dos alunos que foram para mobilidade.

3.1.4 Exceções

Para além do cenário principal de sucesso, foram também apresentadas as seguintes exceções durante o levantamento de requisitos:

- **Redistribuição do financiamento não executado**
  Quando os dados são enviados para os serviços centrais, e depois do orçamento já ter sido verificado, o NM começa a atribuir as bolsas e é feita uma redistribuição do financiamento não executado, isto é, no caso de sobrar verba de uma escola, é alocada noutra que tenha alunos a bolsa zero (alunos aos quais não foi atribuída bolsa por falta de orçamento da escola), com o intuito de maximizar o gasto da verba atribuída.

- **Estudantes bolseiros de ação social**
  Quando existem estudantes bolseiros de ação social, estes têm direito a receber uma bolsa mensal suplementar dos SAS. Nestes casos, os alunos, quando se candidatam, assinalam que são bolseiros de ação social e essa informação fica, também, assinalada no contrato e na MT+.

- **Participantes com NEE**
  Quando existem participantes com NEE, estes candidatam-se com essa indicação e, quando o NM cria a tabela A, essa informação é retida e é também assinalada no contrato e na MT+. Estes alunos podem ter direito a uma bolsa suplementar paga pela Agência Nacional (AN), que é dividida em duas tranches, a primeira (60%) paga em conjunto com a primeira tranche da bolsa de mobilidade e a segunda (40%) paga no fim da mobilidade.

- **ERASMUS para recém-graduados**
  Neste caso, os participantes têm de terminar o estágio até à véspera de perfazer um ano de conclusão do curso e a duração mínima da mobilidade é de dois meses e a máxima de doze meses. O NM tem, portanto, de verificar essas datas durante a emissão dos contratos.

- **Desistências**
  Quando um aluno desiste da mobilidade, contacta o gabinete de relações internacionais da escola, que posteriormente contacta o NM da RUL. O NM altera um campo na tabela A para indicar que o aluno desistiu e o orçamento é libertado para uso posterior. Se o contrato já estiver assinado, é ainda emitido um ofício de devolução.

- **Prazos não cumpridos**
  Muitas vezes, algumas escolas demoram mais tempo que o estipulado a enviar os dados dos alunos que se candidataram para fazer um ciclo de estudos no estrangeiro, o que dificulta a redistribuição do financiamento não executado. Nestes casos, o NM vai começando a atribuir as bolsas e, no caso de sobrar verba de uma escola, é alocada noutra com o objetivo de maximizar o gasto da verba atribuída. As escolas que se atrasam a enviar os dados dos alunos têm mais probabilidade de ficar com alunos a bolsa zero.

- **Antecipação de regresso**
  Se o participante regressar antes da conclusão das unidades curriculares a que se propôs, o NM assinala-o nas plataformas OLS e MT+ e, por norma, o aluno tem de devolver a primeira tranche e não tem direito ao pagamento da segunda.

- **Força maior**
  Existem casos de desistência, regresso antecipado, ou falta de aproveitamento que são geridos de forma diferente se for por causas de força maior, por exemplo doença. Nestes casos, o NM envia o processo para a AN, que o vai avaliar e decidir se implica ou não devolução total ou parcial da bolsa.

- **Adendas**
  Quando existem modificações ao contrato, como aumento de tempo de mobilidade, ou existe algum erro no contrato que é necessário ser alterado e o contrato já foi assinado, são criadas adendas ao contrato com essas alterações para que não seja necessário imprimir e assinar novo contrato.

- **Modificações no contrato**
  Quando existem modificações nos projetos de estágio, adendas ou anulações de contrato, é necessário criar ofícios.

3.2 Análise de requisitos

Esta secção apresenta os atores, os requistos funcionais e os requistos não funcionais do projeto, partindo do levantamento de requisitos, sendo que alguns processos são simplificados para facilitar o uso do sistema, como é o caso do processamento de chegada em que os estados onde existem atualizações na MT+, na análise de requisitos passam a ser um único estado.

3.2.1 Atores

Os atores que vão utilizar o sistema são:

- **NM**
- **Alunos**
- **Diretora do DREI**
- **Área financeira da RUL**
- **Vice-Reitor responsável pela área académica e de mobilidade da RUL**
- **Conselho de Gestão**

3.2.2 Requisitos funcionais

Os requisitos funcionais deste projeto, que foram identificados a partir do levantamento de requisitos, são apresentados nesta secção como casos de uso.

***Use Case* (UC)-01 Criar contrato**

Nome: Criar contrato
Ator principal: NM
Período: Final de maio, junho e início de julho para todos os alunos, e novembro e dezembro para novos alunos de segundo semestre
Pré-condições:

- *Template* de contrato de mobilidade criado,
- O aluno ficou colocado em candidatura *Outgoing* da sua Escola,
- O aluno enviou os documentos necessários (ficha de estudante, cópia do documento de identificação e comprovativo de dados bancários).

Pós-condições:

- O contrato é criado digitalmente com base em *templates* e validado pelo NM.

Cenário principal de sucesso:

1. O NM verifica que os orçamentos das escolas são cumpridos,
2. O NM atribui as bolsas aos alunos,
3. O NM verifica que não sobra verba,

4. O NM regista o alunos na plataforma OLS a partir de uma tabela exportada do sistema,
5. O sistema verifica que o aluno não é bolsa zero,
6. O NM exporta uma tabela do sistema e envia para a área financeira preencher o número de compromisso,
7. A área financeira faz a atribuição do compromisso, desenvolve o compromisso plurianual, caso seja necessário, e o pedido de cabimento e envia para o NM,
8. O NM carrega a tabela com o número de compromisso no sistema,
9. O sistema cria o contrato automaticamente a partir de *templates* dependentes dos parâmetros identificados no levantamento de requisitos,
10. O NM valida o contrato.

Cenário alternativo 1:

1a. - O NM verifica que os orçamentos das escolas não são cumpridos:

1. O sistema informa a escola que necessita de revisão do orçamento,
2. A escola corrige a distribuição do orçamento,
3. O sistema volta a receber os dados dos alunos,
4. O fluxo continua a partir do ponto 1. do cenário principal de sucesso.

Cenário alternativo 2:

3a. - O NM verifica que sobra verba:

1. O NM faz a redistribuição do financiamento não executado,
2. O fluxo continua a partir do ponto 4. do cenário principal de sucesso.

Cenário alternativo 3:

5a. - O sistema verifica que o aluno é bolsa zero:

1. O fluxo salta para o ponto 9. do cenário principal de sucesso.

Cenário alternativo 4:

10a. - O NM deteta erros no contrato:

1. O NM corrige os dados do contrato,
2. O sistema volta a criar o contrato automaticamente,
3. O NM valida o contrato.

Cenário alternativo 5:

Qualquer ponto - O aluno desiste da mobilidade:

1. O NM é avisado pelo aluno que este desistiu da mobilidade,
2. O NM assinala no sistema que o aluno desistiu da mobilidade,
3. O sistema informa a escola da desistência,
4. A área financeira faz o descabimento do valor da bolsa (**nota**: este ponto só acontece se a desistência for depois do ponto 6. do cenário principal de sucesso),
5. O NM elimina o aluno das plataformas OLS e MT+,
6. O NM finaliza o processo de mobilidade.

UC-02 Assinaturas do contrato

Nome: Assinaturas do contrato
Ator principal: Vice-Reitor, aluno
Período: Antes do período de mobilidade
Pré-condições:

- O contrato foi criado e validado.

Pós-condições:

- O contrato está assinado digitalmente por todas as partes.

Cenário principal de sucesso:

1. O Vice-Reitor assina o contrato com assinatura digital,
2. O NM exporta uma tabela para fazer o registo do aluno na plataforma MT+,
3. O sistema verifica se o LA foi importado no sistema,
4. O sistema envia uma notificação ao aluno para este ir assinar o contrato,
5. O aluno assina o contrato com chave móvel digital.

Cenário alternativo 1:

1a. - O Vice-Reitor encontra um erro num contrato:

1. O Vice-Reitor informa que encontrou um erro,
2. O Vice-Reitor explica o erro que encontrou no contrato,
3. O NM recebe uma notificação de erro de contrato,
4. O NM atualiza os dados do contrato,
5. O sistema gera um novo contrato,
6. O fluxo continua a partir do ponto 1. do cenário principal de sucesso.

Cenário alternativo 2:

3a. - O LA não foi importado no sistema:

1. O sistema fica à espera que o LA seja importado no sistema para passar para o próximo ponto do cenário principal de sucesso.

Cenário alternativo 3:

5a. - O aluno encontra um erro no contrato:

1. O aluno informa que encontrou um erro,
2. O aluno explica o erro que encontrou no contrato,
3. O NM recebe uma notificação de erro de contrato,
4. O NM atualiza os dados do contrato,
5. O sistema gera um novo contrato,
6. O fluxo continua a partir do ponto 1. do cenário principal de sucesso.

Cenário alternativo 4:

5b. - O aluno informa que não possui chave móvel digital (por ser estrangeiro, ou por outro motivo):

1. O NM avança o processo para um estado de assinatura manual,
2. O sistema elimina o contrato assinado pelo Vice-Reitor,

3. O sistema envia uma notificação ao aluno para este ir assinar o contrato,
4. O aluno faz o download do contrato e assina-o manualmente,
5. O aluno terá de se dirigir à RUL para entregar o contrato,
6. O Vice-Reitor assina o contrato manualmente,
7. O NM faz o upload do contrato assinado manualmente pelas duas partes.

Cenário alternativo 5:

Qualquer ponto - O aluno desiste da mobilidade:

1. O NM é avisado pelo aluno que este desistiu da mobilidade,
2. O NM assinala no sistema que o aluno desistiu da mobilidade,
3. O sistema informa a escola da desistência,
4. A área financeira faz o descabimento do valor da bolsa,
5. O NM elimina o aluno das plataformas OLS e MT+,
6. O NM finaliza o processo de mobilidade.

UC-03 Pagamento da primeira tranche

Nome: Pagamento da primeira tranche
Ator principal: Área financeira
Atores secundários: Vice-Reitor, Diretora do DREI, Conselho de gestão
Período: Depois das assinaturas dos contratos
Pré-condições:

- Os contratos foram assinados pelos alunos e Vice-Reitor.

Pós-condições:

- O pagamento da primeira tranche foi concluído.

Cenário principal de sucesso:

1. O sistema gera a informação de pagamento da primeira tranche,
2. O NM valida a informação de pagamento,
3. A Diretora do DREI assina digitalmente a informação de pagamento,
4. O Vice-Reitor verifica a informação de pagamento,
5. O Vice-Reitor dá o seu parecer,
6. O Conselho de Gestão autoriza o pagamento,
7. A área financeira procede ao pagamento da primeira tranche,
8. O aluno recebe uma notificação a indicar que irá receber o pagamento.

Cenário alternativo 1:

Qualquer ponto depois da informação de pagamento estar criada - É encontrado um erro na informação de pagamento:

1. O NM atualiza os dados da informação de pagamento,
2. O sistema volta a criar a informação de pagamento,
3. O fluxo continua a partir do ponto 2. do cenário principal de sucesso.

Cenário alternativo 2:

Qualquer ponto - O aluno desiste da mobilidade:

1. O NM é avisado pelo aluno que este desistiu da mobilidade,
2. O NM assinala no sistema que o aluno desistiu da mobilidade,
3. O sistema informa a escola da desistência,
4. A área financeira faz o descabimento do valor da bolsa,
5. O NM elimina o aluno das plataformas OLS e MT+,
6. O NM finaliza o processo de mobilidade.

Nota: Os alunos com bolsa zero não passam por este UC.

UC-04 Processamento de chegada

Nome: Processamento de chegada
Período: Depois do aluno chegar da mobilidade
Pré-condições:

- O aluno voltou da mobilidade.

Pós-condições:

- O processo está encerrado ou preparado para o pagamento da segunda tranche.

Cenário principal de sucesso:

1. O aluno preenche e submete as informações e os documentos necessários ao fecho do processo (Declaração de estada e TR),
2. O sistema verifica se tem todos os documentos necessários para o fecho do processo,
3. O NM verifica que as informações preenchidas coincidem com os documentos submetidos,
4. O sistema verifica que o aluno teve aproveitamento segundo o TR,
5. O sistema verifica que o aluno não é bolsa zero,
6. O sistema confere que o aluno cumpriu o número de dias do contrato.

Cenário alternativo 1:

4a. - O aluno não teve aproveitamento segundo o TR:

1. O sistema comunica com a escola,
2. A escola comunica o número de ECTS a que o aluno teve aproveitamento,
3. O sistema verifica que o aluno teve aproveitamento segundo os ECTS da escola de origem,
4. O processo continua a partir do ponto 5. do cenário principal de sucesso.

Cenário alternativo 1.1:

3b. - O sistema verifica que o aluno não teve aproveitamento segundo os ECTS da escola de origem:

1. O sistema verifica que o aluno não é bolsa zero,
2. O sistema emite um ofício de devolução,
3. O sistema envia uma notificação ao Vice-Reitor,

4. O Vice-Reitor assina digitalmente o ofício,
5. O NM exporta uma tabela de sistema para enviar à tesouraria,
6. O sistema envia uma notificação ao aluno sobre o valor da devolução,
7. O aluno paga a dívida e submete o comprovativo de devolução no sistema,
8. O NM valida o comprovativo de devolução,
9. O NM atualiza a MT+,
10. O sistema envia um e-mail ao aluno a informar que o processo de ERASMUS está encerrado,
11. O sistema comunica à escola o fecho do processo ERASMUS,
12. O sistema fecha o processo de ERASMUS no Fenix.

Cenário alternativo 1.1.1:

1c. - O sistema verifica que o aluno é bolsa zero:

1. O cenário continua a partir do ponto 9. do cenário alternativo 1.1.

Cenário alternativo 2:

5a. - O sistema verifica que o aluno é bolsa zero:

1. Este cenário é igual ao cenário alternativo 1.1.1.

Cenário alternativo 3:

6a. - O aluno não cumpriu o número de dias do contrato:

1. O sistema verifica que o aluno reduziu o número de dias do contrato mais de cinco dias,
2. O sistema recalcula o valor da bolsa,
3. O sistema calcula que o valor pago na primeira tranche é menor do que o novo valor da bolsa,
4. O sistema ajusta o valor da segunda tranche.

Cenário alternativo 3.1:

3b. - O sistema calcula que o valor pago na primeira tranche é maior do que o novo valor da bolsa:

1. O sistema calcula o valor da devolução,
2. O resto deste cenário é igual ao cenário alternativo 1.1.

UC-05 Pagamento da segunda tranche

Nome: Pagamento da segunda tranche
Ator principal: Área financeira
Atores secundários: Vice-Reitor, Diretoria do DREI, Conselho de gestão
Período: Depois do processamento de chegada
Pré-condições:

- Todo o processamento de chegada foi efetuado, incluindo o recálculo do valor

da bolsa, se for esse o caso.

Pós-condições:

- O pagamento da segunda tranche foi concluído e o processo de ERASMUS encerrado.

Cenário principal de sucesso:

1. O sistema cria a informação de pagamento da segunda tranche,
2. O NM valida a informação de pagamento,
3. A Diretora do DREI assina digitalmente a informação de pagamento,
4. O Vice-Reitor verifica a informação de pagamento,
5. O Vice-Reitor dá o seu parecer,
6. O Conselho de Gestão autoriza o pagamento,
7. A área financeira procede ao pagamento da segunda tranche,
8. O NM atualiza a MT+,
9. O sistema envia um e-mail ao aluno a informar que irá receber o pagamento e que o processo de ERASMUS está encerrado,
10. O sistema comunica à escola o fecho do processo ERASMUS,
11. O sistema fecha o processo de ERASMUS.

Cenário alternativo 1:

Qualquer ponto depois da informação de pagamento estar criada - É encontrado um erro na informação de pagamento:

1. O NM atualiza os dados da informação de pagamento,
2. O sistema volta a criar a informação de pagamento,
3. O fluxo continua a partir do ponto 2. do cenário principal de sucesso.

Nota: Os alunos com bolsa zero não passam por este UC.

UC-06 Exceção - Criar adendas

Nome: Exceção - Criar adendas

Ator principal: NM

Período: Entre o final do pagamento da primeira tranche e o início do processamento de chegada

Pré-condições:

- O contrato foi assinado e a primeira tranche foi paga.

Pós-condições:

- A adenda fica criada e, nos casos em que é necessário, o reforço do pagamento é efetuado.

Cenário principal de sucesso:

1. O NM indica no sistema que quer criar uma adenda,
2. O NM preenche o que necessita de ser modificado,
3. O sistema cria a adenda,

4. O NM indica que a adenda mantém o valor da bolsa,
5. O sistema volta ao ponto onde estava antes da indicação de criação da adenda.

Cenário alternativo 1:

4a - O NM indica que a adenda é com prolongamento de bolsa:

1. O sistema faz as contas dos dias de mobilidade, dias de bolsa, valor de bolsa, valor de primeira tranche, valor de segunda tranche e diferença do novo valor de primeira tranche e valor já pago da primeira tranche,
2. O NM exporta uma tabela do sistema e envia para a área financeira fazer o reforço do compromisso com o novo valor da primeira tranche,
3. A área financeira faz o pagamento do restante valor da primeira tranche,
4. O NM indica no sistema que o pagamento foi efetuado,
5. O sistema volta ao ponto onde estava antes da indicação de criação da adenda.

UC-07 Exceção - Força maior

Nome: Exceção - Força maior
Ator principal: NM
Atores secundários: Aluno
Período: Entre o final do pagamento da primeira tranche e o início do processamento de chegada
Pré-condições:

- O contrato foi assinado e a primeira tranche foi paga.

Pós-condições:

- O processo de força maior é fechado, quer seja aceite ou rejeitado.

Cenário principal de sucesso:

1. O NM indica no sistema que quer iniciar o processo de força maior,
2. O aluno submete os documentos comprovativos de despesas,
3. O NM envia os documentos e a justificação de força maior à AN,
4. O sistema fica a aguardar a resposta da AN,
5. A AN aceita o processo de força maior e devolve um valor de bolsa final,
6. O NM insere esse valor no sistema,
7. O sistema verifica que o valor da bolsa é maior do que o pago na primeira tranche,
8. O sistema faz as contas do novo valor da segunda tranche,
9. O processo continua para o Pagamento da segunda tranche.

Cenário alternativo 1:

6a - A AN rejeita o pedido de força maior:

1. O NM indica no sistema que a AN rejeitou o pedido,
2. O sistema volta ao ponto onde estava antes de iniciar o processo de força maior.

Cenário alternativo 2:

8a - O sistema verifica que o valor da bolsa é menor do que o pago na primeira tranche:

1. O sistema faz as contas do valor de devolução,
2. O sistema emite um ofício de devolução,
3. O sistema envia uma notificação ao Vice-Reitor,
4. O Vice-Reitor assina digitalmente o ofício,
5. O NM exporta uma tabela de sistema para enviar à tesouraria,
6. O sistema envia uma notificação ao aluno sobre o valor da devolução,
7. O aluno paga a dívida e submete o comprovativo de devolução no sistema,
8. O NM valida o comprovativo de devolução,
9. O NM atualiza a MT+,
10. O sistema envia um e-mail ao aluno a informar que o processo de ERASMUS está encerrado,
11. O sistema comunica à escola o fecho do processo ERASMUS,
12. O sistema fecha o processo de ERASMUS no Fenix.

3.2.3 Requisitos não funcionais

O sistema Fenix já responde a vários dos requistos não funcionais identificados para o módulo da mobilidade, nomeadamente autenticação, confidenciaidade, histórico de execução de operações e desempenho. Adicionalmente, foram identificados os seguintes requistos não funcionais específicos para este módulo:

- **Segurança na transferência de dados**: Os dados devem ser transferidos com segurança nas comunicações entre as escolas e a RUL,
- **Normalização**: O módulo deve manter a arquitetura existente no sistema.

3.3 Arquitetura

Para automatizar o processo de mobilidade é fundamental que a instância Fenix da RUL comunique com vários componentes, como representado na figura 3.1.

A comunicação entre as instâncias Fenix das escolas e da RUL é essencial durante a receção dos dados dos alunos que já foram previamente selecionados pelas escolas e, no final do processo de mobilidade, para enviar as informações de mobilidade dos alunos para as escolas de origem. Podem ainda ser necessárias algumas comunicações a meio do processo, para gerir certas exceções do processo. Esta comunicação entre as duas instâncias é feita a partir de serviços *web* que utilizam o protocolo de comunicação *Representational State Transfer* (REST) e o formato *JavaScript Object Notation* (JSON) como forma de representação dos dados.

Para que os alunos consigam interagir com a instância de *workflow* dos seus processos é importante que consigam aceder à instância Fenix da RUL. De forma a não ser

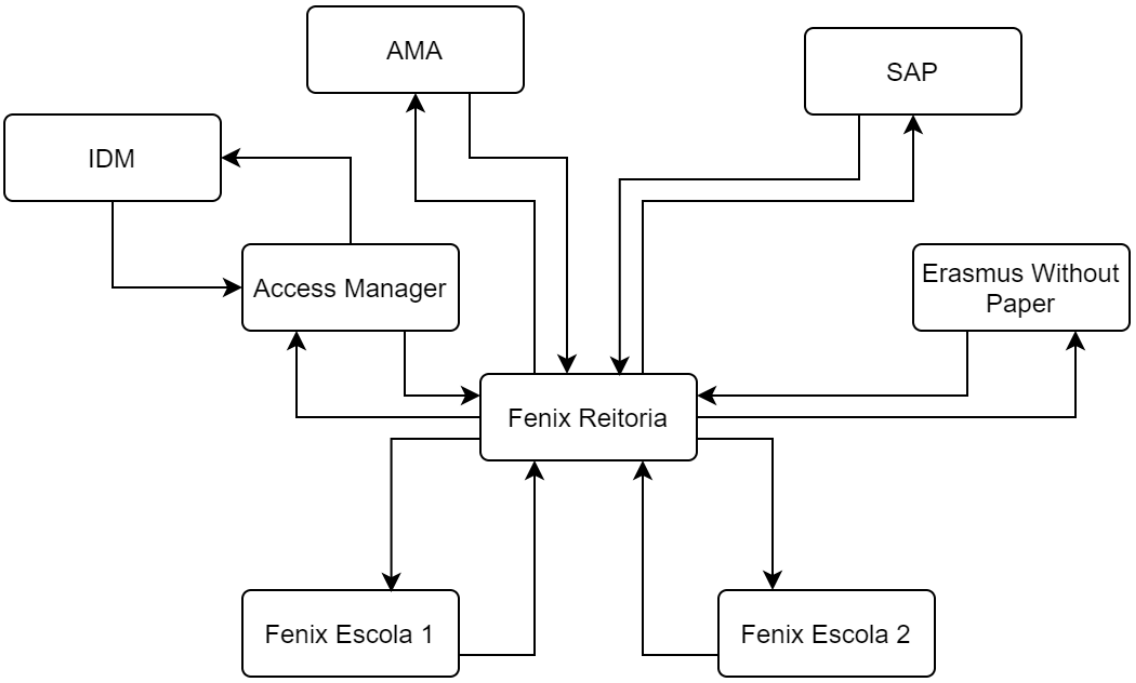

Figura 3.1: Desenho da arquitetura do módulo de mobilidade

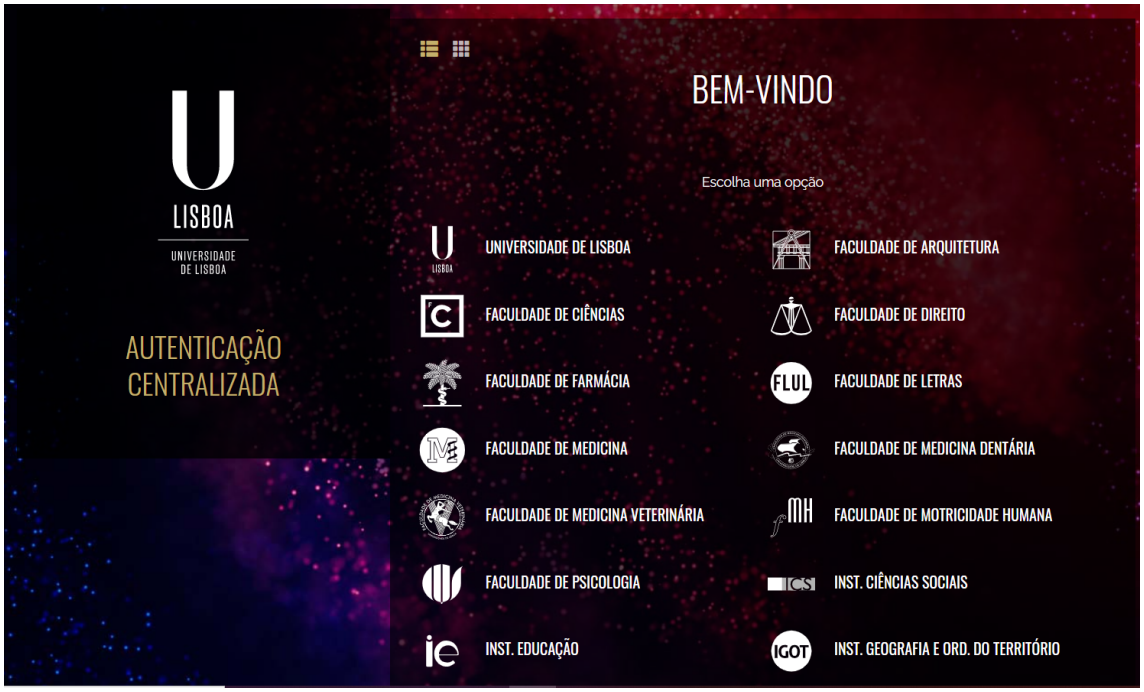

Figura 3.2: Página de autenticação centralizada

necessário criar novos dados de autenticação para os alunos acederem ao Fenix da RUL, foi utilizada a autenticação centralizada do sistema Fenix.

Para explicar em que consiste o método de autenticação centralizada da ULisboa é necessário ter em mente dois conceitos: *Access Manager* (AM) e *Identity Manager* (IDM). O AM inclui um conjunto de servidores e disponibiliza uma página de autenticação com várias instituições que utilizam diferentes métodos de autenticação. O IDM é um repositório com todos os utilizadores da ULisboa.

Quando o aluno tenta entrar no sistema Fenix da RUL, escolhe a opção de autenticação centralizada e é redireccionado para a página de autenticação fornecida pelo AM. Nesta página o aluno escolhe a instituição com que se quer autenticar, figura 3.2, que o redirecciona para a página de autenticação da instituição escolhida e autentica-se com o seu username e password de instituição. No final deste passo, o identificador do aluno é passado ao IDM, este identifica o utilizador e devolve os dados ao AM, que os envia ao sistema Fenix. Assim o aluno fica autenticado no sistema Fenix da RUL sem necessitar de novos dados de autenticação.

A comunicação entre a instância Fenix da RUL e a Agência para a Modernização Administrativa (AMA) é fundamental para o módulo de mobilidade conseguir utilizar a assinatura com chave móvel digital nos documentos gerados pelo módulo, uma vez que é este instituto que fornece este serviço.

Para que os pagamentos das bolsas sejam tratados é necessário que exista comunicação entre a instância Fenix da RUL e o *Systems Applications and Products* (SAP) (serviço financeiro utilizado na RUL). No entanto esta integração não estava prevista para este trabalho, assim como a integração entre o sistema Fenix da instância da RUL e o projeto *Erasmus Without Paper*, que será uma forma de simplificar as comunicações e trocas de dados entre a escola de origem e a escola de destino de uma forma segura.

3.4 Desenho

Esta secção apresenta a proposta de desenho para o módulo de mobilidade, através do diagrama de classes, e do processo de negócio, este último representado através da linguagem de modelação de *workflows* do Fenix.

3.4.1 Diagrama de classes

A imagem 3.3 apresenta o diagrama de classes correspondente ao módulo de mobilidade criado para o sistema Fenix.

A classe *Person* é uma classe do módulo *academic* do Fenix, e tem os atributos necessários para criar um utilizador no sistema. A classe *Student* estende a classe *Person*, é do mesmo módulo e acrescenta alguns atributos relativos ao aluno.

A emissão do contrato de mobilidade necessita de alguns atributos da classe *Student*,

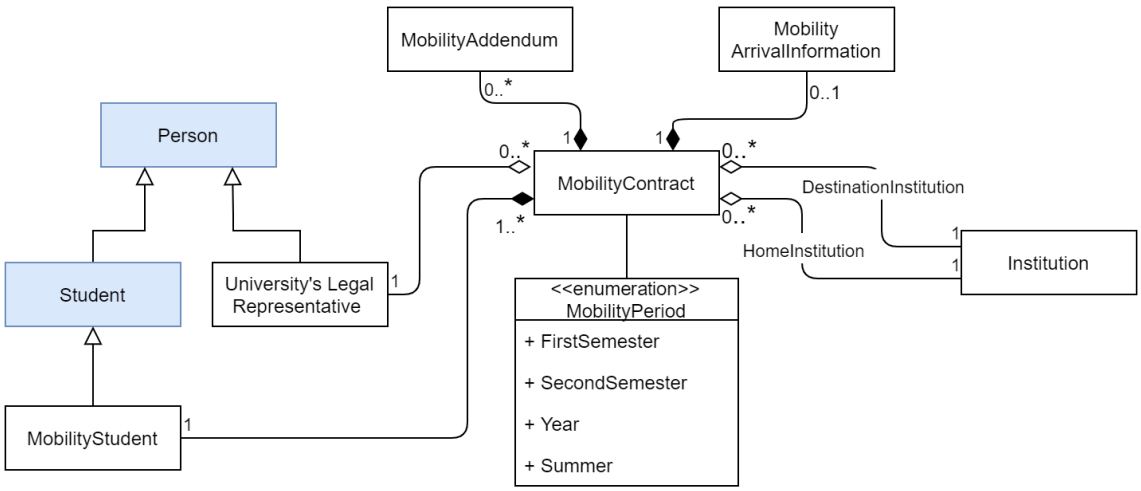

Figura 3.3: Diagrama de classes

mas a informação utilizada no contrato de mobilidade não pode ser alterada depois do contrato ser criado, pois qualquer alteração obrigaria à emissão de um novo contrato. Por este motivo foi criada uma nova classe *MobilityStudent*, que representa o aluno de mobilidade. Esta contém todos os atributos sobre o aluno que são necessários para a emissão do contrato de mobilidade. Desta forma, se a classe *Student* precisar de ser modificada, os atributos da classe *MobilityStudent* mantêm-se.

A classe *UniversityLegalRepresentative* representa o Vice-Reitor que assina o contrato de mobilidade.

A classe *Institution* contém duas relações, pois representa as duas instituições que são mencionadas no contrato de mobilidade, instituição de origem e instituição de destino. Esta classe contém os atributos relativos às instituições que são necessários à emissão do contrato.

A classe *MobilityContract* contém os atributos necessários à construção do Contrato de Mobilidade, como:

- Informação da mobilidade (período de mobilidade, língua de intrução, etc.);
- Informação da área financeira (valores das bolsas);
- Números relativos ao processo (n.º de contrato, n.º de projeto, n.º de compromisso).

Esta é a classe que cria a instância e inicia o *workflow* no sistema Fenix.

A classe *MobilityAddendum* representa as adendas que podem ser criadas e associadas ao contrato quando é preciso fazer alguma modificação ao mesmo. Esta classe é necessária para manter sempre um histórico de todos os atributos. Deste modo as modificações são feitas apenas nas adendas, mas a classe *MobilityContract* continua com os mesmos atributos.

Por fim, a classe *MobilityArrivalInformation* contém a informação relativa à chegada

dos alunos da mobilidade, isto porque muitas vezes os alunos voltam em datas diferentes das do contrato e a bolsa tem de ser recalculada. Mais uma vez, com o objetivo de manter o histórico de todo o processo, foi criada esta classe, que representa a chegada dos alunos, com os atributos finais de mobilidade, como datas, valores de bolsa e aproveitamento de mobilidade.

3.4.2 Processo de negócio

O processo de negócio está desenhado com a linguagem de modelação de *workflows* do sistema Fenix. Para facilitar a sua visualização, o *workflow* foi dividido em cinco imagens que representam os cinco casos de uso principais abordados na secção 3.2.2, sendo que os cenários alternativos não estão representados no *workflow*, de modo a facilitar a legibilidade do mesmo.

Não estão, também, representadas no *workflow* todas as notificações enviadas, uma vez que podem ser enviadas notificações a partir de qualquer estado ou operação para qualquer perfil do processo.

Alguns dos estados do *workflow* podem representar várias tarefas do caso de uso quando são tarefas que estão interligadas, como é o caso do estado "Validação do Orçamento" do excerto de *workflow*, apresentado na figura 3.4, que representa as três primeiras tarefas do UC-01.

Assim que os dados dos alunos são enviados pelas escolas para a RUL, através de uma chamada a um serviço *web*, o sistema cria um utilizador para cada aluno na instância Fenix da RUL e dá início a uma nova instância de *workflow*. O excerto de *workflow* representado na figura 3.4 relativo ao UC-01 é inicializado e o processo passa por um conjunto de estados até chegar à criação do contrato de mobilidade.

Este *workflow* contém o evento "*mobilitySetCompromiss*", que executa a operação "compromissos atribuídos" assim que a tarefa 8 do UC-01 ocorre, sendo despoletada a criação do contrato de mobilidade.

Após a criação e validação do contrato, são realizadas as assinaturas pelo Vice-Reitor e pelo aluno, conforme detalhado no UC-02 e ilustrado na figura 3.5.

As assinaturas devem ser digitais para facilitar o processo, mas também existe a opção de assinatura manual, neste caso, as duas assinaturas devem ser manuais de forma a que não se perca a validação da assinatura digital.

No caso alternativo 2 do UC-02 está descrita uma espera pelo LA que não é visualizada no *workflow*, uma vez que é feita a partir de um validador de separador no estado "Registo na MT+". Este validador não permite que se avance para o estado seguinte se ainda não existir um LA associado ao contrato de mobilidade.

A figura 3.6 representa o excerto de *workflow* relativo ao UC-03, que é o passo seguinte à assinatura do contrato. Aqui é criada uma informação de pagamento que, de seguida, passa por um conjunto de validações, assinaturas e pareceres, até chegar ao pa-

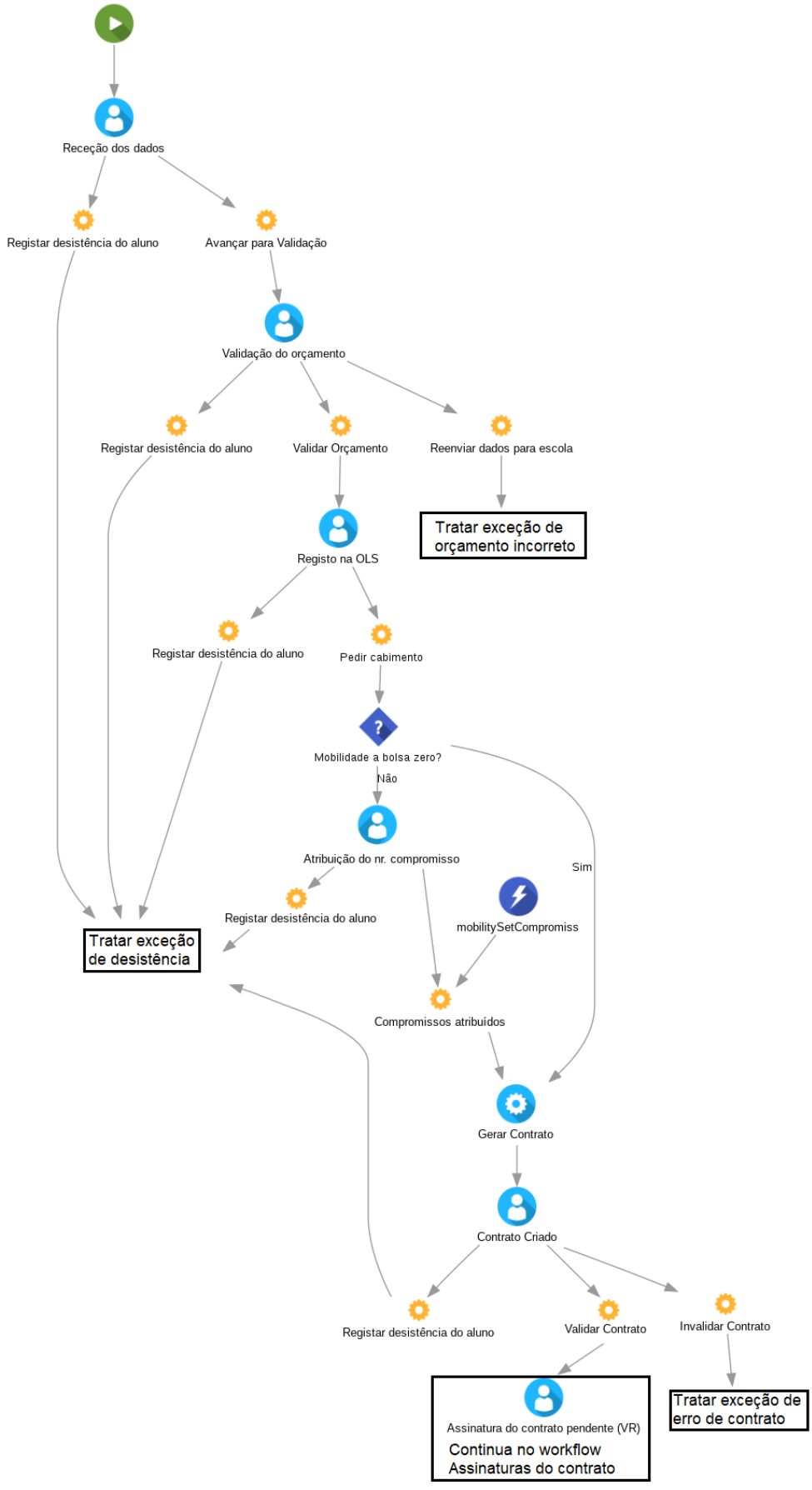

Figura 3.4: Excerto de *workflow* - Criar contrato

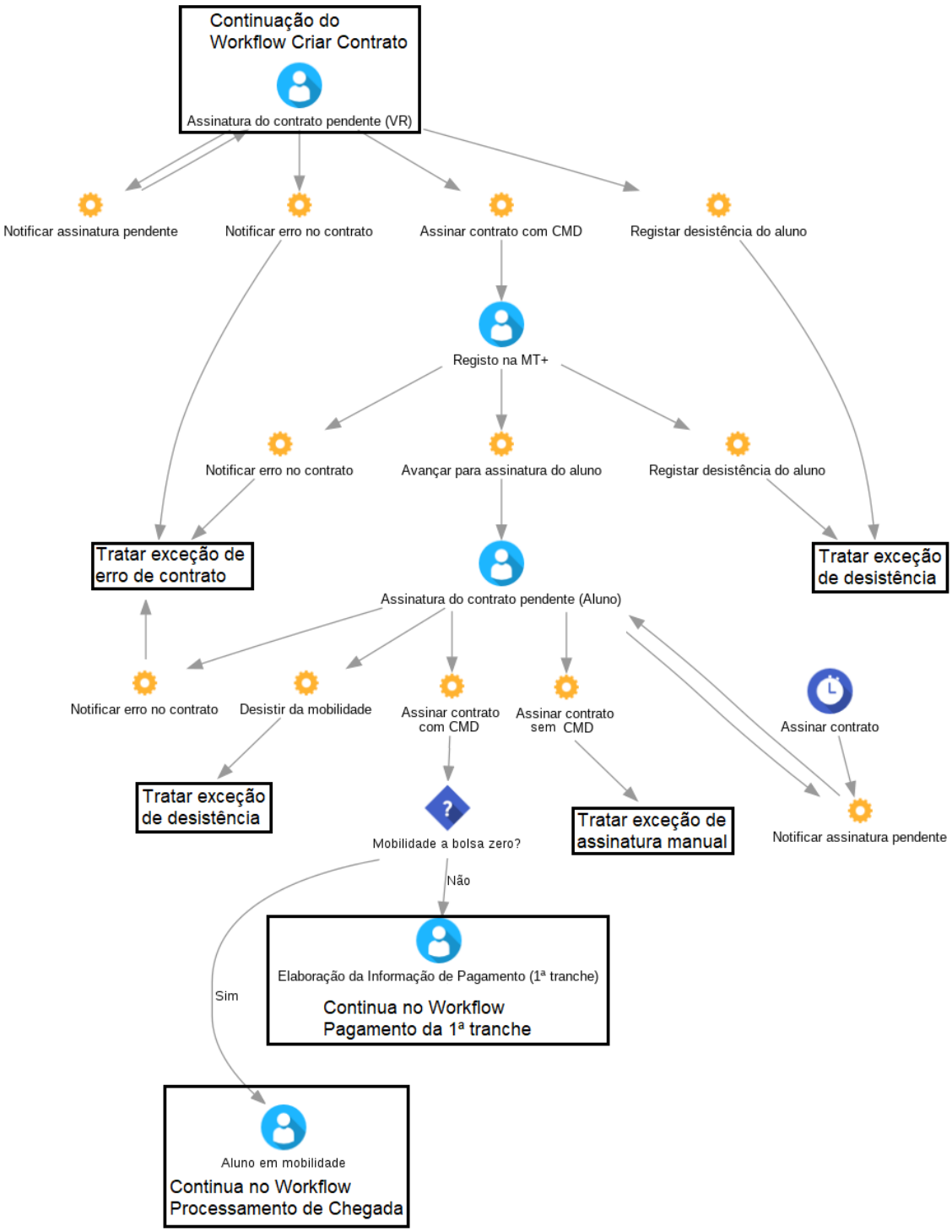

Figura 3.5: Excerto de *workflow* - Assinaturas do contrato

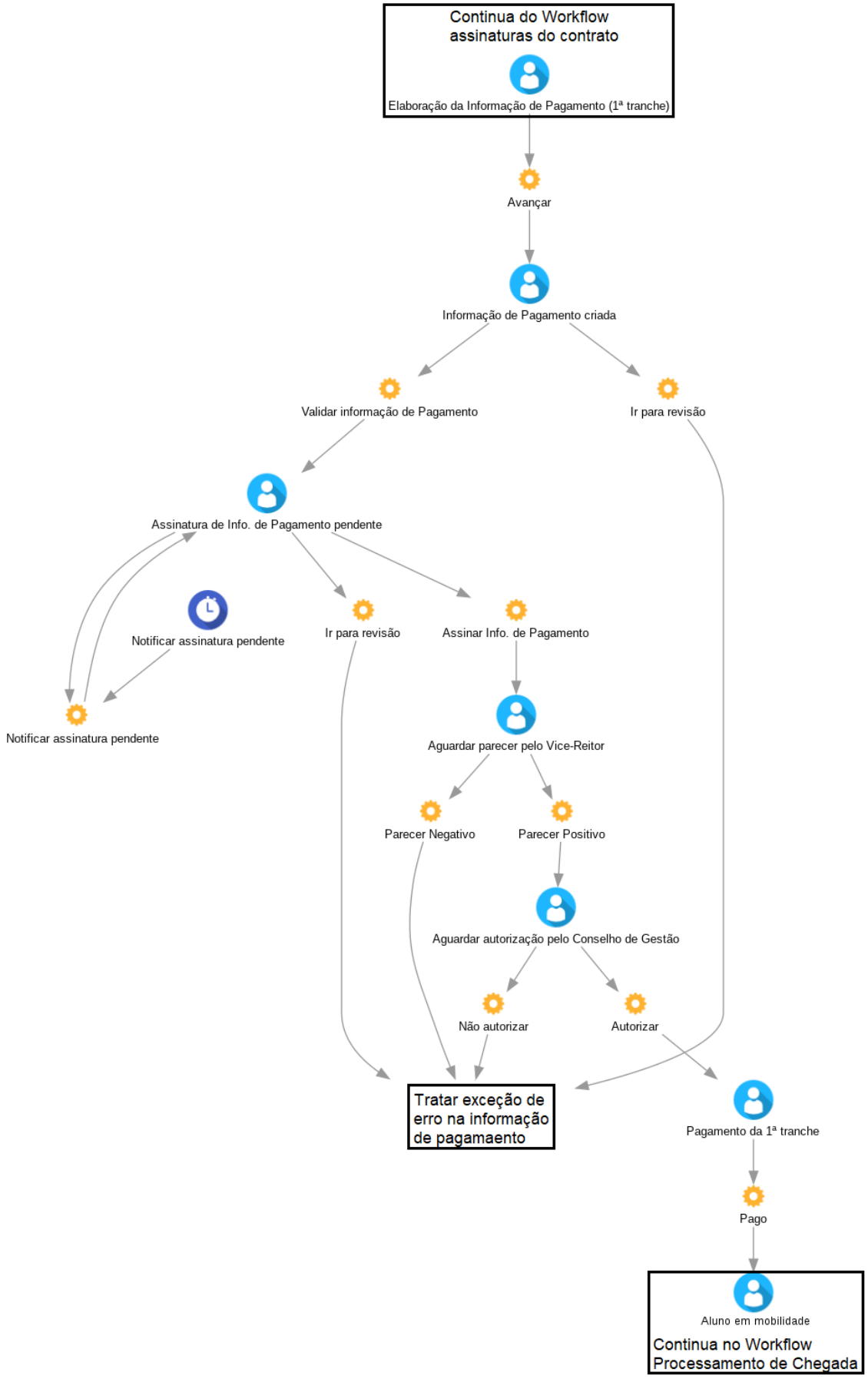

Figura 3.6: Excerto de *workflow* - Pagamento da primeira tranche

gamento da primeira tranche efetuado pela Área financeira.

O processamento de chegada, relativo ao UC-04, está representado na figura 3.7. Este é inicializado a partir do momento em que o aluno envia os documentos obrigatórios de chegada para o Fenix e acaba no pagamento da segunda tranche pela Área financeira, ou no pagamento da devolução pelo aluno.

Na figura 3.8 está representado o excerto de *workflow* relativo ao UC-05, que é muito parecido com o pagamento da primeira tranche, com a pequena diferença de que este finaliza o processo de mobilidade do aluno.

3.5 Sumário

Neste capítulo foram apresentados o levantamento e a análise de requisitos, cujos resultados foram utilizados na definição dos requisitos funcionais e não funcionais. Foi, também, apresentado o desenho da solução. O capítulo seguinte aborda a implementação e avaliação do módulo de mobilidade.

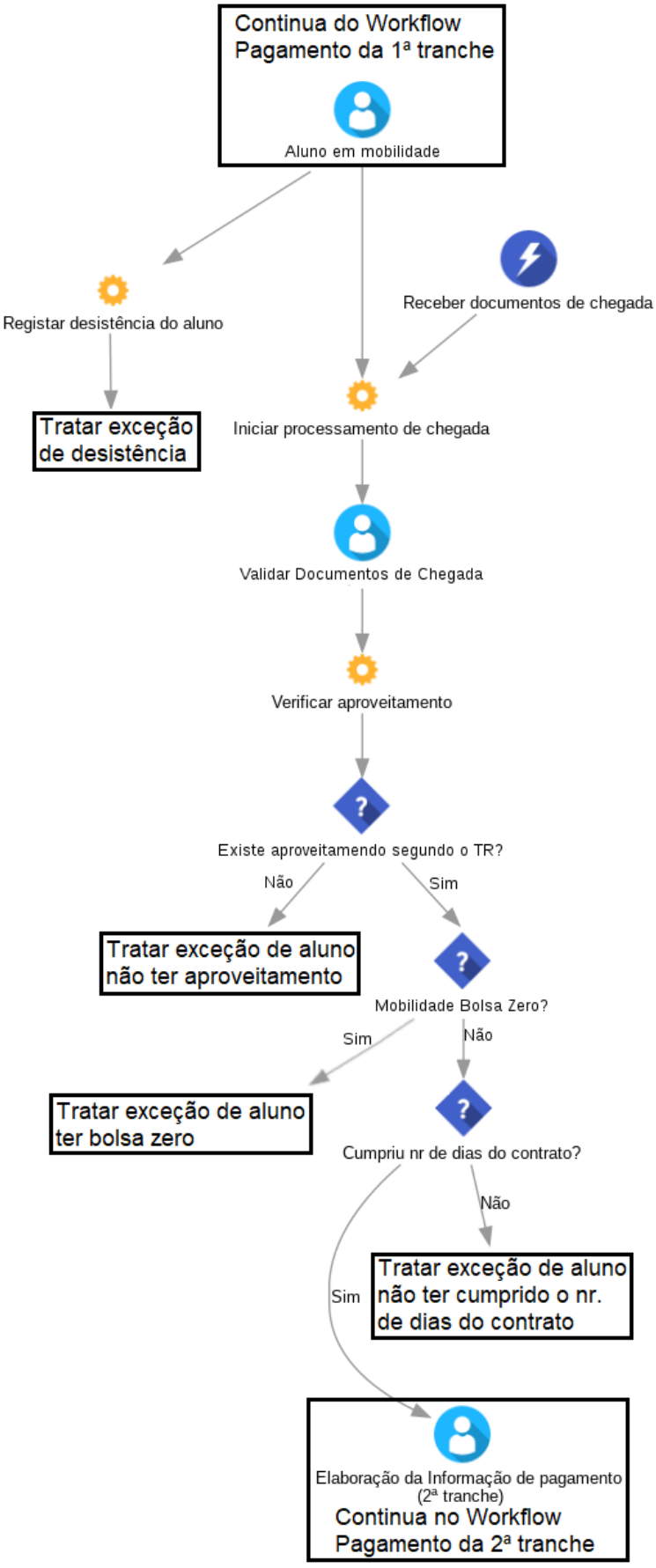

Figura 3.7: Excerto de *workflow* - Processamento de chegada

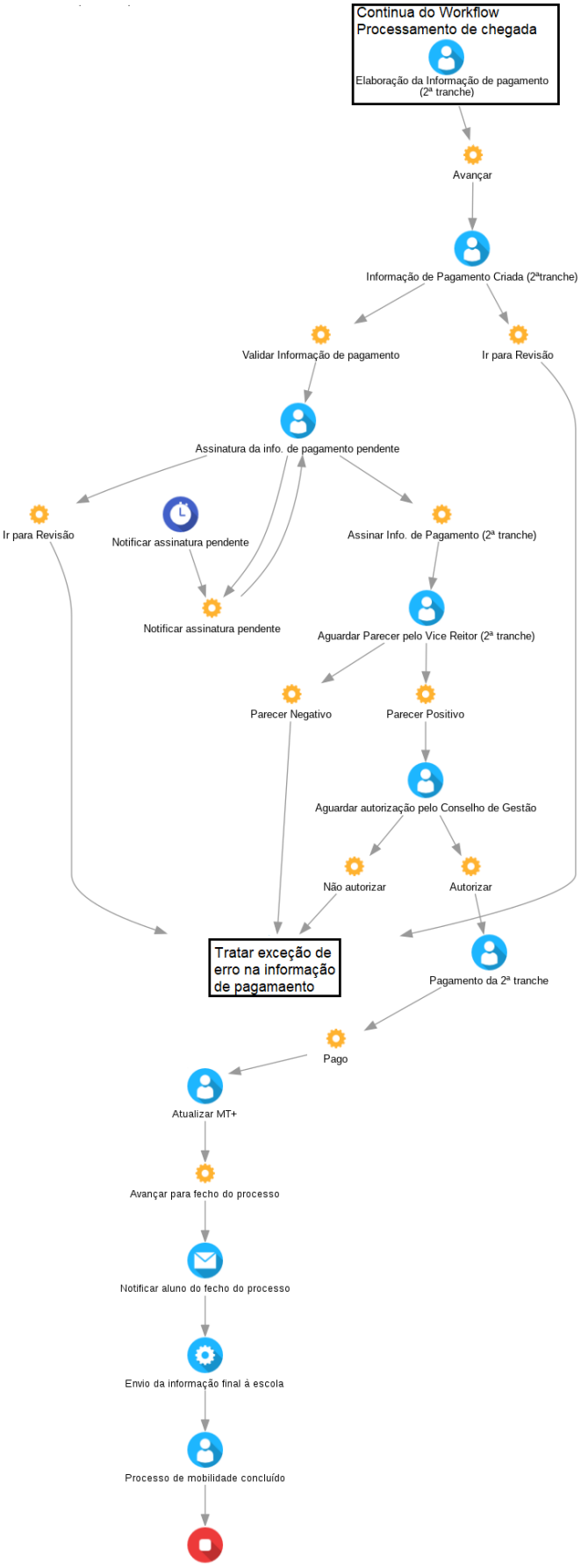

Figura 3.8: Excerto de *workflow* - Pagamento da segunda tranche

Capítulo 4

Implementação e avaliação

Neste capítulo é abordado o processo de implementação do módulo de mobilidade, os passos realizados para a entrada em produção deste módulo na instância Fenix da RUL e a avaliação realizada ao módulo de mobilidade.

4.1 Implementação

A implementação do módulo de mobilidade foi desenvolvida em Java utilizando a plataforma Fenix, a plataforma Omnis, e o Git como sistema de controlo de versões. Este módulo já estava a ser implementado com o objetivo de integrar o projeto *Erasmus Without Paper*, e por isso foi criado um novo *branch* com o intuito de facilitar a implementação do processo ERASMUS relativo aos contratos e bolsas de mobilidade.

Esta secção apresenta a implementação da DSL, da PSL e das extensões de *workflows* do módulo de mobilidade.

4.1.1 DSL

A DSL é uma linguagem específica de domínio onde as entidades, seus atributos e relações entre entidades são definidas.

As entidades e relações entre entidades do módulo de mobilidade foram implementadas de maneira diferente da que está representada no diagrama de classes da figura 3.3 por um conjunto de várias razões: No sistema Fenix, para um utilizador pertencer à classe *Student* é necessário ter um curso associado. No entanto, seria pouco exequível criar todos os cursos da ULisboa na instância Fenix da RUL, por esse motivo os alunos de mobilidade não têm nenhuma relação com a classe *Student* do sistema Fenix.

Ainda contrariando o diagrama de classes apresentado na figura 3.3, as entidades *MobilityStudent* e *Institution* não foram criadas, ficando assim todos os atributos dessas entidades atribuídos à entidade *UlMobContract*, como pode ser visualizado na listagem 4.1. Isto porque os atributos dessas classes são apenas utilizados na criação do contrato de mobilidade e estes não podem ser modificados depois da criação do mesmo.

A entidade *UlMobContract* tem uma relação de um para um com a entidade de domínio *WorkflowInstance*, uma vez que a cada processo de mobilidade tem de estar associada uma instância de *workflow* para ser possível manusear o processo a partir da ferramenta de gestão de *workflows* da Omnis.

```
1  (...)
2  entity mobility.contract.UlMobContract channels (WebJava){
3
4      //Mobility Student
5      String name;
6      String familyNames;
7      String gender;
8      LocalDate dateOfBirth;
9      String nationality;
10     String documentIdNumber;
11     IDDocumentType documentIdType;
12     String fiscalNumber;
13     String address;
14     String areaCode;
15     String areaOfAreaCode;
16     String email;
17     String phoneNumber;
18     Integer studentNumber;
19     Boolean sas;
20     Boolean hasDisabilities;
21     String nib;
22     String bankName;
23     String bankAccountOwner;
24     String degree;
25     String studyAreaCode;
26     String course;
27     Integer completeEnrolledYears;
28     String homeInstitution;
29
30     //MobilityContract
31     UlMobContractMobilityPeriod mobilityPeriod;
32     Integer priorParticipation;
33     String language;
34     String languageCourse;
35     Boolean isConsortium;
36     Boolean isInternship;
37     Boolean combinedMobility;
38     LocalDate startMobilityDate;
39     LocalDate endMobilityDate;
40     Integer mobilityDurationInMonths;
41     Integer restMobilityDurationInDays;
42     Integer totalMobilityDurationInDays;
43     Double monthlySchollarshipValue;
44     Boolean zeroSchollarship;
45     Integer zeroSchollarshipDays;
46     Integer schollarshipDays;
47     Double schoolSchollarshipValue;
48     Integer institutionalContract;
49     String contractNumber;
50     String compromissNumber;
51     Double specialNeedsSchollarship;
52     Double firstPayment;
53     Double secondPayment;
54     LocalDate courseConclusionDate;
55
56     //DestinationInstitution
57     String destinationCountry;
58     String destinationInstitutionErasmusCode;
59     String destinationInstitutionName;
60     Boolean justGraduated;
61     String organizationName;
```

```
62      String organizationType;
63      String organizationAddress;
64      String organizationAreaCode;
65      String organizationArea;
66      String organizationEmail;
67      String organizationPhone;
68
69      //Mobility
70      Boolean virtualMobility;
71      LocalDate startVirtualMobilityDate;
72      LocalDate endVirtualMobilityDate;
73      Boolean isValidated;
74      Boolean desist;
75      Boolean exportedForOLS;
76      Boolean exportedForMT;
77      Integer firstPaymentInfoNumber;
78      Integer secondPaymentInfoNumber;
79      LocalDate firstPaymentInfoDate;
80      LocalDate secondPaymentInfoDate;
81      Integer neePaymentInfoNumber;
82      LocalDate neePaymentInfoDate;
83      String observations;
84      Boolean manualSignature;
85      Boolean multiannualDocument;
86      Boolean learningAgreement;
87      Boolean learningAgreementChanges;
88      Integer addendumIndex;
89
90      manyToOne ExecutionYear executionYear;
91      oneToOne UlMobArrivalInformation arrivalInformation;
92      oneToMany UlMobAddendum contractAddendums mappedBy contract;
93
94      manyToOne UlMobContractType contractType;
95      oneToOne WorkflowInstance workflowInstance;
96      manyToOne DomainRoot domainRoot;
97  }
98  (...)
```

Listagem 4.1: DSL da entidade *UlMobContract*

A entidade *UlMobContractType* tem uma relação de um para muitos com a entidade *UlMobContract*, como pode ser observador na listagem 4.2. Esta entidade contém os atributos de mobilidade que são iguais para todos os contratos, mas que um dia podem ser modificados, como é o caso do número de ECTS necessários para existir aproveitamento de mobilidade. Esta é também a entidade que está diretamente relacionada com as entidades de domínio *Workflow* e *WorkflowRuntimeConfiguration*.

```
1  (...)
2  entity mobility.contract.UlMobContractType channels (WebJava) {
3      DateTime creationDate;
4      String code;
5      UlisboaStudentMobilityLocalizedString name;
6      Integer approvementECTSNumberFor1Semester;
7      Integer approvementECTSNumberFor2Semesters;
8      Integer approvementECTSNumberForLessThan4Months;
9
10     oneToMany UlMobContract contracts mappedBy contractType;
11     manyToOne Workflow workflow;
12     manyToOne DomainRoot domainRoot;
13
14
15     oneToMany WorkflowRuntimeConfiguration workflowRuntimeConfigurations mappedBy
       ulMobContractType;
```

```
16 }
17 (...)
```

Listagem 4.2: DSL da entidade *UlMobContractType*

A entidade *UlMobAddendum* tem uma relação de muitos para um com a entidade *UlMobContract*, como pode ser observado na listagem 4.3, uma vez que um único contrato pode ter várias adendas associadas. Todos os atributos relacionados com o aluno, instituição, ou contrato podem ser modificados com as adendas, no entanto só alguns têm impacto no processamento de chegada. Desta forma apenas os atributos com impacto no processamento de chegada foram criados na entidade adenda, sendo que os outros atributos que possam sofrer alterações ficam registados no atributo *observations*.

```
1  (...)
2  entity mobility.contract.UlMobAddendum channels (WebJava){
3      DateTime creationDate;
4      LocalDate startMobilityDate;
5      LocalDate endMobilityDate;
6      Integer totalMobilityDays;
7      Integer schollarshipDays;
8      Integer zeroSchollarshipDays;
9      Double schollarshipValue;
10     String destinationInstitution;
11     UlMobContractMobilityPeriod mobilityPeriod;
12     Double remainderFirstPayment;
13     Double firstPayment;
14     Double secondPayment;
15     Boolean changeSchollarship;
16     String iban;
17     String observations;
18     Integer paymentInfoNumber;
19     LocalDate paymentInfoDate;
20
21     manyToOne UlMobContract contract;
22     manyToOne DomainRoot domainRoot;
23 }
24 (...)
```

Listagem 4.3: DSL da entidade *UlMobAddendum*

A entidade *UlMobArrivalInformation* tem uma relação de um para um com a entidade *UlMobContract* porque um contrato apenas pode ter uma informação de chegada associada. Os atributos desta entidade são os necessários ao processamento da chegada do aluno de mobilidade, como se pode observar na listagem 4.4.

Depois de criadas as entidades e relações na DSL, o código é compilado e a DML é criada, assim como as classes java das entidades.

```
1  (...)
2  entity mobility.contract.UlMobArrivalInformation channels (WebJava){
3      LocalDate creationDate;
4      LocalDate startMobilityDate;
5      LocalDate endMobilityDate;
6      Integer recognizedECTSsFromTR;
7      Boolean finalRecognition;
8      Boolean recognizedApprovementWithoutECTSs;
9      Boolean recognitionWithECTSs;
```

```
10      Integer recognizedECTSsFromSchool;
11      Double finalSchollarshipValue;
12      Integer totalMobilityDurationInDays;
13      Integer finalSchollarshipDays;
14      Double secondPayment;
15      Double devolutionValue;
16      Boolean majorForce;
17
18      oneToOne UlMobContract ulMobContract mappedBy arrivalInformation;
19      manyToOne DomainRoot domainRoot;
20
21  }
22  (...)
```

Listagem 4.4: DSL da entidade *UlMobArrivalInformation*

4.1.2 PSL

Depois de criadas as entidades e relações na DSL, foi implementada a PSL do módulo de mobilidade, onde são representados os ecrãs e os fluxos entre ecrãs. A listagem 4.5 representa o fluxo dos ecrãs iniciais do módulo de mobilidade.

```
1  (...)
2  flow mobility.contract channel (WebJava)  {
3      searchScreen searchContracts [UlMobContract]
4      (fields  executionYear contractNumber homeInstitution name familyNames email)
5      (searchFields executionYear name documentIdNumber) {
6          dialog selectMultipleEvent executeOperation ->
       executeUlMobContractWorkflowOperation
7          selectMultipleEvent exportContractDataForMT
8          selectMultipleEvent exportContractDataForOLS
9          selectMultipleEvent exportMobilityContracts
10         selectMultipleEvent exportContractDataForCompromiss
11         selectMultipleEvent exportContractDataForCompromissReinforcement
12         selectMultipleEvent exportContractData
13         selectMultipleEvent exportFinalContractData
14         selectMultipleEvent exportArrivalInformationDocuments
15         dialog event uploadCompromiss -> uploadCompromiss
16         dialog event uploadSignedMobilityContracts -> uploadSignedMobilityContracts
17         dialog event uploadMultiannualDocuments -> uploadMultiannualDocuments
18         dialog event uploadLearningAgreements -> uploadLearningAgreements
19         dialog event uploadTranscriptOfRecords -> uploadTranscriptOfRecords
20         viewAction //-> ManageContractUI
21         deleteAction -> searchContracts
22     }
23
24     screen executeUlMobContractWorkflowOperation {
25     }
26
27     screen uploadCompromiss {
28     }
29
30     screen uploadSignedMobilityContracts {
31     }
32
33     screen uploadMultiannualDocuments {
34     }
35
36     screen uploadLearningAgreements {
37     }
38
39     screen uploadTranscriptOfRecords {
40     }
```

```
41
42      searchScreen searchStudentContracts [UlMobContract] (fields name familyNames
        homeInstitution contractNumber language){
43          viewAction // -> ManageContractUI
44      }
45
46      searchScreen searchContractsForImport [UlMobContract] (fields  executionYear
        contractNumber homeInstitution name
47          familyNames startMobilityDate endMobilityDate email)
48          (searchFields executionYear institutionalContract name documentIdNumber) {
49          dialog event uploadSMSContractTables -> uploadSMSContractTables
50          dialog event uploadSMPContractTables -> uploadSMPContractTables
51          viewAction // -> ManageContractUI
52      }
53
54      screen uploadSMSContractTables {
55      }
56
57      screen uploadSMPContractTables {
58      }
59  }
60  (...)
```

Listagem 4.5: Excerto de PSL que representa o fluxo dos ecrãs iniciais do módulo de mobilidade

O ecrã *searchContracts* é o ecrã inicial utilizado pelo NM. Neste ecrã é possível procurar por processos de mobilidade específicos a partir de alguns atributos do processo (linha 5 da listagem 4.5), visualizar alguns atributos desses processos (linha 4 da listagem 4.5), entrar dentro do detalhe de cada um dos processos (linha 20 da listagem 4.5), executar operações em massa para vários processos previamente selecionados (linha 6 da listagem 4.5), importar documentos (linhas 15 a 19 da listagem 4.5), exportar documentos e tabelas com diferentes atributos de processos previamente selecionados (linhas 7 a 14 da listagem 4.5) e apagar processos de mobilidade que possam ter sido mal importados para o sistema (linha 21 da listagem 4.5).

O ecrã *searchStudentContracts* é o ecrã inicial visível apenas para os alunos de mobilidade, onde estes conseguem visualizar alguns atributos do seu próprio processo de mobilidade, como o nome da instituição de origem e o número do contrato, e o estado em que este se encontra. A partir deste ecrã, os alunos conseguem apenas entrar dentro do detalhe do processo de mobilidade.

O ecrã *searchContractsForImport* é utilizado para fazer o import das tabelas com os dados dos alunos, que neste momento são enviadas pelas escolas para a RUL, sendo então um ecrã provisório até que haja comunicação entre as várias instâncias Fenix.

Os ecrãs que se podem visualizar nas linhas 24 a 40 da listagem 4.5, são os ecrãs para onde o sistema vai direcionar quando um evento é selecionado, por exemplo, o evento *uploadCompromiss* direciona para o ecrã *uploadCompromiss*. Este ecrã vai aparecer à frente do ecrã inicial, como se pode visualizar na figura 4.1, porque tem a indicação *dialog* na PSL.

Depois de implementada a PSL, o código é compilado e as classes de apresentação são geradas. A partir dessas classes é possível alterar o que foi criado pela PSL. Podem-se

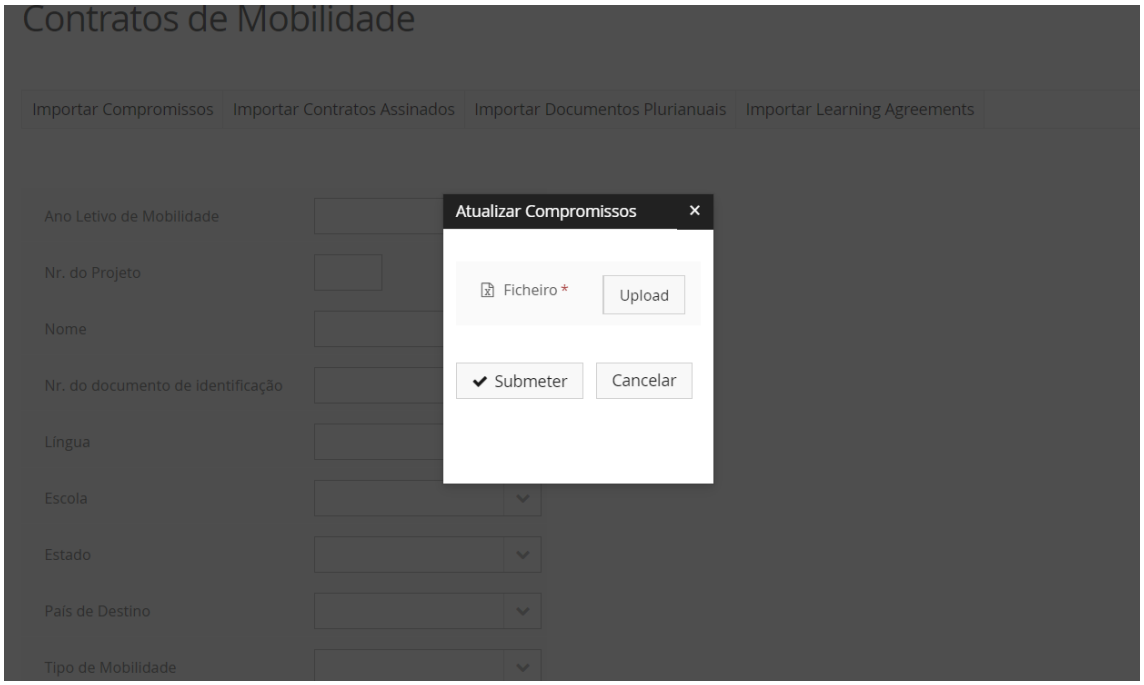

Figura 4.1: Ecrã de exemplo de um *dialog*

acrescentar atributos de pesquisa, como apresentado na listagem 4.6, e de visualização, listagem 4.7, ou implementar o comportamento de cada evento como apresentado na listagem 4.8. Nas figuras 4.2 e 4.3 estão representados excertos do ecrã inicial *SearchContracts* do módulo de mobilidade depois de modificada a classe *SearchContracts*.

```
(...)
@Override
protected Schema getSearchSchema() {
    final Schema schema = super.getSearchSchema();
(...)
    schema.getSchemaSlot(UlMobContract.META().EXECUTION_YEAR()).required()
        .configureInput((Configurator) inputSelectOneConfigurator);
    schema.getSchemaSlot(UlMobContract.META().NAME())
        .configureInput((Configurator) inputTextAreaConfigurator);
    schema.getSchemaSlot(UlMobContract.META().DOCUMENT_ID_NUMBER())
        .configureInput((Configurator) inputTextAreaConfigurator);

    schema.addCustom(i18n("Institutional Contract"),PROPERTY_INSTITUTIONAL_CONTRACT,
        Collection.class).inputComponent(SelectManyComponent.class)
        .addOptionsBind(UlMobContract.META().EXECUTION_YEAR(),
            (OptionsProvider<ExecutionYear, Integer>) ey -> {
                return UlMobContract.findByExecutionYear(ey).stream()
                    .map(co -> co.getInstitutionalContract())
                    .filter(Objects::nonNull).collect(Collectors.toSet());
                }).configureInput((Configurator) inputSelectManyConfigurator)
        .after(UlMobContract.META().DOCUMENT_ID_NUMBER());

    schema.addCustom(i18n("Language"), PROPERTY_LANGUAGE, Collection.class)
        .inputComponent(SelectManyComponent.class)
        .addOptionsBind(UlMobContract.META().EXECUTION_YEAR(),
            (OptionsProvider<ExecutionYear, String>) ey -> {
                return UlMobContract.findByExecutionYear(ey).stream().map(co -> co
                    .getLanguage()).filter(Objects::nonNull)
```

```
                    .collect(Collectors.toSet());
            }).configureInput((Configurator) inputSelectManyConfigurator);

(...)
}
(...)
```

Listagem 4.6: Excerto do método *getSearchSchema()* da classe *SearchContracts*

```
(...)
@Override
protected Schema getDisplaySchema() {
    final Schema schema = super.getDisplaySchema();

    schema.add(i18n("State"), UlMobContract.META().WORKFLOW_INSTANCE()
        .ACTIVE_STATE().STATE().LOCALIZED_NAME())
        .before(UlMobContract.META().EXECUTION_YEAR());
    schema.getSchemaSlot(UlMobContract.META().EXECUTION_YEAR()).visible(false);
    schema.getSchemaSlot(UlMobContract.META().CONTRACT_NUMBER())
        .configureOutput(c -> c.setStyles(Style.WIDTH_MIN.SMALL));
    schema.getSchemaSlot(UlMobContract.META().FAMILY_NAMES())
        .configureOutput(c -> c.setStyles(Style.WIDTH_MIN.SMALL));

    schema.addCustom(i18n("Mobility Period"), "mobilityPeriodSlot", String.class)
        .formatter((UlMobContract c) -> (c.getLatestMobilityPeriodAddendum() == null
            ? c.getMobilityPeriod().getPeriod() : c.getLatestMobilityPeriodAddendum()
                .getMobilityPeriod().getPeriod()))
        .after(UlMobContract.META().FAMILY_NAMES()).visible(false);

    schema.addCustom(i18n("StartMobilityDate"), "startMobilityDateSlot",
        String.class).formatter((UlMobContract c) -> (c.getLatestDateAddendum() == null
            ? formatDatesForSchema(c.getStartMobilityDate()) : formatDatesForSchema(c
                .getLatestDateAddendum().getStartMobilityDate())))
        .after("mobilityPeriodSlot").visible(false);

(...)
        return schema;
}
(...)
```

Listagem 4.7: Excerto do método *getDisplaySchema()* da classe *SearchContracts*

```
(...)
@Override
protected void processExportContractDataForOLSEvent(List<UlMobContract> selection) {
    executeInTransactionalContext(() -> {
        selection.forEach(e -> e.setExportedForOLS(true));
        exportWorkflowData(selection, false, ExportSchemas.getDisplaySchemaForOLS(),
            PROPERTY_OLS));
    });
}

(...)

private void exportWorkflowData(List<UlMobContract> selection, boolean withDocuments,
    Schema schema, String type) {
    final List<WorkflowData> datas = selection.stream().map(i -> new WorkflowData(i.
        getWorkflowInstance(), i.getWorkflowInstance().getNumber(), i.getName()))
        .collect(Collectors.toList());

    if (datas.isEmpty()) {
        return;
    }

    WorkflowDataExporter dataExporter = null;

    if (type.equals( PROPERTY_OLS)) {
        dataExporter = new WorkflowDataExporter(datas,  PROPERTY_OLS + new
```

```
26              DateTime().toString("yyyy-MM-dd_HH-mm"), resultGrid, schema);
27          dataExporter.export();
28      }
29  (...)
30      final FileDescriptor descriptor = withDocuments ? dataExporter.getResult() :
31          dataExporter.getResultWithoutDocuments();
32  (...)
33      download(descriptor);
34  }
35  (...)
```

Listagem 4.8: Implementação do evento *exportContractDataForOLS* de *export* da classe *SearchContracts*

Figura 4.2: Excerto do formulário de pesquisa do ecrã *SearchContracts*

Figura 4.3: Grelha de processos de mobilidade do ecrã *SearchContracts*

De seguida foram criados os restantes fluxos e ecrãs utilizados no módulo de mobilidade, sendo que são ecrãs visualizados apenas dentro do detalhe de cada processo. Estes ecrãs são inicializados de modo semelhante à apresentada anteriormente, com a diferença de que poderão ser ecrãs de criação, de leitura ou de atualização, como se pode ver na listagem 4.9.

```
(...)
flow mobility.contract.workflow channel(WebJava) {
    readScreen readContractInsideWorkflow [UlMobContract] (fields documentIdNumber
    documentIdType fiscalNumber gender nationality studentNumber dateOfBirth address
    areaCode areaOfAreaCode nib bankAccountOwner bankName email phoneNumber degree
    studyAreaCode course destinationInstitutionErasmusCode destinationInstitutionName
    priorParticipation completeEnrolledYears language languageCourse virtualMobility
    startMobilityDate endMobilityDate startVirtualMobilityDate endVirtualMobilityDate
    zeroSchollarship zeroSchollarshipDays totalMobilityDurationInDays schollarshipDays
    monthlySchollarshipValue compromissNumber schoolSchollarshipValue firstPayment
    secondPayment exportedForOLS exportedForMT isConsortium) {
        backEvent hidden
        updateEvent hidden
    }

        updateEvent
    }

    updateScreen updateObservationsInsideWorkflow [UlMobContract] (fields observations
    manualSignature learningAgreement multiannualDocument){
        updateEvent
    }

    readScreen updatePaymentInfoInsideWorkflow [UlMobContract] (fields
    firstPaymentInfoNumber firstPaymentInfoDate secondPaymentInfoNumber
    secondPaymentInfoDate){
        backEvent hidden
        updateEvent
    }
}

(...)
```

Listagem 4.9: Excerto de PSL que representa o fluxo de ecrãs dentro do *workflow*

4.1.3 Documentos e assinaturas digitais

Para ser possível gerar contratos a partir da plataforma *Fenix* foi utilizado o módulo de documentos digitais já existente no sistema. Desta forma foi preciso criar um *template* do documento que será gerado, que necessita de uma fonte de dados. Foi necessário criar uma classe *wrapper -UlMobContractForReport-* para a classe *UlMobContract* com todos os atributos utilizados nos contratos de mobilidade, como mostrado na listagem 4.10.

```
(...)
@DynamicMetaHolder
public class UlMobContractForReport {

    private UlMobContract contract;

    public UlMobContractForReport(UlMobContract contract) {
        this.contract = contract;
```

```
9       }
10
11      public void setContract(UlMobContract contract) {
12          this.contract = contract;
13      }
14
15      public String getContractId() {
16          return contract.getExternalId();
17      }
18
19      public UlMobContract getContract() {
20          return contract;
21      }
22
23      public ExecutionYear getExecutionInterval() {
24          return contract.getExecutionYear();
25      }
26
27      public String getContractNumber() {
28          return contract.getContractNumber();
29      }
30
31      public String getName() {
32          return contract.getName();
33      }
34
35  (...)
36  }
```

Listagem 4.10: Excerto da classe *UlMobContractForReport*

De seguida foi criada uma classe de fonte de dados – *UlMobContractForReportInputStrategy* – que devolve uma coleção de *UlMobContract*, como se verifica na listagem 4.11. Por fim foi criada uma classe *provider* – *WorkflowUlMobContractFieldProvider* – que injecta o *UlMobContract* associado à instância de *workflow* no parâmetro da fonte de dados do relatório, como apresentado na listagem 4.12.

```
1   (...)
2   @ReportInputBinding(UlMobContractForReport.class)
3   public class UlMobContractForReportInputStrategy implements ReportInputStrategy<
        UlMobContractForReport> {
4       private Collection<UlMobContract> ulMobContracts;
5
6       @Override
7       public Collection<UlMobContractForReport> provide() {
8           if (ulMobContracts == null) {
9               return Collections.emptyList();
10          }
11          return ulMobContracts.stream().map(c -> new UlMobContractForReport(c))
12              .collect(Collectors.toList());
13      }
14
15      public Collection<UlMobContract> getUlMobContracts() {
16          return ulMobContracts;
17      }
18
19      public void setUlMobContracts(Collection<UlMobContract> contracts) {
20          this.ulMobContracts = contracts;
21      }
22
23      @Override
24      public void configureInputInterface(Schema schema) {
25          Set<UlMobContract> contracts = UlMobContract.findAll().stream().limit(10)
26              .collect(Collectors.toSet());
```

```
27          schema.getSchemaSlot("ulMobContracts")
28              .caption(I18N.i18n(ArtifactHandler.BUNDLE, "Contracts"))
29              .addOptions(contracts);
30      }
31  }
```

Listagem 4.11: Classe *UlMobContractForReportInputStrategy*

```
1  (...)
2  public class WorkflowUlMobContractFieldProvider implements WorkflowDynamicFieldProvider<
       UlMobContract> {
3
4      @Override
5      public Class<UlMobContract> getType() {
6          return UlMobContract.class;
7      }
8
9      @Override
10     public List<Class<?>> getParametersType() {
11         return Collections.emptyList();
12     }
13
14     @Override
15     public UlMobContract provide(WorkflowInstance instance) {
16         if (instance == null) {
17             Log.error(LogContextStandards.OMNIS_PLATFORM,
18                     "Workflow instance not defined when trying to get dynamic field"
19                     + "value.");
20             return null;
21         }
22         return instance.getUlMobContract();
23     }
24
25     @Override
26     public String getPresentationName() {
27         return I18N.i18n(com.qubit.solution.fenixedu.module
28             .fenixeduUlisboaStudentMobility.ArtifactHandler.BUNDLE,
29                 "label.WorkflowDynamicFieldProvider"
30                 + ".WorkflowUlMobContractFieldProvider");
31     }
32 }
```

Listagem 4.12: Classe *WorkflowUlMobContractFieldProvider*

Para criar a configuração dos relatórios no sistema, primeiro teve de ser construído um relatório para a entidade *UlMobContractForReport*, onde foram acrescentados os atributos da classe, posteriormente foi definida uma fonte de dados para a classe *wrapper* anteriormente implementada e finalmente foi criada a configuração de relatório para os *templates* com todos os atributos necessários.

O passo seguinte foi criar um tipo de *template*, associá-lo à configuração criada, e definir o *template* do contrato de mobilidade como ilustrado na figura 4.4.

Para a geração do contrato estar associada aos processos de mobilidade, o modelo de processo foi reconfigurado. Foram criados quatro novos tipos de documento: um que representa o *template* criado – contrato de mobilidade –, um que representa o pdf gerado – contrato gerado –, um que representa o contrato assinado pelo Vice-Reitor – contrato assinado pelo Vice-Reitor – e um último que representa o contrato final, assinado pelas duas partes (Vice-Reitor e aluno) – contrato assinado pelo aluno. Na figura 4.5 é possível visualizar o ecrã com os tipos de documento deste processo.

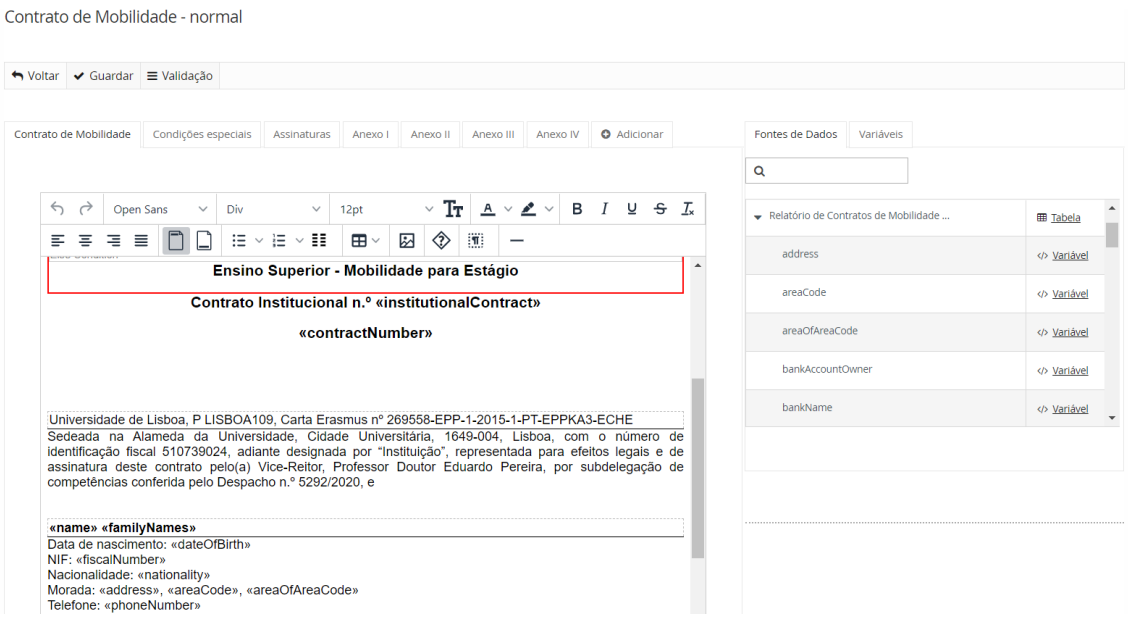

Figura 4.4: Excerto do *template* do contrato mobilidade

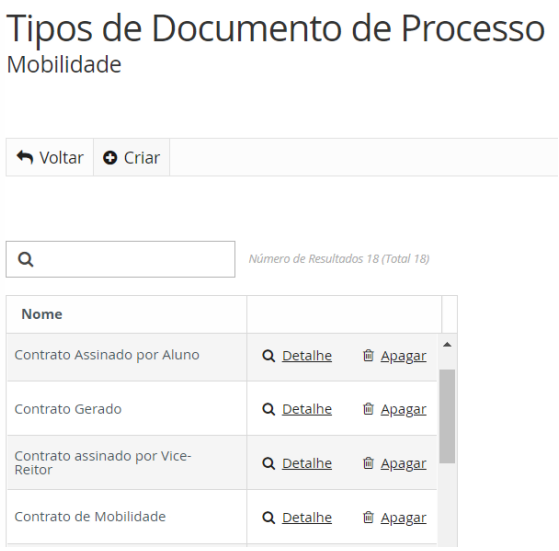

Figura 4.5: Contratos de mobilidade – tipos de documento de processo

De seguida foram criados quatro documentos para estes tipos de documento no modelo de processo, foi adicionado o *template* criado ao documento "contrato de mobilidade" como se consegue visualizar na figura 4.6, foi criada a operação "gerar contrato" do tipo "Gerar Documento Online" como mostrado na figura 4.7 e, finalmente, foram adicionados os documentos "contrato de mobilidade" e "contrato gerado" como parâmetros da operação como ilustrado na figura 4.8.

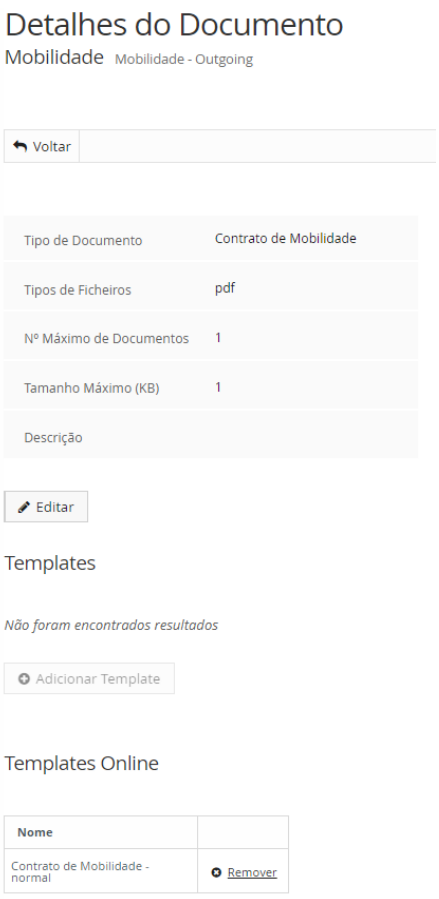

Figura 4.6: Configuração do documento contrato de mobilidade

Figura 4.7: Configuração da operação gerar contrato

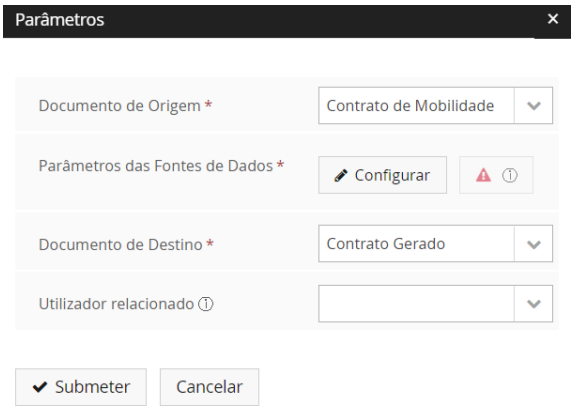

Figura 4.8: Configuração dos parâmetros da operação gerar contrato

Por último, foi criada a operação de assinatura digital, do tipo ”Assinar Documento Digital” ao modelo de processo, como se pode ver na figura 4.9, e acrescentados os documentos ”contrato assinado pelo Vice-Reitor” e ”contrato assinado pelo aluno” como parâmetros da operação, como é ilustrado na figura 4.10.

4.1.4 *Workflows*

Para utilizar a ferramenta de modelação de processos do Fenix para o módulo de mobilidade foi necessário criar um novo tipo de processo e acrescentar a componente base do Fenix e a componente ULISBOA_STUDENT_MOBILITY, criada propositadamente para o módulo de mobilidade, como se verifica na figura 4.11. De seguida foi criado um novo modelo de processo para o tipo de processo criado anteriormente e modelado o processo.

Paralelamente à modelação do processo foram criados perfis de acesso e implementadas operações, separadores customizados e validadores de separadores. Para que estas extensões do *workflow* de mobilidade sejam reconhecidas como tal foi necessário acrescentar, a cada uma das classes de extensão, a componente criada anteriormente da seguinte forma ”@WorkflowComponent(type = ’ULISBOA_STUDENT_MOBILITY’)”.

Perfis de acesso

De forma a que os utilizadores tenham permissões e acessos diferentes consoante o seu papel no processo de *workflow* é necessário criar perfis de acesso e associar os utilizadores aos perfis. A criação dos perfis de acesso é feita na plataforma Fenix, no menu de *worklfows* e, durante a sua criação, é necessário associar um tipo de processo e uma estratégia de validação.

Nos casos em que os processos são visualizados por todo o grupo e todo o grupo pode executar operações em qualquer processo, como é o caso do Vice-Reitor e do NM, a es-

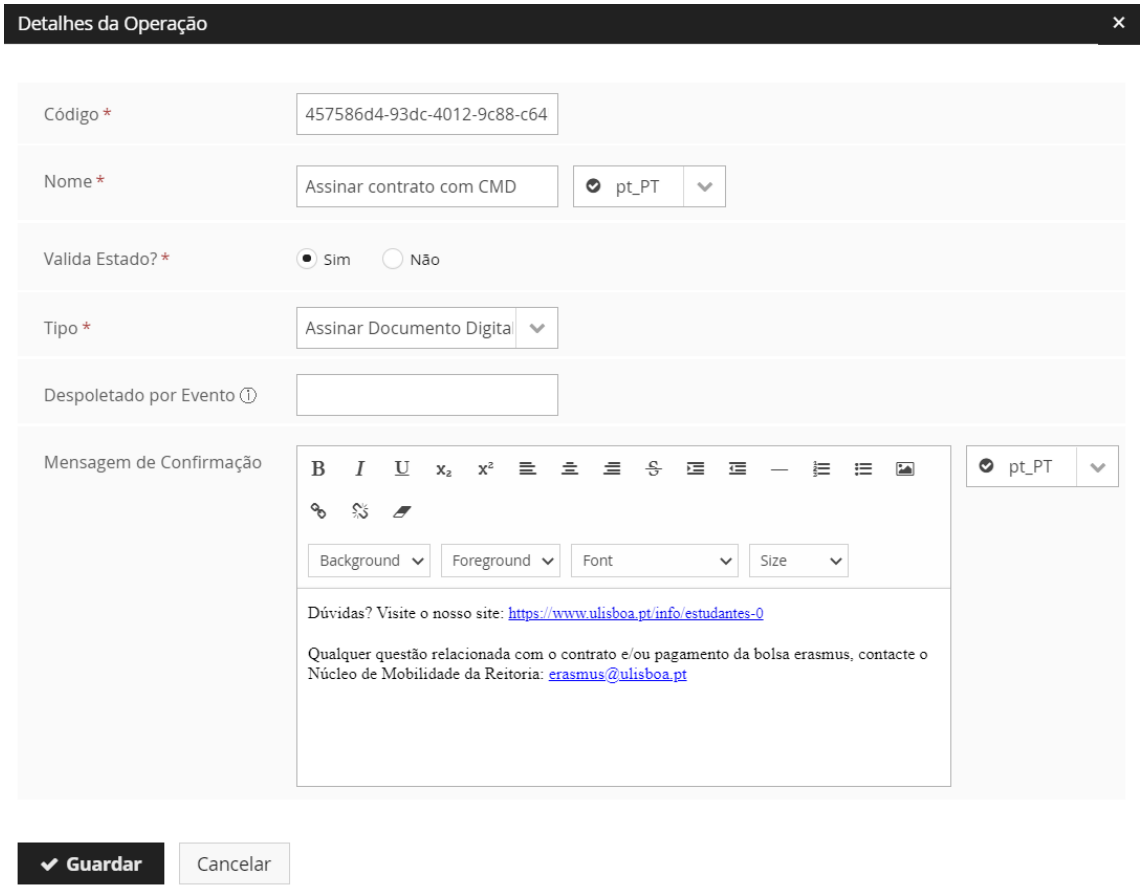

Figura 4.9: Configuração da operação assinatura digital

Figura 4.10: Configuração dos parâmetros da operação assinatura digital

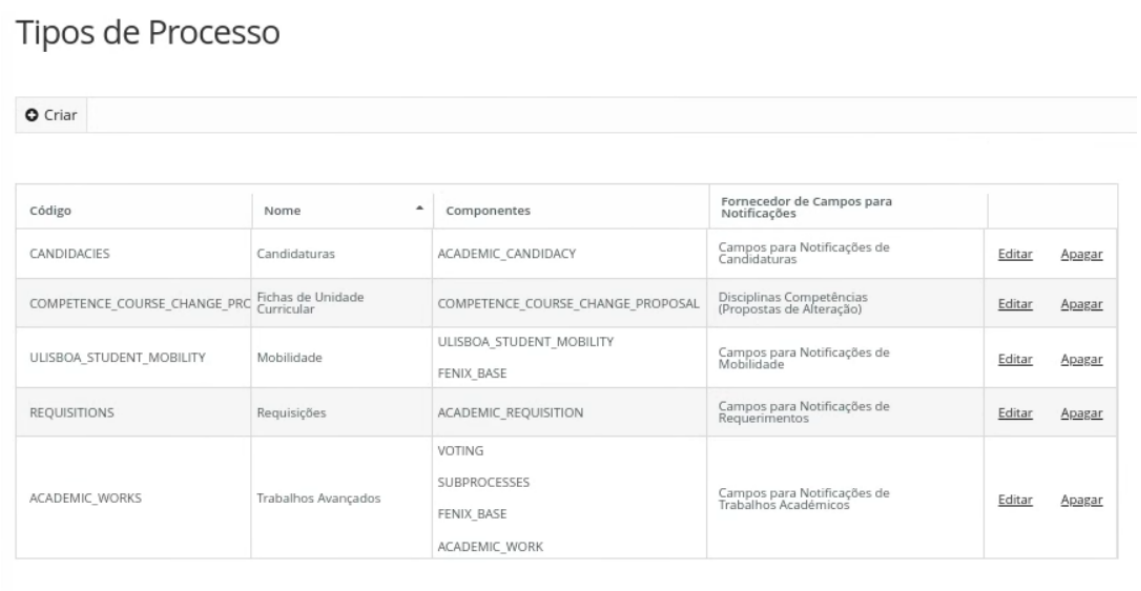

Figura 4.11: Ecrã de tipos de processo do Fenix

tratégia de validação a utilizar é ”grupo dinâmico”. Nestes casos é depois acrescentado como parâmetro o grupo de acesso anteriormente criado no Fenix com os respetivos utilizadores.

Nos casos em que o utilizador só pode visualizar e interagir com o próprio processo, como é o caso do aluno de mobilidade, é necessário implementar a estratégia de validação a ser utilizada: é criada uma nova classe que estende a classe *FenixWorkflowAccessControlGroupValidator*; são reescritos os métodos *isMember()* e *getMembers()*, como se verifica na listagem 4.13; e por último é associada esta estratégia de validação ao perfil de acesso. A figura 4.12 mostra o ecrã dos perfis de acesso dos modelos de processo com os perfis de acesso criados para os processos de mobilidade.

```
1  (...)
2  @WorkflowComponent(type = "ULISBOA_STUDENT_MOBILITY")
3  public class MobilityGroupValidator extends FenixWorkflowAccessControlGroupValidator {
4
5      @Override
6      public boolean isMember(final WorkflowInstance workflowInstance, final Person person
       , final IHasWorkflowParameters target) {
7          return Objects.equals(workflowInstance.getUlMobContract().getDocumentIdNumber(),
8              person.getDocumentIdNumber()) && workflowInstance.getUlMobContract().
9                  getDocumentIdType() == person.getIdDocumentType();
10     }
11
12     @Override
13     public Collection<Person> getMembers(final WorkflowInstance workflowInstance, final
       IHasWorkflowParameters target) {
14         Person person = Person.readByDocumentIdNumberAndIdDocumentType(workflowInstance
15             .getUlMobContract().getDocumentIdNumber(), workflowInstance
16                 .getUlMobContract().getDocumentIdType());
17         return person != null ? Collections.singleton(person) : Collections.emptySet();
18     }
19
```

```
20      public static String getPresentationName() {
21          return I18N.i18n(ArtifactHandler.BUNDLE, "Mobility Group Validator - Student");
22      }
23  }
```

Listagem 4.13: Classe *MobilityGroupValidator*

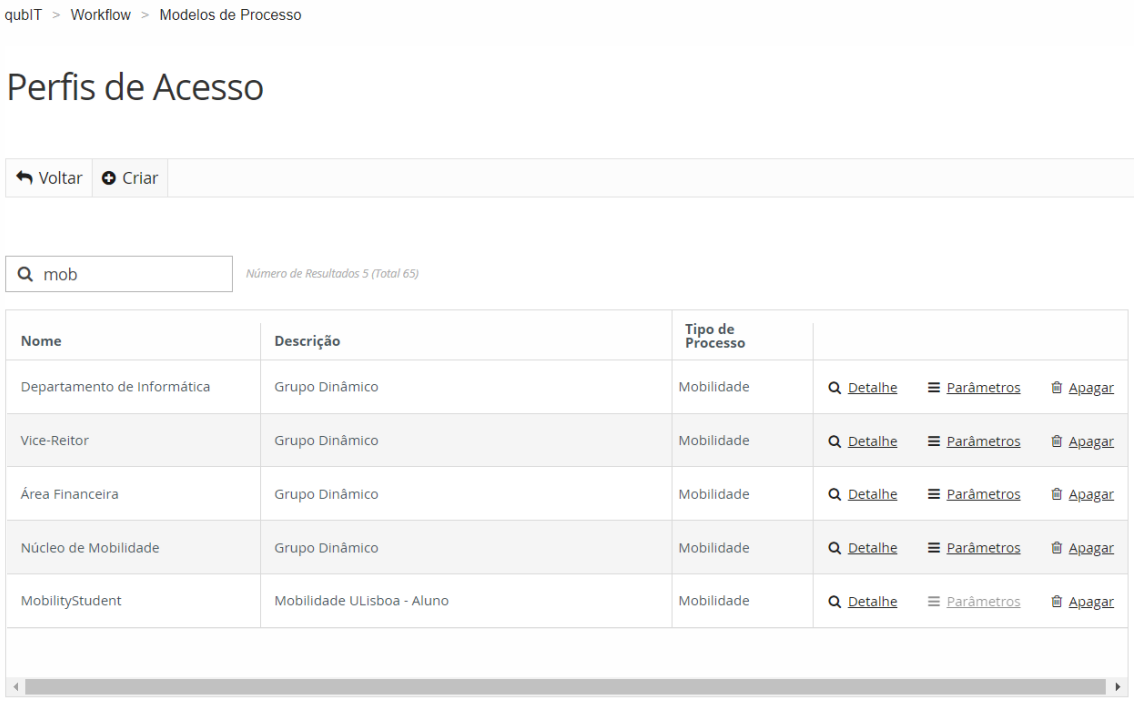

Figura 4.12: Ecrã dos perfis de acesso dos modelos de processo

Separadores

Os separadores costumizados de *workflow* são inicializados na PSL da mesma maneira que qualquer outro ecrã, como se pode observar na listagem 4.9 da secção anterior. Depois de compilada a PSL e gerada a classe do ecrã, é necessário indicar que essa classe implementa a classe *IUILayerWorkflowScreenEditor* para poder ser utilizada como separador do *workflow*.

Os separadores costumizados são ecrãs dentro do processo de *workflow*, e são implementados da mesma forma que qualquer outro, como se consegue visualizar na listagem 4.14.

```
1  (...)
2  @WorkflowComponent(type = "ULISBOA_STUDENT_MOBILITY")
3  public class ReadContractInsideWorkflow extends ReadContractInsideWorkflow_Base
       implements IUILayerWorkflowScreenEditor {
4
5      @Sticky
6      private UlMobContract ulMobContract;
```

```
7
8       private ScreenEditorContainerUI container;
9
10      @Override
11      protected void buildUI() {
12          super.buildUI();
13      }
14
15      public boolean isWritePermission() {
16          return this.container.isWritePermission();
17      }
18
19      @Override
20      protected Schema getSchema() {
21
22          Schema schema = super.getSchema();
23
24          if (ulMobContract.getIsInternship()) {
25
26              schema.add(i18n("Organization Name"), UlMobContract.META()
27                  .ORGANIZATION_NAME());
28              schema.add(i18n("Organization Type"), UlMobContract.META()
29                  .ORGANIZATION_TYPE());
30              schema.add(i18n("Organization Address"), UlMobContract.META()
31                  .ORGANIZATION_ADDRESS());
32              schema.add(i18n("Organization Area Code"), UlMobContract.META()
33                  .ORGANIZATION_AREA_CODE());
34              schema.add(i18n("Organization Area"), UlMobContract.META()
35                  .ORGANIZATION_AREA());
36              schema.add(i18n("Organization Email"), UlMobContract.META()
37                  .ORGANIZATION_EMAIL());
38              schema.add(i18n("Organization Phone"), UlMobContract.META()
39                  .ORGANIZATION_PHONE());
40              schema.add(i18n("Just-graduated"), UlMobContract.META().JUST_GRADUATED());
41          }
42
43          if (ulMobContract.getJustGraduated()) {
44              schema.add(i18n("Course Conclusion Date"), UlMobContract.META()
45                  .COURSE_CONCLUSION_DATE());
46          }
47          return schema;
48      }
49
50  (...)
51  }
```

Listagem 4.14: Classe do ecrã de leitura de processos de mobilidade

Depois de implementada a classe é necessário criar o separador no Fenix, adicioná-lo aos vários estados em que é utilizado e, por fim, acrescentar as permissões de leitura e escrita. As figuras 4.13 e 4.14 apresentam o ecrã de criação de um separador e o ecrã de separadores de um estado, respectivamente.

Operações

Para implementar operações de *workflow* é necessário criar uma nova classe para cada operação. Estas classes precisam de estender a classe *AbstractWorkflowOperation* ou, no caso de ser uma operação condicional, a classe *AbstractSimpleDecisionOperation*, para serem reconhecidas como operações de *workflow*.

Em operações não condicionais o método *execute()* é reescrito, e é neste método que é implementada a operação, como se pode observar na listagem 4.15; nos casos em que a

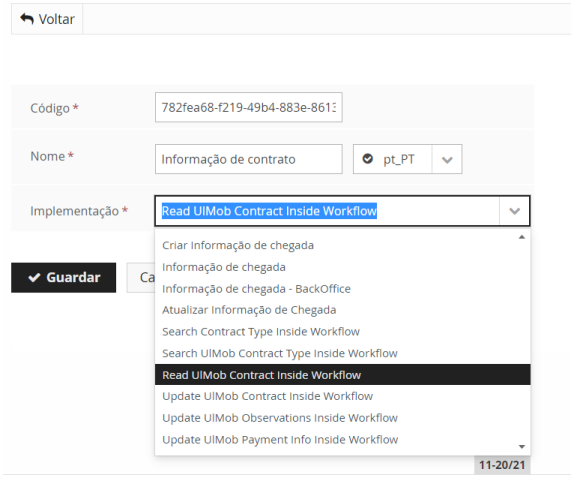

Figura 4.13: Criação do separador informação de contrato

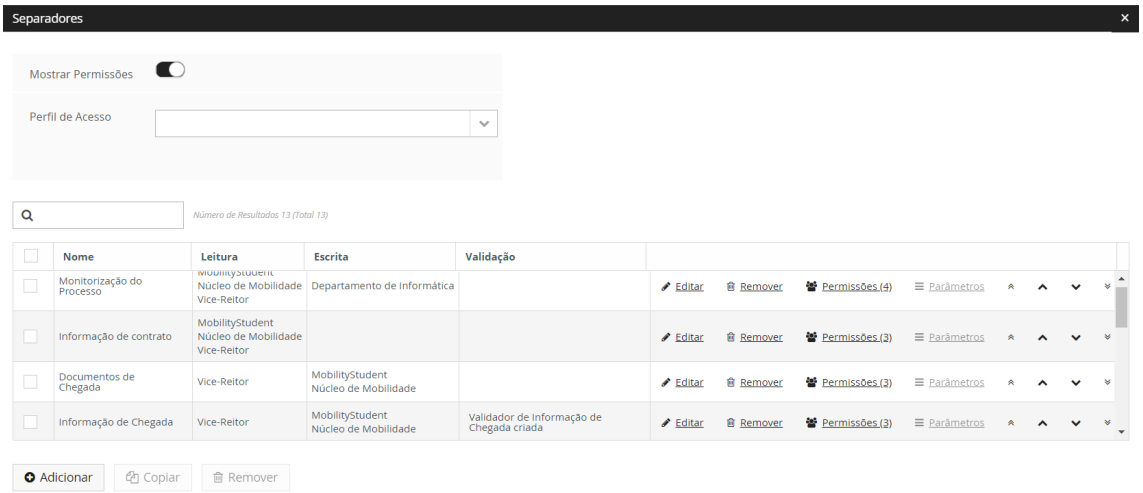

Figura 4.14: Gestão dos separadores de um estado

operação é condicional, é reescrito o método *evaluateCondition()* que devolve um booleano como mostrado na listagem 4.16.

Por fim é colocada a operação implementada como tipo da operação no estado automático, ou na operação respetiva do *workflow* como ilustrado na figura 4.15.

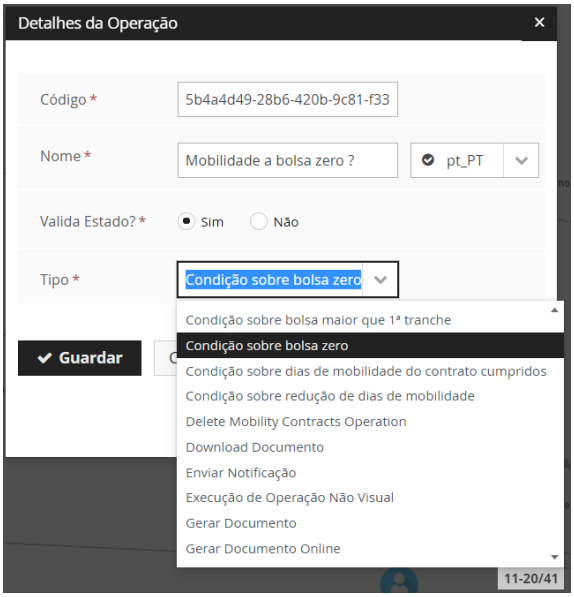

Figura 4.15: Detalhe do estado automático Mobilidade a Bolsa Zero?

```
1  (...)
2  @WorkflowComponent(type = "ULISBOA_STUDENT_MOBILITY")
3  public class ManualSignatureOperation extends AbstractWorkflowOperation {
4      @Override
5      public void execute() {
6          ServiceProvider.getService(TransactionalManager.class)
7              .executeInWriteContext(getClass().getName(), () -> {
8                  getState().getWorkflowInstance().getUlMobContract()
9                      .setManualSignature(true);
10                 getOperation().applyDefaultTransition(getState(), getResponsible());
11         });
12     }
13 (...)
14 }
```

Listagem 4.15: Classe da operação Assinar Contrato sem CMD

```
1  (...)
2  @WorkflowComponent(type = "ULISBOA_STUDENT_MOBILITY")
3  public class ZeroSchollarshipCondition extends AbstractSimpleDecisionOperation {
4      @Override
5      protected boolean evaluateCondition() {
6          UlMobContract contract = getState().getWorkflowInstance().getUlMobContract();
7          Double schollarshipValue = contract
8              .getLatestChangeSchollarshipAddendum() != null ? contract
9              .getLatestChangeSchollarshipAddendum().getSchollarshipValue() :
10             contract.getSchoolSchollarshipValue();
11         return contract.getSchollarshipDays() == 0 && schollarshipValue == 0;
12     }
13 (...)
14 }
```

Listagem 4.16: Classe da condição Mobilidade Bolsa Zero?

Validadores

Para implementar um validador de separador é necessário criar uma nova classe que estende a classe *AbstractScreenValidator* e implementar a classe *IWorkflowScreenValidator* para que este seja reconhecido no modelo de processos.

Nesta classe é reescrito o método *validate()* que devolve uma coleção de *WorkflowScreenValidationMessage*. Na listagem 4.17 é possível visualizar o código do validador correspondente ao utilizado no separador que se pode ver na figura 4.16.

Por fim é necessário colocar o validador no respetivo separador. Na figura 4.14 é possível visualizar o separador ”Informação de chegada” com o validador associado.

```
1  (...)
2  @WorkflowComponent(type = "ULISBOA_STUDENT_MOBILITY")
3  public class ArrivalInformationFilledScreenValidator extends AbstractScreenValidator
       implements IWorkflowScreenValidator {
4
5      @Override
6      public Collection<WorkflowScreenValidationMessage> validate(WorkflowInstanceState
       instanceState, WorkflowScreen workflowScreen) {
7
8          UlMobContract contract = instanceState.getWorkflowInstance().getUlMobContract();
9
10         if (contract.getArrivalInformation() == null) {
11             return Collections.singleton(new WorkflowScreenValidationMessage(
12                     I18N.i18n(ArtifactHandler.BUNDLE,
13                     "The arrival information needs to be filled")));
14         }
15         else {
16             return new ArrayList<WorkflowScreenValidationMessage>();
17         }
18     }
19 (...)
20 }
```

Listagem 4.17: Classe do validador de informação de chegada criada

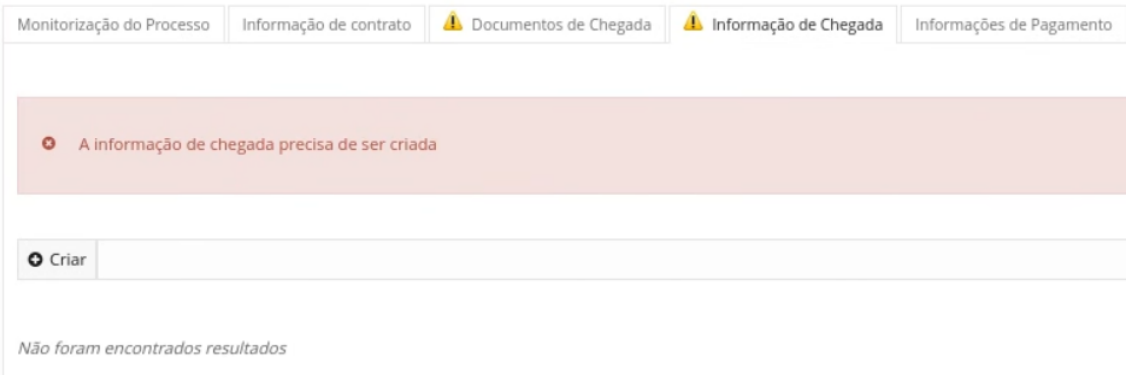

Figura 4.16: Ecrã do separador com o validador de informação de chegada criada

Notificações

As notificações enviadas dentro do *workflow* podem conter atributos do processo, como o nome do aluno. Para ser possível acrescentar esses atributos às notificações enviadas pelo *workflow* é necessário criar uma classe que implementa a classe *IWorkflowNotificationFieldsProvider*, e reescrever o método *getFields()* que devolve um dicionário de *Strings*. Este dicionário contém a *String* a utilizar na notificação e o atributo respetivo do processo, como se pode visualizar na listagem 4.18.

Quando se está a criar a notificação no *workflow* utiliza-se o formato "<String>" para o atributo ser colocado na notificação. Na figura 4.17 é possível observar a criação de uma notificação que utiliza o número de contrato e a escola de origem do aluno. Desta forma as notificações podem ser personalizadas para cada instância.

```
1  (...)
2  @WorkflowComponent(type = "ULISBOA_STUDENT_MOBILITY")
3  public class MobilityNotificationFieldsProvider implements
       IWorkflowNotificationFieldsProvider {
4
5      @Override
6      public Map<String, String> getFields(final WorkflowInstance workflowInstance, final
       Locale locale) {
7
8          final Map<String, String> result = Maps.newHashMap();
9
10         String workflowNr = workflowInstance.getNumber().toString();
11         String contractNr = workflowInstance.getUlMobContract().getContractNumber();
12         String studentName = workflowInstance.getUlMobContract().getName();
13         String destinationInstitution = workflowInstance.getUlMobContract()
14             .getDestinationInstitutionName();
15         String homeInstitution = workflowInstance.getUlMobContract()
16             .getHomeInstitution();
17
18         UlMobArrivalInformation arrivalInformation = workflowInstance.getUlMobContract()
19             .getArrivalInformation();
20         String arrivalInformationMobilityDays = null;
21         String arrivalInformationSchollarshipValue = null;
22         String arrivalInformationSecondPayment = null;
23
24         if (arrivalInformation != null) {
25             arrivalInformationMobilityDays = arrivalInformation
26                 .getTotalMobilityDurationInDays() == null ? null : arrivalInformation
27                 .getTotalMobilityDurationInDays().toString();
28             arrivalInformationSchollarshipValue =arrivalInformation
29                 .getFinalSchollarshipValue() == null ? null : arrivalInformation
30                 .getFinalSchollarshipValue().toString();
31             arrivalInformationSecondPayment = arrivalInformation
32                 .getSecondPayment() == null ? null : arrivalInformation
33                 .getSecondPayment().toString();
34         }
35
36         String addendumRemainderFirstPayment = workflowInstance.getUlMobContract()
37             .getLatestChangeSchollarshipAddendum() == null ? null : workflowInstance
38             .getUlMobContract().getLatestChangeSchollarshipAddendum()
39             .getRemainderFirstPayment().toString();
40
41         result.put("WorkflowNr", workflowNr);
42         result.put("ContractNumber", contractNr);
43         result.put("StudentName", studentName);
44         result.put("DestinationInstitution", destinationInstitution);
45         result.put("HomeInstitution", homeInstitution);
46         result.put("FinalMobilityDays", arrivalInformationMobilityDays);
```

```
47          result.put("FinalSchollarshipValue", arrivalInformationSchollarshipValue);
48          result.put("FinalSecondPayment", arrivalInformationSecondPayment);
49          result.put("AddendumRemainderFirstPayment", addendumRemainderFirstPayment);
50
51          return result;
52      }
53  (...)
54  }
```

Listagem 4.18: Excerto da classe *MobilityNotificationFiledsProvider*

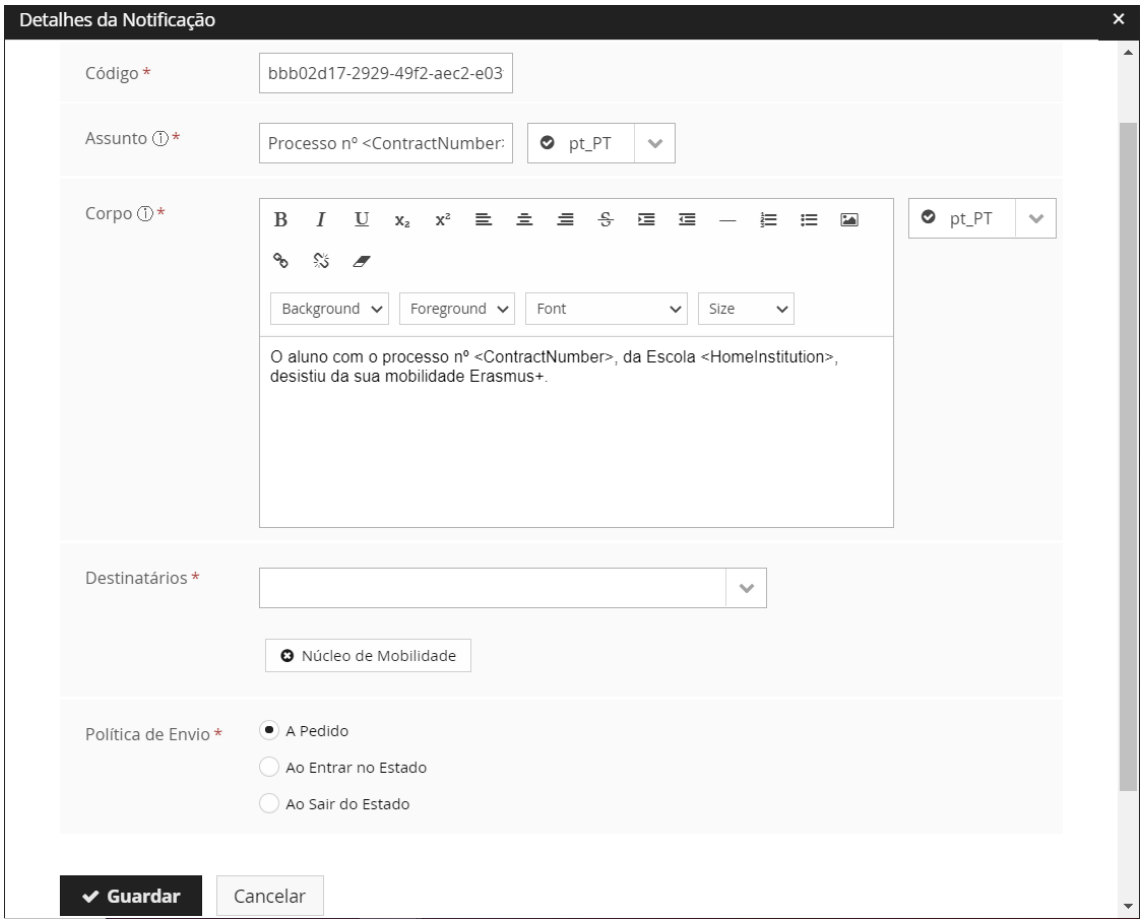

Figura 4.17: Criação de uma notificação de desistência

4.2 Entrada em produção e avaliação

A entrada em produção deste módulo foi condicionada pela pandemia do covid-19, o que levou a que existissem várias atualizações ao longo do desenvolvimento do projeto.

Nesta secção, para além da entrada em produção são abordados ainda os testes funcionais e de aceitação, assim como os testes de usabilidade realizados pelo NM.

4.2.1 Entrada em produção

Durante o desenvolvimento deste trabalho existiram quatro iterações que permitiram responder às necessidades criadas pela pandemia do covid-19.

Na primeira iteração foram colocadas em produção as funcionalidades referentes aos UC-01 e UC-02 mencionados no capítulo Análise e Desenho, evitando, assim, um aglomerado de alunos nas instalações da RUL.

Na terceira iteração foram colocadas em produção as funcionalidades referentes ao UC-04 e ao segmento referente ao fecho do processo do UC-05.

Na segunda e quarta iteração foram colocadas em produção algumas melhorias às funcionalidades que já estavam em produção.

Em todas as iterações o módulo passou por várias fases de teste.

Na primeira fase de testes foi criado um *docker* com as funcionalidades implementadas. Neste *docker* foram feitos testes funcionais pelo Núcleo de Gestão de Sistemas Académicos (NGSA) da RUL e, de seguida, foram corrigidos os erros encontrados. Na segunda fase foi atualizado o ambiente de qualidade do Fenix da RUL com o módulo de mobilidade, foram importadas as tabelas com os atributos dos processos e foram novamente realizados testes funcionais pelo NGSA da RUL.

Na terceira fase de testes, o ambiente de qualidade foi atualizado pela última vez e foram realizados testes de aceitação pelo NM.

No final de todos os testes o módulo foi colocado em produção no Fenix da RUL, foram adicionadas as notificações pretendidas a cada estado ou operação e as tabelas com os atributos dos alunos de mobilidade foram importadas. De forma a que os alunos conseguissem iniciar sessão no Fenix com as credenciais das suas escolas, foi necessário que o NGSA conectasse as contas dos alunos das escolas com os utilizadores criados aquando da importação das tabelas.

Depois da entrada em produção do módulo de mobilidade e até à data da entrega deste documento, foram inicializados 1957 processos de mobilidade e assinados 1255 contratos de mobilidade utilizando o módulo implementado.

4.2.2 Avaliação

Para avaliar o módulo implementado foram feitos testes de usabilidade com base no método SUS, um sistema de escala numérica utilizada para medir a usabilidade de um produto, serviço, *software*, entre outros. Para aplicar este método é necessário que os utilizadores utilizem o módulo e de seguida respondam a um questionário de 10 perguntas padrão, em que o utilizador pode responder com uma escala de 1 a 5, em que 1 corresponde a ”Discordo completamente” e 5 corresponde a ”Concordo plenamente”.

Depois de obter as respostas, a pontuação de cada questionário é obtida da seguinte forma:

- Em perguntas ímpares subtrai-se 1 à pontuação atribuída (exemplo – resposta: 5, pontuação: 5-1 = 4);
- Em perguntas pares subtrai-se a 5 a pontuação atribuída (exemplo – resposta: 2, pontuação: 5-3 = 2);
- Somam-se as pontuações e multiplica-se por 2,5;

Posteriormente a todos os questionários terem a sua pontuação faz-se uma média dos resultados, que podem variar entre 0 e 100, sendo que, segundo estudos realizados à escala SUS ao longo de muitos anos, 68 é uma pontuação acima da média [1].

Estes testes de usabilidade foram feitos com o NM do DREI, uma vez que são os maiores intervenientes na utilização do módulo. O NM é composto apenas por quatro funcionários da RUL que já estavam a utilizar o módulo de mobilidade desde a primeira entrada em produção, pelo que foi apenas requerido que respondessem ao questionário que se encontra no Anexo A. Os resultados obtidos podem ser observados na tabela 4.1.

Tabela 4.1: Resultados obtidos no questionário SUS ao NM

	R.1.	R.2.	R.3.	R.4.	R.5.	R.6.	R.7.	R.8.	R.9.	R.10.	Resultado bruto	Resultado final
Questionário 1	5	2	5	1	3	2	5	1	5	1	36	90
Questionário 2	5	4	4	2	4	4	4	2	4	2	27	67,5
Questionário 3	5	1	4	2	4	1	4	1	4	3	33	82,5
Questionário 4	5	3	4	3	3	2	4	2	4	2	29	72,5
											Média	78,13

A pontuação final do módulo de mobilidade na escala SUS foi de 78,13, pelo que se pode considerar um bom resultado [2].

4.3 Sumário

Neste capítulo foi resumida a implementação do módulo de mobilidade, incluindo as linguagens específicas de domínio e de apresentação, a forma como foi criado o *template* do contrato de mobilidade e os tipos de extensões de *workflow* criadas. Este capítulo ainda apresenta uma secção que engloba os passos realizados para a entrada em produção do módulo e uma avaliação utilizando o método SUS.

Capítulo 5

Conclusão e trabalho futuro

5.1 Conclusão

O processo de mobilidade inclui vários subprocessos que são executados nos Serviços Centrais e nas várias escolas da ULisboa. Apesar das candidaturas já serem suportadas pelo Fenix, a gestão de bolsas e contratos ERASMUS realizada nos Serviços Centrais era, ainda, feita de forma manual, sujeita a erros e incoerências, e pouco ecológica, uma vez que eram impressos inúmeros documentos para serem assinados. Este processo trazia consequências para os alunos nos atrasos dos pagamentos e para os Serviços Centrais na perspetiva de consumo de tempo e recursos humanos.

Com este projeto foi adicionado ao Fenix um novo módulo que suporta a gestão dos processos de mobilidade, o qual inclui a emissão e assinatura de contratos, assim como os cálculos dos dias de mobilidade e de valores de bolsa, tornando o procedimento mais rápido, menos sujeito a erros e passando-o para uma era digital e mais ecológica.

Os prazos de desenvolvimento e requisitos deste projeto foram condicionados pela pandemia, o que levou a que houvesse várias atualizações do módulo em ambiente de produção ao longo do tempo decorrido. Antes de cada uma dessas atualizações foram realizados testes funcionais em ambiente de qualidade por parte do NGSA, e posteriormente pelo NM.

Por último, foram feitos testes de usabilidade pelo NM, utilizando o método SUS. A pontuação obtida com este método foi de 78,13, o que permite concluir que o módulo está num bom caminho, tendo já bastantes funcionalidades em produção, mas que ainda há pontos a melhorar. Esses pontos são abordados na secção seguinte.

A implementação deste módulo foi o primeiro passo para que seja possível existir comunicação entre a instância Fenix da reitoria e as instâncias Fenix das escolas, uma vez que agora estão bem definidas as funcionalidades em que poderão ser necessárias essas comunicações, assim como os atributos a serem enviados durante as mesmas.

Este trabalho permitiu-me desenvolver competências de desenvolvimento de *software*, uma vez que me foi possível participar em todo o projeto. Realizei o levantamento e

análise de requisitos com a ajuda do NGSA e do cliente final, o NM; modelei e implementei o módulo de mobilidade, onde utilizei ferramentas e *frameworks* diferentes das que tinha utilizado durante o curso, com o apoio do NDS; e por último estive envolvida nos testes funcionais e de usabilidade e na entrada em produção na instância Fenix da RUL.

5.2 Trabalho futuro

Tendo em conta a dimensão deste processo existem ainda algumas tarefas a implementar que não faziam parte do plano de trabalho desta dissertação, entre as quais:

- Dar continuidade ao projeto através da modelação e implementação das componentes previstas para as escolas, incluindo a comunicação entre as instâncias Fenix das escolas e a instância da RUL;
- Implementar as assinaturas em massa e com atributos profissionais dentro do sistema Fenix. Assim o Vice-Reitor consegue assinar vários contratos de mobilidade de uma vez sem ser necessário exportar os documentos, assinar noutra plataforma e voltar a importá-los;
- Desenvolver a comunicação entre a instância Fenix da RUL e o sistema SAP. Desta forma será mais fácil de atribuir os números de compromisso aos alunos e deixará de ser necessário exportar uma tabela com alguns dados dos alunos, enviar para a área financeira e voltar a importar a tabela com os números de compromisso;
- Implementar a integração com o projeto *Erasmus Without Paper* [7]. Este projeto visa facilitar as comunicações entre as escolas de origem e as de destino. Prevê-se, deste modo, minimizar os erros correntes do tratamento de vários processos manualmente e em papel, como os casos em que os alunos trazem um TR que não corresponde com as disciplinas do LA.

Bibliografia

[1] 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.

[2] 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.

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

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

[5] João Cachopo and António Rito-Silva. Versioned boxes as the basis for memory transactions. *Science of Computer Programming*, 63(2):172–185, 2006.

[6] Scott Chacon. Git. `https://git-scm.com/about`. (Acedido em 02/01/2020).

[7] EWP. About ewp. `https://www.erasmuswithoutpaper.eu/about`. (Acedido em 30/05/2021).

[8] The Apache Software Foundation. Apache maven project. `https://maven.apache.org/what-is-maven.html`. (Acedido em 02/01/2020).

[9] Oracle. Java 8. `https://java.com/en/download/faq/java8.xml`. (Acedido em 02/01/2020).

[10] Oracle. Mysql. `https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html`. (Acedido em 02/01/2020).

[11] Fundação para a ciência e a tecnologia. Fénix framework. `http://fenix-framework.github.io/`. (Acedido em 05/02/2020).

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

[13] Instituto Superior Técnico. Fenixedu. `https://confluence.fenixedu.org/display/FENIXEDU/Welcome`. (Acedido em 05/02/2020).

[14] ULisboa. Mobilidade para estágios. `https://www.ulisboa.pt/info/mobilidade-de-estudantes-para-estagio`. (Acedido em 25/04/2021).

[15] ULisboa. Organização da ulisboa. `https://www.ulisboa.pt/info/organizacao`. (Acedido em 25/04/2021).

[16] ULisboa. Participantes com necessidades especiais. `https://www.ulisboa.pt/info/participantes-com-necessidades-especiais`. (Acedido em 25/04/2021).

[17] ULisboa. Sobre nós ulisboa. `https://www.ulisboa.pt/sobre-nos`. (Acedido em 25/04/2021).

[18] ULisboa. Relatório de gestão e de atividades 2020. 2021. `https://www.ulisboa.pt/sites/ulisboa.pt/files/documents/files/relatorio_de_atividades_2020_final_aprovado_enviado_tc.pdf`.

Abreviaturas

AM *Access Manager*.

AMA Agência para a Modernização Administrativa.

AN Agência Nacional.

CRUD *Create, Read, Update, Delete*.

DML *Domain Modeling Language*.

DREI Departamento de Relações Externas e Internacionais.

DSL *Domain Specific Language*.

ECTS *European Credit Transfer System*.

ERASMUS *European Region Action Scheme for the Mobility of University Students*.

IDM *Identity Manager*.

JSON *JavaScript Object Notation*.

JVM *Java Virtual Machine*.

LA *Learning Agreement*.

MT+ *Mobility Tool+*.

MVC *Model-View-Controller*.

NDS Núcleo de Desenvolvimento de *Software*.

NEE Necessidades educativas especiais.

NGSA Núcleo de Gestão de Sistemas Académicos.

NM Núcleo de Mobilidade.

OLS *Online Linguistic Support*.

POM *Project Object Model*.

PSL *Presentation Specific Language*.

RAIDES Registo de Alunos Inscritos e Diplomados do Ensino Superior.

REST *Representational State Transfer*.

RGPD Regulamento Geral sobre a Proteção de Dados.

RUL Reitoria da Universidade de Lisboa.

SAP *Systems Applications and Products*.

SAS Serviços de Ação Social.

SUS *System Usability Scale*.

TR *Transcript of Records*.

UC *Use Case*.

ULisboa Universidade de Lisboa.

XML *EXtensible Markup Language*.

Apêndice A

Questionário SUS

O presente questionário foi realizado no âmbito do trabalho de projeto "Gestão de alunos erasmus no sistema fenix".
Este projeto teve como objetivo principal automatizar os processos ERASMUS+ que passam pela reitoria da Universidade de Lisboa.
Agradeço desde já a sua participação.

*Obrigatório

System Usability Scale

Para cada uma das seguintes afirmações, escolha a opção que melhor descreve as suas reações atuais ao módulo de mobilidade desenvolvido para o sistema fenix da Reitoria da Universidade de Lisboa.

1. Gostaria de utilizar este módulo frequentemente *

Marcar apenas uma oval.

	1	2	3	4	5	
Discordo completamente	◯	◯	◯	◯	◯	Concordo plenamente

2. O módulo é desnecessariamente complexo *

Marcar apenas uma oval.

	1	2	3	4	5	
Discordo completamente	◯	◯	◯	◯	◯	Concordo plenamente

4. Necessito de ajuda de um técnico para conseguir usar o módulo *

Marcar apenas uma oval.

	1	2	3	4	5	
Discordo completamente	◯	◯	◯	◯	◯	Concordo plenamente

5. As várias funções do módulo estão bem integradas *

Marcar apenas uma oval.

	1	2	3	4	5	
Discordo completamente	◯	◯	◯	◯	◯	Concordo plenamente

6. O módulo apresenta demasiadas inconsistências *

Marcar apenas uma oval.

	1	2	3	4	5	
Discordo completamente	◯	◯	◯	◯	◯	Concordo plenamente

7. As pessoas irão aprender a usar o módulo muito rapidamente *

Marcar apenas uma oval.

	1	2	3	4	5	
Discordo completamente	◯	◯	◯	◯	◯	Concordo plenamente

8. Muitas das funcionalidades do módulo não se percebe bem para que servem *

Marcar apenas uma oval.

	1	2	3	4	5	
Discordo completamente	◯	◯	◯	◯	◯	Concordo plenamente

9. Senti-me confiante e tranquilo/a a usar o módulo de mobilidade *

Marcar apenas uma oval.

	1	2	3	4	5	
Discordo completamente	◯	◯	◯	◯	◯	Concordo plenamente

10. Tive de aprender demasiadas coisas antes de começar a usar o módulo *

Marcar apenas uma oval.

	1	2	3	4	5	
Discordo completamente	◯	◯	◯	◯	◯	Concordo plenamente

11. Nome do participante *

_____

12. Comentários (opcional)

_____

_____

_____

_____

_____