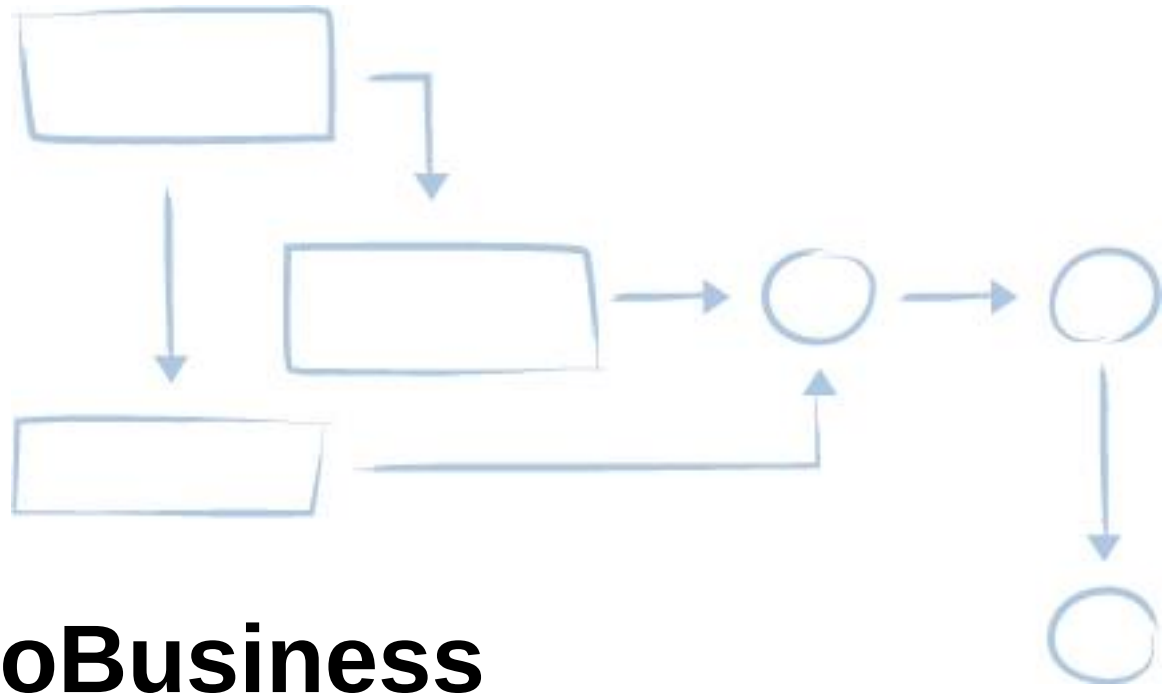


anoBusiness
ferramenta de BPM

Carlos Miguel Marcelino Gama Caldas

Mestrado Integrado em Engenharia de Redes e Sistemas Informáticos

Departamento de Ciência de Computadores

2015

Orientador

Pedro Leite, *Team leader* , ANO Software

Coorientador

Pedro Ribeiro, Professor Auxiliar, Faculdade de Ciências da Universidade do Porto

U.PORTO

FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

Todas as correções determinadas pelo júri, e só essas, foram efetuadas.

O Presidente do Júri,

Porto, _____/_____/_____

Resumo

Este relatório, elaborado no âmbito do Estágio Profissionalizante do Mestrado em Engenharia de Redes e Sistemas Informáticos da Faculdade de Ciências da Universidade do Porto, tem como intuito a descrição das atividades desenvolvidas na ANO Software - Sistemas de Informática, Lda durante os seis meses de estágio.

O desafio colocado neste estágio foi realizar um acrescento às funcionalidades do Projeto F (assim designado por questões de confidencialidade), *software* relacionado com gestão documental e processual. Este, apresentava como limitação a impossibilidade de criar processos estruturados no envio de documentos por parte de uma organização. Os documentos circulavam ao longo de uma organização mas o utilizador era o responsável por definir o seu destino. É neste contexto que surge o interesse no *Business Process Management* (BPM). A integração de um motor de BPM, onde assenta a lógica das regras de encaminhamento, juntamente com uma ferramenta gráfica para o desenho de *workflows*, permitiria ao utilizador definir à priori o "caminho"que o documento vai seguir, automatizando este procedimento.

Assim, foi realizada uma análise aos *Business Process Management Systems* (BPMS) disponíveis no mercado em formato *open source* e, posteriormente, feita uma escolha com base nos seguintes critérios: licenciamento, biblioteca usada para a criação da interface gráfica, existência de uma ferramenta *web* para modelar *workflows*, qualidade do aspeto gráfico, complexidade de interação com o motor e existência e qualidade de documentação. Na fase de implementação, foi realizado um módulo complementar ao Projeto F, para determinar a viabilidade para criar, de raiz, um modelador de processos de negócio. As dificuldades encontradas, dando destaque ao facto do Projeto F e do sistema de BPM utilizarem tecnologias distintas, levaram a um atraso na abordagem a utilizar e à sua integração, o que impossibilitou a sua conclusão no período de estágio definido. No entanto, a empresa dará continuidade ao trabalho realizado.

Abstract

This professional internship report was developed with the purpose of obtaining the Master Degree in Network and Systems Engineering of the Faculty of Sciences - University of Porto. It describes the activities carried on ANO Software - Sistemas de Informática, Lda in a six month internship.

The challenge proposed in this internship was to improve project F (labeled this way due to confidentiality issues), a documental and process management software, with new features. This software lacked the funcionality of creating structured processes on the sending of documents by an organization. The documents flowed through the organization but the user was responsible for determine their destination. In this context, the interest on Business Process Management (BPM) arise. The integration of a BPM engine where the logistics of fowarding rules together with a graphic tool for workflow design would allow the user to define a priori the document path and consequently automating the process.

Therefore, an analysis was conducted to available BPMs in the market under the open source licences. Then, a choice was made based on the following criterias: licensing, library used for the development of graphical user interface, existence of a web tool for workflow modelling, graphical appearance, complexity of engine interaction, and the existence and quality of the documentation provided. In the implementation step, a complementary module to Project F was accomplished to determine the viability of creating businnes process modeler from scratch. The difficulties faced, particularly the fact that Project F and the BPM system use different technologies, lead to a delay on the choice ofb the approach to use and consequently its integration. This made impossible the conclusion in the defined internship time. However, the company will continue the work done.

Conteúdo

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

Num mercado cada vez mais dinâmico e competitivo, as organizações são forçadas a desenvolver produtos e serviços mais específicos em períodos temporais mais reduzidos. O acesso à informação que a Internet proporcionou ao longo dos anos fez diminuir o ciclo de vida dos produtos e evidenciou a necessidade das empresas se adaptarem [1].

Atualmente, o sucesso é o objetivo comum de todas as empresas. Os sistemas de informação assumem um papel preponderante na obtenção desse objetivo, e estes aliados ao *Business Process Management* (BPM), que é um "conjunto de conceitos, métodos e técnicas que suportam o desenho, administração, configuração e a análise dos processos de negócio", estruturam a forma mais eficiente de o atingir [1].

Os processos de negócio representam funções fundamentais para o bom desempenho da organização. A sua importância divide-se em dois grupos: a administração de negócios, cujo interesse passa por melhorar o sistema operacional da organização, aumentando a satisfação dos seus clientes, reduzindo o custo do negócio e procurando estabelecer no mercado novos produtos a um preço cada vez menor e a ciência de computadores, que permite analisar e mitigar as falhas estruturais dos processos e criar sistemas mais robustos e escaláveis [1].

Desta forma foi necessária a criação de ferramentas que possibilitem automatizar a gestão de processos de negócio, conhecidos por *Business Process Management Systems* (BPMS).

1.1 Objetivo

Com o objetivo de introduzir a capacidade de criar processos de negócio estruturados no envio de documentos do Projeto F, aplicação desenvolvida pela ANO Software, este trabalho consiste na integração e adaptação do motor de um sistema de BPM juntamente com uma ferramenta gráfica para o desenho e manutenção de *workflows*, bem como a execução de tarefas de forma totalmente integrada.

O propósito é tornar possível ao utilizador modelar os processos de negócio através de uma interface *web* para, assim, puder atribuir a cada documento o respetivo modelo (com a sequência de passos necessários) para automatizar o processo de negócio de acordo com as regras de encaminhamento previamente definidas.

Deste modo, a finalidade é agilizar este procedimento, colocando no sistema a responsabilidade de definir o trajeto que os documentos irão seguir ao longo de uma organização.

1.2 Abordagem

Inicialmente procedeu-se a um levantamento das soluções de BPMS existentes no mercado, analisando as vantagens e desvantagens que cada uma das ferramentas proporcionam de forma a garantir que a escolha seja sobre o *workflow* que melhor se enquadra no processo de integração.

De seguida, com recurso a alguns exemplos, realizou-se um estudo ao BPMS do ponto de vista do utilizador, de forma a perceber como é que os processos no Projeto F poderiam ser realizados usando a notação e a lógica do BPM.

Procedeu-se um estudo sobre a API, para analisar as suas potencialidades e para tornar, à posteriori, mais fácil o seu manuseamento.

Por fim, realizou-se a etapa de implementação foi dividida em duas fases. A primeira, correspondeu ao conhecimento das tecnologias usadas no Projeto F e à tentativa de criar uma interface gráfica que permitisse ao utilizador modelar os seus processos de negócio. A segunda fase, consistiu na integração de partes de um sistema de BPM com o Projeto F.

1.3 Estrutura do relatório

Ao presente capítulo introdutório seguem-se seis capítulos que abordam as temáticas associadas ao desenvolvimento deste relatório de estágio. De seguida, irei resumi-las de forma estruturada.

No capítulo 2, intitulado "Empresa"será feita uma breve descrição sobre a empresa onde o estágio curricular foi realizado.

No capítulo 3, "*Business Process Management*"será realizada uma descrição teórica abordando os vários constituintes do BPM.

No capítulo 4, "Análise"são apresentadas vários *Business Process Management Systems*. Essa análise permite verificar as diferenças entre as diversas soluções assim como perceber o que motivou a escolha final.

No capítulo 5, "Tecnologias"são abordadas as tecnologias de maior relevância na elaboração deste trabalho.

No capitulo 6, "Implementação"é detalhado todo o processo de implementação realizado até ao momento, apresentando as várias tentativas assim como todas as dificuldades encontradas.

Por ultimo, no capítulo 7, "Trabalho Futuro"são abordados os próximos passos a serem implementados e as conclusões mais relevantes de todo o trabalho desenvolvido.

Capítulo 2

Empresa

A ANO Software [2] - Sistemas de Informática, Lda - foca-se na conceção, desenvolvimento e comercialização de *software*, assim como os respetivos serviços de implementação e consultadoria.

Figura 2.1: Logótipo Ano Software

Atua sobre dois tipos de mercado, o setor público, onde trabalha com a administração pública central e local, órgãos responsáveis pela fiscalização e controlo de infrações de trânsito, realizando cerca de 70% do volume de negócio. Os restantes 30% correspondem ao setor privado como entidades bancárias, seguradoras, e serviços jurídicos e de advocacia. Encontra-se segmentada em oito grandes áreas de negócio [2]:

- Gestão Documental e Processual (onde se inclui o Projeto F);
- eGov - Governo Eletrónico;
- eProcurement - Contratação Pública Eletrónica;
- Contraordernações e multas de trânsito;
- Gestão e faturação para *Utilities*;
- Monitorização do território;
- Transcrição de voz para texto;

- *Outsourcing*;

2.1 Missão

A ANO atua no sector da tecnologia de informação, destacando-se no desenvolvimento de novos produtos, na sua constante evolução e orientada para melhor servir o cliente. Desta forma, é um grupo cujo interesse se concentra em antecipar as necessidades do cliente; estabelecer um relacionamento de confiança; investir na melhoria contínua dos seus produtos; procurar ser uma referência da setor onde opera [2].

2.2 Localização

As instalações encontram-se distribuídas por Portugal - Porto e Lisboa - e por Brasil - São Paulo.

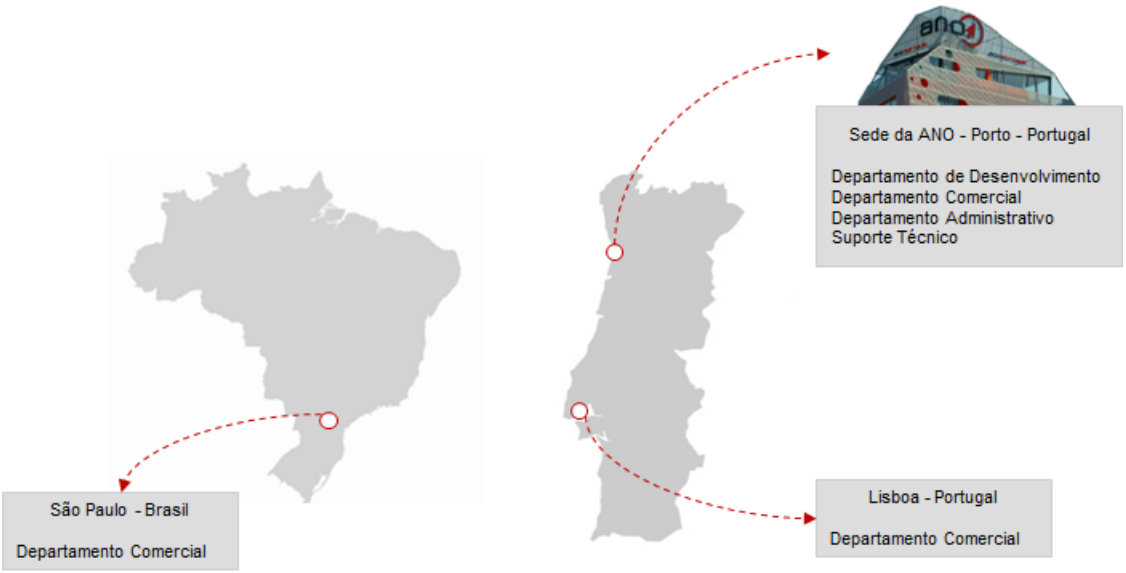


Figura 2.2: Localização da Ano Software

2.3 Certificação

A certificação permite ao cliente o conhecimento do compromisso por parte de uma organização na transparência e eficiência dos seus processos. Uma empresa do sector das tecnologias deve ter uma constante preocupação com aspetos relacionados com a segurança e qualidade dos seus produtos pois estes assumem um impacto significativo para a empresa caso não sejam satisfatórios. Desta forma, é necessário um controlo minucioso com abordagens sistemáticas e equipas competentes.

Trata-se de um instrumento para um sistema de melhoria contínua. A ANO Software está certificada na ISO 9001 (referencial internacional de gestão da qualidade) e na ISO 27001 (referencial internacional de gestão de segurança de informação).

2.3.1 ISO - International Organization for Standarization

As normas ISO (Organização Internacional de Padronização) têm como principal intuito a uniformização dos produtos/serviços através da implementação de regras, de forma a garantir que a qualidade dos produtos é assegurada e sempre que possível melhorada.

Estas normas são criadas de forma geral para que se apliquem a todas as empresas independentemente do tipo, dimensão e produto/serviço que produzem [3].

2.3.1.1 ISO 9001

A fidelização de um cliente passa pelo fornecimento do um produto/serviço que satisfaça todas as suas necessidades. A norma ISO 9001 ajuda as organizações a gerirem a qualidade dos seus produtos/serviços em função dos requisitos do cliente e baseia-se em 8 princípios fundamentais [3]:

1. Focalização no cliente;
2. Liderança;
3. Envolvimento das pessoas;
4. Abordagem por processos;
5. Abordagem da gestão como um sistema;

6. Melhoria contínua;
7. Tomada de decisão baseada em factos;
8. Relação de benefício mútuo com fornecedores.

Com a implementação desta norma, a empresa assume o seu compromisso com a satisfação do cliente e com a legislação aplicável.

A existência de um negócio é justificada pela existência de um cliente, pelo que contentá-lo, perfazendo as suas expectativas deve ser uma prioridade para qualquer entidade.

2.3.1.2 ISO 27001

A ISO 27001 visa auxiliar uma organização a manter segura a informação dos seus respetivos ativos. É o padrão mais conhecido no conjunto de normas da ISO 27000 fornecendo um conjunto de requisitos para a implementação de um sistema de gestão de segurança da informação.

Trata-se de uma abordagem completa à segurança de informação visto que aborda múltiplos fatores tais como telecomunicações, segurança aplicacional, proteção do meio físico, recursos humanos, licenciamento, entre outros. É independente de fabricantes porque se destina ao estabelecimento de processos e procedimentos que posteriormente podem ser materializados de forma diferenciada consoante o ambiente organizacional e tecnologia da entidade [4].

2.4 BPM no Projeto F

A ideia de um sistema de BPM no Projeto F é providenciar um motor de *workflow* estruturado para incumbir algo que este projeto ainda não tinha: a possibilidade de implementar processos de negócio.

Antes, o Projeto F só registava e enviava documentos, fazendo-os transitar por uma organização mas sem um conjunto de regras definidas. O responsável pelo envio de um determinado documento teria que proativamente saber para quem expedir o pedido. Além de tornar este procedimento mais propício ao erro, não corresponde a trabalhar de forma estruturada.

O uso de BPM introduz a capacidade de implementar qualquer modelo de negócio na solução, estendendo assim o encaminhar sem qualquer regra associada. As regras de encaminhamento (de negócio) são previamente definidas e o sistema fica responsável por definir para quem o pedido vai ser enviado tornando este procedimento mais rápido, simplificando também a tarefa de quem regista.

Deixamos assim de falar só de documentos e passamos a falar de processos.

Capítulo 3

Business Process Management

3.1 História do BPM

A origem do BPM remonta a 1880, quando Frederick Taylor, decide implementar metodologia cientifica e melhoria de processos na administração das suas empresas com objetivo de aumentar a sua eficiência e eficácia operacional. Estas técnicas tinham como base o trabalho manual e a produção de processos [5].

Walter Shewhart, juntamente com William Deming e Joseph Juran, aliaram a melhoria de processos com o controlo estatístico levando a uma nova fase na gestão de processos. As suas ideologias, passavam pela medição e limitação das alterações dos processos, implementação de mão de obra especializada, procurando a melhora contínua dos processos. O mercado Japonês reuniu a necessidade de se recuperar do pós-guerra com a sua capacidade disciplinar para se tornarem o foco em programas de melhoria contínua. A Toyota, empresa de produção de automóveis, é um desses exemplos. Desenvolveu o *Toyota Production System* (TPS), que através da descentralização das suas equipas de produção para procurar minimização dos custos e, utilizando como base os resultados obtidos nas experiências, realizavam pequenas alterações para consequentemente melhorar o seu rendimento [5].

Em 1990, numa altura em que o mercado ocidental enfrentava uma recessão económica e uma grande competitividade, especialmente por parte das empresas Japonesas, surgiu a necessidade de introduzir novos ideais à gestão de processos. A reformulação radical, contrariamente à melhoria incremental, a melhoria com base em objetivos e o uso de tecnologias de informação são algumas das medidas que foram implementadas.

Tratava-se da primeira gestão de processos que não tinha como principal objetivo a produção. Mas, a sua dificuldade de implementação levou as organizações a desistirem da sua utilização [5].

A Motorola criou o *Six Sigma* sendo posteriormente popularizado pela General Electric. Algumas das suas características assemelham-se ao controlo estatístico de processos. O *Six Sigma* estava vocacionado para processos de pequena dimensão e para melhoria incremental [5].

No início do século XXI, os processos de negócio e a tecnologia convergem, dando origem ao BPM. Essa união levou à criação de ferramentas capazes de realizar a gestão de processos de negócio de forma automática, conhecidos como BPMS.

3.2 Notação

A extensa quantidade de componentes para representar os processos de negócio levou à criação de notações de forma a facilitar o seu reconhecimento, simular o seu comportamento e analisar possíveis combinações entre eles [6].

O *Business Process Model and Notation* (BPMN) é a notação padrão para os processos de negócio, criada em 2004 pela *Object Management Group* (OMG). Atualmente a versão mais recente corresponde à 2.0 lançada em 2011 [1].

Os componentes do BPMN foram idealizados para serem facilmente distinguíveis e utilizam formas que são familiares à grande parte dos modeladores. A base dos elementos desta notação assenta em cinco categorias [7]:

- Objetos de Conexão
- Objetos de Fluxo
- *Swimlanes*
- Artefactos
- Dados

3.2.1 Objetos de Conexão

Os objetos de conexão são responsáveis por relacionar vários objetos de fluxo.

Figura 3.1: Objetos de conexão básicos - Sequência, Mensagem e Associação

Sequência O fluxo de sequência é representado por uma seta sólida e é usada para mostrar a ordem da sequência de atividades de um dado processo.

Mensagem O fluxo de mensagem é representado por uma seta tracejada com um pequeno círculo no seu inicio e é usado para mostrar o caminho das mensagens entre dois participantes ou entidades do processo.

Associação O fluxo de associação é representado por uma seta tracejada e é usado para associar dados, texto ou artefactos aos objetos de fluxo. Mostram os *inputs* e *outputs* das atividades.

3.2.2 Objeto de Fluxo

Os objetos de fluxo são os principais componentes gráficos para definir o comportamento de um processo. Podem ser de três tipos: eventos, atividades ou *gateways*.

Figura 3.2: Objetos de fluxo básicos - Evento, Atividade e *Gateway*

Evento Os eventos são representados por círculos e permitem identificar onde é que um processo ou subprocesso inicia ou termina.

Na tabela que se segue são apresentados alguns tipos de eventos:

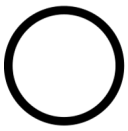
	Um ***start event*** é um mecanismo que instância um processo. Indica onde é que um processo ou subprocesso começa.
	O ***end event*** indica onde o processo ou subprocesso termina.
	O ***message start event*** permite ao receber uma mensagem por parte de um participante iniciar o processo.
	O ***timer start event*** permite iniciar um processo com base numa data específica ou intervalo temporal.
	O ***message end event*** indica que uma mensagem é enviada para um participante no final do processo.
	Um ***terminate event*** indica que todas as atividades no processo terminam imediatamente.

Tabela 3.1: Exemplo de tipos de eventos

Atividade As atividades são representadas por retângulos e é o termo genérico para o trabalho que a empresa realiza. Podem ser tarefas ou subprocessos.

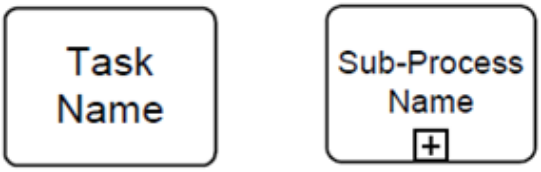


Figura 3.3: Atividades - Tarefas e Sub-processo

Na tabela que se segue são apresentados alguns tipos de tarefas:

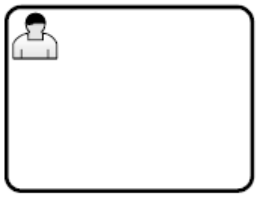
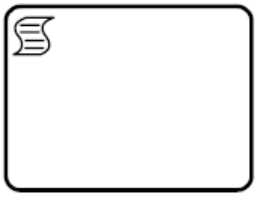
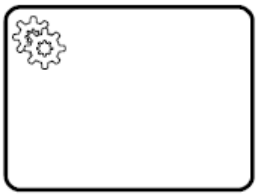
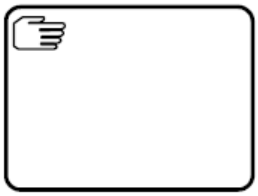
	A ***user task*** é usada para modelar uma tarefa a realizada por um utilizador. Quando o processo alcança esta tarefa, é adicionada à lista de tarefas do utilizador em questão.
	Uma ***script task*** é uma atividade automática. Quando o processo alcança esta tarefa, o *script* correspondente é executado.
	Uma ***service task*** é usada para invocar serviços externos, como *web services* ou aplicações automáticas.
	Uma ***manual task*** define uma tarefa externa ao motor do BPM. Para o motor, é encarada como uma atividade de passagem.
	Uma ***receive task*** é uma simples tarefa que aguarda pela chegada de uma mensagem. Após receber a mensagem o processo segue o seu fluxo.

Tabela 3.2: Exemplo de tipos de tarefas

Gateway Os *gateways* são representados por losangos e são responsáveis por ramificar ou unir os vários caminhos possíveis. O símbolo no seu interior indica o seu comportamento.

Na tabela que se segue são apresentados alguns tipos de *gateways*:

	O ***exclusive gateway*** funciona como um decisor no processo. Com base numa condição escolhe o caminho positivo para dar sequência ao processo.
	O ***parallel gateway*** permite executar vários caminhos em simultâneo. • *fork*: todos os fluxos de sequência de saída são executado em paralelo; • *join*: o *gateway* aguarda até que todas as execuções cheguem à sua porta de entrada;
	O ***inclusive gateway*** funciona como uma combinação entre o *exclusive* e o *parallel gateway*. Permite definir condições, no entanto, neste caso a sequência de fluxo pode seguir por vários caminhos. • *fork*: todas as sequências são avaliadas e as que obtiverem valor positivo são executadas simultaneamente; • *join*: contrariamente ao *parallel*, o *gateway* só vai esperar pelas sequências que foram executadas no passo anterior;
	O ***event-based*** permite tomar decisões com base em eventos. Cada sequência de fluxo tem que estar ligada a um evento, e só é executado quando um dos eventos é ativado.

Tabela 3.3: Exemplo de tipos de *gateways*

3.2.3 *Swimlanes*

As *swimlanes* são um mecanismo para organizar as atividades com base nas diferentes capacidades e responsabilidades através de uma separação gráfica por categorias.

Figura 3.4: *Swimlanes - Pool e Lane*

Pool Representa uma entidade no processo. Funciona como um contentor onde as atividades desse participante são colocadas no seu interior.

Lane É uma subpartição com uma *Pool* que estende as suas dimensões. É usada para organizar e categorizar as atividades.

As *Pools* são usadas quando se pretende separar entidades distintas. Assim, a comunicação entre as diferentes *Pools* tem que ser feita através de mensagens de fluxo. No caso das *Lanes*, são usadas quando se pretende separar atividades com funções especificas dentro da mesma entidade. Neste caso, para a conexão são usadas fluxos de sequência.

3.2.4 Artefactos

Os Artefactos permitem adicionar informação ao diagrama para facilitar o seu entendimento.

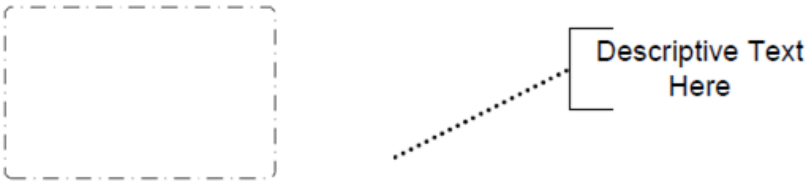


Figura 3.5: Artefactos - Grupo e Anotação

Grupo É representado por um retângulo tracejado. O agrupamento pode ser usado para documentação, mas não afeta os objetos de sequência

Anotação Trata-se de um mecanismo para introduzir informação textual adicional.

3.2.5 Dados

Os dados são elementos que armazenam ou transmitem informação durante a execução de um processo.

Figura 3.6: Dados - Objectos de dados e *Data Store*

Objeto de dados Servem para informar sobre as necessidades para executar as atividades e/ou informar sobre o que estas produzem. Podem representar objetos singulares ou uma coleção de objetos.

Data Store Trata-se um mecanismo para as atividades recuperar e atualizar informações que persistem para além do âmbito do processo.

3.3 Modelo

Um modelo de processos de negócio consiste numa representação gráfica das atividades de um processo. Uma instância representa um caso concreto do sistema operacional de uma entidade. Deste modo, cada modelo pode ter associadas várias instâncias de processos de negócio [1].

Os modelos são a base para implementar processos de negócio, podendo ser realizados através de regras e políticas organizacionais ou através de um BPMS [1].

Supondo um processo como a escrita de um relatório de estágio, em que o aluno envia o documento para o seu orientador e este determina se são necessárias alterações para a sua conclusão. No exemplo ilustrado na figura 3.7 podemos visualizar um possível

modelo para esse processo, permitindo fazer uma breve análise ao funcionamento do BPM assim como à sua notação.

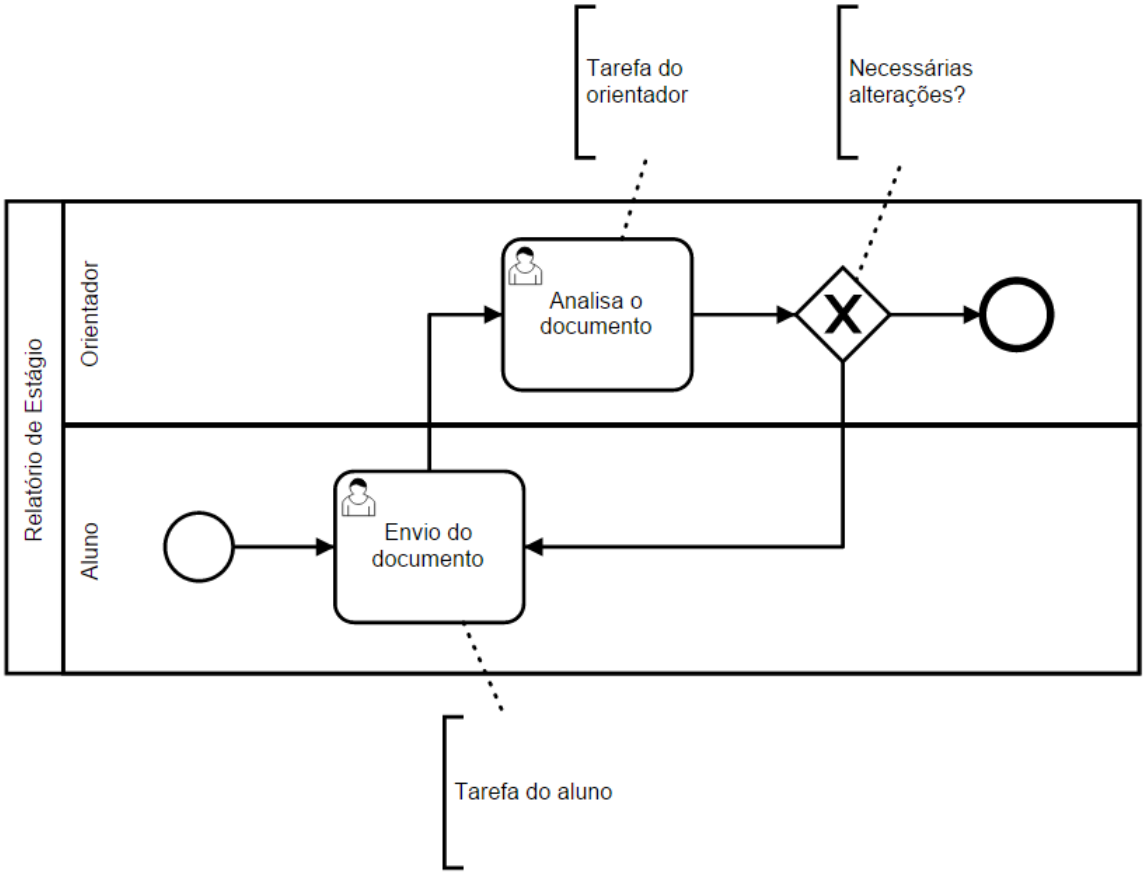


Figura 3.7: Exemplo de um modelo de processos

1. O processo é iniciado, a ação é encaminhada para um utilizador, neste caso, o aluno;
2. O aluno recebe a tarefa, associa-lhe uma versão do relatório e conclui a tarefa;
3. O orientador recebe a tarefa, analisa o documento e indica se são necessárias alterações, dando por terminada a sua tarefa;
4. Com base nessa decisão, na próxima fase do processo, o *exclusive gateway* define qual o caminho que o processo vai tomar. Caso o orientador tenha indicado a necessidade de alterações, então é atribuída uma nova tarefa ao aluno repetindo dessa forma o procedimento; caso contrário, o processo segue em direção ao seu final;
5. O processo termina.

3.4 Sistema

Um BPMS é um conjunto de ferramentas que visam automatizar a gestão de processos de negócio. Usualmente, é uma plataforma que suporta o ciclo de vida dos processos na sua totalidade, oferecendo ferramentas para o desenho de processos, criação de formulários, gestão de tarefas, definição de *workflows*, monitorização em tempo real das atividades assim como criação e análise de relatórios e *dashboards* [1].

Visa auxiliar toda a estrutura de uma organização. Do ponto de vista administrativo, permite analisar os processos que se encontrem concluídos, em andamento ou verificar o seu atraso associado. Os utilizadores envolvidos no processo também ficam com o seu trabalho simplificado uma vez que após executarem as suas tarefas o sistema é o responsável por encaminhar o processo para o passo seguinte.

3.5 Ciclo de vida

O ciclo de vida do BPM permite analisar as várias fases suportadas por um sistema. Estas encontram-se relacionadas e seguem uma estrutura cíclica. Apesar de não de existir uma uniformização relativamente ao número de fases e ao seu conteúdo, neste relatório vamos considerar a seguinte estruturação: Desenho e Análise, Configuração, Execução e Avaliação.

Figura 3.8: Ciclo de vida dos processos. Adaptado de [1]

3.5.1 Desenho e Análise

Esta é fase inicial do processo, onde é realizado um levantamento sobre a sua estrutura organizacional e ambiente tecnológico. A partir desse momento os processos são identificados e representados através de modelos com base em notações gráficas para que as várias áreas de uma organização consigam comunicam eficientemente [1].

Após o desenho, os processos de negócio têm que ser validados. A validação consiste em verificar se o comportamento da instância do processo se reflete com o modelo. Técnicas de simulação podem ser usadas como auxilio à validação, permitindo detetar falhas na execução do processo [1].

3.5.2 Configuração

Segue-se a fase correspondente à implementação. Esta, pode ser realizada através de procedimentos aos quais os funcionários de uma organização têm que obedecer, sem necessidade de recorrer a um sistema. No caso de se utilizar um BPMS, é necessária a escolha de uma plataforma de aplicação. O sistema precisa de ser configurado de acordo com o ambiente organizacional dos processos de negócio. A configuração contém as interações dos funcionários com o sistema assim como a integração de

software com o BPMS. Esta etapa pode também envolver aspetos transacionais através de tecnologias de base da dados [1].

Após a configuração do sistema, a implementação tem que ser testada. Usualmente a unidade de engenharia é a responsável por verificar se o sistema tem o comportamento esperado. Relativamente ao processo, os teste de integração e desempenho são importantes para detetar potenciais problemas de execução [1].

3.5.3 Execução

Findada a fase de configuração, as instâncias do processo podem ser executadas para cumprir o objetivo de negócio da organização. O BPMS é responsável por controlar a sua execução garantindo as restrições especificadas no modelo. Nesta fase pode usar-se ferramentas de monitorização para analisar informação relativa ao estado do processo e técnicas de visualização para verificar facilmente os processos ativos e inativos [1].

3.5.4 Avaliação

A fase de avaliação visa melhorar os modelos e as suas implementações com base nas informações disponíveis. O *Business Activity Monitoring* (BAM) é uma das técnicas usadas para o efeito que permite identificar se uma determinada atividade tem um tempo de execução muito elevado tendo em conta a escassez de recursos necessários [1].

3.5.5 Administração e *Stakeholders*

O ambiente técnico e organizacional da gestão de processos de negócio tem que ser adaptado à dimensão da organização em que está inserido. Em termos tecnológicos, é importante garantir mecanismos eficientes de armazenamento, recuperação e de acesso a modelos de processos. Os vários intervenientes da estrutura organizacional devem ter as suas funções bem definidas e diferenciadas de forma de desenvolver adequadamente a execução dos processos. Desse modo, a estrutura deve ter pessoas responsáveis pela programação, pela arquitetura do sistema, pelo desenho do processo, pela engenharia dos negócios, entre outros [1].

Capítulo 4

Análise

Na atualidade existe uma grande variedade de BPMS. Foi realizado um estudo de mercado para determinar quais as diferenças entre as várias soluções a fim de encontrar a melhor ferramenta para a integração que se pretendeu realizar.

Como requisitos, o BPMS teria que ser baseado em Java, por ser a linguagem de programação usada no Projeto F, e ser *open source* para puder ser adaptado às necessidades exigidas, dando preferência por produtos gratuitos e que fizessem uso da notação BPMN 2.0 como garantia da atualidade do produto.

Esta análise teve como base os seguintes critérios:

- Licenciamento: para garantir que a ferramenta pode ser modificada e distribuída num contexto empresarial onde pode ser integrada num *software* proprietário;
- *Framework* da interface gráfica: idealmente JavaServer Faces por ser a *framework* usada no Projeto F, ou outra *framework* que seja possível integrar;
- Ferramenta *web* para modelar processos: critério preferencial para tornar mais fácil ao utilizador o desenho e modelação de processos;
- Qualidade do interface gráfico: qualidade do aspeto da aplicação *web* e do modelador tendo sido definido com base nas opiniões de vários responsáveis pelo projeto (baixo - médio - alto);

4.1 Matriz de comparação

BPMS	*Framework*	*Engine license*	*Web modeler*	*Modeler license*	Qualidade gráfica
jBPM	GWT	Apache 2.0	Sim	MIT	baixo
Activiti	Vaadin	Apache 2.0	Sim	LGPL	médio
Camunda	GWT	Apache 2.0	Não		alto
Bonitasoft	GWT	LGPL	Sim	GPL	alto

Tabela 4.1: Tabela de comparação dos vários BPMS

4.1.1 Licenciamento

Um dos pontos a ter em consideração no que diz respeito à utilização de algum tipo de *software* é o seu licenciamento. É importante na medida em que permite garantir que o *software* é usado, modificado e distribuído da forma como terá sido previamente idealizado. Esta seleção deve ser baseada na compatibilidade entre os diferente *softwares* e nas restrições que se pretende que os trabalhos derivados detenham. Posto isto, vamos analisar as licenças usadas pelos vários BPMS.

4.1.1.1 GNU GPL v2

A General Public License (GPL) é uma licença de software livre, *copyleft*, que se baseia numa ideologia oposta à *copyright*. Permite a modificação e redistribuição de um programa mas exige os mesmos direitos ao novo programa. Ou seja, qualquer programa ao abrigo desta licença, pode ser modificado, mas a sua nova versão também, e assim sucessivamente, garantindo a continuidade de *software* livre [8].

4.1.1.2 GNU LGPL v2.1

A GNU Lesser General Public License (LGPL) surge aquando da versão 2 da GNU GPL e cede ao *software* que a utiliza mais direitos quando comparada com a licença GNU GPL. Isto deve-se ao facto de que, normalmente a GNU LGPL é utilizada para bibliotecas que permitem o vinculo com *software* que não é GNU GPL nem *open source*. Visto que a GNU GPL indica que é necessário que o outro *software* à qual está se encontra vinculado esteja sob a mesma licença, isto impede desenvolvedores de utiliza-la perante *software* proprietário. A GNU LGPL vem proporcionar uma

alternativa, não sendo necessário que, aquando da vinculação do código de outros projetos, este esteja licenciado com a mesma licença [9].

4.1.1.3 Apache license 2.0

Esta licença permite a todos os sujeitos uma licença de *copyright* para reproduzir, fazer trabalhos derivados do original, sub-licenciar e distribuir o trabalho e os seus derivados na forma de código e objeto. Por se tratar de uma licença não *copyleft*, qualquer utilizador que modifique o programa pode adicionar *copyright* e introduzir à licença termos e condições de uso, de reprodução ou de distribuição [10].

4.1.1.4 MIT license

Trata-se de uma licença permissiva, ou seja a qualquer pessoa que obtenha uma cópia do software e dos ficheiros de documentação associados, lidar com o *software* sem qualquer restrição, incluindo os direitos de usar, copiar, modificar, distribuir, sublicenciar, ou vender cópias do software. A única restrição é de que o software deve ser acompanhado pela licença [11].

4.2 BPMS

4.2.1 jBPM

O jBPM [12] é um BPMS produzido pela jBoss/Red Hat. Desenvolvido em Java, é uma ferramenta totalmente *open source* que permite aos seus utilizadores modelar, executar e monitorizar os processos de negócio. Pode ser executado em qualquer ambiente Java, integrado com outra aplicação ou utilizado como um serviço.

A modelação dos processos pode ser feita através de uma ferramenta designada por *Process Designer* sendo uma parte integrante da sua aplicação *web* ou de um *plugin* para o Eclipse que fornece os componentes do BPMN 2.0 e funcionalidades de *drag and drop* para facilitar a sua usabilidade. A aplicação *web*, desenvolvida a partir da *framework Google Web Toolkit* (GWT), contém também ferramentas para a gestão de tarefas e criação de formulários para serem usados ao iniciar novas instância do processos ou nas tarefas designadas a utilizadores. Na sua parte administrativa,

possibilita gerir os utilizadores e grupos assim como a sua respetiva autorização.

O acesso ao seu motor pode ser feito através da sua API ou através de uma arquitetura *Representational State Transfer* (REST). Esta utiliza o protocolo *Hypertext Transfer Protocol* (HTTP) para a troca de mensagens. A comunicação REST apresenta-se como sendo uma abordagem bastantes escalável e opera sobre pedidos de GET, POST, PUT, DELETE, funcionando como uma alternativa para integrações com projetos que não sejam baseados em Java [13].

Diferencia-se das outras ferramentas analisadas pelo facto de ter um BAM, um sistema para monitorizar as atividades, tendo funcionalidades como a visualização de *dashboards*, possibilidade de exportar dados para o Excel, configuração de tabelas de relatórios interativos, entre outras. Para além disso, o jBPM possibilita interligar a gestão de processos com o Drools, *software* desenvolvido pela mesma organização, responsável pela produção de regras de negócio.

É possivelmente uma das soluções BPM *open source* mais completas do mercado. No entanto, alia à sua completude uma complexidade que torna difícil e confusa a sua utilização quando comparada com outros BPMS. Dispõe de um motor bastante extenso e uma aplicação *web* que apesar das inúmeras funcionalidades, em conjunto com o seu aspeto gráfico extremamente confuso tornam difícil a sua usabilidade.

4.2.2 Activiti

O Activiti [14] é um *workflow* e plataforma de BPM desenvolvido pela Alfresco. Joram Barrez ex-funcionário da jBoss, juntamente com Tom Baeyens, foram os responsáveis por fundar este projeto. A sua mudança para o Activiti deveu-se ao facto de entenderem que a licença usada no jBPM, a LGPL, limitava o alcance a novos utilizadores [15].

O Activiti Explorer é a aplicação *web*, criada através da biblioteca Vaadin, que permite aos seus utilizadores aceder ao motor do Activiti através de uma interface simples e agradável. A gestão de tarefas é um dos instrumentos, onde o utilizador pode visualizar, criar, atribuir e executar tarefas. Visualizar e alterar as definições do processos, analisar as tabelas da base de dados e criar relatórios com base em estatísticas das atividades são outras das possibilidades.

Tal como acontece no jBPM, o processo de modelação pode ser feito de duas formas, através do Activiti Modeler onde pode criar os processos graficamente no próprio *browser* ou com recurso ao Activiti Designer, um *plugin* para o Eclipse que permite

modelar os processos BPMN 2.0 no próprio IDE.

Relativamente à API é relativamente mais simples comparada com a do jBPM. É uma das formas de interagir com o motor, podendo também fazê-lo através de uma arquitetura REST.

Na sua página existe um guia de utilização muito bem documentado que facilita imenso os primeiros passos de utilização desta ferramenta. Para além disso, dispõem de vários livros e de um fórum [16] onde é possível obter rapidamente respostas graças à sua comunidade extremamente ativa.

4.2.3 Camunda

O Camunda BPM [17] é um BPMS que surge através de uma divisão do projeto Activiti. Atualmente é um produto do Camunda Committer e é usado por grandes empresas a nível mundial como a seguradora Allianz e a empresa de aviação Lufthansa Technik [17].

Está disponibilizado sob duas versões, uma *open source* e uma versão *enterprise* que adiciona algumas funcionalidades e serviços, como o suporte 24/7, em troca de um determinado valor monetário.

Relativamente à versão *open source* apresenta bastantes semelhanças comparativamente o seu "antecessor"Activiti. Disponibilizada sob um interface gráfica muito bem conseguida em termos visuais, esta versão é mais limitada em termos de funcionalidades quando comparada com os restantes BPMS.

Apresenta como desvantagem não ter, por defeito, um modelador de processos na sua aplicação *web*. Dispõe de uma ferramenta, designada por Camunda *Cycle*, que auxilia a interligação com um modelador externo. No entanto, não identificamos no mercado soluções gratuitas que possibilitassem essa comunicação. Desta forma, a única maneira de desenhar os modelos é com o recurso ao Eclipse através de um *plugin* como nos casos enunciados anteriormente.

4.2.4 Bonita

Por fim, o Bonita BPM [18] é um produto da Bonitasoft. A Cisco, a Xerox e a Sony são algumas das empresas que fazem uso deste *software* [18]. A qualidade desta ferramenta já lhe permitiu arrecadar inúmeros prémios. O primeiro lugar atingido em 2014 na *Annual International Business Awards*, na categoria de melhor produto do ano para soluções de integração, é um dos destaques [19].

À semelhança do Camunda BPM, contém duas versões disponíveis: uma versão voltada para a comunidade, *open source* e gratuita, e uma versão voltada para empresas que necessita de uma subscrição. Esta, oferece um serviço de suporte profissional e acrescenta uma série de funcionalidades. A personalização da interface do cliente, monitorização das aplicações através de *dashboards* e relatórios, modificações em tempo real sem interromper a execução dos processos, adaptabilidade para sistemas *mobile* e a conexão avançada para sistemas empresariais são alguns dos extras incluídos nesta versão.

Este projeto, na sua versão gratuita, está dividido em três componentes, sendo eles o motor, o Bonita Studio e o Bonita Portal. O motor é uma API Java usada para interagir programaticamente com os processos. O Studio é uma ferramenta que é executada de forma independente do Portal para modelar os processos de negócio. Para a sua execução, é necessário ser instalado no posto onde se quer executar. O Portal consiste numa interface *web*, desenvolvida em GWT, para o utilizador realizar as tarefas que lhe estão associadas e visualizar os processos em que está inserido. Dispõe também de um editor, chamado UI Designer, onde é possível definir o *layout* de *widgets* e dos formulários que serão apresentados ao utilizador.

4.3 Escolha

Todos os BPMS descritos anteriormente são produtos estabelecidos no mercado que já sofreram constantes atualizações ao longo dos anos e dispõem de vários livros que auxiliam a sua utilização.

No caso do Bonita BPM o licenciamento tornou-se um impedimento para a sua utilização. O seu portal para modelar os processos de negócio encontra-se licenciado pela versão 2 do GNU GPL o que obriga a libertação do seu código fonte caso seja feita alguma alteração ao seu conteúdo.

Era importante ter uma interface na aplicação *web* que permitisse ao utilizador modelar os seus processos de negócio. Para pessoas que não sejam ligadas à área das tecnologias de informação, ter de recorrer a um IDE - ambiente de desenvolvimento integrado - como o Eclipse poderia tornar-se um procedimento inviável. Como consequência, o Camunda deixou de ser opção e criou mais um impedimento para uso do Bonitasoft.

O jBPM e o Activiti são projetos que se assemelham dada a origem similar. Assim, a escolha para o processo de integração recaiu sobre o Activiti. Apresenta um motor mais simples, melhor documentado e uma aplicação *web* mais agradável e intuitiva para o utilizador.

4.3.1 Estrutura

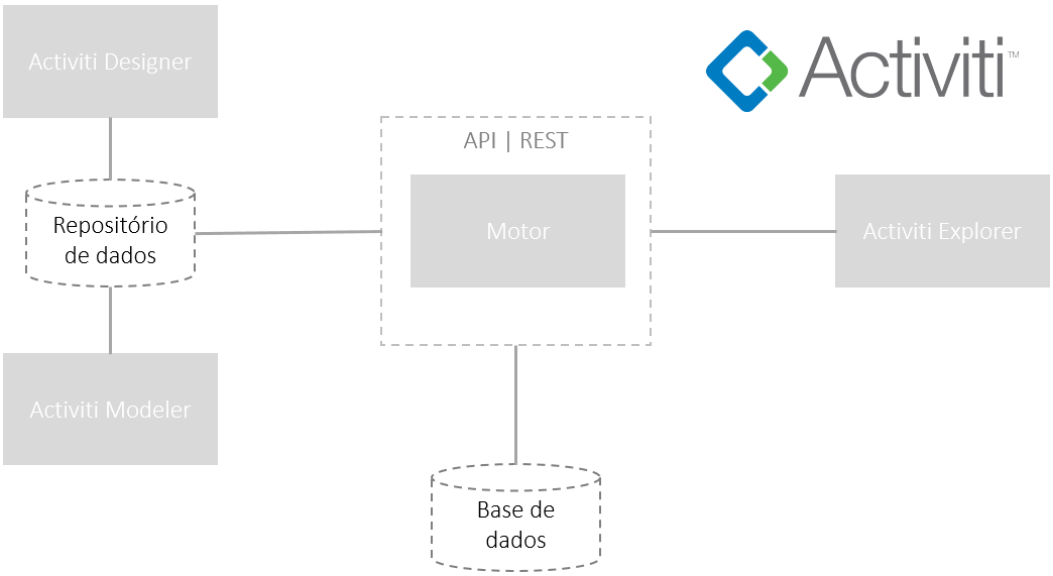


Figura 4.1: Arquitetura do Activiti

O **Motor** é o elemento fulcral do projeto. É onde assenta toda a lógica dos processos BPM. Além das opções que serão descritas seguidamente exequíveis pelo Activiti Explorer, acrescenta mais algumas funcionalidades. Os *event listeners* representam um pedaço de código Java ou um script e permitem adicionar informação técnica sem alterações gráficas no diagrama. Opções como notificar quando uma atividade é iniciada é uma das possibilidades.

O motor do Activiti estende as atividades padrão do BPMN. O programador tem a opção de personalizar uma atividade introduzindo código Java. Desta forma, simplifica

o desenho do modelo e a sua perceção.

O **Activiti Explorer** é uma forma dos utilizadores acederem ao motor através de uma aplicação *web*. Inclui as seguintes funcionalidades:

- Gestão de tarefas: ver lista de tarefas pessoais; ver a lista de tarefas do qual é um dos candidatos à sua execução; criar tarefas não relacionadas com o processo; completar tarefas através dos formulários apresentados; atribuir tarefas a outro utilizador; criar e atribuir subtarefas.
- Gestão: gestão de *deploys*, gestão da definição de processos, analise das tabelas de base de dados.
- Relatórios: os processos concluídos são guardados numa das tabelas da base de dados, as estatísticas que lhes estão associados podem conter informação importante para os responsáveis pela administração de negócios.
- Análise das instâncias dos processos;

O **Activiti Modeler** usado para modelar os processos BPMN 2.0 através do *browser*, abstraindo-se da componente técnica. É acessível pelo Activiti Explorer. Os modelos são convertidos para um formato *eXtensible Markup Language* (XML) e guardados na base de dados.

O **Activiti Designer** voltado para os responsáveis das tecnologias de informação, é um *plugin* para o Eclipse que permite modelar processos BPMN 2.0 no próprio IDE. Suporta todas as funcionalidades dos processos e do motor.

O Activiti suporta a conexão com as seguintes base de dados: H2, MySQL, Oracle, PostgreSQL, DB2 e MSSQL.

4.3.2 Activiti API

O uso da API é a maneira mais comum de interagir com o motor do Activiti. Tipicamente a comunicação inicia-se criando um *ProcessEngineConfiguration* que permite obter um *ProcessEngine*. Assim, torna-se possível aceder aos vários serviços que contêm os métodos de BPM. A API está dividida em sete interfaces onde cada uma corresponde a uma funcionalidade diferenciada [15].

FormService	Inclui os métodos para fazer a gestão de formulários usados para iniciar novas instâncias de processos e completar tarefas.
HistoryService	Contém dados históricos sobre as instâncias de processos que se encontrem concluídas.
IdentityService	Esta interface permite gerir os utilizadores, grupos e a relação entre ambos.
ManagementService	Permite consultar informações e meta-dados sobre as tabelas da base de dados.
RepositoryService	Esta interface fornece funcionalidades para consultar, eliminar e realizar os *deploys* das definições do processo.
RuntimeService	Permite iniciar e consultar as instâncias do processo.
TaskService	Os métodos referentes às tarefas fazem parte desta interface. Possibilita, por exemplo, criar ou atribuir tarefas e consultar a lista de tarefas de um dado utilizador.

A informação mantida na base de dados pode ser acedida de duas formas: através de *query* API ou por consultas nativas. A *query* API contém consultas pré-definidas e de fácil utilização. No entanto, por vezes é necessário executar consultas mais complexas, como a disjunção de consultas. Nesses casos, é possível realizar consultas nativas, através de consultas *Structured Query Language* (SQL) [20].

A construção de processos dinâmicos também é suportada pela API do Activiti. Através desta abordagem, é possível criar processos BPMN 2.0 sem recorrer a uma ferramenta de desenho (como o Activiti Modeler) ou sem ter que efetuar alterações nos ficheiros XML onde os processos estão representados.

Combinando com os métodos descritos anteriormente, é então possível pela API construir o processo, fazer o seu *deploy*, inicia-lo e executa-lo realizando assim todas as operações envolvidas.

Capítulo 5

Tecnologias

Este trabalho foi desenvolvido utilizando algumas tecnologias já existentes. As mais relevantes são o Java, presente na *framework* JavaServer Faces (JSF) utilizada no Projeto F, e no BPMS Activiti, o sistema de gestão de base de dados da Oracle e o Javascript.

O Java a foi tecnologia mais utilizada durante todo o projeto, uma vez que todo o trabalho de *back end* foi realizado com o recurso a esta linguagem, tanto para desenvolver no JSF como para comunicar com o motor do Activiti. A base de dados Oracle é utilizada como sistema de base de dados do Projeto F e do Activiti sendo também responsável pela interligação entre ambos. Para realizar o mapeamento objeto-relacional (ORM) foi utilizado o Hibernate, *framework* suportada por ambos os projetos. Permite simplificar o mapeamento de dados entre uma linguagem de programação orientada a objetos e uma base de dados relacional, onde as tabelas são representadas através das classes que são instanciadas com o seus respetivos registos. O Javascript foi utilizado como linguagem de *front end* para colmatar as debilidades do Primefaces, biblioteca de componentes do JSF. Tanto o Projeto F como o Activiti são projetos Maven. Esta ferramenta auxilia a gestão de dependências e do ciclo de vida dos projetos.

As tecnologias enumeradas anteriormente serão descritas neste capítulo, dando ênfase às suas características mais relevantes no decorrer do trabalho.

5.1 Java

A linguagem de programação Java foi lançada em 1995 por James Gosling e pela sua equipa de programadores na Sun Microsystems, empresa que posteriormente foi adquirida pela Oracle. Atualmente, é uma das linguagens de programação mais usadas à escala mundial [21].

Trata-se de uma linguagem orientada a objetos que permite criar eficientemente aplicações *desktop* e *mobile*. É independente de plataforma, o que permite executar em qualquer ambiente desde que tenha um interpretador de Java instalado. Contém uma biblioteca padrão volumosa, e dispõe inúmeros módulos e *frameworks* que incentivam o seu desenvolvimento.

Existem vários IDEs que aumentam a produtividade do desenvolvimento em Java, o NetBeans [22] é uma das possibilidades, utilizado neste trabalho devido às inúmeras funcionalidades que oferece numa interface organizada e amigável. De entre as funcionalidades disponíveis, podemos destacar a facilidade de *debug* de código com auxilio de *break points*, o suporte para estabelecer conexões com base de dados Oracle e o facto de ter desde já integrado um servidor de aplicação como o GlassFish [23].

5.2 Javascript

Esta linguagem foi criada por Brendan Eich na Netspace em meados dos anos 90. O seu nome sofreu várias alterações ao longo dos anos, desde Mocha, LiveScript até JavaScript como é conhecido na atualidade [24].

Trata-se de uma linguagem de *script* e voltada para o lado de cliente, onde o *browser* é o responsável por processar a sua informação. No entanto, ambientes como o node.js tem impulsionado a sua utilização do lado do servidor.

É indiscutivelmente uma das linguagens mais usadas para programação *web* dada a sua facilidade de aprendizagem e a sua flexibilidade. Para além disso, dispõe de uma série de bibliotecas que simplificam imenso o processo de desenvolvimento, sendo o jQuery uma das mais populares.

5.3 Oracle Database

A base de dados da Oracle [25] é um sistema de gestão de base dados relacional. Desenvolvido em 1977 por Lawrence Ellison, é um dos motores de base de dados relacionais mais confiáveis e populares por parte das organizações sendo amplamente utilizado. É multi-plataforma, e dispõe de várias soluções com base nas necessidades e orçamento disponível.

Apresenta um excelente desempenho mesmo quando desafiado com tarefas exigentes e ultrapassa facilmente o teste ACID (atomicidade, consistência, isolamento e durabilidade) sendo uma garantia de fiabilidade e competência [26].

Outra das vantagens da sua utilização, é o facto de incorporar tecnologia de *flashback*. Em caso de falha da aplicação, permite recuperar eficientemente dados que tenham sido removidos, minimizando o erro humano e o tempo necessário para o efeito [26].

5.4 Apache Maven

Apache Maven [27] é uma ferramenta de automação e compilação utilizada essencialmente em projetos Java. Encontra-se hospedado na Apache Software Foundation.

O Maven utiliza um ficheiro XML, designado por POM onde permite definir as dependências sobre módulos e componentes externos, a ordem de compilação, diretórios e *plugins*. O Maven faz o *download* de bibliotecas Java e dos seus *plugins* dinamicamente de um ou mais repositórios, e armazena-os numa área de cache local tornando mais fácil e ágil todo este processo [27].

5.5 Arquiteturas

5.5.1 Arquitetura de três camadas

O Projeto F assenta num sistema de três camadas. Este modelo consiste em separar a lógica de negócio com a interface do utilizador. Para isso, faz a separação entre a camada de apresentação, camada de negócio e camada de dados. Com esta divisão, os sistemas tornam-se mais flexíveis e independentes entre os componentes da aplicação, possibilitando alterar uma camada sem modificar as restantes. Como consequência

destas propriedades, esta arquitetura tornou-se padrão nos sistemas *web*.

A camada de apresentação engloba todas as classes da aplicação com os quais o utilizador vai interagir. A camada lógica é responsável por todas as classes que prestam o serviço e definem as regras de negócio. Por fim, a camada de dados é responsável pela persistência dos dados e pela comunicação com a camada de negócio [28].

5.5.2 Padrão MVC

O modelo MVC (*Model-View-Controller*) fornece uma forma de dividir as funcionalidades de manutenção e apresentação dos dados na aplicação. Como o próprio nome indica, consiste em separar uma aplicação em três camadas: modelo, visualização e controlo.

O modelo é responsável por representar os objetos de negócio, manter o estado da aplicação e fornecer ao controlador o acesso aos dados. A camada de visualização corresponde à interface que será apresentada ao utilizador, sendo responsável por definir a forma como os dados serão apresentados e em encaminhar as ações para o controlador. A camada de controlo é responsável por fazer a ligação entre a visualização e o modelo [29].

Esta distribuição visa aumentar a flexibilidade e a reutilização de código. Este torna-se mais organizado e a sua manutenção mais simples e menos dispendiosa [29].

5.5.3 Comparação

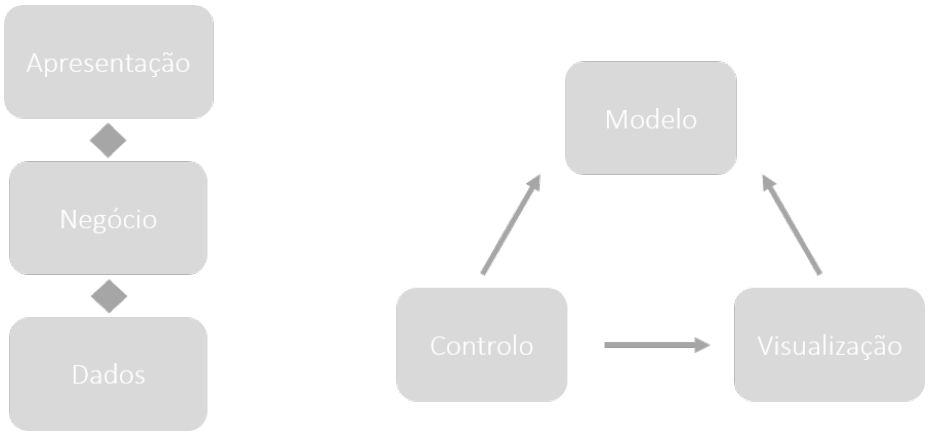


Figura 5.1: Arquitetura de 3 camadas e Padrão MVC

Arquitetura de 3 camadas	Padrão MVC
Relacionado com a arquitetura do sistema	Relacionado com a arquitetura da aplicação e de como os componentes comunicam
Comunicação bidirecional e passando sempre pela camada intermediaria	Comunicação unidirecional
Concetualmente é linear	Concetualmente é triangular

Tabela 5.1: Arquitetura 3 camadas vs Padrão MVC

As duas arquiteturas podem complementar-se e coexistir no mesmo sistema. Caso se pretende uma divisão mais estruturada, é necessária a separação entre a camada de lógica de negócio e o controlo da interface. Deste modo, a camada de controlo e de visualização seriam aplicados na camada da apresentação de uma arquitetura de 3 camadas, e o modelo na camada de dados e de negócio.

5.6 JavaServer Faces

O JavaServer Faces [30] é uma *framework* de aplicações *web* baseada em Java que incorpora características de uma arquitetura Model-View-Controller (MVC) e de um modelo de interfaces gráficas baseadas em eventos. O Projeto F foi desenvolvido através da implementação Mojarra e usa o Primefaces como biblioteca de componentes.

5.6.1 Mojarra

O Mojarra [30], também conhecido como JSF RI, é a implementação de referência de JavaServer Faces desenvolvida pela Oracle. Trata-se da implementação de JSF mais usada estando incluída no próprio GlassFish.

5.6.2 PrimeFaces

O PrimeFaces [31] trata-se de uma biblioteca *open source* baseada em CSS e Javascript desenvolvida para versões a partir do JSF 2.0. Contém um volume alargado de componentes para criar interfaces mais agradáveis e de uma forma simplificada para aplicações web e extensível para aplicações *mobile*.

Desenvolvido pela Prime Technology, empresa sediada na Turquia, é uma das *frameworks* mais utilizadas por parte dos programadores de Java [32]. Segundo os seus

criadores, um dos grandes fatores de diferenciação é o facto de usarem o PrimeFaces em todos os projetos de *front end* que desenvolvem o que lhes permite ter uma visão alargada relativamente à usabilidade e simplicidade, assim como detetar e corrigir rapidamente a existência de *bugs* [32]. A sua comunidade oferece um grande suporte, desde fóruns, livros, exemplos, entre outros.

5.7 Spring

Spring [33] é uma *framework open source*, criada por Rod Johnson, tendo lançado a sua primeira versão em Junho de 2003. Esta *framework* permite criar aplicações de forma eficiente e rápida, oferecendo ainda um ambiente onde é possível efetuar testes a aplicação de forma muito simples.

Apresenta ainda um bom nível de estabilidade, modularidade e extensibilidade. Por esses motivos, desde o seu lançamento tem vindo a ganhar cada vez mais utilizadores, e atualmente estima-se que seja o *framework* mais usado a nível mundial [33].

Uma aplicação que integra o Spring, será executada sob o *container* do Spring. Este oferece essencialmente o IoC (inversão do controlo) e ID (injeção de dependências). Outro aspeto importante, é que esse *container*, oferece outro paradigma, programação orientada a aspetos. Este novo paradigma de programação, permite essencialmente que o código seja mais limpo, e de melhor interpretação, assim como, tenha uma maior facilidade de manutenção do código.

5.8 Hibernate

O Hibernate [34] é uma *framework open source* desenvolvida pela Red Hat que consiste em construir o mapeamento objeto-relacional (ORM).

A flexibilidade encontrada num modelo relacional influencia a utilização do conceito de persistência, onde os dados são guardados nas tabelas da base de dados e convertidos para os recursos do Java [35].

A comunicação com a base de dados pode ser feita com o recurso a um *driver* JDBC (*Java Database Connectivity*). Trata-se de uma especificação de como a linguagem de programação Java irá comunicar com a base de dados, levando os gerenciadores de bases de dados a criar os seus *drivers* específicos. A classe *Driver Manager* é a

responsável por determinar qual o *driver* que realiza essa interligação [35].

5.8.1 HQL

O *Hibernate Query Language* (HQL) é uma linguagem orientada a objetos para realizar consultas à base de dados. A sua sintaxe assemelha-se ao *Structured Query Language* (SQL), com a principal diferença de aceder ao nome e propriedades da classe em detrimento do nome da tabela e das suas colunas graças à persistência do Hibernate [36].

5.8.2 Criteria

A API Criteria é uma alternativa para manipular objetos através de dados presentes nas tabelas de um sistema de base de dados relacional. Permite criar consultas programaticamente podendo aplicar regras de filtragem e condições lógicas [37].

Em termos de legibilidade, para consultas mais complexas, pode tornar-se mais difícil a sua compreensão quando comparado como o HQL.

Capítulo 6

Implementação

O licenciamento das ferramentas que constituem o Activiti oferecem a possibilidade de as alterar, adaptar e utilizar de acordo com as nossas necessidades. A ideia inicial consistia em tirar o máximo partido da aplicação *web* e do modelador do Activiti e usa-los como parte integrante do Projeto F.

Como foi possível visualizar na tabela 4.1, a *framework* da aplicação *web* do Activti não coincide com a que é usada no Projeto F. Dadas as dificuldades de integrar as diferentes *frameworks*, Vaddin com JSF, surgiu a necessidade de procurar uma alternativa.

Assim, colocou-se a hipótese de desenvolver, de raiz, uma interface gráfica em JSF para o desenho e modelação dos processos de negócio que fosse suportada pela lógica do motor do Activiti.

6.1 Gestão de departamentos

A gestão de departamentos foi um módulo desenvolvido para o Projeto F que permitiu explorar os componentes do PrimeFaces e determinar a viabilidade do seu uso para o problema descrito anteriormente.

O encaminhamento de documentos ocorre primariamente ao nível dos departamentos. A mesma realidade para a consulta dos documentos. Os documentos que são possíveis pesquisar são aqueles que foram registados ou que passaram pelas unidades orgânicas que se encontram chefiadas pelo departamento em questão. Em alguns cenários é pretendido que esta consulta tenha em consideração a estrutura orgânica hierarquizada,

ou seja, que os departamentos estejam organizados em termos de chefias. É neste contexto que entra o módulo que desenvolvemos: puder definir que departamento "está acima"de que departamento.

Assim, se pertenço a um departamento e sou a sua chefia, vou puder consultar todos os documentos desse departamento e todos os que se encontram abaixo, definido pela hierarquia.

Este módulo, tem como objetivo possibilitar ao utilizador efetuar a gestão de departamentos, oferecendo opções como criar, editar e visualizar departamentos e, acima de tudo, através de uma interface gráfica, poder introduzir e remover conexões entre os departamentos definindo a hierarquia de uma dada organização.

6.1.1 Visão geral

Na figura 6.3 é possível visualizar a interface criada para listar os departamentos. Esta foi a primeira interação com o JSF e respetivos componentes do PrimeFaces assim como à lógica da arquitetura de 3 camadas. A informação respeitante aos departamentos, mantida na base de dados Oracle, é obtida usando consultas HQL para desenvolver as classes persistentes.

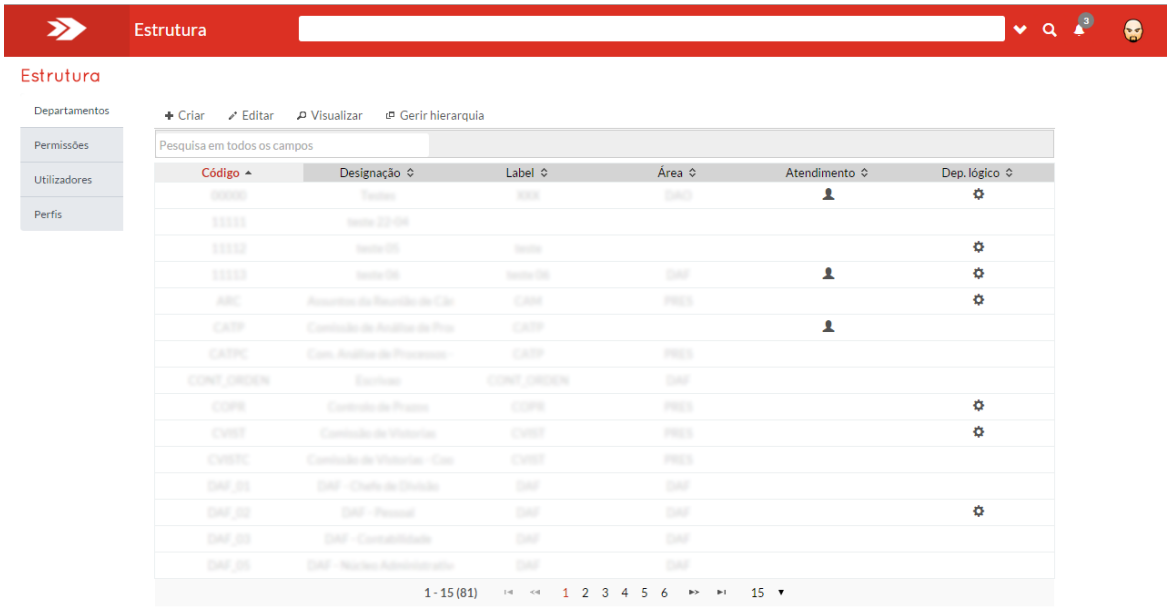


Figura 6.1: Interface da listagem de departamentos

A listagem que apresenta os dados contém uma série de funcionalidades concedidas pelo JSF e PrimeFaces, desde filtragem pelo campo de pesquisa, ordenação, duplo

clique para edição, entre outros. As opções de criar, editar e visualizar departamento correspondem à mesma página XHTML variando apenas o conteúdo apresentado com base na escolha selecionada.

Ao selecionar a opção "Visualizar", é possível verificar a informação detalhada desse departamento (figura 6.2). Nessa interface, observa-se um diagrama que permite verificar quais os departamentos que são diretamente chefiados assim como a sua chefia. A criação desse diagrama será explicada em detalhe na próxima secção, no entanto, é importante referir que sua apresentação foi inserida num componente *Composite*. Este, permite facilmente reutilizar componentes e evitar a duplicação de código, alterando apenas os parâmetros de visualização do diagrama em questão.

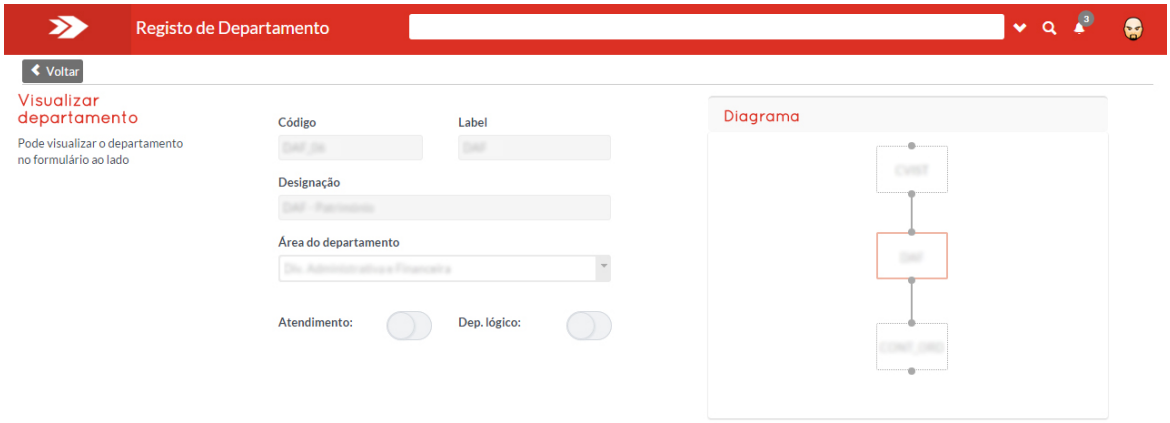


Figura 6.2: Interface da visualização de um departamento

6.1.2 Gestão da hierarquia

Esta interface surge ao selecionar a opção "Gerir hierarquia"e foi desenvolvida para o utilizador poder criar, eliminar ou alterar relações hierárquicas dos departamentos.

Através do departamento selecionado e do número de níveis hierárquicos que se pretende visualizar, é gerada uma representação gráfica dos departamentos, identificando o departamento atual e os seus departamentos hierarquicamente supervisores e inferiores.

O componente *Diagram* do PrimeFaces facilita o processo de criação dos elementos e das suas respetivas ligações. No seguinte exemplo podemos analisar uma possível solução para contruir um departamento:

Element departmento = new Element(nome, dimensãoX, dimmensãoY);

model.addElement(departmento);

Na figura 6.3 podemos verificar que cada elemento contém dois círculos nas suas extremidades. Estes, são os responsáveis pela interligação entre os departamentos.

departmento.addEndPoint(createEndPoint(EndPointAnchor.TOP));

departmento.addEndPoint(createEndPoint(EndPointAnchor.BOTTOM));

A ideia passou por tornar o *EndPoint top* responsável por relacionar o departamento com os seus superiores hierárquicos. Este *EndPoint* só permite enviar ligações. Por outro lado, o *EndPoint bottom* fica responsável por relacionar os inferiores hierárquicos com o departamento, só permitindo receber ligações. Esta abordagem torna mais percetível perceber "quem chefia quem".

Essa representação pode ser vista como uma árvore que é construída por intermédio da relação Departamento - Departamento Pai (hierarquicamente superior) mantida na base de dados. Mais uma vez, o componente *Diagram* do PrimeFaces possibilita facilmente criar essa ligação visual:

model.connect(new Connection(departamento, departamentoPai));

Foi desenvolvido um método para inserir no grafo os departamentos e as suas ligações executando-o recursivamente até atingir o número de níveis especificado.

6.1.2.1 Desenho do diagrama

Tendo o grafo gerado, foi necessário desenvolver um algoritmo para determina a posição em que cada um dos departamentos será colocado, incrementando (ou decrementando) o valor do eixo das ordenadas (para representar níveis hierarquicamente diferentes) e das abcissas (para representar separadamente departamentos no mesmo nível hierárquico). A raiz (departamento inicial), destacado com um contorno a vermelho, é a base para essa construção.

O diagrama onde é apresentada a árvore tem um limite inferior horizontal. Supondo o caso de tentar criar um departamento que se encontre no mesmo nível hierárquico que o anterior. A posição do novo departamento seria obtida através de subtração da posição do departamento anterior relativamente ao eixo das abcissas. No entanto, se o valor desse departamento já tivesse atingido o 0 (valor mínimo), os departamentos ficariam graficamente sobrepostos.

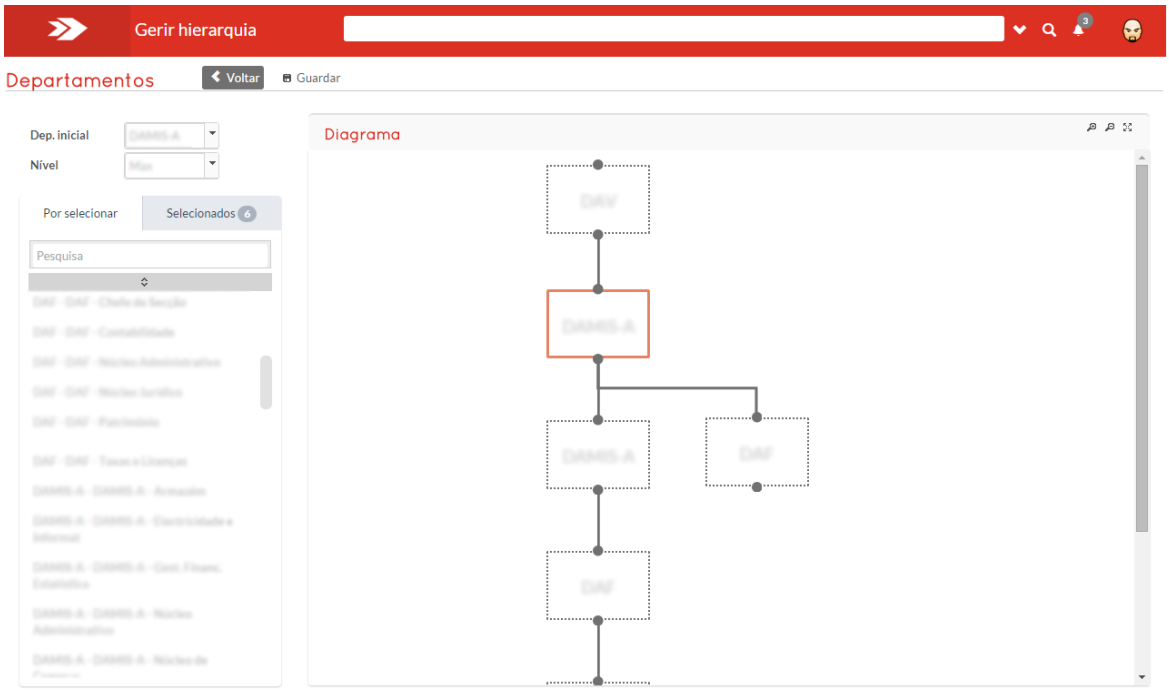


Figura 6.3: Interface da gestão de hierarquia

Era essencial definir a posição ótima da raiz que evitasse esta situação. Como a árvore é construída dinamicamente, o seu posicionamento tem que ser constantemente modificado.

A pesquisa em largura, também conhecido por *Breath-First Search* (BFS), solucionou este problema. Trata-se de um algoritmo utilizado para percorrer sistematicamente os vértices de um grafo dirigido ou não dirigido. Dado um grafo G=(V,E), sendo V o número de vértices, E o número de arestas e uma raiz S, esta pesquisa explora todos os vértices atingíveis a partir de S. Este processo é repetidamente executado utilizando uma estrutura de dados FIFO (*First In First Out*) para garantir que os vértices são utilizados como raiz pela ordem com que são encontrados [38].

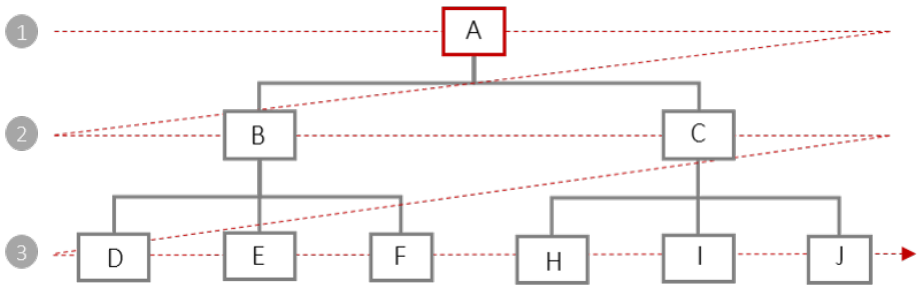


Figura 6.4: Ordem de execução do algoritmo de pesquisa em largura

A figura 6.4 demonstra a ordem de pesquisa a realizar numa árvore utilizando uma pesquisa em largura. A pesquisa inicia-se pelo nó A (raiz) e introduz na fila (FIFO) os seus respetivos filhos, ficando os valores [B,C]. Posto isto, o nó B é retirado da fila e explorados os seus filhos, atualizando o conteúdo da fila [C,D,E,F]. No próximo passo, a partir de C torna-se atingível o H, I e J ficando a fila com o seguinte valor: [D,E,F,H,I,J]. Este procedimento termina quando explorados todos os vértices da árvore obtendo a seguinte ordem de execução A-B-C-D-E-F-H-I-J. Esta estratégia de pesquisa está diretamente relacionada com os conceitos de distância e de caminho mínimo [38].

Como temos apenas uma raiz, podemos introduzir o conceito de níveis à pesquisa. Sendo A a raiz e estando no nível 1, os seus filhos analisados posteriormente encontrar-se-ão no nível 2, e assim sucessivamente. Pensando no caso dos departamentos, podemos verificar que os departamentos D, E e F que estão no nível 3, têm o departamento B como chefia que se encontra no nível 2. Este por sua vez, tem o departamento A (departamento inicial) como superior hierárquico que se encontra no nível 1.

Determinando o número de departamentos associado a cada nível, conseguimos definir o valor posicional da raiz que impeça os seus filhos de exceder as dimensões do diagrama.

Há situações em que o conceito visual de níveis não pode ser implementado como no caso ilustrado na figura 6.5, em que um departamento tem como chefia um departamento que se encontra no mesmo nível hierárquico. No entanto, graças à abordagem utilizada para a ligação entre os departamentos, torna-se menos confuso o seu entendimento (A é superior hierárquico de B).

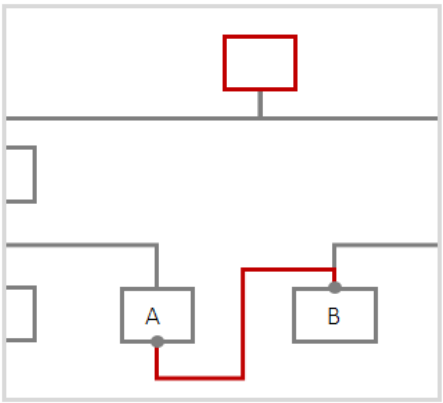


Figura 6.5: Ligação de departamentos no mesmo nível hierárquico

Para árvores de grandes dimensões, essa informação foi também útil para auxiliar o posicionamento do *scroll* horizontal do diagrama. Por defeito, apresenta o conteúdo que esteja mais à esquerda. Com recurso a um método em Javascript, foi possível ajustar o diagrama para o enquadramento da raiz.

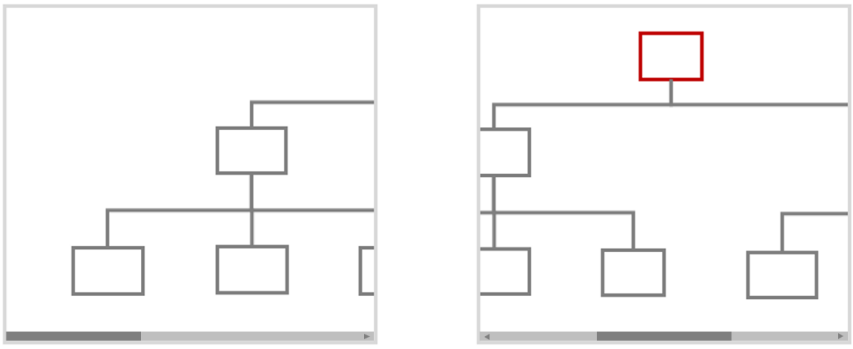


Figura 6.6: Posicionamento padrão (à esquerda) e o posicionamento após aplicar o método para centrar (à direita)

6.1.2.2 Criação de ligações

Como é possível visualizar na figura 6.3, a interface apresenta à sua esquerda um menu com dois separadores. Um deles, corresponde aos departamentos selecionados, que lista todos os departamentos com base nos critérios definidos (Departamento inicial e número de níveis). Ao clicar sobre o seu nome, permite evidenciar a sua localização no diagrama à sua direita alterando a sua cor de fundo.

O outro separador, "Por selecionar", lista os departamentos que:

1. Não têm qualquer tipo de conexões;
2. Fazem parte da hierarquia do departamento selecionado mas ainda não foram abrangidos pelo nível definido.

Neste separador é possível arrastar o departamento para o diagrama com auxilio às funcionalidades de *drag and drop* do PrimeFaces. Caso o departamento movido tenha alguma relação hierárquica com os restantes, é gerada automaticamente a respetiva ligação.

Uma das fragilidades do componente *Diagram* era o facto de não guardar a posição do departamento arrastado, isto é, ao mover um novo departamento para o diagrama,

ambos tomariam a ultima posição selecionada. Através de Javascript foi criado um método que relacionasse o departamento com a sua posição (x,y) no diagrama, sendo atualizada sempre que o departamento é movido.

Ao inserir uma nova conexão entre dois departamentos, um dos pontos a ter em consideração é impossibilitar a criação de ciclos, ou seja, ao criar uma conexão entre um Departamento A - Departamento B (sendo A chefiado por B) é necessário garantir que nenhum superiores hierárquicos do Departamento B tem como pai (superior hierárquico) o departamento A. Deste modo, na camada de negócio, foi implementado um método recursivo que validasse se o Departamento B e a sua estrutura superior hierárquica têm o Departamento A como chefia. Em caso de insucesso, a ligação era impossibilitada e o utilizador notificado.

As informações referentes à inserção ou remoção das conexões são guardadas em listas auxiliares. Só aquando do clique no botão "Guardar"é que essa informação é guardada na base de dados. Esta estratégia minimiza o impacto de um possível erro do utilizador. Caso deseje cancelar as operações, basta sair do página atual que lhe aparecerá um menu a perguntar se pretende eliminar as alterações realizadas.

Para árvores de grandes dimensões, a análise visual da sua hierarquia pode tornar-se mais difícil. Era importante fornecer meios para o utilizador aumentar o espaço de visualização e facilitar o seu manuseamento. O PrimeFaces contém a opção de *zoom* para alguns dos seus componentes, nomeadamente para imagens e gráficos. No entanto, o componente usado para representar o diagrama não corresponde a nenhum dos enunciados.

Deste modo, foram desenvolvidas manualmente algumas opções de visualização recorrendo a eventos de Javascript e a propriedades de *Cascading Style Sheets* (CSS), tais como:

- *Zoom in* através do clique do botão;
- *Zoom out* através do clique do botão;
- *Full screen*;
- *Zoom in* e *Zoom out* através do movimento do *scroll*
- Ao arrastar na zona do diagrama, simular o comportamento do *scroll*, tanto na vertical como na horizontal.

Este módulo foi desenvolvido com sucesso e desempenha as funções para o qual foi idealizado. Criar um modelador com estas características apresentaria como vantagens o facto de estarmos a utilizar a mesma tecnologia e podermos criar e adaptar de acordo com as necessidades do Projeto F para a transação dos documentos.

O componente usado para criar as conexões entre os departamentos, o *Diagram*, foi disponibilizado na ultima versão do PrimeFaces. Talvez o facto de estar numa fase prematura explique as suas debilidades. O número reduzido de funcionalidades e a falta de fluidez evidenciada torna limitativa a sua expansão para um projeto de grande dimensão como o Activiti Modeler, o módulo de desenho do Activiti.

6.2 Integração

Após a realização do módulo para a gestão de departamentos, concluiu-se que com as ferramentas disponíveis dificilmente atingiríamos a mesma qualidade do Activiti Modeler. Impossibilitada a hipótese de integrar as tecnologias distintas, a opção passou por executar os dois projetos, Projeto F e Activiti, em simultâneo mas separadamente.

O primeiro passo correspondeu à importação do motor do Activiti para o Projeto F. Por se tratar de um projeto Maven, bastou incluir de dependência do *activiti-engine* e o seu respetivo repositório. Ao compilar o projeto, automaticamente é feito o *download* de todas as bibliotecas necessárias.

Deste modo conseguimos criar a definição do objeto *processEngine*. Este, é o que nos dá acesso à vasta API do Activiti. A sua configuração foi realizada com recurso a um driver JDBC para criar a comunicação com a sua base de dados.

6.2.1 Módulo de desenho

Foi desenvolvida uma interface no Projeto F que permitisse efetuar a gestão dos modelos de processos. Esta interface dispõe de um menu para criar novos modelos e apresenta uma tabela que lista os modelos existentes, oferecendo opções para alterar e eliminar esses mesmos modelos. Em termos de visualização, tem um filtro de pesquisa que pode ser feita através de qualquer um dos campos da tabela e, todos eles, podem ser ordenados de forma ascendente ou descendente.

A informação relativa aos modelos é mantida numa das tabelas da base de dados do

Activiti, nomeadamente na tabela ACT_RE_MODEL (*Activiti repository model*). O seu acesso foi realizado através da API, onde a interface *RepositoryService* contém os métodos necessários para o efeito.

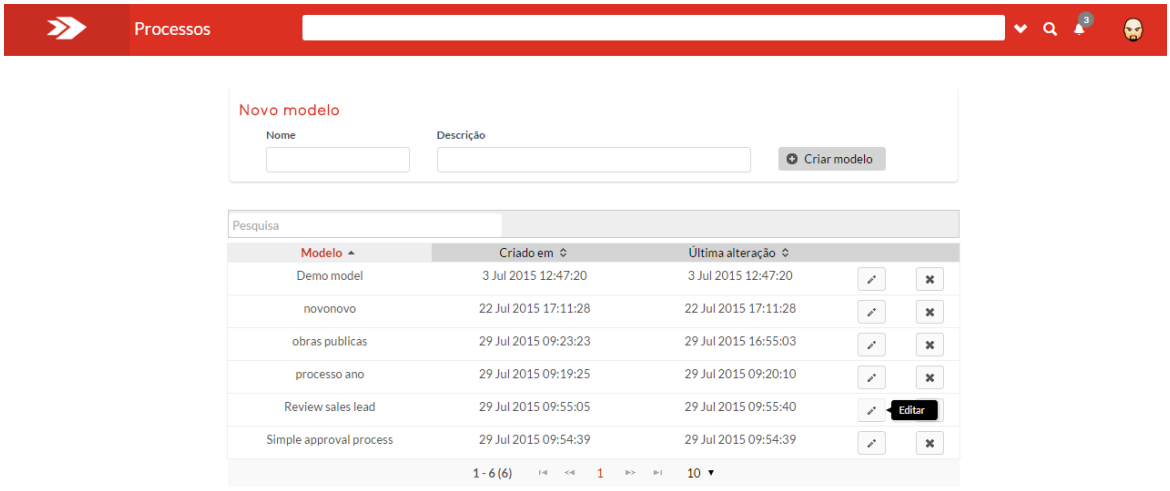


Figura 6.7: Interface da gestão dos modelos

Ao aceder ao Activiti Explorer, deparamo-nos com uma página para o utilizador efetuar o seu *login*. O objetivo passava por transitar de uma aplicação para outra sem a necessidade do utilizador voltar a escrever o seu *login* ao aceder ao Activiti Explorer, procurando tornar este procedimento o mais transparente possível.

Para isso, ao criar um novo utilizador no Projeto F, foi criada uma função que introduzisse também essa informação na tabela de utilizadores do Activiti.

Apesar das várias funcionalidades e da facilidade de uso, a arquitetura REST do Activiti não permite que seja feita a autenticação de utilizadores utilizando esse recurso. Também a sua API não contém métodos para autenticar um utilizador. Por se tratarem de aplicações distintas, o uso de *Cookies* e de *Session Storage* foi uma impossibilidade visto que o *browser* ao aceder a outra aplicação limpa o seu conteúdo.

Assim, a solução encontrada passou por enviar no próprio *Uniform Resource Locator* (URL) o valor do utilizador, a sua *password* e o identificador do modelo que queremos visualizar à semelhança do que acontece num método de comunicação HTTP GET. Como medida de segurança, esses valores foram cifrados do lado do Projeto F usando o algoritmo *Advanced Encryption Standard* (AES) [39]. O único ataque conhecido a este algoritmo requer um tempo computacionalmente impraticável para que seja possível comprometer o sistema.

Do lado do Activiti, o módulo *activiti-explorer* do seu motor contém uma classe,

designada por *ExplorerApp*, que controla a necessidade de apresentação da página inicial de *login* baseando-se no facto de conter, ou não, um utilizador autenticado. Essa informação está presente no método *authenticate* da classe *DefaultLoginHandler*. Por defeito, esse método retorna o valor *null* o que implica a inexistência de algum utilizador. Este é o local onde a lógica de *login* único tem de ser implementada.

Através do conteúdo do URL, esses dados são decifrados com o recurso ao algoritmo AES e usados para criar e autenticar um utilizador com o respetivo *username* e *password*, originando assim uma "ponte"entre as duas aplicações.

Como foi dito anteriormente, no URL é também enviado o identificador do modelo. É importante uma vez que o objetivo é aceder diretamente ao modelador e não obrigar o utilizador a executar uma sequência de passos dentro do Activiti Explorer para o conseguir. Esse identificador permite, após o processo de autenticação, definir qual o modelo que queremos apresentar e redirecionar para a página do Activiti Modeler com o modelo correspondente.

A criação de um novo modelo e a sua alteração têm um funcionamento semelhante. Ao criar o modelo, por intermédio do método *repositoryService.newModel()*, é-lhe atribuído um identificador sequencial. Após agregar o valor do nome e da descrição, o modelo é guardado na base de dados. É então executada a função que permite aceder ao modelador do Activiti mas, neste caso, é utilizado o identificador gerado para o novo modelo.

A figura 6.7 e a figura 6.8 permitem visualizar a interação que existe entre as duas aplicações quando a opção de editar o processo "*Review sales lead*"é selecionado.

Cada documento tem que ter um processo associado. Deste modo, era necessário relacionar um tipo de documento com um modelo de processos de negócio. Para isso, na tabela correspondente ao tipo de documento do Projeto F foi adicionado um campo responsável por armazenar o código identificador do modelo, criando assim uma ligação entre tipo de documento - modelo de processos.

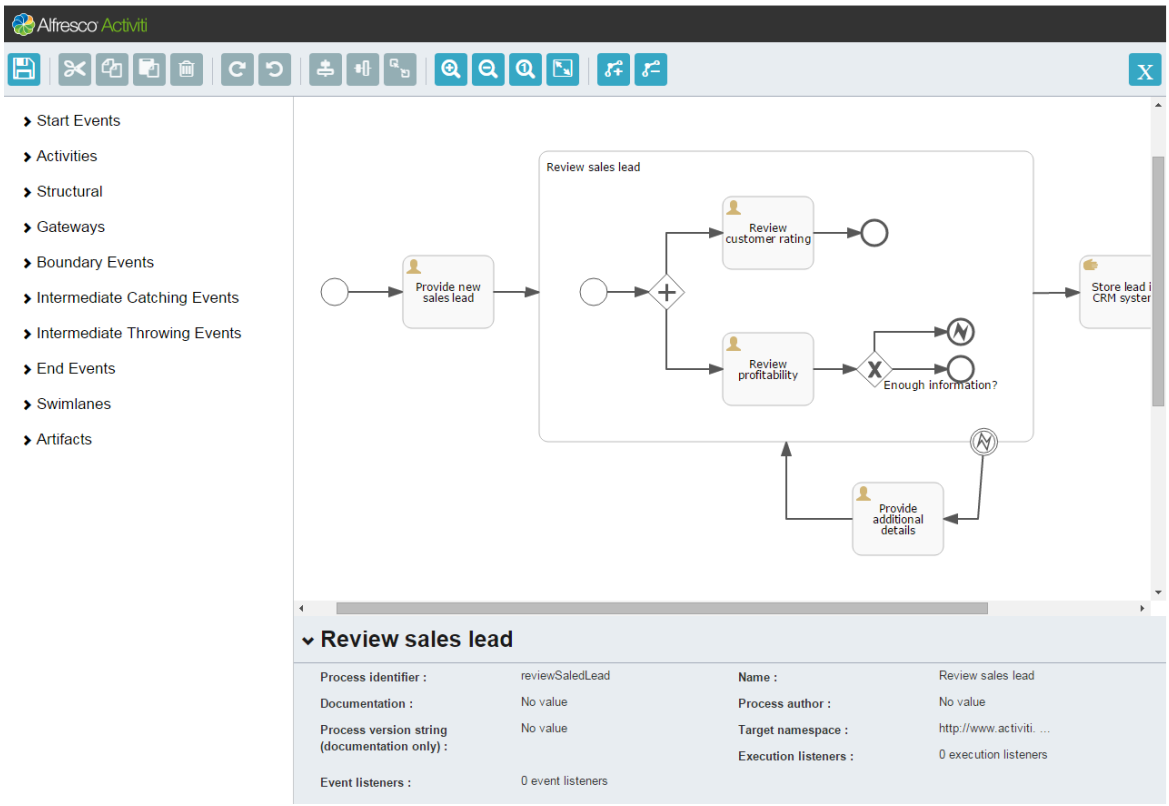


Figura 6.8: Interface do modelador do Activiti

6.2.2 Módulo de tarefas

Ao longo da execução dos processos de negócio, há tarefas que requerem interação humana para serem executadas. Este módulo foi desenvolvido para auxiliar o utilizador a consultar as tarefas que lhe estão associadas.

A tabela ACT_RU_TASK (*Activiti Runtime Task*) da base de dados do Activiti é responsável por armazenar as tarefas que estão ativas, ou seja, à espera de ser executadas. Utilizando a interface *TaskService* da API, obteve-se o conjunto das tarefas destinadas para um dado utilizador.

Foi construído um menu para apresentar a listagem de tarefas. Ao realizar o *login* no Projeto F, o utilizador é notificado sobre as suas tarefas pendentes, indicando:

- Nome da tarefa;
- Nome do processo;
- Data/hora de início da tarefa;

Figura 6.9: Interface das notificações de tarefas

Ao selecionar uma das tarefas, a página é reencaminhada para o Activiti Explorer onde o utilizador a poderá consultar de forma detalhada e realizar as funções associadas.

Com base no exemplo descrito anteriormente do envio de uma versão do relatório de estágio do aluno para o seu orientador, a tarefa a realizar pelo aluno seria semelhante à que é visível na figura 6.10.

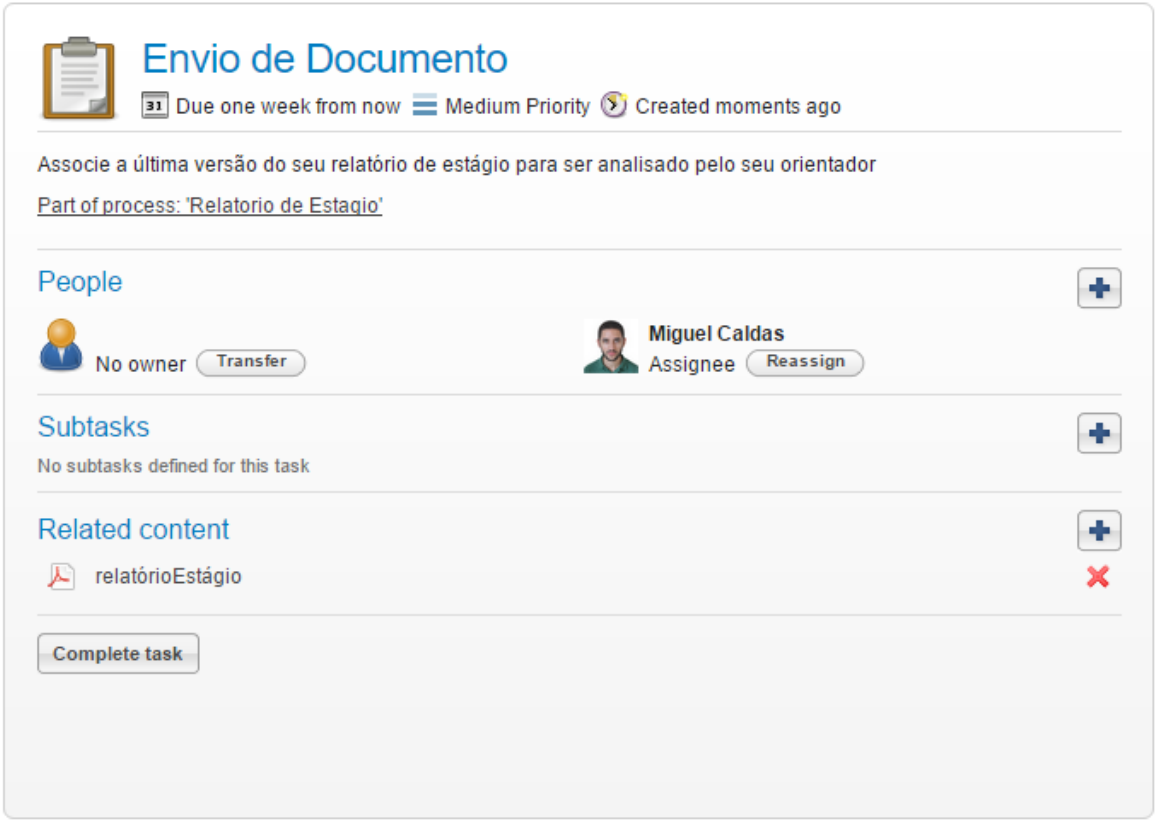


Figura 6.10: Interface de consulta de tarefas do Activiti Explorer

Capítulo 7

Conclusão e trabalho futuro

Este relatório descreve o trabalho realizado ao longo dos seis meses de estágio na ANO Software. O desafio proposto consistiu no desenvolvimento e integração de um motor de BPM no Projeto F de forma permitir a criação processos estruturados com regras de encaminhamento previamente definidas.

A sua completude não foi alcançada, mas, foram percecionados os pontos fortes e fracos do projeto, auxiliando assim a sua desenvoltura até este se encontrar conforme o idealizado.

A inexistência de uma ferramenta de BPM em JavaServer Faces deu origem a dúvidas e atrasos sobre a abordagem mais adequada a utilizar. A gestão de departamentos foi um módulo desenvolvido com sucesso para o Projeto F que, para além das suas funcionalidades, permitiu evidenciar as limitações gráficas do PrimeFaces e excluir a sua utilização para a criação de um modelador BPM.

Neste momento, o processo de integração com o Activiti contém as seguintes funções:

1. ligação com o Activiti Modeler para modelar processos de negócio;
2. edição e remoção de modelos;
3. associação de modelos a tipos de documentos;
4. notificação de tarefas;
5. ligação com o Activiti Explorer para executar tarefas;

Desta forma, para o Projeto F ter todas as funcionalidades inicialmente pretendidas, são necessárias as seguintes implementações:

1. criar uma interface onde o utilizador executa as tarefas e as opções inerentes no próprio projeto - atualmente o utilizador acede ao Activiti Explorer para realizar as tarefas;
2. implementar as regras de negócio e um *workflow* com a lógica dos documentos criando assim um conceito de processo estruturado;
3. interligar o movimento dos documentos com a estrutura definida no *workflow*;
4. estender o módulo de desenho do Activiti, adequando-o para as necessidades da criação de *workflows* de documentos;

A realização deste estágio na ANO Software contribuiu de forma bastante positiva e enriquecedora para a minha aprendizagem pois tive a oportunidade iniciar a minha experiência num contexto empresarial e de colocar em prática os conhecimentos adquiridos ao longo da minha formação académica. Mais importante ainda é o fato de me ter possibilitado adquirir conhecimentos das diversas tecnologias envolvidas e dos desenvolvimentos atingidos neste trabalho serem resultados relevantes para a empresa, aos quais será dada continuidade no futuro.

Referências

[1] M. Weske, *Business Process Management: Concepts, Languages, Architectures*. Springer, 2nd ed., 2012.

[2] "Página oficial da ANO Software." `http://www.ano.pt/`. Acedido em 08-2015.

[3] "ISO 9001." `http://www.iso.org/iso/home.html`. Acedido em 08-2015.

[4] "ISO 27001." `https://www.27001.pt/index.html`. Acedido em 08-2015.

[5] J. Jeston and J. Nelis, *Business Process Management: Practical guidelines to sucessful implementations*. Springer, 3rd ed., 2013.

[6] T. Allweyer, *BPMN 2.0: Introduction to the Standard for Business Process*. Books on Demand, 2nd ed., 2009.

[7] "Business Process Model and Notation." `http://www.omg.org/spec/BPMN/2.0/PDF/`, 2011.

[8] "GNU General Public License, version 2." `http://www.gnu.org/licenses/old-licenses/gpl-2.0.html`. Acedido em 08-2015.

[9] "GNU General Public License, version 2.1." `http://www.gnu.org/licenses/old-licenses/lgpl-2.1.html`. Acedido em 08-2015.

[10] "Apache License, version 2.0." `http://www.apache.org/licenses/LICENSE-2.0`. Acedido em 08-2015.

[11] "MIT License." `http://opensource.org/licenses/MIT`. Acedido em 08-2015.

[12] "Página oficial do jBPM." `http://www.jbpm.org/`. Acedido em 08-2015.

[13] E. A. D. of the Systems and N. A. Center, *Guidelines for Implementation of REST*. 2011.

[14] "Página oficial do Activiti." `http://activiti.org/index.html`. Acedido em 08-2015.

[15] T. Rademakers, T. Baeyens, and J. Barrez, *Activiti in Action*. Manning Publications, 2012.

[16] "Activiti forum." `http://forums.activiti.org/`. Acedido em 08-2015.

[17] "Página oficial do Camunda." `https://camunda.com/`. Acedido em 08-2015.

[18] "Página oficial do Bonita." `http://www.bonitasoft.com/`. Acedido em 08-2015.

[19] "Stevie Awards - Bonita." `http://www.bonitasoft.com/content/stevier-award-best-new-product-year`. Acedido em 08-2015.

[20] "Activiti User Guide." `http://www.activiti.org/userguide/`. Acedido em 08-2015.

[21] "Linguagens de programção mais populares." `http://pplware.sapo.pt/informacao/top-10-linguagens-de-programacao-mais-populares/`. Acedido em 09-2015.

[22] "Página oficial do NetBeans." `https://netbeans.org/`. Acedido em 09-2015.

[23] "Página oficial do GlassFish." `https://glassfish.java.net/`. Acedido em 09-2015.

[24] "Página oficial do JavaScript." `https://www.javascript.com/`. Acedido em 09-2015.

[25] "Página oficial da Oracle." `https://www.oracle.com/database/index.html`. Acedido em 09-2015.

[26] "Oracle Database Online Documentation." `http://docs.oracle.com/en/`. Acedido em 09-2015.

[27] "Página oficial do Apache Maven." `https://maven.apache.org/`. Acedido em 09-2015.

[28] A. Kumaraswamipillai and S. Arulkumaran, *Java/J2EE Job Interview Companion*. 2009.

[29] J. P. F. Gorla and I. J. Foschini, *Arquitetura para Desenvolvimento Web Baseado em JSF 2.0 utilizando Padrões de Projeto*. 2013.

[30] "Página oficial do JSF." `https://www.javaserverfaces.java.net`. Acedido em 09-2015.

[31] "Página oficial do PrimeFaces." `https://www.primefaces.org`. Acedido em 09-2015.

[32] "Why PrimeFaces." `http://www.primefaces.org/whyprimefaces`. Acedido em 09-2015.

[33] "Página oficial do Spring." `http://projects.spring.io/spring-framework/`. Acedido em 09-2015.

[34] "Página oficial do Hibernate." `http://hibernate.org/`. Acedido em 09-2015.

[35] D. H. Luckow and A. A. de Melo, *Programação Java para a WEB*. novatec, 2010.

[36] "Hibernate - Query Language." `http://www.tutorialspoint.com/hibernate/hibernate\_query\_language.htm`. Acedido em 09-2015.

[37] "Hibernate - Criteria Queries." `http://www.tutorialspoint.com/hibernate/hibernate\_criteria\_queries.htm`. Acedido em 09-2015.

[38] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms*. The MIT Press, 3rd ed., 2009.

[39] F. I. P. S. Publication, *Announcing the ADVANCED ENCRYPTION STANDARD (AES)*. 2001.