PM

Modelação e Implementação de Software de Apoio a Processos de Logística

PROJETO DE MESTRADO

Magno José Gomes Andrade
MESTRADO EM ENGENHARIA INFORMÁTICA

UNIVERSIDADE da MADEIRA
A Nossa Universidade
www.uma.pt

maio | 2018

Modelação e Implementação de Software de Apoio a Processos de Logística

PROJETO DE MESTRADO

Magno José Gomes Andrade
MESTRADO EM ENGENHARIA INFORMÁTICA

ORIENTADOR
David Sardinha Andrade de Aveiro

CO-ORIENTADOR
Luís Nuno Brito Figueiroa Jardim Costa

Resumo

A importância da *internet* nas nossas vidas tem vindo a aumentar ao longo dos anos, pelo que a utilização de programas informáticos no nosso quotidiano é inevitável, o mesmo sucede-se com as organizações que para conseguirem sobreviver no mercado empresarial precisam de ferramentas que auxiliem os funcionários na tomada de decisão.

O crescimento da *internet* e o rápido desenvolvimento do mundo económico e financeiro das empresas tem como consequência uma grande procura de *software*.

Este projecto tem como objectivo o desenvolvimento de componentes de um protótipo que funciona como um sistema de informação incluindo um gestor de fluxo de processos, assim como a modelação de processos recolhidos da empresa ligada ao sector da logística usando a linguagem DEMO, para testes e execução dos mesmos no protótipo desenvolvido.

A primeira parte do documento tem uma introdução ao DEMO e explica as vantagens e desvantagens dos sistemas de Fluxo de Trabalho e o porquê do seu aparecimento. São comparadas aplicações semelhantes ao protótipo a ser desenvolvido actualmente existentes no mercado.

Um dos contributos principais foi o levantamento exaustivo dos processos de negócio da empresa, utilizando a linguagem DEMO.

Foram efectuadas análises aos resultados das entrevistas realizadas aos funcionários desta empresa, que constituiu o cenário prático deste projecto com o objectivo de desenhar e modelar os processos de negócio da empresa usando o DEMO.

Antes do desenvolvimento do protótipo foram estudadas várias soluções tecnológicas para determinar as melhores opções para o protótipo, tendo em conta as tendências tecnológicas do mercado de *software* e os requisitos funcionais e não funcionais obtidos.

Os componentes desenvolvidos ao longo deste projecto de mestrado inserem-se num protótipo parte de um projecto mais vasto, a ser realizado de forma colaborativa com outros colegas de licenciatura, mestrado e doutoramento, a trabalhar no Laboratório de Engenharia Organizacional do *Madeira Interactive Technologies Institute*.

O documento descreve sucintamente o protótipo global e componentes desenvolvidos pelos restantes colegas e detalha ao pormenor os componentes desenvolvidos pelo mestrando, bem como a aplicação deste com a modelação dos processos obtida na empresa de logística.

Palavras-chave

DEMO
Fluxo de Trabalho
Formulários
Processos
Protótipo
Tarefas

Abstract

The importance of the internet in our lives has been increasing over the years so much that the use of computer software in our daily lives is inevitable, the same happens with organizations that in order to survive in the business market, they need tools that help employees in their daily decisions.

The growth of the internet and the quick development of the economic and financial world of companies result in a great search for software.

This project aims to develop modules of a prototype that works as an information system including a process flow manager, as well as the modeling of processes collected from the company linked to the logistics sector using the DEMO language for testing and execution of the same in the developed prototype.

The first part of the paper has an introduction to DEMO and explains the advantages and disadvantages of workflow systems and why they appear. In the document similar applications that currently exist on the market were compared to the prototype developed.

One of the main contributions was the exhaustive survey of the company's business processes, using the DEMO language.

Analyzes were made to the employees interviews of this company, with the aim of designing and modelling the company’s business processes using DEMO.

Before the development of the prototype, several technological solutions were studied to determine the best options for the prototype, taking into consideration, the technological trends of the software market and the functional and non-functional requirements acquired.

The modules developed during this master's project are part of a prototype of a larger project, to be carried out in a collaborative way with other colleagues of degree, master's and PhD, working in the Laboratory of Organizational Engineering of the Madeira Interactive Technologies Institute.

The document briefly describes the overall prototype and modules developed by the other colleagues and explains in detail the modules developed by the master's degree, as well as the application in the logistics company which is part of this project.

Agradecimentos

Dedico esta secção para agradecer a algumas pessoas que contribuíram e ajudaram a ultrapassar as dificuldades, para que este projecto de mestrado tivesse decorrido da melhor forma.

Gostaria de começar por expressar o meu agradecimento ao orientador Professor David Aveiro, por toda a ajuda e orientação dada ao longo do ano.

Ao co-orientador Eng. Luís Costa, pela ajuda e contribuições prestadas e pela sua disponibilidade sempre que foi necessário em todas as fases do projecto.

A todos os funcionários da empresa, pelo excelente acolhimento e por me fazerem sentir em casa desde o primeiro momento.

Agradecer também aos colegas, Guilherme Neves, Maria Gonçalves e Duarte Pinto que desenvolveram alguns componentes do protótipo, pela ajuda e contribuição prestada.

E como não poderia deixar de ser, o meu maior agradecimento vai para os meus pais José Maria e Esmeralda e meus irmãos Rúben, Edgar e Antonela, pela forma como me ajudaram a superar todos os obstáculos que foram surgindo ao longo do caminho e me apoiaram nos dias mais difíceis.

Índice

Lista de Figuras

Acrónimos

API - Application Programming Interface
ATD – Actor Transaction Diagram
BPMN - Business Process Model and Notation
CSS – Cascading Style Sheets
DEMO - Design & Engineering Methodology for Organizations
HTML – Hypertext Markup Language
JSON - JavaScript Object Notation
MVC – Model-view-controller
OFD – Object Fact Diagram
OPL – Object Property List
ORM – Object-relational mapping
PDF – Portable Document Format
PHP – Hypertext Preprocessor
PIF – Performa Informa Forma
PSD – Process Structure Diagram
SQL - Structured Query Language
WYSIWYG - What You See Is What You Get
XHR – XMLHttpRequest

1. Introdução

O capítulo 1 apresenta as motivações, objectivos, definição do problema, empresa e a estrutura da dissertação de mestrado.

1.1. Motivação

Mais de metade dos projectos informáticos falham em cumprir satisfatoriamente as expectativas. Uma das principais causas identificadas em pesquisas científicas [1] [2] [3] [4] é um conhecimento insuficiente/inadequado da realidade organizacional a ser automatizada ou suportada por um sistema de informação (SI).

A disciplina de Engenharia Organizacional fornece métodos e ferramentas para colmatar esta falha. A área de logística nesta empresa tem um grande volume de trabalho e complexidade processual, com muitas especificidades a nível de logística dado o carácter muito particular da rede viária e a orografia da ilha, bem como integração específica com logística nacional e internacional. A solução informática que actualmente suporta os respectivos processos apresenta várias lacunas, não providenciando informação importante que tem de ser verificada manualmente. Há a motivação de utilização das modernas técnicas de Engenharia Organizacional para a criação de uma visão estratégica e global do negócio e suas especificidades, que permita depois um levantamento de requisitos sistemático e implementação de um sistema de informação que dê um suporte mais eficaz às necessidades de gestão e informação da empresa.

1.2. Objectivos

Os objectivos inicialmente propostos foram:

1. O desenvolvimento de uma plataforma exclusivamente dedicada à empresa ligada ao sector dos transportes, em que deveriam ser suportados vários módulos/processos, como por exemplo: Leitura de pedidos; Gestão de recursos; Apoio à decisão na atribuição de recursos aos pedidos; Monitorização e controlo de indicadores de desempenho da operação (com aplicação de filtros/critérios parametrizáveis); Gestão de reclamações; Gestão de pessoal específica para condutores; Integração com o actual sistema de facturação.

2. A utilização dos métodos disponíveis da área de investigação de Engenharia Organizacional, para a obtenção pormenorizada dos processos existentes nesta empresa de logística e posterior derivação e especificação dos requisitos fundamentais para a implementação de uma futura plataforma informática com os processos alcançados.

Entretanto a meio do desenvolvimento do projecto, o objectivo número um foi modificado e concluiu-se que seria mais benéfico a extensão de um protótipo já em desenvolvimento num

âmbito de projecto mais alargado no M-ITI, de nome DISME (*Dynamic Information System Modeler and Executer*).

Esta extensão do protótipo passou a ser o objectivo principal na realização desta tese.

O protótipo foi desenvolvido em conjunto com dois colegas de mestrado e um colega com bolsa de investigação.

A maior parte dos objectivos concretizados foi o desenvolvimento de vários componentes de especificação de processos usando a linguagem DEMO (*Design & Engineering Methodology for Organizations*), especialmente o Painel de Controlo que controla o fluxo de tarefas de um processo. Outros componentes foram desenvolvidos e estão explicados no Capítulo 5.

De salientar o objectivo número dois, que foi o levantamento exaustivo dos processos da empresa para posterior modelação utilizando os diagramas do DEMO.

1.3. Definição do Problema

Nesta secção é descrito o problema a ser resolvido pela utilização e aplicação dos métodos de Engenharia Organizacional para a construção da plataforma em causa.

A dimensão e a complexidade que estão presentes na realidade de uma empresa, por vezes tornam-se difíceis de gerir e consequentemente trazendo problemas na eficiência e funcionamento dos processos associados à mesma. Portanto, à medida que esta complexidade cresce, a criação de sistemas de informação também torna-se complexa pelos desafios que surgem na obtenção dos requisitos essenciais ao funcionamento do sistema. Como descrito anteriormente, a fraca e pouco cuidada recolha de requisitos faz com que os sistemas de informação implementados não cumpram realmente as necessidades reais. Assim sendo, os próprios sistemas falham na assistência à empresa para a elaboração de iniciativas de forma eficaz. As necessidades de pequenas mudanças são também comuns à medida que o negócio evolui, mas vão sendo adiadas pela necessidade de envolvimento de fornecedores externos ou colaboradores de TI, muitas vezes com pouca disponibilidade.

1.4. Empresa

Por razões de confidencialidade nesta subsecção apenas é apresentada a actividade empresarial da empresa à qual foi interveniente neste protótipo desenvolvido.

É uma empresa de origem madeirense, que está presente em várias áreas de negócio em Portugal, como também em outros países, estas áreas de negócio são por exemplo, transportes e energia.

O projecto a ser desenvolvido é destinado a uma destas empresas na área de negócio ligada ao transporte de mercadorias, em que se incluem por exemplo, o transporte de contentores, gás e de materiais diversos.

1.5. Estrutura da dissertação

O relatório está estruturado em 6 capítulos que vão desde a exposição teórica até a descrição da parte prática, que foi necessária para todo o desenvolvimento do protótipo. Para cada um dos capítulos é descrito sumariamente o que está apresentado em cada um deles.

1. **Introdução**
   - Na introdução é exibida as razões em termos de objectivos, motivação e os problemas a resolver no desenvolvimento do projecto, como também um breve resumo da empresa à qual foi realizada a recolha de requisitos.
2. **Estado de arte**
   - Contém uma introdução a teoria do DEMO e toda a exposição teórica sobre os sistemas de Fluxo de Trabalho no seu todo, como também as aplicações já existentes no mercado e que se enquadram nessa categoria incluindo comparações com o protótipo a ser desenvolvido.
3. **Especificação**
   - É apresentada a ideia geral para o protótipo a ser desenvolvido, os requisitos não funcionais e funcionais que o protótipo deve conter e é exibida a arquitectura a ser usada e respectiva explicação. Após a obtenção dos requisitos e a escolha da arquitectura são apresentadas várias soluções tecnológicas que se enquadram com a arquitectura seleccionada. A base de dados também está presente em formato de diagrama DEMO do tipo OFD.
4. **Análise da empresa**
   - Neste capítulo estão evidenciados todos os métodos utilizados para a recolha de uma descrição e funcionamento da empresa através de entrevistas, uma breve explicação do funcionamento de uma parte (processo dos transportes) dos diagramas DEMO construídos e as dificuldades encontradas ao longo da análise e modelação dos diagramas.
5. **Implementação**
   - O capítulo 5 engloba todos os componentes de especificação que foram implementados e os *plugins* utilizados no protótipo, com explicações e imagens que demonstram a funcionalidade de cada um e abrangendo as várias versões efectuadas até a versão final, juntamente com os problemas e soluções encontradas que envolvem o protótipo na generalidade.
6. **Conclusão**
   - No capítulo da conclusão estão indicados as lições que foram obtidas com o desenvolvimento deste projecto de mestrado, como uma reflexão sobre o projecto na sua globalidade e algumas sugestões para trabalho futuro do desenvolvimento do protótipo.

2. Estado de Arte

O capítulo Estado de Arte engloba uma introdução teórica da linguagem DEMO, como também os seus conceitos específicos e que são baseados na teoria-Ψ.

Inclui toda a exposição teórica sobre os sistemas de Fluxo de Trabalho, as suas funcionalidades gerais que normalmente contêm, as vantagens e desvantagens da utilização destes sistemas no ambiente de operação das empresas.

Por fim, a apresentação de várias plataformas que são consideradas sistemas de Fluxo de Trabalho e a sua comparação com as funcionalidades do protótipo a ser desenvolvido.

2.1. DEMO (*Design & Engineering Methodology for Organizations*)

Toda a secção relativa ao DEMO tem por base a referência [1].

A abordagem à ontologia de uma organização apresentada em seguida é a abordagem DEMO, que por sua vez está sustentada na teoria-Ψ. Seguindo a distinção feita nesta teoria entre as noções de função e construção de um sistema, ao conjunto de bens ou serviços que uma entidade fornece ao seu ambiente dá-se o nome de *negócio da organização*; isto representa a perspectiva de função. Às actividades colectivas de uma organização em que estes serviços são desempenhados ou entregues, incluindo os actores humanos que desempenham estas actividades, é dado o nome de *estrutura da organização*; isto representa a perspectiva de construção. As organizações são artefactos projectados e criados, tal como carros, aviões ou sistemas de informação. O que diferencia as organizações é o facto dos seus elementos activos serem seres humanos, mais especificamente os seres humanos na sua função social ou como sujeito.

2.1.1. Axioma da Transacção

Na teoria Ψ[1], estes sujeitos desempenham dois tipos de actos: actos de produção (*P-acts*) e actos de coordenação (*C-acts*). Ao desempenharem actos de produção os sujeitos contribuem para a criação de bens ou serviços que são disponibilizados ao ambiente. Um acto de produção pode ser material (como criar e transportar bens) ou imaterial (como decidir sobre a atribuição de um seguro ou vender bens). Ao realizar um acto de coordenação, os sujeitos entram em compromissos uns para com os outros (e cumprem estes compromissos) com a finalidade de serem realizados actos de produção. Exemplos de actos de coordenação são "pedido", "promessa" ou "rejeição". O efeito da realização de actos de coordenação é que tanto quem os realiza como o destinatário dos mesmos ficam envolvidos num compromisso tendo como objectivo a criação de um acto de produção.

[1] letra grega PSI, que por sua vez é um acrónimo para a expressão *Performance in Social Interaction*

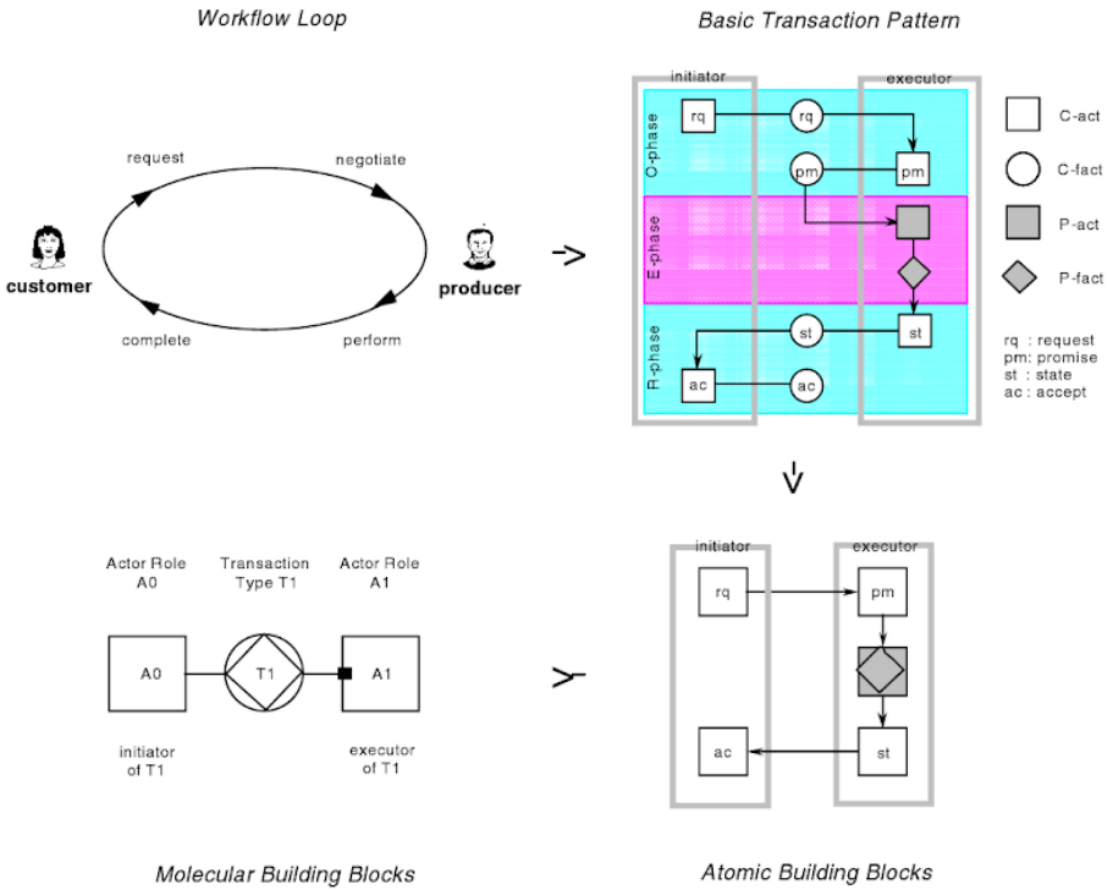

Figura 1 - Blocos da construção ontológica de uma organização.

Os actos de coordenação e produção ocorrem num padrão de coordenação genérico chamado de transacção. A Figura 1 - Canto Superior Direito - exibe o padrão básico de transacção como a formalização do Fluxo de Trabalho mais informal desenhado no canto superior esquerdo.

A transacção desenvolve-se em três fases: a fase do pedido (*Order-phase*), a fase da execução (*Execution-phase*) e a fase do resultado (*Result-phase*). Na fase do pedido o iniciador e o executor negoceiam para atingir um consenso sobre o facto de produção que o executor irá criar. Os principais actos de coordenação na fase do pedido são o pedido e a promessa. Na fase de execução o facto de produção é criado pelo executor. Na fase de resultado o iniciador e o executor negoceiam um consenso sobre o facto de produção produzido (que pode diferir do que foi realmente pedido). Os principais actos de coordenação na fase do resultado são a declaração e a aceitação.

Os termos “iniciador” e “executor” substituem os termos coloquiais “cliente” e “produtor”. Além disso estes termos fazem referência aos papéis de actor em vez dos sujeitos. Um papel de actor é definido como a autoridade e responsabilidade para ser o executor de um tipo de transacção. Os papéis de actor são desempenhados por sujeitos de tal forma que, um papel de actor pode ser desempenhado por vários sujeitos e um sujeito pode desempenhar vários papéis de actor. Em geral os papéis de actor não podem nem devem corresponder directamente em nome nem em conteúdo a cargos ou funções organizacionais. Como tal, o nome destes papéis serão,

sempre que possível, uma pequena variação do nome da transacção executada. Por exemplo num cenário de aluguer de carros, a transacção *entrega de viatura* será executada pelo *entregador de viatura* (nome alternativo: *responsável por entrega de viatura*). E, idealmente, um papel de actor deve ser desempenhado pelo mesmo sujeito para todos os actos sob a responsabilidade desse actor, nomeadamente o pedido e a aceitação pelo iniciador e a promessa e a declaração pelo executor. Desta forma há uma definição clara de responsabilidades que permite uma melhor e mais informada discussão sobre quais devem ser as reais necessidades em termos de funções ou cargos organizacionais e mapeamento de papéis de actor a estas funções e, por sua vez, mapeamento de pessoas concretas às funções definidas.

O percurso real de uma transacção pode ser mais complexo que o padrão básico de transacção representado na Figura 1. Isto é acomodado na teoria Ψ por duas extensões formais ao padrão básico. A primeira extensão cobre os casos em que os dois actores discordam. Por exemplo, em vez de prometer, o executor, pode responder a um pedido declinando-o, e em vez de aceitar, o iniciador, pode responder à declaração rejeitando-a. A este processo de "declinar" ou "rejeitar" chamamos de estados de discussão que poderão levar à paragem prematura de uma transacção e eventuais cancelamentos de outros actos e/ou transacções.

A segunda extensão consiste em adicionar quatro padrões de cancelamento, um para cada um dos principais passos da transacção (pedido, promessa, declaração e aceitação).

Para melhor compreender estes quatro padrões de cancelamento consideremos uma florista.

Padrão de cancelamento do pedido (canto superior esquerdo da Figura 2) - Logo após pedir um *bouquet* de tulipas vermelhas, constatamos o quão bonitas são as tulipas rosa, e dizemos à florista que queremos as tulipas rosa. A transacção actual é cancelada e é iniciada uma nova transacção.

Padrão de cancelamento da promessa (canto superior direito da Figura 2) – A florista descobre que o último *bouquet* de tulipas foi vendido pelo seu assistente a outro cliente. Ao cancelar a promessa, a florista declinou o acto do pedido. Não existe outra escolha razoável a não ser permitir o cancelamento. Deste estado de cancelamento a escolha mais razoável é desistir, pois fazer o mesmo pedido novamente não faz sentido.

Padrão de cancelamento da declaração (canto superior direito da Figura 2) – Depois de a florista colocar o *bouquet* de tulipas cor-de-rosa à nossa frente, reparamos que não estão frescas. A florista tem agora a opção de cancelar a declaração. Nesta altura podemos aceitar as flores e consequentemente aceitar o resultado da transacção ou aceitar que a florista cancele a declaração e nos faça um novo *bouquet*.

Padrão de cancelamento da aceitação (canto inferior esquerdo da Figura 2) – Ao sair da florista começamos a sentir remorso, de que o *bouquet* não é exactamente como deveria ser. Voltamos a entrar e explicamos o problema à florista, que o resolve, trocando o *bouquet* por outro (nota, isto inclui o cancelamento do acto de declaração por parte da florista).

O padrão básico junto com estas duas extensões é chamado o padrão completo de transacção e está representado na Figura 2. Este padrão é considerado como sendo uma lei socioeconómica: todas as transacções em todos os tipos de organização seguem caminhos de actos ao longo deste padrão.

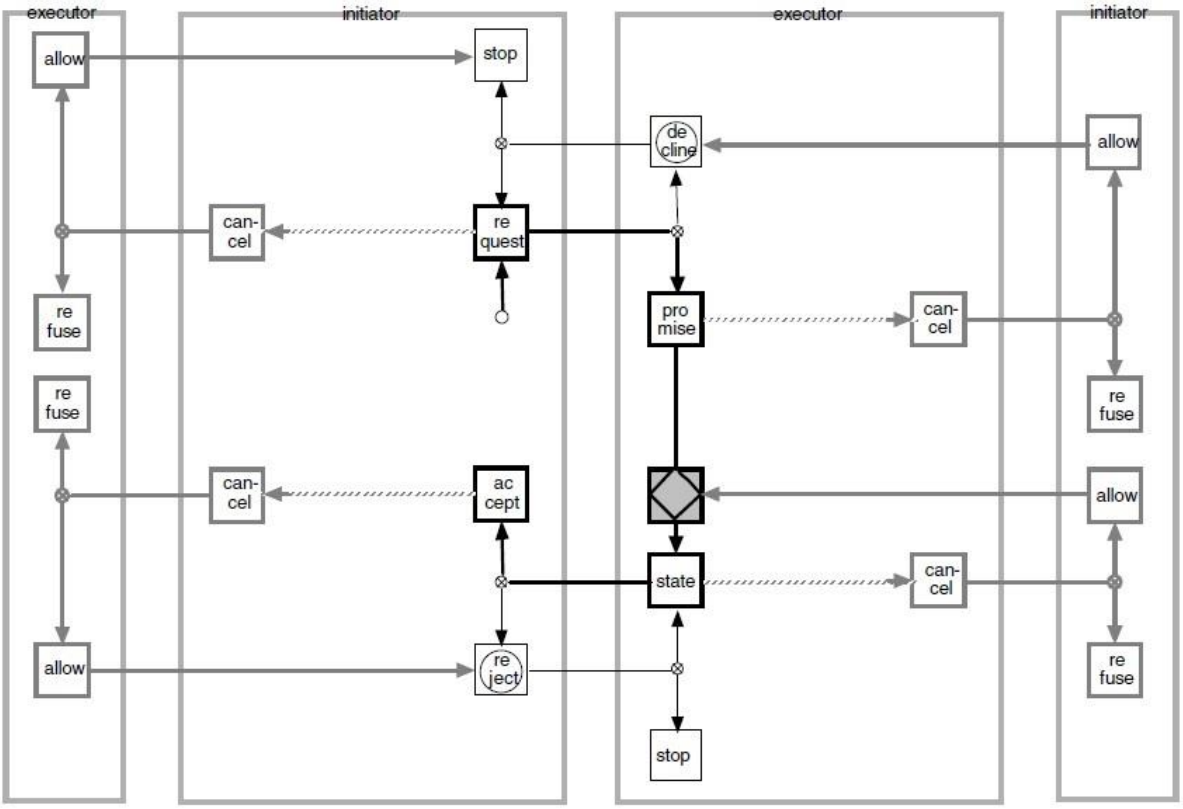

Figura 2 - Padrão de Transacção Completo.

Desempenhar um acto de coordenação não significa necessariamente comunicação verbal ou escrita. Em particular a promessa e a aceitação são muitas vezes realizadas pelo acenar ou outro acto não-verbal. Por exemplo numa padaria colocar o pão em cima do balcão em frente ao cliente conta como a realização da declaração. Além disso os actos de coordenação podem ser desempenhados tacitamente. Isto significa que não existe qualquer acto que conte como a realização de um acto de coordenação. Actos de coordenação tácitos devem ser compreendidos como concordância (implícita ou explicita) com o contrato que governa a transacção. Um exemplo, na mesma padaria, a promessa; pode ser realizada sem ser observável qualquer sinal.

Realizar tacitamente um acto de coordenação não significa contudo, que este conte menos do que realizar esse mesmo acto de forma explícita. Isto ganha maior relevância em casos de falha. Para evitar ao máximo possível mal-entendidos, é aconselhável a realização de todos os actos de coordenação explicitamente. Uma das principais vantagens das tecnologias de informação e comunicação modernas é que o seu custo é bastante reduzido; no comércio electrónico, como compra de artigos na *internet*, quase todos os actos de coordenação são realizados explicitamente.

No canto inferior direito da Figura 1 é mostrada a notação composta do padrão de transacção básico. Cada acto de coordenação e resultante fato de coordenação são representados por um símbolo composto; o mesmo se aplica aos actos de produção e respectivos factos de produção.

No canto inferior esquerdo o padrão de transacção completo é representado por um único símbolo, chamado de transacção; este consiste num diamante (representando a produção) embebido num disco (representando a coordenação). Tipos de transacção e papéis de actor são as

unidades moleculares da construção de processos de negócio e de organização, sendo os passos da transacção os átomos desta construção.

A primeira abstracção feita pela teoria Ψ para se chegar ao modelo ontológico de uma organização é a aplicação do axioma da transacção, resultando numa enorme redução da complexidade, cerca de 70%, a nível de documentação.

2.1.2. Axioma da Distinção

Para ser totalmente entendida a essência da operação das organizações, em seguida são introduzidas as distinções entre três habilidades humanas, que são exercidas tanto em actos de coordenação como em actos de produção (Figura 3).

No que toca à coordenação a habilidade *forma* diz respeito ao formato em que é transmitida a informação (falar, ouvir), a capacidade *informa* diz respeito aos aspectos do entendimento do conteúdo (formular, interpretar) e a habilidade *performa* corresponde ao envolvimento num compromisso (expor compromisso, invocar compromisso).

Figura 3 - Três capacidades humanas.

Quanto à produção a habilidade *forma* diz respeito à produção documental ou datalógica (armazenar, copiar), a habilidade *informa* diz respeito à produção informacional ou infológica (deduzir, computar, calcular) e a *performa* diz respeito à produção essencial numa organização (criar, decidir, julgar), por isso também chamada de produção ontológica.

A segunda abstracção feita na teoria Ψ, com o axioma da distinção, faz com que para se chegar ao modelo ontológico de uma organização apenas seja considerada a habilidade *performa* no que toca à produção, abstraindo-se assim dos actos de produção a nível datalógico e infológico. Isto resulta numa segunda grande redução de complexidade, também estimada em cerca de 70% em termos de documentação. Concluindo a habilidade *performa* é a habilidade essencial para a realização um negócio de qualquer tipo, no que toca tanto a coordenação como a produção.

Segue-se um exemplo da indefinição e ambiguidade nas técnicas tradicionais de modelação de processos.

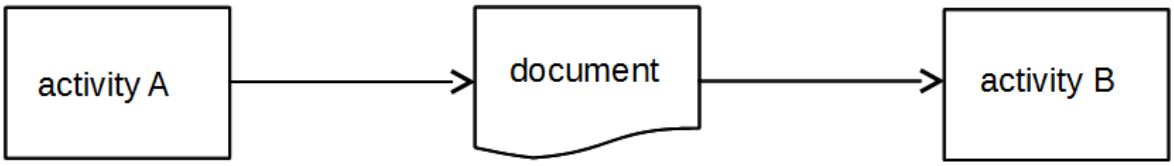

Figura 4 - Indefinição na modelação de um processo.

Passar o documento da actividade A para a actividade B é:
Apenas um acto **datalógico**?
Exemplo: A entrega o documento a B para ser arquivado.
Ou também um acto **infológico**?
Exemplo: A informa B sobre o conteúdo do documento.
Ou ainda um acto **ontológico**?
Exemplo: A pede a B para desempenhar algo.

Os axiomas da teoria Ψ, solidamente baseados em outros campos da ciência, constituem poderosas ferramentas de abstracção que permitem a especificação da operação e estrutura essencial das organizações, tanto a nível de interacções, como a nível de factos de negócio, independentemente da especificação ou não de todos os actos de cada transacção e da forma como a coordenação e produção é implementada, recorrendo-se a tecnologia de qualquer tipo – incluindo-se aqui na noção de tecnologia, o recurso a *software* e/ou pessoas.

2.1.3. Teorema da Organização

O teorema da organização diz que a organização de uma entidade é constituída pela integração em camadas de três sistemas: a organização do negócio (*B-organization –Business*) a organização intelectual/informacional (*I-organization – Intellect/Information*) e a organização documental (*D-organization – Document*). As relações entre elas são que a organização documental suporta a organização intelectual, e a organização intelectual suporta a organização de negócio. A integração é estabelecida através da unidade coesiva do ser humano. Todos os três sistemas pertencem à categoria de sistemas sociais, ou seja são semelhantes no que toca à coordenação: os elementos são sujeitos que entram em acordo e cumprem compromissos uns para com os outros no que toca a actos de produção. Eles diferem apenas de produção: a produção na organização de negócio é ontológica, a produção na organização intelectual é infológica e a produção na organização documental é datalógica. As organizações: de negócio, intelectual e documental consistem em sistemas que são facetas do todo da organização.

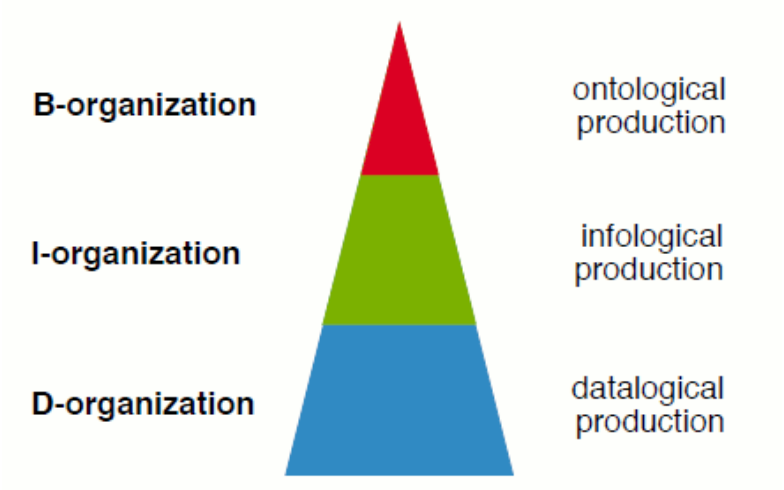

Figura 5 - Representação do teorema da organização.

A Figura 4 exibe de outra forma as camadas das três facetas acima referidas. A distinção entre perspectiva de função (F) e de construção (C) serve para exibir como estas camadas se relacionam de uma forma mais precisa.

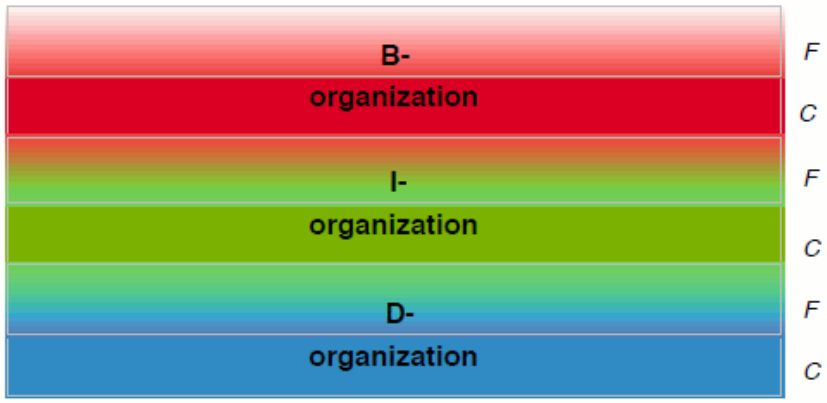

Figura 6 - Integração de uma organização em camadas.

Como mostra a Figura 5 a função da organização intelectual suporta a construção da organização de negócio e a função da organização documental suporta a construção da organização intelectual. Para enfatizar a natureza intermediária da perspectiva de função a barra correspondente tem as cores da barra superior, inferior e a junção de ambas. Podemos pensar que o ambiente de construção de uma entidade é a barra de função na organização de negócio. Na prática, isto consiste numa especificação explícita e concreta do contexto social onde a entidade opera.

Para discutir o significado exacto da integração dos três aspectos organizacionais, a organização intelectual é escolhida como ponto de partida. Tomando em perspectiva a função podemos ver que a organização intelectual fornece os serviços de informação para a construção

da organização de negócio, isto é, para os *B-actors*. É crucial reconhecer que o serviço de informação fornecido é determinado por estes *B-actors* e que é portanto (funcionalmente) descrito em termos de operação da organização de negócio. Por exemplo algum B-actor numa companhia pode querer saber ao final de cada dia de trabalho qual foi o seu volume de negócio. Volume de negócio é uma noção que significa algo para ele ou ela. Esta noção encaixa num quadro de outras noções económicas que são básicas para o seu papel na organização de negócio. Em contraste com este *B-actor* um *I-actor* não dá valor ao conceito de volume de negócio. Por exemplo não vai ficar nervoso se os valores do volume de negócio baixarem. Apenas utiliza o termo para fazer a entrega do serviço. Nada mais é que uma etiqueta para o desfecho de um cálculo em particular. De qualquer forma, *I-actors* fazem parte da construção da organização intelectual e, consequentemente estão preocupados com o seu dever como executor de *I-transactions* nas quais *I-products* são produzidos. Um exemplo do resultado de uma *I-transaction* é o de um total diário com um conjunto de valores bem definido que é calculado. Este resultado, etiquetado de volume de negócio, é entregue ao iniciador da transacção. Então, para o *I-actor* executante, volume de negócio é definido como a soma total de alguma coisa, ao contrário do iniciador da transacção. Este iniciador da transacção é o *B-actor* que necessitou saber o volume de negócio. O assentar das camadas da organização de negócio e da organização intelectual é como se segue. O sujeito que cumpre o papel *B-actor* dispõe de todas as três habilidades humanas, como discutido anteriormente. Na realização de actos de produção ontológicos, ele ou ela usa exclusivamente a capacidade *Performa*. Por outro lado, como iniciador ou executor de *B-transactions*, ele ou ela precisa de todas as três habilidades para realizar actos de coordenação. O sujeito temporariamente toma a forma de *I-actor*, nessa forma está perfeitamente apto a participar de *I-transactions*. Depois de o ter feito, e depois de ter aceitado o resultado da *I-transaction*, o sujeito muda de volta, por assim dizer, à sua forma de *B-actor*, e retoma o seu trabalho nessa forma, agora fornecido com o conhecimento do valor do volume de negócios. Um raciocínio semelhante aplica-se ao assentamento de camadas da organização intelectual e organização documental.

2.1.4. Modelação ontológica com o DEMO

No DEMO, o modelo ontológico completo de uma organização é entendido como o modelo da sua organização de negócio. Este modelo é composto de quatro submodelos, representados por determinados diagramas, tabelas e listas: o modelo de construção (CM) - constituído pelo Modelo de Interacção (IAM) e pelo Modelo de *Interstriction* (ISM) - o Modelo de Processo (PM), o modelo de acção (AM) e o Modelo de Estado (SM). Estes modelos, representados na Figura 5, constituem o modelo ontológico completo da organização de negócio e, subsequentemente representam o modelo ontológico da entidade correspondente. O triângulo nesta figura é idêntico à parte superior do triângulo da Figura 5.

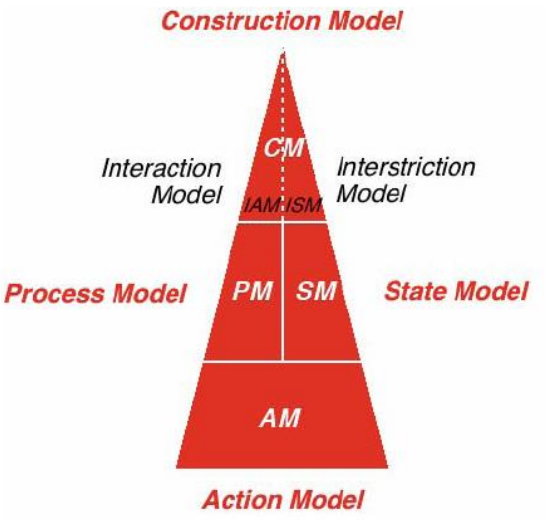

Figura 7 - DEMO modelos de aspecto ontológicos.

À esquerda na Figura 7, estão os diagramas em que os modelos de aspecto são expressados, enquanto à direita, estão as tabelas e o seu respectivo cruzamento de informação entre os modelos. Uma breve descrição para cada um dos modelos de aspecto é feita em seguida.

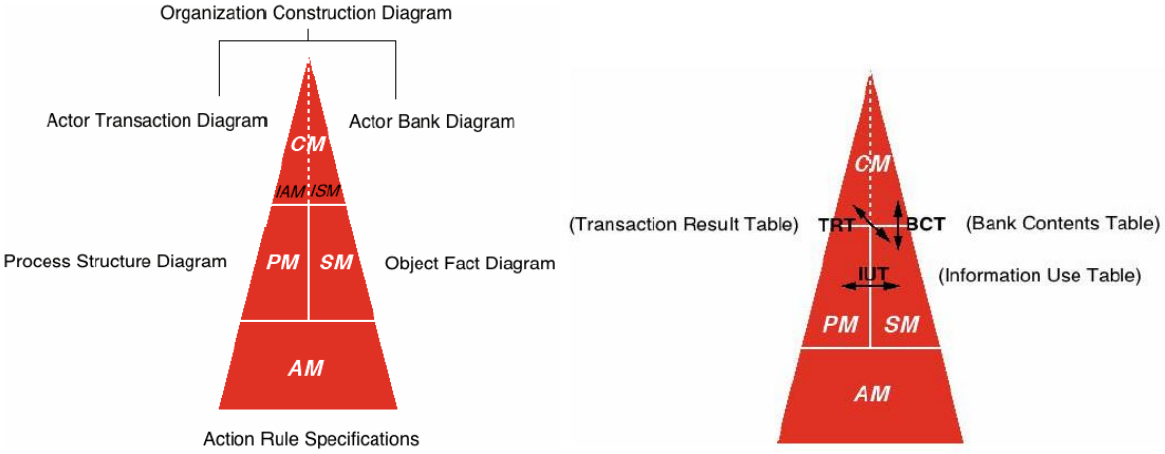

Figura 8 - Diagramas e Tabelas dos modelos DEMO.

O Modelo de Construção (CM) específica a construção do sistema organizacional identificando os tipos de transacções e os papéis de actor associados, bem como as ligações de informação entre os papéis actor e os bancos de informação. É o modelo mais conciso e está dividido em duas partes, a parte activa, o modelo de interacção (IAM), representado no modelo de actores e transacções (ATD – *Actor Transaction Diagram*), e da parte passiva, o modelo *Interstriction* (ISM). A estrutura de interacção de uma organização consiste nos tipos de transacção em que os papéis actor identificados participam como iniciador ou executor. Enquanto a estrutura *interstriction* mostra a estrutura do sistema passivo ou seja, as ligações de informações entre os papéis de actor e bancos de informação contendo os factos de negócio.

O Modelo de Processo (PM – *Process Model*) contém para cada tipo de transacção no CM, o padrão de transacção específico (padrão básico, modelo padrão, padrão de cancelamento) do tipo de transacção. Em outras palavras, ele contém os detalhes dos tipos de transacção identificados. Nomeadamente, estão contidas no PM, representado no Diagrama de Estrutura de Processos (PSD – *Process Structure Diagram*), as relações causais (relações em que um acto de coordenação de uma transacção provocam a ocorrência de um acto de coordenação ou produção de outra transacção) e condicionais (relações em que um acto de coordenação ou produção de uma transacção tem de esperar pela ocorrência de um acto de coordenação numa outra transacção) entre operações.

O Modelo de Acção (AM) especifica as regras de negócio obrigatoriamente formuladas que servem como directrizes para os actores em como lidar com a sua agenda. O AM contém uma ou mais regras de acção para cada tipo de *agendum* (acto de coordenação ocorrido). Estas últimas regras podem ser consideradas como regras de negócio imperativas.

O Modelo de Estado (SM) especifica o espaço de estados e o espaço de transições do mundo de produção incluindo: classes de objectos, tipos de fato (os objectos ou informação centrais de negócio), tipos de resultados (resultados da execução de transacções, i.e., factos de negócio afectando o estado de objectos anteriormente referidos) e regras de existência (regras de unicidade e dependência de factos). Estas últimas regras podem ser consideradas como regras de negócio declarativas. O SM é representado no Diagrama de Factos e Objectos (OFD – *Object Fact Diagram*).

2.2. Sistemas de Fluxo de Trabalho

Dado que, com a modelação DEMO é possível especificar os vários processos de uma empresa, os sistemas de Fluxo de Trabalho são o termo genérico para as aplicações que permitem a posterior derivação destes processos para um sistema de informação, utilizando para isso uma das notações existentes actualmente, como por exemplo o BPMN (*Business Process Model and Notation*).

Posto isto, o objectivo principal desta subsecção é a apresentação, explicação e as funcionalidades gerais existentes neste tipo de sistemas de Fluxo de Trabalho.

Uma introdução do que é um sistema de Fluxo de Trabalho, as suas vantagens e desvantagens e o impacto destas no quotidiano das empresas.

Por fim, a comparação entre os sistemas de Fluxo de Trabalho e o protótipo a ser desenvolvido em termos das vantagens e desvantagens que o protótipo visa resolver.

2.2.1. Introdução ao Fluxo de Trabalho

O conceito de Fluxo de Trabalho surgiu pelo rápido desenvolvimento do mundo económico e financeiro das empresas e também devido ao acelerado aparecimento de tecnologias relacionadas com a *internet* [5].

O Fluxo de Trabalho pode ser considerado uma classe do tipo completa ou parcial de vários processos de negócio, que passam a ser realizados de forma automática, de acordo com um conjunto de regras, documentos, informações e tarefas que são movimentadas entre a transmissão e execução num processo de negócio [5] [6] [7].

Como já referido, um Fluxo de Trabalho disponibiliza tarefas que podem ser sequenciais e relativas ao trabalho diário, ou mais complexas, em que estas são realizadas concorrentemente e afectando outras, seja antes ou após a sua execução [7].

Inicialmente, os processos eram realizados inteiramente por humanos que manuseavam objectos físicos, isto foi substituído pelos sistemas de informação, graças à introdução de novas tecnologias. Estes sistemas de informação realizam de forma automática tarefas com o objectivo principal de fazer cumprir as regras de negócio, que antes eram validadas pelos humanos [6].

2.2.2. Sistemas de gestão de Fluxo de Trabalho

Os sistemas de gestão de Fluxo de Trabalho permitem às organizações empresariais definir e controlar as várias actividades que fazem parte dos seus processos de negócio. Estes sistemas reúnem opções para as organizações medirem, analisarem e melhorarem continuamente os processos. Os melhoramentos podem ser a curto ou a longo prazo, a curto prazo quando existem problemas que ocorrem ocasionalmente durante o dia-a-dia, ou a longo prazo para redefinir partes do fluxo de processo evitando problemas que podem acontecer no futuro [7].

O sistema pode realizar as tarefas de forma em que é necessário a intervenção humana (próxima), ou sem a intervenção de humanos, apenas mostrando progresso (livremente). Por acréscimo ao grupo de tarefas, o Fluxo de Trabalho também define a ordem de invocação das tarefas e as condições ou regras, que devem fazer com que as tarefas sejam invocadas, a sua sincronização e fluxo de dados [6].

Os sistemas de Fluxo de Trabalho normalmente são integrados com outros sistemas independentes que são utilizados pela organização, por exemplo: sistemas de gestão documental, bases de dados, *e-mails* e sistemas de facturação [7].

Hoje em dia, existem no mercado várias ferramentas que permitem a definição destes processos, tendo deste modo, uma estrutura de processos da empresa semelhante a uma árvore [5].

2.2.2.1. Funcionalidades Gerais

Ferramenta de definição de processo: uma ferramenta do tipo gráfica ou de texto para definir os processos de negócio. Cada tarefa do processo de negócio fica associada a uma pessoa ou aplicação de computador. As regras são criadas para determinar como é que as tarefas progridem ao longo do Fluxo de Trabalho e o que é necessário realizar para cumprir esses requisitos [7].

Simulação e prototipar: alguns sistemas permitem uma simulação prévia da estrutura de processo criada, antes da mesma ser colocada como sistema final [7].

Controlo, iniciação e automatização de tarefas: o processo de negócio definido na funcionalidade anterior é iniciado e os recursos humanos e técnicos são agendados para completar cada uma das actividades, à medida que o processo progride. As tarefas podem ser invocadas automaticamente. A automatização tem de ser previamente configurada e a utilização de um método capaz de verificar e invocar as tarefas [7]. Por exemplo, no caso do protótipo a ser

desenvolvido existirão transacções que serão de auto activação e estas são iniciados automaticamente consoante um horário agendado.

Tomada de decisão baseada em regras: as regras e condições são concebidas para cada passo da tarefa, para ser determinado como é que os dados do Fluxo de Trabalho são processados, encaminhados, localizados e controlados [7]. Por exemplo, nos modelos DEMO utilizam-se as ligações causais e de espera, nas ligações causais são controladas as transacções que devem ser iniciadas quando ocorre um certo acto nessa transacção.

Encaminhamento de documentos: em sistemas simples, isto pode ser conseguido ao passar um ficheiro ou pasta, de uma origem para um destino [7].

Listas de trabalho: as listas permitem cada trabalhador identificar as tarefas correntes que precisam de acções por parte dos trabalhadores para poderem prosseguir. Os sistemas apresentam o estado das tarefas correntes e quanto tempo é necessário para realizar as mesmas [7]. Por exemplo, no protótipo a ser desenvolvido e na sessão de cada utilizador poderão ser apresentadas apenas as transacções a que corresponde a responsabilidade de executar esse passo.

Notificação de eventos: os utilizadores têm a possibilidade de ser informados por notificação, quando existem actualizações nos passos das tarefas [7]. Por exemplo, o protótipo a ser desenvolvido poderá ter uma comunicação em tempo real, aplicando uma tecnologia que permita ao utilizador receber uma notificação quando existe alterações nas tarefas.

Monitorização de processos: o sistema pode fornecer valiosas informações do trabalho presente, futuro e dos processos e tarefas por finalizar. O sistema do mesmo modo pode apresentar problemas de gestão em certos processos a decorrer no momento, ou em momentos distantes [7].

Acompanhamento e registo das tarefas: As informações dos passos das tarefas são registados, guardando uma data de criação, alteração e até, quem e quando foram os utilizadores que realizaram essas alterações [7].

Administração e segurança: um componente de permissões incorporado no sistema principal é o responsável por identificar os participantes e os seus respectivos privilégios. Devem apenas apresentar as tarefas e dados, que correspondem ao nível de acesso do utilizador em questão [7]. Por exemplo, no protótipo a ser implementado, um utilizador pode ter vários papéis/funções e estas podem conter vários actores (são os responsáveis por realizar as transacções/tarefas).

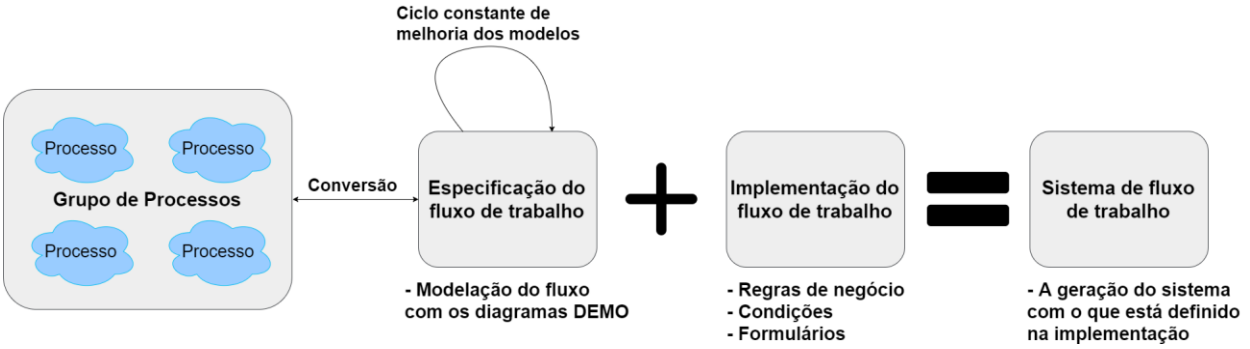

Figura 9 - Exemplo de sistema de Fluxo de Trabalho. Adaptado de [6].

Na Figura 9 é possível vislumbrar as fases para a obtenção de um sistema de Fluxo de Trabalho.

- A primeira fase é a recolha e escolha dos processos candidatos.
- A segunda fase é a especificação e modelação desses processos escolhidos, empregando uma linguagem de modelação de processos.
- A terceira fase é a passagem destes modelos construídos na segunda fase, para o protótipo dinâmico de geração do Fluxo de Trabalho.

Como referido anteriormente, a Figura 9 representa em termos gerais o que é necessário para a criação de um sistema de Fluxo de Trabalho. Neste momento, é possível interligar a modelação dos processos utilizando a linguagem DEMO e os componentes do protótipo a ser desenvolvido.

No protótipo a ser desenvolvido, a "Especificação do fluxo de trabalho" é a modelação dos processos com a linguagem DEMO, ao passo que a "Implementação do fluxo de trabalho" é a posterior derivação desta modelação de processos, em regras de negócio, condições e formulários.

O rectângulo "Implementação do fluxo de trabalho" da Figura 9 é o protótipo a ser desenvolvido e que estará detalhado nos capítulos seguintes.

2.2.2.2. Os benefícios dos sistemas de Fluxo de Trabalho

Oportunidades para fazer alterações na organização: os sistemas de Fluxo de Trabalho ajudam os departamentos das organizações, em todos os patamares da área de negócio para cumprir as mudanças fundamentais e para operar de maneira eficaz. As mudanças podem incluir a modificação da estrutura da organização e também a forma como se define a orientação de todo o departamento da organização [7].

Oportunidades para efectuar alterações no processo: já que os sistemas de Fluxo de Trabalho forçam as organizações a examinar, analisar, estruturar e definir os seus processos de negócio, esta é a altura ideal para considerar a redefinição destes processos. É essencial que um dado processo seja analisado e aperfeiçoado antes da sua implementação, de maneira a evitar más práticas na execução desse processo [7].

Menos controlo humano na gestão dos processos: visto que, os passos das tarefas, regras e condições são definidas no sistema, o sistema é responsável por fazer cumprir vários requisitos e por isso não é fundamental a intervenção humana rigorosa na gestão do processo de negócio em termos da sequência desse fluxo de tarefas [7].

Facilmente especificar as regras do Fluxo de Trabalho: o sistema deve proporcionar a hipótese do utilizador alcançar transformações nas tarefas que fazem parte de um processo, apenas realizando a modificação da regra de fluxo sem ter de proceder a alteração do sistema [5].

Determinar naturalmente o estado da tarefa: o utilizador encontra de forma fácil o estado da tarefa em questão e pode então proceder aos passos necessários para a continuação da execução da tarefa. Ao mesmo tempo pode verificar qual o problema, que não permite o passo seguinte do Fluxo de Trabalho, analisando o estado do processo [5].

Apresentar o trabalho ao utilizador: tipicamente um trabalhador necessita de procurar e verificar que tarefas tem de realizar. No sistema é muito adequado do ponto de vista do utilizador colocar as tarefas/actividades que tem de executar à frente do utilizador. Deste modo, o sistema tem por obrigação de posicionar as tarefas que precisam de ser completadas logo que, o utilizador concretizar a autenticação com as suas credenciais [5].

2.2.2.3. As desvantagens dos sistemas de Fluxo de Trabalho

Apesar de muitos benefícios, os sistemas de Fluxo de Trabalho têm algumas limitações importantes [6]. De salientar, que o uso dos sistemas de Fluxo de Trabalho não resolvem os problemas intrínsecos existentes nas organizações, se para o caso em que o processo em si, já está mal estruturado e definido, sem a presença destes sistemas [7].

Resistência humana: os factores humanos representam o maior obstáculo à aceitação das aplicações de sistemas de Fluxo de Trabalho. Os funcionários das organizações consideram muitas vezes que estes sistemas retiram a hipótese de tomada de decisão dos mesmos, ou são um instrumento para reduzir o trabalho que tem de executar no seu dia-a-dia. Ao mesmo tempo, acreditam que um sistema com este formato é uma invasão para a sua privacidade porque, tudo fica registado e tudo é controlado [7].

Complexidade: quando os processos da organização são estruturados e definidos para serem usados no sistema, os mesmos não podem ser muito detalhados pela simples razão de que um sistema que trata e gere cada pequeno detalhe do processo pode tornar-se excessivo, incidir em partes desnecessárias ou até mesmo na resistência humana.

Desempenho desadequado: os sistemas de Fluxo de Trabalho geralmente não conseguem suportar uma quantidade grande de fluxos de trabalho por dia. Este mau desempenho afecta directamente o funcionamento das empresas em toda a sua área de trabalho [6].

Perda de flexibilidade: alguns processos requerem que os trabalhadores se mantenham flexíveis e apliquem o julgamento pessoal [7].

Custos técnicos de implementação: os sistemas de Fluxo de Trabalho podem ser complexos, precisando por isso, de uma quantidade importante de recursos para a sua implementação [7].

Custos na definição de processos complexos: os processos presentes nas organizações por si só, podem ser complicados para definir e para que haja sucesso nesta definição, é crucial o compromisso entre o modelador e os responsáveis pelos processos da empresa, sublinhando que é capaz de ser utilizada uma quantidade considerável de tempo [7].

Falta de interoperabilidade entre os sistemas de Fluxo de Trabalho: está directamente ligado com a falta de padrões pré-definidos para os sistemas de Fluxo de Trabalho. É uma falha que não permite os sistemas de Fluxo de Trabalho comunicarem entre si ou com outras aplicações de forma transparente [6].

Ferramentas de fraca qualidade para analisar, testar e de resolução de problemas: estas ferramentas permitem verificar, analisar e simular como os processos comportam-se no sistema de Fluxo de Trabalho. A fraca sofisticação e qualidade das ferramentas influencia a modelação, dificultando o rápido desenvolvimento de protótipos e a facilidade da especificação do Fluxo de Trabalho [6].

2.2.2.4. Como escolher os processos

Todos os processos existentes numa organização, podem ser modelados e definidos num sistema de Fluxo de Trabalho no entanto, à priori, é possível obter uma triagem dos processos que são imediatamente candidatos a serem especificados no sistema.

Os processos que mais são beneficiados com a sua incorporação num sistema deste tipo são os que envolvem muitos documentos ou informações que são passadas entre várias pessoas de diferentes departamentos ou empresas, de destacar que processos mais simples também beneficiam da sua colocação no sistema de Fluxo de Trabalho [4].

Alguns critérios para encontrar os processos prioritários: [4]

- **Processos demorados:** podem ser complexos, por causa da transmissão de várias informações por diferentes entidades dificultando assim, a velocidade de resolução do processo em questão. Além do facto de serem complexos, é vital que seja possível saber em que estado se encontra o processo de forma fácil.
- **Satisfação das entidades envolvidas:** processos que incluem entidades externas à organização como por exemplo clientes são outro tipo de processos candidatos, a satisfação do cliente quando realiza uma tarefa de um processo é importante para o sucesso de uma organização. O processo é candidato, se uma tarefa de um processo não é satisfatória para um cliente, porque é demasiado confusa ou não existe uma estrutura definida de como ocorre o fluxo de dados, com perda de dados entre os caminhos desse fluxo.
- **Qualidade do processo final:** se o "produto" final obtido por um processo não é de boa qualidade, ou em que existem vários retrocessos, que prejudicam a qualidade do resultado final do processo.

Aplicando os critérios acima referidos, é viável identificar os processos que têm mais impacto no funcionamento da organização. Após essa identificação, é realizada a escolha dos processos mais prioritários para o sistema de Fluxo de Trabalho.

2.2.3. Sistemas de Fluxo de Trabalho e o protótipo a ser desenvolvido

Apresentadas assim as vantagens e desvantagens gerais dos sistemas de Fluxo de Trabalho, o protótipo a ser desenvolvido visa a integração das várias vantagens, como também colmatar as desvantagens portanto, o protótipo a ser desenvolvido visa resolver vários problemas, a que as empresas estão sujeitas no seu dia-a-dia.

Em seguida, está exibida a enumeração das vantagens que o protótipo deve ter, tendo em conta as vantagens expostas nas subsecções anteriores:

- **Oportunidades para fazer alterações na organização**: com os modelos do DEMO, é possível ter uma visão mais completa e respectiva análise de como é efectuado todo o fluxo de um determinado processo, detectando dessa maneira falhas e problemas na execução destas tarefas. Estes modelos possibilitam oportunidades para a realização de alterações na estrutura da organização e no processo em si.

- **Menos controlo humano na gestão dos processos:** sendo que o protótipo a ser implementado rege-se pelas regras e condições do DEMO, existe menor necessidade de intervenção humana na gestão destes processos, porque o protótipo é responsável por controlar, organizar e manter a coerência em todos os passos das transacções destes processos.

- **Facilmente especificar as regras do Fluxo de Trabalho:** como os componentes que integram o protótipo devem permitir a especificação de forma dinâmica, as modificações numa determinada tarefa/transacção deverão ser realizadas facilmente, usando os componentes a ser desenvolvidos no protótipo, sem a alteração geral do núcleo do sistema.

- **Determinar naturalmente o estado da tarefa:** o Painel de Controlo a ser implementado no protótipo, deve fornecer os estados actuais das tarefas/transacções presentes no sistema e por essa razão o utilizador encontrará de forma fácil, em que estado uma determinada tarefa está no momento actual e pode decidir se pretende continuar para o passo seguinte.

- **Apresentar o trabalho ao utilizador**: dado que o Painel de Controlo apresentará as tarefas/transacções que estão a ocorrer e estas estão colocadas à frente do utilizador, o mesmo não necessita de forma manual, analisar e verificar que passos ou tarefas precisa de executar, para acabar ou iniciar um dado processo de negócio.

Em seguida estão exibidas as desvantagens que o protótipo deve colmatar:

- **Complexidade**: os processos que são especificados no protótipo são a passagem dos diagramas obtidos pelos métodos e técnicas de engenharia organizacional com a modelação DEMO, por essa razão os modelos DEMO permitem a diminuição da complexidade dos processos.

- **Desempenho desadequado:** a base do protótipo são as regras e condições da modelação DEMO, pelo que o protótipo deve suportar uma grande quantidade de informações em termos de processos e transacções, que ocorrem no dia-a-dia da empresa.

- **Custos técnicos de implementação:** o protótipo em termos de especificação deve ser totalmente dinâmico, permitindo que os recursos utilizados sejam só despendidos na área da recolha de requisitos e na modelação dos diagramas, que pode ser efectuada por gestores com formação em modelação DEMO, sem ser necessário possuir competências de programação.

- **Custos na definição de processos complexos:** se a pessoa ou grupo responsável pela modelação dos processos de negócio da empresa for um funcionário da empresa e tiver competências em modelação DEMO, a especificação destes processos é menos demorada, pela simples razão de já conhecer os meandros dos processo e respectivos detalhes do fluxo de documentos e informações entre as transacções e os processos. A modelação DEMO é facilmente compreendida por pessoas formadas na área de gestão, mesmo sem formação na modelação.

2.3. Plataformas relacionadas

Nesta secção é apresentado um conjunto de aplicações que estão directamente relacionadas com o protótipo a ser desenvolvido.

De entre as várias opções disponíveis em termos de plataformas, foram escolhidas as seguintes pela semelhança das funcionalidades e documentação suficientemente detalhada destas funcionalidades com as presentes no protótipo a ser desenvolvido, para a realização de análises e comparações.

Para cada aplicação são descritas as funcionalidades, em que se adequa a sua comparação com as funcionalidades presentes no protótipo a ser desenvolvido. Apenas duas funcionalidades das aplicações referidas abaixo são alvo de comparação com o projecto a ser desenvolvido.

As plataformas:

- *Mendix*
- *AppGyver*
- *IBM Designer Forms*

2.3.1. *Mendix*

O *Mendix* é uma aplicação que fornece aos utilizadores o poder para construir e continuamente melhorar aplicações personalizadas a uma escala e velocidade sem precedentes. A aplicação dá a possibilidade de se construir aplicações dedicada à parte móvel ou de *internet*. Esta concede um conjunto de ferramentas para todo o ciclo de vida de uma aplicação, desde a concepção e desenvolvimento até ao lançamento e operação [8].

2.3.1.1. Funcionalidades

1. **Desenvolvimento visual**

Esta é uma das funcionalidades, que é termo de comparação com o protótipo a ser desenvolvido. São funcionalidades gráficas que facilitam a colaboração entre os responsáveis de desenvolvimento dos programas e de negócio. É possível criar as próprias páginas sem utilizar código de programação. Aproveitando uma página com ferramentas de edição gráfica (WYSIWYG - *What You See Is What You Get*) constrói-se um *website* de *internet* totalmente *responsive* e também as interfaces para a aplicação móvel.

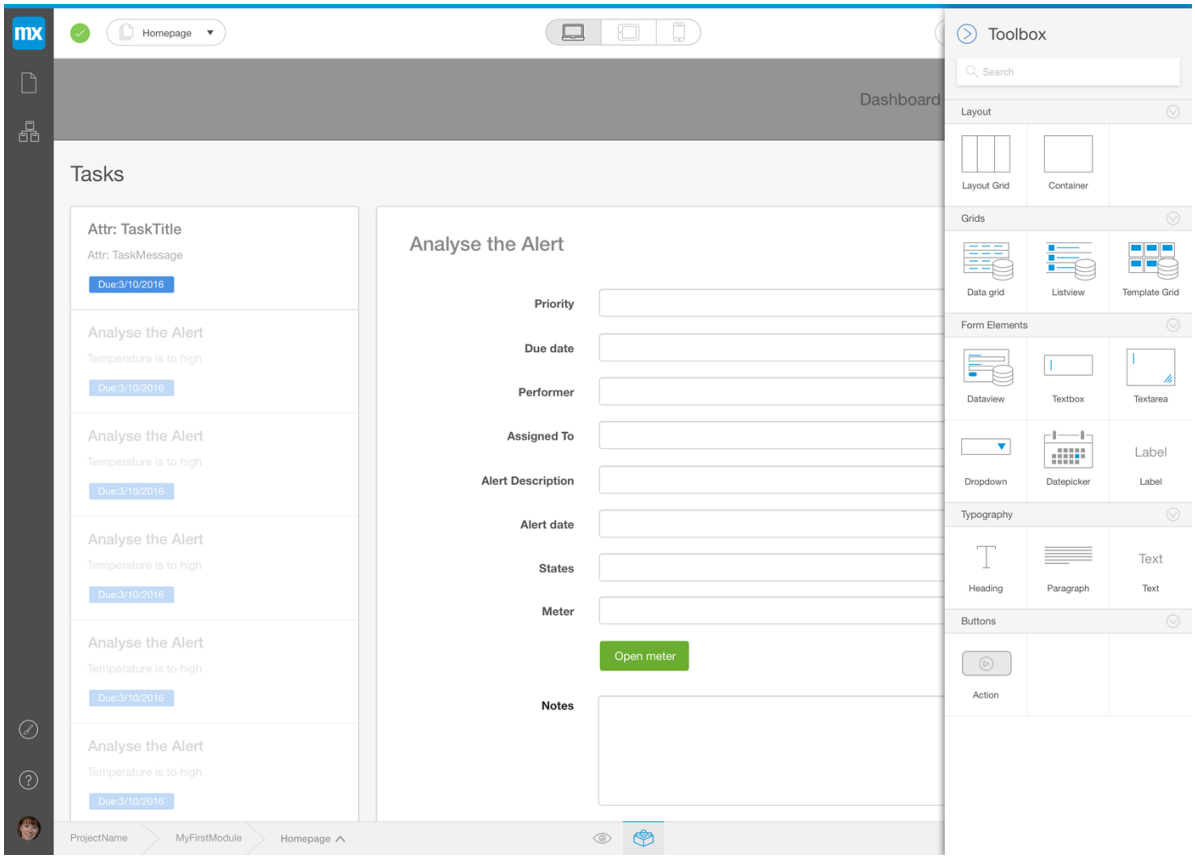

Figura 10 – Um formulário no *Web Modeler* [8].

2. **Lógica da aplicação**

Outra funcionalidade é a eventualidade de criar fluxos de negócio, com o auxílio de modelos visuais e com a colaboração de responsáveis da empresa. A lógica da aplicação criada no *Mendix* pode ser implementada de duas maneiras, em *MicroFlows* ou *NanoFlows*.

MicroFlows

Os *MicroFlows* têm a sua notação baseada no BPMN. Cada *MicroFlow* possui um grupo de elementos que interagem entre si.

Estes elementos podem ser: [9]

- Eventos (*Events*) – que representam os pontos de início e fim de um *MicroFlow* ou operações num ciclo, que são consideradas especiais.
- Fluxos (*Flows*) – estabelecem a conexão entre os elementos.
- Entradas (*Gateways*) – tem como função, lidar com a realização de escolhas e a fusão de caminhos diferentes.
- Actividades (*Activities*) – são as acções que são executadas num *MicroFlow*.
- Artefactos (*Artifacts*) – fornecem entradas e permitem que comentários possam ser feitos num *MicroFlow*.
- Manipulação de erros (*Error handlers)* – podem ser configurados numa actividade (*Activities*), entrada (*Gateways*) ou num ciclo, para definir como lidar com um erro.

Sem um servidor para executar o processamento de toda a lógica criada necessária, os *Microflows* que são executados no lado do cliente, não funcionam em páginas *offline.* A qualquer fluxo do lado do cliente aplica-se o que foi mencionado anteriormente. No entanto, o *Microflow* funciona se for executado fora do âmbito do cliente [10].

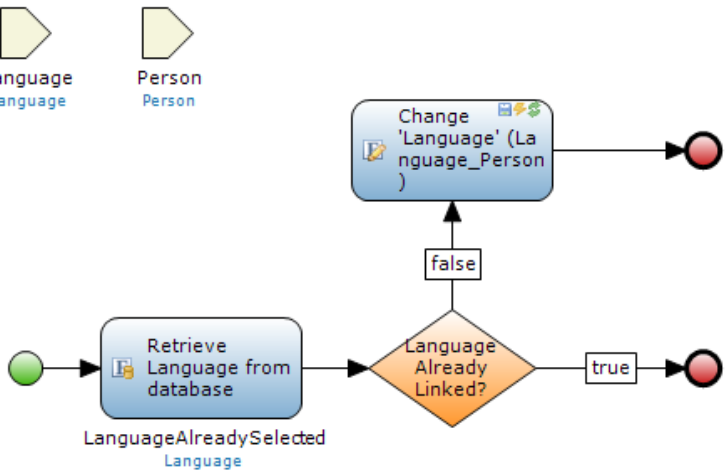

Figura 11 – Um exemplo de *MicroFlow* [11].

NanoFlows

Os *NanoFlows* (ver Figura 12) são semelhantes aos *Microflows* porque, permitem expressar lógica que se pretende ter na aplicação desenvolvida contudo, estes dispõem de uma grande

vantagem, que é a possibilidade de serem executados directamente no navegador de *internet* ou dispositivo, isto significa que podem ser aplicações em modo *offline* [12].

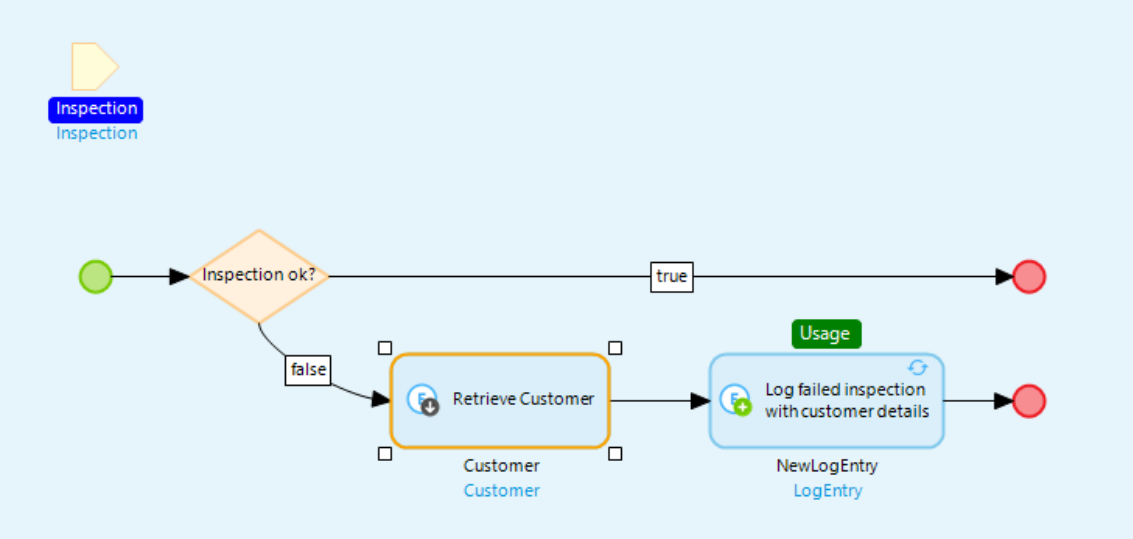

Figura 12 – Um exemplo de *NanoFlow* [12].

Os *NanoFlows* devem ser utilizados quando a lógica dessa aplicação é do tipo móvel, num modo *offline*. Estes fluxos foram criados essencialmente para que o seu uso fosse destinado às aplicações móveis em modo *offline*, porque é a lógica que funciona neste tipo de aplicações. Dado que, todas as acções relacionadas com a base de dados serão executadas numa base de dados *offline*, este tipo de lógica tem um desempenho de execução muito rápido [12].

Este tipo de lógica deve também ser aplicado quando não é necessária conexão, ou seja, numa aplicação *online*, como por exemplo na lógica da interface do utilizador, validações de formulários e em barras de navegação. No entanto, no caso de várias acções relacionadas com uma base de dados, cada acção cria automaticamente um pedido individual na rede de execução do *Mendix*, logo os *NanoFlows* não devem ser utilizados nestes casos em que existem uma grande quantidade de acções deste género.

Existem três diferenças principais entre os *NanoFlows* e *MicroFlows*:

- Quando um *NanoFlow* está a realizar as suas acções, as acções dos clientes são directamente executadas. Por exemplo, uma acção para abrir uma página é imediatamente concretizada e não no fim de toda a lógica estar completa, como acontece com os *MicroFlows*, que apenas abrem a página quando recebem o resultado do *MicroFlow*.
- Algumas expressões não suportam algumas variáveis usadas nos *MicroFlows*.
- Os *NanoFlows* não são executados dentro de uma transacção, se eventualmente ocorrerem erros, não existe a reversão de alterações anteriormente realizadas.

Tal como os *MicroFlows*, os *NanoFlows* são baseados na notação de BPMN. São compostos pelos seguintes elementos:

- Eventos (*Events*) – que representam os pontos de início e fim de um *NanoFlow* ou operações num ciclo, que são consideradas especiais.
- Fluxos (*Flows*) – estabelecem a conexão entre os elementos.
- Entradas (*Gateways*) – tem como função lidar com a realização de escolhas e a fusão de caminhos diferentes.
- Actividades (*Activities*) – são as acções que são executadas num *NanoFlow*.
- Artefactos (*Artifacts*) – fornecem entradas e permitem que comentários possam ser feitos num *MicroFlow*.

2.3.2. *AppGyver*

A aplicação *AppGyver* permite, criar aplicações lógicas avançadas visualmente, construir regras de negócio e automatizações, para tecnologias ligadas ao universo das notificações e *e-mails* sem a utilização de código. Criar interfaces de aplicação com bibliotecas pré-definidas, com a função de arrastar e soltar componentes, definir e desenhar as estruturas das páginas e de navegação graficamente. Efectuar pesquisas dinâmicas e combinar várias fontes de dados [13].

2.3.2.1. Funcionalidades

1. Fluxo de Trabalho

Criar cada processo (ver Figura 13) beneficiando da tecnologia arrastar e soltar, usando conceitos como pontos de partida, entradas e tarefas, e em cada parte do processo definir o que os funcionários têm de fazer. Isto tudo implementado e publicado com apenas um clique [13].

Eliminando assim *e-mails*, chamadas e reuniões desnecessárias, conseguindo que todos os utilizadores estejam sincronizados com notificações e alertas, optimizando a eficiência da organização em todo o processo. Automaticamente organizar as entradas de dados indispensáveis dos utilizadores e decisões a serem tomadas [13].

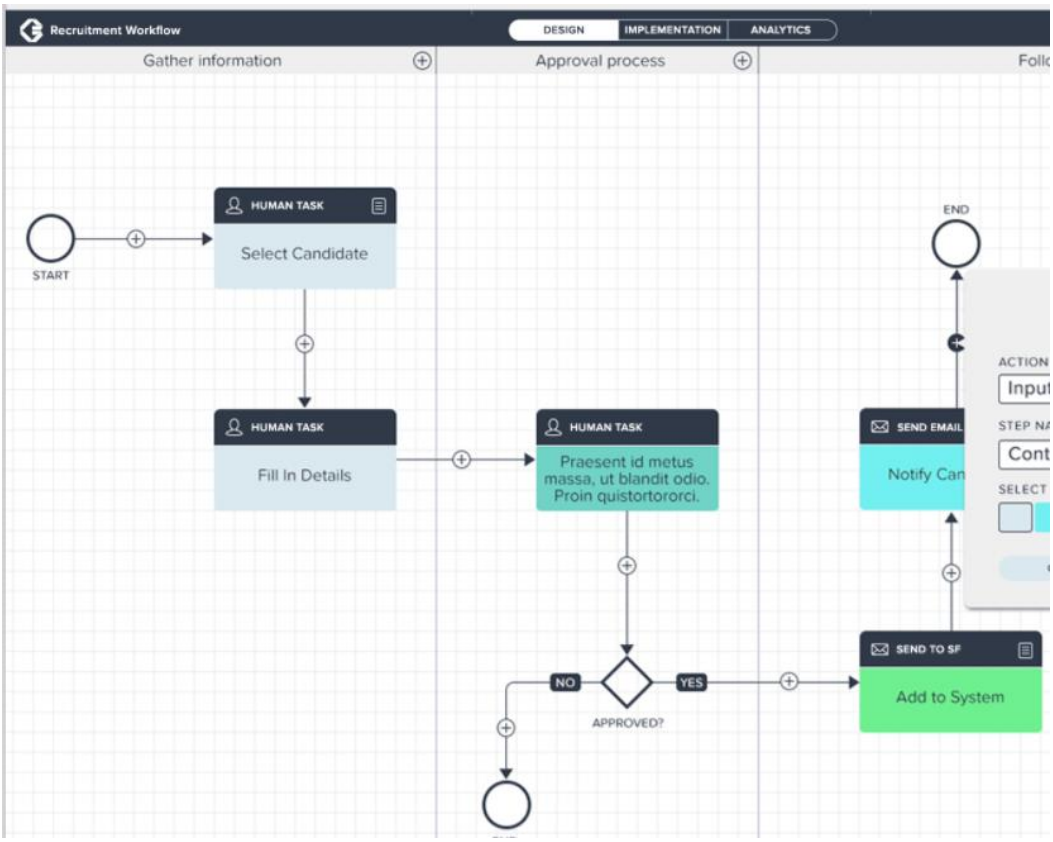

Figura 13 – Ferramenta de criação de processos [13].

2. Formulários personalizados

Ferramenta avançada de formulários (Figura 14) para a construção dinâmica, recorrendo a função disponibilizada de arrastar e soltar. Desenho de formulários complexos em poucos minutos, este editor de personalização é uma aplicação de *internet*, logo pode ser executada em qualquer sistema operativo, com o uso de um navegador de *internet*. Existe a hipótese de ajustar e adicionar formulários em tempo real, à medida que as escolhas estão a ser tomadas [14].

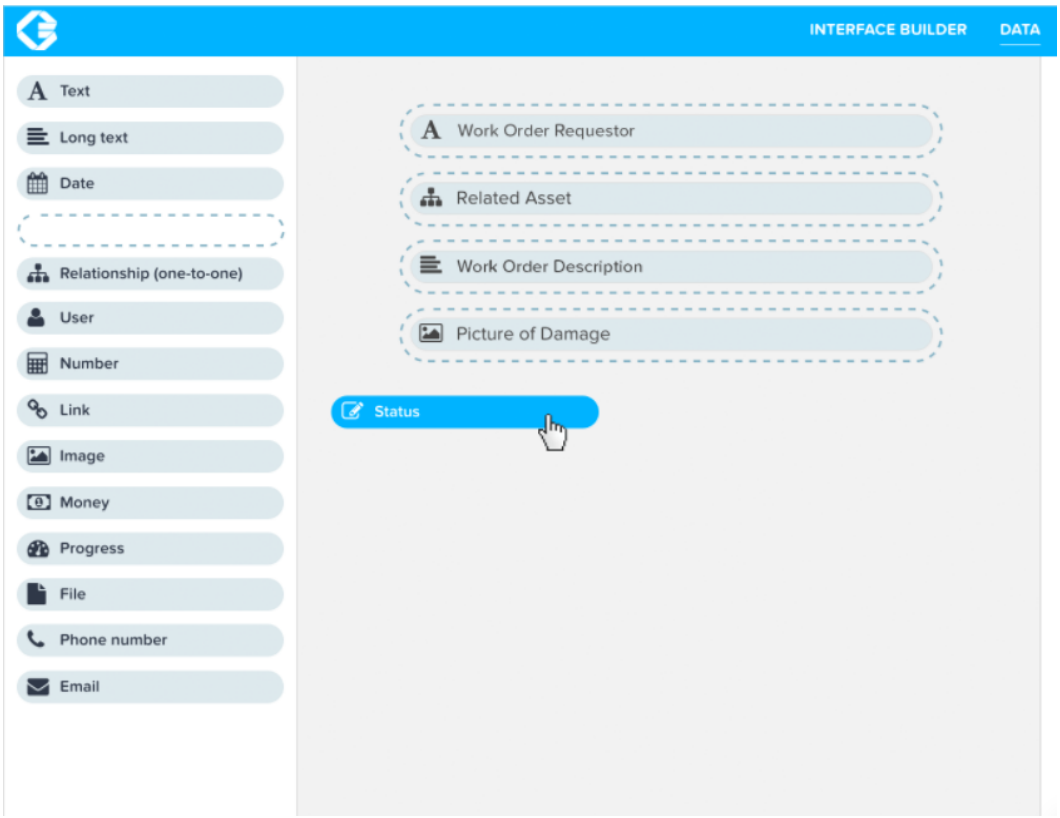

Figura 14 – Ferramenta avançada de formulários [14].

2.3.3. *IBM Designer Forms*

O editor *IBM Designer Forms* é do tipo WYSIWYG, esta ferramenta permite o desenho de formulários, possuindo uma palete de opções de vários tipos, que podem ser inseridos no formulário com a função de arrastar e soltar. Os formulários são facilmente inseridos pela organização de itens numa página [15].

2.3.3.1. Funcionalidades

Existem duas formas de construir os formulários, pela função de arrastar e soltar ou pela escrita de código na janela a que corresponde essa opção.

Em seguida e após o formulário estar concluído, a sua pré-visualização é realizada no *IBM® Forms Viewer* ou no *IBM Forms Server – Webform Server*. No caso de haver formulários em versão papel, é facilmente recriável estes formulários aplicando as funcionalidades e opções do editor, o editor também permite a conversão de formulários de várias versões, como por exemplo do formato PDF (*Portable Document Format*) para um formulário digital [15].

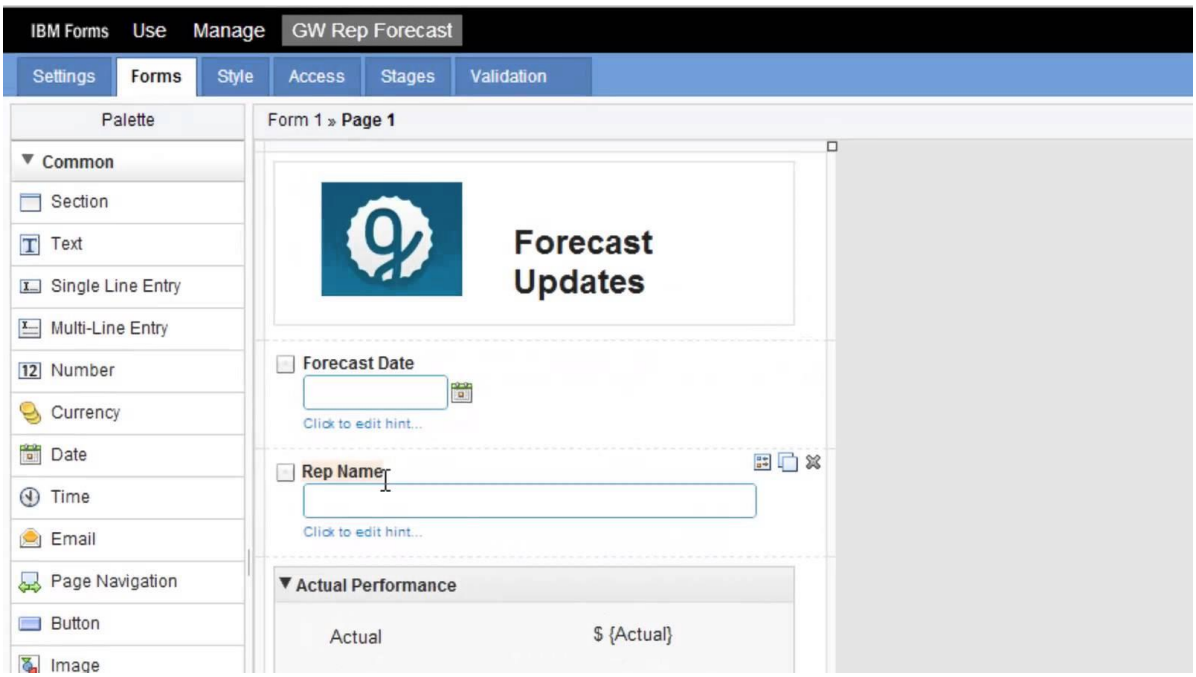

Figura 15 – Ferramenta avançada de formulários.

Na Figura 15 está exibida a ferramenta de desenho de formulários, onde é possível verificar a presença de uma palete com vários elementos, que podem ser usados para a criação de formulários. Do lado direito da palete está a pré-visualização do desenho do formulário que está a ser especificado.

2.3.4. Discussão sobre funcionalidades de plataformas relacionadas e objectivos do projecto

2.3.4.1. *Mendix* e *AppGyver*

As duas funcionalidades principais estão previstas no protótipo a desenvolver, mas com duas distinções.

A primeira funcionalidade, no caso do *Mendix* e *AppGyver* é totalmente gráfica, em que se utiliza a função de arrastar e soltar os elementos no local onde se pretende que fiquem colocados.

No protótipo a ser desenvolvido, as propriedades de um formulário são inseridas através de um formulário, no entanto, a geração destes campos no Painel de Controlo serão gerados de forma automática e dinâmica, tal como ocorre no *Mendix* e *AppGyver*.

A segunda funcionalidade [9] [13] é a construção da lógica da aplicação. Um *MicroFlow/NanoFlow* pode realizar diversas acções, tais como criar e actualizar objectos, apresentar páginas, realizar escolhas e definindo o que os funcionários têm de fazer.

É uma forma visual de expressar, o que normalmente é efectuado em código de programação. A notação utilizada nos micros fluxos é baseada no BPMN [9].

A diferença entre as plataformas, em relação a funcionalidade em questão, é que o protótipo a ser desenvolvido, em termos de fluxos irá reger-se pela notação de modelação DEMO e todo fluxo de transacções de um processo seguirá as regras descritas e especificadas por essa

modelagem. O fluxo de dados e a ordem de execução das transacções são regras definidas com o uso dos diagramas DEMO.

2.3.4.2. *IBM Designer Forms*

Entre o protótipo a ser desenvolvido e o *IBM Designer Forms*, unicamente é possível comparar a segunda funcionalidade, em virtude de o *IBM Designer Forms* ser uma ferramenta de desenho de formulários em páginas. Esta funcionalidade no protótipo a ser desenvolvido é também possível, mas não nos moldes do *IBM Designer Forms*, que utiliza uma técnica de WYSIWYG, para além de oferecer a opção de converter formulários [16] que estão em formato PDF para o formato usado no programa. A opção de converter formulários, não é uma característica do protótipo a ser desenvolvido.

Relativamente a primeira funcionalidade, o *IBM Designer Forms* não possui método de controlo de fluxo de tarefas ao longo do estado de vida de um processo.

2.3.5. Comparação entre o Painel de Controlo a ser desenvolvido e os existentes

O Painel de Controlo do protótipo a ser desenvolvido é uma geração de um conjunto de formulários e propriedades, de acordo com o foi efectuado na especificação de processos da organização com os diagramas DEMO, isto é, que todo o fluxo de tarefas é realizado com as regras que o DEMO dispõe aos modeladores. Este fluxo tem por base a teoria PSI da linguagem DEMO e segue directivas específicas. O Painel de Controlo é fixo em termos estéticos, mas o seu conteúdo altera-se consoante o que está especificado nas tarefas (transacções) dos componentes de especificação.

Nas plataformas relacionadas não existe uma modelação com regras pré-definidas, em termos de fluxo entre tarefas, as regras e a forma como os dados avançam entre as tarefas são realizadas com a modelação BPMN. As plataformas relacionadas não possuem apenas uma zona onde está concentrada toda a gestão dos processos e necessitam de personalizar páginas para colocar os componentes que pretendem.

3. Especificação

Neste capítulo é explicada a ideia geral para o protótipo a ser desenvolvido, os respectivos requisitos nomeadamente os não funcionais e funcionais, a explicação da arquitectura seleccionada, como também as tecnologias mais apropriadas com base nas informações recolhidas na subsecções 3.1, 3.2 e 3.3, de forma a obter a melhor solução que cumpra os objectivos necessários, em relação às funcionalidades que este protótipo deverá ter, para satisfazer as necessidades dos utilizadores finais e restantes partes interessadas.

3.1. Ideia geral para o protótipo

Na Figura 16, é possível analisar que o protótipo estará dividido em duas partes, em que o utilizador final (alguém com permissões de administrador) pode através da interface de edição que será desenvolvida pelos alunos de mestrado, fazer as alterações consoante os processos que a empresa possui, isto é, criar e editar as transacções, como também os formulários associados a cada transacção, permitindo especificar completamente a interface de inserção para os utilizadores finais (sem permissões de administrador).

Assim sendo, o protótipo é extremamente flexível, em termos das possibilidades de edição disponíveis. Estas alterações devem ser efectuadas, em conjunto com as modificações e melhoramentos elaborados nos modelos de engenharia organizacional (DEMO).

A especificação inserida nos componentes que integram o protótipo é obtida dos vários diagramas do DEMO, nomeadamente do ATD, PSD e OFD (juntamente com a OPL - *Object Property List*).

Em suma, a estrutura da base de dados do protótipo "guarda uma estrutura de base de dados dinâmica", que é escolhida e especificada pelo utilizador final (alguém com permissões de administrador), sem que o mesmo tenha de dispor de competências de programação, somente necessita de ter formação no desenho dos modelos de engenharia organizacional, os quais aproximam-se muito à "linguagem/representação" utilizada dentro das empresas.

Figura 16 – Ideia geral do protótipo a ser desenvolvido.

3.2. Requisitos

Os requisitos não funcionais são gerais para toda a equipa de desenvolvimento, uma vez que aplicam-se ao protótipo no seu todo.

Os requisitos funcionais estão mais focados na parte desenvolvida pelo mestrando, mas alguns deles podem ser aplicados a outras funcionalidades implementadas pela restante equipa de desenvolvimento.

3.2.1. Não funcionais

Requisitos de aparência e experiência

O sistema deve ser esteticamente agradável para os utilizadores finais.
O sistema deve ter uma barra de navegação apelativa e de fácil interpretação.
O sistema não deve ser complicado de utilizar.
O sistema deve ter uma interface de navegação simples.
O sistema deve ter um Painel de Controlo personalizável.

O sistema deve ser totalmente personalizável (especificação nos componentes) em relação à base de dados, de acordo com os requisitos recolhidos utilizando os métodos de engenharia organizacional (DEMO).

O sistema deve ser capaz de guardar as especificações feitas nos componentes pelos utilizadores.

Requisitos de usabilidade

O sistema deve ser fácil de aprender e entender.

O sistema deve ser fácil de especificar, em termos de formulários presentes nas transacções.

O sistema deve permitir a fácil procura utilizando para o efeito filtros, pesquisas e ordenação para encontrar os itens que se pretende em cada um dos componentes de especificação.

O sistema deve permitir a fácil inserção e edição dos itens colocados nos componentes de especificação.

O sistema deve permitir a fácil inserção e preenchimento dos formulários das transacções, consoante a especificação realizada nos componentes de especificação (primeira função).

O sistema deve informar o utilizador final utilizando mensagens de sucesso e de erro consoante o caso.

O sistema deve permitir a utilização de uma pesquisa dinâmica, em conformidade com o que foi especificado pelo administrador nos componentes de "especificação" (primeira função).

O sistema deve funcionar em pelo menos dois dos navegadores de *internet* mais conhecidos (*Firefox* e *Chrome*).

Requisitos de desempenho

O sistema deve suportar uma grande quantidade de utilizadores.

O sistema deve responder de forma rápida as acções que o utilizador pratica.

As mensagens de sucesso e de erro devem estar disponíveis no mínimo 3 segundos para que o utilizador final possa identificar a situação ocorrida.

Requisitos de operação

O sistema não funciona se o navegador tiver a função de *Javascript* desactivada, uma vez que é utilizada a biblioteca de *AngularJS*.

O navegador utilizado deve ter a função de *Javascript* activa.

O sistema deve funcionar em qualquer tipo de rede, independentemente do desempenho da mesma.

O sistema deve funcionar independentemente de haver ou não ligação com o servidor fornecido pela *framework Pusher*.

Requisitos de segurança

Os utilizadores finais não podem ver a informação total de outros utilizadores finais.

O administrador pode ter acesso a todas as partes do sistema.

As transacções criadas não podem ser acedidas por utilizadores finais, sem a permissão para fazê-lo.

No caso em que a especificação envolva várias empresas no mesmo protótipo, a informação das mesmas não pode estar visível a outras.

O carregamento de ficheiros para o servidor onde está alojado o protótipo, também deve estar apenas disponível a quem tem permissões de visualização.

Requisitos de portabilidade

O sistema deve adaptar-se em termos de aparência (ser *responsive*) a qualquer *hardware* e resolução de ecrã em que seja executado, seja um computador ou telemóvel.

O sistema deve poder ser executado em qualquer país, ou em casos especiais e pela decisão da empresa ser executado apenas na rede interna desta.

Restrições do protótipo

O sistema deve:

- Ter uma base de dados.
- Ter uma comunicação em tempo real em algumas partes do sistema.
- Ter um componente de página de acesso (*login*).
- Ser o mais dinâmica possível em relação a usabilidade, utilizando para isso uma *framework* que o permite, exemplo: *AngularJS*.

3.2.2. Funcionais

Gerais

- Quando uma acção é realizada na aplicação, deve ser mostrada uma notificação de sucesso ou não sucesso.
- Quando a aplicação está a carregar, como por exemplo uma página ou a receber dados da base de dados, deve ser mostrado uma notificação dessa ocorrência.

Primeira Função (Modelação) - Especificação dinâmica da estrutura de processos

Base de dados

- Deve guardar informações sobre os utilizadores finais para o login.
- Deve guardar a estrutura de processos (“especificação”) obtida com os modelos do DEMO e que foi inserida através dos componentes por um utilizador.

Administrador

- Só o papel de administrador tem acesso as áreas da primeira função (Modelação).
- O administrador deve poder inserir e editar as áreas que compõem a especificação dinâmica (primeira função/Modelação).
- As áreas são tipos de processo, transacções, entidades, relações, estrutura de processos e propriedades.
- Deve ser possível ter uma parte de filtros, em que é possível filtrar os dados presentes nos componentes.
- Deve ser possível ordenar os dados dos componentes.

Segunda função (Execução) - Execução dos processos.

Base de dados

- Guardar os “vistos” dos utilizadores consoante as transacções, os “vistos” são a notificação de que um utilizador já viu uma dada tarefa/transacção.

- Guardar as tarefas iniciadas ou prosseguidas pelos utilizadores.

Utilizador Final

- O utilizador final deve poder aceder ao protótipo utilizando credenciais de *login*.
- Só devem aparecer as transacções e processos que correspondem ao utilizador com sessão actualmente iniciada no protótipo, através dos papéis e actores DEMO do mesmo.
- No Painel de Controlo deve ser possível criar novas instâncias de transacção.
- Deve ser possível prosseguir para um estado da transacção/tarefa actualmente existente.
- Quando um utilizador segue para o estado seguinte de uma tarefa, é necessário verificar se existe uma Ligação de Espera e não permitir a sua continuidade.
- Se uma certa tarefa causa (Ligações de Causa) uma outra tarefa, esta nova tarefa deve ser inserida automaticamente na base de dados.
- Se uma certa propriedade de um formulário necessita de informações de outra propriedade de outra transacção, deve apenas apresentar os dados em relação a esse processo.
- Verificar para cada acto da tarefa/transacção se existem Tipos de Entidade ou Relação, e apresentar as propriedades em formato de formulário para cada um dos actos.
- Os actos das tarefas/transacções que tenham formulários devem ser separados e exibidos por abas.
- Ao abrir um estado de uma tarefa, é necessário verificar se o estado actual já tem o formulário preenchido.
- No caso em que o formulário não está preenchido no estado actual, apresentar o formulário para realização do preenchimento por um utilizador.
- Não apresentar no Painel de Controlo, os processos que estão definidos com o modo *finished* no campo correspondente na base de dados.
- Deve ser possível informar que a transacção/tarefa foi vista (um género de *acknowledge*).
- O utilizador final deve ter acesso a uma pesquisa dinâmica, construída com a especificação feita na primeira função (Modelação).

3.3. Arquitectura

A arquitectura do sistema é do tipo cliente-servidor.

Numa arquitectura deste tipo, o cliente é responsável por efectuar pedidos de recursos ao servidor. O servidor trata e avalia os pedidos provenientes de um ou mais clientes, entregando os dados pretendidos a cada um dos clientes.

Na Figura 17, o cliente é um utilizador, que usa um navegador de *internet* para efectuar um pedido a uma aplicação de *internet* presente num servidor, o servidor responde ao cliente, esse mesmo servidor também comunica, se tal for necessário com a memória de armazenamento persistente (base de dados). Mais detalhadamente, o servidor que engloba o *Web* API e o SQL (*Structured Query Language*), deve apenas retornar e receber dados (em que o *Web* API é uma API [17]), isto oferece uma grande flexibilidade, uma vez que os dados não estão ligados a recursos ou métodos. Assim sendo, a API (*Application Programming Interface*) pode receber múltiplas chamadas de diversas aplicações independentes, desde que sejam cumpridos os requisitos de dados para cada função da API. Uma API não tem estado, ou seja os pedidos executados são independentes uns dos outros e cada um deles inclui os dados precisos para

completar a tarefa. A grande vantagem que a API oferece, é que no desenvolvimento futuro de outro tipo de aplicações de âmbito diferente (por exemplo aplicação móvel ou de *desktop*), a API que tem como funções receber e enviar dados, acaba por ser utilizada para obter dados com os métodos anteriormente implementados na primeira aplicação [17].

Logo na aplicação futura (a ser desenvolvida), o único desenvolvimento é o do *front-end*, uma vez que o *back-end (a* API*)* desde logo encontra-se implementado.

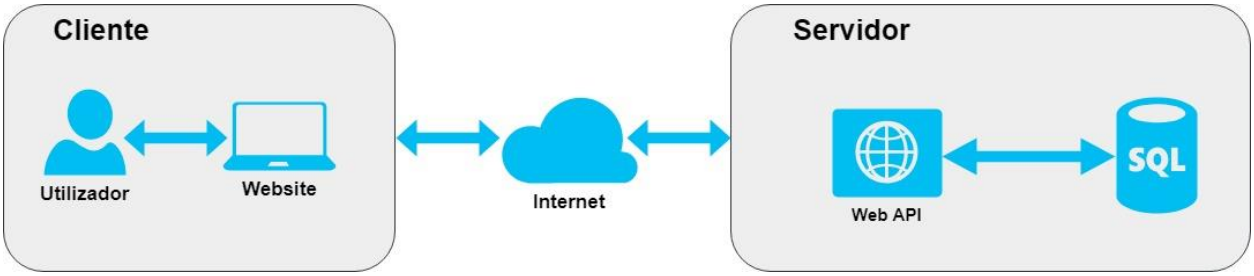

Figura 17 – Arquitectura Geral.

3.4. Descrição das tecnologias

As tecnologias a serem empregues devem depender da análise dos requisitos efectuada neste capítulo, em que interessa ter em conta várias questões, que nos ajudaram na sua escolha.

Os requisitos funcionais e não funcionais devem estar sempre presentes, uma vez que o problema a ser solucionado depende de certas tecnologias existentes no mercado.

O protótipo é um produto criado para um conjunto de partes interessadas, sejam eles utilizadores finais, empresas e terceiros, pelo que é necessário avaliar a forma de tornar a experiência agradável e interessante, ter atenção ao equipamento (*hardware*) utilizado por estas partes, estar atento às tendências tecnológicas e verificar se é possível através da tecnologia escolhida a utilização de código de terceiros (*plugins*) permitindo assim a poupança de tempo na implementação. Tendo em atenção a utilização de tecnologias que sejam gratuitas e que ofereçam uma boa comunidade de suporte e documentação.

3.4.1. A escolha das tecnologias

Para a escolha das tecnologias e com a arquitectura geral na equação, foi efectuada a divisão das tecnologias em quatro categorias fundamentais para o funcionamento da aplicação. Sendo uma aplicação, que tem como requisito ser executada em qualquer sistema operativo com o uso de um navegador de *internet*, as tecnologias direccionadas a esse requisito são as ligadas à *internet* e navegadores. Isto significa que as tecnologias das quatro categorias fazem parte do desenvolvimento de aplicações de *internet*.

Das quatro categorias, unicamente existe comparação na categoria tecnologia de comunicação em tempo real, uma vez que apenas essa foi da responsabilidade do mestrando a análise e pesquisa da solução mais apropriada. Em relação as restantes categorias estão apresentadas algumas vantagens que são uma mais-valia no desenvolvimento do protótipo.

3.4.1.1. Tecnologia do lado do servidor

A tecnologia do lado do servidor é um método para o desenvolvimento de *websites*, em que um pedido de um utilizador é executado num servidor. Este tipo de tecnologia é utilizado para limitar o acesso a dados e também para não permitir a modificação do código fonte por parte destes utilizadores [18].

3.4.1.1.1. *Laravel*

A tecnologia do lado do servidor recaiu sobre a *framework Laravel*, em termos de linguagem na parte do servidor. As razões que ostentam esta escolha são:

- A facilidade e rapidez de aprendizagem.
- Imenso conteúdo, em termos de tutoriais e informação sobre esta tecnologia.
- Uma tecnologia estável e em constante desenvolvimento.
- Grande número de contribuidores.
- Gratuita.

Outras razões mais específicas foram: [19]

- Criação de sistemas de autenticação e autorização: uma aplicação de *internet* precisa de um componente que consiga provar que os utilizadores que utilizam o sistema são o que dizem ser, e por isso impedem os outros que não são autorizados a usar as funcionalidades do sistema. O *Laravel* fornece pré-instalado este sistema de autenticação.

-Funcionalidades de transmissão em tempo real: o *Laravel* tem disponível um grupo de bibliotecas, que possibilitam a aplicação ter alertas/notificações em tempo real, em que o utilizador recebe e envia informações para outros, sem a necessidade de actualizar as páginas através de um *WebSocket*. O grupo de bibliotecas são *Pusher, Redis e Socket.IO* [20].

- Uma classe de mapeamento relacional de objectos: o *Eloquent* ORM (*Object-relational mapping*) [21] está incluído na instalação do *Laravel*, que proporciona uma implementação para as quatro operações básicas aplicadas numa base de dados, o CRUD [22]. Cada tabela tem um modelo que é utilizado para interagir com essa tabela. Os modelos permitem a especificação de relações que existem entre tabelas através de métodos já desenvolvidos, para obter dados de outras tabelas sem o uso de linguagem SQL [23].

- Funções de multi idioma [24]**:** o *Laravel* dispõe de funcionalidades para obter de forma fácil várias palavras de múltiplos idiomas. Para cada idioma existe uma pasta e dentro dela existem um ou vários ficheiros (relativos a diferentes páginas) em formato PHP (*Hypertext Preprocessor*), que retornam um conjunto de *strings* JSON (*JavaScript Object Notation*) com as traduções [25].

- Separação do código de lógica, do código de interface: o padrão de arquitectura de *software* do *Laravel* é o MVC (*Model-view-controller*) [26], o padrão separa o código lógico (PHP) do código de interface (HTML - *Hypertext Markup Language*). Assim as alterações no código da interface não interferem com o código lógico. A separação do código permite uma

melhor organização em termos de estrutura de cada página, obtendo assim simplicidade na procura de erros e na realização de mudanças.

- Técnicas de seguranças a vulnerabilidades: as vulnerabilidades e os problemas de segurança são um tema que deve estar sempre presente no desenvolvimento de aplicações. O *Laravel* ajuda a proteger a aplicação das mais diversas vulnerabilidades, sem que o utilizador necessite de configurar ou implementar código suplementar.

3.4.1.2. Tecnologia do lado do cliente

Na tecnologia do lado do cliente, a programação desenvolvida é executada no navegador de *internet*, para que isso aconteça, este código é transferido do servidor para o cliente, depois o computador do cliente é responsável pela execução deste *script.* De salientar que normalmente a linguagem utilizada nestes *scripts* tem de estar activa nas opções do navegador de *internet* [27].

3.4.1.2.1. *AngularJS*

O *Javascript* iria trazer problemas a longo prazo, dado que é difícil compor uma estrutura organizada entre os vários componentes [28]. A falta de organização origina problemas a detectar e resolver falhas que o sistema eventualmente vai conter, logo a primeira opção para tecnologia do lado do cliente foi o *AngularJS.*

O *AngularJS* foi seleccionado pela sua simplicidade de implementação com o *Laravel* e com o objectivo de tornar o protótipo em algumas zonas das páginas mais interessante para o utilizador, sem ser necessário o carregamento total da página.

Razões principais foram: [28] [29] [30]

- É fácil de aprender: começar a desenvolver com *AngularJS* é intuitivo, com um conjunto de elementos adicionados ao HTML é possível aplicar no imediato algumas funções que o *AngularJS* comporta e verificar o seu comportamento com o mínimo de código.

- Dados bidireccionais: o *AngularJS* concede uma comunicação em duas direcções, isto significa que qualquer mudança de dados que ocorra nas vistas ou no modelo, essas mudanças são automaticamente propagadas para a respectiva vista pelo *AngularJS.*

- Directrizes personalizadas: as directrizes são elementos de HTML que podem ser totalmente novos, ou mudanças a elementos já existentes no HTML. Aos elementos existe a possibilidade de serem adicionadas novas funcionalidades, em termos estéticos ou de lógica.

- Inserção de dependências: é a definição de funções fora e independentes do código presente nos controladores do *AngularJS*. Com as dependências desenvolvidas, estas são introduzidas nos controladores que se pretende que "herdem" as funções dessas dependências. Com tal característica é mais fácil desenvolver, testar e compreender os métodos implementados [31].

3.4.1.2.2. *BootStrap*

A interface de uma aplicação é um dos requisitos mais importantes, em virtude de os utilizadores quando interagem com uma aplicação de *internet* ser por intermédio de páginas com elementos que reagem as acções causadas pelos utilizadores. Se a interface não é apelativa, de fácil compreensão ou mal organizada, os utilizadores não conseguem adaptar-se ao funcionamento da aplicação. O *BootStrap* acomoda de raiz código CSS (*Cascading Style Sheets*), em que a construção da interface da aplicação é produzida sem qualquer obrigação na definição de código CSS. Unicamente é preciso organizar os elementos de HTML e aplicar as classes de CSS pretendidas.

Principais vantagens: [32] [33]

- **Fácil para começar a aprender:** não existe a necessidade por parte do desenvolvedor de dominar o código CSS, e ter de construir e criar definições de código CSS tudo de raiz. Apenas é preciso incluir os ficheiros da biblioteca no projecto e começar a aplicar as classes de CSS pré-definidas.

- **Sistema de "grade":** o *BootStrap* é totalmente *responsive,* com as suas doze colunas e componentes disponíveis. A "grade" dispõe de duas opções, fixa ou flexível (dinâmica) que é configurada com algumas mudanças. Outra característica do *BootStrap* é esconder um certo bloco de conteúdo exclusivamente para alguns dispositivos com um certo tipo de resolução de ecrã.

- **Estilo base nos elementos HTML:** uma vez que uma aplicação de *internet* envolve diversos elementos, tais como listas, tabelas ou botões. Todos os elementos de HTML estão personalizados e melhorados esteticamente pela biblioteca.

- **Lista extensa de componentes:** os componentes são parte essencial na interface de uma aplicação de *internet*, o *BootStrap* acomoda um grupo de menus, paginação, painéis de conteúdo, barras de progresso ou até elementos de notificação. Estes componentes podem ser customizados a qualquer momento adicionando novos estilos.

- ***Plugins* de *Javascript*:** importa realçar que estes elementos não contam com interactividade de raiz, por essa razão estão acessíveis inúmeros *plugins* que acrescentam essa funcionalidade aplicando o Javascript e mantendo o estilo dos elementos.

- **Boa documentação:** nos mais variados *websites* existem tutoriais, documentação e suporte para qualquer problema que surja. O próprio *website* do *BootStrap* dispõe de um conjunto de exemplos para os diversos elementos e componentes.

Por todas as vantagens anteriores, o *BootStrap* foi escolhido como a base da interface da aplicação desenvolvida.

3.4.1.3. Tecnologia de armazenamento persistente

São tecnologias com o objectivo de guardar dados de forma persistente, normalmente em discos rígidos. Estes dados podem ser acedidos posteriormente, usando para esse efeito uma das tecnologias do lado do servidor [34].

3.4.1.3.1. *MySQL*

O facto de na tecnologia do lado do servidor a escolha ter incidido na *framework Laravel*, a primeira opção para o armazenamento persistente teria de ser um tipo de base de dados que o *Laravel* na sua essência suportasse de raiz com um ORM (mapeamento de objectos relacional), sem a utilização de ORMs de terceiros que poderiam estar desactualizadas. Sendo que o *Laravel* dispõe de um ORM próprio, o *Eloquent* [21] e tendo em conta a prática por parte da equipa de desenvolvimento, foi escolhida a base de dados *MySQL*.

3.4.1.4. Tecnologia de comunicação em tempo real

A tecnologia de comunicação em tempo real permite estabelecer uma conexão entre um navegador de *internet* e um servidor, esta conexão é uma ligação persistente entre o cliente e o servidor, possibilitando o envio de dados em qualquer momento, seja por parte do cliente ou servidor [35].

3.4.1.4.1. *Pusher* e *OneSignal*

Dado que, as tecnologias anteriormente escolhidas não possuem as características de aplicação em tempo de real, em que as informações introduzidas e inseridas na base de dados por um dos utilizadores ou por vários devem reflectir-se em todas as aplicações abertas dos restantes utilizadores, sem que este tenha de actualizar a aplicação de forma manual. Este é nos dias de hoje, um requisito fundamental nas aplicações modernas.

A ideia é a aplicação de *internet* do lado do cliente estar desenvolvida em HTML e *Javascript* (ou *framework*) e que interaja com o navegador de *internet* para a criação de uma ligação entre o navegador e o servidor, que é o responsável por fazer o envio das mensagens para os clientes através de uma ligação *WebSocket* [20] [36]. A maioria dos navegadores de *internet* já suporta este tipo de ligações. Este tipo de comunicação fornece uma abordagem mais flexível e robusta em relação ao estar continuamente a verificar se existem alterações em termos de dados na aplicação [20].

Tendo isto em consideração, foram tidas em conta duas alternativas na integração deste tipo de tecnologia no protótipo, a primeira foi uma das opções disponíveis na documentação do *Laravel* relativamente à transmissão de eventos usando o *Pusher* [37] e a segunda o *OneSignal* [38].

Se considerarmos as duas tecnologias de tempo real independentes de qualquer *framework* em que sejam implementadas, o *OneSignal* tem duas vantagens em relação ao *Pusher* que é o número de mensagens e de utilizadores ilimitados. O *Pusher* também é gratuito mas tem um limite de número de mensagens e de utilizadores conectados, a partir desse limite é inevitável aderir a uma mensalidade de um conjunto possível de planos.

Ao ponderarmos o uso de uma das tecnologias de comunicação com o *Laravel*, o *Pusher* é uma das três opções de comunicação em tempo real disponibilizado pela documentação, pelo que a dificuldade de integração torna-se mais fácil e menos demorada, uma vez que a maior parte dos ficheiros precisos já estão inseridos na *framework* e a própria documentação contém exemplos reais.

O *Pusher* é uma Interface de Programação de Aplicações (API) simples, para rapidamente e facilmente integrar funcionalidades de tempo real com comunicação bidireccional via *WebSockets* em aplicações de *internet*, móveis ou qualquer outro dispositivo de *internet*. O *Pusher* fornece um conjunto de bibliotecas que permitem a abstracção da combinação de instruções que ocorrem para o envio e recepção entre cliente-servidor e vice-versa. O modelo do *Pusher* são canais que os utilizadores/clientes "subscrevem" e ficam imediatamente disponíveis para enviar e receber mensagens, utilizando para o efeito os canais respectivos. Os canais são de vários tipos, sejam públicos ou privados, em que a própria tecnologia fornece os mecanismos de autenticação [37].

Escolhido assim o *Pusher* e para uma implementação de acordo com a documentação do *Laravel*, foi utilizada a biblioteca própria *Laravel Echo* que está desenvolvida em *Javascript* e que permite de forma fácil "subscrever" e "escutar" os canais. Esta biblioteca necessita também da instalação da biblioteca do *Pusher*, dado que é a API escolhida para a transmissão de mensagens [20].

3.4.2. Arquitectura com as tecnologias

Após a descrição destas tecnologias, é viável a construção de um diagrama com uma arquitectura mais detalhada de como as tecnologias vão comunicar entre si, o diagrama está representado na Figura 18.

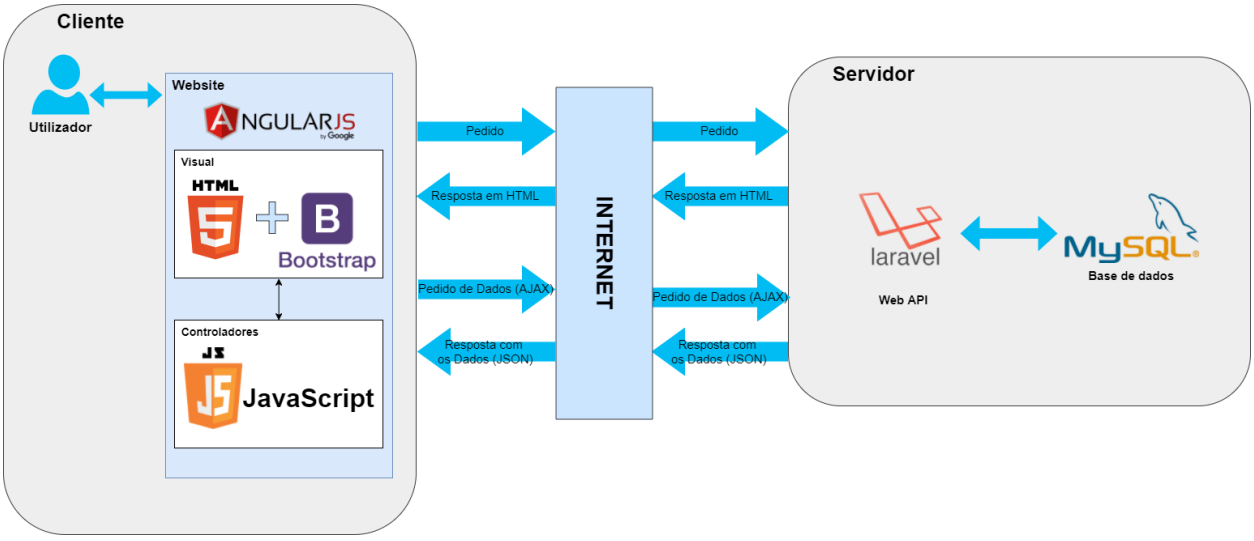

Figura 18 – Arquitectura detalhada.

Como é possível verificar na Figura 18, o navegador de *internet* efectua um pedido ao servidor, que por seu lado envia a página em HTML a ser "lida" pelo navegador.

Em certas áreas do protótipo, o *AngularJS* executa pedidos de dados em chamadas do tipo *Ajax* para o *Laravel*, que por sua vez retorna os dados em formato JSON. Do lado do *AngularJS*, os dados são recebidos e tratados e as respectivas partes do HTML são actualizadas, sem ser necessário o utilizador actualizar a página pela função do navegador de *internet*.

3.5. Estrutura conceptual da base de dados

Analisando num âmbito geral o projecto, a base de dados é um dos componentes mais importantes para o correcto funcionamento do protótipo a ser desenvolvido, em razão de que todo projecto estar dependente dos dados que ali vierem a ser guardados. Toda a especificação de processos de uma organização é efectuada com interacções com as tabelas da base de dados desenhada.

Nas imagens seguintes é apresentado o diagrama OFD da estrutura conceptual da base de dados publicado no artigo da referência [39], que descreve a estrutura conceptual do protótipo, que é um esforço conjunto entre vários colaboradores no âmbito do desenvolvimento do protótipo e são apresentadas as classes necessárias para uma suficiente compreensão do funcionamento do protótipo.

O diagrama OFD está dividido em duas imagens devido a problemas de elegibilidade.

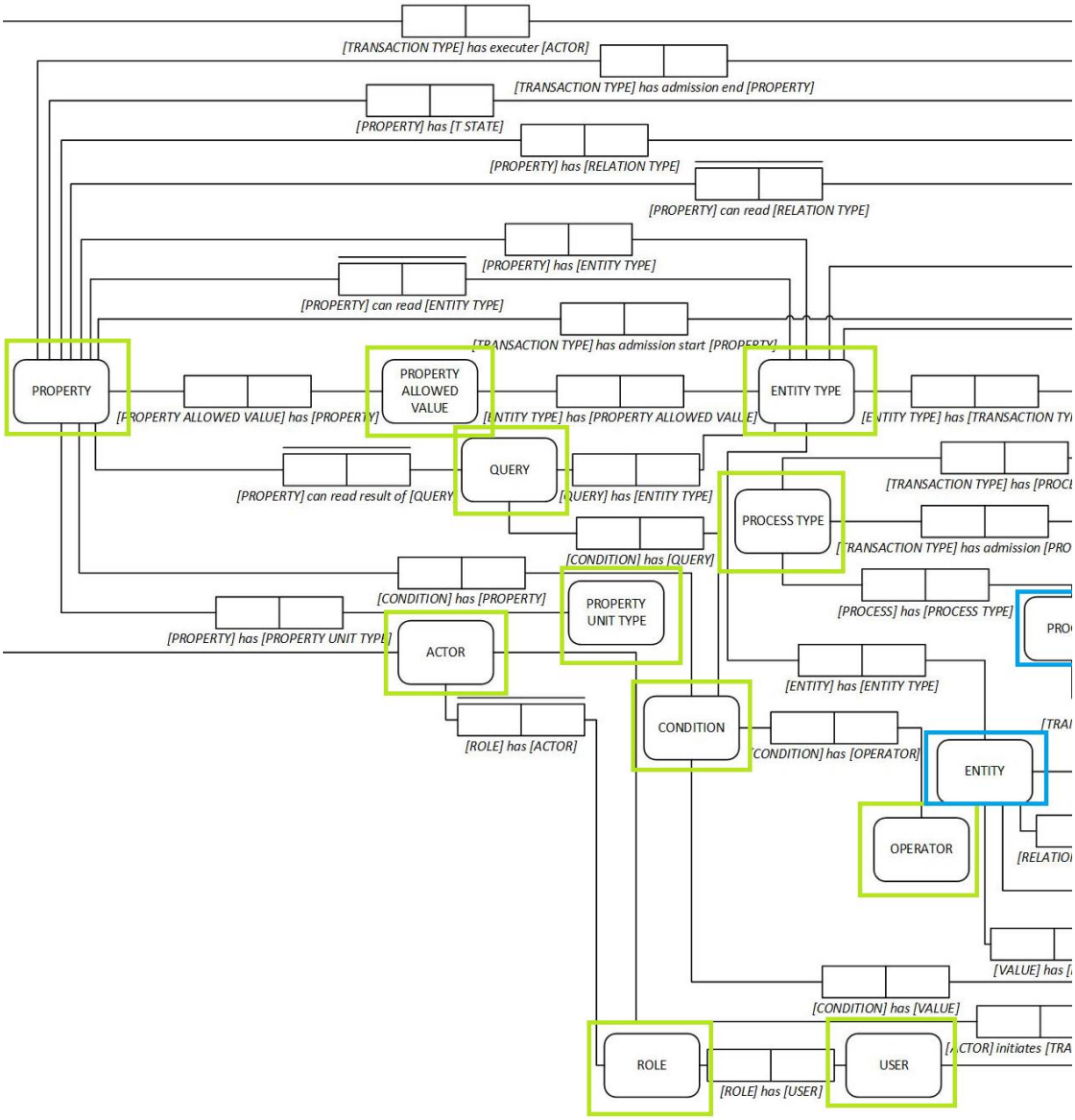

Figura 19 – Primeira parte do diagrama OFD da base de dados [39].

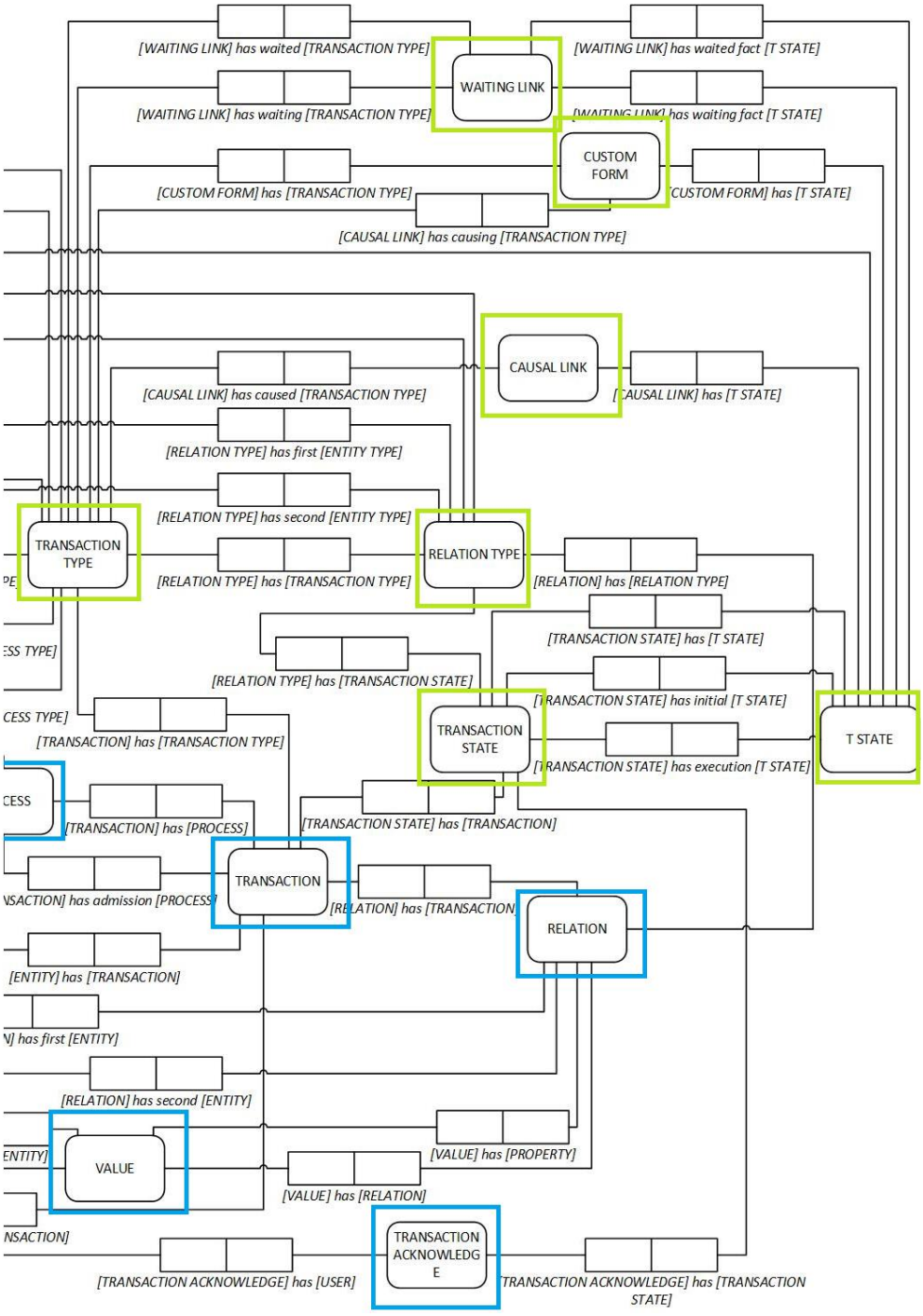

Figura 20 – Segunda parte do diagrama OFD da base de dados [39].

Na Figura 19 e Figura 20, os rectângulos de cor verde representam as tabelas da base de dados responsáveis por guardar a especificação dos processos da organização obtidos através dos diagramas DEMO, enquanto que os rectângulos de cor azul representam as instâncias inseridas, de acordo com os processos que estão a ocorrer no dia-a-dia de uma organização.

3.6. Funcionalidades

Na subsecção de funcionalidades, são exibidas em formato de lista, as funções que o protótipo à data da entrega do relatório tem incorporadas e em funcionamento.

As funcionalidades são divididas de duas formas, implementadas e não implementadas.

Funcionalidades implementadas:

	Mestrando	**Restantes membros da equipa de desenvolvimento**	**Comum a todos (existiu divisão de trabalho)**
Painel de Controlo Dinâmico	✓		
Os cinco tipos de actos principais do DEMO são suportados	✓		
Página de autenticação	✓		
Comunicação em tempo real de notificações	✓		
Protótipo *web responsive* (adapta-se a resolução do ecrã)	✓		
Sistemas de vistos (*acknowledge*)	✓		
Especificar processos de negócios de uma organização			✓
Multi idioma			✓
Validação dos campos do lado do cliente			✓
Criação de formulários visualmente sem código		✓	
Agrupamento de vários formulários num só		✓	
Reordenação de propriedades dos formulários de forma visual		✓	
Gestão de utilizadores		✓	

Funcionalidades não implementadas:

1. Histórico das especificações de processos
2. Possibilidade de editar um valor de uma propriedade
3. Restringir acesso a documentos de outras organizações a outros utilizadores
4. Interligação entre processos, no caso de processos que dependem entre si
5. Tarefas de auto activação, o sistema é que as cria automaticamente
6. Validação dos campos do lado do servidor

7. Tipos de actos do DEMO restantes em falta
8. Possibilidade de guardar um registo de visualizações de utilizadores nas transacções
9. Opção de um dado utilizador final externo poder registar-se na aplicação sem a intervenção do administrador
10. Hipótese de um utilizador da organização realizar uma autenticação e impersonificar uma certa entidade

4. Análise da empresa

O capítulo da análise da empresa contém todos os métodos que foram empregues para a especificação e modelação dos processos de negócio da empresa. Esta modelação foi realizada através de várias entrevistas, com o objectivo de compreender toda a área de negócio da empresa. Com as entrevistas foi possível redigir um texto com a descrição e funcionamento completo da empresa e através da análise deste posterior derivação para os diagramas DEMO.

4.1. Entrevistas

Para a definição e análise dos processos de negócio da empresa, foram conduzidas várias entrevistas aos intervenientes desses processos e por razões de confidencialidade não estão presentes neste relatório. Dos departamentos existentes na empresa, o maior foco foi na área de operações de logística ainda assim, as outras áreas em que se incluem a facturação e recursos humanos, também foram alvo de perguntas para a construção da primeira fase da metodologia DEMO, a análise PIF (*Performa Informa Forma*).

No total foram realizadas 139 perguntas para compreender todas as operações efectuadas pela empresa.

4.2. Descrição e funcionamento completo da empresa

Dada a inexistência de documentos físicos/textos com uma descrição e funcionamento da empresa, foi efectuada uma recolha de informações por meio de entrevistas, perguntas e uma presença física para analisar toda a operação da empresa durante vários meses.

Todos os nomes de pessoas e empresas são fictícios, por uma questão de confidencialidade dos dados de pessoas e outras empresas que estão envolvidas no processo.

Esta secção contém além da descrição e funcionamento, a análise PIF exibida pelas várias cores presentes no texto e que é utilizada para o desenho e modelação dos diagramas DEMO.

A TransSantana é uma empresa de transporte de carga, a maioria dos seus clientes são transitários aos quais existem preços estabelecidos {T30 Execute} para o transporte de carga. {T02 Execute}

Para o transporte de carga existem vários motoristas, divididos em grupos para transporte de gás e contentores, no transporte de gás necessitam de habilitação específica para este transporte, no entanto podem fazer transporte de contentores. Todos os motoristas têm ADR (por exemplo: matérias perigosas) e CAM.

Para o transporte de contentores {T73 Execute}, cada transitário envia por *email* um ficheiro em excel que contém os contentores a serem transportados pela TransSantana {T02 Execute}. Os valores a serem cobrados aos transitários {T71 Execute} estão previamente acordados/tabelados {T30 Execute}. Os contentores prioritários são os refrigerados e os que têm

poseima. Estes contentores com poseima são inspecionados pela Alfândega quando chegam ao cliente final.

Por vezes o transporte de contentores necessita de licenciamento {T67/68 Execute} por parte das autoridades competentes para que possa circular na rede viária porque excede o peso ou tamanho permitido por lei.

Este licenciamento pode ser feito {T67/68 Execute} à câmaras, polícias e etc.

Em casos em que é preciso ajuda no transporte de contentores é pedida ajuda {T43 Request} a empresa **Vicente Transportes ou Moniz Transportes** (transportador terrestre). Os valores a serem pagos a estes prestadores de serviços externos {T82 Execute} estão previamente acordados/tabelados {T30 Execute}. O armador é o proprietário dos navios e de todos os contentores. Quem trata da papelada com a alfândega são os transitários.

Apenas os contentores cheios tem documentação, esta documentação (BL e outro tipo de documentação) em papel é entregue por um estafeta que trabalha para os transitários e colocado no arquivo (escritórios antigos da TransSantana perto do Porto).

A BL (*Bill of Landing*) é um documento emitido/criado pelo armador para os contentores que chegam dos navios. Os contentores vazios não possuem documentação.

Os condutores precisam destes documentos para proceder ao transporte dos contentores.

Na entrada e saída de contentores no Porto é necessária a autorização do armador, esta autorização é anunciada ao Porto (ao programa da ODP) e é feita pelos transitários ao armador.

A TransSantana não tem conhecimento do que está presente dentro dos contentores, essa informação está presente na BL (*Bill of Landing*).

Pela plataforma de "Movimento de Trelas" (*website* da ODP) é possível saber quais os contentores que já foram descarregados pelo operadores do Porto (do navio) no Porto do Seixal, assim como as entradas e saídas dos condutores (com ou sem carga, por exemplo abastecer).

Os clientes (transitários ou os próprios recebedores) podem pedir que os contentores sejam entregues no estaleiro da TransSantana no Porto Moniz {T76 Request}. Raramente os contentores ficam no Porto do Seixal mais do que cinco dias porque é necessário o pagamento pela sua "estadia".

O transporte de contentores pode ser colocado em vários lugares {T76/78 Execute} antes mesmo de ser entregue no local de destino previamente fornecido pelo cliente (transitário). {T02 Execute}

Esta gestão de entregas de transporte de contentor {T01 Execute} é da responsabilidade da Rita que decide por questões de logística ou de ordem dos clientes finais {T01/78 Execute} transportar e colocar os contentor nesses lugares escolhidos . {T78 Execute}

São sempre feitas guias de transporte pelos condutores sempre que estes desloquem carga {T02/76 Execute} ou não {T78 Execute}.

Os motoristas habitualmente já tem camiões predefinidos ou a Rita é que decide que camiões a utilizar {T04 Execute}.

Um empregado procede ao preenchimento de uma folha em excel com informação destas guias de transporte que depois é utilizada (partilha de ficheiro por servidor) pela parte da facturação.

Descrição detalhada

Cada transitário envia por *email* à Rita (ou Bruna) a listagem de chegadas de contentores dos navios {T02 Request} a serem transportados pela TransSantana {T73 Execute}.

Alguns transportes de contentores necessitam de licenciamento {T67/T68 Execute} para poder circular, a Rita é responsável por obter e entrar em contacto {T67/T68 Request} com as autoridades competentes para fazer o transporte de contentor {T73 Execute}.

Por vezes o transporte de contentores necessita de licenciamento {T67/T68 Execute} por várias razões, seja transporte de produtos perigosos {T66 Execute} que precisam de alguma autorização especial ou porque excede o tamanho ou peso permitido pelo código da estrada ou da rede viária, estas licenças são pedidas previamente, seja a câmara, polícia ou outra entidade.

No caso em que a TransSantana não dispõe dos recursos necessários para o transporte de contentores, os serviços são passados {T43 Request} à **Vicente Transportes ou Moniz Transportes,** os serviços são passados através de um contacto telefónico {T43 Request} entre a Rita e o Gustavo (gere a frota da V&T) no caso da **Vicente Transportes** e Lucas no caso da **Moniz Transportes**. A MT só faz os serviços da Serra de Água Produtos Alimentares, nos restantes casos é sempre a F&C que faz esses serviços {T43 Execute}. A esses serviços que foram passados à **V&T**, **Moniz Transportes** ou **Transitários do Norte** a Rita cria uma nota de encomenda no programa de facturação. No caso do Porto Santo, a empresa responsável pelo transporte de contentores ou diversos é feito pela **Transitários do Norte**, estes serviços são sempre passados {T43 Request} implicitamente (sem contacto entre as duas empresas) pela TransSantana à **Transitários do Norte**, dado que a TransSantana não têm recursos físicos no Porto Santo. Neste caso a Rita faz uma nota de encomenda para a **Transitários do Norte**, quando recebe a nota de encomenda feita pelo Guilherme da ViaMar (**Containers)** à TransSantana. A TransSantana paga estes serviços {T82 Execute} à **Transitários do Norte** e a ViaMar (**Containers)** paga a TransSantana, o Guilherme da CNT informa {T02 Request} a **Transitários do Norte** mas existem casos em que informa a TransSantana por *email* dos contentores a serem transportados no Porto Santo {T02 Execute}, se isso acontecer (informar a TransSantana por *email*) a Rita pode de imediato criar as notas de encomenda à **Transitários do Norte** mesmo que não tenha recebido as notas de encomenda por parte da **Containers** (Guilherme). Apenas os contentores cheios é que possuem documentação.

Na chegada de contentores nos navios - O Leonardo da LMPS (Linhas Marítimas Porto Santo) é que vai buscar à mala do navio os documentos relativos aos contentores cheios da Containers. O Enzo (Multitrans) e Gabriel (Atlanticotrans) é que vão buscar ao aeroporto os documentos relativos aos contentores cheios da Multitrans ou Atlanticotrans. Estes documentos são colocados nos antigos escritórios (ao pé do Porto) da TransSantana.

No caso de contentores para exportação (serem carregados nos navios {T22 Execute}), os motoristas entregam a documentação à Rita que por sua vez coloca tudo num envelope que depois o Henrique (TransSantana) ou um motorista recolhe essa documentação (envelope) no escritório da TransSantana e entrega-os à ODP ao pé do Porto. O Leonardo pega nesta documentação que está na ODP e entrega no navio (Containers, Atlanticotrans e NMN). No caso da Multitrans, o Enzo é o responsável por recolher a informação nos antigos escritórios da TransSantana, colocar num envelope e entregá-lo no lugar correcto para ser enviado por avião.

No caso de transporte de contentores do cliente para o Porto do Seixal {T73 Execute} (exportação) a documentação é entregue ao condutor pelo próprio cliente final e este coloca a informação nos antigos escritórios da TransSantana. Nota: Pode acontecer serem chamados aos escritórios da TransSantana e os motoristas passam pelos antigos escritórios, recolhem a sua documentação ou a documentação de todos e entregam nos escritórios da TransSantana.

No caso dos contentores com resíduos (sucata, cartão, papel, vidro e pneus) (RN e CMRJ), em que existem guias do ambiente e talões de pesagem, estes são colocados pela Rita num

envelope à parte. No caso dos resíduos hospitalares (Chão da Ribeira), em que existem guias diferentes e talões de pesagem, estes são colocados pela Rita num envelope à parte. Nos contentores com resíduos diversos e hospitalares, são todos cobrados à NMN {T71 Execute}. Antes de colocar a documentação nos envelopes, a Rita faz digitalizações e verifica se falta alguma {T41/42 Execute} (pedida pelo Rúben da Norte {T41/T42 Request}) documentação dos contentores com guias do ambiente {T41 Execute} e talões de pesagem {T42 Execute} para estas digitalizações serem enviados por *email* para o Rúben da Norte, para que este saiba o conteúdo do contentor e a sua pesagem.

Norte -> RN e CMRJ.

O Leonardo recolhe esta documentação que está em envelopes separados e efectua a mesma operação como descrito em parágrafos anteriores.

As pesagens são feitas em locais diferentes {T70 Execute} dependendo da situação. Quem é responsável por iniciar a pesagem {T70 Request} são os motoristas, no entanto não são eles que realizam a pesagem. {T70 Execute}

Para as autorizações para entrada/saída do Porto, é necessário a autorização do armador {Request} nos seguintes casos:

- Contentores cheios: entrada
- Contentores vazios: saída

Nos seguintes casos não é necessário a autorização do Porto {Request}:

- Contentores cheios: saída porque possuem as BLs.
- Contentores vazios: entrada.

Se o motorista não for autorizado a sair, este informa por telemóvel a Rita que por sua vez entra em contacto por telemóvel com o cliente transitário para este contactar o armador, e o armador permitir (anunciar). Nota: se for um caso de extrema urgência o transportador pode entrar em contacto por telefone directamente com o armador (Norte).

	Entradas	**Saídas**
Contentores cheios	Sim	Não
Contentores vazios	Não	Sim

Nota: Sim e Não para designar se é preciso autorização.

Entradas:

- Contentores cheios - o transitário entra em contacto por *email* com o armador, o armador informa o Porto através do programa da ODP;
- Contentores vazios - não é preciso autorização;

Saídas:

- Contentores cheios - não é necessário autorização porque os contentores têm BL;
- Contentores vazios - o transitário informa o transportador que precisa de contentores vazios, o transitário entra em contacto com o armador para pedir a saída dos contentores e o armador envia um *email* para os transitários, transportador e Porto (ODP).

Não é necessário a autorização do armador (Norte) para ir buscar um contentor vazio a um cliente final mas na mesma a Rita informa por telefone o armador, desde que o contentor esteja fora do Porto a comunicação não necessita da intermediação do transitário (transitário -> armador) a informar o armador que precisa daquele contentor. O objectivo do armador ser informado de que a TransSantana vai buscar um contentor a um cliente final é para haver um controlo dos contentores da sua parte.

Existem casos em que já se sabe que depois de ir buscar aqueles contentores cheios ao cliente final {T02 Execute} é necessário levar x contentores vazios para lá {T02 Execute}. (RN e CMRJ)

Os contentores com guias do ambiente normalmente não têm problemas na entrada do Porto, visto que a comunicação é feita directamente com o armador (Norte) sem a intermediação do transitário. Normalmente a Rita pede ao Rúben da Norte autorização para a saída de contentores vazios do Porto (sucatas, RN e CMRJ), na entrada de contentores cheios normalmente isso não acontece.

Em termos de avaliações visuais {T08,T09... Execute}, os contentores apenas são avaliados visualmente {T08,T09 Execute} pelos motoristas quando são colocados em cima da trela/sideloader. No caso de serem carregados no local do cliente *às vezes* é feita a avaliação antes de carregar de o contentor. {T09 Execute}

Os casos em que os contentores são avaliados visualmente {T08,T09…. Execute} são os seguintes, quando vão carregar um contentor cheio no Porto {T08 Execute} e no local do cliente final {T09 Execute} e carregar um contentor vazio no Porto {T08 Execute} e no local do cliente final {T09 Execute}, esta última hipótese raramente acontece.

Segue a seguinte tabela de avaliações visuais de contentores normais possíveis.

Tipo	**No Porto**	**No cliente**
Carregar contentor cheio	Sim	Sim
Carregar contentor vazio	Sim	Sim (raro acontecer)
Descarregar contentor cheio	Não	Não
Descarregar contentor vazio	Não	Não

Nota: Nestas avaliações não é feito qualquer registo.

No transporte de contentores de gás {T72 Execute} da empresa BoaVentura Gás, devem ser feitas avaliações visuais dos contentores no Porto e no UDG {T06/T07/T63/T64 Execute} (destino final), sendo que estas são registadas num papel na qual existem duas versões, uma para o transporte de contentores cheios e outra para vazios, isto quer dizer que os motoristas preenchem esse papel quando carregam o contentor de gás cheio no Porto {T21 Execute} e quando "carregam" o contentor de gás vazio no UDG {T23/T24 Execute}.

No Porto e se houver algum problema no contentor a ser carregado (ver seguinte caso), o motorista deve ligar por telefone à Rita que em seguida deve ligar ao cliente final (ou transitário, no entanto os transitários preferem a primeira opção) para saber se podem prosseguir com a

entrega do contentor, e alertar a ODP para fazer um relatório do caso, esse relatório é depois enviado por *email* pela Rita para o transitário.

Também cabe aos motoristas fazer a verificação do estado dos selos dos contentores cheios {T13/T14 Execute} quando estes são carregados no Porto, deve ser verificado também se os números dos selos colocados nos contentores correspondem aos números presentes na BL (contentores normais) {T18 Execute} ou "ficha de verificação visual contentores criogénicos GNL" (contentores de gás) a que corresponde esse contentor. {T15/T17 Execute}

Os contentores normais tem até três selos, existindo apenas dois (verde para vazio, amarelo para cheio) nos contentores de gás (BoaVentura Gás).

Quando os contentores (normais e gás) são carregados em cima da trela/side-loader {T21,T23,T24 Execute}, deve ser sempre verificado o estado dos punhos {T11 Execute} e se estes estão devidamente fechados. {T10 Execute}

O responsável por criar os selos é o armador, ou seja a NMN (Navegação Marítima da Madeira).

Quando o cliente final enche os contentores com carga , tanto o motorista ou o cliente final pode colocar os selos nesse contentor.

O cliente final pode pedir os selos directamente à Norte ou pedir à TransSantana que entregue um "pacote" de selos através dos seus motoristas. É comum os motoristas levarem um pacote de selos quando entregam contentores vazios a um cliente final.

Os contentores vazios que saem do Porto normalmente tem um selo branco, isto quer dizer que estão limpos, no entanto quando este não existe, o motorista ou cliente final pode verificar se o contentor está limpo. {T83 Execute} Para o cliente final este selo não tem qualquer significado, dado que são apenas um controlo para a transportadora.

Sempre que os motoristas desloquem-se com ou sem carga (contentor vazios ou cheios) {T02/78 Execute}, devem proceder ao preenchimento de uma guia de transporte (contentores normais, de sucata e resíduos hospitalares) ou nota de entrega (no caso da LMPS – Linhas Marítimas Porto Santo). Na nota de entrega estão todos os movimentos por cada viagem do Golfinho, isto quer dizer que todos os movimentos realizados na rampa antes da viagem do Golfinho são colocados nesta folha. (se forem vários condutores (raramente acontece) a fazerem os movimentos na rampa fica tudo na mesma folha, escrevem condutor1/condutor2).

No caso do transporte da cisterna (que têm uma substância química) {T69 Execute}, a cisterna é do armador mas a substância presente na cisterna é da Multitrans (transitário), a Rita envia os talões de contagem para a QDM e para a Multitrans. A QDM é que comunica a TransSantana a distribuição semanal (descargas a realizar) {T69 Request}. O transporte é cobrado a Multitrans {T71 Execute}. Existe um preço fixo independente do número de descargas, ou seja só é cobrado quando fica vazia. {T71 Execute}.

Todos os condutores fazem uma folha de ponto se por exemplo os condutores ficarem a ajudar alguém no estaleiro, neste caso não existem guias de transporte mas colocam essa informação na folha de ponto.

No transporte dos contentores {T73 Execute} se forem autuados pela autoridades competentes, o motorista deve ficar com o auto de contra ordenação e em seguida informar a Rita que depois envia *email* aos transitários a explicar a situação ocorrida.

No caso de reclamações relativas ao transporte de contentores esta pode ser feita directamente {T32 Request} com a Rita por telefone, que pode dar uma justificação e o cliente final aceitar essa justificação. Se isso não acontecer a Dra. Sofia ou Eng. Miguel devem ser

informados para averiguar o que realmente aconteceu {T33 Request}. A reclamação também pode ser feita pelo cliente final ao transitário que por sua vez contacta e informa a TransSantana para averiguações.

É feita uma gestão dos motoristas {T40 Execute}, dos seus tipos de carta, fim de validade e alturas de renovações. Os motoristas devem ser os responsáveis {T58 Execute} por estar atentos a validade dos documentos que precisam para exercer a profissão de motorista.

A empresa tem de dar horas de formação aos motoristas, em vez de dar formação é a empresa que paga a renovação dos documentos necessários para exercer a profissão, no entanto a 1ª vez que tiram esse documento o seu pagamento é da responsabilidade do condutor.

Na gestão dos equipamentos {T65 Execute} é necessário verificar a validade das inspecções obrigatórias (IPO) e ADR {T60 Execute} para saber quando devem ser realizadas as respectivas renovações. Ver as validades é da responsabilidade do Eng. Eduardo (Metalúrgia), mas o Manuel (motorista) e Rita também verificam às vezes.

Existe uma tabela de preços de trajecto para cada cliente (transitário) {T30 Execute}. Pode ser iniciada a qualquer altura uma negociação/renegociação entre a TransSantana e o cliente (transitário). {T29 Request}

Para orçamentos {T27 Request}, o contacto pode ser feito por *email*, telefone para a Rita, no entanto quem define os valores é a Dra. Sofia ou Eng. Miguel. A negociação também pode ser feita {T27 Request/Promise} directamente com a Dra. Sofia ou Eng. Miguel. A decisão de aceitar o orçamento {T28 Execute} é da responsabilidade do Eng. Miguel ou Dra. Sofia se o primeiro estiver ausente por motivo de férias.

Parte da facturação

Com a informação das guias de transporte preenchidas pelos condutores e colocadas na folha de excel pela Rita, a D. Isabela preenche as folhas de excel divididas pelos condutores, matrículas (camiões) e clientes (empresas). Todas estas informações são divididas por semanas de cada mês, em que existe um ficheiro de excel de cada mês e várias folhas das várias semanas preenchidas consoante os serviços prestados (ex: guias de transporte). Nestas folhas são colocadas as despesas, receitas e o saldo. Toda esta informação dos serviços prestados é agrupada por cliente (empresa) em que é possível ver que valor de vendas é que a TransSantana tem para com esse cliente naquele mês. No caso da Multitrans existe uma divisão pelo excel dos vários tipos de serviços (por ex: grupagem, paletes) prestados por indicação do cliente.

A D.Isabela insere as notas de encomenda aos fornecedores com os valores que foram acordados previamente(existentes numa tabela) no Outono e consoante as informações da D.Rita, a D.Bruna recebe as facturas em papel de quem não tem facturação electrónica e tem de verificar as facturas das empresas subcontratadas (fornecedores). Tem de fazer um mapa em excel da passagem de serviços destas empresas. Verificar se os serviços que foram facturados por essas empresas subcontratadas foram requisitados pela TransSantana. Estas facturas podem ser enviadas pelos fornecedores para a TransSantana por correio ou *email*.

No caso de haver um erro (ex: falta do iva na factura) nos serviços contratados (factura), os fornecedores tem de ser contactados seja por *email*, telefone a informar sobre o erro para que estes emitam uma nota de crédito para anular essa factura.
No caso de estar tudo correcto as facturas são digitalizadas pela D.Bruna e enviadas por *email* para o portal Facturas, este encarrega-se de criar automaticamente as guias de entradas referentes a essas facturas. Na etapa final a D.Bruna interliga as guias de entrada com a nota de encomenda correspondente.

Uma nota de encomenda é feita pela TransSantana mais propriamente pela Rita esta é considerada como um serviço externo. A empresa subcontratada (fornecedores) faz uma factura que corresponde a essa nota de encomenda. A factura dos fornecedores é enviada pelo "Facturas" para o Outono da TransSantana, que insere neste sistema como guia de entrada (ex: quantidade, valor e desconto), alguns destes campos são verificados pelo Eng. Miguel e outros pelo pessoal responsável pela parte Administrativa. O Eng. Miguel em seguida verifica se existe uma nota de encomenda pedida a essa empresa referente a essa guia de entrada e no caso de haver uma correspondência entre estes dois, dá autorização para converter esta guia de entrada numa factura "final" (que teoricamente é igual a factura inicialmente enviada pelo fornecedor), esta facturada indica o valor que tem de ser pago pela TransSantana ao fornecedor (empresa subcontratada), em caso de anomalias ou erros nas guias de entrada, a TransSantana contacta o fornecedor que entretanto emite uma nota de crédito para anular a factura original.

4.3. DEMO

Por razões de confidencialidade, os diagramas completos não estão presentes neste relatório, no entanto são apresentadas algumas partes destes e uma breve explicação do funcionamento do processo mais importante da empresa na secção 4.3.1.

4.3.1. Introdução

A análise PIF está apresentada e realizada na secção "Descrição e funcionamento completo da empresa", juntamente com a descrição da empresa. A análise PIF é a primeira fase para obtenção das transacções e respectivos nomes.

Com a análise PIF foram obtidas 83 transacções que pertencem a vários processos, contando com transacções que já estão a funcionar em sistemas desenvolvidos por outras empresas de *software*, mas que são essenciais para o fluxo de tarefas no seu todo.

Em seguida é apresentada uma breve explicação geral do fluxo do tipo de processo Transporte e o seu funcionamento no protótipo:

Para a prestação de qualquer transporte, um determinado cliente tem de requisitar um serviço à empresa, com a iniciação de uma transacção "Realização de transporte" (ver Figura 21), ao mesmo tempo tem de preencher um formulário e o seu conjunto de campos, escolhendo também o tipo de material a ser transportado.

Após esta requisição de serviço de Transporte, é automaticamente gerado um novo processo que fica associado a este cliente.

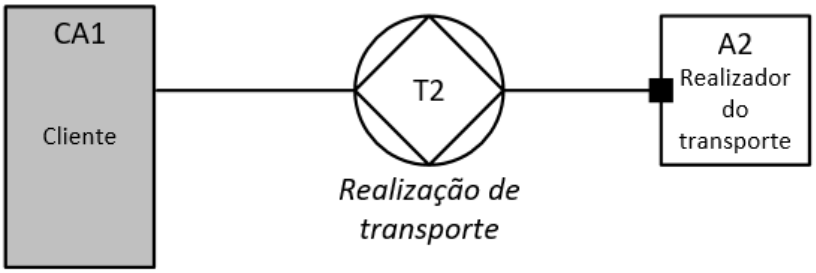

Figura 21 – Excerto do diagrama ATD da parte do processo Transportes.

Todas as transacções que são necessárias para a realização e finalização do serviço requisitado são associadas a este novo processo.

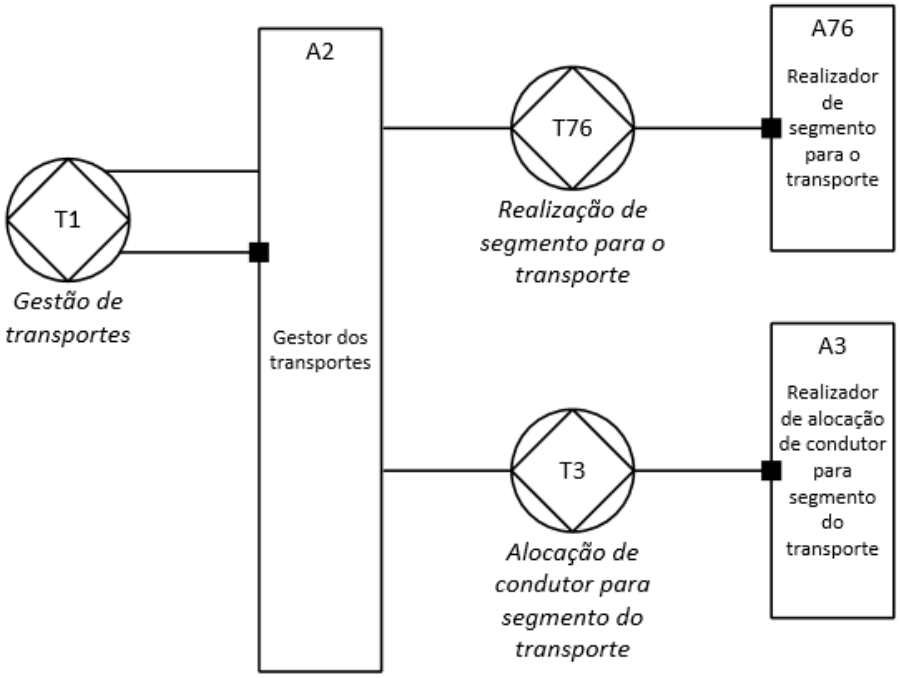

Figura 22 – Excerto do diagrama ATD do actor "Gestor dos Transportes".

Após a transacção "Realização de Transporte" ser iniciada, existem um conjunto de transacções necessárias que são iniciadas por parte da pessoa responsável da logística, para que o processo relativo a esse transporte seja concretizado/finalizado.

Podemos considerar que uma transacção do tipo "Gestão de transportes" é responsável por iniciar esse conjunto de transacções mencionado no parágrafo anterior.

Como é possível verificar pela Figura 22, o actor "Gestor dos Transportes" (pessoa designada para iniciar esta tarefa/transacção), é responsável por iniciar essas duas transacções (T3 e T76), uma para definir o segmento de percurso a ser realizado (T76) e também escolher o condutor que irá realizar esse transporte (T3).

Aliado a estas transacções, estão algumas de avaliação visual do transporte requisitado (não estão presentes nas figuras), que deve ser efectuado pelos motoristas no local da recolha da carga.

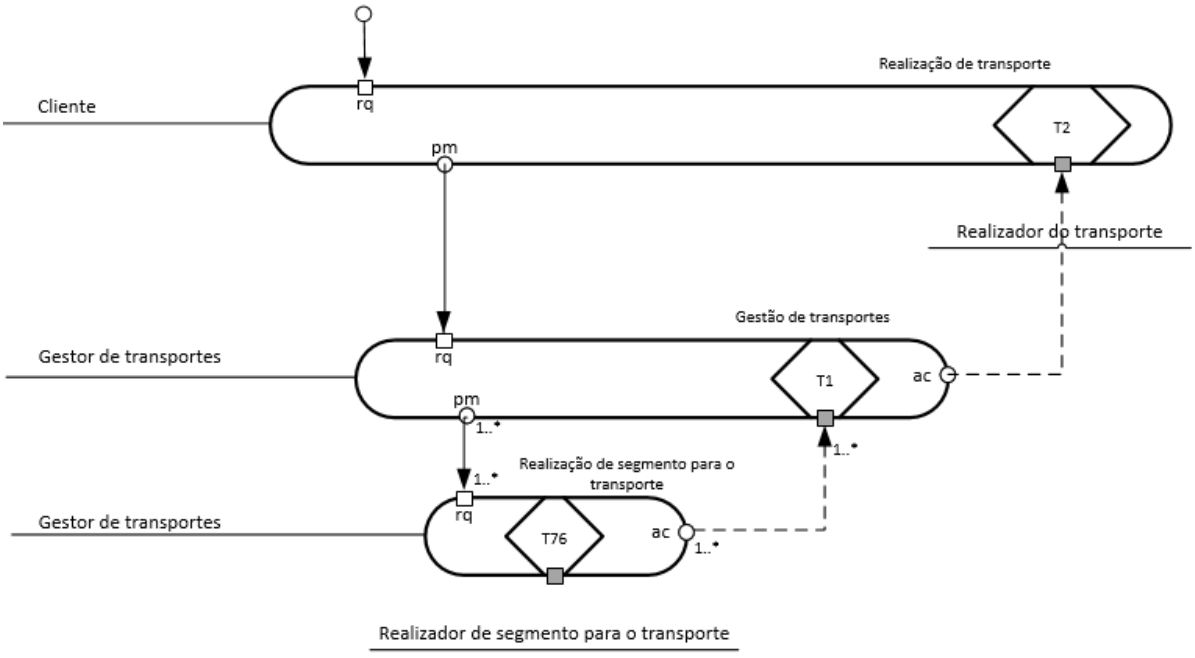

Figura 23 – Excerto do diagrama PSD da parte do processo Transportes.

Na Figura 23, a transacção "Realização de Transporte" só termina (a Ligação de Espera no acto de execução não permite fazer o acto seguinte da transacção) quando a transacção "Gestão de transportes" (T1) e todas as transacções em baixo (a transacção "Realização de segmento para o transporte" – T76), estão completadas/realizadas, isto significa que é possível analisar quais os processos ainda em falta e controlar o estado dos mesmos facilmente no protótipo.

Após a conclusão de um dado processo este deve passar para o estado finalizado.

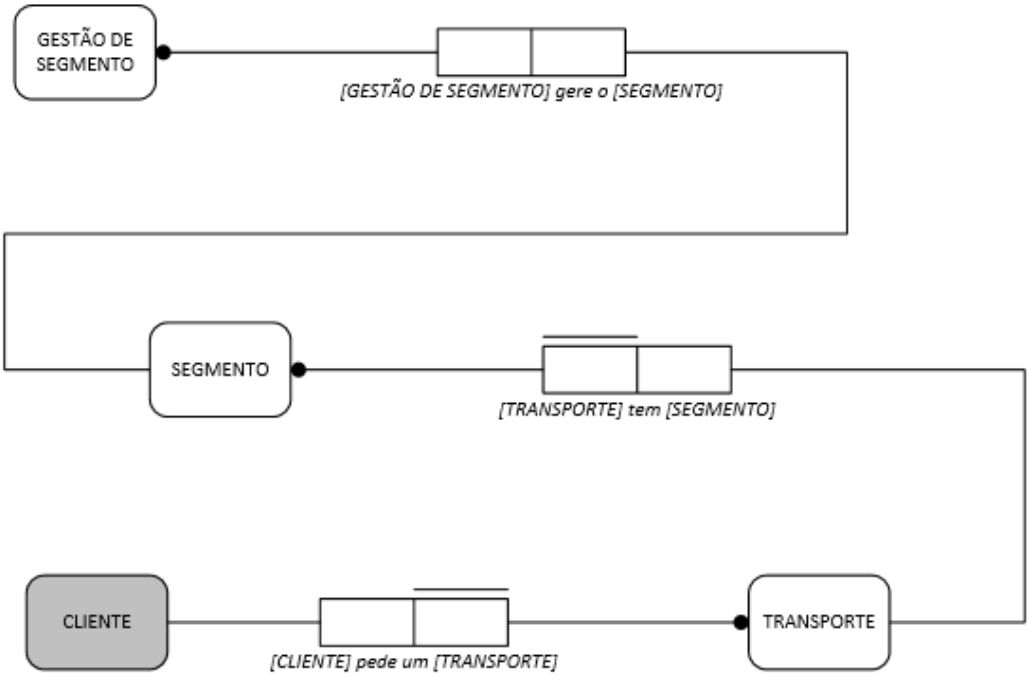

Figura 24 - Excerto do diagrama OFD da parte do processo Transportes.

Na Figura 24, está exibido um excerto do diagrama OFD, em que existem quatro classes, e os respectivos tipos de facto associados a estas classes. No tipo de facto "[CLIENTE] pede um [TRANSPORTE]" existem factos, como por exemplo, Cliente X pede um Transporte Y.

Em relação ao OPL foram identificadas 109 propriedades que fazem parte das várias classes especificadas no OFD.

4.3.2. Desafios em aberto

Com a realização das entrevistas e da descrição e funcionamento da empresa foi possível obter várias conclusões, que se traduziram em vários desafios em aberto na especificação dos processos da organização utilizando a metodologia DEMO.

Nos parágrafos seguintes são listados vários desafios em aberto.

1. **Decisão da existência de avaliações visuais.**

Feita a análise das tarefas efectuadas pelos motoristas num dado transporte foi conseguido determinar que são realizadas avaliações visuais ao transporte estipulados, em vários momentos da realização do transporte. No momento, não existe um sistema que controle e possua um histórico de avaliações visuais elaboradas.

O desafio reside na escolha destas avaliações visuais tornarem-se em um registo inserido no sistema, dada a quantidade de trabalho que já têm de realizar. Estes registos permitem um maior

controlo nos vários transportes. Foi constatado também que as avaliações visuais devem ser separadas em dois tipos, ao equipamento e ao transporte em concreto.

2. **Que campos devem ter as avaliações visuais.**

No caso de existirem registos das avaliações visuais é necessário a especificação dos formulários e de todos os campos que cada deve conter.

Com a existência destes registos, não foi ainda bem especificado que campos os formulários devem conter.

3. **Na realização dos pedidos de transporte**

No momento actual em que não existe sistema, na maior parte das situações são utilizadas folhas de Excel para se manter os registos dos pedidos efectivados. Em determinados cenários os pedidos de transporte podem ser pedidos por SMS ou até mesmo por *email*, as pessoas responsáveis por estes pedidos variam consoante o tipo de transporte.

O desafio está no facto das pessoas responsáveis por realizar alguns destes tipos de pedidos, nem serem os clientes a quem a organização vai proceder a facturação do serviço prestado. Isto traz problemas na definição de quem é responsável por fazer estes actos, tendo em conta a carga de trabalho associada.

4. **Na gestão documental das guias do ambiente**

Se o tipo de transporte contiver guias do ambiente, as mesmas devem ser digitalizadas e enviadas para o cliente que requisitou o serviço. Actualmente, as guias são enviadas por *email* em formato PDF. As guias do ambiente de vários transportes são agregadas e enviadas todas de uma vez por *email*, perdendo assim o rasto de qual guia do ambiente pertence a um dado transporte realizado.

O desafio está no facto de na implementação do sistema, a pessoa responsável por associar/anexar estes documentos a cada um dos tipos de transporte não ser algo eficiente, atrasando assim as restantes tarefas que tem de concluir.

5. Implementação

Neste capítulo, o foco principal e detalhado é dado aos componentes desenvolvidos pelo mestrando, ilustrando e explicando para que serve e como funciona cada um dos componentes implementados, assim como as versões e alterações efectuadas ao longo do tempo. Os componentes que foram desenvolvidos pelos restantes elementos da equipa de desenvolvimento são mencionados e explicados de forma breve, uma vez que são necessários para o funcionamento dos restantes componentes, sabendo que existe relações e dados que são necessários entre estes.

Como explicado em capítulos anteriores, o protótipo é dividido em duas funções principais, a primeira função (Modelação) que é a configuração e especificação de acordo com os diagramas obtidos com a abordagem DEMO. A segunda função (Execução) é a geração e criação automática de interface consoante as especificações e configurações realizadas nos componentes da primeira função (Modelação) e execução do fluxo de processos.

À primeira função (Modelação) correspondem os seguintes componentes:

- Gestão de Processos -> Tipo de Processo
- Gestão de Transacções -> Tipos de Transacção em que se inclui o *T State*
- Gestão de Entidades -> Tipos de Entidade
- Diagrama de Estrutura do Processo -> Ligações Causais e Ligações de Espera

À segunda função (Execução) correspondem os seguintes componentes:

- Painel de Controlo
- Login
- Comunicação entre clientes - *Pusher*

5.1. Estrutura e organização de desenvolvimento

5.1.1. Estrutura

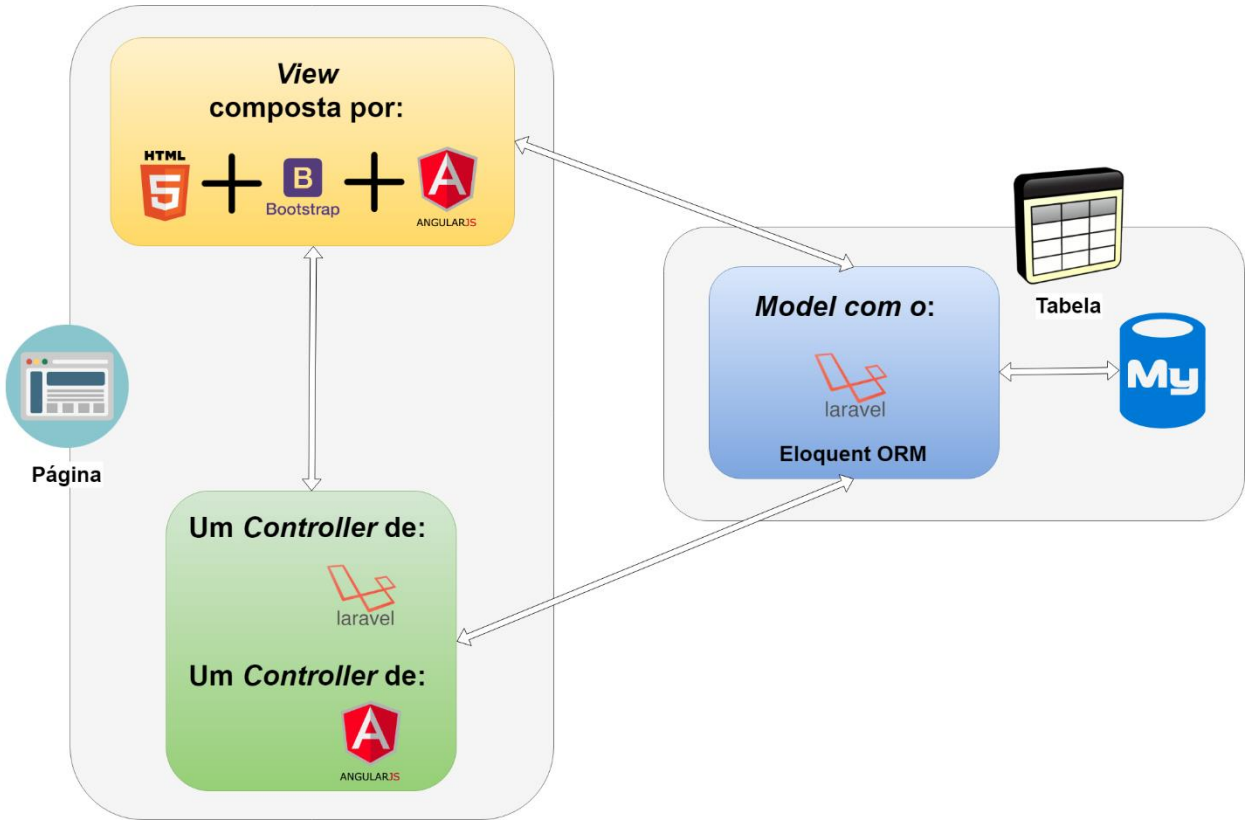

Figura 25 – Estrutura e organização dos ficheiros de código.

Como é possível observar na Figura 25, para cada página existe uma *view*, um *controller* de *Laravel* e um de *AngularJS*.

A *view* de cada página é composta por HTML e pelas directivas do *AngularJS* e com a personalização de interface a cargo da biblioteca de *BootStrap*.

Existe comunicação entre a *view*, *controller* e o *model*.

Os pedidos são realizados pelo *AngularJS* ao *Laravel*, este é responsável por obter as informações da base de dados e por todas as alterações feitas.

5.1.2. Organização do desenvolvimento

Dado que vários colegas desenvolveram partes do protótipo, todas as semanas existiam reuniões de grupo juntamente com o orientador, para verificar o trabalho realizado durante a semana.

Antes destas reuniões com o orientador, os colegas que participaram neste projecto juntavam-se e era efectuada a integração dos vários módulos desenvolvidos no protótipo principal, para posterior demonstração na reunião com o orientador.

5.2. Primeira função (Modelação)

5.2.1. Gestão de Processos – Tipos de Processo

Nesta página (ver Figura 26), são definidos os Tipos de Processo que compõem o conjunto de processos que uma determinada empresa abrange na sua área de negócio. O objectivo do Tipo de Processo é a definição da estrutura que a segunda função (Execução) do protótipo utiliza para a criação das instâncias dos processos que ocorrem dentro de uma empresa.

Tendo por base o exemplo da empresa ligada a este projecto, sempre que um cliente inicia uma transacção designada de "Realização de transporte", a segunda função (Execução) do protótipo tem de obrigatoriamente de criar uma nova instância de processo automaticamente e essa transacção criada é associada a esse processo, como também as restantes transacções que fazem parte do mesmo tipo de processo.

Versão Final

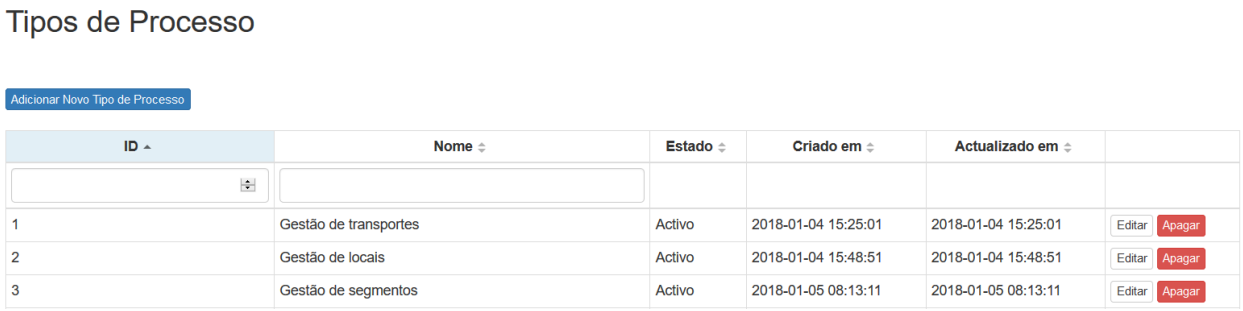

Figura 26 – Exemplo de listagem de tipos de processo da empresa.

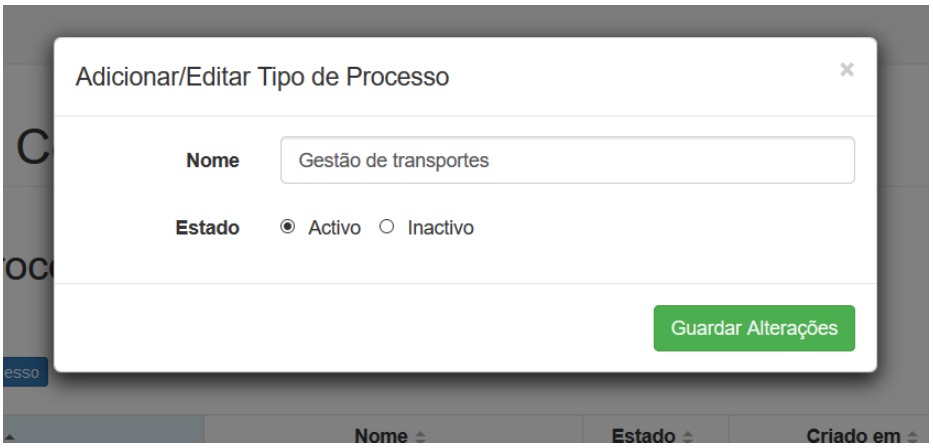

Figura 27 – Formulário para criação/edição de um tipo de processo.

5.2.2. Gestão de Transacções

Nesta secção, a gestão é separada em dois componentes, que são os Tipos de Transacções e os Actos de Transacção (*T State*).

No primeiro componente são definidos os tipos de transacções, que são capturados pelos diagramas DEMO, um Tipo de Transacção está sempre associado a um Tipo de Processo e a um executor. A um Tipo de Processo porque uma transacção só pode exclusivamente pertencer a um dos conjuntos de Tipos de Processo existentes, para que se possa associar a transacção gerada a uma instância de processo. A um executor porque seguindo o paradigma dos modelos DEMO, uma transacção tem apenas um e só um executor, que é responsável pelos actos de transacção padronizados pelas teorias e modelos. O Tipo de Transacção também deve conter se este origina a geração de uma nova instância de processo ou se deve ser associada a uma instância de processo já existente. Da mesma forma a transacção pode ser de auto activação, isto significa que a instância da transacção deve ser gerada sem a intervenção humana usando para isso mecanismos automáticos.

As imagens seguintes (Figura 28, Figura 29, Figura 30, Figura 31, Figura 32 e Figura 33) exibem as várias versões deste componente que foram realizadas até a versão final.

Versão 1

Type your se | Type yo | Type your searc | Type your searc | | Type your searc | Submeter pedido

Process Type	ID	Name	Result Type	State	Created_at	Updated_at	Deleted_at	Executer	Add New Transaction Type
Gestão de locais	2	Criação de locais	O local foi criado	active	2018-01-04 15:49:34	2018-01-04 15:49:34		Criador de local	Edit History Delete
Gestão de transportes	3	Realização de segmento para o transporte	O segmento para o transporte foi realizado	active	2018-01-05 08:20:20	2018-01-11 17:24:35		Realizador de segmento para o transporte	Edit History Delete
	1	Realização de transporte	O transporte foi realizado	active	2018-01-04 15:33:15	2018-01-04 15:33:15		Realizador do transporte	Edit History Delete

Figura 28 – Exemplo de listagem de Tipos de Transacção.

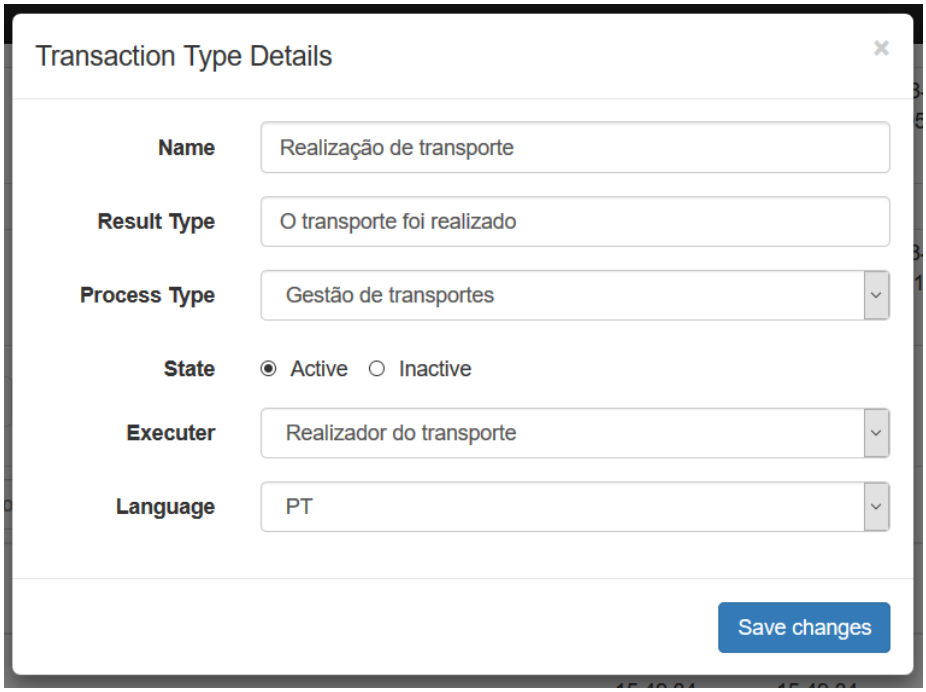

Figura 29 – Formulário para criação/edição de um Tipo de Transacção na versão 1.

Versão 2

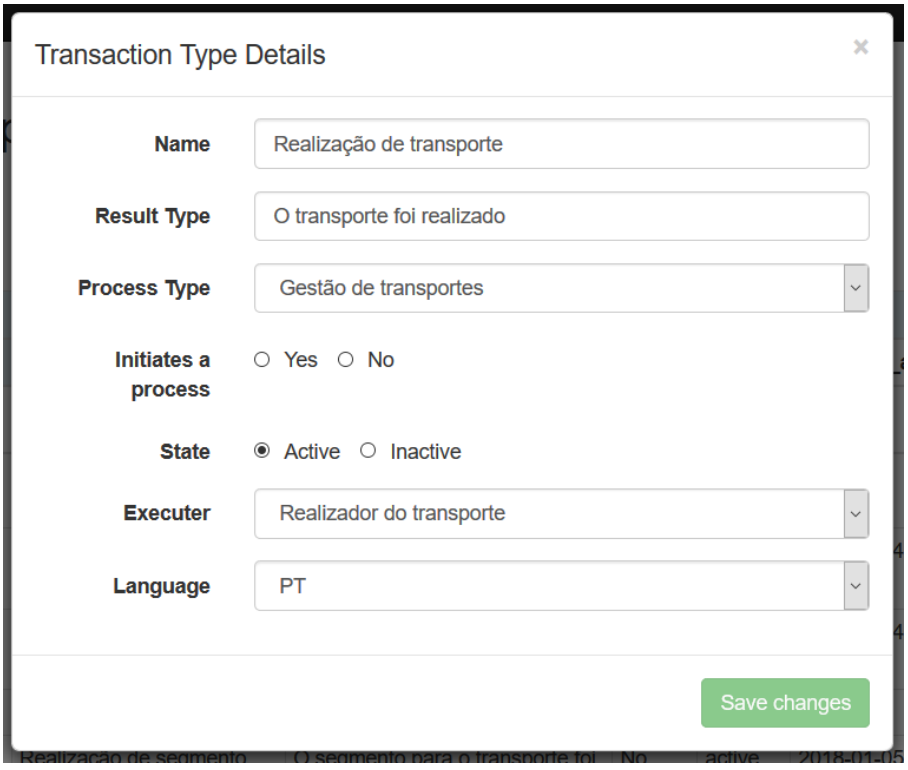

Figura 30 – Formulário para criação/edição de um Tipo de Transacção na versão 2.

Versão 3

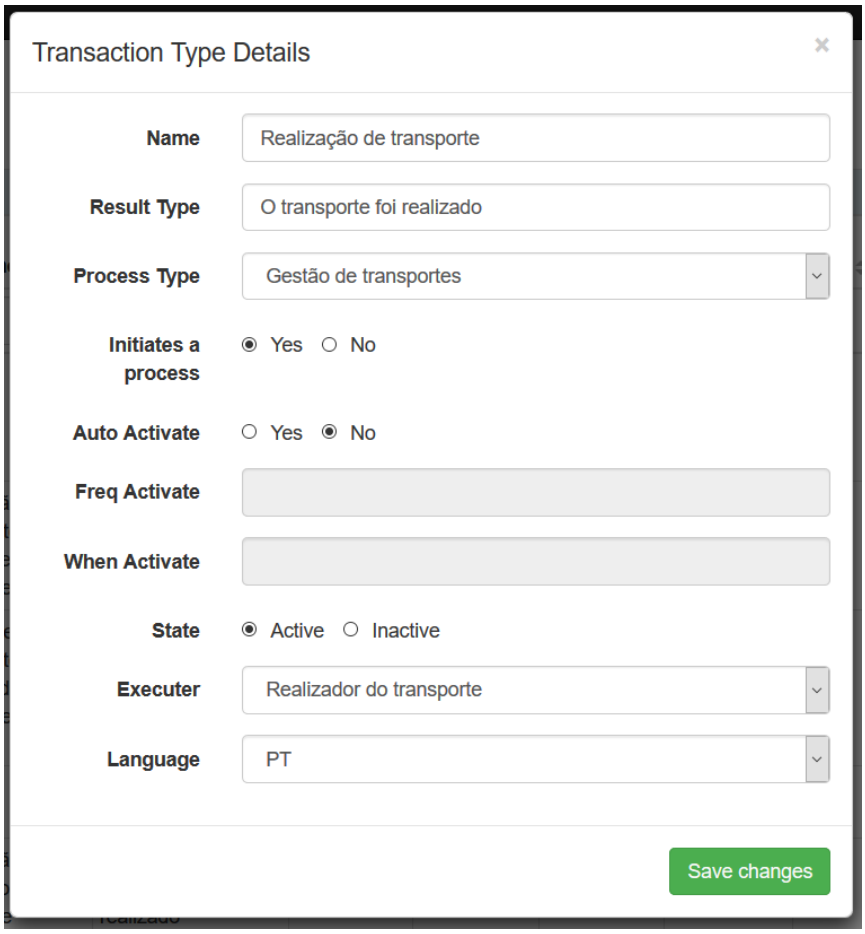

Figura 31 – Formulário para criação/edição de um Tipo de Transacção na versão 3.

Versão final

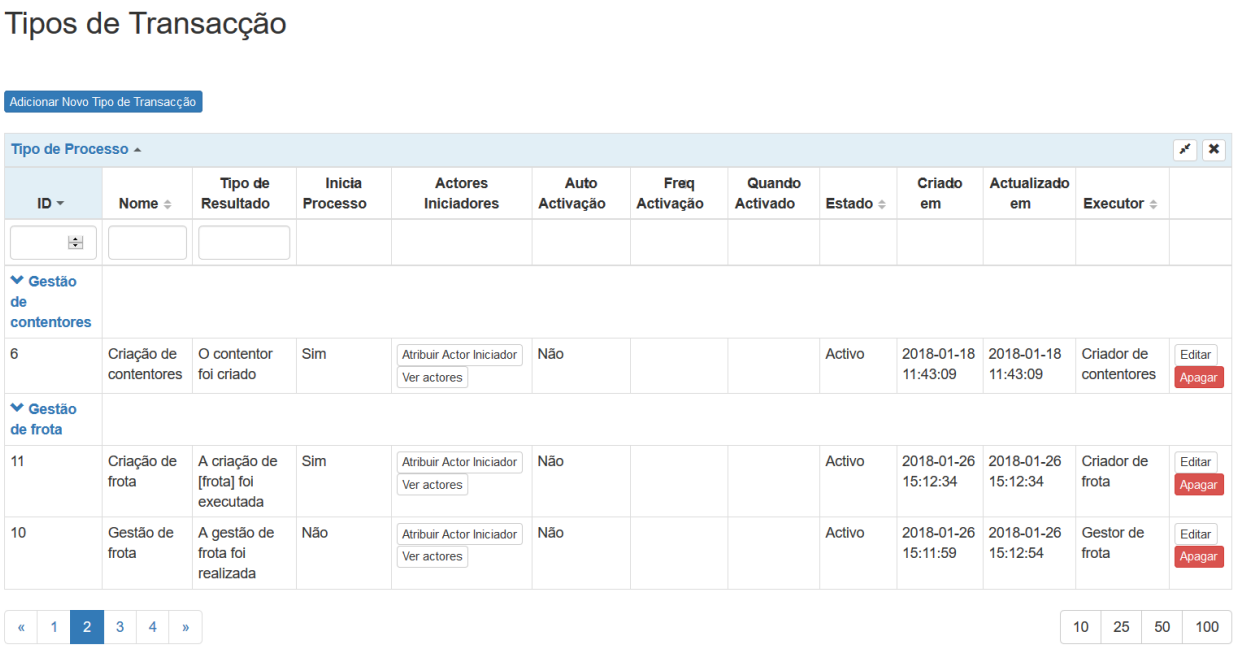

Figura 32 – Exemplo de listagem de Tipos de Transacção na versão final.

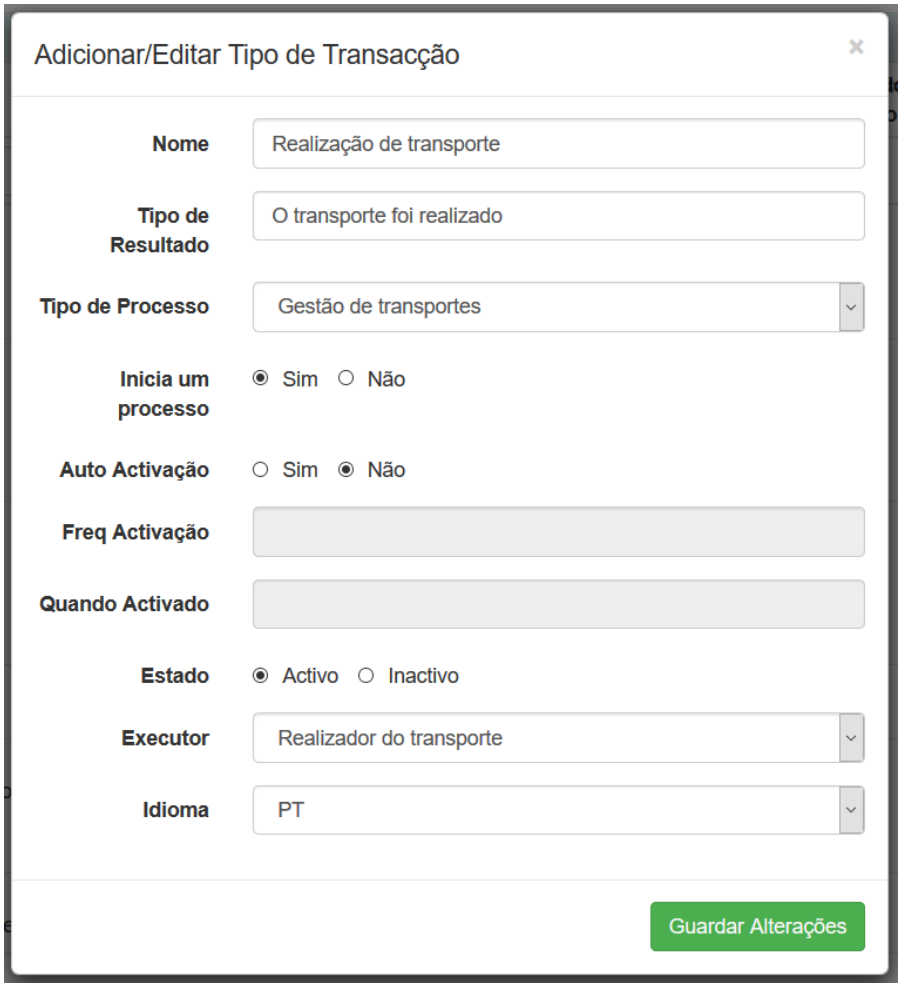

Figura 33 – Formulário para criação/edição de um Tipo de Transacção na versão final.

As diferenças entre as várias versões devem-se ao facto de alterações realizadas nas tabelas da base de dados e também pela adição de novas funcionalidades após o aparecimento de transacções e processos que exigiam a utilizam dessas opções.

Na versão 1 (ver Figura 28, Figura 29), apenas se considerava alguns tipos de parâmetros, facilmente foi concluído que quando um utilizador inicia uma transacção raiz (que é a primeira transacção que faz parte de um processo), deve ser o sistema a criar esse processo, em função da definição feita na transacção (se inicia ou não um processo), a versão 2 (ver Figura 30) reflecte estas alterações.

Nos diagramas DEMO, mais concretamente no diagrama PSD podem existir transacções que activam-se (iniciam) sem a intervenção do utilizador, são consideradas transacções de auto activação e que iniciam-se numa certa ocasião e com uma determinada frequência. A diferença entre a versão 2 (ver Figura 30) e a 3 (ver Figura 31) é o acrescento dessas possibilidades.

5.2.2.1. *T State*

No segundo componente são colocados os Actos de Transacção que existem na metodologia DEMO, como por exemplo, o pedido, promessa, execução, declaração e aceitação, estes são os cinco actos que existem sempre numa transacção. Este componente é extremamente importante pela relação existente entre os vários componentes do protótipo.

Nas imagens seguintes, é possível verificar como está implementado o componente *T State*.

Versão final

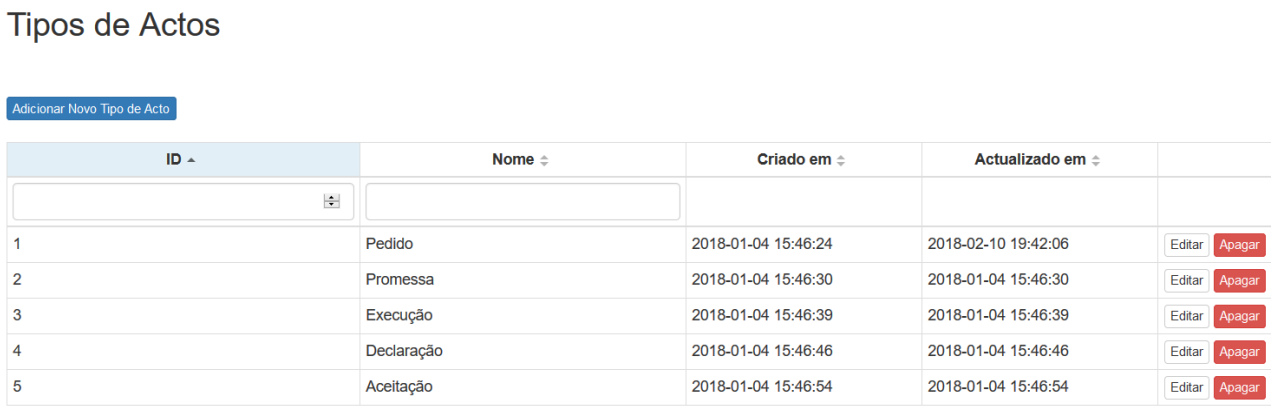

Figura 34 – Exemplo de listagem de Tipos de Actos na versão final.

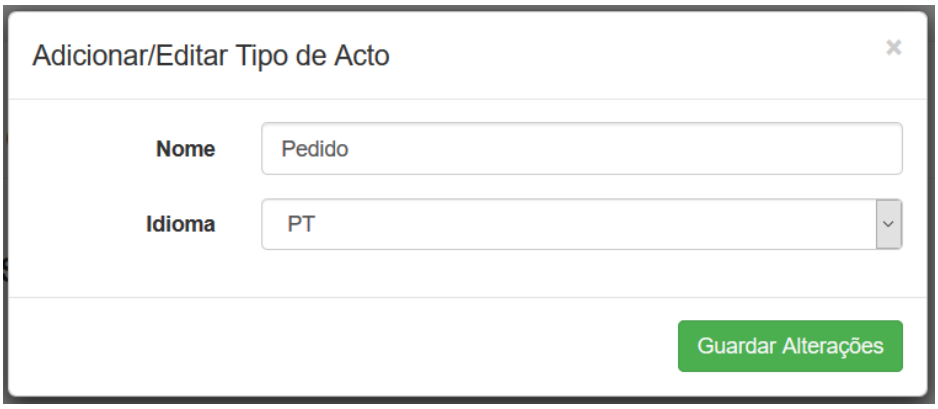

Figura 35 – Formulário para criação/edição de um Tipo de Acto na versão final.

5.2.3. Gestão de Entidades - Tipos de Entidade

Esta gestão pode ser assimilada como se fosse a definição da tabela de uma base de dados que é responsável por guardar os registos/dados correspondentes. Um Tipo de Entidade é a derivação da parte das *classes* que é alcançada com o diagrama OFD. De salientar, que o Tipo de Entidade não é nada mais do que a definição estrutural do que deve ser guardado na base de dados tal e qual a funcionalidade de uma tabela.

Versão 1

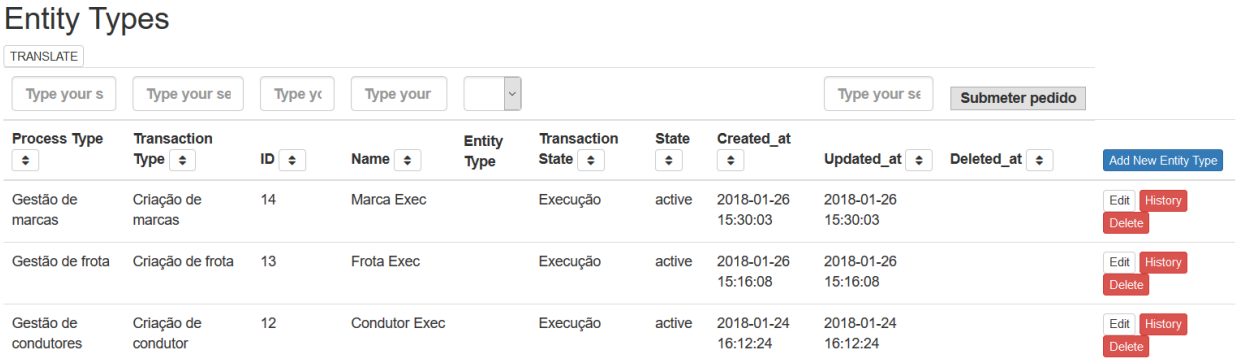

Figura 36 – Exemplo de listagem de Tipos de Entidades da empresa na versão 1.

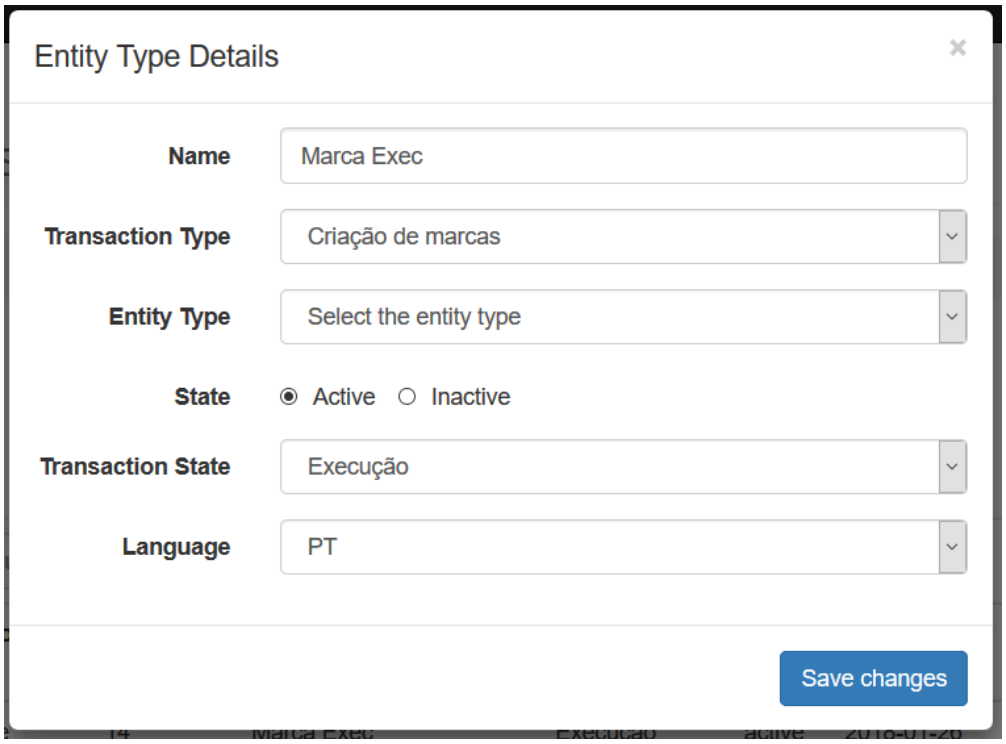

Figura 37 – Formulário para criação/edição de um Tipo de Entidade da empresa na versão 1.

Versão final

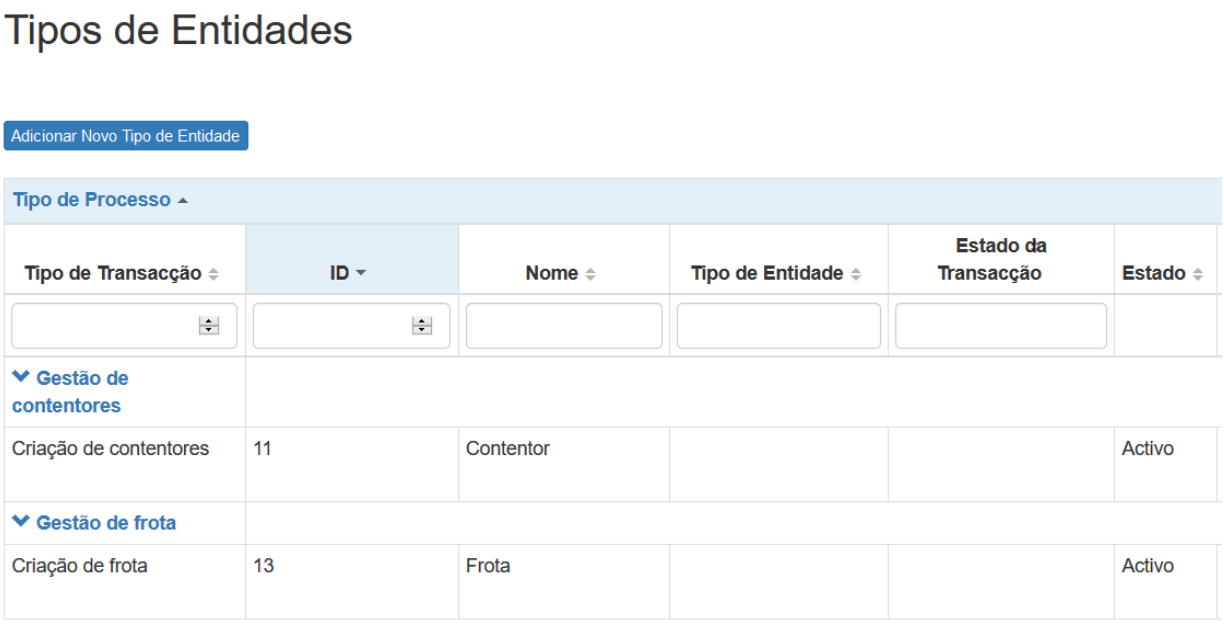

Figura 38 – Exemplo de listagem de Tipos de Entidades da empresa na versão final.

Figura 39 – Formulário para criação/edição de um Tipo de Entidade da empresa na versão final.

Na versão 1, mais propriamente na Figura 36 foi utilizado um esquema antigo para realização de ordenação e procura de Tipos de Entidade. Com a introdução do *plugin Ng-Table*, esse esquema foi removido e substituído pela tabela gerada pelo *Ng-Table* (ver Figura 38).

De realçar, que foram introduzidos novos campos no formulário de especificação (ver Figura 39), dado que anteriormente não existia uma especialização entre os Tipos de Entidade. Essa especialização consiste em utilizar a mesma transacção em vários Tipos de Entidade, em casos que um Tipo de Entidade "pai" tem associados Tipos de Entidade "filho" porque as entidades "filho" herdam todas as propriedades da entidade "pai", uma vez que são propriedades comuns aos "filhos".

5.2.4. Diagrama de Estrutura do Processo - Ligações Causais

As Ligações Causais são as regras de causa das transacções, em outras palavras, que Actos de Transacção originam o início de outras transacções dependentes destas de forma automática (sem intervenção humana). Com estas regras, o utilizador não necessita de manualmente iniciar as transacções, que são as seguintes no fluxo do processo. Estas regras são depois utilizadas pelo Painel de Controlo de forma automática.

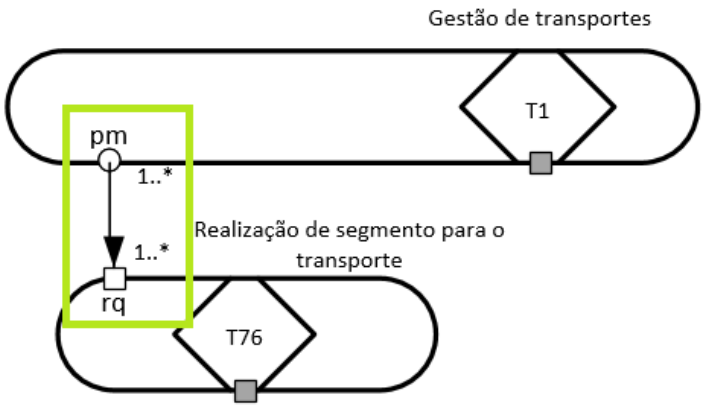

Figura 40 – Diagrama PSD com duas transacções e uma Ligação Causal.

Na Figura 40 são exibidas duas transacções, em que a transacção "Gestão de transportes" no acto "Promessa" (pm) origina a criação do pedido (rq) da transacção "Realização de segmento para o transporte", esta criação é uma Ligação Causal devido a conexão que está dentro da cor verde na figura.

Para especificar estas ligações no protótipo (ver Figura 41), é necessário preencher qual a transacção e o acto que origina o pedido de outra no campo *Causing Transaction* e *Transaction State* respectivamente, a transacção que vai ser criada no campo *Caused Transaction* e a multiplicidade nos campo *Min* e *Max* que representa o número de vezes que a transacção vai ser criada.

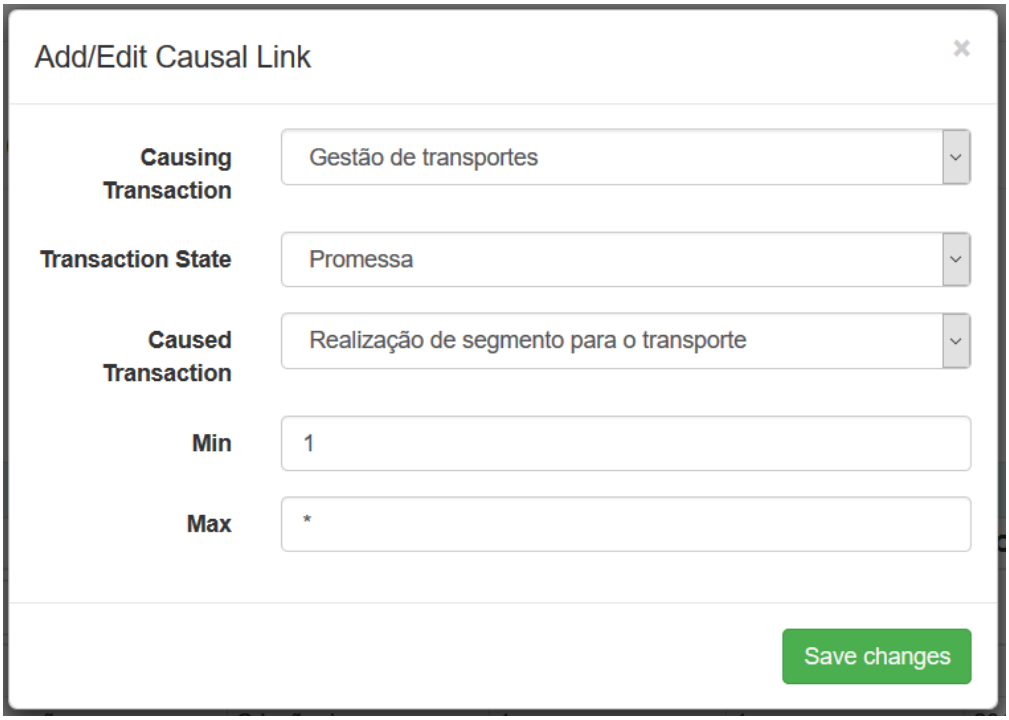

Figura 41 – Formulário para criação/edição de uma Ligação Causal.

5.2.5. Diagrama de Estrutura do Processo - Ligações de Espera

As Ligações de Espera são as transacções que só tem autorização para continuar para o acto seguinte, se e só se, a transacção e o acto escolhido já foram executados pelo utilizador.

Estas são ligações em que a transacção que "espera" por outra, obrigatoriamente precisa de informação proveniente dessa transacção. Tendo por base, o exemplo (ver Figura 42) da empresa interveniente, quando é iniciada uma transacção "Realização de transporte" o acto de execução, não pode ser realizado se não estiver realizado o acto de aceitação da transacção "Realização de segmento para transporte".

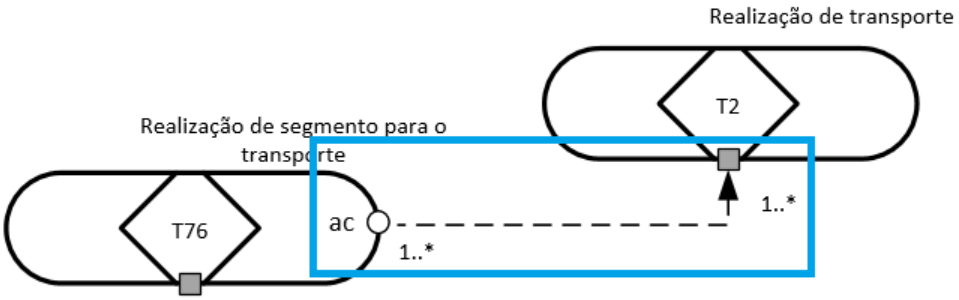

Figura 42 – Diagrama PSD com duas transacções e uma Ligação de Espera.

Com o exemplo da Figura 42, que está num formato de diagrama PSD dos modelos DEMO, é possível especificar a respectiva Ligação de Espera (colocada dentro do rectângulo azul).

Analisando o diagrama e tendo em conta as regras DEMO para este tipo de diagramas, no formulário da Figura 43, a transacção "Realização de segmento para o transporte" corresponde ao campo *Waited Transaction* que é a transacção esperada para prosseguir o fluxo de processo, no campo *Waited Fact* é o acto de "Aceitação" que é o acto esperado. Aos campos *Waiting Transaction* e *Waiting Fact* é a transacção "Realização de transporte" e o acto de "Execução" respectivamente, é a transacção que fica "à espera" de uma ou mais transacções.

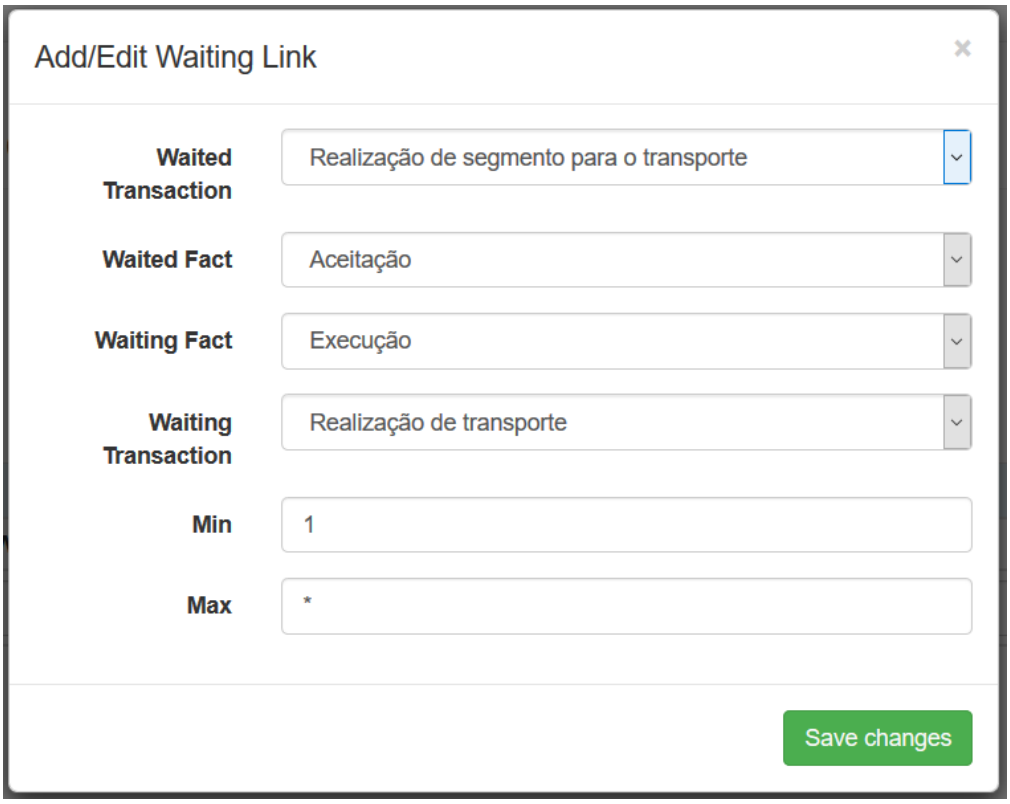

Figura 43 – Formulário para criação/edição de uma Ligação de Espera.

5.3. Segunda função (Execução)

5.3.1. Painel de Controlo

O Painel de Controlo é em conjunto com a Pesquisa Dinâmica a característica mais importante do protótipo. A função principal do Painel de Controlo é a constituição de todo o painel especificado na função de Modelação (primeira função).

Apenas devem aparecer as tarefas, que o utilizador tem permissão para dar continuidade e fazer a restrição das tarefas com processos já finalizados.

Sempre que exista a acção para continuar outro estado de uma tarefa, é necessário o sistema verificar se o mesmo é permitido, seguindo as Ligações de Espera definidas pelo diagrama PSD do DEMO.

Uma tarefa que tem uma Ligação de Causa, que gera uma outra tarefa deve ser feita de forma automática pelo sistema.

Versão 1

Na primeira versão do Painel de Controlo era preciso criar um novo processo ao clicar no botão 1 da Figura 44 e então só depois criar uma nova tarefa clicando no botão 2, isto quer dizer que sempre que era gerado um novo processo o utilizador tinha de completar dois passos para o conseguir fazer.

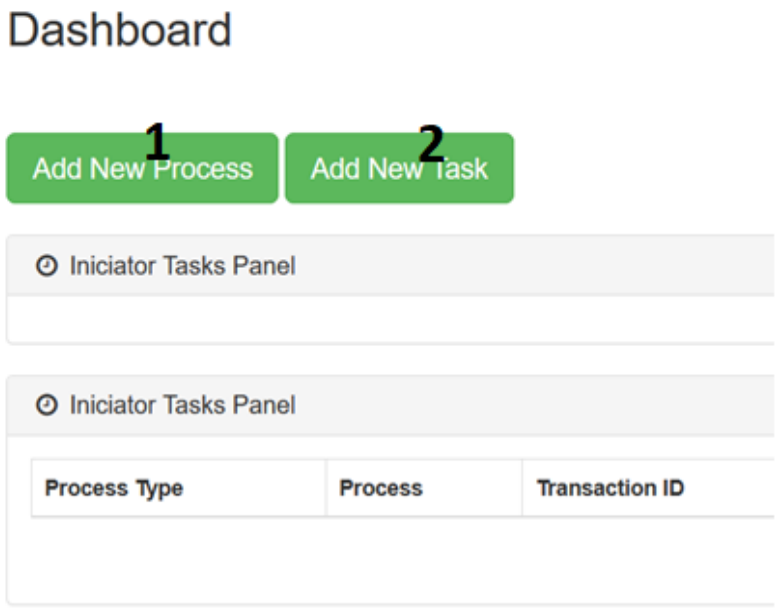

Figura 44 – Painel de Controlo na versão 1.

Figura 45 – Adicionar um novo processo na versão 1.

Na Figura 45, estão apresentados os campos necessários para a criação de uma nova instância de processo.

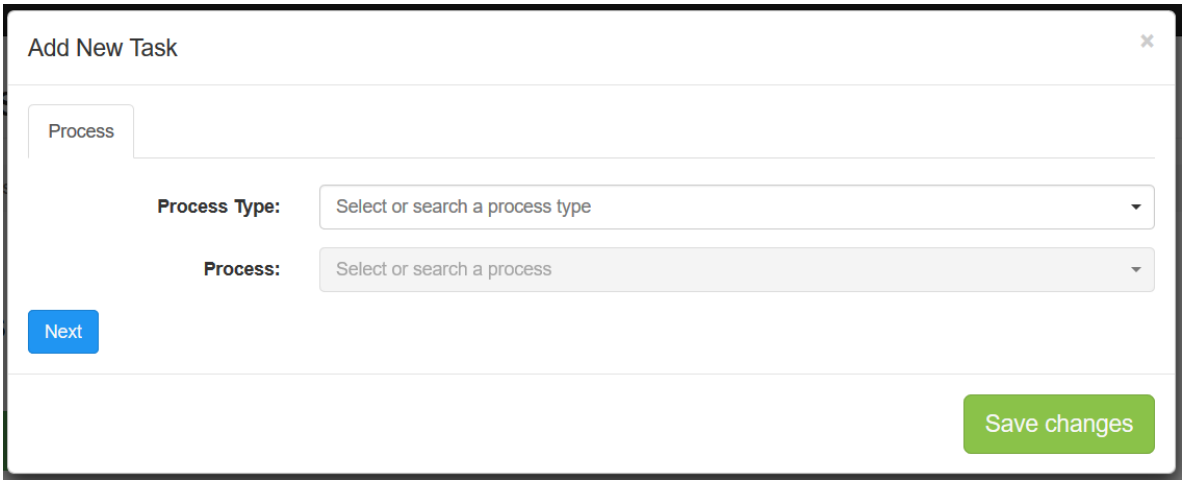

Figura 46 – Criação de uma nova tarefa (transacção) na versão 1.

Após a criação de uma instância de processo na Figura 45, o utilizador criava uma nova tarefa seleccionado o Tipo de Processo e a instância de processo correspondente (ver Figura 46).

Versão 2

Concluiu-se que ter de seleccionar um tipo de tarefa numa caixa de selecção para criar uma nova tarefa não era o mais indicado em termos de usabilidade, o utilizador tinha toda a vez que pretendia criar uma tarefa, de procurar o Tipo de Transacção numa caixa de selecção tornando-se num processo moroso e penoso.

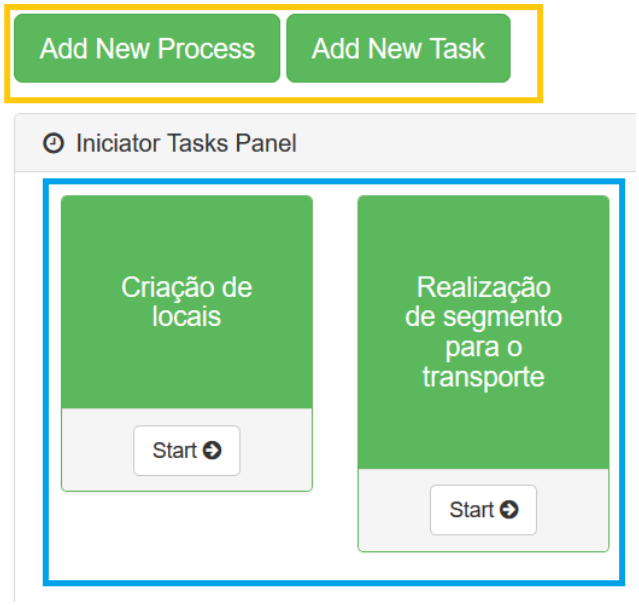

Figura 47 – Painel de Controlo com as funções novas na versão 2.

Na Figura 47, é possível verificar que agora as tarefas (Tipos de Transacção, de cor azul) estão colocados num painel "*Iniciator Tasks Panel*", tornando a experiência em termos de

usabilidade mais agradável e fácil para o utilizador, uma vez que se reduz o número de cliques a realizar. Enquanto que, na versão anterior estavam colocados dentro de um *modal* (rectângulo de cor amarela).

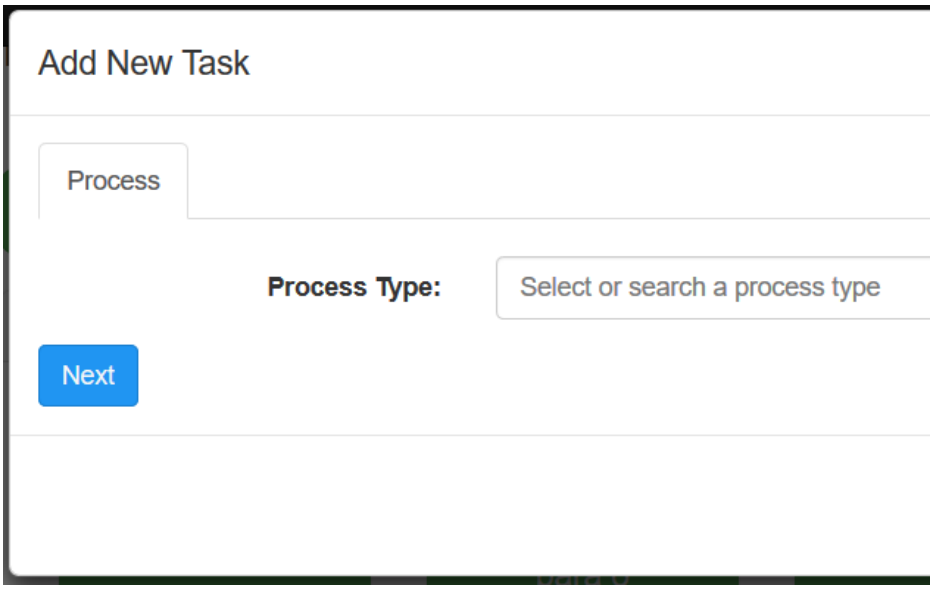

Figura 48 – *Modal* de uma tarefa na versão 2.

Quando uma tarefa tem de ser criada, é necessário escolher uma instância de processo a que essa tarefa pertence, como está exibido na Figura 48.

Versão 3

Na versão 3 e analisando a forma como os utilizadores podiam de forma involuntária ou voluntária criar novas instâncias de processo, sem o mesmo ser necessário porque já existiam instâncias de processo, em que as tarefas poderiam ser associadas.

Foi introduzida uma alteração na base de dados e no componente que permite definir a estrutura de um Tipo de Transacção, que quando é realizada uma tarefa, esta cria um novo processo automaticamente. A alteração tem fundamento, dado que num processo existe sempre uma tarefa que o inicia, porque é a primeira tarefa desse processo.

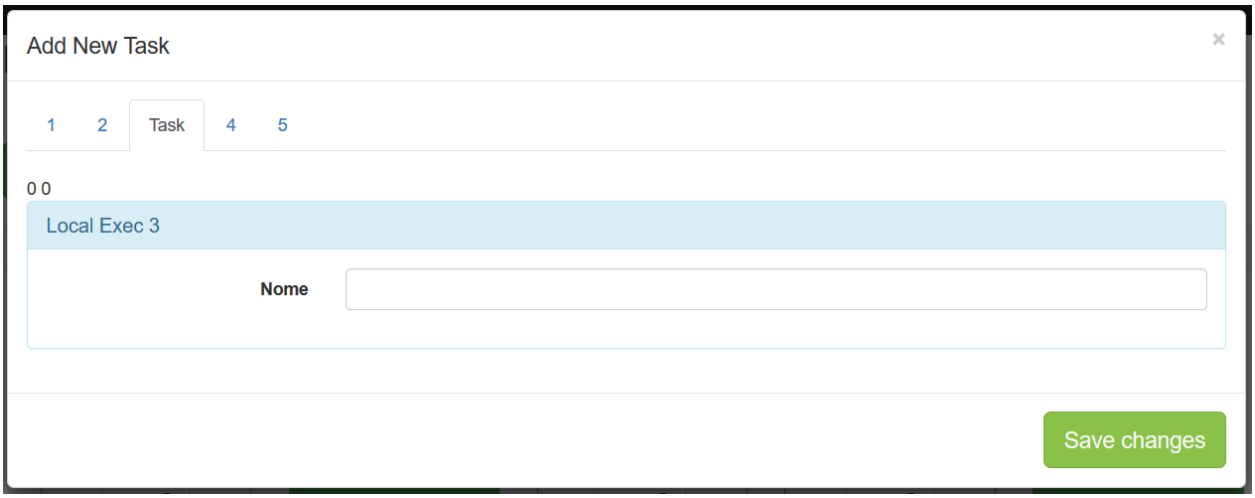

Figura 49 – *Modal* com formulário de uma tarefa na versão 3.

Na Figura 49, um exemplo de uma tarefa que inicia uma nova instância de processo, neste momento o utilizador já não precisa de seleccionar o processo a que a tarefa pertence.

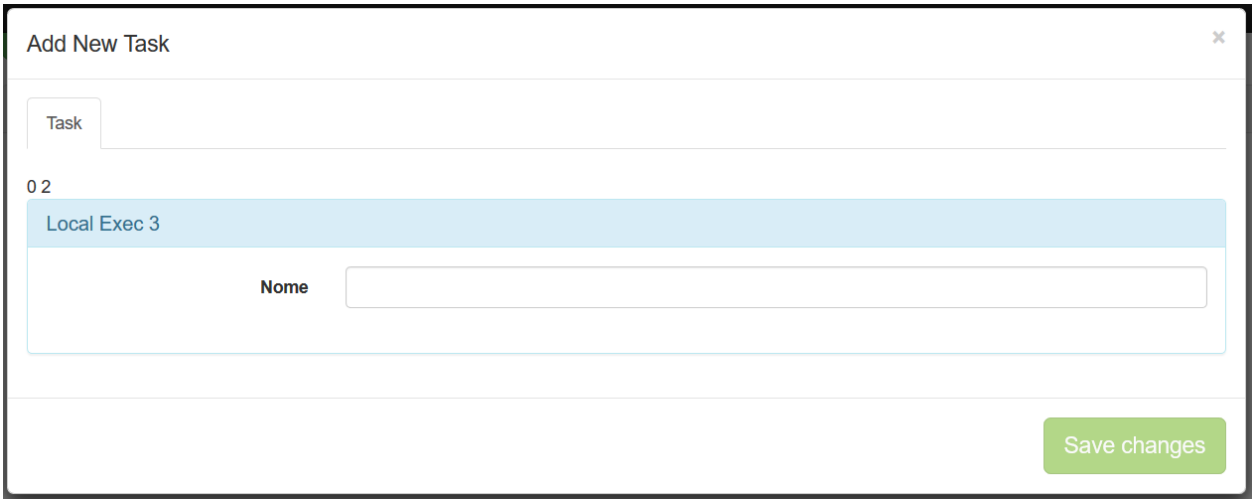

Figura 50 – *Modal* com formulário com as abas necessárias na versão 3.

Entretanto, a maneira de o *modal* carregar as informações relativas aos formulários da transacção mudou ligeiramente, apenas é carregado os Tipos de Actos que contém propriedades/formulários.

Versão 4

Nesta versão, para cada Tipo de Actos em que existisse propriedades, era criado novos Tipos de Entidade na página Gestão de Entidades e atribuído o Tipo de Acto.

Foi executada uma mudança na forma de estruturar os Tipos de Entidade e das propriedades, unicamente existe um Tipo de Entidade e as propriedades é que fazem parte do Tipo de Acto, evitando assim a redundância de entidades que na sua génese são iguais.

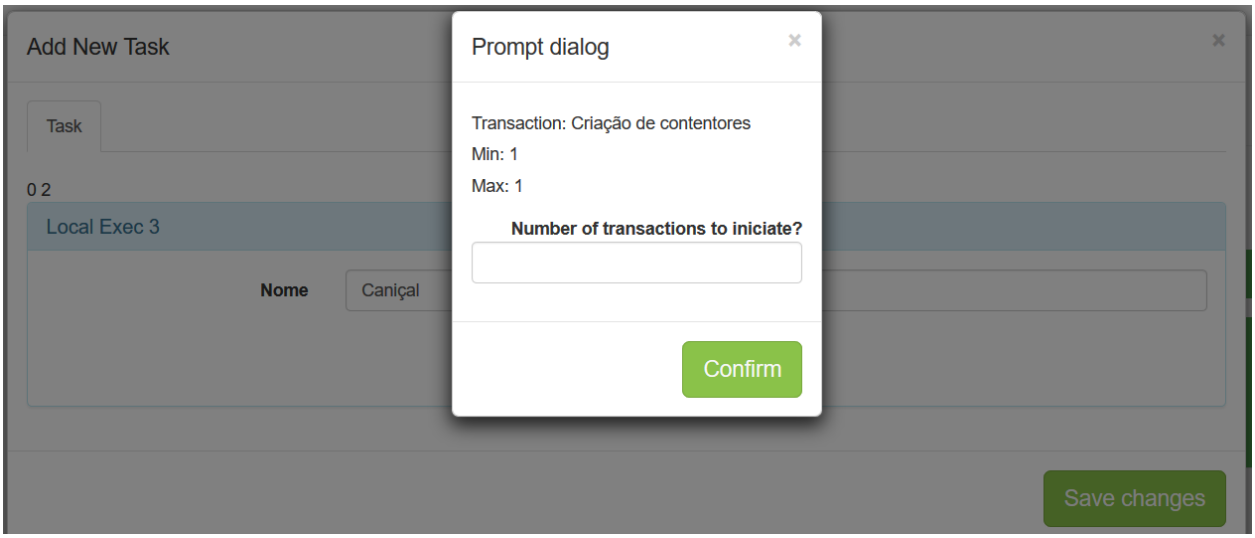

Figura 51 – Uma tarefa com uma Ligação de Causa.

Na Figura 51, está exposta uma tarefa que contém uma Ligação de Causa a outra tarefa, pelo que o utilizador tem de escolher quantas tarefas (transacções) pretende iniciar.

Versão 5

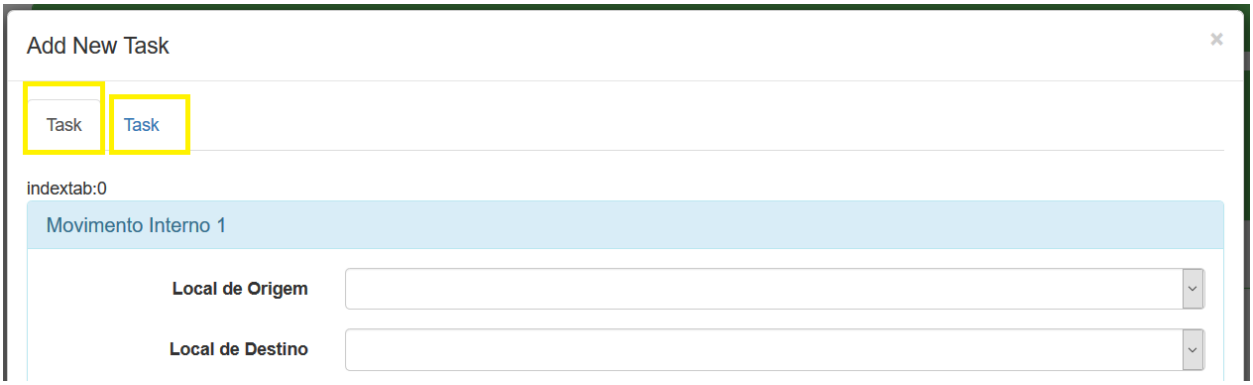

Figura 52 – Tarefa e abas correspondentes na versão 5.

Na Figura 52, se um utilizador é iniciador e executor de uma determinada tarefa então deve ser realizado um carregamento total de todos os Actos da transacção, fazendo o pedido das propriedades nos Tipos de Acto ao servidor um a um, em vez de clicar no botão seguinte.

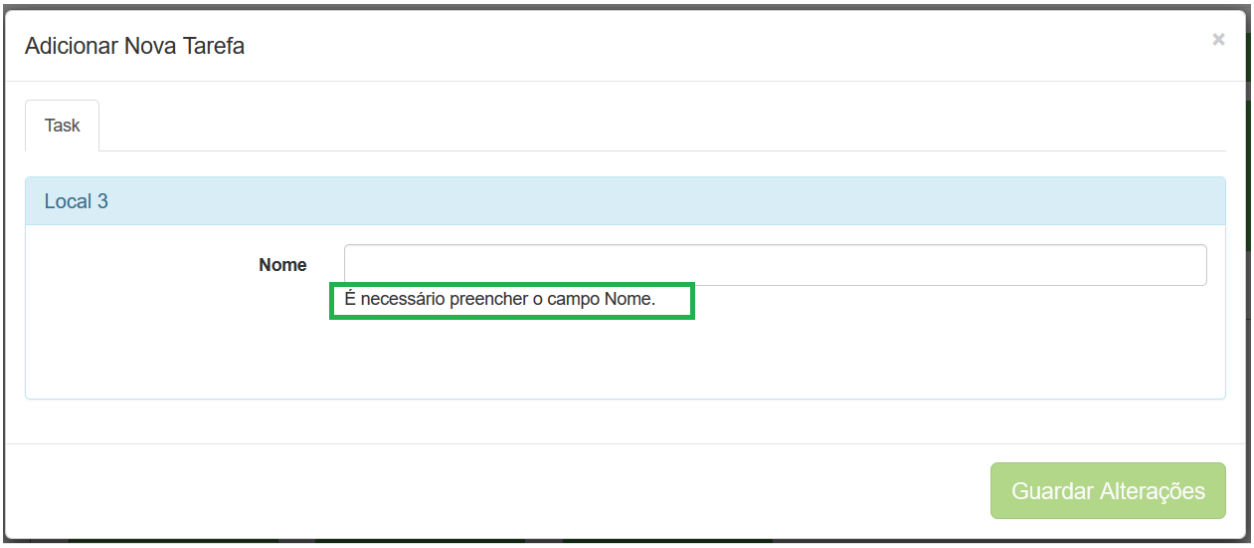

Figura 53 – Validação dos campos dinâmica.

Na Figura 53, está exibida como é realizada a validação dinâmica dos campos de um formulário.

Versão 6

Na versão 6, foi adicionada uma pequena funcionalidade, para o caso em que uma certa propriedade é uma regra de negócio definida pela especificação de processo, isto quer dizer que essa propriedade é dependente de outras num dado formulário.

A ideia geral é o armazenamento na base de dados de pequenos pedaços de código em *JavaScript* ou PHP em determinadas propriedades e que essas regras sejam executadas quando essas propriedades estiverem apresentadas no formulário a que correspondem.

```
element.bind('change',
    function()
    {
        var arr_tab = scope.$parent.modal_formTab.tab;
        var id_tab = arr_tab.findIndex(x=>x.type === 3);
        var arr_tab_origin = scope.$parent.modal_formTab.tab[id_tab].propsform;
        var id_prop_origin = arr_tab_origin.findIndex(x=>x.id === 29);
        var id_prop_source = arr_tab_origin.findIndex(x=>x.id === 30);
        var arrayKeys = Object.keys(scope.$parent.modal_formTab.tab[id_tab].propsform[id_prop_origin].fields);

        if (scope.$parent.modal_formTab.tab[id_tab].propsform[id_prop_origin].fields[arrayKeys[arrayKeys.length-1]]==14)
        {
            angular.element(document.getElementById('30'))[0].disabled = true; scope.$parent.modal_formTab.tab[id_tab].propsform[id_prop_source].mandatory=0;
        }
        else
        {
            angular.element(document.getElementById('30'))[0].disabled = false; scope.$parent.modal_formTab.tab[id_tab].propsform[id_prop_source].mandatory=1;
        }

        scope.$parent.$apply();
    }
);
```

Figura 54 – Exemplo de pequeno trecho de código para um campo.

Na Figura 54 está apresentado um exemplo de código em que um campo depende da escolha das opções de um anterior para ser desbloqueado.

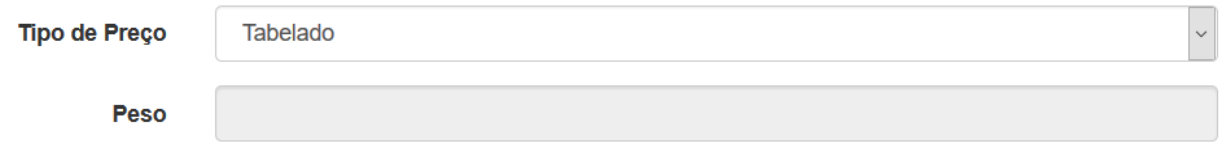

Figura 55 – Campo Tipo de Preço que desabilitou o campo Peso.

Tendo por base a Figura 55, o exemplo em concreto da regra é que se no campo Tipo de Preço for seleccionada a opção "Tabelado", o campo Peso deve ser bloqueado e não deve obrigatório o seu preenchimento por parte do utilizador.

Se eventualmente for escolhido a opção "A peso", o campo Peso deve estar desbloqueado e o seu preenchimento é obrigatório pelo que é necessário efectuar a validação do campo, como apresentado na Figura 56.

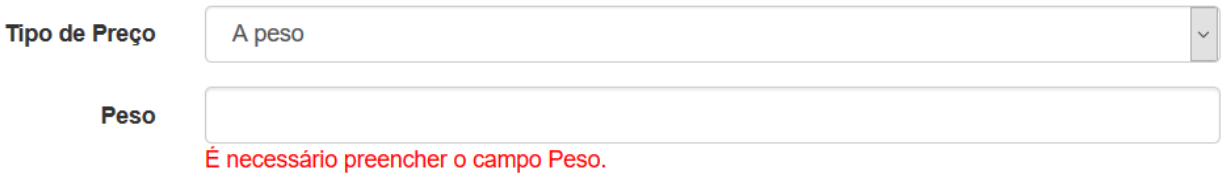

Figura 56 – Campo Tipo de Preço que habilitou o campo Peso.

Versão final

Os utilizadores quando necessitam de criar uma nova tarefa, a mesma cumpre apenas um pedido ao servidor com todas as propriedades de todos os Tipos de Actos, nas versões anteriores era executado um pedido da cada vez para cada Tipo de Acto, o que gerava alguma lentidão no carregamento das propriedades no *modal*.

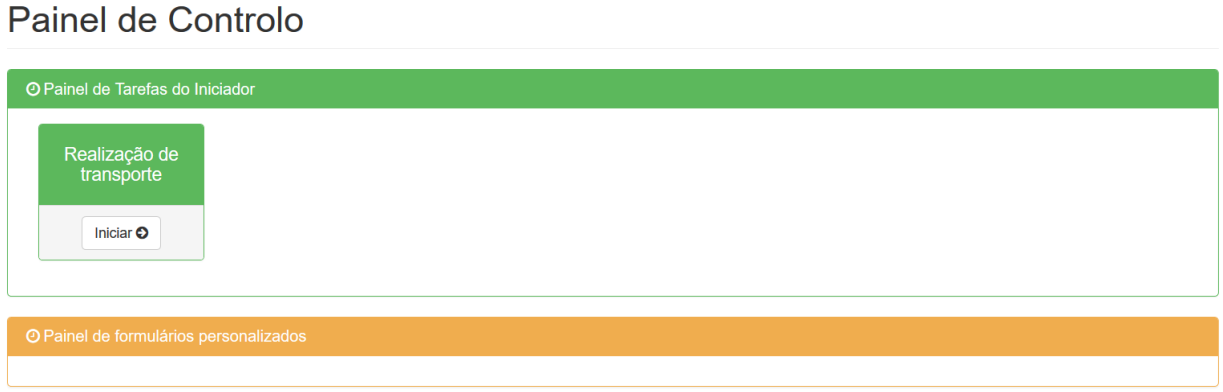

Figura 57 – Painel de Tarefas do Iniciador que apresenta as transacções que utilizador pode iniciar.

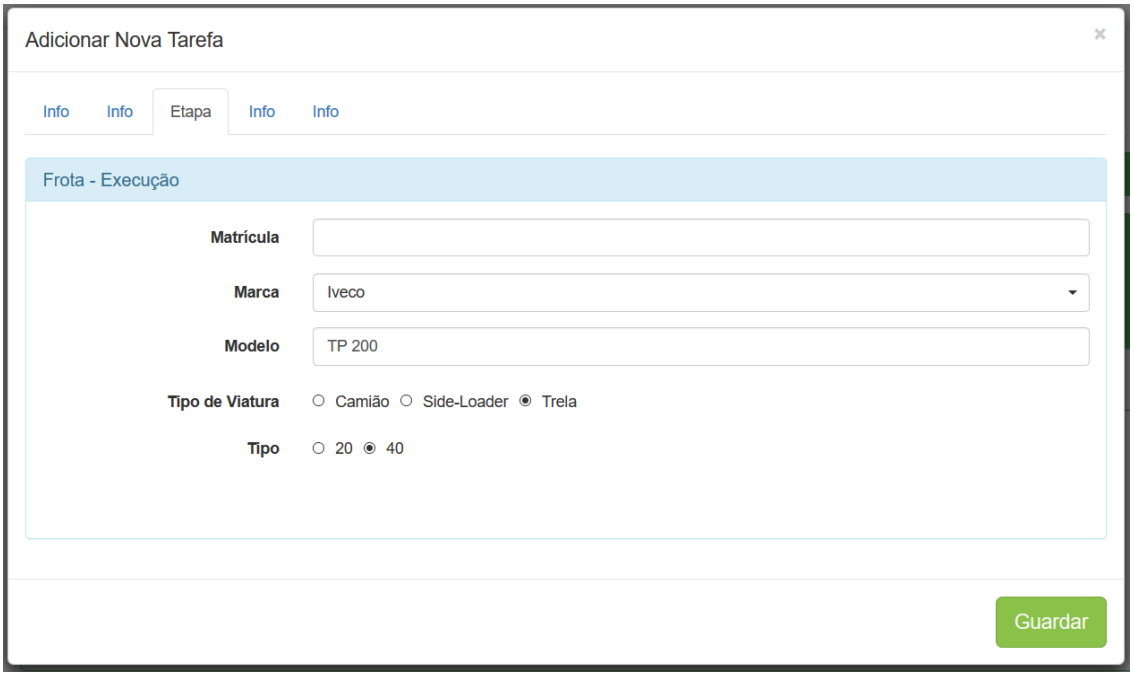

Figura 58 – Exemplo de criação de uma instância de transacção do tipo Frota.

Na Figura 58, estão inseridos alguns melhoramentos em termos de usabilidade principalmente na parte das caixas de selecção em que se usa o *plugin UI Select* para facilitar a procura dos elementos pretendidos.

Figura 59 – Verificação no Painel de Controlo de uma Ligação de Espera.

Na Figura 59, está representado uma tarefa/transacção que tem uma Ligação de Espera e que neste caso em concreto, não pode prosseguir para o estado seguinte porque existem outras transacções que necessitam de ser terminadas primeiro.

Figura 60 – Sistema de vistos no Painel de Controlo.

Na Figura 60, está exibido o sistema de vistos, em que os "vistos" são a notificação de que um utilizador já viu uma dada tarefa/transacção. Para que um utilizador possa usufruir desta funcionalidade, deve clicar no botão "Visto" (representado pelo número 1 na figura), após essa operação é apresentada a data, hora e o utilizador (representado pelo número 2 na figura) que realizou essa acção.

5.4. Geral

5.4.1. *Login*

Todo o protótipo só pode ser acedido por utilizadores que estejam registados na base de dados, uma vez que os dados que a aplicação de Fluxo de Trabalho tem, são de origem privada e não podem estar disponíveis de maneira pública. Por essa razão e pela recolha e análise dos requisitos em capítulos anteriores, a autenticação é um componente importante no protótipo.

Em virtude de a *framework Laravel* já dispor de componentes pré definidos a quando da sua instalação, tornou a tarefa de desenvolvimento mais fácil de realizar.

No seu núcleo, a autenticação do *Laravel* é constituída por "guardas" e "provedores". As "guardas" definem como os utilizadores são autenticados para cada pedido realizado. Os "provedores" definem como os utilizadores são obtidos do armazenamento persistente ou não volátil [40].

Os métodos do *Laravel* obtêm automaticamente os utilizadores, utilizando o *Eloquent* (que é o seu ORM e o construtor de base de dados). Todos estes métodos pré definidos são personalizáveis, consoante o que a aplicação requerer.

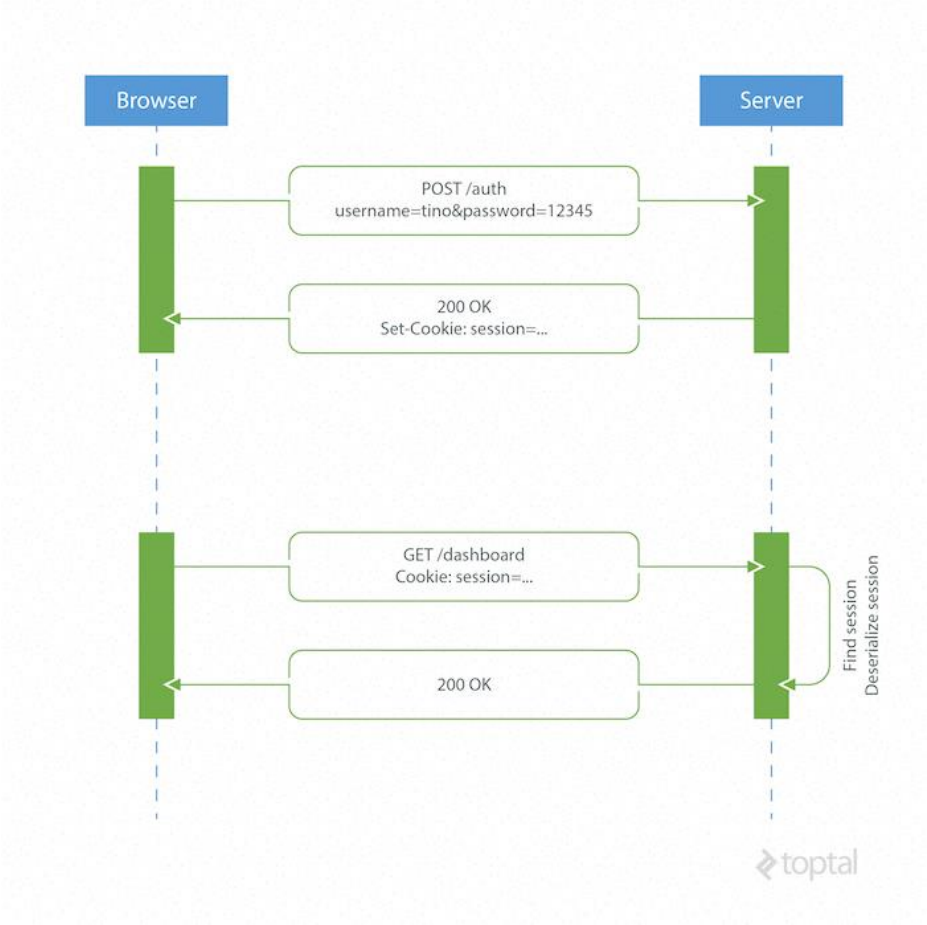

Figura 61 – Autenticação baseada no servidor [41].

A Figura 61 demonstra o funcionamento de uma autenticação normal baseada no servidor, que é exactamente o que a autenticação pré definida do *Laravel* realiza.

Como é possível observar na Figura 61, o navegador de *internet* executa um pedido com os dados de autenticação de um determinado utilizador e o servidor compara os dados recebidos do navegador com os dados da base de dados, se estes dados estiverem correctos o servidor retorna uma resposta com o estado *OK*, ao mesmo tempo é criada uma variável de sessão, que identifica que o utilizador realizou a autenticação com sucesso. No caso da autenticação do *Laravel*, além da variável de sessão é ainda gerado um *cookie*, para que se possa manter o estado da autenticação de um certo utilizador.

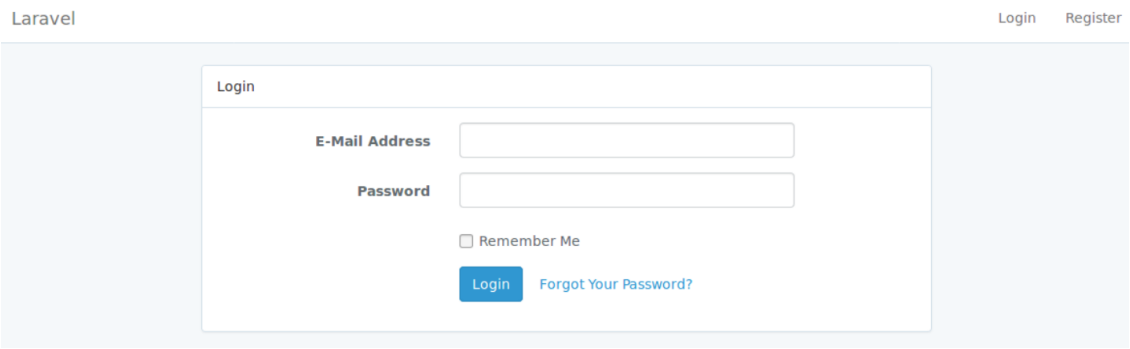

Figura 62 – Interface por defeito da autenticação de utilizadores.

Estes componentes pré definidos já dispõem de interface de utilizador por defeito, como apresentado na Figura 62, apesar disso foi alterada a interface de utilizador (ver Figura 63), para que a mesma possa estar de acordo em termos estéticos com o restante da interface desenvolvida.

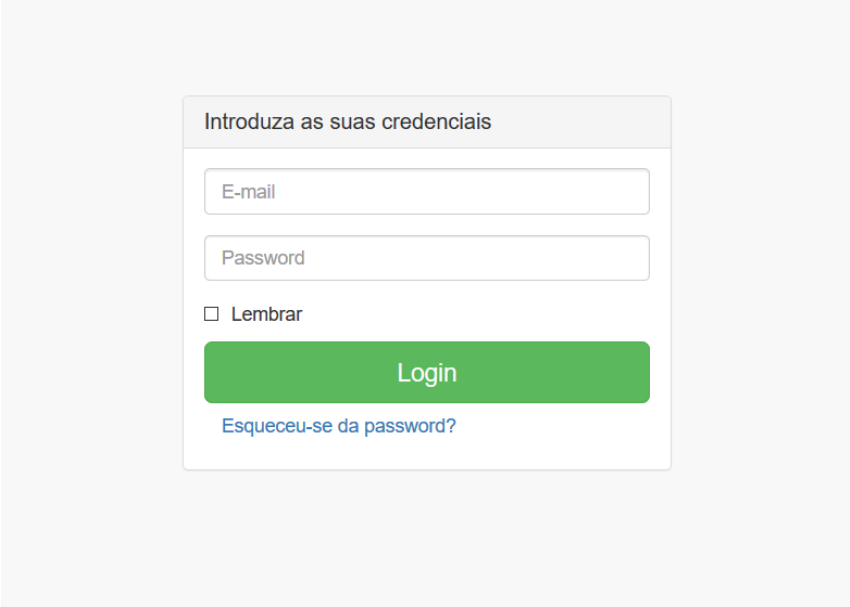

Figura 63 – Interface da autenticação de utilizadores já personalizada de acordo com o modelo utilizado.

Os dados de autenticação dos utilizadores estão colocados numa tabela (*users*) da base de dados definida para o efeito.

As rotas do protótipo, que são os caminhos empregues para executar as funções dos controladores do *Laravel*, encontram-se protegidas com o mediador de autenticação de nome *auth* (ver Figura 64) facultado pelo *Laravel*, isto quer dizer, que para a execução destas funções é indispensável a autenticação do utilizador [42].

```php
Route::middleware('auth')->group(function () {

Route::get( uri: '/', action: 'DashboardController@index');

Route::get( uri: '/home', action: 'HomeController@index');
```

Figura 64 – Exemplo de duas rotas protegidas com o mediador *auth*.

5.4.2. Comunicação entre clientes - *Pusher*

É no Painel de Controlo, que ocorre a única comunicação (ver Figura 65) entre os utilizadores do protótipo, por causa da importância que o Painel de Controlo possui, é neste "lugar" que o fluxo das transacções pode ser alterado e modificado consoante as acções que vão sucedendo no dia-a-dia dos negócios da empresa. Sempre que seja criado ou actualizado um estado de uma transacção, deve ser verificado quais os utilizadores aos quais importa receber uma notificação sobre essas transacções e ao mesmo tempo executar uma actualização parcial da página, ou seja, na tabela que contém as tarefas que correspondem aquele utilizador.

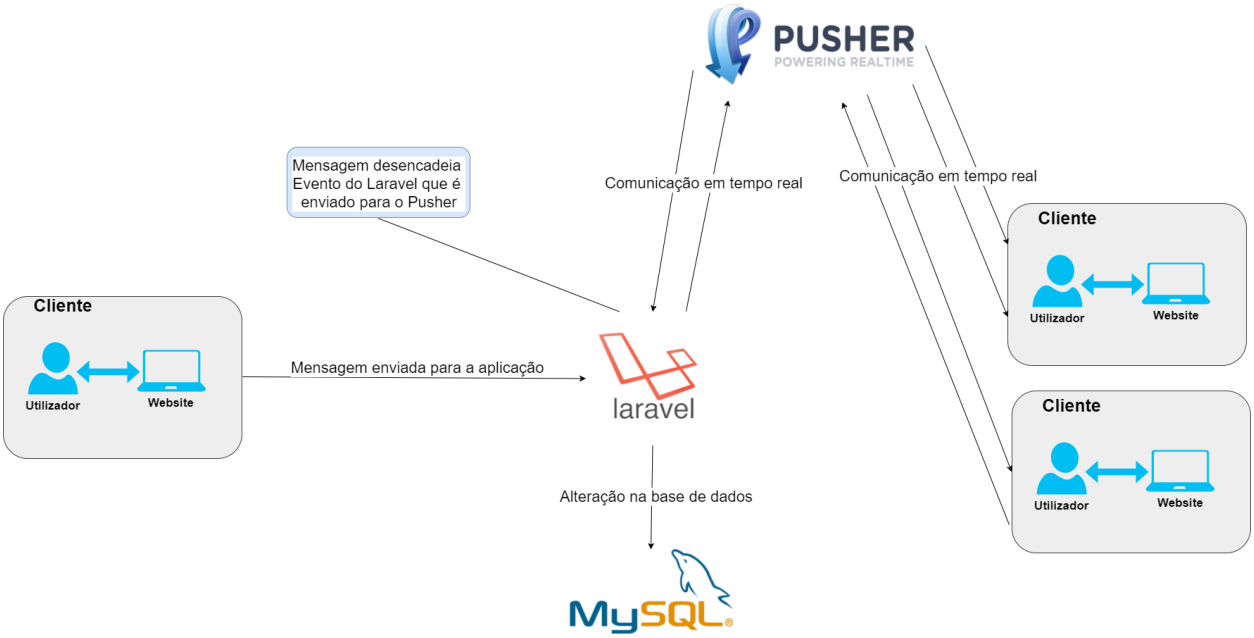

Figura 65 – Representação gráfica do funcionamento da comunicação entre a aplicação e os clientes.

Nas imagens seguintes, é possível assistir ao comportamento de uma comunicação entre utilizadores e a sua importância.

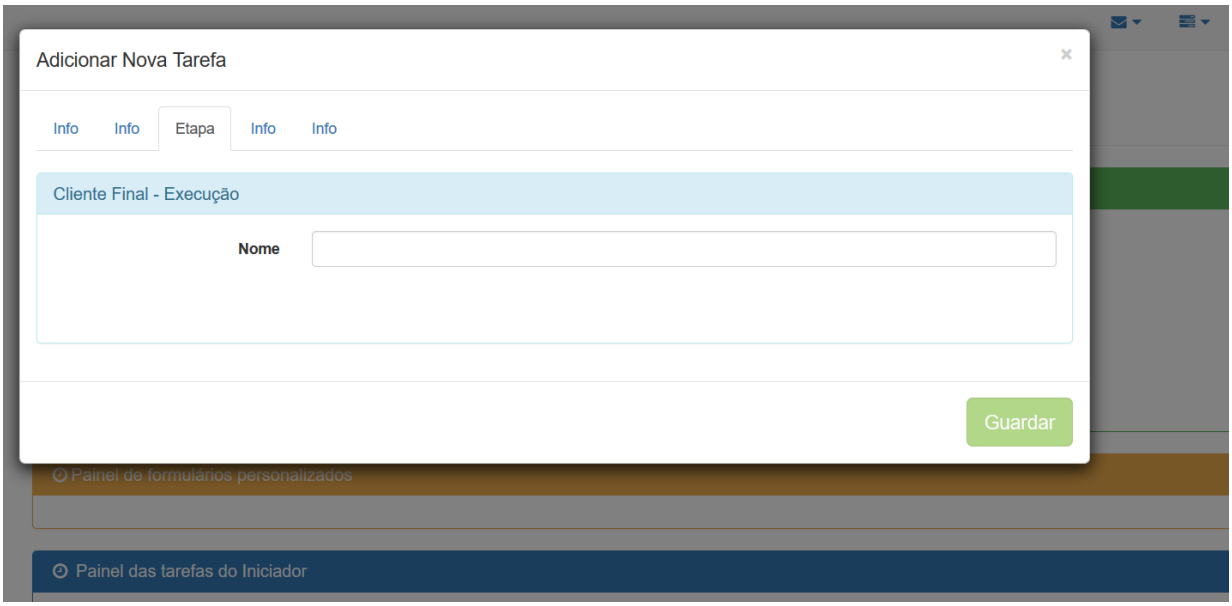

Figura 66 – *Modal* para criação de uma nova tarefa.

Na Figura 66 - O utilizador 1 está a criar uma nova tarefa e preenche os dados do formulário dessa transacção, quando carregar no botão "Guardar", deve ser feita a comunicação entre o utilizador 1 e todos os utilizadores, que devem segundo a estrutura definida nas permissões da base de dados, receber a notificação e respectiva actualização da tabela.

Neste caso em particular, consideremos que existe um utilizador 2, que é o receptor desta notificação.

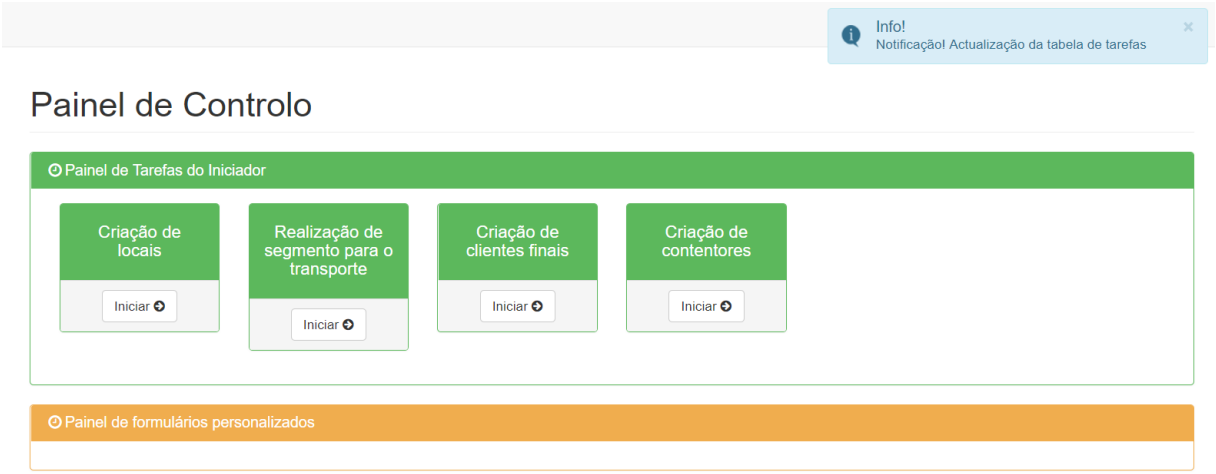

Figura 67 – Apresentação no canto superior direito da notificação.

Na Figura 67, o utilizador 2 recebe a notificação no seu Painel de Controlo e é informado dessa situação com uma mensagem representada a azul ciano no canto superior direito, com o seguinte formato e conteúdo "Info! Notificação! Actualização da tabela de tarefas". Simultaneamente a tabela de tarefas é actualizada, sem o recurso do utilizador ter de actualizar a página pelas opções do navegador de *internet*.

5.5. Restantes componentes

Os seguintes componentes foram implementados por os restantes mestrandos, pelo que a sua explicação neste capítulo é breve e pouco detalhada.

5.5.1. Gestão de Transacções - *Actor Initiates T*

As regras e condições do DEMO estabelecem que cada transacção existente deve ter pelo menos um actor iniciador, que é como o nome sugere, o actor responsável por iniciar a transacção em concreto. Por exemplo, uma dada transacção pode ter vários actores iniciadores. Esta especificação é vital para o funcionamento do Painel de Controlo, quando o mesmo apresenta as transacções que um dado utilizador pode iniciar.

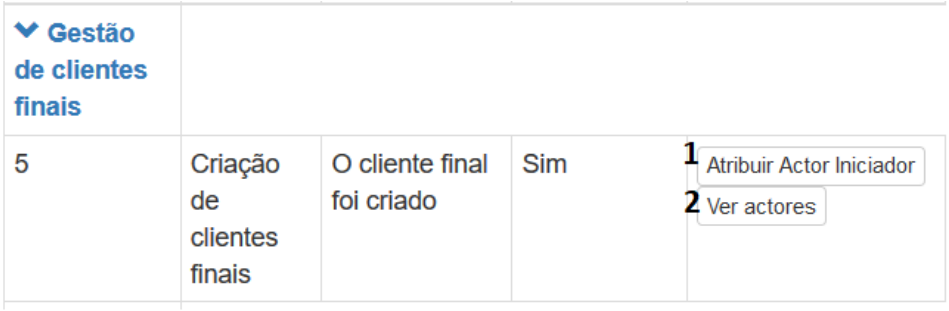

Figura 68 – Uma transacção e respectivos botões para adicionar os actores iniciadores.

Se o utilizador escolher o botão número 1 ("Atribuir Actor Iniciador") na Figura 68, o *modal* a apresentar é o apresentado na Figura 69.

No caso do botão número 2 ("Ver actores") na Figura 68, é apresentado a lista de todos os actores actualmente atribuídos.

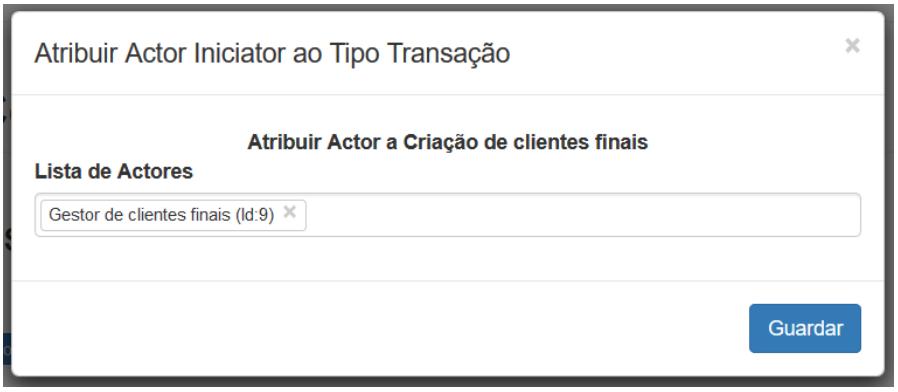

Figura 69 – Formulário de atribuição de actores iniciadores ao tipo de transacção.

No formulário da Figura 69, são adicionados os actores que iniciam essa transacção com o recurso do módulo *UI Select*. De mencionar que na Figura 69 apenas existe um actor iniciador associado a essa transacção.

5.5.2. Gestão de Propriedades – Entidade

Na Gestão de Propriedades são inseridas novas propriedades para as respectivas entidades. Cada propriedade tem o seu nome, o tipo de valor, tipo de campo, tipo de acto em que vai apresentar-se no formulário entre outros campos apresentados nas imagens seguintes (ver Figura 70 e Figura 71).

Figura 70 – Listagem de duas propriedades com base no exemplo da empresa.

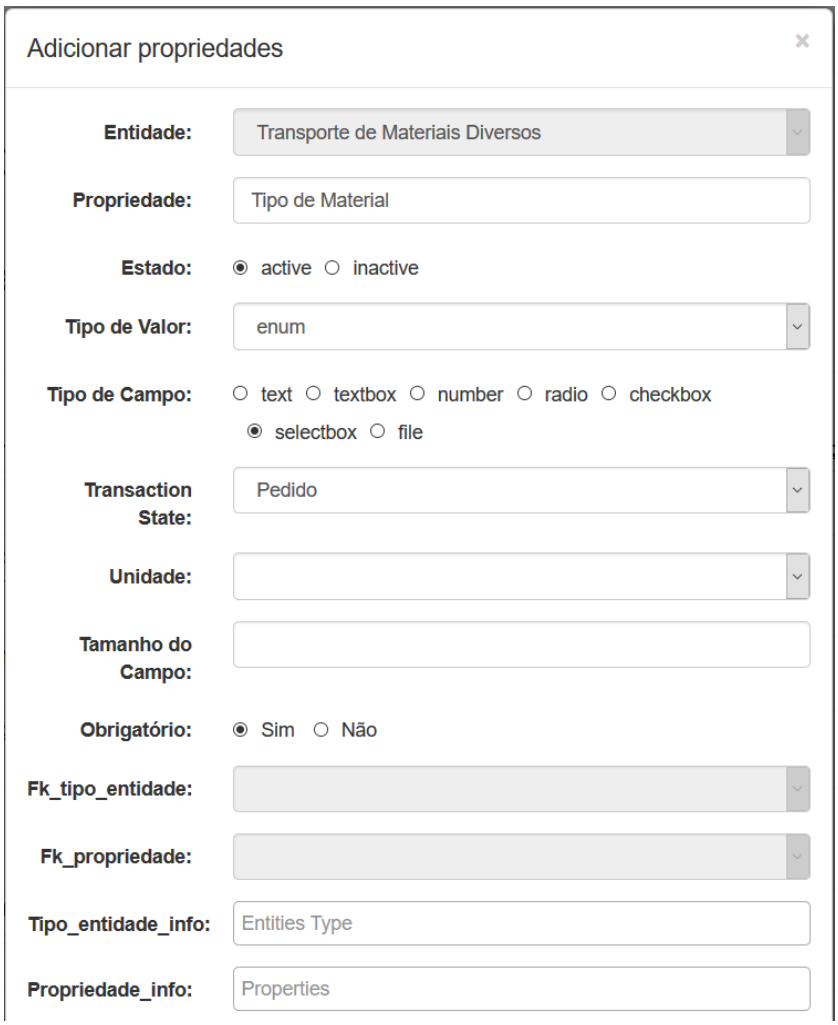

Figura 71 – Formulário para edição de uma propriedade pertencente a entidade Transporte de Materiais Diversos.

5.5.2.1. Reordenação dinâmica

Na Reordenação Dinâmica é utilizado o módulo *UI Sortable*, que possibilita a ordenação das propriedades de um Tipo de Entidade ou Relação usando a funcionalidade de arrastar e soltar.

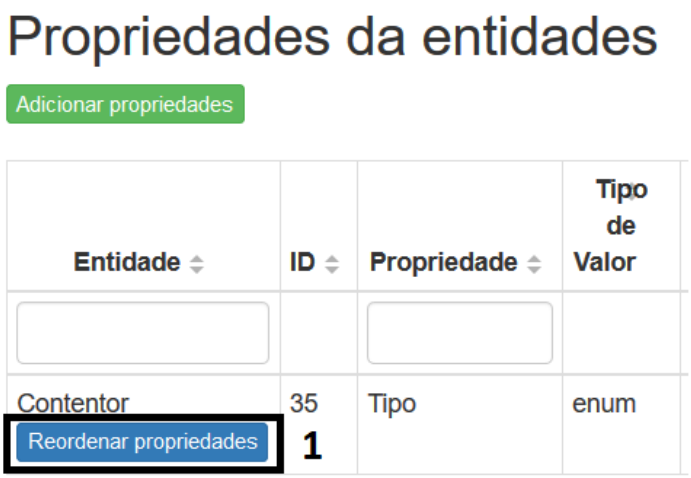

Figura 72 – Tipo de Entidade com o respectivo botão que encaminha para o *modal* reordenar propriedades.

Na Figura 72, o utilizador para realizar a reordenação de um Tipo de Entidade deve clicar no botão número 1 que abre o *modal* presente na Figura 73.

Figura 73 – *Modal* com as propriedades para o Tipo de Entidade Contentor.

Na Figura 73, o utilizador arrasta as propriedades com o rato pressionado e solta-as na ordem pretendida. As configurações são guardadas logo após essa acção.

5.5.3. Gestão de Relações – Tipos de Relação

Um Tipo de Relação é utilizado quando existe uma situação de muitos para muitos, como representado na Figura 74, um transporte pode ter vários segmentos relacionados, como também um mesmo segmento pode estar associado a vários transportes. A função de um Tipo de Relação é a mesma da multiplicidade numa base de dados, na multiplicidade de muitos-muitos.

Figura 74 – Lista com um Tipo de Relação entre os Tipos de Entidade Segmento e Transporte.

Observando a Figura 75, na criação do Tipo de Relação é necessário que os dois Tipos de Entidade já estejam inseridos nas tabelas da base de dados usando o componente para o efeito (Gestão de Entidades - Tipos de Entidade).

A um Tipo de Relação tem de existir sempre uma transacção, que efectivamente invoca o formulário correspondente.

Figura 75 – *Modal* com os campos para a edição do Tipo de Relação.

5.5.4. Pesquisa Dinâmica

A Pesquisa Dinâmica tem por base a construção das pesquisas usando as especificações efectuadas nos componentes Gestão de Entidades e Propriedades. À medida que a gestão de entidades ou propriedades sofre alterações a pesquisa dinâmica é actualizada com as novas alterações.

A construção dessa pesquisa é escolhida pelo utilizador, em que este selecciona as opções mais importantes com uma interface gráfica e sem o recurso a código de programação.

Na Figura 76, são escolhidas todas as propriedades com um valor qualquer para serem exibidas na listagem.

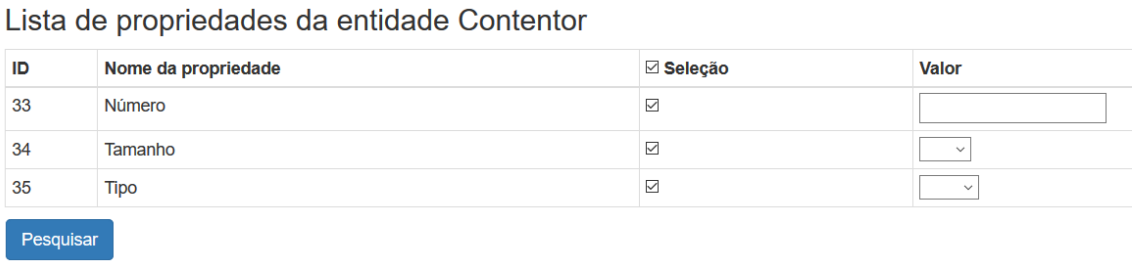

Figura 76 – Pesquisar os contentores e apresentar os valores de todas as propriedades.

Com a pesquisa realizada na Figura 76 é mostrada a listagem dos dados (Figura 77) das instâncias dessa entidade.

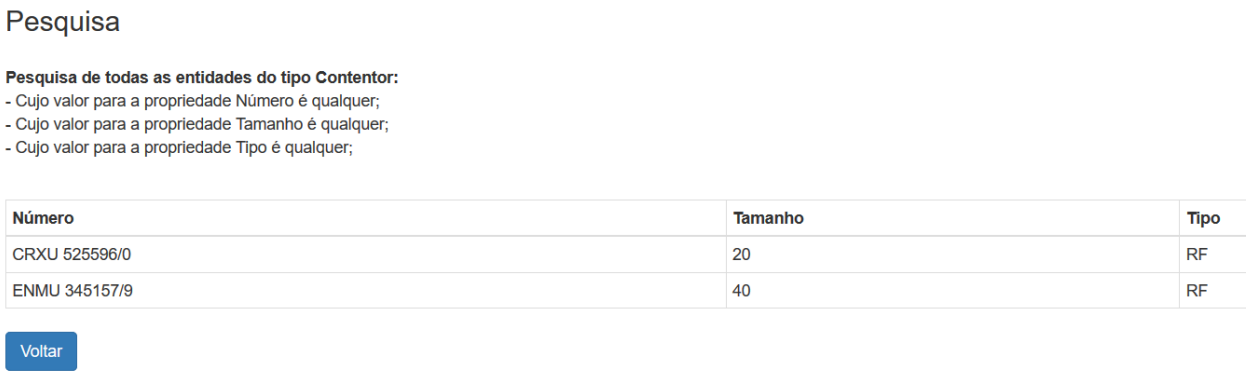

Figura 77 – A listagem da pesquisa de contentores escolhida anteriormente.

Na Figura 77, é apresentada uma lista dos contentores existentes no protótipo actualmente com os respectivos campos escolhidos na pesquisa dinâmica.

5.5.5. Pesquisa Guardada

Através da Pesquisa Dinâmica realizada na subsecção anterior, é possível guardar as pesquisas para futuramente utilizá-las no restante protótipo.

Para guardar a pesquisa é necessário definir um nome para essa pesquisa como observado na Figura 78. São guardadas as condições que fazem parte desta pesquisa.

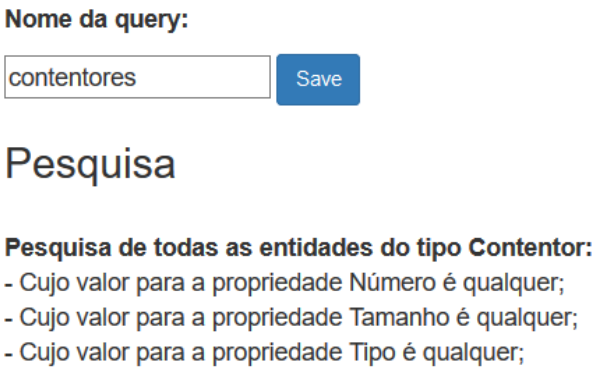

Figura 78 – Pesquisa Dinâmica a ser guardada.

As pesquisas que foram guardadas estão presentes numa página de nome Pesquisas Guardadas (ver Figura 79) em que é possível ter acesso a uma listagem das mesmas, respectivas condições e também executar as pesquisas.

Figura 79 – Uma pesquisa que está guardada e respectivas condições.

5.5.6. Formulários Customizados

Este componente é utilizado para agrupar formulários criados de várias transacções, para serem apresentados e preenchidos pelo utilizador ao mesmo tempo. Componente útil no caso de mediante uma determinada regra de negócio do DEMO ter de ser realizadas duas ou mais transacções ao mesmo tempo.

O utilizador especifica o nome do formulário e os Tipos de Actos que o mesmo vai apresentar. Após a especificação do nome e Tipo de Acto, é necessário especificar os Tipos de Transacção que fazem parte desse Formulário Customizado, para efectuar tal operação necessita de clicar no botão "Adicionar Tipos de Transacção" (ver Figura 80, onde está colocado o número 1), será apresentado o *modal* da Figura 81.

De sublinhar que na Figura 81 é usado o módulo *UI Select* para a selecção dos Tipos de Transacção.

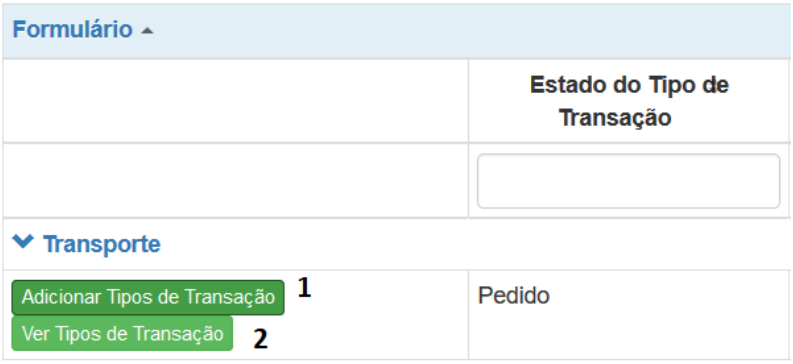

Figura 80 – Exemplo de Formulário Customizado criado.

Figura 81 – Atribuir Tipos de Transacção a um Formulário Customizado.

5.5.7. Gestão de Unidades - Tipos de Unidades

Os Tipos de Unidades (ver Figura 82) são frequentes nas propriedades que denotam uma certa unidade de medida em formato símbolo de unidade, como por exemplo kg (quilogramas), l (litros).

Figura 82 – Lista com um Tipo de Unidade existente.

5.5.8. Gestão de Actores

Na Gestão de Actores são criados os actores responsáveis por iniciar e executar as transacções. Um actor pode estar associado a vários papéis. Essa funcionalidade está representada na Figura 83.

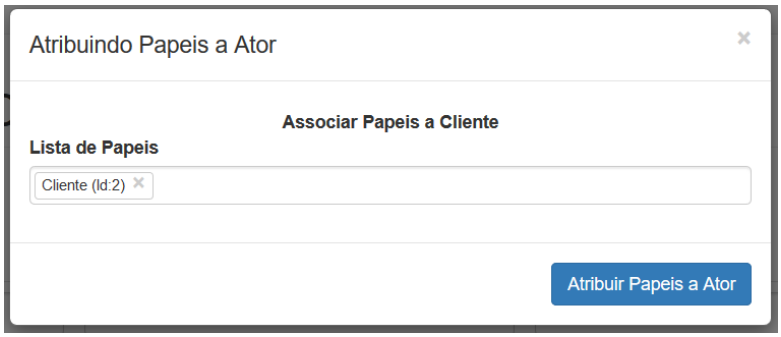

Figura 83 – Atribuição de papéis a actores.

5.5.9. Gestão de Papéis

A Gestão de Papéis (ver Figura 84) é a parte onde se associa os papéis aos actores e aos utilizadores. Um actor pode ter vários papéis e um papel pode pertencer a vários actores.

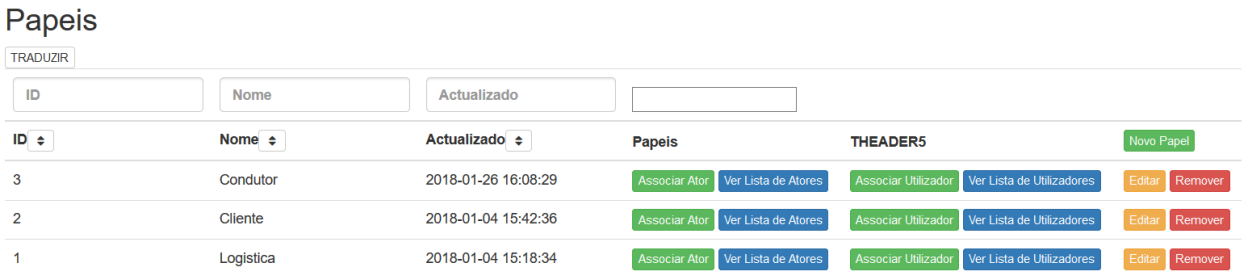

Figura 84 – Lista com três papéis e respectivos botões para atribuições.

Na Figura 85, está apresentado a forma de associar os actores das transacções do DEMO aos respectivos papéis existentes.

Figura 85 – Associação de actores a papéis.

Como observado na Figura 86, é possível atribuir múltiplos utilizadores ao mesmo papel ou até mesmo a vários papéis.

Figura 86 – Atribuir utilizadores a papéis.

5.5.10. Gestão de Idiomas

Os idiomas que uma determinada organização suporta são definidos na Gestão de Idiomas. Por exemplo, este componente controla que idioma estão disponíveis para serem utilizados nos Tipos de Processo e Tipos de Entidade.

Na Figura 87 é possível verificar como se apresenta a listagem da Gestão de Idiomas.

Figura 87 – Uma lista com um idioma.

5.5.11. Gestão de Utilizadores

Na Gestão de Utilizadores (ver Figura 88) é possível alterar dados dos utilizadores, criar novos utilizadores ou até associar utilizadores a papéis.

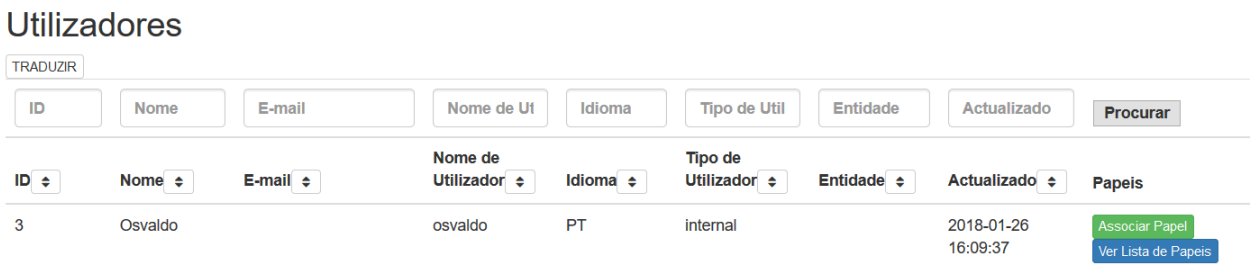

Figura 88 – Uma lista com um utilizador.

Como acontece na Gestão de Papéis também é possível associar a um determinado utilizador vários papéis, ou seja, fazer o processo inverso ao exibido previamente (ver Figura 89).

Figura 89 – Associar papéis a um utilizador.

A Figura 90 exibe que campos é preciso preencher para criar ou editar um certo utilizador que está na base de dados.

Figura 90 – Formulário de criação/edição de utilizadores.

5.6. Utilização de todos os *plugins*

A escolha do *AngularJS* proporcionou uma grande variedade de *plugins* disponíveis de forma gratuita e com suporte recente. Com esse conjunto de opções foram seleccionados alguns *plugins* para transmitir uma melhor usabilidade aos futuros utilizadores do protótipo.

Todos os *plugins* apresentados e explicados neste tópico foram utilizados pelos restantes elementos da equipa de desenvolvimento. Alguns dos *plugins* estão presentes em vários locais dos componentes do protótipo.

Os *plugins* são:

- *UI Bootstrap*
- *Angular Growl*
- *Ng-Table*
- *Angular Loading Bar*

Os seguintes *plugins* foram propostos por outros elementos da equipa de desenvolvimento:

- *Angular File Upload*
- *UI Select*
- *UI Sortable*

5.6.1. *UI Bootstrap*

O *UI Bootstrap* é um *plugin* que adiciona uma função dinâmica aos componentes presentes na biblioteca de *Bootstrap*. O *plugin* contém um conjunto de componentes que permitem a sua implementação usando os controladores e lógica do *AngularJS* [43].

Existem várias dependências que são requisitos obrigatórios para o correcto funcionamento do *UI Bootstrap*, as dependências são o *AngularJS*, *Angular-Animate*, *Angular Touch* e *Bootstrap*.

O *AngularJS Material* é no seu âmbito uma *framework* de estrutura de componente de interface baseada na implementação de especificação de *design* de *material* do Google. O módulo dispõe de um conjunto de componentes de interface com base neste *design* de material.

A título de exemplo em seguida são apresentadas as interfaces (ver Figura 91 e Figura 92) de alguns dos componentes que fazem parte do módulo *AngularJS Material*:

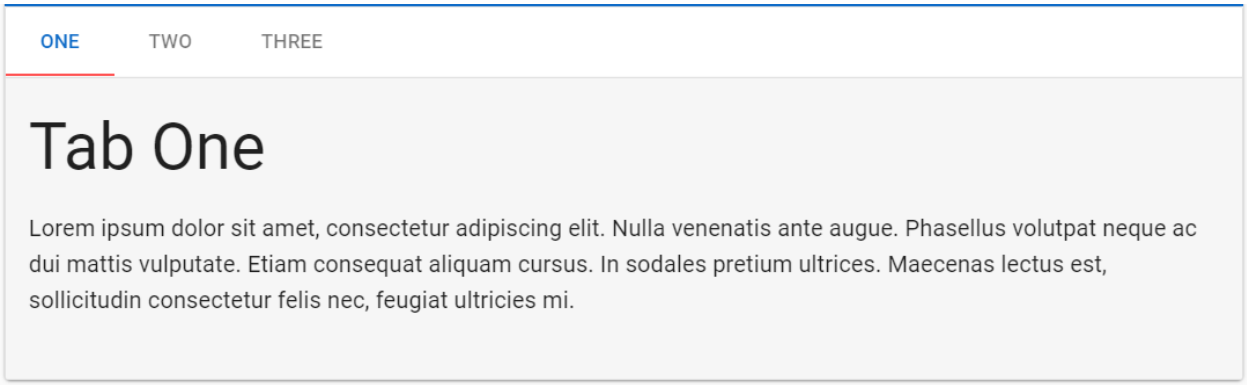

Figura 91 – Exemplo de interface das abas com o *plugin AngularJS Material.*

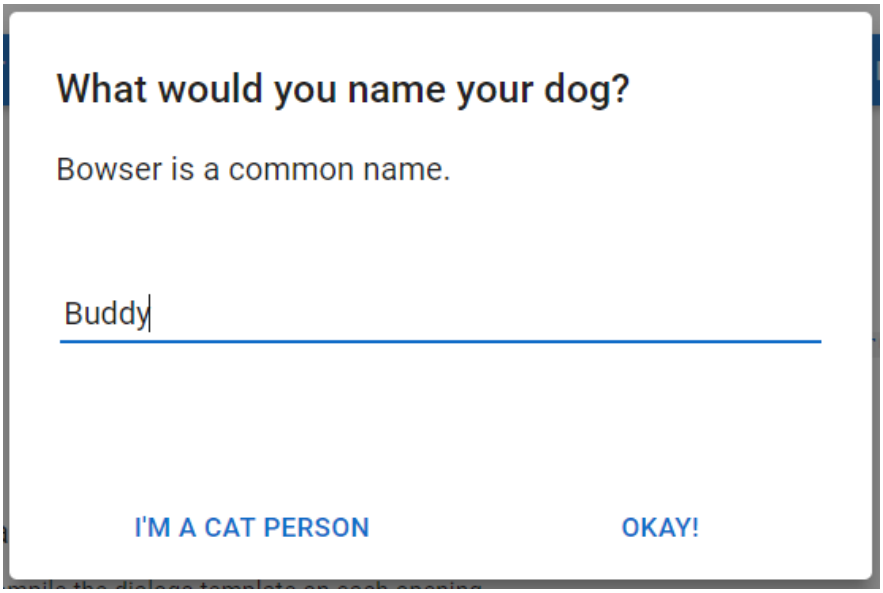

Figura 92 – Exemplo da interface do *modal* do *plugin Material.*

Tendo em conta os dois módulos (ver Figura 91 e Figura 92) apresentados nos parágrafos anteriores a escolha recaiu sobre o *UI Bootstrap* dada a sua integração com a biblioteca de *Bootstrap*.

Como especificado no capítulo 3, a interface do protótipo está desenvolvido em *Bootstrap* pelo que a escolha de um *plugin* que permitisse a interligação com o *AngularJS* era uma escolha óbvia pelas suas vantagens como por exemplo em relação a facilidade de implementação, personalização já disponível fornecida pelo *Bootstrap* e acesso a várias funções através do *AngularJS*.

Os componentes usados no protótipo estão apresentados nas imagens seguintes: [43]

- *Modal* (*ui.bootstrap.modal*): é um serviço para criar janelas em formato *modal*, ou seja é o utilizador mantém-se na mesma página mas é aberta uma subpágina por cima da página principal. Os modais tem uma interface e um controlador que inclui a lógica para os elementos que o *modal* possui. Na Figura 93 é apresentado um *modal* em que se pode constatar a página principal atrás do *modal*.

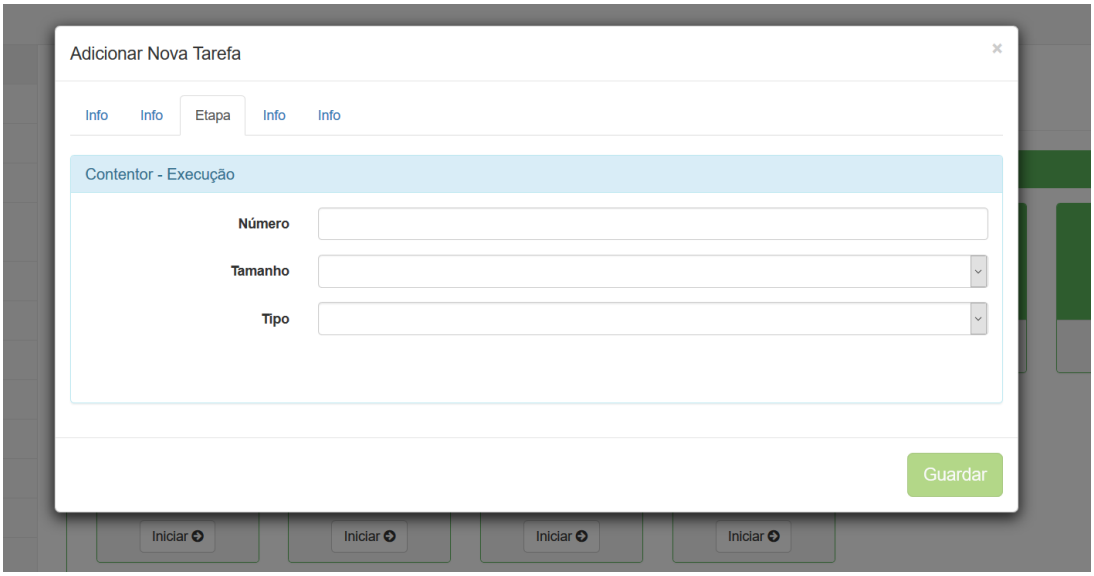

Figura 93 – Um *modal* de uma transacção na página Painel de Controlo.

- Abas (*ui.bootstrap.tabs*): as abas são um componente muito útil para as situações em que existem um conjunto de informações que se referem ao mesmo *modal* mas que são conteúdo diferenciado. Como por exemplo, no *modal* de uma transacção (ver Figura 94), cada transacção consoante o que foi especificado pelo administrador pode ter múltiplos formulários e que são preenchidos ao mesmo tempo mas que fazem parte do mesmo *modal*.

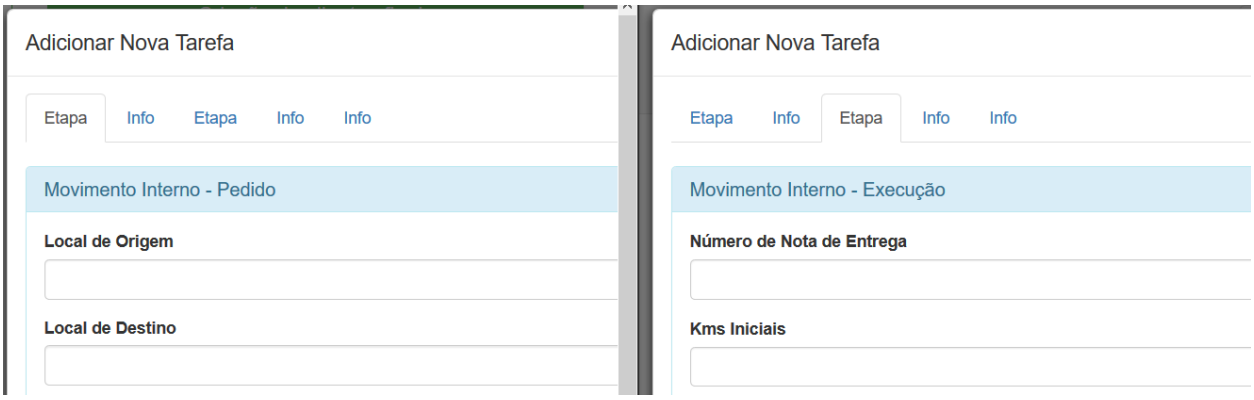

Figura 94 – Uma transacção e o mesmo *modal* com as abas em funcionamento na página Painel de Controlo

- *Accordion (ui.bootstrap.accordion)*: a directiva de *accordion* permite a exibição de uma lista de itens ou conteúdo dentro dessa zona definido para o *accordion*. O clique em cima do cabeçalho permite apresentar ou esconder o conteúdo, dependendo do estado actual do *accordion*. Na Figura 95 é apresentado um exemplo de onde é usado o componente *accordion* no formulário de uma transacção.

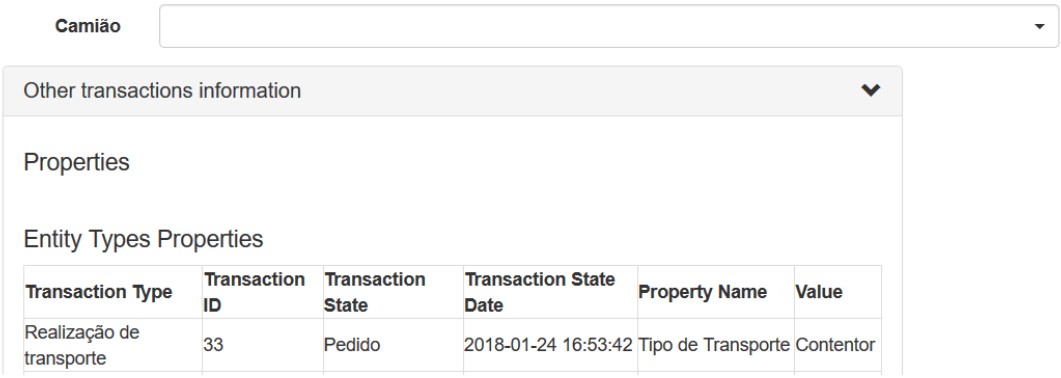

Figura 95 – Utilização do componente para mostrar informações de propriedades a outras de diferentes entidades.

5.6.2. *Angular Growl*

É um *plugin* que aproveita as interfaces estéticas dos alertas presentes na biblioteca do *Bootstrap* mas tem implementado várias opções em termos de funcionalidade, por exemplo quanto tempo deve ficar o alerta/notificação disponível até desaparecer e animações de aparecimento gradual e de desaparecer. Os alertas são executados por meio dos controladores do *AngularJS* [44].

Em qualquer acção que o utilizador final interaja com as funções do protótipo este é alertado/notificado de que as acções foram realizadas com sucesso ou não (ver Figura 96).

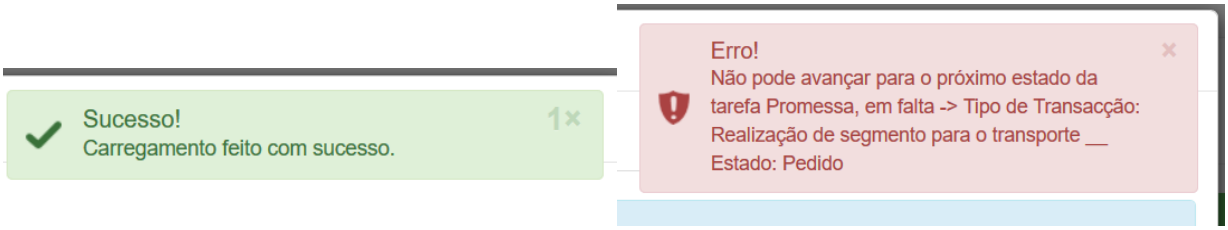

Figura 96 – Dois tipos de notificações usadas no protótipo.

5.6.3. *Ng-Table*

O *Ng-Table* [45] é um módulo, que introduz um grupo de funcionalidades a uma tabela de dados. A lógica dessas funcionalidades está desenvolvida e incorporada na biblioteca e estas estão acessíveis para serem utilizadas pelos utilizadores.

As seguintes funções do *Ng-Table* são as mais importantes:

- Paginação
- Ordenação
- Filtros de pesquisa
- Agrupar registos comuns
- Formatar tabela

O *Smart Table* [46] é um módulo, que naturalmente apresenta dados numa tabela com um grupo de funções integradas. O módulo é extremamente pequeno em termos de tamanho e não

tem dependências com outros *plugins*, pode ainda ser reduzido mais se for retirados alguns extras.

As seguintes funções do *Smart Table* são as mais importantes:

- Paginação
- Ordenação
- Filtros de pesquisa
- Formatar tabela
- *Pipe/ajax plugin*

Para o protótipo desenvolvido e segundo os requisitos obtidos, as funções essenciais são paginação, ordenação, filtros de pesquisa e agrupamento de registos comuns. A última função de agrupamento de registos é agrupar registos com uma coluna que tem os dados dessa em comum e colocar como se fosse uma categoria (ver Figura 97). Na figura 73, o primeiro rectângulo com o número 1 é a coluna que deve ser agrupada num nível, no segundo rectângulo com o número 2 são duas transacções que têm o mesmo tipo de processo e que por isso ficam debaixo desse nível, no número 3 está colocada a ordenação que é efectuada apenas nesse nível por estar activa a função de agrupar e no número 4 está o um dos campos para realizar a pesquisa.

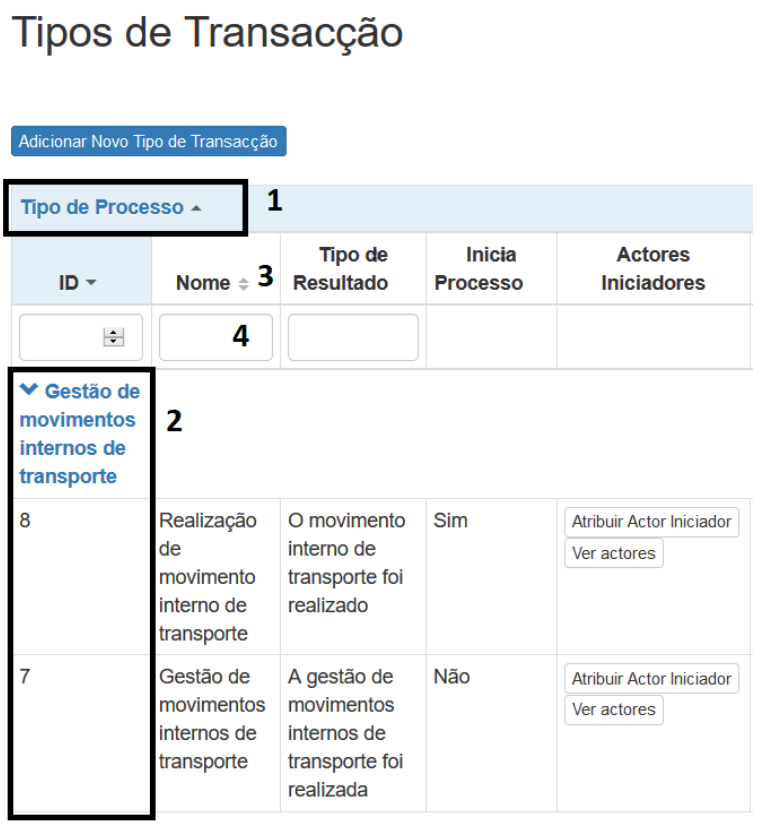

Figura 97 – Duas transacções agrupadas pelo tipo de processo com um só nível.

Apresentadas assim as funções dos dois módulos de introdução de funções numa tabela de dados, concluiu-se que o *Ng-Table* é o único dos dois *plugins* que dispõe essa funcionalidade de agrupamento. De sublinhar que somente consegue agrupar um só nível.

5.6.4. *Angular Loading Bar*

Adiciona automaticamente uma função, que alerta os utilizadores que um certo recurso está em carregamento. Este módulo adiciona essa função sempre que é efectuado um pedido de XHR (*XMLHttpRequest*) no *AngularJS*. Quando existem múltiplos pedidos ao mesmo tempo, os tempos são agrupados e cada resposta incrementa a barra de progresso com o valor apropriado. A grande vantagem do uso deste *plugin* é que apenas se inclui na aplicação e a gestão da barra de progresso é feita de forma automática. Numa barra de progressos normal é preciso controlar o estado da barra à medida que as informações são recebidas, isto é uma desvantagem quando se tem uma aplicação que faz muitos pedidos independentes em simultâneo [47].

O módulo está presente em todo o protótipo e faz a gestão da barra de progressos de todos os pedidos automaticamente, como verificado na Figura 98.

Figura 98 – À esquerda a notificação de carregamento editada de um recurso e à direita o carregamento fornecido por defeito pelo módulo.

5.6.5. *Angular File Upload*

O *plugin* é utilizado para guardar uma fila temporária de ficheiros, suporta por exemplo, funções de arrastar e soltar, envio, progresso de envio. Funciona com qualquer tecnologia de programação do lado do servidor. Quando um ficheiro é colocado na fila de temporária, vários eventos consoante a operação efectuada são despoletados automaticamente, exemplo: adicionar ou remover ficheiro [48].

O *Angular File Upload* está presente na página Painel de Controlo nas funções de "Iniciar uma Tarefa" e nos "Formulários Personalizados" caso existam propriedades do tipo ficheiro.

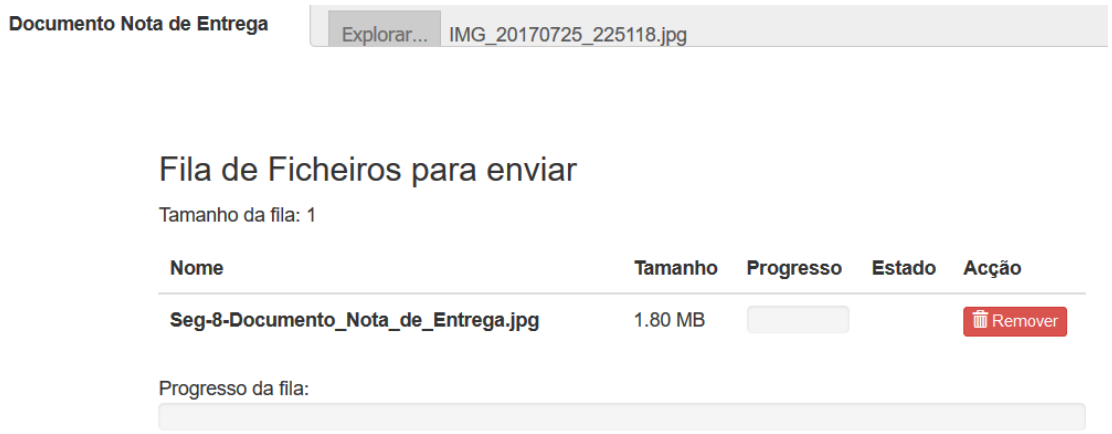

Figura 99 – Propriedade do tipo ficheiro e respectiva fila de ficheiros num formulário de uma tarefa.

5.6.6. *UI Select*

É um módulo indicado para complementar as caixas de selecção (*select box*) com funcionalidades de pesquisa automática à medida que o utilizador está a escrever a palavra ou até múltipla selecção de elementos ao mesmo tempo. É uma adaptação das bibliotecas *Select2/Selectize* que estão implementadas em *jQuery*. Este *plugin* pode ser utilizado em conjunto com o *BootStrap* e assim mantendo a estética da interface intacta [49].

No protótipo desenvolvido está presente em algumas zonas, nomeadamente:

- Painel de Controlo ao "Iniciar uma Tarefa" nos campos do formulário da mesma.
- Nas páginas Gestão de Actores, Papéis, Utilizadores, para atribuir actores nos papéis e papéis nos utilizadores.
- Na página gestão de Formulários Personalizados (atribuir os Tipos de Transacção) e Tipos de Transacção (atribuir actores iniciadores).

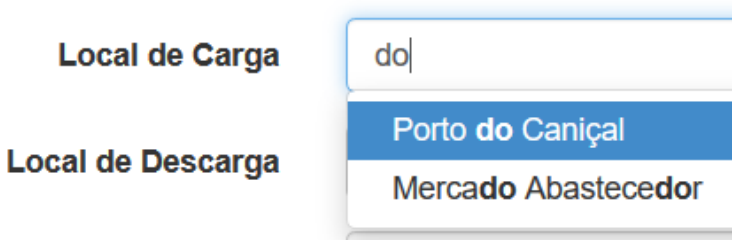

Figura 100 – Caixa de selecção com auto completar.

5.6.7. *UI Sortable*

O *plugin UI Sortable* permite reordenar uma matriz de dados com a função de arrastar e soltar. É baseado em *jQuery* mas o seu uso é destinado ao *AngularJS*, no entanto contém algumas dependências para conseguir funcionar, tais como o *jQuery* e o *jQuery UI* [50].

Nas seguintes imagens é possível verificar a localização do botão (ver Figura 101, onde está situado o número 1) para abrir o *modal* que corresponde ao reordenamento das propriedades num

Tipo de Entidade, neste caso de um Tipo de Entidade Contentor. Após o clique no botão o *modal* que é mostrado é o da Figura 102.

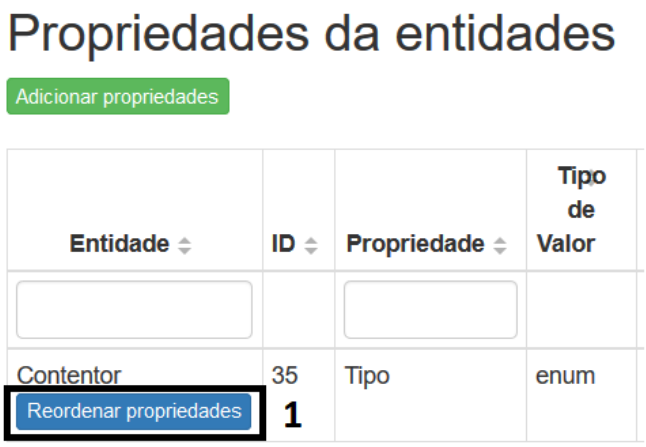

Figura 101 – Tipo de entidade com respectivo botão que encaminha para o *modal* reordenar propriedades.

Figura 102 – Várias propriedades da entidade Contentor em que é possível reordená-las com o arrastar e soltar.

O *UI Sortable* está presente nas páginas Gestão de Propriedades, nas subcategorias Entidade e Relação, para reordenar as propriedades seja de Tipos de Entidade ou de Relação.

5.7. Problemas e soluções encontradas na aplicação

Nesta secção o foco é dado aos problemas e respectivas soluções que surgiram ao longo do desenvolvimento do protótipo. Estes problemas em certos casos afectaram o protótipo na sua globalidade pelo que a análise e comentários dos restantes colegas que também desenvolveram a aplicação foi essencial para a resolução dos mesmos.

Em relação aos problemas encontrados na sua globalidade os problemas mais relevantes foram:

1. Uma lentidão no carregamento de todas as páginas devido ao elevado número de *plugins* usados.

A lentidão do protótipo a carregar os elementos todos era evidente e sendo assim houve um trabalho de análise as várias formas de optimizar a aplicação no momento de carregamento.

As seguintes optimizações foram realizadas:

- **A ordem de como os elementos são carregados**

Se os elementos colocados na secção <*head*> não estão pré-carregados, os restantes elementos das secções seguintes não serão carregados enquanto o carregamento dos elementos da secção <*head*> não estiver concluído. Isto significa que por exemplo qualquer código de *Javascript* colocado dentro desta secção conduz a um aumento no tempo de carregamento da página [51].

Pelas razões dos parágrafos anteriores a maior parte das bibliotecas de *Javascript* deve ser colocada no fim do ficheiro HTML se tal for possível, permitindo assim que a maioria do conteúdo da página (por exemplo: tabelas e imagens) seja carregada e "lida" primeiro [52].

- **Bibliotecas de código *javascript* de tamanho reduzidas**

Como no protótipo são utilizados múltiplos *plugins* de terceiros, o seu carregamento na página é maior dependendo do tamanho em termos físicos do ficheiro. Inicialmente as bibliotecas usadas eram as não-reduzidas em vez das reduzidas, em algumas situações a diferença em termos de tamanho físico de ficheiros era de significativa. Tudo isto contribuía para o problema. A solução foi a substituição de todas as bibliotecas para as de tamanho reduzido. No entanto existe uma desvantagem desta solução, o código fica não legível porque as quebras de linha, os espaços adicionais e os comentários são removidos, dado que, tudo isto no ficheiro aumenta o seu tamanho provocando uma diminuição de desempenho [51] [53].

- **Bibliotecas de código *javascript* combinadas**

Quando um *website* é aberto num navegador de *internet* são realizadas diversas conexões de forma a obter-se as imagens, as páginas HTML, determinadas bibliotecas entre outros elementos. Podem existir múltiplas conexões que são executadas no carregamento do *website*, quanto maior o número de conexões maior será o impacto no desempenho [52].

Tendo em conta o protótipo desenvolvido e os múltiplos *plugins* utilizados, o número de conexões para obter estas bibliotecas é crítico na performance. A solução para este problema é a combinação destes ficheiros múltiplos em apenas um, o que faz com que seja só efectuada uma conexão para a obtenção do ficheiro em concreto. É utilizado uma funcionalidade que o *Laravel* dispõe que é o *Laravel Mix* que permite combinar múltiplos ficheiros, tanto de *Javascript* como de CSS em apenas um. Esta funcionalidade é uma excelente solução porque não é necessário combinar os vários ficheiros num só, de forma manual [52].

2. O uso de uma biblioteca de tradução de *AngularJS* em que havia um momento instantâneo em que aparecia as variáveis da tradução.

Para a tradução dos elementos estáticos de HTML presente nas várias páginas do protótipo foi utilizado inicialmente um módulo de *AngularJS* de terceiros para efectuar essa tradução em vários idiomas.

O *plugin* escolhido foi o *Angular Translate* [54] que é uma biblioteca desenvolvida em *Javascript*, que permite a tradução dos elementos em múltiplos idiomas. Este *plugin* oferece várias funcionalidades como a especificação das traduções tanto nos próprios controladores do *AngularJS* ou mesmo em vários ficheiros de formato JSON e que depois são carregados de forma assíncrona consoante a página em questão e também a possibilidade de incorporar a tradução com outros *plugins*.

A grande vantagem deste em relação ao módulo de tradução do *Laravel* é que é possível a tradução dos elementos sem o carregamento total da página.

No entanto a utilização deste módulo acarretou diversos problemas na sua incorporação com o protótipo.

Dado o elevado número de páginas existentes foi decidida a divisão dos ficheiros de tradução de JSON para cada página. Essa divisão obrigou a utilização do carregamento assíncrono destes ficheiros para realização da tradução. O carregamento assíncrono destes ficheiros origina um problema de FUOC (*Flash of untranslated content*), que está devidamente identificado na documentação de suporte do *plugin*. Como os ficheiros de tradução são carregados de forma assíncrona, quando a página é carregada aparecem as variáveis de tradução usadas durante um período de tempo porque os ficheiros de tradução ainda não foram obtidos do servidor [55].

Este problema está exemplificado na Figura 103.

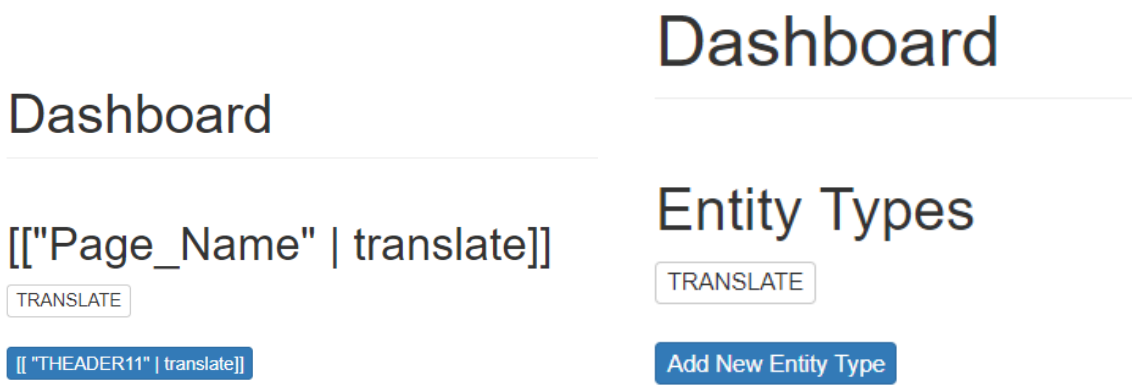

Figura 103 – Lado esquerdo apresenta o problema de FUOC e lado direito o funcionamento correcto da tradução.

Após este problema que identificamos foram realizadas várias pesquisas para encontrar uma solução, mas não existia uma alternativa viável, por esta razão foi utilizado o módulo de tradução já disponível no *Laravel*.

3. Inicialmente era utilizado o *modal* disponibilizado pela biblioteca do *BootStrap* e não era possível controlar os eventos de abrir e fechar do lado do controlador de *AngularJS*.

Uma vez que o elemento *modal* é usado em várias partes do protótipo, foi decidido inicialmente utilizar o que estava disponível pela biblioteca de *BootStrap*, mas com o contínuo

desenvolvimento do protótipo chegou-se a um ponto que era importante conseguir controlar os métodos de fechar e abrir o *modal*, como também a separação do código que fazia parte do controlador principal e da lógica que era só utilizado dentro do *modal*.

Pelas razões acima apresentadas, foi feita a alteração do *modal* de origem da biblioteca de *BootStrap* para o *modal* disponível no *plugin UI BootStrap* que está exibido na secção 5.6.1, com essa alteração é possível cada *modal* contar com um sub-controlador próprio dentro do controlador principal, separar os modelos de interface (*templates* HTML) do modelo principal, o que não acontecia com o *modal* de origem do *BootStrap*.

No Painel de Controlo surgiram os seguintes problemas principais:

1. No carregamento das abas que não apresentavam o conteúdo, devido as chamadas de *AngularJS* para obter os dados serem assíncronas e por essa razão a aba seguinte era criada mas o conteúdo da aba anterior ainda não tinha sido recebido.

As várias tarefas que um utilizador pode iniciar ou continuar são totalmente dinâmicas, como já mencionado em capítulos anteriores, em virtude desta situação a criação das abas tem de ser feita de forma dinâmica dependendo da especificação efectuada.

A grande dificuldade é que as abas utilizadas que são parte do *plugin UI BootStrap* não suportam oficialmente a geração de abas dinamicamente e isso origina um conjunto de problemas.

Os dois principais problemas são:

a) Na criação de uma aba dinâmica não é possível seleccionar essa aba como a activa pelo código de *Javascript* sem utilizar a função *Timeout.*
b) E no conteúdo das abas que é também dinâmico, o mesmo é obtido por chamadas assíncronas dado que depende do que está presente na base de dados.

Para o problema a) foi utilizada a solução de usar o *Timeout* para “activar” a aba correspondente, após a criação de uma aba dinamicamente.

No problema b) (apresentado na Figura 104) inicialmente e como recurso foi empregue também a função *Timeout*, ou seja a aba seguinte só era gerada após um determinado tempo em milissegundos, permitindo assim que o conteúdo fosse “desenhado” pelo navegador de *internet*. Como mencionado, esta solução foi conveniente como uma abordagem de recurso porque as chamadas são assíncronas e o seu tempo de resposta é variável dependendo de um conjunto de circunstâncias. A solução final é a aplicação das promessas disponibilizadas pelo *AngularJS* em conjunto com o código anteriormente criado.

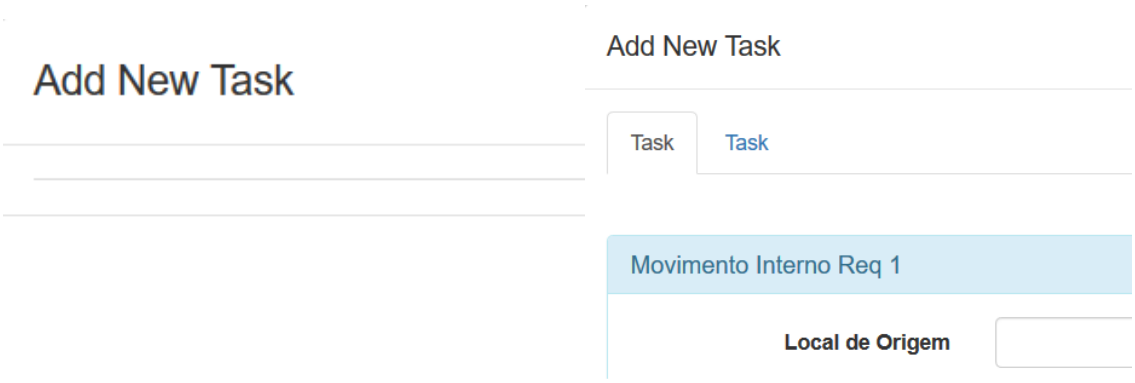

Figura 104 – Lado esquerdo apresenta o problema de carregamento e lado direito o funcionamento correcto.

2. Não era possível apresentar os dados de uma certa estrutura de dados em cada uma das abas correspondentes.

Uma tarefa (transacção) pode ter várias abas, pelo que é usado uma estrutura de dados em formato de *Array*, que recebe todos os dados correspondentes a cada uma das abas. Para aceder a esta estrutura de dados, cada aba tem de ter um dado índice.

Como todas as variáveis que são declaradas no *scope*, estão disponíveis em todas as zonas do controlador onde foram criadas, e a sua alteração de valor é repercutida em todas as partes onde é utilizada, seja no controlador de *Javascript* ou no HTML por causa do modelo de funcionamento do *AngularJS* que é a Ligação de Dados (*Two Way Data-binding*), o modelo em causa é responsável pela sincronização automática dos dados entre os elementos do modelo e da vista, a vista é uma projecção do modelo em todos os momentos, quando o modelo é alterado, a vista reflecte as mudanças e vice-versa [56].

Logo, o modelo de funcionamento de Ligação de Dados não pode ser implementado para este caso em concreto, por isso este índice de uma aba tem de ser "passado" através do método *onload* e também para atribuir os dados obtidos pelos pedidos assíncronos apenas à aba correspondente.

6. Conclusões

Relativamente aos objectivos inicialmente propostos na atribuição do projecto, estes sofreram alterações ao longo do projecto, em que a ideia inicial era de desenvolver uma aplicação de *internet* específica para esta empresa.

No entanto a modelação de processos com o recurso dos métodos e técnicas de engenharia organizacional continuou a ser uma das tarefas a ser efectuadas para a conclusão deste projecto.

6.1. Trabalho efectuado

Inicialmente os trabalhos realizados foram dedicados a recolha dos requisitos e a efectuar entrevistas para a obtenção das informações necessárias a modelação dos processos na noção fornecida pelo DEMO.

Com estes trabalhos realizados surgiu um dos maiores desafios na realização deste projecto que esteve directamente ligado com a recolha de informações para modelar os processos de negócio utilizando a linguagem DEMO, a dificuldade de expressar como funciona um certo tipo de negócio tanto da parte do mestrando como dos funcionários prejudica muito uma recolha de informações acertada para um desenho satisfatório dos diagramas.

Com os objectivos iniciais alterados e o foco num sistema de gestão de Fluxo de Trabalho foram implementados vários componentes.

Dado este foco num protótipo do tipo de Fluxo de Trabalho foram efectuadas várias pesquisas e análises das várias tecnologias existentes no mercado actualmente, de forma a cumprir as metas alcançadas com os requisitos pretendidos em mente.

As tecnologias usadas no protótipo são base de muitos outros produtos disponíveis no mercado de *software*, tendo sempre em consideração que a usabilidade foi um factor importante para a utilização do *AngularJS*, para permitir uma melhor facilidade de uso por parte de qualquer utilizador com ou sem capacidade informáticas intermédias.

Como o protótipo "comporta-se" em termos de funcionamento como um conjunto de informações e documentos "transferidos" entre diversos "actores" em várias tarefas foi realizado uma pesquisa para encontrar os pontos fortes e fracos destes sistemas e como o protótipo pode agregar estes pontos fortes e colmatar os pontos fracos dos sistemas de Fluxo de Trabalho actualmente disponíveis.

Para uma melhor compreensão das funcionalidades gerais fornecidas por estes sistemas de Fluxo de Trabalho existentes hoje em dia, a comparação de funcionalidades oferecidas entre o protótipo desenvolvido e estas plataformas relacionadas, permitiu uma melhor escolha do que o protótipo podia disponibilizar ao utilizador e de maneira a distinguir-se das restantes plataformas relacionadas.

O sistema de gestão tem um potencial enorme em termos de flexibilidade, uma vez que é totalmente dinâmico e o mesmo adapta-se aos processos de negócio da organização.

Conclui-se que a implementação destes componentes permite um uso básico do sistema em versão protótipo, no entanto não foi possível desenvolver todas funcionalidades e aplicar todos os requisitos obtidos, devido a dimensão e o desafio que este projecto em específico engloba.

6.2. Trabalho futuro

O trabalho futuro deve passar pela introdução do padrão de transacção completo e não apenas os cinco tipos de actos principais do DEMO. Em termos de base de dados, a introdução de base de dados não relacionais tem de ser uma alternativa a ser considerada, conhecido o elevado número de relações entre as tabelas, provavelmente até uma solução híbrida. A interligação de processos é outro ponto fundamental a ser implementado.

A conversão da gestão de propriedades com uma funcionalidade de arrastar e soltar, eliminando o tradicional formulário e tornando o componente mais dinâmico para o utilizador.

A introdução do histórico para haver versões das especificações realizadas e das instâncias criadas.

A parte da segurança é algo a ter em conta em relação ao acesso indevido de ficheiros utilizando as funcionalidades que o *Laravel* tem disponíveis.

Um dos contributos foi a identificação de um conjunto de desafios em aberto presentes na modelação e análise dos processos de negócio da empresa, estando por isso previstas reuniões para solucionar estes desafios.

Outra parte extremamente importante e que à data ainda não foi efectuada é a realização de vários tipos de testes principalmente no âmbito dos não funcionais ligados a usabilidade do protótipo.

Pelo facto de já terem sido obtidos vários contributos em termos de modelação e implementação ao longo do projecto e também porque o protótipo está ainda num estado inicial, não foi possível a realização destes vários tipos de testes, esperando que futuramente estes possam ser realizados juntamente com as partes interessadas.

Existe um longo caminho a percorrer em termos de desenvolvimento até que a aplicação possa ser totalmente utilizada por uma organização.

7. Referências

[1] J. L. G. Dietz, *Enterprise ontology: theory and methodology*. Berlin ; New York: Springer, 2006.

[2] D. Aveiro, J. Tribolet, e D. Gouveia, *Advances in Enterprise Engineering VIII*. .

[3] J. de Jong, «Designing the Information Organization from Ontological Perspective», em *Advances in Enterprise Engineering V*, 2011, pp. 1–15.

[4] M. R. Krouwel e M. Op ’t Land, «Combining DEMO and Normalized Systems for Developing Agile Enterprise Information Systems», em *Advances in Enterprise Engineering V*, 2011, pp. 31–45.

[5] Y. Yinlan, «The research of management system based on workflow», em *Computer Science & Education (ICCSE), 2012 7th International Conference on*, 2012, pp. 842–846.

[6] D. Georgakopoulos, M. Hornick, e A. Sheth, «An overview of workflow management: From process modeling to workflow automation infrastructure», *Distrib. Parallel Databases*, vol. 3, n. 2, pp. 119–153, 1995.

[7] A. DiCaterino, K. Larsen, M.-H. Tang, e W.-L. Wang, «An introduction to workflow management systems», STATE UNIV OF NEW YORK AT ALBANY, 1997.

[8] «Mendix 7 Harnesses Open Platform Ecosystem to Ignite Digital Innovation and Enable Rapid Delivery of Smart Apps», *Mendix*, 08-Jun-2016. [Em linha]. Disponível em: https://www.mendix.com/press/mendix-7/. [Acedido: 20-Jan-2018].

[9] «Microflows - Mendix 7 Reference Guide», *Mendix Documentation*. [Em linha]. Disponível em: https://docs.mendix.com/refguide/microflows. [Acedido: 20-Jan-2018].

[10] «Offline - Mendix 7 Reference Guide», *Mendix Documentation*. [Em linha]. Disponível em: https://docs.mendix.com/refguide/offline. [Acedido: 02-Abr-2018].

[11] «The “Many and Primary” Design Pattern», *Mendix and Beyond*, 24-Jul-2012. .

[12] «Nanoflows - Mendix 7 Reference Guide», *Mendix Documentation*. [Em linha]. Disponível em: https://docs.mendix.com/refguide/nanoflows. [Acedido: 02-Abr-2018].

[13] «Mobilize Every Process». [Em linha]. Disponível em: https://www.appgyver.com/use-cases/workflow/. [Acedido: 24-Jan-2018].

[14] «Powerful Form Builder». [Em linha]. Disponível em: https://www.appgyver.com/use-cases/smart-forms/. [Acedido: 14-Jan-2018].

[15] «IBM Knowledge Center - IBM Forms Designer». [Em linha]. Disponível em: https://www.ibm.com/support/knowledgecenter/en/SSS28S_8.2.1/Designer/lfd_overview.html. [Acedido: 17-Jan-2018].

[16] «IBM Knowledge Center - Converting to IBM forms». [Em linha]. Disponível em: https://www.ibm.com/support/knowledgecenter/en/SSS28S_8.2.0/Designer/lfd_converting_pdf_files.html. [Acedido: 17-Jan-2018].

[17] «What is a RESTful API?», *MuleSoft*, 02-Out-2017. [Em linha]. Disponível em: https://www.mulesoft.com/resources/api/restful-api. [Acedido: 22-Jan-2018].

[18] «What is Server-side Scripting?» [Em linha]. Disponível em: https://www.computerhope.com/jargon/s/server-side-scripting.htm. [Acedido: 20-Mai-2018].

[19] «Advantages of using the Laravel framework». [Em linha]. Disponível em: https://www.etatvasoft.com/blog/advantages-of-using-the-laravel-framework/. [Acedido: 27-Jan-2018].

[20] «Broadcasting». [Em linha]. Disponível em: https://laravel.com/docs/5.5/broadcasting. [Acedido: 30-Jan-2018].
[21] «Eloquent: Getting Started». [Em linha]. Disponível em: https://laravel.com/docs/5.5/eloquent. [Acedido: 02-Fev-2018].
[22] «CRUD», *Wikipédia, a enciclopédia livre*. 28-Set-2017.
[23] «Eloquent: Relationships». [Em linha]. Disponível em: https://laravel.com/docs/5.5/eloquent-relationships. [Acedido: 01-Fev-2018].
[24] «Localization». [Em linha]. Disponível em: https://laravel.com/docs/5.5/localization. [Acedido: 26-Jan-2018].
[25] «JSON», *Wikipédia, a enciclopédia livre*. 17-Jan-2018.
[26] «MVC», *Wikipédia, a enciclopédia livre*. 22-Out-2017.
[27] «Differences between Client-side and Server-side Scripting». [Em linha]. Disponível em: https://www.sqa.org.uk/e-learning/ClientSide01CD/page_18.htm. [Acedido: 20-Mai-2018].
[28] «AngularJS advantages and limitations | SoftElegance's Blog». .
[29] «Advantages and disadvantages of AngularJS | Software Developer India». [Em linha]. Disponível em: http://www.software-developer-india.com/advantages-and-disadvantages-of-angularjs/. [Acedido: 19-Jan-2018].
[30] D. Lamb, «jQuery vs. AngularJS: A Comparison and Migration Walkthrough». [Em linha]. Disponível em: https://www.airpair.com/angularjs/posts/jquery-angularjs-comparison-migration-walkthrough. [Acedido: 01-Fev-2018].
[31] tutorialspoint.com, «AngularJS Dependency Injection», *www.tutorialspoint.com*. [Em linha]. Disponível em: https://www.tutorialspoint.com/angularjs/angularjs_dependency_injection.htm. [Acedido: 10-Fev-2018].
[32] H. Chouhan, «6 Reasons to Choose the Bootstrap CSS Framework», *OSTraining*. [Em linha]. Disponível em: https://www.ostraining.com/blog/coding/bootstrap/. [Acedido: 04-Fev-2018].
[33] N. Patel, «Bootstrap 4 Release Date & It's New Features in Detail», *WordPress, HTML & Web Development Best Practices*, 17-Nov-2017. .
[34] «Database», *Wikipedia*. 25-Mai-2018.
[35] M. Ubl, E. K. P. October 20th, e 2010 Comments: 3 Your browser may not support the functionality in this article, «Introducing WebSockets: Bringing Sockets to the Web - HTML5 Rocks», *HTML5 Rocks - A resource for open web HTML5 developers*. [Em linha]. Disponível em: https://www.html5rocks.com/en/tutorials/websockets/basics/. [Acedido: 20-Mai-2018].
[36] «WebSockets», *MDN Web Docs*. [Em linha]. Disponível em: https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API. [Acedido: 30-Jan-2018].
[37] «Understanding Pusher». [Em linha]. Disponível em: https://pusher.com/docs/. [Acedido: 05-Fev-2018].
[38] «OneSignal - Multi-platform Push Notification Service». [Em linha]. Disponível em: https://onesignal.com/. [Acedido: 05-Fev-2018].
[39] David Aveiro, Duarte Pinto, e Magno Andrade, «DEMO Based Dynamic Information System Modeller and Executer».
[40] «Authentication». [Em linha]. Disponível em: https://laravel.com/docs/5.5/authentication#. [Acedido: 04-Fev-2018].

[41] T. Tkalec, «JSON Web Token Tutorial: An Example in Laravel and AngularJS», *Toptal Engineering Blog*. [Em linha]. Disponível em: https://www.toptal.com/web/cookie-free-authentication-with-json-web-tokens-an-example-in-laravel-and-angularjs. [Acedido: 30-Jan-2018].
[42] «Authentication Protecting Routes». [Em linha]. Disponível em: https://laravel.com/docs/5.5/authentication#protecting-routes. [Acedido: 01-Fev-2018].
[43] «UI Bootstrap». [Em linha]. Disponível em: https://angular-ui.github.io/bootstrap/#!#getting_started. [Acedido: 07-Fev-2018].
[44] J. Stevens, *angular-growl-2: growl-like notifications for angularJS projects*. 2018.
[45] «ngTable Examples». [Em linha]. Disponível em: http://ng-table.com/#/. [Acedido: 03-Fev-2018].
[46] «Smart Table documentation». [Em linha]. Disponível em: http://lorenzofox3.github.io/smart-table-website/#top. [Acedido: 02-Fev-2018].
[47] «Angular Loading Bar». [Em linha]. Disponível em: https://chieffancypants.github.io/angular-loading-bar/. [Acedido: 01-Fev-2018].
[48] Alex, *angular-file-upload: Angular File Upload is a module for the AngularJS framework*. 2018.
[49] «AngularJS ui-select». [Em linha]. Disponível em: https://angular-ui.github.io/ui-select/. [Acedido: 05-Fev-2018].
[50] *ui-sortable: jQuery UI Sortable for AngularJS*. AngularUI, 2018.
[51] «11 Tips to Optimize JavaScript and Improve Website Loading Speeds», *Hiring | Upwork*, 13-Dez-2016. [Em linha]. Disponível em: https://www.upwork.com/hiring/development/11-tips-to-optimize-javascript-and-improve-website-loading-speeds/. [Acedido: 01-Mar-2018].
[52] «Speed Up Your Javascript Load Time – BetterExplained». [Em linha]. Disponível em: https://betterexplained.com/articles/speed-up-your-javascript-load-time/. [Acedido: 01-Mar-2018].
[53] «9 Tips to Reduce Page load Time and Speed Up your Website», *Truconversion*, 22-Mar-2016. [Em linha]. Disponível em: https://www.truconversion.com/blog/conversion-rate-optimization/9-tips-to-reduce-page-load-time-and-improve-website-speed/. [Acedido: 01-Mar-2018].
[54] *angular-translate: Translating your AngularJS 1.x apps*. angular-translate, 2018.
[55] «Doc: Asynchronous Loading». [Em linha]. Disponível em: https://angular-translate.github.io/docs/#/guide/12_asynchronous-loading. [Acedido: 25-Fev-2018].
[56] «AngularJS: Developer Guide: Data Binding». [Em linha]. Disponível em: https://docs.angularjs.org/guide/databinding. [Acedido: 04-Mar-2018].